NTNU

Norwegian University of
Science and Technology

Creating and Testing Continuous Acquisition Software with GPS Timestamps

Eirik Ferdinand Sandberg

Master of Science in Mechanical Engineering
Submission date: February 2017
Supervisor: Ole Andre Øiseth, KT

Norwegian University of Science and Technology
Department of Structural Engineering

Summary

The main objective of this thesis has been to write software in LabVIEW enabling GPS time synchronization between several Compact RIOs when performing continuous acquisition of acceleration data. By providing GPS timestamps to the individual acceleration samples, unlimited compact RIOs can be set up to measure large structures without any need for communication. The measurements would be synchronized in the post processing of the data, removing the complicated process of communication and synchronization during measurements.

Several additional functions are incorporated in the software. This includes signal filtering, downsampling, a trigger condition for writing to file, and a logical user interface. As requested from the faculty, a thorough explanation of the software's functionality and the reasoning behind them has been made, and is included in this thesis.

To confirm the functionality of the program, several tests have been performed on the various elements of the software. Some of the most important are presented here. As a final trial, a field test of the program was executed on a foot bridge in Trondheim to ensure the program would work in a real life situation. This process is also presented in this thesis, together with a modal analysis of the bridge.

Results from all the tests of the software are deemed successful. The software is working as intended, and is highly reliable.

Sammendrag

Hovedmålet med denne oppgaven har vært å skrive et program i LabVIEW som tillater synkronisering med GPS-tid mellom flere Compact RIOer under kontinuerlige målinger av akselerasjonsdata. Ved å gi hver enkel akselerasjonsmåling et tidsstempel i GPS-tid, kan et ubegrenset antall Compact RIOer settes opp for å måle store strukturer uten behovet for kommunikasjon. Målingene blir synkronisert i post-prosesseringen, noe som fjerner behovet for den kompliserte prosessen med kommunikasjon og synkronisering underveis.

I tillegg er flere andre funksjoner innarbeidet i programmet. Det inkluderer signalfiltrering, downsampling, et kriteria for å skrive til fil og et oversiktlig brukergrensesnitt. Som forespurt fra fakultet er en grundig forklaring av funksjonaliteten og begrunnelsen bak denne blitt gjort, og er inkludert her.

For å bekrefte funksjonaliteten til programmet er flere tester designet og gjennomført på de forskjellige elementene. Noen av de viktigste er presentert her. Som en siste test er en felttest av programmet blitt gjennomført på en gangbro i Trondheim. Dette for å forsikre om at programmet fungerer i en praktisk situasjon. Denne prosessen er inkludert i denne oppgaven sammen med en modalanalyse av broen.

Resultatene fra alle testene er sett på som positive. Programmet fungerer som det skal og virker å være meget pålitelig.

Preface

This master thesis is the end of the 5-year study programme Master of Science in Mechanical Engineering. It is carried out at the Department of Structural Engineering, under the Faculty of Engineering Science and Technology at the Norwegian University of Science and Technology in Trondheim. The work presented in this thesis is a result of 20 weeks work mainly during the fall semester of 2016, corresponding to 30 credits.

I am very grateful for the support and guidance from my supervisor Assosiate Professor Ole Øiseth, at the Department of Structural Engineering, NTNU.

I would also like to thank PhD-candidates Knut Andreas Kvåle and Gunnstein Frøseth for help during the process. And again Knut Andreas Kvåle and two master thesis students for great help during the field test at Bratsberg foot bridge.

And lastly to Bjørn Strickert Schølberg and the rest of the Lab Engineers at the Department of Structural Engineering for technical help in creating the necessary equipment.

Table of Contents

List of Figures

Abbreviations

MDOF	=	Multi-degree-of-freedom
ZOH	=	Zero order Hold
OMA	=	Operational modal analysis
SSI	=	Stochastic Subspace Identification
Cov	=	Covariance
FFT	=	Fast Fourier Transform
PSD	=	Power Spectral Density
FRF	=	Frequency Respons Function
SVD	=	Singular Value Decomposistion
NI	=	National Instruments
cRIO	=	Compact Rio
FPGA	=	Field Programmable Gate Array
I/O	=	inn/out
PPS	=	Pulse-per-second
VI	=	Virtual Instrument
DMA	=	Direct Memory Access
RMS	=	Root Mean Square

Chapter 1

Introduction

1.1 Problem Description

For this master thesis the Department of Structural Engineering at NTNU expressed a need for software capable of continuous acquisition of acceleration data. Because of previous problems with reliability in data synchronization this program was to feature synchronization via GPS timestamps. The motive of providing GPS timestamps to the individual acceleration samples, is that with it unlimited compact RIOs can be set up to measure large structures without need any for communication. The software was to be written in the graphical programming language LabVIEW and to be used on the Compact RIO platform, both produced by National Instruments. This is the main product of this thesis.

To prove functionality and reliability of the written software several tests have been designed and preformed. In addition, a field test of the program was executed on a foot bridge in Trondheim. This to make sure the program would work in a real life application and to enable a small modal analysis of the structure.

1.2 Structure of Paper

Chapter 2. Theory

This chapter will provide the necessary theory. This includes background to understand modal analysis and signal processing. The reader may use this to understand the theoretical foundation of this thesis, and the references included for further reading.

Chapter 3. Programming in Labview

This chapter will briefly go through the hardware used in this project, before explaining in depth the different functions and methods used in the software and the motivation behind them.

Chapter 4. Testing the Program

Here the different tests designed and executed for the software will be documented and explained.

Chapter 5. Modal Analysis of Bratsberg Foot Bridge

This chapter will present the entire modal analysis of the Bratsberg foot bridge. The chapter will begin with the experiment setup, move on to the actual measuring, present the results and comment on the accuracy of these.

Chapter 6. Final Remarks

The successfulness of the written software will be commented on, and some thoughts on further work shared.

Chapter 2

Theory

This chapter will focus on the theory needed to understand the mechanisms of signal processing, structural dynamics and modal analysis. It will begin with the equation of motion and solving the eigenvalue problem, then some methods for simple frequency analysis will be presented, before a walk through the operational modal analysis method used in this thesis. Lastly some signal processing terms and principles will be explained. A understanding of calculus and basic statistics is expected from the reader.

2.1 The Eigenvalue Problem

The eigenvalue problem is a much encountered problem in mathematics. In structural dynamics it is the issue of solving the equation of motion for different dampening models. This section will go through the different solutions with the different simplifications and assumptions of the dampening matrix.

2.1.1 Equation of Motion

The dynamic behavior of a structure with several degrees of freedom, or a multi-degree-of-freedom (MDOF) system, can be described with the equation of motion. Normally given in the time domain on the form:

$$[M]\ddot{u}(t) + [C]\dot{u}(t) + [K]u(t) = f(t) \tag{2.1}$$

Where $[M]$, $[C]$ and $[K]$ are the matrix of mass, dampening and stiffness respectively, and $\ddot{u}(t)$, $\dot{u}(t)$ and $u(t)$ is the acceleration, velocity and displacement respectively. Lastly, $f(t)$ represents the force vector. This system with matrices and vectors of the nth order, represents a system with n degrees of freedom. To solve this equation a solution can be

assumed on the form of:

$$u(t) = pe^{\lambda t} \tag{2.2}$$

Derivatives of this equation gives velocity and displacements equal to:

$$\dot{u}(t) = \lambda pe^{\lambda t} \tag{2.3}$$

$$\ddot{u}(t) = \lambda^2 pe^{\lambda t} \tag{2.4}$$

Inserting equantion (2.2), (2.3) and (2.4) into the equation of motion, and assuming force vector equal to zero, yields:

$$(\lambda^2[M] + \lambda[C] + [K])[\Phi] = 0 \tag{2.5}$$

Solving this equation for λ is done with different methods depending on the dampening model of the system. Most large constructions are underdamped meaning the solution will be complex conjugated couples in the form of:

$$\lambda_j = -\omega_n\zeta \pm i\omega_n\sqrt{1-\zeta^2} \tag{2.6}$$

Here ω_n and ζ is the natural frequency and modal dampening respectively, both belonging to the nth mode. From (2.6) we can extract these two quantities and also the damped natural frequency, ω_d,:

$$\omega_n = |\lambda| \tag{2.7}$$

$$\zeta = -\frac{Re(\lambda)}{|\lambda|} \tag{2.8}$$

$$\omega_d = Im(\lambda) \tag{2.9}$$

In order to calculate the mode shapes, a common assumption is zero dampening. This is obviously an unrealistic approximation, as the free response of the structure will not decay over time. With this approximation the equation of motion is reduced to:

$$[M]\ddot{u}(t) + [K]u(t) = 0 \tag{2.10}$$

And, using equation (2.7), this leads to the linear eigenvalue problem:

$$([K] - \omega_n^2[M])\Phi_n = 0 \tag{2.11}$$

Where Φ_n is the eigenvector with the eigenvalue ω_n. This equation has a solution if the determinant of the combined matrix inside the parenthesis is zero. Since the matrices $[K]$ and $[M]$ are symmetric, real and positive definite matrices of size $n \times n$, the determinant will be a nth order polynomial with n solutions which can be solved. Each solution being a real eigenvector, Φ_n, corresponding to a natural frequency, ω_n. All these eigenvectors are mutually orthogonal.

This is also know as a linear eigenvalue problem. However, the zero dampening assumption is not always applicable, and other more complicated solutions must be found.

2.1.2 Classical Dampning

One way to make the equation of motion into a linear eigenvalue problem without having zero dampening is with classical dampening. This assumption is based on the modal matrix $[\Phi]$, which is an column-wise merging of the neigen vectors of the system, defined as: $[\Phi] = [\Phi_1, \Phi_2, ...\Phi_n]$. This matrix has the property of diagonalizing the mass and stiffness matrices. This is done by by pre- multiplying by the modal matrix's transpose and post multiplying by the modal matrix itself. This will acquire the modal mass and modal stiffness matrix as defined by:

$$[\tilde{M}] = [\Phi]^t[M][\Phi] \tag{2.12}$$

$$[\tilde{K}] = [\Phi]^t[K][\Phi] \tag{2.13}$$

These are both diagonal matrices, meaning they only contain non-zero entries on the diagonal. If in the same way the dampening matrix is diagonalizeable by $[\Phi]$ it falls under the classical dampening definition [1]. Another way of describing this is through a formulation by Coughey and Kelly[2]. So, if the dampening matrix satisfies the identity

$$[C][M]^{-1}[K] = [K][M]^{-1}[C] \tag{2.14}$$

it falls under the classical dampening definition. This definition also indirectly includes

the fact that the dampening matrix now is directly dependent on the mass and stiffness matrices.

To create a dampening matrix directly dependent on the mass and stiffness matrices there are several ways. One popular and simple way is with the Rayleigh dampening definition. This defines the dampening matrix, $[C]$, as linearly proportional to $[M]$ and $[K]$ such that:

$$[C] = \alpha[M] + \beta[K] \tag{2.15}$$

Where α and β are constants.

2.1.3 The quadratic eigenvalue problem

When the classical dampening approximation is insufficient, and a dampening matrix dependent on the mass and stiffness matrices is not accurate enough, more general solutions are used. This leads to the need of solving the quadratic eigenvalue problem. This is displayed in equation (2.16).

$$([K] + \omega_n[C] + \omega_n^2[M])\Phi_n = 0 \tag{2.16}$$

Here both ω_n and ω_n^2 are present, which makes it a mathematical problem much harder to solve. The most common way of solving it, is with a state-space formulation.

State Space Model

The state-space model's only assumptions is that the mass, dampening and stiffness matrices are symmetric, positive definite and of dimension $N \times N$. The state-space formulation may be used to transform a second order differential equation, into two differential equations of the first order. The derivations of this state space model are in large parts adapted from Rainieri and Fabbrocino [2].

Defining:

$$y(t) = \begin{bmatrix} u(t) \\ \dot{u}(t) \end{bmatrix} \tag{2.17}$$

and its derivative:

$$\dot{y}(t) = \begin{bmatrix} \dot{u}(t) \\ \ddot{u}(t) \end{bmatrix} \tag{2.18}$$

From the equation of motion in (2.1) we pre-multiply with $[M]^{-1}$ and introduce (2.17) and (2.18). We also factorize $f(t)$ so:

$$f(t) = [B]p(t) \tag{2.19}$$

where $[B]$ is the position matrix describing where the forces are applied, and $p(t)$ is the forces dependence on time. Rearranged, this can be shown as:

$$\ddot{u} = \Big[-[M]^{-1}[K] \quad -[M]^{-1}[C] \Big] y(t) \; + \; [M]^{-1}[B]p(t) \tag{2.20}$$

$\dot{u}(t)$ can be written as:

$$\dot{u}(t) = \Big[[0] \quad [I] \Big] y(t) \tag{2.21}$$

Where $[I]$ is the identity matrix. From this it can be shown that:

$$\dot{y}(t) = [A]y(t) \; + \; [\bar{B}]p(t) \tag{2.22}$$

Where:

$$[A] = \begin{bmatrix} [0] & [I] \\ -[M]^{-1}[K] & -[M]^{-1}[C] \end{bmatrix}; \qquad [B] = \begin{bmatrix} [0] \\ [M]^{-1}[\bar{B}] \end{bmatrix} \tag{2.23}$$

In structural dynamics equation (2.22) is often referred to as the state equation. One of the two differential equations found by the state space formulation. The other is the observation equation which describes the observable response of the system. If the system is measured at l locations the observation equation can be found through defining:

$$u_l(t) = [C_a]\ddot{u}(t) + [C_v]\dot{u}(t) + [C_d]u(t) \tag{2.24}$$

Where $[C_a]$,$[C_v]$ and $[C_d]$ are the location matrices of acceleration, velocity and displacement respectively. Inserting equation (2.17) and (2.20), and rewriting we get:

$$u_l(t) = \Big[[C_d] - [C_a][M]^{-1}[K] \quad [C_v] - [C_a][M]^{-1}[C] \Big] y(t) + [C_a][M]^{-1}[\bar{B}]p(t) \tag{2.25}$$

Defining the *output inuence matrix*, $[C_c]$, and the *direct transmission matrix*, $[D_c]$, as[2]:

$$[C_c] = \begin{bmatrix} [C_d] - [C_a][M]^{-1}[K] & [C_v] - [C_a][M]^{-1}[C] \end{bmatrix} \tag{2.26}$$

$$[D_c] = [C_a][M]^{-1}[\bar{B}] \tag{2.27}$$

Gives us:

$$u_t(t) = [C_c]y(t) + [D_c]p(t) \tag{2.28}$$

Which is the preferred form of the observation equation. The state space equation given in (2.22), and this version of the observation equation is the basis for solving the quadratic eigenvalue problem.

Discrete State Space Model

In most computational uses the signal will be discrete, and so there is a need for a discrete version of a state space model. With a sampling period of Δt, the discrete time instant t_k becomes $t_k = k\Delta t$. For this model to be correct we need to make an assumption about the behaviour of time dependent variables between the samples. The Zero Order Hold is a simple assumptions that does this. It claims the input between samples are piecewise constant. A discrete version of the State equation given in equation (2.22), and the observation equation, given in equation (2.28), yields respectively:

$$y_{k+1} = [A_j]y_k \ + \ [B_j]p_k \tag{2.29}$$

$$u_k = [C_j]y_k + [D_j]p_k \tag{2.30}$$

Where $[A_j]$, the *discrete state matrix*, $[B_j]$, the *discrete input influence matrix*, $[C_j]$, the *discrete output influence matrix* and $[D_j]$, the *discrete direct transmission matrix*, are defined as:

$$[A_j] = e^{[A]\Delta t} \tag{2.31}$$

$$[B_j] = ([A_j] - [I])[A]^{-1}[B] \tag{2.32}$$

$$[C_j] = [C_c] \tag{2.33}$$

$$[D_j] = [D_c] \tag{2.34}$$

This is currently a deterministic discrete state space model. However, for real life application, there is a need to introduce a stochastic contribution to account for all unknown force input and model inaccuracies. So, to equation (2.29) the variable w_k is added, representing process noise, and to equation (2.30) v_k is added, representing sensor noise. These equations thus yielding:

$$y_{k+1} = [A_j]y_k \ + \ [B_j]p_k + w_k \tag{2.35}$$

$$u_k = [C_j]y_k + [D_j]p_k + v_k \tag{2.36}$$

This model contains both known and unknown(stochastic) input. But, in the case of operational modal analysis, as will be explained in a later section, there is no known input. This leads to a model where all excitation is caused by unknown forces. We remove the known input contributions from (2.35) and (2.36), and are left with:

$$y_{k+1} = [A_j]y_k \ + w_k \tag{2.37}$$

$$u_k = [C_j]y_k + v_k \tag{2.38}$$

This model is the basis for a group of techniques called Stochastic Subspace Identification (SSI), which is what will be used for modal analysis later in this thesis.

2.2 Frequency Domain

This section will explain the transformation from time to frequency domain, and go through some of the most frequently used frequency analysis tools.

2.2.1 Fourier Transform

The Fourier transform is a technique based on the insight that any continuous signal which varies in time can be expressed as the sum of an infinite number of harmonic signals. This transformation into harmonic signals will result in a complex valued function of frequency, where the real part is the frequency's amplitude and the imaginary part is the phase angle. This way, by transforming a signal into what is called the frequency domain, it is easy to see what frequencies the original signal consist of.

The transform has many normal notations, here, the notation used will be [22]:

$$U(f) = \int_{-\infty}^{\infty} u(t)e^{-2\pi ift}dt \tag{2.39}$$

The transformation is linear and can be inversed. The inverse thus being:

$$u(t) = \int_{-\infty}^{\infty} U(f)e^{2\pi ift}df \tag{2.40}$$

This definition is however only for continuous signals of infinite length. For practical applications, like the ones used in this thesis, the signal will be discrete and of finite length. This calls for a discrete version of the Fourier transform, which can be defined as[4]:

$$U_k = \sum_{n=0}^{N-1} u_n e^{-2\pi ink/N} \tag{2.41}$$

Where U_k is the frequency content at frequency k, u_n is the nth time domain sample, and N is total number of samples in time domain.

In many computational applications it's normal to use a discrete Fourier transform algorithm called the Fast Fourier Transform (FFT). This algorithm optimizes the process by reducing the discrete Fourier matrix into several sparse matrices, reducing the computational time greatly[7].

2.2.2 Power Spectral Density

The power spectral density (PSD) contains the properties of the frequency domain and is the frequency domain equivalent of variance[6]. It can be calculated simply by squaring the absolute value of the Fourier transform of a signal, but now, instead of displaying Fourier amplitude, it displays the power of the signal. Similarly to the simple Fourier transform, its normal use is to inspect the plot visually as an early indication of where the big frequency contributions are. In structural dynamics, this can be used a early indication of what the natural frequencies of a measured structure may be. In more statistical terms the power spectral density can be found by defining $y_k(t)$ as a signal of a finite length T, thus making $Y_k(f,T)$ its Fourier transform. With these definitions the power spectral density of $y_k(t)$ is defined as[2]:

$$S_{yy}(f) = \lim_{T\to\infty} E\Big[\frac{1}{T}Y_k^*(f,T)Y_k(f,T)\Big] \tag{2.42}$$

Where * denotes complex conjugate. Note that this is a *two-sided spectrum*, meaning the spectrum is symmetric around zero. In many practical applications it normal to apply a *one sided spectrum*[2], which is defined simply as $G_{yy} = 2S_{yy}$, where the range begins at zero and thus keeps the total power of the signal intact.

2.2.3 Cross Spectral Density

Assuming the auto power spectral density is the equivalent of variance for a signal, the cross power spectral density is the covariance of two signals. Hence, if two signals are $y_k(t)$ and $x_k(t)$, and their Fourier transforms are $Y_k(f,T)$ and $X_k(f,T)$, then the cross spectral density is:

$$S_{yx}(f) = \lim_{T\to\infty} E\Big[\frac{1}{T}Y_k^*(f,T)X_k(f,T)\Big] \tag{2.43}$$

This too is a *two sided spectrum* as explained in 2.2.2.

2.2.4 Coherence function

From the auto and the cross spectral densities the coherence function is defined as:

$$Coh_{yx}(f) = \frac{|S_{yx}(f)|^2}{S_{yy}(f)S_{xx}(f)} \tag{2.44}$$

This is a function that shows the correlation between the two signals at the given frequency with a value in the range of 0 to 1, where 0 is completely unrelated, while 1 is perfectly related. If two signals are expected to follow each other this can be used as a way of confirming this. In practical applications this too may be an early indication of whether an expected natural frequency is real, and several signals correlate at this frequency, or not real, and for instance made by local noise on one of the signals.

2.2.5 The Frequency Response Function

The frequency response function(FRF) is defined as the function transforming the input of the system to the output. This may for instance be force input to acceleration output which is the normal use in structural dynamics. It can be defined from the equation of motion in the frequency domain, which is obtained by transforming both sides of the equation of motion in the time domain showed in equation 2.1. This gives the following:

$$\Big[-f^2[M] + if[C] + [K]\Big]U(f) = F(f) \tag{2.45}$$

Here $U(f)$ is the system output, and $F(f)$ is the input. The definition of the FRF making it:

$$H(f) = \frac{U(f)}{F(f)} \tag{2.46}$$

Meaning:

$$H(f)^{-1} = \Big[-f^2[M] + if[C] + [K] \Big] \tag{2.47}$$

for systems based on the equation of motion.

The FRF plays a major role in many of the techniques of modal analysis.

2.2.6 Welch's Estimate

A signal can often contain a lot of noise A way of removing some of this noise is by making a PSD with what is known as a Welch's estimate. The estimate provides noise reduction by dividing the signal into K parts finding the spectral density of every part and averaging these. The part are often overlapping each other. This can be shown mathematically as:

$$S_{yy}(f) = \frac{1}{K} \sum_{k=1}^{K} p_k \tag{2.48}$$

Here p_k is the PSD of part k. Although the Welch's estimate reduces noise it might introduce other problems to the frequency analysis. A negative effect may be the appearance of new frequency components in the PSD, which were not present in the actual signal. This is called leakage, and is caused by discontinuities at the ends of the sections the signal is divided into. To minimize the negative effects of this phenomenon the signal parts are multiplied by a function made to weigh the middle of a section heavier than the edges. The most common of these functions is a Hanning-window, which is given by[8]:

$$w(t) = 1 - cos^2(\frac{\pi t}{T}); \qquad 0 \leq t \leq T \tag{2.49}$$

This window is zero at both ends of the section time, which will remove discontinuities and reduce the leakage.

2.3 Operational Modal Analysis

Operational Modal Analysis (OMA) is an umbrella-term for methods utilized for modal analysis with response data only. The methods aim to find primarily natural frequencies, mode shapes and dampening coefficients from a large amount of measured output data. Welch's estimate, as mentioned, is a OMA-method.

For dynamic structures the response data is usually acceleration measured after the construction of the structure, and often during working conditions. There are many different

approaches to OMA. This thesis will in this section concentrate on the covariance-driven stochastic subspace identification(Cov-SSI), which is what will be utilized for the modal analysis of Bratsberg foot bridge. This is a fairly complicated method in Operational Modal Analysis.

A section will also be dedicated to explaining the stabilization diagram which is the end product of the Cov-SSI method.

2.3.1 Covariance-driven Stochastic Subspace Identification

To explain this method we need to look deeper into some qualities of the stochastic discrete state space model derived in 2.1.3. In most OMA methods the stochastic parameters w_k and v_k from (2.37) and (2.38), are zero mean white noise stochastic processes, also known as the white noise input. This means they have covariance matrices defined as[9]:

$$E = \left[\begin{Bmatrix} w_p \\ v_p \end{Bmatrix}^T \begin{Bmatrix} w_q \\ v_q \end{Bmatrix} \right] = \begin{bmatrix} [Q^{ww}] & [S^{wv}] \\ [S^{wv}]^T & [R^{vv}] \end{bmatrix} \tag{2.50}$$

when $q = p$ and
0 when $q \neq p$
Here q and p represent arbitrary time constants. The assumption of white noise is vital to the SSI principle, and when the input w_k and v_k are white noise, so is the output u_k. This means the covariance matrix for output and output shifted i steps becomes:

$$[R_i] = E \begin{bmatrix} u_{k+i} & u_k^T \end{bmatrix} \tag{2.51}$$

These i matrices have all the information necessary to identify the process. A covariance equivalent model can then be defined as the estimated state-space model characterized by correct covariance. This means the model is able to describe the statistical properties of the system response[3]. The covariance matrix of the state vector is:

$$\Sigma = E \begin{bmatrix} y_k & y_k^T \end{bmatrix} \tag{2.52}$$

This is independent of the measurement and process noise, thus giving us:

$$E \begin{bmatrix} u_k \; {w_k}^T \end{bmatrix} = 0 \tag{2.53}$$

$$E \begin{bmatrix} u_k \; {v_k}^T \end{bmatrix} = 0 \tag{2.54}$$

This, with the previous assumptions made for this model, may with some mathematical reformulations, lead to the relations:

$$[\Sigma] = [A_j][\Sigma][A_j]^T + [Q^{ww}] \tag{2.55}$$

$$[R_0] = [C_j][\Sigma][C_j]^T + [R^{vv}] \tag{2.56}$$

$$[G] = [A_j][\Sigma][C_j]^T + [S^{wv}] \tag{2.57}$$

$$[R_i] = [C_j][A_j]^{i-1}[G] \tag{2.58}$$

Where $[G]$ is the next state output covariance matrix defined as:

$$[G] = E\Big[y_{k+1} \;\; u_k\Big] \tag{2.59}$$

For the Cov-SSI method the qualities displayed in (2.58), are very important. This is because the R_i matrix can be calculated directly from output, and its decomposition therefor allows the estimation of the state-space matrices and a solution of the system identification problem [3].

With these identities in place, a derivation of the method will be given.

Derivation of Cov-SSI

The input to this method is the measured data, gathered in a matrix $[Y]$, where each column is a data channel, for example consisting of acceleration data. hence this matrix will have a dimension of $(N \times l)$ where N the number of samples per channel, and l is the number of channels.

A difficulty with the method is the fact that a system of order n is observable/controllable if and only if the observability/controllability matrix is of rank n as well. This can be a problem as it is normally unknown. Since underestimation of n will lead to incorrect results, it is normal to overestimate it. This will however lead to the creation of nonphysical poles. These are mathematical solutions that do not apply to the physical system and most therefor be separated from the real modes of the system afterwords.

The calculations begin with computing the output correlation matrices

$$[R_i] = \frac{1}{N-1}[Y_{1:N-i}][Y_{i:N}]^T \tag{2.60}$$

Here R_i is the correlation matrix with time lag i, and the subscript of the Y-matrices shows which rows, or block rows, are included from the original Y matrix. Then the matrices are joined together in a Toeplitz matrix:

$$[T_{1|i}] = \begin{bmatrix} [R_i] & [R_{i-1}] & \cdots & [R_1] \\ [R_{i+1}] & [R_i] & \cdots & [R_2] \\ \vdots & \vdots & \ddots & \vdots \\ [R_{2i-1}] & [R_{2i-2}] & \cdots & [R_i] \end{bmatrix} \tag{2.61}$$

Each correlation matrix having dimension $(l \times l)$ giving the Toeplitz matrix dimension $(li \times li)$. To identify the modal parameters a dimension of at least $(n \times n)$ is needed for the Toeplitz matrix. This makes the number of block rows having to fulfill the inequality:

$$li \geq n \tag{2.62}$$

Assuming estimation of system rank is completed, the factorization of the the Toeplitz matrix is:

$$[T_{1|i}] = \begin{bmatrix} [C_j] \\ [C_j][A_j] \\ \vdots \\ [C_j][A_j]^{i-1} \end{bmatrix} \begin{bmatrix} [A_j]^{i-1}[G] & \cdots & [A_j][G] & [G] \end{bmatrix} = [O_i][\Gamma_i] \tag{2.63}$$

Where the vertical matrix, $[O_i]$, is the observability matrix and the horizontal matrix, $[\Gamma_i]$ is the reversed controllability matrix with dimensions $(li \times n)$ and $(n \times li)$ respectively. Then so a *singular value decomposition*(SVD) is completed. Yielding:

$$[T_{1|i}] = [U][S][V]^T = \begin{bmatrix} [U_1] & [U_2] \end{bmatrix} \begin{bmatrix} [S_1] & 0 \\ 0 & [S_2] \end{bmatrix} \begin{bmatrix} [V_1] \\ [V_2] \end{bmatrix} \tag{2.64}$$

Where $[U]$ and $[V]$ are unitary matrices, and $[S]$ is a diagonal matrix containing the singular values of the system. The factorization of $[S]$ into $[S_1]$ and $[S_2]$, is done so that $[S_1]$ contains all the non-zero singular values in descending order, while $[S_2]$ contains all singular values equal to zero. This makes it possible to reduce equation (2.64) into:

$$[T_{1|i}] = [U_1][S_1][V_1]^T = [O_i][\Gamma_i] \tag{2.65}$$

Where $[U_1]$ and $[V_1]$ has dimensions of $(li \times n)$ and $(n \times li)$ respectively. To express $[O_i]$ and $[S_i]$ explicitly we divide (2.65) into:

$$[O_i] = [U_1][S_1]^{1/2}[I] \tag{2.66}$$

$$[\Gamma_i] = [I]^{-1}[S_1]^{1/2}[V_1]^T \tag{2.67}$$

Where $[I]$ is the identification matrix. It is chosen as the transformation matrix here, as it is the simplest of the infinite similarity transformation matrices that could have been used.

From the definitions of $[O_i]$ and $[\Gamma_i]$ in (2.63) we can now extract $[G]$ and $[C_j]$.

To find $[A_j]$ we need a different approach. First the one-lag shifted Toeplitz matrix is needed. It is defined as:

$$[T_{2|i+1}] = \begin{bmatrix} [R_{i+1}] & [R_i] & \cdots & [R_2] \\ [R_{i+2}] & [R_{i+1}] & \cdots & [R_3] \\ \vdots & \vdots & \ddots & \vdots \\ [R_{2i}] & [R_{2i-1}] & \cdots & [R_{i+1}] \end{bmatrix} = [O_i][A_j][\Gamma_i] \tag{2.68}$$

From the previous definitions $[A_j]$ can be expressed as:

$$[A_j] = [S_i]^{-1/2}[U_1]^T[T_{2|i+1}][V_1][S_i]^{-1/2} \tag{2.69}$$

Now, with all these identities in place, it is possible to solve the system end extract the modal identities.

Solving for Modal Properties

Solving the eigenvalue problem for $[A_j]$ will provide us with $[M_j]$, here a diagonal matrix containing the eigenvalues μ, and $[\Psi]$, containing the complex eigenvectors.

$$[A_j] = [\Psi][M_j][\Psi]^{-1} \tag{2.70}$$

To find the physical mode shapes from $[\Psi]$, we pre-multiply by $[C_j]$:

$$[\Phi] = [C_j][\Psi] \tag{2.71}$$

where $[C_j] = [C_c]$ under the zero order hold condition[9]. From here it is an simple process to convert the values to continuous time to find the modal parameters of each mode. Using (2.31) we get:

$$\lambda_m = \frac{ln(\mu_m)}{\Delta t} \tag{2.72}$$

And so the natural frequency, damped natural frequency and dampening ratio of mode m are:

$$\omega_{n,m} = |\lambda_m| \tag{2.73}$$

$$\omega_{d,m} = Im(\lambda_m) \tag{2.74}$$

$$\xi_m = -\frac{Re(\lambda_m)}{|\lambda_m|} \tag{2.75}$$

Now, if the order of the system was overestimated, which is common, some of these eigenvalues will not be actual physical modes, only mathematical poles. To separate these from each other a stabilization diagram is used.

2.3.2 Stabilization diagram

As a method to filter out the mathematical poles created from overestimation of the system order, a stabilization diagram can be used. To create a stabilization diagram, all found poles in all calculated system orders are plotted as a function of frequency. Because the real physical modes ideally would be the at the same frequency for all system orders, they would ideally appear as straight vertical lines in the diagram. But, because of noise and non perfect information the poles will also be non-perfect. They will however vary less with system order than the mathematical poles.

Therefore, depending on the system, different criteria for variation in natural frequency and dampening can be set by the user. These are usually defined:

$$\frac{|\omega(n) - \omega(n+1)|}{\omega(n)} < a \tag{2.76}$$

$$\frac{|\zeta(n) - \zeta(n+1)|}{\zeta(n)} < b \tag{2.77}$$

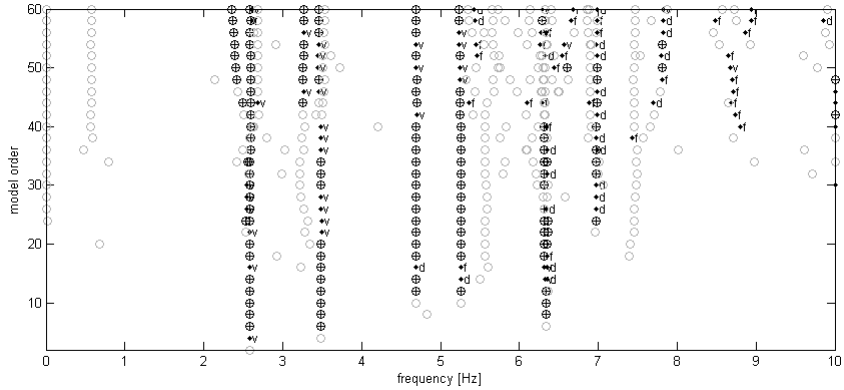


Figure 2.1: Stabilization Diagram example.

Where n symbolizes mode number, and a and b are criteria set by the user. The natural frequency usually varies little with system order, and so the criteria a is normally low, typically 0.01. For low-damped systems the variations in dampening is often higher and harder to estimate. This normally leads to a higher criteria for dampening variations, typically 0.05.

Figure 2.1 shows an example of a stabilization diagram. The modes that are within the criteria are marked as black, while the unapproved are in grey. The lower frequency modes are easily detectable as clear vertical lines. However the modes from 6 Hz and higher are more chaotic and further investigation might be necessary.

2.4 Signal Processing

The appropriate signal processing is very important during all sorts of measurements, and in post processing measured signals to insure high quality data and correct results. If the filtering, sampling rate or synchronization is incorrect this can easily corrupt the data, making it useless. This section explains the theory behind some of the different signal processing techniques, and the consequences of wrong use.

2.4.1 The Nyquist frequency and Aliasing

When measuring a signal, the sample frequency f_s is an important parameter. A high sample frequency leads to more accurate, higher resolution data, able to capture higher frequencies. However, it leads to heavier computational needs and larger file sizes. By definition, to capture a signal of a certain frequency, a sampling rate of two samples per

period is required. This minimum sample frequency is called the Nyquist frequency f_N and is defined:

$$f_N = \frac{1}{2\Delta t} = \frac{1}{2} f_s \tag{2.78}$$

Where f_s is the frequency to be captured, and Δt is the the time between each sample, or the sampling period.

The sampling frequency is a parameter that requires consideration before any measuring is initiated. If the measured signal is to be acceleration of a large slender construction with presumably low natural frequencies, the sampling frequency should be low, in order to not make the files unnecessary large. If the object of measurement is a vibrating machine part, it needs to be considerably higher.

If the sampling is set to only capture lower frequency signals, but there are other unwanted signals present with frequencies above the Nyquist frequency, it might cause an effect called aliasing. This happens under the digitalization of an analog signal when the sampling is ”synchronized” with the unwanted signal in such a way that frequencies not really present appear to be. After the measurements are done, there is no way to distinguish the real frequencies from the ones created by aliasing.

Figure 2.2 shows an example of aliasing, where the red line is the captured signal. As can be seen the captured signal was never present in the original.

To avoid this phenomenon it is necessary to filter out the higher frequencies with a signal filter before the digitalization of the signal.

2.4.2 Signal Filtering

A ideal filter would completely remove all contributions from unwanted frequencies while leaving the desired frequency band with its original amplitude. This is not possible. It is however possible to approach the ideal by increasing the order a filter. This will however become very computationally heavy with high filter order.

A filter’s response to a signal is often divided into three parts. Pass band, the signal frequencies that should remain as they are. Stop band, the part of the signal desired to be minimized. And transition band, the transition created by the filter between the two former.

Another filter property that can have negative effects on the signal’s resemblance and amplitude stability in the pass band is ripples, which is instability in the filters response. This can cause distortions both in the pass and stop band. Here a compromise must be made between accurate filtering, computational speed, and other filter qualities. Several filter types are available. Here only the Butterworth filter will be described.

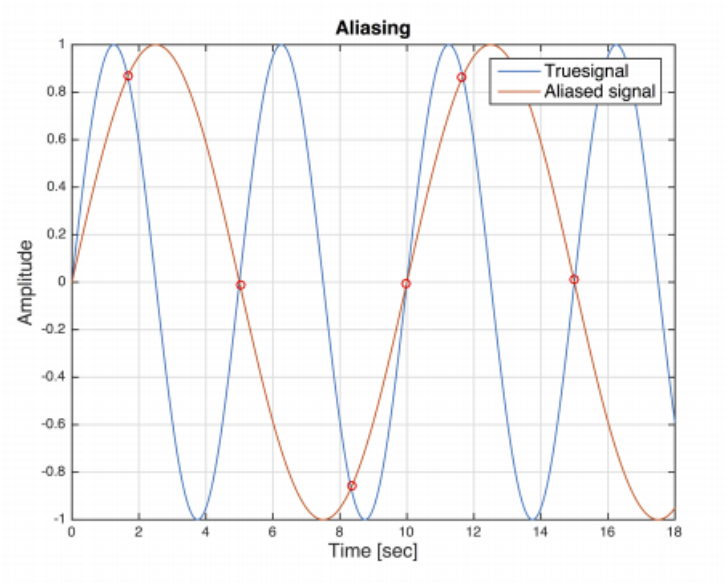


Figure 2.2: Example of Aliasing.

Butterworth Filter

The Butterworth filter is designed to have a flat as possible pass band, i.e no ripples. A low pass Butterworth filter has its response described by[10]:

$$|H(i\omega)|^2 = \frac{1}{1 + (\frac{\omega}{\omega_n})^{2N}} \tag{2.79}$$

Where i is the complex constant, ω is frequency in rad/s, ω_c is the cut-off frequency and N is the order of the filter. In figure 2.3 we see the response of the low-pass Butterworth filter from LabView[11] with different orders. Here the cut-off frequency, from where the signal ideally would be removed, is set to 100. The cut-off must of course be set in consideration with the sampling frequency as to minimize the signal coming through above the Nyquist frequency. It is normal as a safety precaution not to have the cut-off higher than 80% of the Nyquist frequency.

In figure 2.3 it is possible to see how the filter approaching the ideal rectangular response with increasing order. The Butterworth filter is however not always the preferred filter, because of the slow roll of, or wide transition band, in the lower orders.

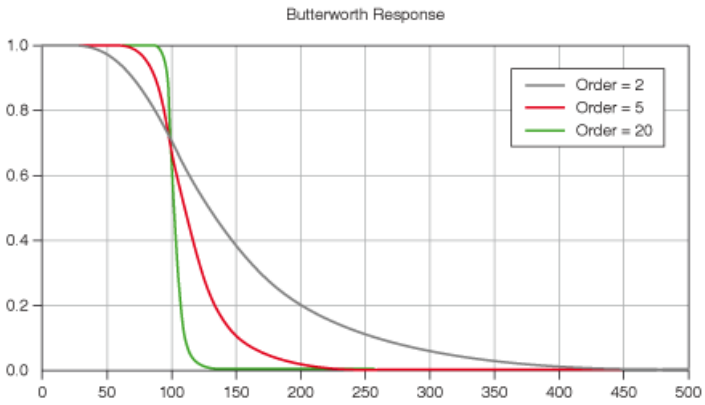


Figure 2.3: Response of low-pass Butterworth filter

2.4.3 Decimation

Decimation is the process of downsampling, or resampling, a signal after it's been recorded. This is a normal process to decrease file size. It can however lead to aliasing much in the same way as the continuous-to-digital sampling process can. Here too a filter is required to remove the effects of the unwanted frequencies.

Chapter 3

Programming in Labview

This chapter will focus on the different needs and solutions of the software written during this master thesis. There will also be a brief explanation of the functionality of the hardware, and how to properly apply the advantages of these.

3.1 Objective of Program

It is the authors impression that there have been problems for previous projects to write a continuous acquisition program with sufficient reliability and synchronization of data. Especially when using a compact RIO expansion chassis. The request from the faculty for this master thesis was to write a reliable program with perfect internal data synchronization and with the option of external synchronization through GPS time stamping. This external GPS synchronization would make each Compact RIO measuring system completely independent of each other during measuring, and removing the need for an expansion chassis, the connection wires and the problems that may arise here. The data synchronization would then happen after measurements are completed.

The program should also contain functionality to assure high quality uncorrupted data output, ease-of-use, and supervision of the program's current state. This means the software should include:

- A configurable low pass filter, which, in coordination with the sampling rate, should prevent aliasing.
- A possibility for downsampling the data before storage, also in coordination with the low pass filter.
- An option to view the program's status and the captured data in real-time for visual inspection.

- To create files to save the data in a column-wise format.

The final version of the program additionally contains a trigger value for a chosen channel, which will initiate the writing of data, and functionality to choose time duration of each stored file. There is also implemented a option for a communication with an anemometer through a NI 9871 module. This is however neither tested nor used during this project since it was introduced too late in the process. Therefore, it does not work at this point.

3.2 Hardware

3.2.1 Compact RIO

The hardware used for all testing and measuring in connection with this thesis was mainly the Compact Rio 9036 and 9067 made by National Instruments (NI). These are similar in all relevant ways, and will both be referred to as cRIOs. The cRIOs contain two processing targets, a real-time processor for communication and signal processing and an user-programmable FPGA to implement high-speed control[12].

The real-time processing unit is made as a stable platform to implement control algorithms and offers a wide range of reconfigurable I/O modules and several internal clock frequencies. It is programmed through National Instruments graphical programming language LabVIEW or C, C++ or JAVA.

The field programmable gate array, or FPGA, allows high performance data manipulation on a reconfigurable fabric which facilitates true parallelism in the program with custom timing and triggering in hardware. It can be programmed separately and is connected to the real-time unit through an internal PSI-bus. It can only be programmed with LabVIEW.

The compact RIO system is connected to a computer with an Ethernet cable which provides the user with the interface and configuring opportunities. Its normal use is as an industrial controller where size and ruggedness is important. Normally used as a headless system (without user interface), often in harsh conditions, stll with connection options for supervision or viewing logged data[13].

A power supply of 24V is used for power, and for the purposes of data storage a USB memory stick was used directly plugged into the cRIO .

3.2.2 Other Hardware

The following hardware was used in addition to the cRIOs to measure accelerations and GPS timestamp these:

- Kistler K-Beam Triaxial MEMS Capacitive AC Accelerometer, model 8395A2DOATTA00
- NI 9234 dynamic signal acquisition module

Figure 3.1: Compact RIO-9036. Here seen with 6 modules for acceleration data input and one GPS synchronization module

- NI 9467 GPS Timestamping and Synchronization Module
- Kistler Reference Shaker Type 8921B02
- Magnets, attachable to the accelerometers with screws
- Data and power cables
- Coaxial extension cables
- Ethernet cable
- USB memory sticks
- Power supply set to 24V
- Computer with LabVIEW installed

The accelerometer used from Kistler[14] measures accelerations in all three directions and outputs an analog signal of $\pm$ 4 Volt for each channel, which equals a acceleration range of $\pm$ 2g. The values oscillate around 0 and the minimum available frequency is 0 Hz. When connected to the NI 9234 modules the three channels are separated into three analog input channels. Here a power supply is also necessary. Each NI 9234 module[15] has 4 channels for analog input. The sampling frequency of these modules range from 1.652 kHz to 51.200 kHz.

The NI 9467 module[16] uses GPS satellites to give out a pulse per second (PPS) time stamp accurate down to $\pm$ 100ns. There are several time standards available. It can also

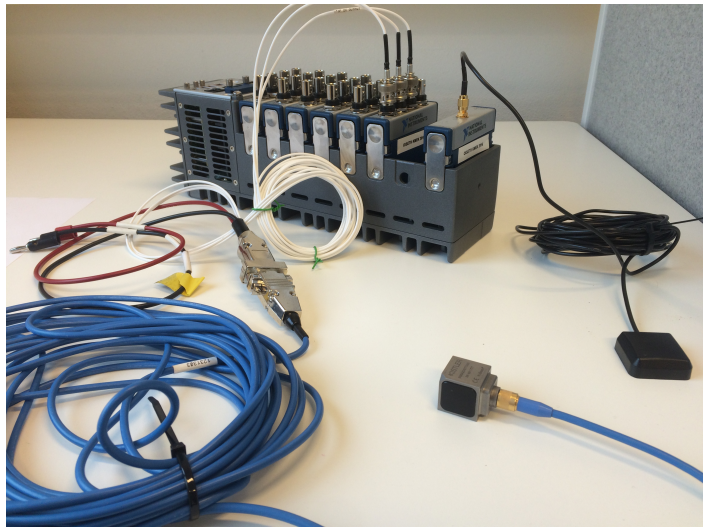


Figure 3.2: Setup with a NI 9467 GPS module (right) with antenna. And a Kistler accelerometer connected to NI 9234 (left) with necessary wires.

be used for GPS positioning, although this function is not used here.

The Kistler Reference Shaker [19] is a vibration exciter specifically designed to verify/calibrate the sensitivity of accelerometers. There are 7 available frequencies in the range of 15.92 to 1280 Hz with adjustable amplitude.

3.3 LabVIEW Programming

This section will explain all the different functionality and solutions of the written software and the mechanisms and reasoning behind them. As the FPGA offers significantly higher reliability and performance regarding input/output processing, this is the natural choice for the acquisition of data. However, the FPGA lacks the possibility for data storage. This means the data must first be communicated to the real-time target, then saved from there. The communication is done through a FIFO, which is explained in section 3.3.2. This means, as both the FPGA and the real time target is used, two programs are needed. As the real time target contains the upper level program this will also be referred to as the host, and the FPGA also referred to as the target.

A program is in LabVIEW-terms called a Virtual Instrument, or a VI. A VI consist of a front panel and a block diagram. The front panel is the constructed user interface of the program with the buttons and controls for operating it, and a block diagram describes the flow of data and the actual functionality. VIs are organized in projects. Here the different hardware components are listed, such as the compact Rio, the FPGA and the I/O modules, and organized with the different programs. It is also possible to add new hardware and

access for example the different I/O channels and the FIFOs to change settings.

The version of LabVIEW used for this project was LabVIEW 2015 SP for Windows made by National Instruments. To gain access to and to install software on the cRIO, NI MAX is also necessary. The explanation of the program will begin with an attempt of outlining the general way in which data flows and the programs function. After this an explanation of the the data acquisition VI will be given in detail, before the text will move on to explaining the internal data transmission and end with the data storage VI. A basic knowledge of programming is expected from the reader.

3.3.1 General Program Flow

The two VIs must be started separately. The FPGA VI, or target VI, should be started first. When starting the FPGA VI it will begin by waiting for a notice from the Host informing the target that it is ready. When this notice is received the target will implement some start-up settings. The host VI will begin by implement some settings, including overwriting the user set settings on the FPGA if they are not equal. The host will then send a notice to the target VI letting it know it is ready, and so they both move on.

Now the FPGA VI will wait for GPS synchronization. Here it will stay until the GPS signal is locked, or the user manually bypasses it. When this is done the FPGA will start reading data from the modules and writing them to the FIFOs. This will fo on until the program is stopped.

Meanwhile the host VI has started reading the data from the FIFO, but as long as the program is not told to write to file, it will just be deleting the data. In other words the host is continuously emptying the FIFO while waiting for a trigger to write to file. This trigger can be two things, a user set condition for a chosen acceleration channel to exceed a chosen acceleration value, or a manually pressed button telling the program to write.

When the writing condition is fulfilled, the program will enter a loop where it waits for one of the FIFOs to have a user set number of elements. When this happens this number of data elements will be written to file. When writing these elements is completed, the program will return to checking for the proper amount of elements. This will continue until the currently written file exceeds the user set file length.

This marks one completed file. The program will return to waiting for a trigger to write, and if fulfilled, it will begin another file. The GPS synchronization loop constantly runs in parallel at the target VI, and if locked will feed GPS timestamps to a file written in parallel. If GPS is not locked this file will still be written but only contain ones.

With this in place, the different parts of the software will be explained more in detail, and the choices more reasoned.

3.3.2 Data Acquisition VI

In figure 3.3 we see an overview of the block diagram from the FPGA VI.

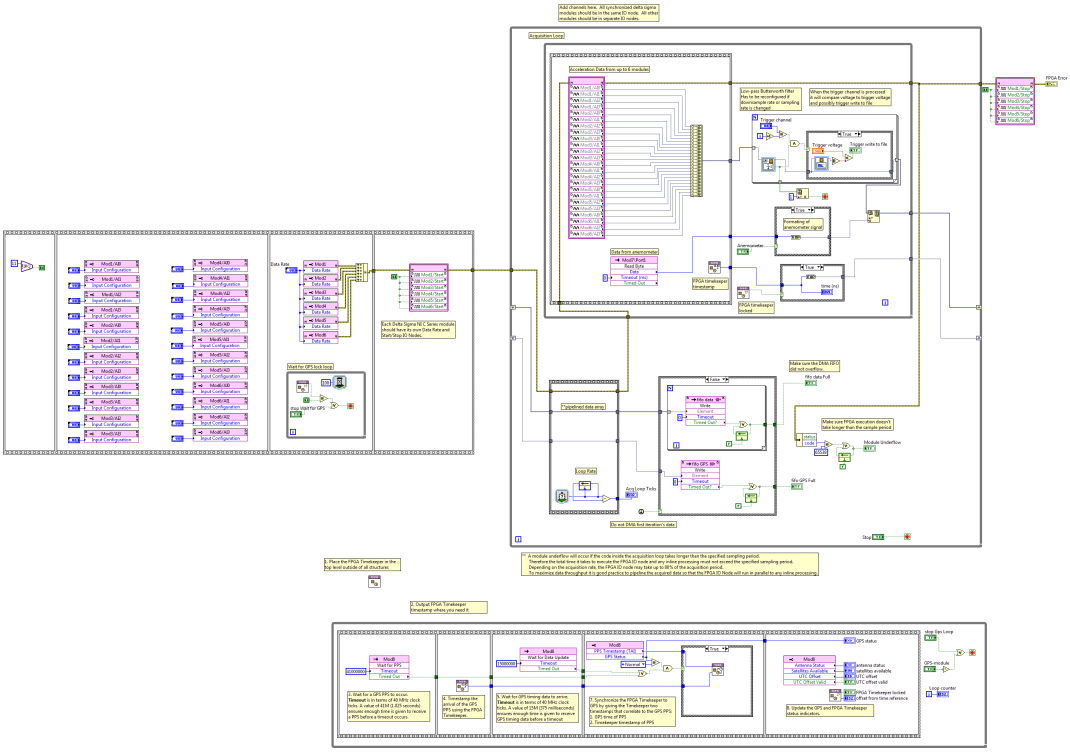


Figure 3.3: The Block diagram from the data acquisition VI

Although it's hard to make out the details from this figure, it is possible to see three main parts. The section diagram on the left is the initialisation commands. The main acquisition loop is the top right loop and the GPS synchronization loop is on the bottom.

Initialisation

The initialisation part of the program consist mainly of setting certain options on the modules to their correct value from the controllers located at the front panel. See figure 3.10. These include the sampling rate and the DC/AC setting. It also contains a small loop to stall the program until GPS signal is acquired or the user presses a button to proceed anyway. And lastly it starts the acquisition modules.

Data Acquisition and Filtering

The acquisition loop is split in two. One part to acquire and filter the data, and one part to write it to FIFOs. The top part is seen in figure 3.4. An absolutely essential part of this program is that all the samples from the different channels are taken at the exact same time. To accomplish this, all the internal clocks of the modules must be synchronized in their respective settings, and all the samples must be taken in a single operation. This operation is completed in the 24 lines high node, with wires going into an 'Build Array' function. At the same time as this acquisition, a time stamp must be taken from the FPGA timekeeper which is synchronized with the GPS time.

The way this program is currently constructed, the GPS time stamp will be taken somewhat before the acceleration is acquired. When the program enters the section where these data are retrieved, the timestamp will be taken immediately. The acceleration will however not be acquired before the timing matches the sampling rate of the modules. Hence the time lag here will be the sampling period minus the time the program uses to operate on the data. The downsampling will not affect the actual time lag, but will make it smaller in comparison with the output rate. However, since the goal with the GPS timestamp is to synchronize with other files, the lag will be the same in both files and cancel out. Even so, this is still an area of possible improvement.

After acquisition the accelerating data is built into a vector with length equal to the number of channels, and then sent to a filter.

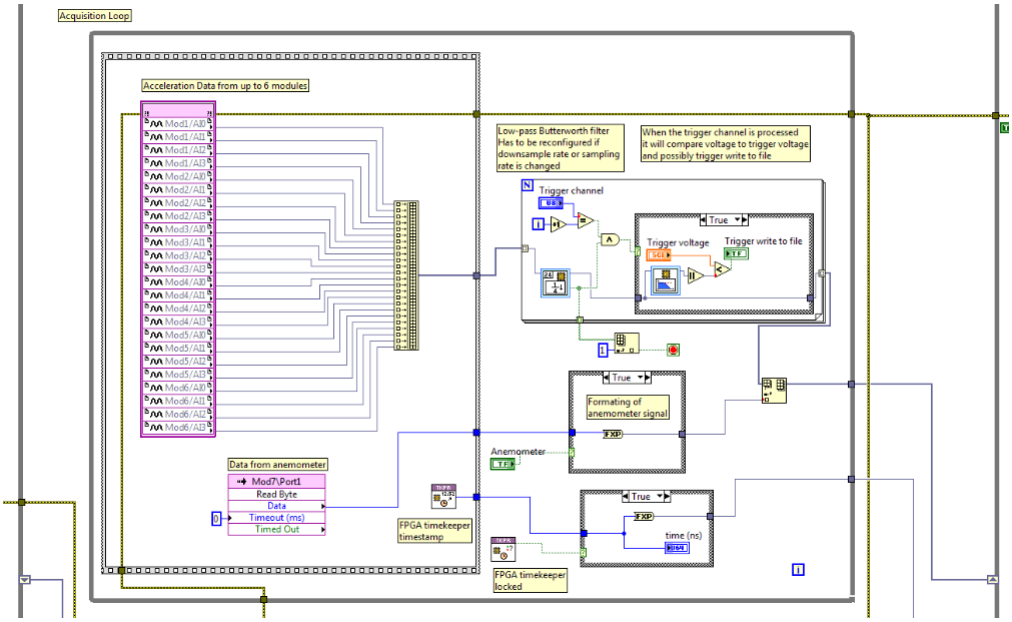


Figure 3.4: The acquisition part of the block diagram

Several filters are available for use in LabVIEW FPGA. The chosen signal filter is the pre-built LabVIEW rational resampler filter[17]. This combines downsampling and a low-pass filter into one express VI. This resampler works by taking in one channel at the time, working its way through all channels every loop iteration. If the resampler is set to down-sample by a factor of 4, the resampler will return valid output every 4 loop iteration. This valid output will be a average of these 4 last samples for all channels. The filter will also give a true boolean values that is used to stop the downsampling loop, and make sure only valid data is sent to the next step.

The rational resampler will also have filtered the signal for for frequencies above the Nyquist frequency. Which, if the downsampling rate is still 4, would be 1/8 of the original input frequency, according to equation (2.78). In figure 3.5 the dialog box for the rational resampler can be seen, including the filter response. As can be seen, the cut-off frequency is at around 0.125 of the original input rate, which is 1/8 as expected. The filter response is very good with a flat response in the passband, before quickly dropping down to a -100dB attenuation with some rippling effects in the stop band.

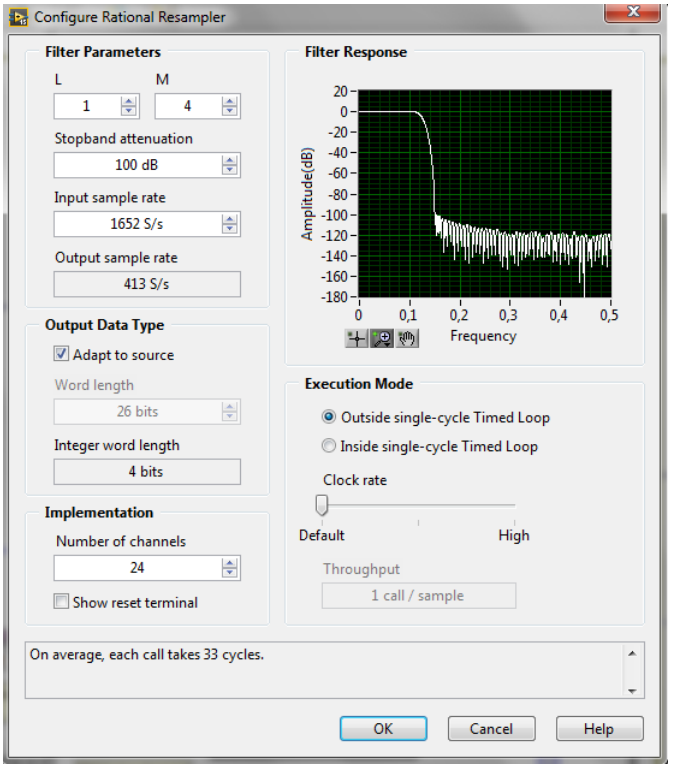


Figure 3.5: The settings of the rational resampler including the filter response.

This is also the location of the trigger condition. One user chosen channel is sent trough

another low pass filter in an effort to remove frequencies unwanted in the trigger signal. Then it is compared to a trigger condition, also set by the user on the front panel of the target VI. If the trigger condition is true a light will be lit on the front panel and a signal will be sent to the host.

Together with the acquisition of acceleration and GPS timestamps, is the operation for communication with the anemometer. This is as previously mentioned not functioning, and is only sending out invalid data.

There are some operations preformed on the data from the time stamp and anemometer. As a way of increasing readability of the output files, the program will check if the local timekeeper is synced with the GPS time. If it is not, the value of 1 is sent from there to show there is no timestamp. The anemometer will, as long as a button is turned off on the front panel, send the value of 0, and be joined into the vector with acceleration data. The time stamp will be kept separate because it is a larger data type. After this the data will be pipelined.

Pipelining

The process of pipelining is to increase the performance of the VI. By the use of a shift register the data is sent back to the ”start” of the while loop allowing the loop to process the two parts in parallel. And, since this is done on a FPGA where the code is implemented in hardware, it is in true parallel, meaning the two signals are literary being processed at the same time. It does however take two loop iterations for the first data to be written to the FIFOs.

FIFOs

In figure 3.6 the bottom half of the acquisition program is shown. The primary function here is writing data to the FIFOs. A FIFO, or first inn first out, is a method for organizing and manipulating a data buffer, where the first element written into the buffer, is the first element read at the receiving end. There are several types of FIFOs in LabVIEW, but the most appropriate here is a DMA target-to-host FIFO. DMA, or direct memory access, allows the FIFO to store the data in dedicated block memory on the FPGA, which the host can access and read. The size of the FIFO should be 5 to 10 times bigger than the read bulk size. The speed on the host side is slower than on the FPGA, and needs therefore to operate on data in bulks. How big these bulks need to be, depend on the sampling rate. The reason there are two different FIFOs for the acceleration data and GPS timestamps is that while the acceleration data is a 32 bit data format, the time stamp is 64 bit. The FIFO needs all data in the same size, and since the time stamp data can’t be made smaller without losing accuracy, the only way to the program could be limited to one FIFO was by making all the other channels 64 bit, using unnecessary large memory resources. One could argue that the reliability of the synchronization is compromised by keeping the timestamps and the acceleration data separated. However, because of the way the program is structured, it is impossible to loose sync. The different path are directly connected in such a way that one

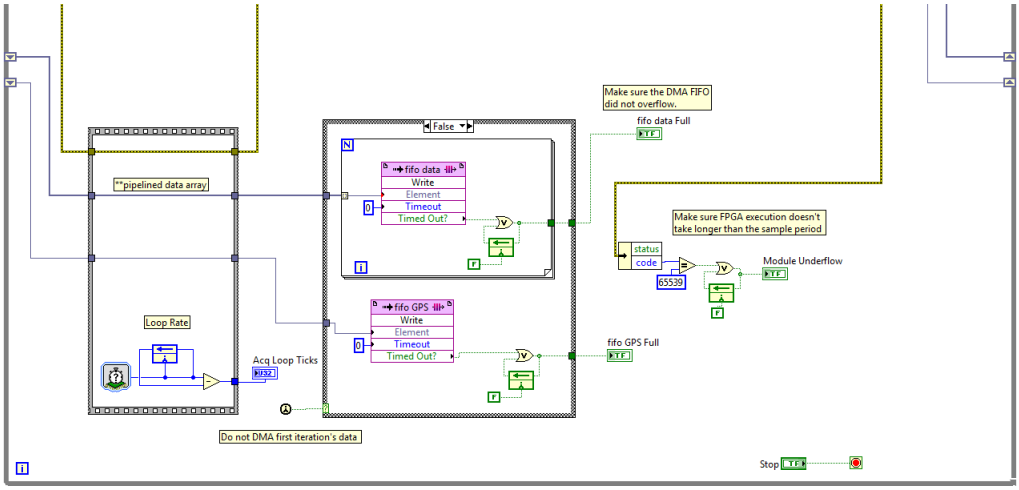


Figure 3.6: Writing to FIFO

cannot move ahead in the data flow without the other.

One way it is possible to loose data however, is if the FIFO becomes full, also referred to as overflow. When this happens, information will not be written to the FIFO and thereby lost. If this happens, a light will turn on in the front panel, basically marking the results as useless.

Underflow may also be an issue. This happens when the host tries to read info when the FIFO is empty, or if the loop period is longer that the sampling period. The former should not be a problem in this implementation, as the host does not read data before a certain amount is stored in the FIFO. The latter however, can be a problem with high sample rate.

GPS Synchronization

The GPS synchronization loop can be seen in figure 3.7.

If the GPS module is synchronized to the GPS time, it will receive a pulse-per-second (PPS) signal giving it the correct GPS time. First loop with sync, the program will match the internal FPGA clock to the GPS time. This will make the internal FPGA clock, or timekeeper, continuously able to give accurate time stamps in GPS time. Every loop after this, the FPGA timekeeper will be compared to the GPS time and corrected. At the end of every loop, with or without sync, the program will indicate the status of the GPS module and timekeeper on the FPGA front panel. This includes whether the FPGA timekeeper is locked to GPS time or not, and the timekeepers offset from GPS time at the previous sync. Buttons to stop the loop is also provided.

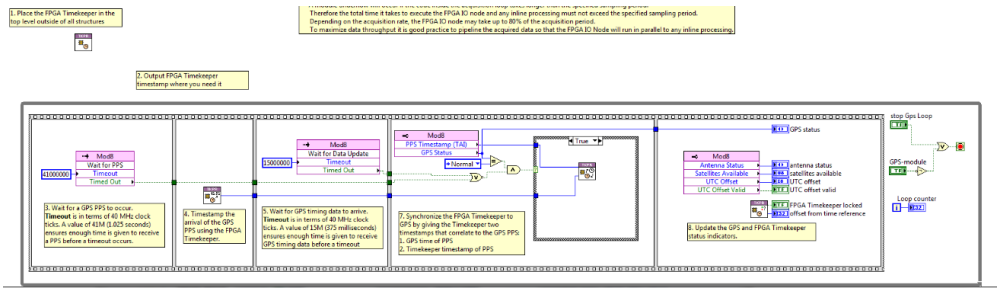


Figure 3.7: The GPS time synchronization loop.

The GPS time standard used is TAI, or International Atomic Time in English, and it is based on some 400 atomic clocks around the world making the basis for the standard second length. In LabVIEW it is a 64 bit positive integer showing the number of nanoseconds elapsed 00:00 January 6. 1980.[18][16].

3.3.3 Host VI

The main purpose of the Host VI is to read data from the FIFOs and write them to file. To do this it needs to create and close files and make sure the data is organized properly before writing. It also provides the user with a live update of the data being processed, in the form of two wave charts showing the acceleration and the frequency content of two separately chosen channels.

In figure 3.8 the entire block diagram of the Host VI can be seen.

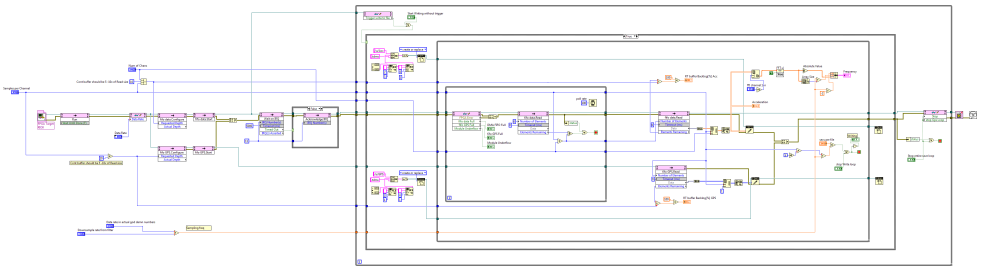


Figure 3.8: The block diagram from the Host VI.

Reading and organizing data

Reading data of a FIFO is, by definition, done one element at a time in the same order they were written. To avoid underflow the program will check how many elements are waiting. When this number exceeds a user set limit the VI will read the limit number of elements. These elements will be arranged into an array of n columns, where n is the number of

channels. For the FIFO containing acceleration, n is a user set number that must match the number of acceleration channels plus 1 from the anemometer. If this number does not match, the data will be written to file in a very chaotic unorganized way, which makes it very hard to process.

From the FIFO containing the timestamps, all the elements are organized in a vector where the first element is the timestamp for the first row of the acceleration matrix, the second timestamp is for the second row, and so on.

Displaying Real-Time Data

As a way of checking if all the channels are giving proper data, real-time wave charts are used to display acceleration data on the front panel. One chart directly displays the acceleration data, and one preforms a simple fast Fourier transform and outputs the frequency content. Both charts display user chosen channels. This is however only preformed when the program is writing data, not when it is waiting for the trigger condition to be satisfied.

Creating and Saving Files

The file format used to store the data is .tdms. A binary format specifically made by National Instruments for LabView, thus being the preferred option. It can be opened in Excel or Matlab, although it needs conversion to be operated on.

Because the program has the ability make files with a certain length, then reset and write a new file, there was a need to generate unique file names so that they are not overwritten. This process is shown for the acceleration file in figure 3.9

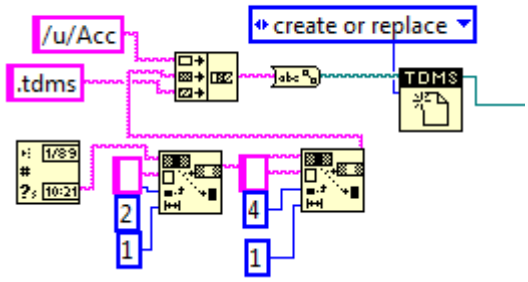


Figure 3.9: The create file section of the block diagram.

There are several ways of doing this. One option would be to call them 1,2,3.. and so on, i.e. increasing numbers as the VI produces more files. But this would lead to the files being overwritten if the program was restarted. The current solution is to use the internal clock to timestamp the file name with a $0000AM/PM$ type format. This file name will

have either Acc or GPS added in front of it, depending on which file it is. Although the time format file name is not a perfect solution since there is always a possibility files will be overwritten if the program runs for several days, it works for the current uses.

There are two different files written. One containing the acceleration matrix, and one with the GPS timestamp vector. The reason for doing this is the same as for using two FIFOs; different data types. If both were to be stored in the same file, without loosing precision in the timestamp, the data size of the acceleration data would have to be doubled. With the current 24 acceleration channels the combined data size is almost halved with the current solution. More precise the used space is 13/25, or 52%, of the alternative one-file solution
.
The writing to file process is simple. If the write command is told to write an array to file, it will write in the same rows/columns format as the array. If a new array is written it will be put below the data already there.

When the length of the file(in seconds) is longer than the user specified time, the file will be closed. The program in its current state will return to waiting for the trigger condition, and if satisfied will begin writing a new file. It is worth noting that there is functionality to begin writing without trigger. If desired, setting the trigger condition to zero will make the program constantly write, but the data will be divided into the desired length. If one large file is the preferred format, the file length can be set to, say, 100 years. This way the program will only create and write to one file(until there is no more space).

3.3.4 Front Panels

The front panels of the two programs are quite self explanatory and will only be briefly presented here.

FPGA Front Panel

In figure 3.10 FPGA VI front panel can be seen. The box inn the top left corner contains the sample rate setting and buttons for whether a GPS module and an anemometer is present. It also contains a live feed of the FPGA timekeeper time after GPS synchronization.

The top middle box contains the error lights and error message display, a stop button to stop the acquisition loop, and a tick counter showing how many ticks the acquisition loop uses per iteration. How much one tick is depends on the internal clock. Here the internal clock is 40MHz making one tick equal to 1/4,000,000 second.

The top right box contains the GPS status updates, the button to manually stop waiting for GPS synchronization, and a button to stop the entire GPS loop.

The bottom left box is the settings for the different accelerometer input, and whether they

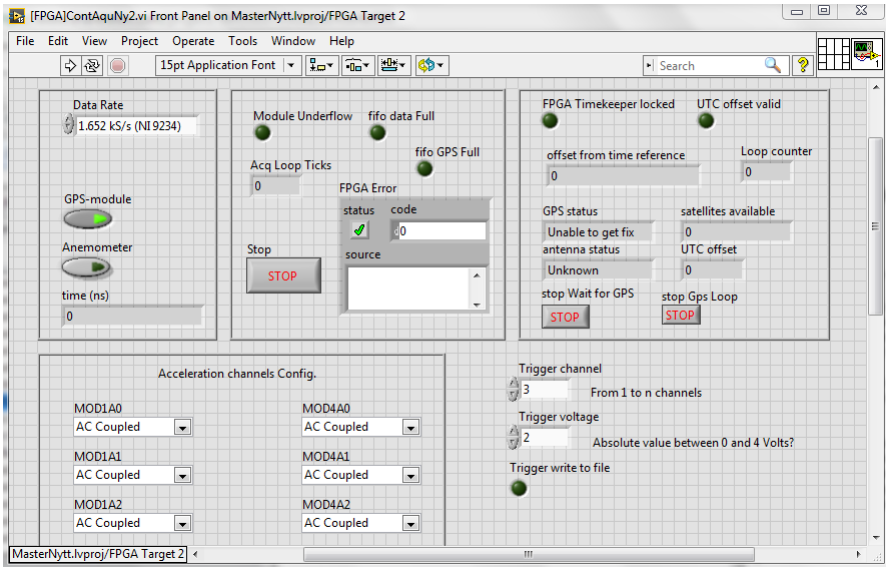


Figure 3.10: The Front Panel of the FPGA VI

are AC or DC. The bottom left contains the trigger settings, and a light that turns green when the condition is fulfilled.

Host VI Front Panel

In figure 3.11 the front panel of the Host VI is shown. The controls and indicators are:

- Sample per Channel: How many elements the write loop reads and writes each bulk per channel. Also used to set FIFO sizes.
- Data Rate: Sets the sample rate of the modules.
- secs per file: Length of files in seconds.
- Num of Chans: Number of channels. Must match the number of channels written to the FIFO.
- Downsample rate from filter. Used here mostly in calculations. Must be equal to the downsampling rate in the rational resampler.
- Data Rate in Numbers. Used in calculations, must be the same as the Data Rate, and must be typed in manually.
- RT Backlogs. Real-time percentage of how full the FIFOs are.
- Stop write loop. Stop the writing of the file manually, and return to waiting for trigger.

- Stop entire save loop. Stops the entire VI when the next file is completed.
- Start Writing without trigger. Manually triggers the writing to file.

In addition there are lights to indicate errors and whether the program is currently writing to file. The two large windows are the real-time updates of the acceleration and the frequency content of the chosen channels.

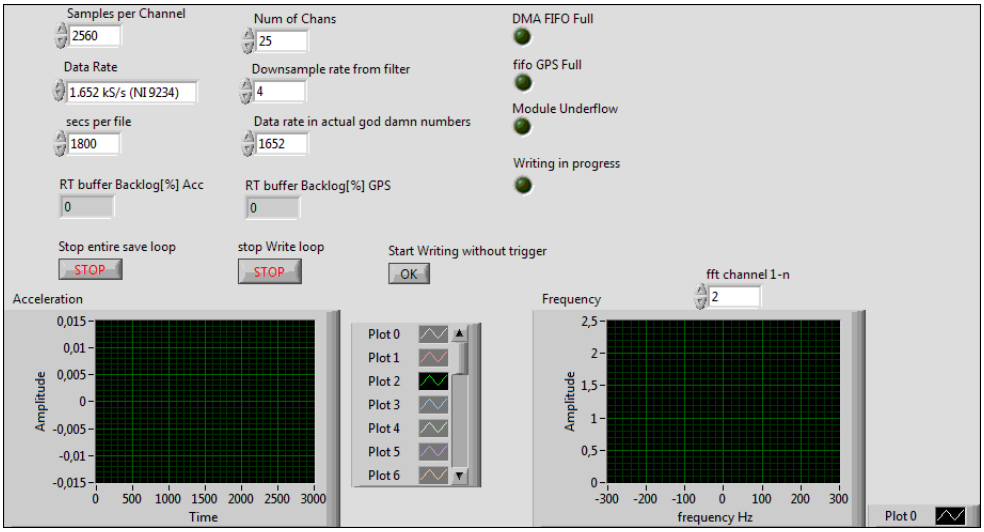


Figure 3.11: The Front Panel of the Host VI

3.3.5 Incorrect Shutdown

Incorrect shutdown, either by force aborting the program without using the incorporated shutdown buttons, or shutting off the power, is not recommended. It might lead to start up issues next time the program is turned on as the different elements are not reset properly. It might also lead to the corruption of the stored data, even though this has not been the case in the tests preformed on this issue.

Another consequence is that the number of elements in the acceleration columns and the GPS columns might be different. The reason for this is simply that writing the single column GPS file is quicker than the multiple channel acceleration file, so the GPS is ahead of the acceleration during each bulk. With normal shutdown both bulks will be completed and the lengths equal. The solution to this is simply to remove the last elements of the GPS vector, so the lengths become equal.

3.3.6 Changing the Program

There are especially two aspects of the program that are likely to be changed often. The first is the filter and downsampling settings, and the second is the number of acceleration

channels.

The first is a simple process, access the block diagram of the FPGA VI and change the properties to the preferred settings. Make sure that this matches with the sample and downsampling rate of the front panel settings. Changing these settings will require recompiling the FPGA, which is a ca 1.5 hour process.

To change the number of channels, the acceleration acquisition node most be reconfigured. Remove the unwanted channels and the wires connected to them. The size of the 'build array' function most also be changed. In addition the rational resampler must be reconfigured to match the number of channels. This process will also require recompiling. Lastly, the number of channels setting on the front panel must match. If there is a need to remove a whole module, all instances of this module most be removed from the program. To add channels or modules the process is reversed, but in addition the internal clock of the new modules must be synchronized to the others in the module's properties.

Any other changes will require a thorough understanding of LabVIEW and all the elements of the written software.

Chapter 4

Testing the Program

To ensure the program was functioning properly several tests were designed to investigate whether the different elements of the VIs were behaving properly. The most important functions to test were:

- GPS-synchronization. Making sure the timestamps are equal from several compact RIOs so that the post synchronisation is correct.
- Signal filtering. Verify that unwanted frequencies are removed before sampling to prevent aliasing.
- Reliability. Confirm that there is no data loss, that the program can run for an extended period of time and work with all the different settings including high sample rate.

Several test were performed during the development of this program, and many after its completion. The two most important tests will be presented in this paper, as they cover many of the previously tested areas, and their results are representative of for the tests not shown here. These tests are:

- A test to confirm the Rational resampler is functioning properly, both the downsample function and the low-pass filer.
- Verify that the GPS timestamps are equal from several cRIOs and assert that they match their respective acceleration

4.1 Test 1: Signal Filtering

4.1.1 Set Up

To test the signal filtering a way of producing signals with a known frequency was needed. For this the Kistler reference shaker[19] was used. This makes it possible to change frequency and amplitude while writing to file. Hence, by changing the frequency gradually

higher towards and past the cut-off frequency the signal should disappear along with its frequency content.

The reference shaker does not offer a continuous specter of frequencies, but 7 different frequencies with adjustable amplitude. These are 15.92, 40, 80, 159.2, 320, 640, and 1280 Hz, and the amplitudes are 1, 2, 5, 10 and 20 m/s^2 RMS, RMS being the root mean square.

In figure 4.1 the entire set up is seen. From the left, this is: the power supply, the Compact RIO-9036, the Kistler reference shaker with a mounted accelerometer, and a lap-top computer. The compact RIO is connected to the lap-top with an Ethernet cable and the accelerometer through the coaxial cable. It also contains a memory stick for data storage. Both the cRIO and the accelerometer are connected to the power supply.

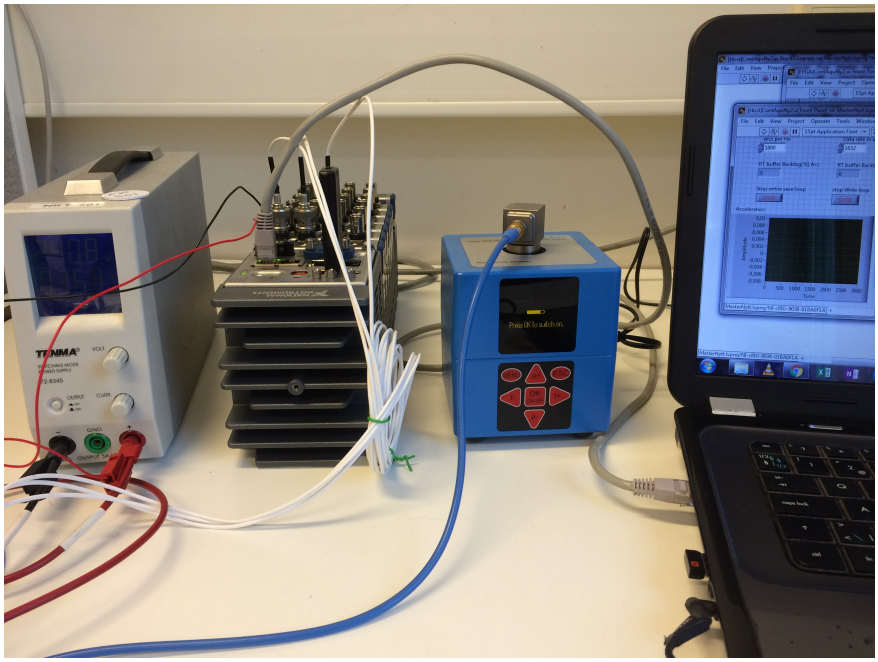


Figure 4.1: The complete setup for the filter test.

In the early faces of the program, a normal low-pass Butterworth filter was used before a manual downsample function. But it was decided that the rational resampler was a better option because of its filter response, and the fact that it averages the samples while downsampling.

Several tests were performed on the filtering during its development both before and after the change of filter. Three of the tests completed after the change of filter will be presented in this paper as they represent the most important qualities of the filter and the downsample function, and as they are representative of the others. These are:

- Sampling and downsampling rate at normal values, i.e. 1652 S/ch/s with downsampling ratio of 4.
- Sampling and downsampling rate at relatively high values, i.e 10240 S/ch/s and a downsampling ratio of 16.
- Sampling ratio, filter and downsampling rate at the highest possible values, i.e. 51200 S/ch/s and downsampling ratio of 64.

Where S/ch/s is samples/channel/second

4.1.2 Test with Normal Values

Execution

After all the equipment and VIs were up and running, the reference shaker was turned on. In this test neither the GPS time stamps nor the trigger condition were used, so these conditions were manually bypassed. The shaker started at a low frequency which was gradually increased until it was well past the cut-off frequency. The settings that affected performance during this test were:

- Sample rate: 1652 Samples/channel/second (S/ch/s)
- Downsample rate: 4
- Number of acceleration channels: 24
- Stop-band attenuation: -100dB

This makes the output rate 413 S/ch/s and the cut-off frequency equal to 206.5 Hz. The file length was set to 200 seconds.

Every time the frequency was changed at the shaker, so was the amplitude, so as to make it visible on the acceleration plot both in real time and in the results.

The shaker began at 40 Hz, was changed to 80, 159.2, 320 and lastly 640 Hz. The amplitude started at 5, was then changed at the same time as the frequency to 10, 5, 10 and 20 m/s^2 RMS. Each of the settings were kept for about 40 seconds. The two last frequencies are higher than the cut-off and should therefore not be visible in the results.

Results

In figure 4.2 and 4.3 the acceleration time series and its frequency content is displayed.

It is easy to see that the three first frequencies of 40, 80 and 159.2 Hz are there. After the change to 320 Hz after about 125 seconds, the signal is almost gone. A closer view of

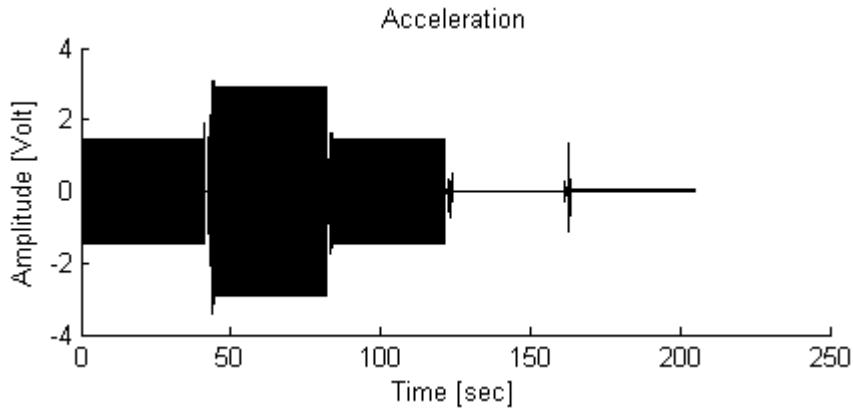


Figure 4.2: The Acceleration time series.

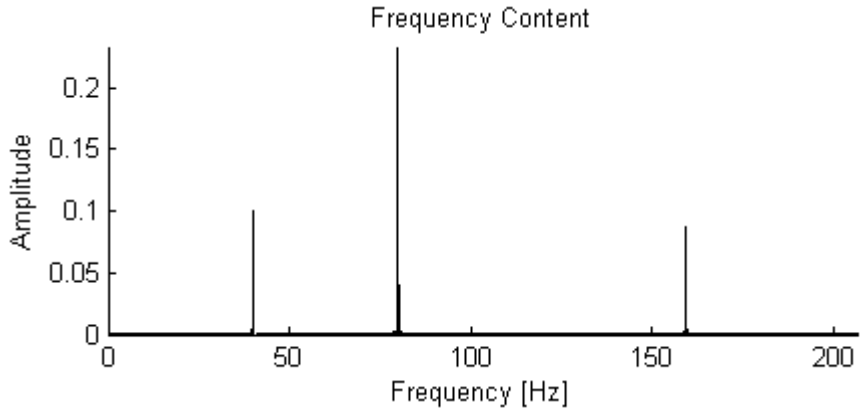


Figure 4.3: Frequency Content

this section shows values of about 0.05 Volt. The same applies to 640 Hz. The signal is almost completely gone and this part should, without the filter, have had an amplitude of 20 m/s^2 RMS. There is some noise at the intersections where the frequency is being changed.

The frequency content in figure 4.3 clearly shows the three peaks of 40, 80 and 159,2 Hz. There are no other peaks present.

4.1.3 Test with High Values

Execution

This test was preformed with relatively high values. But another aspect of this test is that it was designed to have its cut-off frequency at exactly the same as one of the frequencies offered by the shaker, namely 320 Hz. This provides some insight into how the filter works

in the transition band area. The performance affecting values were:

- Sample rate: 10240 Samples/channel/second (S/ch/s)
- Downsample rate: 16
- Number of acceleration channels: 24
- Stop-band attenuation: -80dB

This makes the output rate 640 S/ch/s and the cut-off frequency equal to 320 Hz. The file length was set to 90 seconds.

In this experiment the shaker was started at 159.2 Hz, moved up to 320 and lastly set at 640 Hz. During the entire session the amplitude was kept constant at 2 m/s^2 RMS.

Results

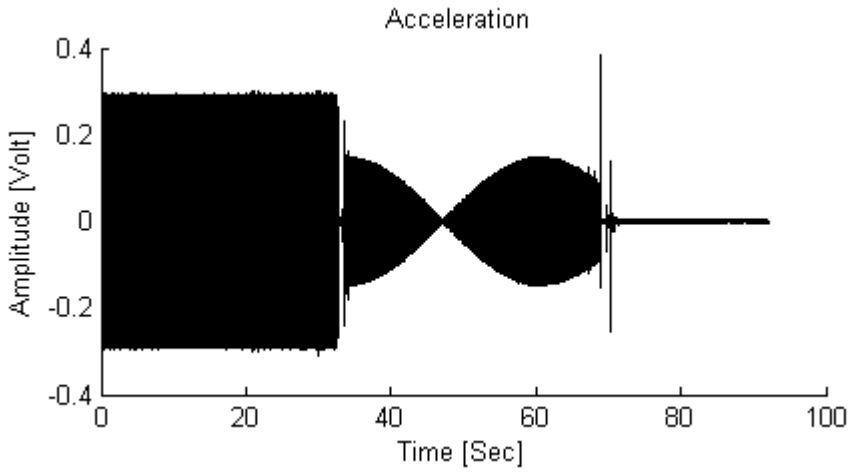


Figure 4.4: The Acceleration time series.

As can be seen from figure 4.4 the values from the first frequency are as expected, and the signal from the third part is filtered away as it should. However, the signal amplitude from the frequency equal to the cut-off is not stable. The amplitude looks like it is slowly changing value to follow a harmonic curve with a period of about 30 seconds.

The frequency content confirms this. The frequency of 159.2 Hz is present and there is a smaller contribution at 320 Hz. There are no other peaks.

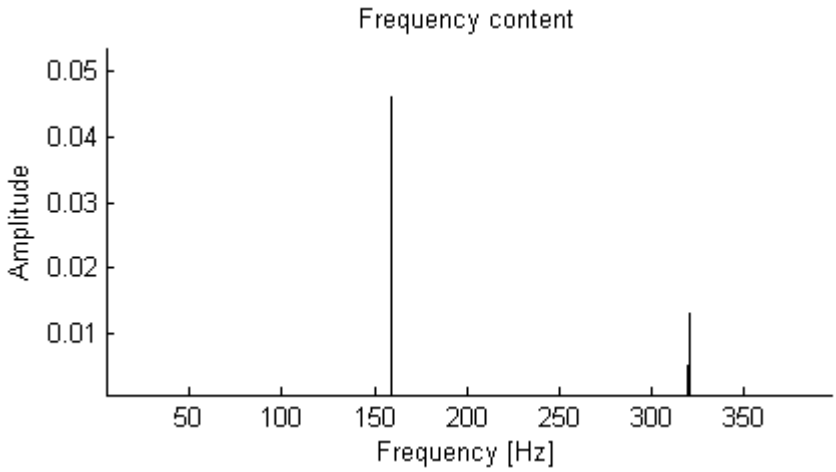


Figure 4.5: Frequency Content from the relatively high values test

4.1.4 Test with Highest Possible Values

Execution and Results

This test was designed in the same way as the previous one. But here the settings affecting performance are:

- Sample rate: 51200 Samples/channel/second (S/ch/s)
- Downsample rate: 64
- Number of acceleration channels: 24
- Stop-band attenuation: -80dB

51200 is the highest possible sample rate from the modules. A downsample rate of 64 was the highest that the rational resampler would allow with 24 channels and -80dB attenuation. This would lead to a output sample rate of 800 S/ch/s and an cut-off frequency of 400 Hz.

However, as soon as the program started to write, it gave notice of underflow. This could mean two things, (1) the host program tried to read from the FIFO when no data was present, or (2) the FPGA execution takes longer than the sample period. In this case the problem was the second. This was possible to tell because the frequency shown in the real-time frequency content was not the actual frequency, and because the front panel showed the 'number of ticks per cycle' was (exactly) 100000. With an internal clock of 40 MHz 100000 ticks gives a loop rate of 400 Hz. This is only half of the 800 Hz output sample rate necessary to keep up with the current settings.

4.1.5 Test 1: Conclusion

From the first test, and several other similar tests, it is clear that the rational resampler is working as intended. Both the output sample rate and the cut-off frequency are correct and the signals with frequencies above cut-off are almost completely removed. There is some noise in the acceleration time series when the shaker changes frequency, but this is probably lower frequencies that are not supposed to be removed by the filter.

The second test shows some strange results. It is clear that the filter is working for the values above the cut-off frequency as there are no signs of aliasing, or any signal in the time domain. As for the frequency equal to cut-off the amplitude of the signal coming through the filter changes. The frequency of this signal however, does not, as shown by the frequency content. This is important, as anything else could have created aliasing. Why the amplitude changes in this way is however unknown. The conclusion being, that the cut-off frequency should have some margin down to any frequency that is required to be captured correctly.

From the third test it can be concluded that the program does not have the capacity to run at those settings, actually far from it. With an output sample rate of half the required, the rational resampler is taking up far too many resources to keep up. The way to fix this problem, is to reduce some of the settings. Either the number of channels, the module's sampling rate, the stop band attenuation or a combination of these. It should be noticed that the downsample rate is not mentioned here. This is because it does not affect how long the rational resampler spends at each sample from the modules, and that is where the problem is. Since the loop rate for the acquisition loop was half of what it needed to be to uphold the sampling rate of 51200, by using half the channels, i.e. 12, the program should operate very close to its limit. (Although probably not functioning because of the few operations being used on other functions than individual channel processing.) It is however not recommended to use the software this close to operating limits.

The fact that these settings are too extreme is however not a problem for the designated uses. Most large constructions have their natural frequencies well below 50 Hz, and the program is shown to work well at a sample rate of 10240, which should be more than enough. It may also be mentioned that a test from before the program was expanded from 8 to 24 channels, ran without problems at 51200 S/ch/s, although this will not be documented here.

Should it somehow still be desirable to increase the performance of the program, one possibility would be to replace the rational resampler with several low-pass Butterworth filters connected to the separate channels before they are collected in an array. This would allow the filters to run in parallel on the FPGA.

4.2 Test 2: GPS synchronization

4.2.1 Set Up and Execution of Test

In order to test the post synchronization for two different compact RIOs, a way to create two equal acceleration signals were needed. By doing this, if the synchronization works, the signals from the parallel cRIOs should follow each other perfectly.

The solution was a simple cantilever beam created from a long thin iron plate, two wooden plates and a vice, as shown in figure 4.6. On each side of the iron beam, at the same place, an accelerometer was put, held in place with a magnet. One accelerometer was connected to each of the two compact RIOs. This way, when movement of the plate was induced, the accelerometers felt the same acceleration. Note that the two accelerometers do not share a coordination system. Two of the axes, y and z, are rotated 180 degrees in relation to each other. The direction in which this system will vibrate is the z-direction.

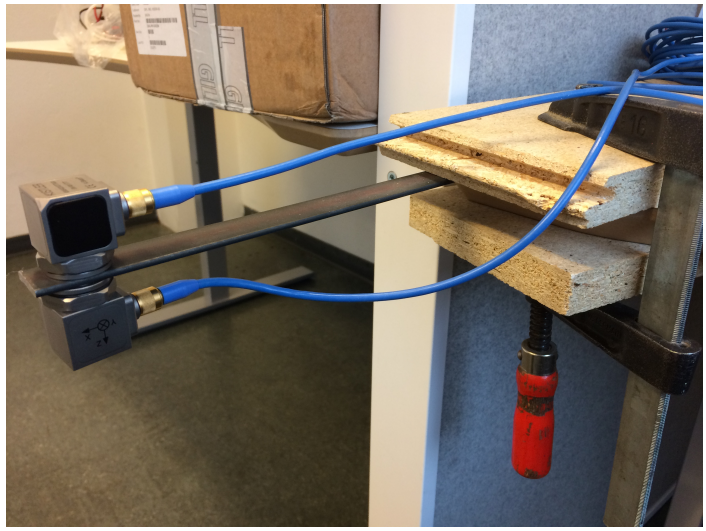


Figure 4.6: The setup to create equal acceleration signals for two compact RIOs.

Several tests were preformed with this setup, and as the program was in its final version, many other functions of the program were also tested. The notable ones being triggering the saving of data with a trigger condition and channel, controlling the length of the file and the file name generation. The test also checks whether when the chosen file length is reached, the program will return to waiting for the trigger condition.

The test was preformed by starting the two VIs on both compact Rios manually one by one. When both the programs were running and waiting for the trigger condition the plate was excited by dragging it a small distance downwards and releasing, giving the beam free decreasing vibration. This was done by hand. The process was repeated several times over the course of the test.

In order to test the other aspects of the program, some different settings were tried. By trying to write several files in one session (without restarting), it was simultaneously possible to check different settings for trigger condition and channel. By setting the trigger condition higher than 5 Volts, which is maximum output from the accelerometers, the program should not start writing data regardless of the movements induced and the chosen trigger channel. Likewise, by setting the trigger condition to zero the writing should commence regardless of movement and channel. The length of the file was also tested. This was done by checking if the difference from the first to the last time stamp matched the user set file length, and whether the sampling/downsample rate matched the number of samples.

All through the experiment the sample rate was set at 1652 S/ch/s. The downsample rate was 4, making the output sample rate 413 S/ch/s and the cutoff frequency 206,5 Hz. The file length was set to 180 seconds.

4.2.2 Results

The results were imported to Matlab for post processing. Figure 4.7, shows the entirety of both acceleration series plotted on top of each other against their own separate GPS time vector. The coordination system from series 2 has been rotated to match that of series 1. The GPS time series has been normalized to start at zero by subtracting the starting time of the series that began first. These plots are all in the z-direction.

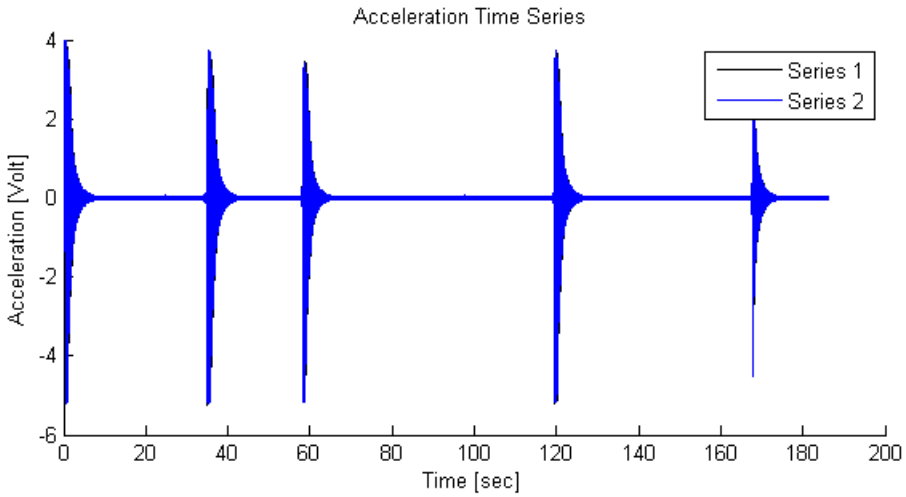


Figure 4.7: The entire acceleration series

It is hard to see both time series in this large scale figure but the general form of 5 damped free vibrations is clear. The file also seems to last the appropriate time of 180 seconds.

Figure 4.8 shows the free damped vibration better. This is a close view on the third manual excitation at around 60 seconds. It can be seen that in the beginning of the excitation,

the accelerations are too high for the accelerometer to measure. The acceleration range seems to be from 3 to -5 Volts. Where 3 is upwards, against gravity. This applies to both accelerometers.

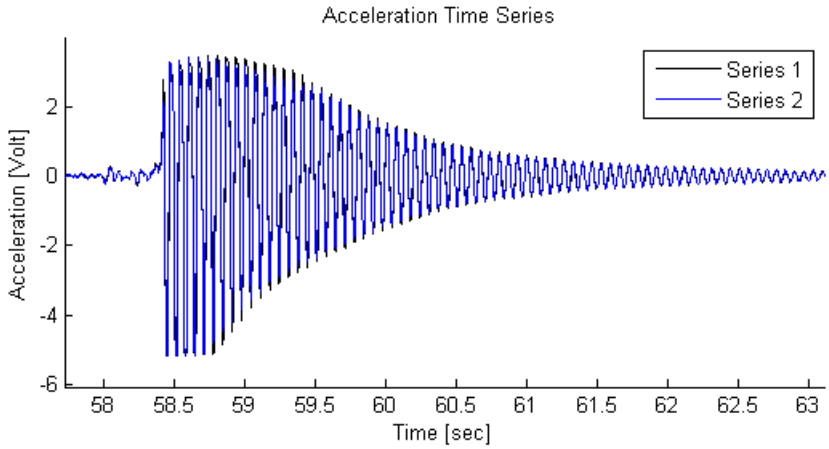
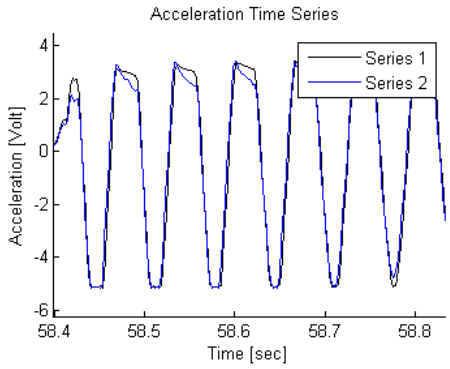
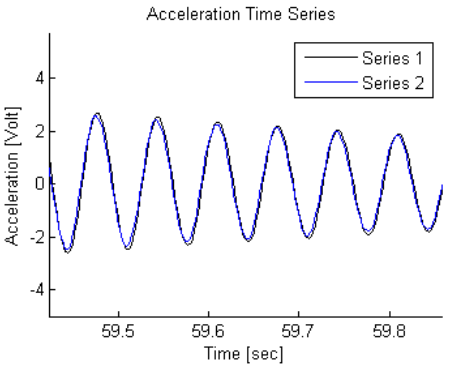


Figure 4.8: a zoomed view of the third excitation

In figure 4.9 a close view is shown, to see the individual signals. Figure 4.9a shows the maximum range of the accelerometer. The signals appear to differ a little at the maximum value as signal 1 has a sharper peak than signal 2. At the minimum value of -5 volts the signals are indistinguishable. This applies to the signals in figure 4.9b too, where the signals peak at more normal values. Not only are the signals now symmetric around 0, but the signals look exactly like a standard harmonious damped signal.



(a) A zoomed picture of the signals at maximum range



(b) A zoomed picture of the signals at normal values

Figure 4.9: Close ups of the signals at different locations

In figure 4.10 a very low value part of the signals between the excitations is shown. Here the only excitation is noise, and there are clear differences between the signals. The second time series appear to have an additional excitation compared to the first. These values are very small, but constant over the course of the measured signal when there is no other movement.

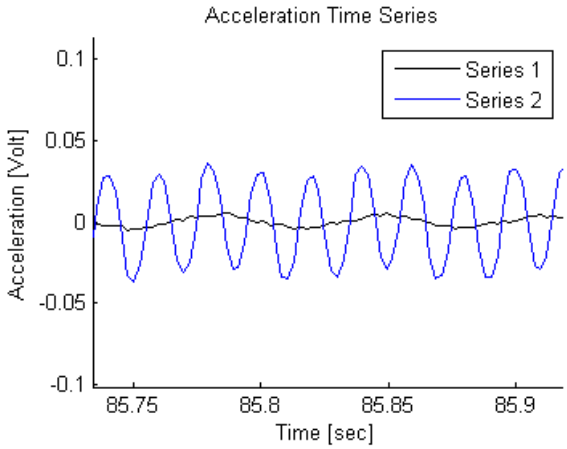


Figure 4.10: Very close picture between excitations

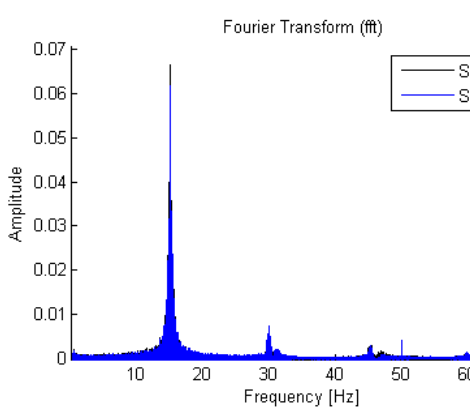
As seen in figure 4.11a the Fourier transformation shows one major peak at about 16 Hz. This is probably the first natural frequency of the cantilever beam. It is also possible to see traces of what could be the second mode at about 31 Hz. As seen in figure 4.11b the signals have different frequency content at a very narrow frequency band at about 50 Hz, this appears to be the contribution from the additional excitation described in the previous paragraph. Besides this small uneven part, the frequency content of the two signals is almost identical.

In figure 4.12 the beginning of the two signals are shown. The two signals were triggered by the same movement and the two separate programs began writing with a time difference of about 1ms. At the end of the signals, series two lasts about 1ms longer than series 1, meaning they are the same length in time duration. Both these acceleration time series, and the two GPS series, are exactly 76800 samples long.

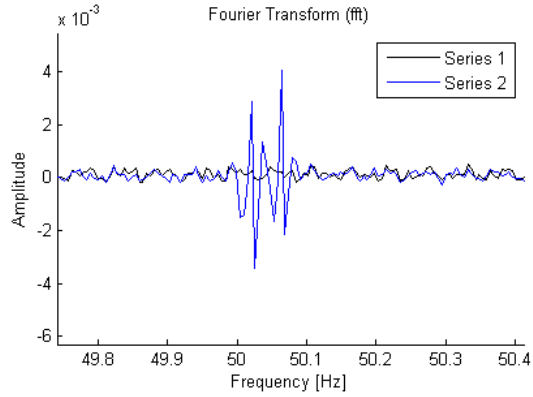
4.2.3 Test 2: Conclusion

From these results it can be concluded that the GPS synchronization between independent compact RIOs is working perfectly. The signals follow each other as accurately as can be expected, both in time and frequency domain.

It is seen in figure 4.7 that the signal length is about 185 seconds, even though the file length was set to 180 sec. The reason for this is that the host program that writes to file



(a) Frequency content of both signals



(b) A close up at what appears to be the frequency of the noise contribution

Figure 4.11: Frequency content of test 1

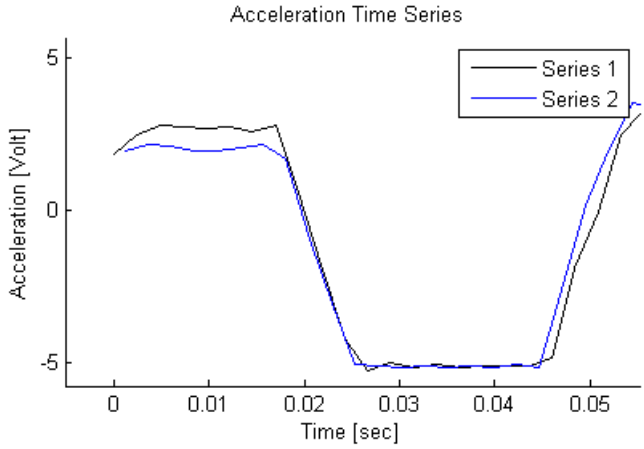


Figure 4.12: The start of the time series

handles data from the FIFO in bulk. When the time passed exceeds the user-set file length, the program finishes writing the bulk of data it is currently working on.

The constant additional excitation detected from signal 2 and displayed in figure 4.10 is considered to be noise. The source of the noise is however unverified. Tests were made to detect the source by changing the wires, uncoiling the wires and switching which accelerometer was connected to which compact RIO. The only conclusion was that the accelerometers were not the cause. It is possible that the power source was the cause of the disturbance, but after one wire change it disappeared, and without any changes it reappeared. Thus it remains unverified.

The other functions that were tested with this test, or tests like it, were also proven to be working properly. By setting the trigger to over 5 Volts the program did not start writing to file, and by setting it to zero the writing started immediately. The file length was also adjusted successfully ranging from 30 seconds to several hours.

4.3 Conclusion

From the tests presented here it can be concluded that the software is functioning as intended. All the different functionality is working properly, including the GPS timestamp synchronization and low-pass filter. Throughout the testing and general use of the program there has been no data loss. The files containing the acceleration and GPS timestamps have consistently had the exact same length, down to a single element. (Except with incorrect shutdown, as explained in section 3.3.5.)

The only unexpected behaviour in the software was discovered after the measuring of Bratsberg foot bridge, as mentioned in section 5.3. Here the length of the vectors containing the same GPS time span were slightly different. (9 elements difference in 11.39 million.) This is either caused by slightly different sampling rates in the acceleration modules, or by GPS time offsets in the beginning and/or end of the time series after being cut.

If the difference in length grows linearly with increasing time spans the problem is likely to be the modules sample rate. The solution here is to remove the extra elements evenly distributed throughout the longer file. If the difference in length does not increase with file length, the problem is more likely to be GPS time offsets. In this case investigation should be completed to see whether the start or the end (or both) are out of sync and then manually remove elements to ensure sync. The solution to GPS time difference would however only be necessary if it occurs on a larger scale. On the scale in which it occurs after measuring Bratsberg foot bridge, it would not matter how the extra elements were removed, it would not noticeably change the synchronization of the acceleration series or compromise the modal analysis.

All in all the software development is deemed a success.

Chapter 5

Modal Analysis of Bratsberg Foot Bridge

Figure 5.1: Set up at Bratsberg foot bridge

To test the written program in a practical, real life situation, measuring the response of a foot bridge seemed ideal. Experiments on this type of construction does not need as much administration and security measures as many others, as there is no need to stop any traffic, no need for heavy tools and no practical problems in accessing the location or mounting the accelerometers.

Several bridges were considered, and it was decided that Bratsberg foot bridge was the best option. It is a long slender bridge made primarily of wood. It has 4 supports, giving it 5 separate spans. A small excursion was made to the location in advance to see if it had any noticeable response that could be interesting to analyse, and to look for practical solutions to mount the necessary equipment. The conclusion being positive, the Bratsberg foot bridge was approved for testing.

5.1 Experiment Setup

The Bratsberg foot bridge is a relatively long bridge of 80 meters. Because the bridge has as much as 5 separate spans, it was decided to use all the available 12 accelerometers. The accelerometers were located in 6 pairs, each pair had one accelerometer on both sides of the bridge. There were one pair at the middle of each span, and in addition on pair was placed at the quarter point of the middle span. This span is expected to have the highest response values, and it was therefore considered the best place to mount the last pair of accelerometers to capture some higher degree modes. The pair in the middle of the middle span is expected to have the highest maximum response values i.e. acceleration, velocity and displacement.

In figure 5.2 the setup and locations of the different accelerometers, the compact RIOs and the power supply can be seen.

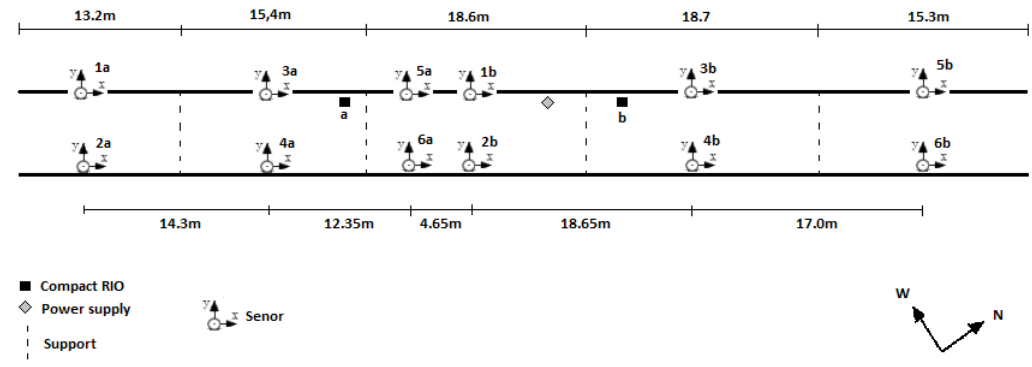


Figure 5.2: Set up at Bratsberg foot bridge

Since a big part of the experiment was testing whether the GPS time stamp synchronization would work in the field, two compact RIOs were brought. These were completely independent of each other during the measuring.

The first 6 accelerometers, 1a to 6a, were connected to the cRIO marked with an a. Likewise the second 6 accelerometers, 1b to 6b, were connected to the cRIO marked with a b. Extension wires were used when needed, and all accelerometers measured all three directions: x, y and z. Both the compact RIOs were put inside a cabinet designed for this type of experiment. They contain several 24V power outlets, to which both the cRIOs and all the accelerometers were connected. The cabinets in addition offer a cover from the weather. The cabinets were separately connected to the power supply.

To attach the accelerometers to the bridge some custom made steel plates were screwed on to the wooden beams running along the edges of the bridge, on these plates the accelerometers were attached with magnets. This can be seen in figure 5.3

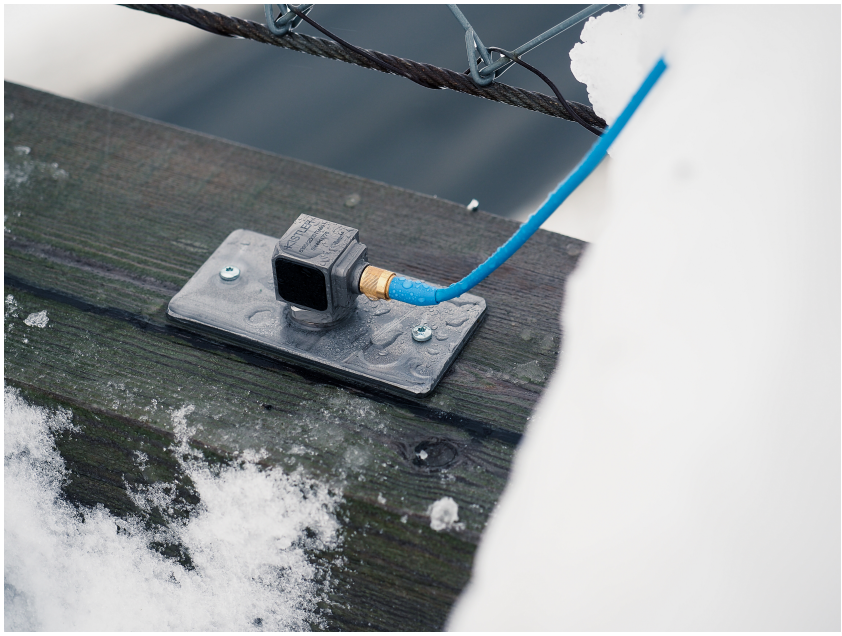


Figure 5.3: Setup at Bratsberg foot bridge

The Compact RIOs were compiled the day before the experiment as not to lose time on location. The chosen values for the program were:

- Sample rate: 1652 Samples/channel/second (S/ch/s)
- Downsample rate: 4
- Number of acceleration channels: 24

- Stop-band attenuation: -100dB

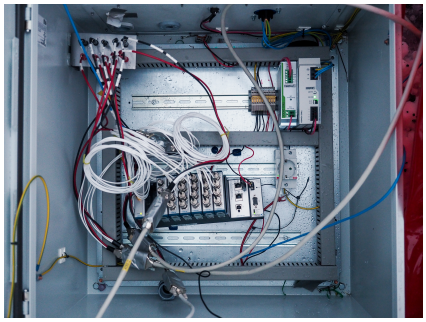
These values were primarily chosen because they were well tested and known to work.

The experiment was preformed in the middle of January, and, as may be expected, there was some snow on the bridge. How much this affected the dynamic properties of the bridge is hard to say.

5.2 Measuring

The measuring was done by turning on the separate RIOs one by one and manually triggering the program to write after the GPS signal was acquired. The signal was strong enough for synchronization after about a minute of waiting for both RIOs. Initiating the programs one by one created some lag between the start of the files on the different RIOs.

In the beginning there was a practical problem with the signals. A visual inspection of the signals was preformed to confirm that all the channels produced reasonable signals. Here the channels on the second RIO gave unlikely signals. This was recognized as noise resembling the signal given by the accelerometers when they are lacking power. The error was identified as being the custom made power source for the accelerometers not working. Consequently the accelerometers were connected to an other power source which solved the problem.



(a) A zoomed picture of the signals at maximum range

(b) A zoomed picture of the signals at normal values

Figure 5.4: Close-ups of the signals at different locations

The measurements were planned to consist of two series of two hours each. But because the power supply unexpectedly shut down twice while recording, the early time series have a more random length. The problem with the power supply was identified as a closed air intake and resolved. Thus the last time series is the correct 2 hours. Of the earlier time series that were interrupted, one ended with a length of about 20 minutes, and two with about 5.

During the time series, different types of excitation was performed. The 4 participants of the experiment spent some time replicating normal traffic on the bridge by walking and running across the bridge with some distance. The construction was also purposely excited in its natural frequency by the participants, by jumping together in a common rhythm and finding the natural frequency. During this type of excitation the bridge was noticeably moving, and this could be seen with the naked eye. Lastly the bridge was left alone to natural excitation by wind for long periods of time.

5.3 Post Processing

The results were brought into Matlab for synchronization and post processing. When imported there were 24 channel of output from each RIO but only 18 channels in use. The remaining channels contained only noise and were deleted. The time series did not begin or end at the same time. As a consequence the time and acceleration vectors were cut at both ends so they only contained information in the same time span.

At this point an unexpected problem occurred. During the same GPS time period one of the time vectors were 9 elements longer than the other(of about 11.39 million elements). This is either caused by slightly different sampling rates in the acceleration modules, or by GPS time offsets in the beginning and/or end of the current time series. Here the problem is minimal and the extra elements were removed evenly distributed from the longer file. The consequences of this problem are discussed in the conclusion of Chapter 4 in section 4.3.

Once the vectors started and ended at the same GPS-time and were equally long, they were inserted into one matrix. They were ordered in the manner of 1x, 1y, 1z ,2x.. and so on. The 18 first channels are from the compact RIO marked 'a' on figure 5.2, and the following 18 are from 'b'.

Now that all the channels were synchronized into one matrix, the signal was decimated. The output sample rate during the experiment was 413 S/ch/s which was unnecessarily high, and will take a lot of processing power to analyze properly. As all the natural modes of interest were expected below 10 Hz the signal was decimated to 20 Hz. This was done with the Matlab decimate function[20], which applies a low pass filter to avoid aliasing.

All this carried out a Matlab add-on called Macec 3.2 was used to analyze the data, and the OMA method covariance-driven stochastic subspace identification was used for the modal analysis.

5.4 Results from Bratsberg Foot Bridge

In this section the results of the analysis will merely be presented and commented on. The discussion will happen in the next section.

Two of the time series and their results will be presented. One is the 2 hours long time series, and the other is the 20 minutes long first time series. These are the two longest and should therefore contain the most accurate information about the properties of the bridge. They will be presented together through the results so the differences and equalities can be described.

From here on the 2 hours long one and the 20 minutes long one will be referred to as '2Hours' and '20Min' respectively. The channels will be referred to as 1 to 36, where 1 is sensor 1 direction x, 2 is sensor 1 direction y and so on till 36, which is sensor 12 direction z.

All the results will focus on the vertical direction, or z, as this is the direction where most of the excitation and natural modes are expected. The presentation of the results will begin wit the time series in themselves, move on to power spectral densities, then show the stabilization diagrams with the natural frequencies and damping estimates, and end with the physical mode shapes.

5.4.1 Time Series

In figure 5.5 and 5.6 the time series of 2Hours and 20Min are shown respectively. In both figures two channels are shown, both in z direction, but from different sensor positions. Channel 18 comes from RIOa while channel 24 comes from RIOb. These channels are next to each other on the bridge and are therefore expected to follow each other closely.

All the highest peaks in 2Hours is caused by deliberate excitation, i.e. the participants jumping on the middle span. The part where the participants were running on the bridge is in the time span of 4000 to, but not included, the peak at about 4400s. The rest is the bridge's excitation under normal working conditions, influenced only by natural forces and people crossing.

The 20 minutes time series have no deliberate excitation, and is only subject to people crossing the bridge and natural forces. And as a consequence of this, its highest values are about one tenth of 2Hours.

The highest acceleration values are found in channel 12 for both time series. For 2Hours the values is $2.45 m/s^2$, while for 20Min the maximum value is $0.203 m/s^2$

In figure 5.7 a close view of the peak at about 4500s from 2Hours is shown. This shows a typical acceleration section of the bridge during excitation. The closeness of the two neighbouring channels in movement, is a good indication that the GPS timestamp synchronization has been completed successfully.

Figure 5.7 shows a typical response from the middle of the bridge during deliberate excitation, i.e. participants of the experiment jumping on the middle span.

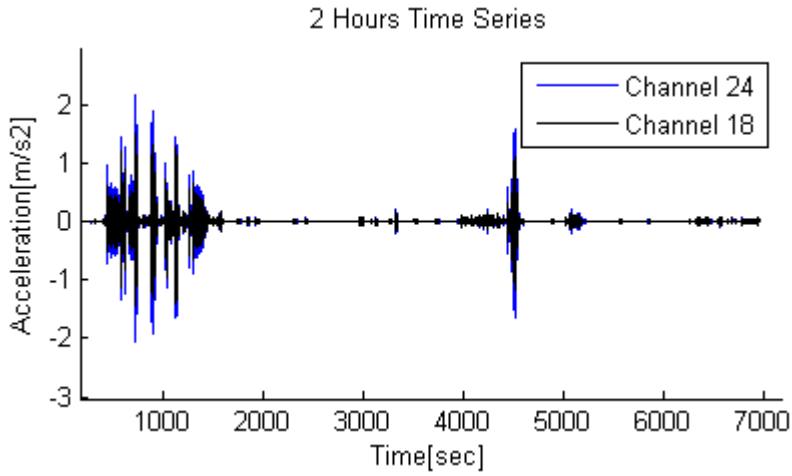


Figure 5.5: 2Hours entire time series

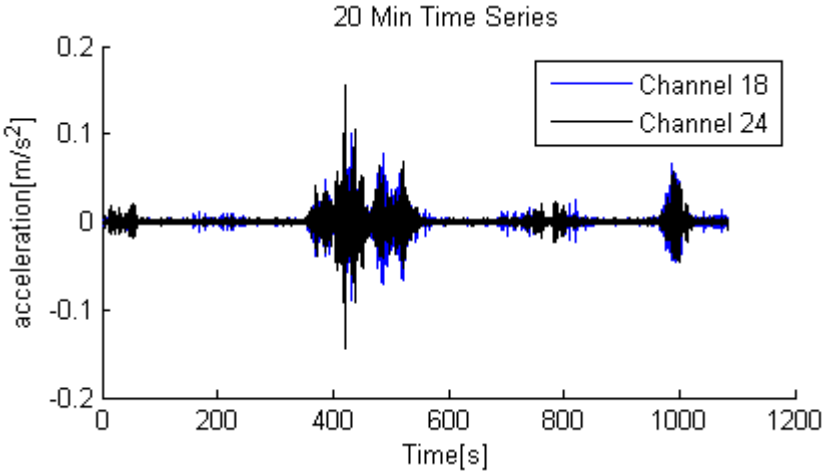


Figure 5.6: 20Min entire time series

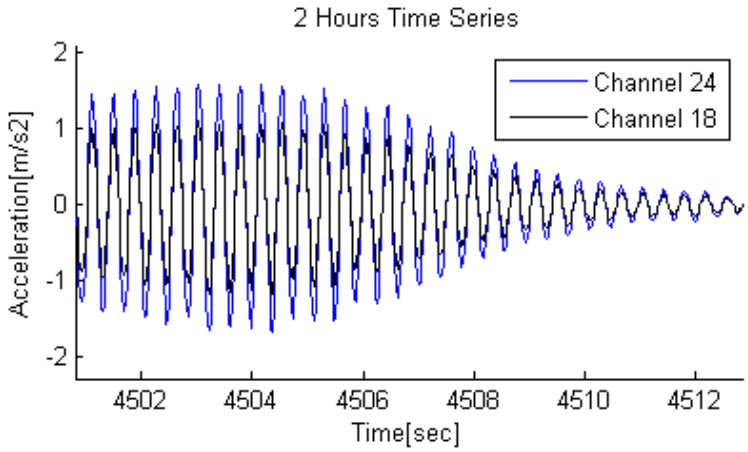


Figure 5.7: 2 Hours time seires close view

5.4.2 Auto Power Spectral Density

The auto power spectral densities(APSD) shown in figure 5.8 and 5.9 are Welch's one-sided averages made with 8 Hamming windows with no overlap. This is done using the cpsd[21] command in Matlab.

In these figures all the channels in the z-direction are present.

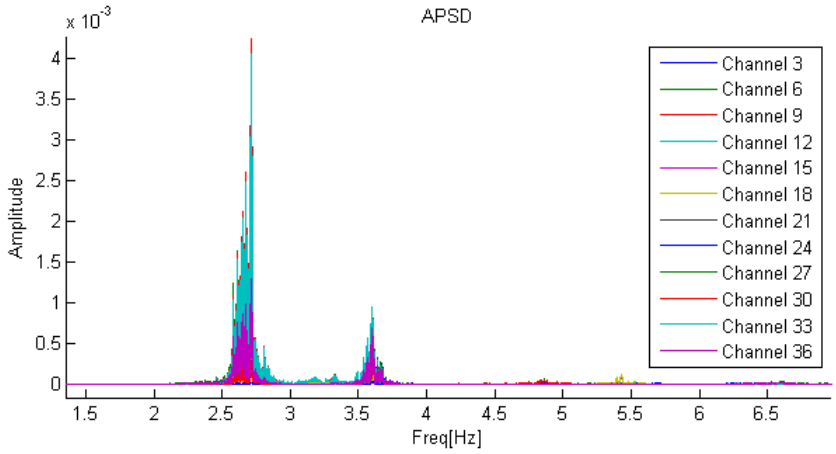


Figure 5.8: APSD of 2Hours

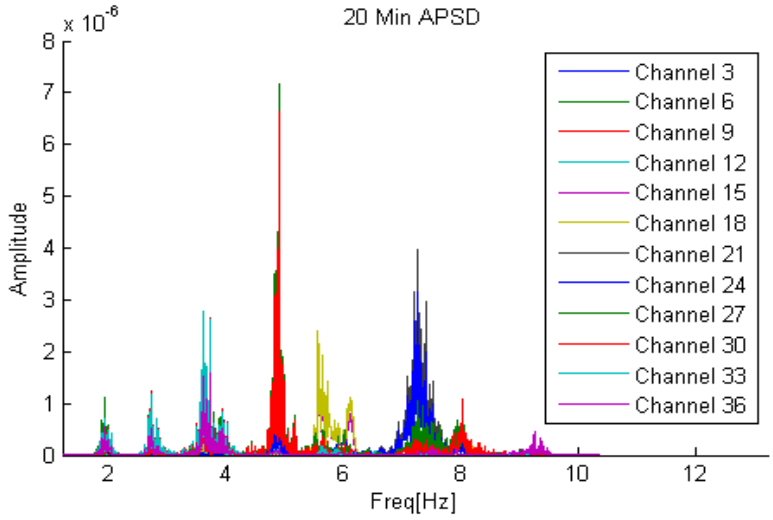


Figure 5.9: APSD of 20Min

The 2Hour APSD is dominated by two frequencies at about 2.7 Hz and 3.7 Hz. These are likely to be the natural frequencies excited by jumping. In the 20Min APSD there are

other frequencies that dominate. Although here, the maximum values of the amplitudes are substantially smaller, at about one thousandth of the peak values of 2Hours.

In figure 5.10 a magnified view of the 2Hour series reveals that some of the same frequencies are found here as well, but with very small values compared to the dominating frequencies.

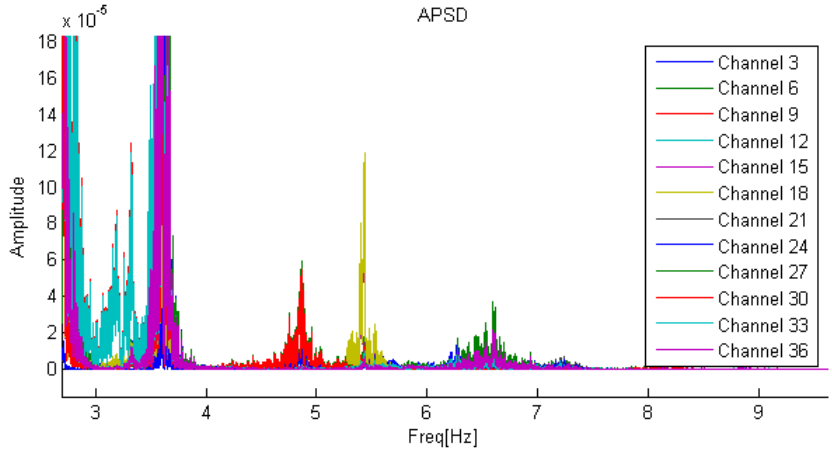
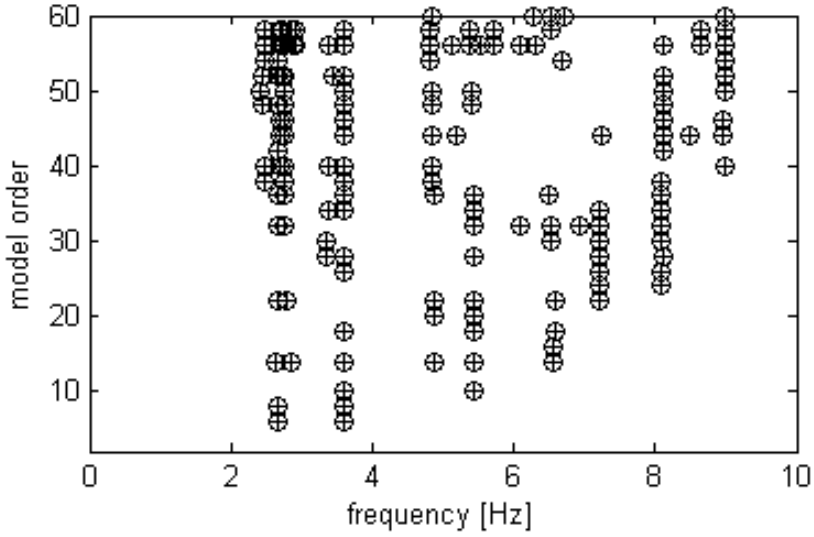


Figure 5.10: A closer look on the APSD of 2Hours

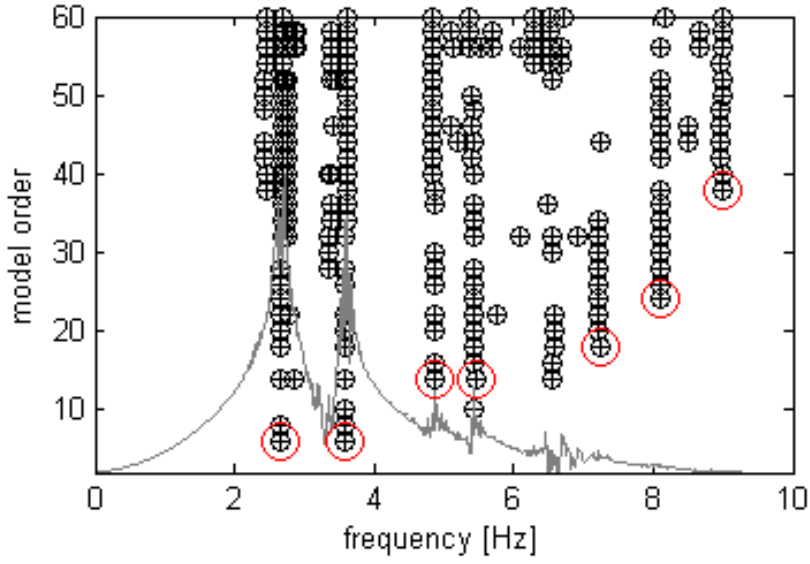
5.4.3 Stabilization Diagram

Below, in figure 5.11a and 5.11b, are two stabilization diagrams of the 2 hours time series. The first diagram is with normal criteria for frequency and dampening change over system order, i.e. 1 and 5%. These criteria are expressed in (2.76) and (2.77). However since there are many unclear lines in this diagram the criteria were loosened to 2 and 10% for the second diagram. Modal analyses such as these are known for large uncertainties, especially in the dampening estimates, and the reduction in strictness is done only to more clearly show the lines. In this second figure the power spectral density as calculated by Macec is also shown. There are clear peaks in the PSD for the early modes, but for the modes in the higher frequency range the response is more or less non existing. The modes that are extracted for further use are marked by red rings. In these diagrams only the stable modes are shown.

The same procedure with the same criteria is shown for 20Min in 5.12. Here the PSD show more clear peaks and the peaks line up better with the modes with higher frequencies. There are many of the same stable lines as in 2Hours up till about 6 Hz, however in the higher frequency range the modes differ more in the different time series.

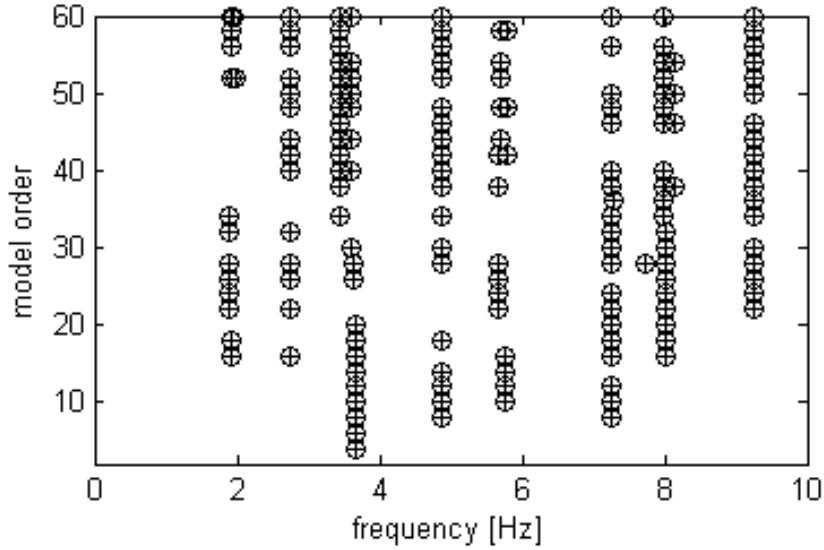


(a) Stabilization diagram of 2Hours with strict conditions

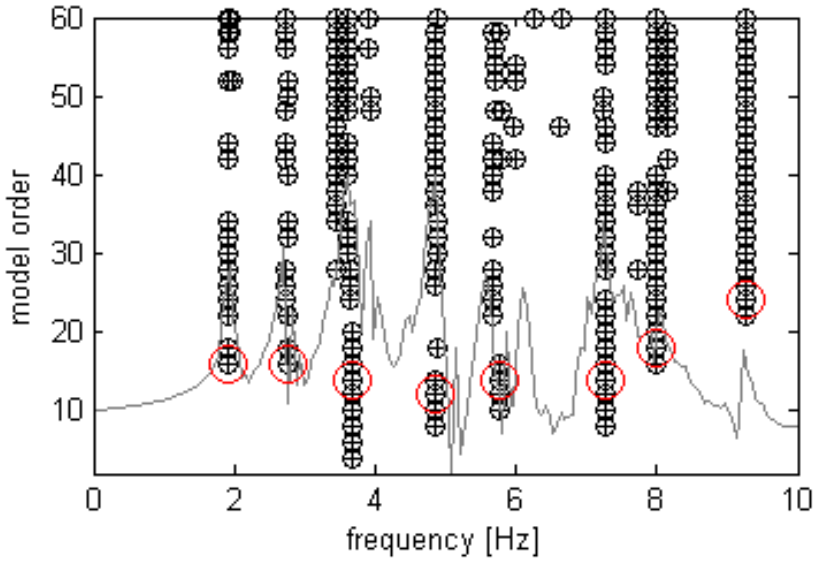


(b) Stabilization diagram of 2Hours with loose conditions

Figure 5.11: Stabilization diagrams of 2Hours



(a) Stabilization diagram of 20Min with strict conditions



(b) Stabilization diagram of 20Min with loose conditions

Figure 5.12: Stabilization diagrams of 20Min

5.4.4 Natural Frequencies and Dampening ratios

The natural frequencies ω_n and the dampening coefficients were extracted from the stabilization diagram. These are only approximate values as they have differences also in the stabilization diagram. The different series have many of the same natural frequency values in the low range of the frequency specter, but as the frequency increases the values become further apart. The 20 minutes series also have one value at 1.85 Hz before they both have a mutual natural frequency at 2.6 Hz.

The dampening ratios ζ are quite uncertain in analysis like these. Therefore the range of dampening is shown for each mode. Some of the modes differ little while some have quite wide spans. The dampening from the different time series have some correlation.

The natural frequencies and dampening ratios from 2Hours are:

Mode	ω_n [Hz]	ζ [%]
1	2.6	1.0 - 1,4
2	3.6	0.5 - 0.8
3	4.85	1.6 - 2.1
4	5.4	0.6 - 0.8
5	7.2	1.8 - 1.9
6	8.1	1.8 - 2.1
7	8.95	2.6 - 3.3

The natural frequencies and dampening ratios from 20Min are:

Mode	ω_n [Hz]	ζ [%]
1	1.9	3.2 - 4.2
2	2.7	0.6 - 0.9
3	3.6	3.1 - 3.8
4	4.85	1.0 - 1.3
5	5.6	0.8 - 3.8
6	7.25	1.6 - 2.2
7	8.00	2.0 - 2.1
8	9.2	1.2 - 1.3

5.4.5 Mode Shapes

The mode shapes plots are also extracted from Macec. Since 20Min has one natural frequency without any clear counterpart in 2Hours this is plotted alone. The other values are plotted in pairs for comparison. These figures should be considered with the help of the sensor set up in 5.2 to understand how the bridge moves. The ends of the bridge are not plotted as there were no sensors at these locations. They movement of these can however be assumed to be more or less zero. The bridge also has 4 supports that are not indicated on the following figures. Because of the limited number of sensors it is possible that the mode shapes are missing one or several orders of movement. They are all shown here regardless for later discussion.

As seen from the below figures the mode shapes from the different series are very similar to each other. The only significant difference is in the comparison of 2Hours mode 5 with 20Min mode 6. Here the mode shape from 2Hours appears to be a torsional mode while the 20Min mode is mostly vertical. Also notable is that the first two modes of 20Min are almost exactly alike.

The early modes are pure vertical modes. This applies to all modes up to number 4 for 2Hours, and mode 6 for 20Min. In 2Hours the modes number 5 and 6 are torsional, while for 20Min there is only one torsional mode, which is number 7. In all the 3 torsional modes identified, the movement is mostly centered around the southern end of the bridge. The last mode pair, at about 9.1 Hz, shows a new vertical modes. In this last pair, it is noteworthy that the highest excited nodes are the ones in the quarter point of the bridge. As this is the only pair not in the middle of their respective spans, this might be an indication of a mode of an order which this sensor setup does not completely capture.

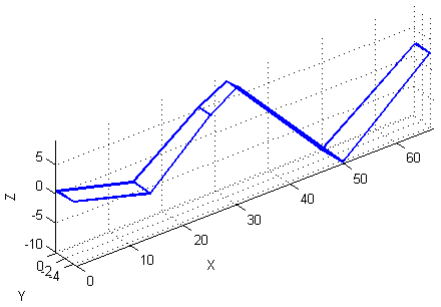
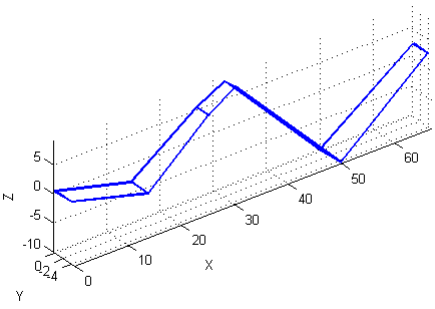
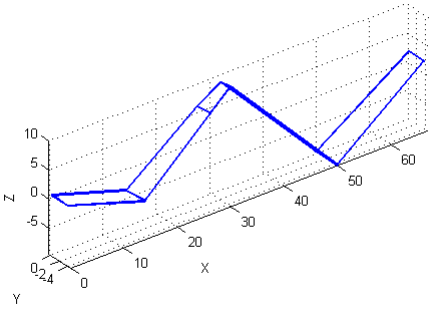


Figure 5.13: 20Min Mode 1

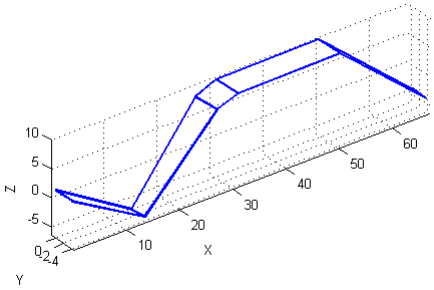


(a) 2Hours Mode 1

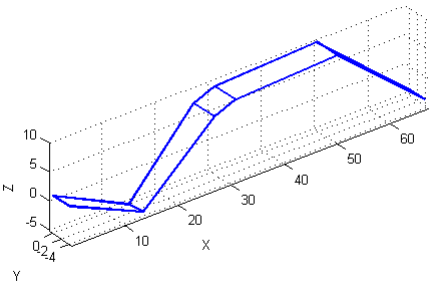


(b) 20Min Mode 2

Figure 5.14: Comparison of mode shape at 2.6 - 2.7 Hz

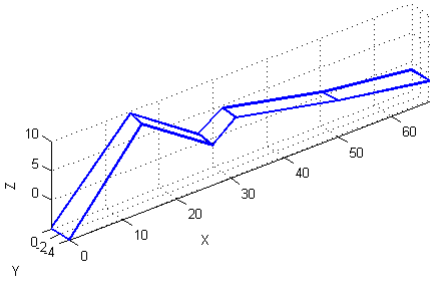


(a) 2Hours Mode 2

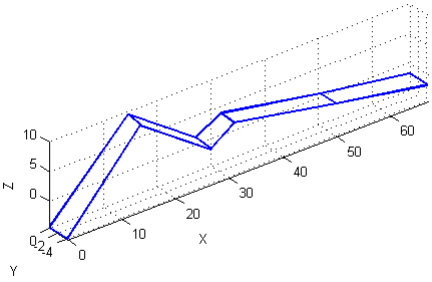


(b) 20Min Mode 3

Figure 5.15: Comparison of mode shape at 3.6 Hz

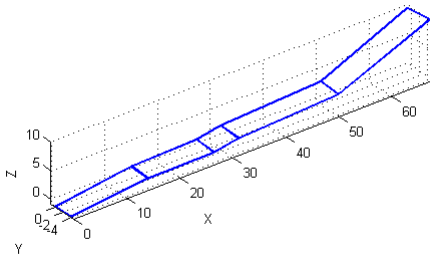


(a) 2Hours Mode 3

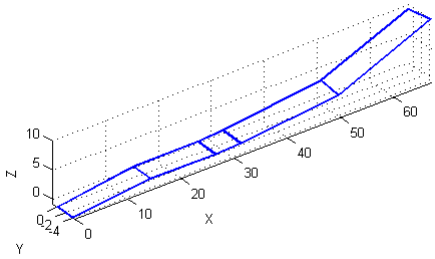


(b) 20Min Mode 4

Figure 5.16: Comparison of mode shape at 4.85

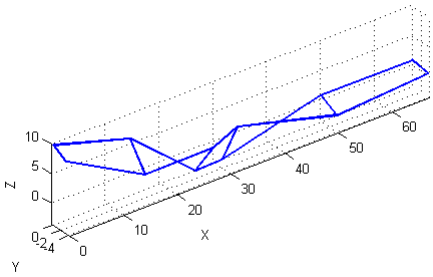


(a) 2Hours Mode 4

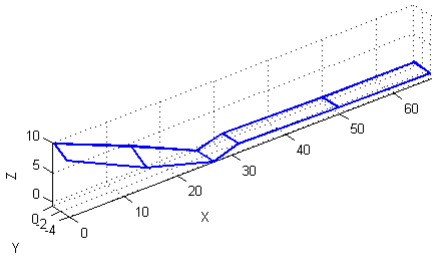


(b) 20Min Mode 5

Figure 5.17: Comparison of mode shape 5.4 - 5.6

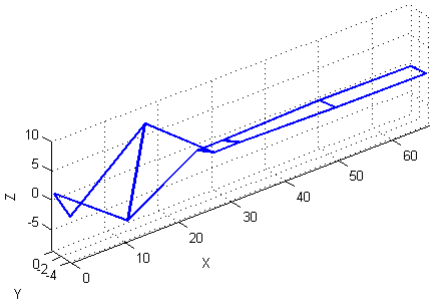


(a) 2Hours Mode 5

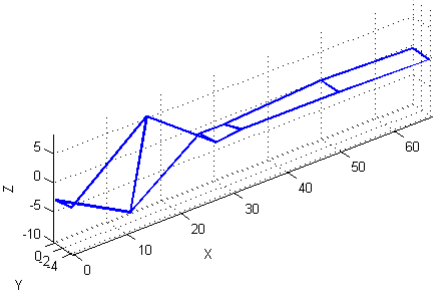


(b) 20Min Mode 6

Figure 5.18: Comparison of mode shape 7.2 - 7.25

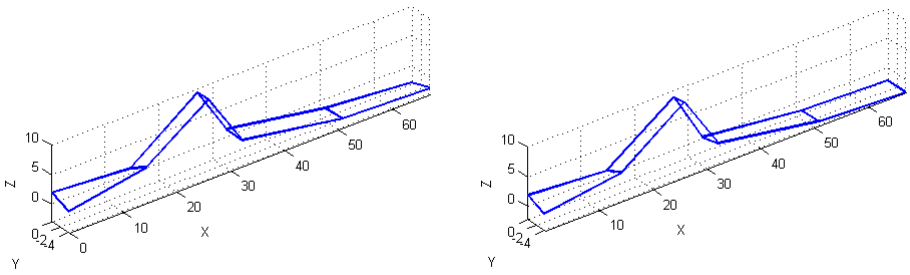


(a) 2Hours Mode 6



(b) 20Min Mode 7

Figure 5.19: Comparison of mode shape 8.0 - 8.1



(a) 2Hours Mode 7 **(b)** 20Min Mode 8

Figure 5.20: Comparison of mode shape at 9.0 - 9.2

5.5 Discussion

In this section some thoughts about the results and their accuracy will be presented.

The different time series are both deemed to be accurate. Both the magnitude of the acceleration and the general movement of both 2Hours and 20Min seem very plausible. The synchronization of the different cRIOs through the GPS time stamps also seems successful, meaning the written software has functioned as it should. This is important for further use.

The auto spectral densities from the two series are very different. The 2Hours series is dominated by the two frequencies at 2.6 and 3.6, which are likely to be the natural modes set in movement by deliberate excitation. 20Min seems to be more dominated by noise and natural forces, as can be seen both from the amplitudes of the acceleration series and from the more equally distributed amplitudes of the PSD. There are certain advantages to both. By deliberate excitation it is quite certain that the modes found here are real. However, as the noise input of 20Min is a lot closer to the white noise input assumption used by the Cov-SSI method, these results might create a more mathematically accurate model if the noise was powerful enough to excite the bridge into actual modes of movements. Either way the peaks of both series are similar in frequency, although 20Min have more clear peaks.

The stabilization diagrams are not as clear as could be hoped. Especially the 2Hours diagram with strict conditions displayed in 5.11a is weak. The supposed lines have large sections of unstable modes and cannot be clearly seen until figure 5.11b where the conditions are loosened. Here the lines become clearer and they can be seen aligning with the peaks from the PSD. This is however only clear for the four first modes, in the three last ones there are no visible peaks.

The diagram from 20Min in figure 5.12b is better, the lines are clearer and there are fewer stable modes that does not match any clear line. In figure 5.12 the lines become even clearer and matches the peaks of the PSD to a much grater extent than the 2Hours. These clear peaks might again be due to the force input in 20Min being closer to the white noise input assumption, which theoretically excites all modes equally. Based on the stabilization diagrams, the 20 minutes time series appears to return more reliable results especially in the higher frequency modes.

The natural frequencies are, despite somewhat different stabilization diagrams, quite similar. Almost all off them can be easily paired without having huge variations in frequency. There are some differences, but this is to be expected. The biggest difference is the first natural frequency in 20Min, at about 1.9 Hz, which has no obvious counterpart in 2Hours. As the mode shape figures show the two first modes of 20Min have the same mode shape, which is equal to the first in 2Hours.

The dampening estimates of analyses like these always have a high degree of uncertainty. It is however normal for large slender structures like Bratsberg foot bridge to have damp-

ening coefficients around 1-5%. With this in mind the estimates seem plausible. The estimates vary somewhat between the different series, so that it is hard to be sure of an exact estimate.

The mode shapes are similar for all but one pair of the natural frequencies. This is a good indication that the captured frequencies are real. However, because of few sensors placed on the bridge during measurements there is no guarantee that the mode shapes are correct. There might be several movements of the bridge that are not captured, and in the same way that aliasing of a signal shows signal frequencies that are not there, the bridge's movement is aliased to appear to move in ways it does not. The modes are still included as a way of comparing the different series and to possibly separate vertical modes from torsional.

The first mode shape found in both series, or the two first in 20Min, seem correct. This is how the bridge's first mode shape is expected to be, with every bridge span being out of phase with the last while the supports are more or less not moving. The first mode is usually the easiest to excite and it is therefore likely that the actual mode is the one found in 2Hours by deliberate excitement of the bridge. The dampening estimate of 1.0 - 1.4, also seems very likely for this system. And with a lot of free vibration in this mode it is likely that the model can represent the dampening fairly good. Why 20Min has found two similar first modes is unknown, but these ones seem less likely as there is zero sign of a natural frequency at 1.9 Hz in 2Hours.

The mode at 3.6 is also considered to be real, as it is clearly shown in 2Hours. The mode shape is however not. This is a good example of a mode shape where some information is missing. This mode shape would require the third support (at about 40 in the x direction in the mode shape plots) to move in phase and amplitude with the two spans on each side. This is unlikely, and it is more probable that the mode shape should go down to about zero at the support and back up to the next sensor. This would also be a very likely second mode shape.

The rest of the mode shapes will not be commented on. It should have been possible to find the first torsional mode with the information provided by the sensor locations. This might be the 2Hours' mode 5 but it is hard to tell because the 20Min did not capture this. The mode at about 8.0 Hz is probably a torsional mode, but there might be missing information here as well. It is however considered a good sign for accuracy that the mode shapes in general are similar.

Chapter 6

Final Remarks

All in all the written software and its use is a success. The programs are functioning as planned, all the different elements are working properly and it is fairly easy to use. The software should provide the faculty and future projects with an easy setup for measuring large structures when several Compact RIOs are necessary.

The field test is also deemed successful. Even tough the the results from the different time series vary a little, the early modes are believed to be identified correctly. But most importantly, the software worked perfectly, also in the field.

In the future it would be interesting to see the software deployed on a greater scale. Possibly a large bridge with many compact RIOs deployed in parallel. Perhaps with a functioning anemometer to measure force input. The need to measure large structures is unlikely to disappear in the future, and this software can perhaps especially be of help in smaller, more short-time projects where a shorter setup time is a priority.

References

[1] Henri P.Gavin, "Classical Damping, Non-Classical Damping and Complex Modes"

[2] Carlo Rainieri and Giovanni Fabbrocino. Operational modal analysis of civil engi-neering structures. Technical report, Springer, 2014.

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

[4] https://en.wikipedia.org/wiki/Discrete_Fourier_transform

[5] https://en.wikipedia.org/wiki/Fourier_transform#Discrete_Fourier_Transforms_and_Fast_Fourier_Transforms

[6] Einar Strmmen. Theory of bridge aerodynamics. Springer Science & Business Media, 2010

[7] https://en.wikipedia.org/wiki/Fast_Fourier_transform

[8] Sindre Aavik Schanke. Master Thesis 2015. 'Operational Modal Analysis of Large Bridges'

[9] Olsen and Hansen, Master Thesis 2016 'Measuring vibrations and assessing dynamic properties of tall timber buildings'

[10] Roger G. T. Mello, Jurandir Nadal, and Liliam F. Oliveira. Digital Butterworth filter for subtracting noise from low magnitude surface electromyogram. Computer Methods and Programs in Biomedicine, 87(1):2835, 2007.

[11] https://zone.ni.com/reference/en-XX/help/371361J-01/lvanlsconcepts/lvac_butterworth_filters/

[12] http://www.ni.com/compactrio/whatis/

[13] https://en.wikipedia.org/wiki/CompactRIO

[14] https://www.kistler.com/?type=669&fid=53192

[15] http://www.ni.com/pdf/manuals/374238a_02.pdf

[16] http://www.ni.com/datasheet/pdf/en/ds-537

[17] http://zone.ni.com/reference/en-XX/help/371599G-01/lvfpga/rational_resmpler_fltr/

[18] https://www.timeanddate.com/time/aboututc.html

[19] https://www.kistler.com/?type=669&fid=56604

[20] https://se.mathworks.com/help/signal/ref/decimate.html

[21] https://se.mathworks.com/help/signal/ref/cpsd.html

[22] https://en.wikipedia.org/wiki/Fourier_transform