Design of a fully automated identification tool for the Nyquist autotuner

R.H.M. Solberg

DCT 2005.109

Traineeship committee:
Dr.Ir. M.J.G. van de Molengraft *
Ir. H. Uijen **

* Eindhoven University of Technology
  Department of Mechanical Engineering
  Control Systems Technology

** Nyquist Industrial Control Eindhoven
  Department Development and Engineering

Eindhoven, July 2005
Preface

This report contains the results of a four month traineeship which has been carried out at the Development & Engineering department at Nyquist Industrial Control in Eindhoven, the Netherlands. I would like to thank all the members of the Development & Engineering as well as the Software Engineering department. Especially Huub Uijen, Geert van der Zalm, Istvan Eperjesy and Cedric Schmeits for their support during this traineeship. I also would like to thank René van de Molengraft for his advice during our discussions.
Abstract

The traineeship of four months has been carried out at a company named Nyquist Industrial Control. Nyquist is a company which is specialized in the development and sale of operating systems for the OEM machine builders market. The systems of Nyquist stand out by high flexibility and accuracy. On the one hand this is achieved by the own development of qualitative and innovative systems and on the other hand by the expertise of the own motion engineers.

One of the developments at Nyquist is automated tuning of motion systems (PID and Feedforward). The "autotuner" eventually has to be able to tune a motion system totally by itself, according to some wishes of the user. With the application of automated tuning the expertise of a motion engineer is, for relative standard systems, no longer necessary. An additional advantage is the ability to re-tune a system, quickly and easy. Often after a system has been in operation for a while, it will have undergone some wear which affects the performance. In this case a motion engineer is also no longer necessary. Finally the autotuner will strongly reduce the costs.

The original assignment of this traineeship was to look at improvements for this algorithm and to integrate them to a new, improved autotuner. By experimenting with the tuner and out of interviews with motion engineers and people of product management, insight has been gathered for what improvements are desirable.

One of the improvements turned out to be machine-identification. The autotuner makes use of the transfer function of the system and therefore it is of great importance that the transfer is determined as good as possible. To provide enough time to design and develop this as completely as possible the assignment was altered. The assignment was revised to: shorten and improve the identification procedure and make it more machine friendly. The goal was to create a universal, automatic identification module that can determine the transfer of a system. This transfer could subsequently be used by the autotuner.

The original identification procedure of the autotuner was done by direct (closed-loop) input-output measurements with the use of one by one excited sine waves, each of a different frequency. This appeared to be a time-consuming, machine unfriendly and disturbance sensitive process. During the measurements also no movement was enforced on the system and as a result of that the machine was continuously pushed through its friction during the excitations. The finally obtained transfer contained bias because of the influence of external unwanted noise in the system. It contained also influences of the non-linear effects because of the friction behaviour.

It has been demonstrated that by determining the Sensitivity and Proces Sensitivity (by excitation with a broad banded signal) the deviations, caused by the unwanted noise, do not appear in the transfer of the system. By also applying a movement on the system during the measurement some non-linear effects can be reduced. To obtain a basis for further investigation, first the transfers of the experimental setups have been derived. The first measurements have been done by using a TU/e DACS AQI data-acquisition system and Matlab. Subsequently measurements with the Nyquist hardware have been done to prove that the motion control systems of Nyquist are capable of producing a similar result.
The analysis of the measurements first has been done with Matlab. However the module had to become a stand-alone program, so the use of Matlab was no longer possible. Furthermore a machine builder does not want to purchase a Matlab license for every product. This demanded a stand alone algorithm. Therefore an investigation on spectral analysis and Fast Fourier Transformation has been carried out. With the obtained knowledge a spectral analysis algorithm has been derived and this has eventually been setup in C-code.

Before a universal identification measurement could be done, a few parameters still needed to be derived. Namely the polarity, the reference movement, the (low-bandwidth) controller and the amplitude of the excitation signal. The polarity can be derived out of a breakaway experiment. Herewith an increasing voltage is set on the system until a desired stroke is reached. Two different reference movements may be chosen, namely a JOG mode or a repeating movement. The JOG mode may be used for free axis and the repeating movement for limited stages. For the controller an attempt has been done to derive a mass-estimation method. Via the estimated mass a controller with a desired bandwidth could then be calculated. However, experimental measurements showed bad universal applicability due to (position depending) frictions in the experimental setups. In the end a controller optimization algorithm has been integrated. Herewith the controller is increased until a desired tracking, of the enforced movement, has been obtained. For the excitation signal random noise is used for now, because this is currently the best performing excitation signal in the Nyquist hardware. The level for this is determined out of the intensity of the closed-loop control signal during the enforced movement. Limited stages are also protected against driving out of a specific range. This is done by monitoring the stroke of the stage. The stage will be stopped if the allowed positions are exceeded.

To give insight in the obtained identification procedure, all above-mentioned parts finally have been combined to a demonstration program named NYCEIdent. The program has been written in Visual Basic (VB) and C, in the course of which the graphical interface and the standard functions are operated in VB and the complex calculations are executed in a C-program. To prove the working of the procedure the experimental setups have been identified with NYCEIdent. The obtained transfers have been measured very silently and showed a successful result. The final identification procedure takes now approximately 150 seconds.
Samenvatting

De stage van vier maanden is extern uitgevoerd bij een bedrijf genaamd Nyquist Industrial Control. Nyquist is een bedrijf dat is gespecialiseerd in het ontwikkelen en verkopen van besturingssystemen voor de OEM machinebouwers markt. De systemen van Nyquist onderscheiden zich door een hoge flexibiliteit en nauwkeurigheid. Dit wordt enerzijds bereikt door de eigen ontwikkeling van kwalitatieve en innovatieve systemen en anderzijds door de expertise van de eigen motion engineers.

Een van de ontwikkelingen bij Nyquist is het geautomatiseerd tunen van motion systemen (PID en feedforward). De "autotuner" zou uiteindelijk een motion systeem, afhankelijk van bepaalde gebruikerswensen, geheel zelfstandig moeten kunnen afregelen. Door de toepassing van geautomatiseerd tunen is, bij relatief eenvoudige systemen, de expertise van een motion engineer niet meer nodig. Een bijkomend voordeel is de mogelijkheid een systeem snel en gemakkelijk opnieuw af te regelen. Want indien een machine een tijd lang in bedrijf is ondergaat deze vaak enige slijtage die de prestatie beïnvloed. Hierbij is dan wederom geen motion engineer meer nodig. Dit zal uiteindelijk de kosten sterk reduceren.

In het verleden is bij Nyquist reeds gewerkt aan een autotuner algoritme. De oorspronkelijke opdracht van deze stage was te kijken naar verbeteringen voor dit algoritme en deze te integreren tot een nieuwe, verbeterde autotuner. Door te experimenteren met de tuner en uit gesprekken met motion engineers en mensen van product management is inzicht verworven in welke optimalisaties wenselijk zijn.

Een van de gevonden optimalisatiepunten bleek machine-identificatie. De autotuner maakt gebruik van de overdrachtsfunctie van het systeem en het is dus van groot belang dat de overdracht zo goed mogelijk wordt bepaald. Om genoeg tijd te voorzien om dit zo volledig mogelijk te kunnen ontwerpen en uitwerken is de stageopdracht aangepast. De opdracht in aangepast tot : het verkorten, verbeteren en vriendelijker maken van de identificatieprocedure. Het doel was het creëren van een universele, automatische identificatie module die de overdracht van een systeem kan bepalen. De overdracht zou vervolgens door de autotuner kunnen worden gebruikt.


Er is aangetoond dat door het bepalen van de Sensitivity en Proces Sensitivity (via excitatie door een breedbandig signaal) de afwijkingen, door de ongewenste externe verstoringen, niet meer terugkomen in de systeemoverdracht. Door tevens het systeem te laten bewegen gedurende de meting kunnen enkele niet lineaire effecten worden gereduceerd. Om een basis te verkrijgen voor verder onderzoek zijn eerst de overdrachten van de experimentele opstellingen bepaald. De eerste metingen zijn gedaan
met een TU/e DACS AQI data-acquisitiesysteem en Matlab. Vervolgens zijn er metingen gedaan via de Nyquist hardware om aan te tonen dat de motion control systemen van Nyquist in staat zijn een gelijkend resultaat te produceren.

De analyse van de metingen is eerst gedaan met behulp van Matlab. Echter omdat de module een stand alone programma moest worden was het gebruik van Matlab niet meer mogelijk. Daarnaast zal een machinebouwer ook zeker geen Matlab-licentie willen aanschaffen voor elk product. Dit vereiste dan een stand alone algoritme. Daarom is er onderzoek gedaan naar spectrale analyse en Fast Fourier transformatie. Met de opgedane kennis is een spectrale analyse algoritme afgeleid en dit is uiteindelijk uitgewerkt in C-code.

Voordat een universele identificatiemeting kon worden gedaan, moesten er eerst nog een aantal parameters bepaald worden. Namelijk de polariteit, de referentie beweging, de (laagbandbreedte) regelaar en de amplitude van het excitatiesignaal. De polariteit kan worden bepaald uit een Breakaway experiment. Hierbij wordt er een toenemend voltage op het systeem gezet totdat een gewenste uit- slag is bereikt. Er kunnen twee verschillende referentie bewegingen worden gekozen, namelijk een JOG mode of een repeterende beweging. De JOG mode kan worden gebruikt voor vrije assen en de repeterende beweging voor gelimiteerde stages. Voor de regelaar is getracht een massaschattingsmethode af te leiden. Via de geschatte massa zou dan een regelaar met een gewenste bandbreedte kunnen worden berekend. Echter experimentele metingen hebben een slechte universele toepasbaarheid laten zien, vanwege (plaatsafhankelijke) wrijvingen in de experimentele opstellingen. Uiteindelijk is een regelaar optimalisatietool algoritme geïntegreerd. Hierbij wordt de regelaar zo lang verhoogt tot dat er een gewenst volggedrag, van de opgegeven beweging, is bereikt. Voor het excitatiesignaal is voorlopig uitgegaan van random ruis, omdat dit momenteel het best presterende excitatiesignaal is in de Nyquist hardware. Het niveau hiervoor wordt bepaald uit de intensiteit van het closed-loop regelsignaal gedurende de referentiebeweging. Gelimiteerde stages worden ook geschermd tegen het wegschieten uit een bepaald gebied. Dit wordt gedaan door de slag van de stage te volgen. De stage zal worden gestopt zodra de toelaatbare posities worden overschreden.

Om inzicht te geven in de verkregen procedure, zijn alle bovengenoemde onderdelen uiteindelijk samengevoegd tot een programma genaamd NYCeIdent. Het programma is geschreven in Visual Basic (VB) en C, waarbij de grafische interface en de standaardfuncties worden bestuurd in VB en de complexe berekeningen worden uitgevoerd in een C-programma. Om de werking van de procedure te bewijzen zijn de experimentele opstellingen geïdentificeerd met NYCeIdent. De verkregen overdrachten zijn zeer stil gemeten en hebben een succesvol resultaat laten zien. De uiteindelijke identificatieprocedure neemt ongeveer 150 seconden in beslag.
Contents

Preface ii
Abstract iii
Samenvatting v

1 Introduction 2
  1.1 Motivation 2
  1.2 Problem statement 2
  1.3 Literature overview 3
  1.4 Outline of the report 3

2 Improvements for Autotuning 5
  2.1 Introduction 5
  2.2 Autotuning 5
    2.2.1 "Smart" autotuner 5
    2.2.2 Experimentation 5
    2.2.3 Interviews 6
    2.2.4 List of improvements 7
  2.3 Experimental setups 7
    2.3.1 Fourth order system 7
    2.3.2 Philips V.R.S. system 8
  2.4 Discussion 8

3 System Identification 9
  3.1 Introduction 9
  3.2 Identification 9
    3.2.1 Direct measurement 10
    3.2.2 Sensitivity measurement 10
    3.2.3 Process sensitivity 11
  3.3 System transfers 12
3.4 Nyquist hardware .................................................. 13
  3.4.1 Testsignal generation ........................................ 14
  3.4.2 Tracing & Signal processing ................................. 15
3.5 Spectral analysis .................................................. 17
  3.5.1 analysis items .................................................. 17
  3.5.2 Algorithm ...................................................... 22
3.6 Discussion ......................................................... 22

4 Identification parameters ........................................ 24
  4.1 Introduction ...................................................... 24
  4.2 Allowable stroke ................................................ 24
  4.3 Polarity ........................................................... 24
  4.4 Reference Movement .......................................... 25
  4.5 Controller ......................................................... 25
  4.6 Excitation signal power ....................................... 28
  4.7 Discussion ......................................................... 29

5 Demonstration Program ........................................... 30
  5.1 Introduction ...................................................... 30
  5.2 Programming ..................................................... 30
  5.3 Testing ........................................................... 32
  5.4 Discussion ......................................................... 33

6 Conclusions & recommendations ............................... 34
  6.1 Conclusions ....................................................... 34
  6.2 Recommendations ............................................... 35

A Skew Notchfilter .................................................. 36

B Spectrum analysis source code ................................... 37
  B.1 Spectral analysis code ......................................... 37
  B.2 Crosspower calculation code ................................. 42
  B.3 Fourier transformation algorithm code ...................... 42

C Testing NYCeldent .................................................. 45
  C.1 Fourth order system ........................................... 45
  C.2 V.R.S. system .................................................... 46

Bibliography ......................................................... 47
Chapter 1

Introduction

1.1 Motivation

The technological progress over the last two decades has caused an enormous improvement in motion control. This has lead to an increase in accuracy and efficiency of machines. However, the level of technology in a machine increases, its price has to decrease. So to stay competitive in the motion control market the control systems have to be more and more sophisticated but at an always lower purchase price. Next to the purchase price, integration costs are becoming more and more important. One of the possibilities to reduce the integration costs is to design the product in a way that it is very universally applicable and so “ready-for-use” as possible. At Nyquist all the products are already for 80 percent ready-for-use (time-to-market) and only 20 percent has still to be designed specific for the customer (design in). This is mostly the work of the motion engineers. So another possibility to lower the integration costs is to reduce the part for these engineers.

One of the ideas at Nyquist was to automate the tuning of motion systems. With the ability to automatically tune a system many tuning hours, done by the engineer, can be saved. Also the retuning of a system after an amount of operation hours does not require an expert anymore. So with an autotuner the costs of integration as of maintenance can be reduced.

In the past, Nyquist has already developed an autotuning method. This method is able to calculate a controller with the highest possible bandwidth. The basis for the calculation is the transfer function of the system which has been derived out of identification measurements. The method however does not always work and it contains only one specific way of tuning. Nyquist requested an investigation for possible improvements and other tuning methods to optimize the autotuning approach.

In this report first the result of the improvement-investigation will be presented. This will lead to the final assignment for this traineeship, namely system-identification. The rest of the report therefore will be about the design and development of a machine identification procedure for the final autotuner. The next section presents the two problem statements. After that the literature overview is presented, followed by the outline of this report.

1.2 Problem statement

Recently at Nyquist an autotuning-algorithm has been designed. This algorithm is not optimal yet and it needs to be optimized. An investigation should be done on improvements for the now used
algorithm and possible expansions for it. The findings should eventually be implemented in the old algorithm. Possibilities are identification, tuning strategy and Feedforward tuning. This leads to the following problem statement:

Obtain the desired improvements for the autotuner, investigate them and implement the solutions to make a new optimized autotuner algorithm.

One of the improvements turned out to be machine-identification. The autotuner uses the transfer function to calculate the controller. The better the identification is, the better the algorithm will eventually work. To be able to investigate this problem thoroughly the assignment has to be revised. The final goal is to create a sufficient universal machine identification procedure. To demonstrate the results the procedure should be translated into a demonstration program. The procedure should also be able to work together with the autotuner algorithm. The new problem statement goes:

Design and develop an identification procedure which can universally determine the transfer of a motion system. Make a demonstration program of the obtained procedure to present the new identification. The final procedure should also be suitable for implementation in the new autotuner-algorithm.

1.3 Literature overview

During this traineeship several Literature sources have been consulted. To obtain insight in the already developed autotuner the report [3] of two graduate students has been studied. They developed the first complete autotuner at Nyquist. A lot of information about motion control and mechanical systems can be found in [4]. This book has been used as a sort of reference book for evaluating ideas. Various ways for identification measurements are fully described in [13] and [7]. The report of Johan Boot [2] gives a very nice, brief overview about the (mathematical) differences between the direct measurement and Sensitivity measurement of a system.

The course notes of Signal Analysis [6] form a good basis for the spectral analysis of the identification measurements. The final spectral analysis has been written according to Welch’s method. This method is very similar to the method used in Matlab. To compare the methods also the spectral analysis of Matlab has been consulted [8]. The course notes of Digital Motion Control [13] also present a little information about experiments for obtaining measurement parameters. The paper of R. Rijs [10] gives a good explanation about breakaway experiments.

To be able to write the final demonstration program several books on programming offer a lot of help. The most knowledge on C can be found in [5]. Because the program has been written in visual Studio also the books about Visual Basic [12] and C++ [1] turned out to be very useful. The program talks with the Nyquist motion controller. The information about the abilities of this controller and the available programming library can be found in [9].

1.4 Outline of the report

Chapter 2 presents the first investigation about the improvements for the autotuner. Out of experimentation with the tuner and out of interviews with people of Nyquist a list of improvements will be set up. Next the experimental setups will be shown on which the experiments for this traineeship will be done.

In Chapter 3 the investigation for the new identification method will be explained. A comparison
will be made between the direct measurement and Sensitivity measurement. Also the Nyquist hard-
ware will be compared with another data-acquisition system. Finally, a stand-alone spectral analysis
program will be derived.

For doing an identification measurement a few other parameters have to be known. Namely the
investigation of the polarity, enforced movements, controller determination and the scaling of the ex-
citation signal. This will be presented in Chapter 4.

All of the above acquired knowledge has to be combined. The resulted demonstration program will
be treated in Chapter 5. With the program several tests will be done, which will also be presented in
this chapter. Based on results, conclusions and recommendations can be made, which will be given
in Chapter 6.
Chapter 2

Improvements for Autotuning

2.1 Introduction

This chapter presents the autotuner and the investigation on improvements for it. Out of own experimentations and interviews a list of improvements will be set up. Also the systems for experimentation will be introduced.

2.2 Autotuning

The autotuner cannot be discussed in detail, because the autotuner is company confidential. Therefore only a brief overview will be given of the tuner and no pictures or details will be presented. The explanation about the first investigation will therefore only be superficial.

2.2.1 "Smart" autotuner

The "smart" autotuner is a stand-alone algorithm which can communicate with a motion controller. The algorithm walks through a couple of steps during its operation. To autotune an axis first a system-identification is done after which an off-line controller-calculation can be started.

To identify a system the algorithm first sets the polarity and next a controller is optimized by the Ziegler Nichols method. With the obtained controller a run of closed-loop identification measurements is done by exciting the system with a sine wave at a specific frequency. Out of the differences between the input and output of the system the transfer is calculated. Next this transfer is used to calculate the optimal controller settings. The controller is optimal when the allowable phase-margin and gain-margin have been reached. During the calculation several parameters may be altered to influence time domain aspects (e.g. overshoot vs. settling time).

2.2.2 Experimentation

To get to know the autotuner a number of experiments have been done on several experimental setups. The first thing noticed, during the experiments, is the time consumption of the identification. The procedure takes about 10 minutes. The excitation with several, in frequency increased, sine waves
happens to look very harsh. Especially on resonating frequencies, the systems shake tremendously. Another drawback is the resolution of the measurement. Now the standard resolution is 150 points, but if the resolution of the transfer has to be doubled, twice as much sine waves have to be evaluated. This results in a duplication of the measurement time. The measurement of one point (sine wave) takes about 4 seconds, so for a resolution of 1,000 points (which is quite normal in a transfer evaluation) it would take 67 minutes.

The second thing noticed is the limitation in the notch filter. The algorithm is able to find a peak in the transfer, but the width of the notch has to be set by the user. Also the Notch frequencies cannot be set apart from each other. With this functionality a Notch filter could be made skew, this can eventually save some phase loss and give a higher reachable bandwidth. (see A).

The strategy of the controller calculation is the third item. The algorithm uses two values (Phase-margin and Gain-margin) for the evaluation of the allowable controller setting. By evaluating the Sensitivity instead, there would be no flaw in the Nyquist criterion. Also the parameters for effecting the time-domain now must be done by hand, but maybe there is a connection between the frequency-domain and the time-domain. The algorithm would then be able to auto-tune time effects as well.

The last item is Feedforward. The Feedforward parameters are static and calculated out of breakaway measurements and step responses. This method may be optimized by evaluating the tracking performance. Friction in a system is mostly position dependent, so the Feedforward parameters may also depend upon the position. Furthermore adaptive algorithms or Iterative Learning control may be a possibility. By optimizing the Feedforward the feedback will become less important.

2.2.3 Interviews

To evaluate the found aspects for improvement and to obtain more insight in the wishes of the practical field, interviews have been held with Nyquist Employees. To get technical information as well as customer information people of the department of development & engineering and product management have been questioned.

The results of the interviews are pretty much overlapping with the author’s own findings. Everybody thinks that a skew, self fitting notch filter would be a useful supplement. There also was a great interest in other tuning strategies and the relation between time-domain and frequency-domain. The opinion about Feedforward seems to be important, but not one of the crucial subjects. An addition to the author’s findings would be the ability to tune multiple sorts of drives. The present tuner only tunes Current-loop drives, but at Nyquist also Velocity-loop drives are used and these should also be automatically tunable. Another addition is the ability to check the results. E.g. tracing the reference signal would give insight in the spectral content of the control signal. Herewith insight in the controller effort could be obtained.

The subject that attracts the most attention is system-identification. First of all the measurement time should be much less and second it should be much more machine-friendly. By showing this identification method to a client, it simply would be frightening and time-consuming. The bad resolution of the transfer and the deviations in it might also lead to blindness in the system-behavior. Resonances could simply be missed which results in a bad controller. Solving this problem by increasing the amount of excited sine waves would lead to much longer identification times. Therefore another identification method has to be obtained.
2.2.4 List of improvements

By combining all the above explained items and arranging them in order of importance the following list of improvements has been set up:

1. Improvement of System identification.
2. Use of an oblique Notchfilter.
3. Investigation on other Tuning strategies.
4. Investigation on Feedforward.
5. Tuning of different drive-types.
6. Ability to check the auto-tuned results.

As stated before the first item has been chosen for extensive research. This investigation will be presented in the following chapters.

2.3 Experimental setups

To test the autotuner and the different algorithms for functionality and errors everything must be thoroughly tested by experimentation. At Nyquist two different experimental setups are available, namely a fourth order system and a V.R.S. (Variable Robot System) system.

2.3.1 Fourth order system

This setup consists of a motor with an encoder and on the other side a rotating mass (also with encoder) which is coupled to the motor via an elastic rod (see figure 2.1). The interconnected rod acts as a torsional-spring. If the system is coupled back via the motor-encoder the system acts as fourth order (anti-resonance and resonance in the transfer). If the system is coupled back via the rotating mass-encoder the system acts as second order (one resonance in the transfer).

Figure 2.1: Fourth order system
2.3.2 Philips V.R.S. system

A more practical system is the Philips V.R.S. robot. This is a system with an x,y,z and rotating axis (see figure 2.2). The robot is a very rough mechanical system. The transmission of the motor to the linear axis is done by a cog-wheel on a rack. Because of the modular set-up (e.g. stacked axis) of the system the transfer for the V.R.S. shows various mass-decouplings.

![V.R.S. system](image)

Figure 2.2: V.R.S. system

2.4 Discussion

The results of the investigation on improvements for the autotuner have been presented. A list of improvements has been setup. The first item is chosen to investigate thoroughly and it will be elaborated in the following chapters. Also the experimental setups have been presented. All the experimentation will be done on these setups.
Chapter 3

System Identification

3.1 Introduction

This chapter presents the investigation on methods for machine identification. Two different methods will be explained. Next the transfer functions of the two experimental setups will be determined with the use of known hardware. With the obtained transfers there is a basis for what the program and the Nyquist hardware should be able to determine in the end. Therefore first the Nyquist hardware will be tested for functionality. Next a spectral analysis algorithm will be derived to analyze the measurement data.

3.2 Identification

The present autotuner has a built in identification module. The identification is done via a direct measurement of the input and output signal of the system. To preserve the system within the specifications, the measurements are done closed loop. A scheme of the system is presented in figure 3.1.

This is a linear system in closed loop. The input \( r(t) \) is a reference trajectory and \( y(t) \) describes the output of the system. Both \( n(t) \) and \( m(t) \) are random uncorrelated noises which are not influenced by any input. For the analysis the system needs to be excited. Therefore an extra input \( w(t) \) is used.
3.2.1 Direct measurement

A direct measurement has some major drawbacks in case there is external noise (influence of \( n(t) \) and \( m(t) \)) in the system. A brief overview about identification methods and the mathematical proof is stated in [2]. The transfer is calculated by dividing the output \( y(t) \) by the input \( u(t) \), see figure 3.1. By writing out the power spectra of the input and output the following formulae are derived [2]:

\[
S_{uy} = \frac{1}{T} U \ast Y = \left( S \ast SC \ast CHS_{rr} + S \ast SC \ast H \ast HS_{nn} - S \ast SC \ast S_{mm} \right)
\]

\[
S_{uu} = \frac{1}{T} U \ast U = \left( S \ast SC \ast CS_{rr} + S \ast SS_{ww} + S \ast SC \ast CH \ast HS_{nn} + S \ast SC \ast CS_{mm} \right)
\]

\[
H = \frac{S_{uy}}{S_{uu}} = \frac{C \ast CHS_{rr} + HS_{ww} - C \ast H \ast HS_{nn} - C \ast S_{mm}}{C \ast CS_{rr} + S_{ww} + C \ast CH \ast HS_{nn} + C \ast CS_{mm}}
\]

When only the spectral part \( S_{ww} \) is present, the FRF (Frequency Response Function), \( H(f) \) is measured. However, when there is external noise, this takes care for a biased estimate of the FRF. In the extreme situation with only noise \(-1/C\) will be measured.

The coherence also gives no great assistance. Writing out the whole coherence would become very large and inconvenient. For the illustration only the terms \( w(t) \) and \( n(t) \) will be taken along. The coherence then becomes [2]:

\[
\gamma_{uy}^2 \approx \frac{|H|^2 |S|^{4} S_{ww}^2 - 2|C||H|^4 |S|^4 S_{ww} S_{nn} + |C|^2 |H|^4 |S|^4 S_{ww}^2}{|H|^2 |S|^{4} S_{ww}^2 + 2|C|^2 |H|^4 |S|^4 S_{ww} S_{nn} + |C|^2 |H|^4 |S|^4 S_{nn}^2}
\]

The coherence function does not give any good indication, because in the case of only external noise it will go to one as well.

There can be concluded that by using the direct method, undesired noise causes a distortion in the estimated FRF. The coherence function is also inaccurately influenced by this external noise and therefore it cannot be used as a good quality-indicator for the measurement.

3.2.2 Sensitivity measurement

Instead of measuring by the direct method the most common technique for measuring in closed loop is the determination of the Sensitivity. The Sensitivity can be derived by measuring the input \( u(t) \) and dividing it by the measurement of the excitation \( w(t) \), see figure 3.1. This method is fully explained in [13]. By again writing out the power spectra of these signals, the following formulae can be derived [2]:

\[
S_{wu} = \frac{1}{T} W \ast U = SS_{ww}
\]

\[
S = \frac{S_{wu}}{S_{ww}}
\]
The advantage, which directly can be seen, is that by measuring the sensitivity there will be no bias. The FRF can easily be derived out of the sensitivity [2]:

\[
S(f) = \frac{1}{1 + C(f)H(f)} \tag{3.6}
\]
\[
H(f) = \frac{1 - S(f)}{C(f)S(f)} \tag{3.7}
\]

The undesired uncorrelated noise inputs average themselves out by measuring the sensitivity. The coherence now becomes [2]:

\[
\gamma_{wu}^2 = \frac{S_{ww}}{|C|^2S_{rr} + S_{ww} + |C|^2|H|^2S_{nn} + |C|^2S_{mm}} \tag{3.8}
\]

The advantage is that the coherence will become lower with more external noise. This makes it a good quality-indicator for the measurement. A clear disadvantage in the determination of the FRF out of the sensitivity is that the controller has to be known. The controller can be derived by a frequency response function (the controller parameters are known) or it can be measured out.

### 3.2.3 Proces sensitivity

By also measuring the proces sensitivity the controller will automatically be derived out. the spectra of this measurement give the following formulae [2]:

\[
S_{wy} = \frac{1}{T} W * Y = SHS_{ww} \tag{3.9}
\]
\[
SH = \frac{S_{wy}}{S_{ww}} \tag{3.10}
\]

With the following coherence [2]:

\[
\gamma_{wy}^2 = \frac{|H|^2S_{ww}}{|C|^2|H|^2S_{rr} + |H|^2S_{ww} + |H|^2S_{nn} + S_{mm}} \tag{3.11}
\]

So by measuring the proces sensitivity, also no bias will appear due to external noise. The coherence gives the same result as for the sensitivity. The FRF can eventually be obtained by dividing the proces sensitivity by the sensitivity. This gives also an unbiased result. The advantage of this method is that there is no knowledge of the controller necessary. Writing out the formulae gives [2]:

\[
SH = \frac{S_{wy}}{S_{ww}} \tag{3.12}
\]
The determination of the coherence gives a problem. Because the coherence of three different signals cannot be determined. A suggestion for a coherence look-a-like stated in [2] would be the multiplication of the coherence of the sensitivity and the process sensitivity. The determination of this coherence nevertheless gives a little distorted image of what is expected. Therefore only the coherence of the sensitivity will be taken along, because this also gives a quality-indication, but no confusion.

### 3.3 System transfers

To obtain a basis for the further investigation the transfers of the experimental setups have to be derived. This can be done with a TU/e DACS AQI data-acquisition interface which is directly linked to Matlab. Because of great experience with this hardware and the ability of using a rapid prototyping environment, the transfers can be derived fast and very accurate. Via Simulink a measurement model has been setup as can be seen in figure 3.2.

![Figure 3.2: Matlab measurement scheme](image)

The scheme shows great similarity with figure 3.1. By measuring the excitation $w(t)$ and the control signal $u(t)$ the sensitivity can eventually be derived, as explained before. Also two possible reference trajectories can be used, namely a sine wave (for limited stages) or a jog mode (for free axis). The axis or stage must be moving during the experiment. As stated in [13] some non-linearities as for example cogging or stick slip can be reduced by applying a movement during the experiment. If the axis would be kept still, the non-linear effects would influence the outcome of the FRF by showing extra stiffness. For the controller only a gain is used. The $K_p$ must be kept as low as possible, because the bandwidth then will be at a low frequency. During the measurement everything below the bandwidth will be attenuated, so the FRF will therefore only be estimated properly above the bandwidth. However if the bandwidth will be chosen to low the reference movement will not be followed and the friction effects will return. So a proper tradeoff between a good reference movement and a low bandwidth has to be
made.
For the excitation signal $w(t)$ a broad banded signal is used. By using a broad banded signal as for instance white noise a whole spectral band of frequencies will be excited continuously. By exciting all frequencies much information is gathered in a short time, in contradiction to the excitation by single sine waves as used in the autotuner. The measurement time may therefore be much less. This will also be treated in section 3.5.

By now doing measurements on the experimental setups data is collected. The data can be evaluated by the transfer function estimate algorithm (TFE) in Matlab. The obtained transfer can be evaluated. If the transfer is not yet good enough, the reference movement parameters or the excitation signal amplitude may be adapted. Subsequently the measurements can be repeated as well as the evaluation until the final satisfying transfers will be obtained. So for now the transfers are optimized by making assumptions based on experience. The transfer of the fourth order system is presented in figure 3.3.

![Figure 3.3: Transfer fourth order system (feedback over mass)](image)

The transfer was measured via the encoder of the mass. The system then acts as second order and this can also be seen in the transfer. Mass (-2 slope) and a resonance spike can easily be identified. The transfer of the V.R.S. robot has been obtained in a similar way and is presented in figure 3.4. The transfer of the x-axis has been measured. The expected mass-decouplings also return in the identified transfer.

### 3.4 Nyquist hardware

The final goal is to be able to determine the same accurate system transfers. In the previous section the transfers have been obtained with the use of Matlab and an AQI data-acquisition unit. At Nyquist Matlab cannot be used in the tools (yet). Therefore the spectral analysis, as done by Matlab (TFE), has to be redeveloped for use in the Nyquist tools. This will be explained in section 3.5. Another item is
the use of other hardware. At Nyquist they produce their own motion control products, so the measurement hardware has to be one of the own controllers. For the investigation a NYCe3512 motion control unit has been provided. This unit is able to control four axis. The unit is connected to the host pc by a firewire interface. The main control loop runs on the unit itself, whereas the parameters and the operation are controlled by the host pc. The main control loop is presented in figure 3.5.

Next to the feedback- and feedforward-controller and the filters a few other features are provided. The unit is also able to trace predefined signal parts of the loop and the digital signal processor (DSP) is able to generate testsignals. The generated testsignal can be injected in several places in the loop.

### 3.4.1 Testsignal generation

To make sure that the quality of the measured data is the same as for the AQI measurement first the hardware has to be tested. One important part is the generation of the excitation signal (e.g. random noise). By connecting the analog output of the unit to the AQI an impression of the quality of the
noise can be made. Random noise is generated by the DSP and the output is measured by the AQI. Calculating the spectral density will give insight in the spectral information of the generated noise, as can be seen in figure 3.6. The unit runs on 1 kHz and according to the shannon theorem the DSP should generate noise over the whole bandwidth until very near 500 Hz (Nyquist frequency). Figure 3.6 shows that the noise is accurately generated, it contains no holes or cut-offs in the frequency band.

![Spectral density of the random noise (DSP)](image)

3.4.2 Tracing & Signal processing

Another important part is the tracing of the measurement data. The unit has its own memory chip on which the trace-data is stored. At certain intervals the memory buffer will be full and the data has to be downloaded to the host pc. Analyzing the traced and downloaded data will give insight in the quality of the trace. Next to the tracing the whole signal processing of the unit should be tested. Besides the noise generation and the tracing, the processing loop may contain flaws which influence the measurement. By combining these two last items the whole unit can be tested at once. Therefore a stand-alone program has been made which has the same functionality as the Simulink program shown in figure 3.2. The “identifier” program as shown in figure 3.7 has been written in both Visual Basic and C. The full setup of the program will not be discussed here because this setup will also be explained in chapter 5. The final program namely has exactly the same setup. The program is able to both control the Motion unit and to retrieve the trace data. With the program an axis can be chosen for identification, and the polarity of the loop can be checked and set right. On the chosen axis a reference movement can be executed. This may be a Jog motion or a repeating point to point movement (PtP). A sine wave is in this case not possible because this type of signal is generated by the DSP. The DSP is already occupied because it generates the excitation signal (random noise) and only one signal can be generated at a time.

Now by using the same settings as for the Simulink measurement an identical measurement can be done via the motion control unit. The traced data is captured on disk and can eventually be analyzed by the same algorithm in Matlab. For the test the fourth order system has been measured and the obtained transfer is presented next to the former obtained transfer in figure 3.8. The analysis shows a similar transfer and the conclusion can be made that the motion unit is perfectly capable to produce
similar results.

Figure 3.7: interface of the "identifier" program

Figure 3.8: Transfer of the fourth order system measured via AQI and the Motion control unit
3.5 Spectral analysis

The hardware has been fully examined and no problems were discovered. The next item to solve is the spectral analysis. Until now the data has been examined with the help of Matlab. This is not longer possible in the final program, so the algorithm, as used in Matlab, has to be reformulated in a self written algorithm.

3.5.1 analysis items

The transfer function estimation algorithm (TFE) in Matlab uses several data-analyzing items. To get insight in the algorithm, the mathematics source code book of Matlab [8] has been consulted. Out of the source code a couple of important aspects for spectral analysis could be derived. To obtain a full understanding about these aspects also the notes of signal analysis have been consulted [6]. A little discussion about the items will be presented in the following subsections.

FFT algorithm

The first important item is the fourier transformation of the measured data. Fourier transformation is the dissolution of a time domain signal in sine and cosine functions in the frequency domain. The measured data is collected by a discretely sampled unit and therefore the data also will be discrete. Because the data is sampled, it contains only a limited band of frequencies. The highest possible frequency in the signal \( x(t) \) is half the value of the sample frequency \( f_b \) (theorema of Shannon). For the transformation we may handle an observation interval with a finite length \( T_0 \). The path of the signal \( x(t) \) outside this interval is not known. The signal can be assumed periodic with ground period \( T_0 \) as shown in figure 3.9 [6].

The periodic signal \( x_p(t) \) can be described by the complex fourier series and the accompanying fourier coefficients with ground-frequency \( f_0 = 1/T_0 \). The formulae as stated in [6] are:

\[
x_p(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk2\pi f_0 t}
\]

\[
c_k \approx \frac{1}{T_0} \int_{t_0}^{t_0+T_0} x_p(t)e^{-jk2\pi f_0 t} dt, \quad k = 0, \pm 1, \pm 2, \pm 3, \ldots
\]

Figure 3.9: Sampled signal over finite observation interval
To calculate the Fourier coefficients (3.16) in fact an integration about the ground-frequency would be necessary, but only the sampled values \( x(t_0 + nT_b) \) are available. As method of approach for the integrant the summation goes [6]:

\[
c_k \approx \frac{1}{T_b} \sum_{n=0}^{N-1} x(t_0 + nT_b)e^{-j2\pi f_0(t_0 + nT_b)T_b} = \\
= \frac{T_b}{T_0} e^{-j2\pi f_0/T_0} \sum_{n=0}^{N-1} x(t_0 + nT_b)e^{-j2\pi nT_b/T_0} = \\
= \frac{1}{N} e^{-j2\pi f_0/T_0} \sum_{n=0}^{N-1} x(t_0 + nT_b)e^{-j2\pi n/N}, \quad k = 0, \pm 1, \pm 2, \pm 3, \ldots
\]

This approach is valid because no higher frequencies than \( \frac{1}{2} f_b \) will be present in the signal \( x(t) \). Next the line-spectrum of the periodic signal \( x_p(t) \) can be pointed out, see figure 3.10 [6].

As a consequence of sampling the spectrum repeats itself around every multiple of the sample-frequency \( f_b \). This indicates that the only useful information lies inside the frequency area \( (-\frac{1}{2} f_b, \frac{1}{2} f_b) \).

For a matter of fact the information may be transformed to only the positive side \( (0, \frac{1}{2} f_b) \). Negative frequencies do not really exist and are only a product of the complex transformation, so they may be eliminated.

The FFT algorithm in Matlab is based on the discrete Fourier transformation as explained above. Matlab follows the radix(2) method which is nothing more than a very efficient way of calculating the discrete transform. Several methods have been investigated, but finally the same algorithm (which was obtained in C-code) was used.

**Resolution**

Another important aspect of discrete Fourier transformation is the resolution. The resolution can be stated by the length of the observation interval. With the data sampled at a certain frequency the resolution is set by taking an interval \( T_0 \). This time \( T_0 \) together with the sample frequency gives the
number of sample points $N$ that will be taken along in the transformation. The resolution then becomes $f_c = 1/T_0$ or $\frac{1}{2N}$.

In the FFT-algorithm the desired resolution must be stated by handing the number of points $N$. The radix(2) discrete fourier transformation algorithm divides the data in an equal amount of even and uneven data points. Therefore the number of points $N$ is usually chosen as a power of two (e.g. 512, 1024, 2048, ...). By handing more points the resolution becomes smaller and the spectrum lines come nearer to each other.

**Leakage**

Refining the resolution also limits the phenomenon called leakage. The transformation of measured data over $N$ points gives the information on the specific spectrum lines. If a frequency in the data coincides with a spectrum line the transformation gives just the right frequency and amplitude, see figure 3.11 a. But the resolution is limited and if a frequency in the data falls between two spectral lines the transformation is not able to give the right result. The information will then leak to neighboring frequency lines, see figure 3.11 b. By increasing the number of FFT points $N$ (NFFT) there are more spectrum lines (frequencies) and the leakage will diminish.

![Figure 3.11: Transformation result of coinciding and non-coinciding data](image)

**Averaging**

The number of data points is not only inherent to the NFFT, but also to the length of the measured data. To obtain a transformation with a specific resolution, $N$ data points will be needed. This however gives the result of one data block. In practice this normally gives a rough estimation as can be seen in figure 3.12 (a). Now by taking much more data points than the desired resolution, the transformation can be done several times and the solutions can be averaged. By averaging the solution, the obtained transform will be much smoother because the non-periodic parts will be averaged out. The obtained transfer will eventually look much better as can be seen in figure 3.12 (b).

**Windowing**

Next to a finer resolution the utilization of a window will also limit the leakage. The observation interval with time $T_0$ is assumed periodic. In practice the signal of the interval may not start and end at zero on the borders as can be seen in the top of figure 3.13 [13]. This influences the coinciding with the spectral lines and therefore creates leakage. Another aspect of the observation interval is the difference between the end point and the beginning point between two following periodic intervals, see figure 3.13 in the middle. These jumps will lead to very high frequency components during dissolution. These components will come back in the solution as pretending frequencies. This phenomenon is called aliasing. By applying a window as for instance a Von Hann window (see figure 3.14) the leakage...
and aliasing effects can be reduced. By multiplying the window with the data the signal is forced to zero at the borders, see the bottom of figure 3.13. The discontinuities in the periodic assumed signal will then be limited.

In the literature many windows can be found which fulfill these demands, everyone has its own advantages and disadvantages. The mutual differences only are very subtle, and therefore only the most applied type (Von Hann) will be used in the algorithm. The window is set up in two halves and the first half can be stated as:

$$w_{vh} = \frac{1}{2} \left(1 - \cos\left(\frac{2\pi n}{NFFT + 1}\right)\right) \quad \text{for} \quad n = 1, \ldots, NFFT/2$$ (3.18)

The window can then be completed by mirroring the first half to the right after the last point. This way an NFFT-point Von Hann window has been created which already was shown in 3.14.

![Figure 3.12: Effect of averaging on the transfer estimation](image1)

Figure 3.12: Effect of averaging on the transfer estimation

![Figure 3.13: Jump-like effects due to assumed periodicity and the effect of windowing](image2)

Figure 3.13: Jump-like effects due to assumed periodicity and the effect of windowing
The last point of interest for the algorithm is the overlap of the following observation intervals. As stated before a large amount of data will be analyzed by a specific NFFT-sized transformation and the solution will be averaged. Normally this is done by windowing the following data-intervals as can be seen in figure 3.15 (a). By the use of overlap the following windows can be shifted over each other, see figure 3.15 (b). The shifting has the advantage that with the same amount of data more averages can be done, or the same amount of averages can be done with less data. Another advantage is that less data will be thrown away, so more available information (in the data) will be used. See shaded parts in figure 3.15.

Figure 3.14: Von Hann window

Figure 3.15: Use of overlap
3.5.2 Algorithm

Now with the obtained knowledge about spectral analysis and the obtained Radix(2) FFT-algorithm the whole spectral analysis can be setup. The algorithm has to begin with opening the measurement data and taking the desired NFFT points and the percentage for overlap. With this information the number of data-arrays (k) must be calculated. The measurement data then must be divided in k arrays. Next a loop must be started which walks through a couple of steps, k times. By doing this, all the data arrays will be analyzed and in the end the solutions can be averaged. The first step in this loop must be multiplying the data with the stated Von Hann window (windowing). Next the windowed arrays will go through the fourier transformation. Out of the resulted data the powers and cross-powers can be calculated:

\[
\begin{align*}
    \text{Power} &= (u, y, w)^2 \\
    \text{CrossPower} &= (u, y) \ast \text{conjugate}(w)
\end{align*}
\]

(3.19)

All the calculated powers and cross-powers must be summed together. The obtained data then must be split in half, because only the positive (frequent) data is needed. Next the Sensitivity an Proces Sensitivity can be calculated:

\[
\begin{align*}
    S &= \text{CrossPower}(w, u)/\text{Power}(w) \\
    PS &= \text{CrossPower}(w, y)/\text{Power}(w)
\end{align*}
\]

(3.20)

Finally the transfer and the coherence can be calculated:

\[
\begin{align*}
    H &= PS/S \\
    \text{Coherence} &= \text{CrossPower}(w, u)^2/(\text{Power}(w) \ast \text{Power}(u))
\end{align*}
\]

(3.21)

In a flowchart the algorithm looks like figure 3.16. The algorithm has been written in C-code and is fully reflected in B. To Test this algorithm the measurement data out of the former hardware-test has been analyzed. In figure 3.17 the results of both matlab and the self written algorithm have been plotted over each other. The result shows that the obtained transfers are exactly the same, so the conclusion may be drawn that the self written algorithm functions perfectly.

3.6 Discussion

Two different methods of machine identification have been presented. The method of measuring Sensitivity and Proces Sensitivity has been proven to be much better, because external noise does not effect the FRF estimation. Next the transfers of the experimental setups have been obtained. These act as a basis for the the kind of transfers which the final approach has to obtain. To be sure that the Nyquist hardware is capable of performing good measurements, the hardware has been tested. No problems have been detected during these tests. Finally a spectral analysis algorithm has been derived. First a literature study has been performed which pointed out several important items for the analysis. These points finally have been processed in a self written analysis-code. Comparison of the C-code with the Matlab code has shown that the algorithm is able to produce an equal result.
Figure 3.16: Flowchart of Spectral analysis algorithm

Figure 3.17: Transfer calculation comparison between Matlab and spectrum analysis algorithm
Chapter 4

Identification parameters

4.1 Introduction

In the previous chapter the analysis of the measured data has been discussed. A good analysis also needs good measured data. This chapter presents the investigation about acquiring good starting parameters for performing an identification measurement. At the beginning nothing of a system, that should be identified, is known. Only some hardware parameters will be available, for instance the encoder resolution, the allowed stroke of a stage and the type of drive. In the former sections the parameters have been guessed out of experience (e.g. the AQI measurements). The main goal is to make an automated program, so the missing parameters now have to be obtained via automatic sequences. The investigations about the different parameters will be presented in the following sections.

4.2 Allowable stroke

To protect a limited stage against striking to the end points the stage has to be protected. The motion controller is equipped with a “quick stop” function. The position of the stage can be monitored by the encoder position. By tracing the encoder, the position can be compared with the maximum allowed position. As soon as the position exceeds the allowable value, the quick stop procedure will be fired. This loop then functions as a safety guard for the stage position.

4.3 Polarity

The first unknown parameter is the polarity of the measurement system. By sending a positive control signal the encoder should also measure a positive movement. If this is the case, there will be a good feedback coupling. If the measured position is negative the encoder has been wired backwards. Now the polarity (the sign of the encoder signal) must be inverted to obtain a good feedback coupling.

A good proposal for polarity measurement is stated in [10]. The polarity of a system can be measured by doing a ramp excitation on the axis. By applying a slowly increasing voltage and by looking at the position measurement the so called breakaway voltage can be detected. The axis (position) will break away at a certain voltage as can be seen in figure 4.1. These measurements can also give a measure for the stick-slip. The use of a break-away measurement for polarity therefore may be a little exaggerated, but the autotuner needs the break-away voltages for friction compensation. Therefore the choice is made to equip the identification with this detection, the measurement then needs to be done only once, and this saves time.

The two important parameters are the slope of the ramp function and the break-away detection level.
(increments that the axis is expected to move). The choice for the slope is important for obtaining the coulomb and viscous friction parameters, which are used in the autotuner for evaluating the feed-forward as stated in [3]. For the slope of the ramp a proposal is given of 0.2 V/s. The detection level is very dependent on the resolution of the encoder. Therefore the user has to make a choice for the amount of increments from where the axis is expected to be broken away. After applying the ramp function, the system brakes on friction or damping. As soon as the detection level is exceeded the output is set zero and the system will brake. In case of nearly no damping or friction the system is still protected by the allowable stroke loop and the axis cannot shoot out of range. Most systems will have some direction dependency in the stick-slip level and thus also in the break-away voltages. Therefore the measurement must be done in both directions. The polarity finally can be calculated by:

\[
POLARITY = \text{sign}(\text{Controller}_{out}) \times \text{sign}(\text{encoderlevel})
\]  

(4.1)

The polarity eventually can be set by the library parameter "measurement system direction" in the SAC (single axis controller) library of the motion controller.

![Figure 4.1: Breakaway test](image)

### 4.4 Reference Movement

As already defined in section 3.3 the axis must have a reference movement during the measurement to reduce non-linear effects. In the Simulink program two different movements were chosen, namely a JOG mode and a sine wave. The JOG mode was meant for free axis and the sine wave for limited stages. In the Identifier program that was written in section 3.4.2 this was solved by using the JOG mode and a repeating PtP movement. This functionality is directly available in the motion controller. In the previous chapters this worked perfectly, so for the final method the same solution will be applied.

### 4.5 Controller

To be able to follow the reference movement the system needs to be set closed loop. Therefore also a controller is necessary. The controller must be tuned to a very low bandwidth, because everything below the bandwidth will be controlled away. So the excitation signal will also be handled below the bandwidth-frequency. As stated before nothing of the system is known at the start, the parameters for a controller therefore cannot be known without any estimation.

By identifying the (low-frequent) mass, a controller could be calculated out of the mass information.
With an estimation for the mass of the system, a "mass line" with a slope of -2 could be drawn in a bode plot. The bandwidth then could be chosen at a desired (low) frequency. The controller consists only of a gain Kp. Most system have at least little damping, so only a Kp level will mostly be sufficient. If this is not the case also a damping factor Kv may be chosen afterwards. The spectrum analysis algorithm is not influenced by the type of controller (as stated before). To obtain the mass information of the system several methods have been considered.

The first method investigated was a Ziegler Nichols measurement. An example of this measurement was found in [10]. During the measurement the Kp of the system is increased by steps until the system becomes unstable. The Kp just at the instability point would be the point of interest. At this point the system would be vibrating at a specific frequency. This frequency should be identified and together with the Kp value this could give an estimation for the mass. From the obtained point a "mass line" could be drawn in a bode plot, and a controller could be calculated. One of the major drawbacks of this type of measurement is that the system is forced to instability. This is something that always should be avoided. Another drawback is that the instability point could be found just in a resonance or anti-resonance spike. If from this point a -2 line would be drawn major errors in the mass estimation would be made. The calculated controller then would be wrong and this might influence the measurement.

The second method investigated was obtaining information out of the breakaway measurement. After the system has broken away, the axis will be accelerated further by the ramp function. To accelerate the system friction has to be overcome and mass has to be pushed forward. A simple model can be examined that describes this phenomenon:

\[ Ma + w = a t \]  

with \( M \)=mass, \( a \)=acceleration, \( w \)=coulomb friction, \( \alpha \)=ramp slope

\[ Ma = \alpha t - w \quad \rightarrow \quad Ma = C - w \]  

integrating twice gives:

\[ Mx = \frac{1}{2} (C - w) t^2 \]  

with \( C \)=ramp value at certain time \( t \), \( x \)=position

The viscous friction is assumed to be negligible. The coulomb friction is assumed to be constant. To be able to calculate this parameter out, two measurements can be done with different ramp slopes. Out of these two measurements \( w \) can be ruled out and by taking the ramp parts and the position parts at the same time \( (t_1 = t_2) \) the mass can be calculated:

\[ Mx_1 = \frac{1}{2} (C_1 - w) t_1^2 \]  
\[ Mx_2 = \frac{1}{2} (C_2 - w) t_2^2 \]  

\[ M = \frac{C_1 - C_2}{\Delta x}, \quad t_1 = t_2 \]  

in practice however this does not work so perfectly and simple. Systems could suffer from great
position dependent frictions and then the breakaway-measurements are not accurate anymore. In figure 4.2 several measurements with two different ramp slopes are plotted which have been measured on the fourth order system. The measurement results are all different instead of overlapping and the mass-estimations then also will vary several orders. This approach again did not give a satisfying result. The data was also analyzed with a least squares fit and the experiment was expanded with a kalman filter. The fit showed the same results and the kalman filter only gave a good result with good guessed boundary values. These two approaches also did not contribute to estimating a better result.

Finally the mass-estimation procedure has been cancelled. For the controller an updating algorithm has been designed. The algorithm consists of a loop that starts the reference movement and subsequently traces the position error of each movement step. The error is afterwards compared with a desired performance value. If the performance is not yet achieved the value of the Kp controller is updated by a factor. The starting value for the Kp is taken very small \((1 \times 10^{-9})\) to be sure that the starting value is always small enough for most systems. This way it will function very universal. Because the starting value is very little the loop would take a lot of time if the update factor was small. Increasing the factor would quicken the updating, but a bigger update factor will also deteriorate the accuracy of the performance. Therefore the loop is equipped with a variable update factor. The factor is bigger if the performance is not near at all and the factor decreases if the performance becomes near the desired value. The updating loop is showed below in pseudo code, see 4.3. The movement amplitude is the amplitude of the reference movement, the desired performance is the prefilled performance value of the movement and the update factor is the percentage (allowed between 10% - 40%) by which the Kp is minimally updated. With this factor a little consideration between time and accuracy can be made.

The drawback of this method is that the bandwidth of the obtained controller is not known. The bandwidth will be low-frequent but it is not possible to choose the frequency as in the previous methods. This is only a problem when very low-frequent information is wanted. Mostly the user is only interested in the frequencies from the finally desired bandwidth, so this method will do perfectly in most cases.
4.6 Excitation signal power

The last parameter that is unknown is the amplitude of the excitation signal. As stated before, the excitation signal that will be used in the program is random noise. Several broad banded excitation signals have been investigated, for instance random noise, pseudo random binary sequences, random number and chirp signals. There is only a little difference in the solution by using the first three signals. A chirp signal is a sine wave which increases and decreases in frequency in a specific time interval. The energy distribution in this signal is more efficient, because only one frequency is excited each interval. The major drawback is that the measured (spectral) information is much less, because not all frequencies are excited each time step. Therefore the measurement time would have to be much larger to obtain a similar result. The increase in time was not acceptable and also no chirp-functionality exists in the Nyquist hardware, therefore this option has finally been turned down. Eventually the most frequently used signal, random noise, will be used.

In practice the amplitude of the signal is mostly estimated out of experience and several starting measurements. To be able to automate this item insight about the dependencies for this amplitude has to be gathered. The amplitude must be large enough to excite the system in that order that the axis/stage moves with enough resolution (encoder increments). Because no mass is known, no approximation based on the mass can be made. The only signal that is related to the behaviour of the system is the control-signal of the reference movement. The control-signal puts enough energy in the system to move the axis/stage at a certain performance level. To get enough resolution for a good measurement, the amplitude of the excitation signal should be relatively equal to the voltage of the control signal. The control signal as well as the excitation signal are summed together in the loop. By making the amplitudes nearly equal there should be enough extra excitation to measure a good response. Several tests finally delivered the following sequence.

First the control-signal of one whole reference movement is traced and then the maximum voltage
in the trace is calculated. Next this voltage is scaled by a desired percentage. The final obtained value will be used as the random noise amplitude. Tests have showed that 80 percent of the maximum voltage gives a good starting parameter for the identification measurement. Looking at the coherence there might be a poor result at lower frequencies. The amplitude may then be taken a little higher (e.g. 150 %) and the measurement should be repeated. The coherence then will show a better measurement at lower frequencies.

4.7 Discussion

The search for methods that are able to determine the missing identification parameters has been presented in this chapter. The polarity of the system now will be obtained out of a breakaway measurement. The reference movement will be the same procedure as was used in section 3.4.2. To obtain a controller several mass-estimation methods have been investigated, but neither of the methods produced an accurate result. Finally a controller updating algorithm has been derived. This method consists of a loop that monitors the position error and increases the Kp level until the desired performance is reached. The drawback of this method is that the obtained controller is low-bandwidth, but the exact frequency is not known. For the excitation signal random noise will be used. The required power for this signal will be obtained out of tracing the controller signal. A specific factor of the maximum value, out of the traced controller signal, will then be taken as the noise-amplitude.
Chapter 5

Demonstration Program

5.1 Introduction

This chapter presents the finally written demonstration program. It is named NYCeIdent. The program has been written to demonstrate the obtained identification procedure. The program works together with a NYCe3512 motion controller. In the following section the setup and layout of the program will be described. The program should finally proof itself and therefore it has been tested on the experimental setups, this is presented in section 5.3.

5.2 Programming

The autotuner is executed as a software application. The identification will be added as a module to this application. Therefore the identification algorithm also must be executed in a software program. This will eventually be done at the System Engineering department of Nyquist. To demonstrate the algorithm and to convince the employees of the utility of the obtained procedure a demonstration program should be written. The program has been written in visual basic as well as C. The User interface has been made in Visual basic, because in this program a nice graphical interface can be rapidly made. The calculations and controls have been made in Visual basic as well as C. Visual Basic has many limitations in making calculations and the timing of processes. Therefore the complicated traces, controls and calculations have been programmed in C. The communication between the two programs have been setup by a mailbox system (Glue Server) which already was created at Nyquist before. The Interface in Visual Basic as well as the C-server are able to communicate with the Single Axis Controller library. This library contains all the commands and processes to employ the motion controller. A communication scheme is presented in figure 5.1.

The program has to contain all the aspects that have been derived in the previous chapters. The graphical interface has to acquire all the required knowledge for setting the program parameters. After the program has been set up, the several loop modules may be employed one after another. in figure 5.2 a flowchart of the program is presented.

The axis must be chosen and connected for every identification. The allowable stroke has to be chosen by the user, because this is very dependent on the type of machine/axis. The movement must be selected, namely a JOG or a PtP movement. The amplitude and a time scale of the movement must be chosen according to the allowed stroke of a stage or a specific area that needs to be investigated. For the JOG-mode only a amplitude is necessary which sets the slope of the ramp function. The breakvalue (increments) must be chosen according to the type of encoder which is used on the axis. This because
an encoder with a very fine resolution needs more increments to detect breakaway than a less accurate encoder. The startvalue for the Kp controller now is set on $1 \times 10^{-9}$. This value may be altered, but this should be done with caution! The update rate is set on 20 percent. The value may be altered between 10 and 40 percent to shift a little between speed and accuracy. The performance of the movement must be set by a percentage. This percentage must be chosen according to the movement amplitude, the axis then must follow by that percentage of the amplitude. To maintain a accurate movement, the percentage must also increase if the amplitude increases. Measurements have showed that a noise amplitude, which is set at 80 percent of the maximum controller voltage, is a good starting guess for the first measurement. The value may be increased during the following measurements to improve the measurement quality. The duration of the measurement mostly may be 60 seconds. Together with 1024 NFFT points and 50 percent overlap this will give a satisfying result in most cases. The last three named values may be altered to save time or to improve the resolution or the smoothness of the transfer. The first measurement on a system with NYCeIdent has to be done completely. For the next measurements on the same axis some parts of the program may be bypassed. The breakaway test, the controller estimation algorithm and the noise amplitude estimation may be skipped. If the obtained values of these modules are already obtained to satisfaction, it would be only time consuming if these parts would still be repeated.

The program starts after all the values are filled in and the start button is clicked. The filled in values will be checked and if the values are correct the flowchart will be walked through (see figure 5.2). First the breakaway measurements are done and the polarity of the system is set. Next the movement is initiated and the controller updating loop will be walked through. After this the axis or stage will be moving according to the chosen movement type and the desired performance. Now a trace is made of the controller signal during a movement step and the noise amplitude will be calculated. Afterwards the real identification measurement will start and the identification data will be traced and stored. Finally the data will be analyzed by the spectral analysis algorithm and afterwards the data can be plotted by the plotter in NYCeIdent. The program also consists of a save button. This is not used yet because
with this button the way of saving the analysed data must be chosen specific for the autotuner. This is unknown now and must be defined by Nyquist in the future. The program is only for demonstration of the identification and therefore this is not important at this point.

The program as it is written now is an *Advanced mode* version. Most parameters must be filled in by experienced users. For a *Basic mode* version all the parameters may be prefilled in as stated above. Only the axis, the allowable stroke and the whole movement always need to be filled in by the user at every new identification measurement.

---

**5.3 Testing**

The flowchart eventually has been put into code and a fully operational program has been realized. An impression of the HMI is showed in figure 5.3. The program finally has to prove itself by testing it on the experimental setups. The fourth order system as well as the V.R.S. robot have been identified by NYCeIdent. The results are plotted in C. The test results are accurate and by this the functionality of the program seems to have been achieved. The final identification now takes approximately 150 seconds.
All the obtained identification parts have been combined into one operational demonstration program. The program has been written in two programming languages, namely Visual Basic and C. The program consists of a Graphical HMI and is able to communicate with a NYCe motion controller. After filling in several input parameters the program will walk through several steps. At the end the measurement data is analyzed by the spectrum analysis algorithm, and the result can be plotted in the plot utility of the interface. The program has been successfully tested on the experimental setups with a much shorter identification time.

Figure 5.3: Graphical interface of NYCeIdent

5.4 Discussion

All the obtained identification parts have been combined into one operational demonstration program. The program has been written in two programming languages, namely Visual Basic and C. The program consists of a Graphical HMI and is able to communicate with a NYCe motion controller. After filling in several input parameters the program will walk through several steps. At the end the measurement data is analyzed by the spectrum analysis algorithm, and the result can be plotted in the plot utility of the interface. The program has been successfully tested on the experimental setups with a much shorter identification time.
Chapter 6

Conclusions & recommendations

6.1 Conclusions

The Nyquist autotuner has been improved. Out of experimentation and interviews a list of improvements has been setup. The first item on the list turned out to be machine identification. The assignment has been altered to only this item. A study into identification methods has learned that measuring the sensitivity and proces sensitivity is a much less noise-sensitive identification process than direct measurement. Therefore the matter for identification has been altered to this method.

To obtain a basis for further investigation, the transfers of the experimental setups have been derived. This has been done with the use of a TU/e DACS AQI data-acquisition device with a direct link to Matlab. Because of great experience with this hardware and the rapid prototyping environment the transfers could be derived fast and very accurate. The parameters for this measurement have been derived by experimenting and experience of the author.

At Nyquist they produce their own motion control products, so the measurement hardware for identification finally had to be one of the own products. To be sure that the Nyquist hardware is capable of doing the same qualitative measurements, the hardware has been tested. The testsignal generation, the tracing and the signal processing have been successfully tested. This finally showed that with the Nyquist products similar transfers could be derived.

The data finally had to be analyzed without the use of Matlab. Therefore a study has been done on spectral analysis and Fast Fourier Transformation. With the acquired knowledge and with the help of the Matlab source code a new spectral analysis algorithm has been derived. This algorithm has been fully written in C-code.

A good analysis also needs good measured data. Before a good identification measurement can be done a few other parameters have to be identified. At the beginning nothing of a system is known, besides only some hardware specifications. The main goal was to make an automated method, so several experiments have been developed to acquire the missing parameters.

Finally all the acquired knowledge has been gathered together to one universally applicable identification method. To be able to demonstrate the obtained procedure, a program has been written. The program is called NYCIdent. It has been written in both Visual Basic and C-code. The program is also able to communicate with a Nyquist motion product, so real identifications can be done. To finally proof the working of the derived identification procedure, the program has been successfully tested on the experimental setups.
The final tests have convinced Nyquist of the quality of this identification method and the decision is made to improve the autotuner with this module. The program can also function without the aim for autotuning, and therefore the program will also become a stand-alone software tool.

6.2 Recommendations

The demonstration software has been designed on a sample frequency of 1000 Hz. This will eventually only give spectral information to 500 Hz. To be able to acquire much further spectral contents in the future, the sample frequency of the final software module should be much higher.

With a higher sample frequency also the spectral axis of the measurements will be much longer. There may also be an interest in evaluating only a specific frequency part. For this the excitation signal should be able to be marked out. This could be done by a FIR (Finite Impulse Response)-filter. With this filter, the excitation signal could be cut off at certain frequencies. The spectral analysis then would only give information within this frequency band. With the obtained smaller spectral axis, the resolution in the transfer-part would then be better.

The excitation signal used in the algorithm is random noise. Doing measurements with this type of signal finally needs filtering in the spectral analysis to reduce aliasing and leakage effects. By creating an excitation signal which only generates signals at the spectral frequencies of the fourier resolution, an aliasing- and leakage free measurement could be developed. Signals like multi-sines are specifically developed for this matter. A good recommendation for further reading about this subject is the book of Johan Schoukens [11].

At Nyquist another student of the university is working on the autotuner now. Many investigation subjects of this student also require some identification measurements (e.g. friction identification). These finally derived experiments may in the future also be added to the now developed identification tool. This would finally make the identification tool more complete and more universally applicable.
Appendix A

Skew Notchfilter

The function for a Notchfilter can be stated as:

\[
N = \frac{1}{(2\pi f_1)^2} s^2 + \frac{2\beta_1}{2\pi f_1} s + 1
\]
\[
+ \frac{1}{(2\pi f_2)^2} s^2 + \frac{2\beta_2}{2\pi f_2} s + 1
\]

(A.1)

with \( f_1 \) the zero and \( f_2 \) the pole. and for a notch \( \beta_1 < \beta_2 \leq 1 \) with the depth \( \beta_1/\beta_2 \).

Normally the two frequencies are chosen equal. This gives a straight notch with a defined depth and width. Using a notch filter contributed to the phase loss around the bandwidth point. By shifting the pole upwards the notch can be made oblique. The end point of the notch becomes higher. The phase will shift also by the skewness and this gives less phase loss before the notch. See ellipse in figure A.1. By setting the frequencies (\( f_1 \) and \( f_2 \)) apart the notch will become skew, but also the depth will rise a little. The depth factor therefore has to be adapted after making the notch skew.

![Bode Diagram](image)

Figure A.1: Bode diagram skew notchfilter
Appendix B

Spectrum analysis source code

B.1 Spectral analysis code

/*
   // NYQUIST Industrial Control
   //
   // Copyright (c) Nyquist B.V. 2005 Eindhoven, The Netherlands
   // Internet: www.nyquist.com
   //
   // Project Name : spectrum analysis
   // Source Name   : spectrum2.c
   //
   // Author        : Ramon Solberg
   // Description   : spectrum analysis module for NYCEIdent.
   //                 routine spectrum analysis of 3 measured real data arrays
   //                 using radix2 method
   */
#include "math.h"
#include "fft.h"
#include <stdlib.h>
#include <stdio.h>

int Spectrum(const int N,const int NFFT,const int noverlap,double Kp)
{

    const double PI=3.141592653589793238;

    /* in te geven variabelen uit server */
    /*int powerOfTwo = 10;
    int noverlap=512;
    int N=60000; //length dataarray
    double Kp=.001;*/

    FILE *dat;
    FILE *amp;
    FILE *phs;
    FILE *fc;

    //...
int i, j, k;
int n = NFFT;
float tmp, u, y, w;
*TyvRe, *TyvIm, *Cw, *Cw, *C;
int index = 0;
float Fs = 1000.0;
int powerOfTwo = (int) (log10(n) / log10(2));

/* ---- allocate memory -----------------------------------------------*/
re1 = (double*) malloc( n*sizeof(double));
re2 = (double*) malloc( n*sizeof(double));
re3 = (double*) malloc( n*sizeof(double));
data1 = (double*) malloc( N*sizeof(double));
data2 = (double*) malloc( N*sizeof(double));
data3 = (double*) malloc( N*sizeof(double));
window = (double*) malloc( n*sizeof(double));
Hre = (double*) malloc( (n/2)*sizeof(double));
Him = (double*) malloc( (n/2)*sizeof(double));
Wu2 = (double*) malloc( n*sizeof(double));
Yy2 = (double*) malloc( n*sizeof(double));
Uu2 = (double*) malloc( n*sizeof(double));
Pw = (double*) malloc( (n/2)*sizeof(double));
Puu = (double*) malloc( (n/2)*sizeof(double));
PuuRe = (double*) malloc( (n/2)*sizeof(double));
PuuIm = (double*) malloc( (n/2)*sizeof(double));
TvURe = (double*) malloc( (n/2)*sizeof(double));
TvUIm = (double*) malloc( (n/2)*sizeof(double));
PwyRe = (double*) malloc( (n/2)*sizeof(double));
PwyIm = (double*) malloc( (n/2)*sizeof(double));
TyvRe = (double*) malloc( (n/2)*sizeof(double));
TyvIm = (double*) malloc( (n/2)*sizeof(double));
Cw = (double*) malloc( (n/2)*sizeof(double));
C = (double*) malloc( (n/2)*sizeof(double));
AMP = (double*) malloc( (n/2)*sizeof(double));
PHS = (double*) malloc( (n/2)*sizeof(double));

printf("opening data\n");

dat=fopen("data2.dat", "r"); /*open data block*/
if (dat==NULL)
{
printf("Datafile not available\n");
exit(1);
}

for(i=0; i<N; i++)
{
scanf(dat, "%f\%f\%f\%f\%f\%f\%f\%f\%f\n", &tmp, &u, &y, &w); /* read data */
data1[i]=u;
data2[i]=y;
data3[i]=w;
}

for(i=0;i<n/2;i++)            /*fill arrays with zeros*/
{
Pw[i]=0;
Pyy[i]=0;
Puu[i]=0;
PwuRe[i]=0;
PwuIm[i]=0;
PwyRe[i]=0;
PwyIm[i]=0;
}

printf("determine number of windows\n");
k=(int) floor((N-noverlap)/(n-noverlap));              /* number of windows */

printf("number of executed windows: \%d\n", k);
printf("define hanning window\n");

if (hanning(n,window))
{
 printf("error in von Hann window, data maybe not even\n");
 exit(1);
}

printf("transforming\n");

for(i=1;i<k;i++)
{
im1 = (double*) malloc( n*sizeof(double));
im2 = (double*) malloc( n*sizeof(double));
im3 = (double*) malloc( n*sizeof(double));
rewu = (double*) malloc( n*sizeof(double));
imwu = (double*) malloc( n*sizeof(double));
rewy = (double*) malloc( n*sizeof(double));
imwy = (double*) malloc( n*sizeof(double));

for (j=0;j<n;j++)                        /* define index width */
{
    re1[j]=window[j] * data1[j]+index;   /* use windowing */
    re2[j]=window[j] * data2[j]+index;
    re3[j]=window[j] * data3[j]+index;
}

index+=n-noverlap;

Radix2FFT( powerOfTwo , re1 , im1 );          /* fft U (ref signal) */
Radix2FFT( powerOfTwo , re2 , im2 );          /* fft Y (servo position) */
Radix2FFT( powerOfTwo , re3 , im3 );          /* fft W (noise)*/
for(j=0;j<n;j++) Uu2[j]=pow(sqrt( re1[j]*re1[j] + im1[j] * im1[j] ),(double)2);
for(j=0;j<n;j++) Yy2[j]=pow(sqrt( re2[j]*re2[j] + im2[j] * im2[j] ),(double)2);
for(j=0;j<n;j++) Ww2[j]=pow(sqrt( re3[j]*re3[j] + im3[j] * im3[j] ),(double)2);

CrossPower(n,re1,im1,re3,im3,reWU,imWU); /* determine crosspower part S */
CrossPower(n,re2,im2,re3,im3,reWY,imWY); /* determine crosspower part Ps */

for(j=0;j<n/2;j++) PwW[j]=PwW[j]+Ww2[j]; /* calc sum of power spectra */
for(j=0;j<n/2;j++) Puu[j]=Puu[j]+Uu2[j];
for(j=0;j<n/2;j++) Pyy[j]=Pyy[j]+Yy2[j];
for(j=0;j<n/2;j++) PwuRe[j]=PwuRe[j]+reWU[j];
for(j=0;j<n/2;j++) PwuIm[j]=PwuIm[j]+imWU[j];
for(j=0;j<n/2;j++) PwyRe[j]=PwyRe[j]+reWY[j];
for(j=0;j<n/2;j++) PwyIm[j]=PwyIm[j]+imWY[j];
free(im1);
free(im2);
free(im3);
free(reWU);
free(imWU);
free(reWY);
free(imWY);

} printf("transfer calculation\n");

for(i=0;i<n/2;i++) TwuRe[i]=PwuRe[i] / PwW[i]; /* create sensitivity re */
for(i=0;i<n/2;i++) TwuIm[i]=PwuIm[i] / PwW[i]; /* create sensitivity im */

for(i=0;i<n/2;i++) TwyRe[i]=PwyRe[i] / PwW[i]; /* create proces sensitivity re */
for(i=0;i<n/2;i++) TwyIm[i]=PwyIm[i] / PwW[i]; /* create proces sensitivity im */

/* calculation H=Ps/S */
for(i=0;i<n/2;i++) Hre[i]=(TwyRe[i]*TwuRe[i] + TwyIm[i] * TwuIm[i]) / (pow(TwuRe[i],(double)2)+pow(TwuIm[i],(double)2));
for(i=0;i<n/2;i++) Him[i]=(TwyRe[i]*TwuIm[i] + TwyIm[i] * TwuRe[i]) / (pow(TwuRe[i],(double)2)+pow(TwuIm[i],(double)2));

/* ---- transformation Hre en Him to gain and phase ------------------------*/
for (i=0;i<n/2;i++) AMP[i]= fabs(sqrt( pow(Hre[i],(double)2) + pow(Him[i],(double)2)));
for (i=0;i<n/2;i++) PHS[i]= 180/PI * atan2( Him[i] , Hre[i] );

/* --------------------------------------------- */
for(i=0;i<n/2;i++) Cuw[i]= pow(sqrt( PwuRe[i] * PwuRe[i] + PwuIm[i] * PwuIm[i]) , (double)2) / (PwW[i]*Puu[i]); /*create coherence Sensitivity*/

amp = fopen("AMP.dat","w");  /*enable writing of data*/

for(j=0;j<n/2;j++) fprintf(amp,"%f\n",AMP[j]);  /*write amplitude*/

40
phs = fopen("FHS.dat","w"); /*enable writing of data*/
for (j=0; j<n/2; j++) fprintf(phs,"%f\n",PHS[j]); /*write phase*/
fc = fopen("cohere.dat","w"); /*enable writing of data*/
for (j=0; j<n/2; j++) fprintf(fc,"%f\n",Cwu[j]); /*write coherence*/
printf("writing data\n");

/* ---- close data files -----------------------------------------------*/
fclose(dat);
fclose(amp);
fclose(phs);
fclose(fc);

/* ---- free memory allocations ---------------------------------------*/
printf("free memory\n");
free(re1);
free(re2);
free(re3);
free(data1);
free(data2);
free(data3);
free(window);
free(Hre);
free(Him);
free(Ww2);
free(Uu2);
free(Yy2);
free(Pwu);
free(Pyy);
free(Pun);
free(PwuRe);
free(PwuIm);
free(TwuRe);
free(TwuIm);
free(PryRe);
free(PryIm);
free(TwyRe);
free(TwyIm);
free(Cwu);
free(Cwy);
free(C);
B.2 crosspower calculation code

/*
   // NYQUIST Industrial Control
   //
   // Copyright (c) Nyquist B.V. 2005 Eindhoven, The Netherlands
   // Internet: www.nyquist.com
   //
   // Project Name : crosspower calculation
   // Source Name  : crosspower.c
   //
   // Author       : Ramon Solberg
   // Description  : crosspower calculation for spectrum analysis module.
   */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void CrossPower(const int n,double *re1,double *im1,double *re2,double *im2, double *reXY, double *imXY)
{
  int i;

  for(i=0;i<n;i++)
  {
    reXY[i]= re1[i] * re2[i] + im1[i] * im2[i];
    imXY[i]= re1[i] * (-1*im2[i]) + im1[i] * re2[i];
  }
}

B.3 Fourier transformation algorithm code

/*
   // NYQUIST Industrial Control
   //
   // Copyright (c) Nyquist B.V. 2005 Eindhoven, The Netherlands
   // Internet: www.nyquist.com
   //
   // Project Name : fft module
   // Source Name  : fft.c
   //
   // Author       : Istvan Eperjesy / Ramon Solberg
   // Description  : Radix2FFT algorithm for spectrum analysis module.
   */

#include "fft.h"
#include "math.h"

// Radix2FFT fft using Radix2 method
// re: input and output real part,
// im: input and output imaginary part
// powerOfTwo: 2^(powerOfTwo) number of points in re (and also in im)

// example:

void Radix2FFT( int powerOfTwo, double *re, double *im )
{
    int n,i,i1,j,k,i2,l1,l2;
    double cl,c2,tx,ty,t1,t2,u1,u2,z;

    /* Calculate the number of points */
    n = 1 << powerOfTwo;
    //for ( i=0 ; i<powerOfTwo ;i++)
    //    n *= 2;

    /* Do the bit reversal */
    i2 = n >> 1;
    j = 0;

    for (i=0;i<n-1;i++)
    {
        if ( i < j )
        {
            tx = re[i];
            ty = im[i];
            re[i] = re[j];
            im[i] = im[j];
            re[j] = tx;
            im[j] = ty;
        }
        k = i2;
        while (k <= j)
        {
            j -= k;
            k >>= 1;
        }
        j += k;
    }
}


} /* Compute the FFT */
c1 = -1.0;
c2 = 0.0;
l2 = 1;
for ( l=0 ; l < powerOfTwo ; l++)
{
    l1 = l2;
l2 <<= 1;
    u1 = 1.0;
    u2 = 0.0;
    for (j=0;j<l1;j++)
    {
        for (i=j;i<n;i+=l2)
        {
            i1 = i + l1;
            t1 = u1 * re[i1] - u2 * im[i1];
            t2 = u1 * im[i1] + u2 * re[i1];
            re[i1] = re[i] - t1;
            im[i1] = im[i] - t2;
            re[i] += t1;
            im[i] += t2;
        }
        z = u1 * c1 - u2 * c2;
        u2 = u1 * c2 + u2 * c1;
        u1 = z;
    }
    //if ( inverse )
    //    c2 = sqrt((1.0 - c1) / 2.0);
    //else
    //    c2 = -sqrt((1.0 - c1) / 2.0);
    //    c1 = sqrt((1.0 + c1) / 2.0);
}

/* Scaling for forward transform */
//if (!inverse)
//{
//    // for (i=0;i<n;i++) {
//        //    re[i] /= n;
//        //    im[i] /= n;
//    // }
//}
}
Appendix C

Testing NYCeIdent

C.1 Fourth order system

Figure C.1: fourth order system (feedback over motor)
C.2 V.R.S. system

Figure C.2: V.R.S. system (y-axis)
Bibliography


