ISSN 0280–5316
ISRN LUTFD2/TFRT--5708--SE

Multirate Control of a DVD Player

Johan Kaunitz

Department of Automatic Control
Lund Institute of Technology
August 2003

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* August 2003 *Document Number* ISRN LUTFD2/TFRT--5708--SE
Author(s) Johan Kaunitz	*Supervisor* Bo Lincoln, LTH Hans BergHäll, AudioDev AB *Sponsoring organisation*

Title and subtitle

Multirate Control of a DVD Player

Abstract

In this master theses an existing control system of a DVD player has been modified such that it has become more feasible to implement using limited hardware resources. The discrete controller has already been implemented for evaluation purposes using FPGA technology, and is working well. So far however, low resource consuption has not been a major concern. This is gradually changing though, as more features are being planned for the same hardware, potentially taking up more resources than currently available.

The approach taken here is to use high level software tools and a powerful general purpose real-time processor to create a new software implementation of the controller and use it to control the real DVD drive. Using this setup, some interesting design parameters for possible custom hardware implementations has been evaluated under real circumstances. One notable such parameter, which is highly correlated to the size of the resulting FPGA implementation, is the choice of wordlength for the representation of signals, states and coefficients in the controller. To efficiently reduce this parameter, the design of the controller has been altered somewhat to make it less sensitive to the lower numerical accuracies introduced, leading to a multirate controller.

To analyse what impact the suggested changes would have on the performance of the control system, extensive tests has been made with different design parameters. This includes experiments with track-jumps, which are highly dependent on the properties of the controller. The tests have indicated what savings that seem possible to achieve under the prevailing conditions.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280–5316		*ISBN*
Language English	*Number of pages* 1–45	*Recipient's notes*
Security classification		

Sammanfattning

I detta examensarbete har ett reglersystem för en DVD-spelare modifierats så att det bättre lämpar sig att implementeras med begränsade hårdvaruresurser. Den diskreta regulatorn finns redan implementerad med FPGA-teknik i utvärderingssyfte, och fungerar bra. Hittills har dock inte låg resursförbrukning varit av högsta prioritet. Detta håller på att förändras då fler funktioner, som kan behöva mer resurser än vad som finns tillgängliga i nuläget, planeras för samma hårdvara.

Tillvägagångssättet har varit att använda avancerade programvaruverktyg och en kraftfull generell realtidsprocessor för att skapa en ny mjukvaruimplementation av regulatorn och använda denna för att styra den riktiga processen. Med denna uppsättning har sedan vissa intressanta designparametrar för potentiella hårdvaruimplementationer kunnat studeras under verkliga förhållanden. En speciellt intressant sådan parameter, som i hög grad påverkar storleken på den resulterande FPGA-kretsen, är valet av ordlängd för signaler, tillstånd och koefficienter i regulatorn. För att effektivt kunna minska denna parameter har designen av regulatorn behövt ändras något för att göra den mindre känslig mot högre numerisk osäkerhet, vilket lett till en multirateregulator.

För att analysera vilka effekter de föreslagna ändringarna skulle ha på regulatorns prestanda, har tester gjorts med olika designparametrar. Även tester av spårhopp har gjorts då dessa är särskilt beroende av regulatorns egenskaper. Dessa tester har gett en fingervisning om hur stora besparingar som verkar möjliga att uppnå under gällande omständigheter.

Contents

1 Introduction

This theses will address some issues involved with implementing a digital controller for a regular, commercially available, home entertainment DVD player. The reason why a custom controller is needed for this particular DVD player is that *AudioDev* wants to evaluate the possibilities of using a commercially available DVD drive in conjunction with their line of test equipment for CD and DVD manufacturers. In order to make accurate and meaningful measurements with the drive, a custom and well-understood controller is needed since the original controller shipped with the DVD player doesn't provide the level of flexibility required for advanced testing.

A working prototype controller has already been designed and implemented using FPGA-chips. It is however believed that this implementation is somewhat sub-optimal, in the sense that it could probably be implemented using less chip-space. This would ideally make room for other interesting features on the same chip. Currently, the controller is implemented using 32-bit fixed point arithmetics; enough so that surely no significant errors would be introduced from numerical uncertainces. The calculations are executed in an ALU running on one of the FPGA-chips. If the required wordlength of the operands could be lowered, the ALU could then be implemented using fewer gates — in fact, the number of gates needed to implement a multiplier increases almost quadratically with the number of bits in the operands, so this certainly looks like a good target for reducing the size. A positive side effect of a smaller ALU is that it can be clocked at a higher rate, paving way for the possibility of a faster sample rate, should the need arise.

To begin with, some background information is provided that introduces the DVD drive and the properties of the DVD media, as well as the design and the features of the current controller. Then the problem is formulated and analyzed and some solutions are presented, implemented and tested. Finally, the results are discussed and some conclusions are drawn. Also, some suggestions for further work are made.

2 Background

2.1 Drive

A typical DVD player is built using low-cost mechanical, optical and electrical components and is highly dependent on automatic control and advanced coding techniques to function. A sketch of the physical parts of a DVD system can be seen in Figure 1.

The data is recorded as so-called lands and pits of different lengths (see Figure 2) on the disc. It is arranged in a track-spiral going from the center of the disc all the way towards the edge, as can be seen in Figure 1. Data is extracted by rotating the disc and sending a laser-beam towards the surface of the disc. By studying the amount of light reflected from the track as lands and pits pass by, the recorded data-stream can be recreated. To protect against occasional errors

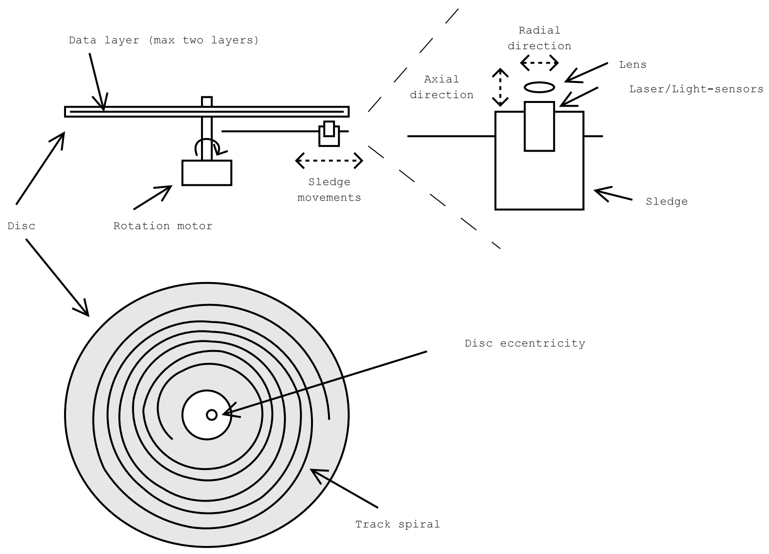

Figure 1: Sketch of the physical parts that constitute a DVD system.

in the data-stream, caused by e.g. scratches on the surface, the data contains redundancy in the form of error-correcting codes. The lens' position has to be continously adjusted using automatic control in order to stay on track and in focus.

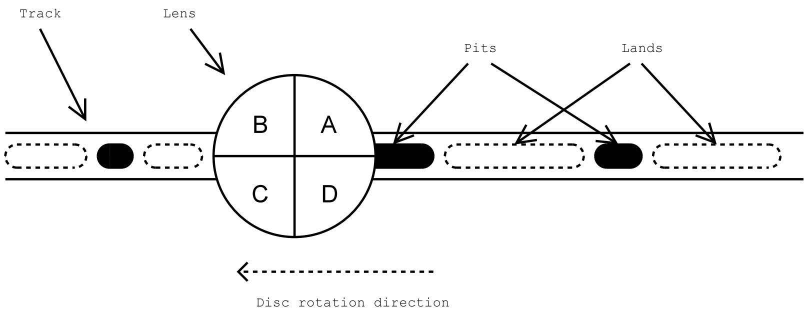

Figure 2: Illustration of the lens while reading data from a track.

The surface of the lens is actually divided in four areas, as shown in Figure 2, and has a separate light-sensor attached to each one of these. By studying the relative differences in the amount of light recieved from each sensor, it is possible to deduce all information needed to control the lens movements both in the axial and radial directions. The signals derived from these sensors are summarized in table 1. Note that the lens is actually astigmatic and has two different focuses. At the optimal working point the lens should be positioned in the axial direction such that the data-layer of the disc is located exactly in the middle of the two different focuses of the lens, i.e. neither one is actually *in* focus. This scheme makes it possible to detect diversions in the axial direction from the ideal working point, so that a servo can be designed to keep the lens around that point.

Name	*Description*
HF	High frequency data carrying signal constructed as $A+B+C+D$. This signal is sliced and sent to a decoder to extract the data-stream from the disc.
TCS	Track Crossing Signal, derived as HF filtered through a low-pass filter. Used to detect track crossings when starting the radial servo.
FE	Focus Error, i.e. the offset from the center of the focus points. Calculated as $A + C - B - D$, where A and C have a separate focus point from B and D.
DPD	The offset of the lens position from the center of the track. Generated by some processing (called Differential Phase Detection) of the signals $A + C$ and $B + D$. Referred to as RE (Radial Error) from now on.

Table 1: Available output signals from the DVD drive.

The inputs available to control the DVD system is summarized in table 2.

Name	*Description*
u_{rad}	Moves the lens in the radial direction by adjusting the strength of a magnetic field surrounding the lens.
u_{foc}	Moves the lens in the axial direction by adjusting the strength of another magnetic field surrounding the lens.
u_{rot}	Controls the speed of the electrical rotation motor.
u_{sledge}	Controls an electrical motor that can move the sledge across the disc.

Table 2: Available input signals to the DVD drive.

2.2 Controller

The control system of the DVD player has four separate control loops, described in short below:

Focus servo Makes the lens stay focused on a specific layer by keeping the *Focus Error* signal (FE) near zero.

Radial servo Makes the lens follow a track by keeping the *Radial Error* signal (RE) near zero.

Sledge servo Makes the sledge follow the tracks (slowly) as the lens can only see a few hundred tracks sideways from a given static position of the sledge.

Rotation servo Keeps the information flow constant by continously adjusting the rotation speed of the disc depending on the radial position of the pickup. This mode is called CLV, *Constant Linear Velocity*, as opposed to the less commonly used alternative CAV, *Constant Angular Velocity*, where the disc rotates at a constant speed independent of radial position.

To begin reading from a disc, the startup of these controllers has to be synchronized in the following way:

1. Begin rotating the disc at a constant speed (CAV mode) by applying a constant voltage to the rotation motor.
2. Find the focus point by moving the lens to an end-point position, either as close to the surface of the disc as possible or as far away from it as possible, depending on which layer (if the disc is a dual-layer disc, otherwise it does not matter) to focus on. Then slowly move to the center while keeping FE under observation. When the lens is out of focus, FE will be practically zero, but when a layer starts coming into focus FE will begin making an S-curve as can be seen in Figure 3. When this is detected, the focus servo can be turned on and the lens will be locked in on the layer.

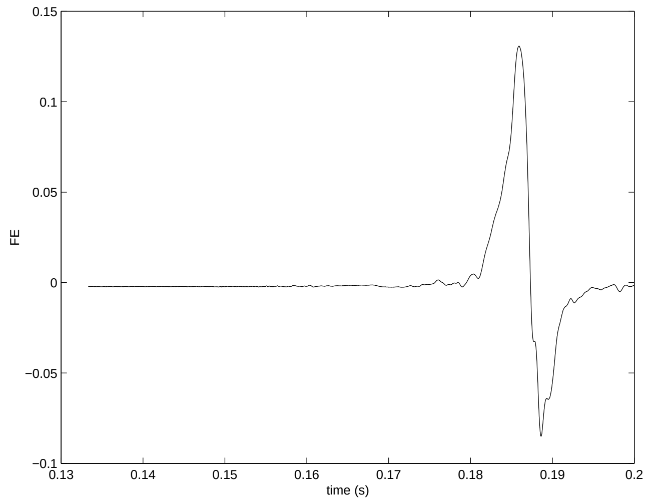

Figure 3: S-curve used to detect a focus-layer before activating the servo. The linear area lies in between the positive and negative peaks.

3. To lock the radial servo to a track, the controller monitors the RE signal as tracks are passing by under the lens, and waits for the particularly favourable situation when tracks are passing by at the slowest possible rate. See Figure 4 for an example of this. Waiting for this event gives the highest probability of successfully catching the track when the radial servo is turned on. This heuristic is actually not sufficient; the controller also needs to observe the *Track Crossing Signal* (TCS) to make sure a track is actually present at a slow passage.
4. When locked on to a track, the decoder can start doing meaningful interpretations of the HF signal and, among other things, extract sector

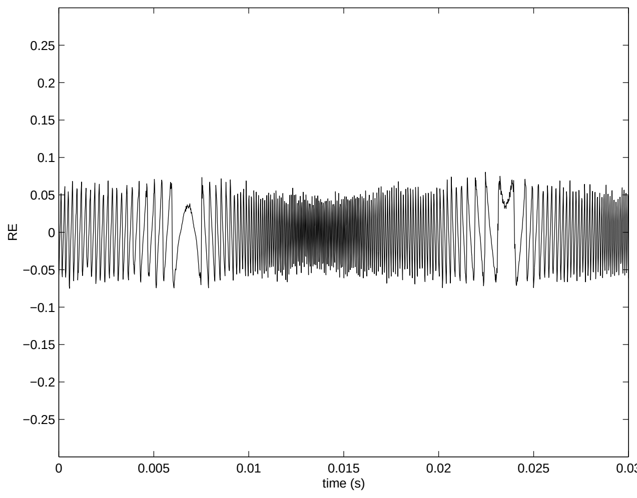

Figure 4: Appearance of RE as tracks passes by while focusing on a specific layer. Illustrates how the disc eccentricity makes tracks appear slower or faster.

information from the data-stream. To get a constant data-rate, the rotation motor can be switched to CLV mode.

2.2.1 Radial servo

A second order linear model from u_{rad} to RE exists, and the corresponding bode plot is included in Figure 5.

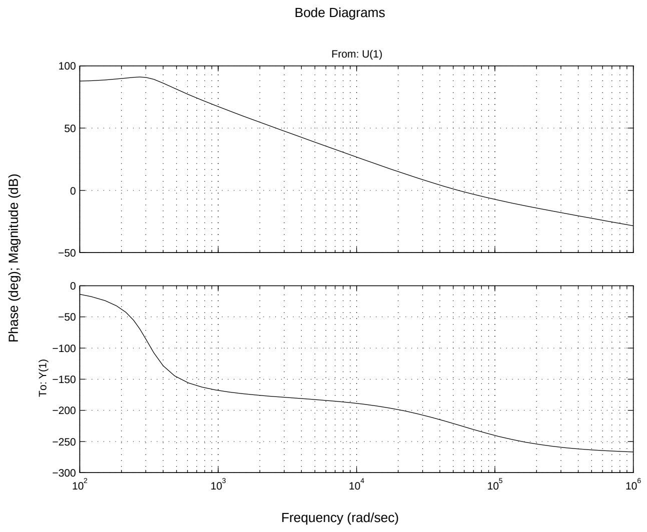

Figure 5: Bode plot of the lens process in the radial direction.

The radial servo is designed to follow the DVD specification, standard ECMA-267. According to this specification, the bandwidth of the open loop system should be around 2.4 kHz. This can be accomplished by doing linear feedback on a reconstruction of the two states of the process. A model is provided in Figure 6.

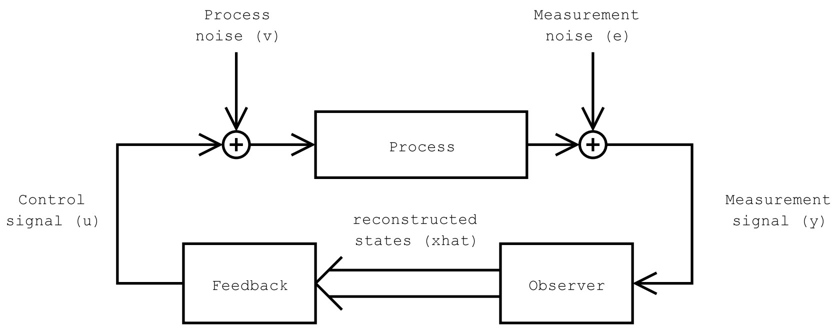

Figure 6: Model of the system without oscillative disturbance.

The reconstruction is made with an optimal Kalman filter K (with direct action, hence $\bar{K}$ below).

$$\begin{aligned} \hat{x}(k+1|k) &= \Phi\hat{x}(k|k-1) + \Gamma u(k) + K\left(y(k) - C\hat{x}(k|k-1)\right) \\ \hat{x}(k|k) &= \hat{x}(k|k-1) + \bar{K}\left(y(k) - C\hat{x}(k|k-1)\right) \end{aligned} \tag{1}$$

The Kalman filter is derived by taking the properties of the noise that is acting on the system into account, and it is optimal in the sense that it minimizes a criterion expressing the variance of the reconstruction error. The equations (2) describe the process

$$\left\{ \begin{aligned} x(k+1) &= \Phi x(k) + \Gamma(u(k) + v(k)) \\ y(k) &= Cx(k) + e(k) \end{aligned} \right. \tag{2}$$

where v and e are discrete-time Gaussian white-noise processes with zero-mean value and variances as defined below.

$$E\{v(k)v^T(k)\} = R_1 \tag{3}$$
$$E\{v(k)e^T(k)\} = R_{12} \tag{4}$$
$$E\{e(k)e^T(k)\} = R_2 \tag{5}$$

The reconstruction error is defined as $\tilde{x} = x - \hat{x}$, and the expression which is minimized by the optimal Kalman filter is

$$P(k) = E\left\{\tilde{x}(k)\tilde{x}^T(k)\right\}. \tag{6}$$

The design of the observer when using this method lies in specifying the relative differences between the noise variances. In our case, the process noise v is modelled as acting on the input of the system, and is thus one-dimensional and going through the same Γ matrix as the input signal. Further, since y is one-dimensional, so is e. Assuming that the covariance between v and e is zero ($R_{12} = 0$) left are two parameters, namely the scalar matrices R_1 and R_2. However, since the variances are only relative, this is effectively one parameter which needs to be specified. For this process, reasonable performance has been achieved by selecting R_1 an order of magnitude less than R_2.

The feedback is then

$$u(k) = -L\hat{x}(k|k) \tag{7}$$

where L is selected using LQG design, see e.g. [1]. Using this method, the feedback law (7) will minimize a loss function

$$J = \sum_{k=0}^{\infty} \left(x^T(k) Q_1 x(k) + u^T(k) Q_2 u(k) \right) \tag{8}$$

where Q_1 and Q_2 are design parameters. These matrices are relative weights selected such that an element with a high value suppresses a state (or signal). There is no practical limit on how quickly the lens can be moved and thus the penalty of the control signal, Q_2, can be very low. Furthermore, since the main objective is to keep the radial error ($= y$) near zero, Q_1 can rather intuitively be chosen as $C^T C$, which gives a controller that minimizes $\sum y^2$, i.e. the power of RE. This might lead to a quite aggressive controller though. To make it behave better, the loss function could be extended to penalize the speed of y as well.

Although this controller probably would be able to follow a track, depending on the exact design and the condition of the disc, it would have problems with a low frequency disturbance at the rotation frequency (10-20 Hz), originating from the disc eccentricity of up to 100 tracks sideways. To handle this, a disturbance model of the track offset that is driven by white noise is included in the controller, and the Kalman filter is extended to estimate the current offset so that the output from the radial servo can be adjusted accordingly. A model of the new situation is provided in Figure 7.

The disturbance is modelled as a second order linear system

$$\left\{ \begin{array}{rcl} z(k+1) & = & \Phi_o z(k) + \Gamma_o w(k) \\ y(k) & = & C_o z(k) \end{array} \right. \tag{9}$$

where the eigenvalues of Φ_o is chosen as two poorly damped complex conjugate poles at the rotation frequency of the disc (≈ 14 Hz).

The observer (1) is thus modified to estimate all the states

$$x_e = \begin{bmatrix} x \\ z \end{bmatrix}$$

from the extended system

$$\left\{ \begin{array}{rcl} x_e(k+1) & = & \Phi_e x_e(k) + \Gamma_e u(k) + \Gamma_e v(k) + \Gamma_w w(k) \\ y(k) & = & C_e x_e(k) + e(k) \end{array} \right. \tag{10}$$

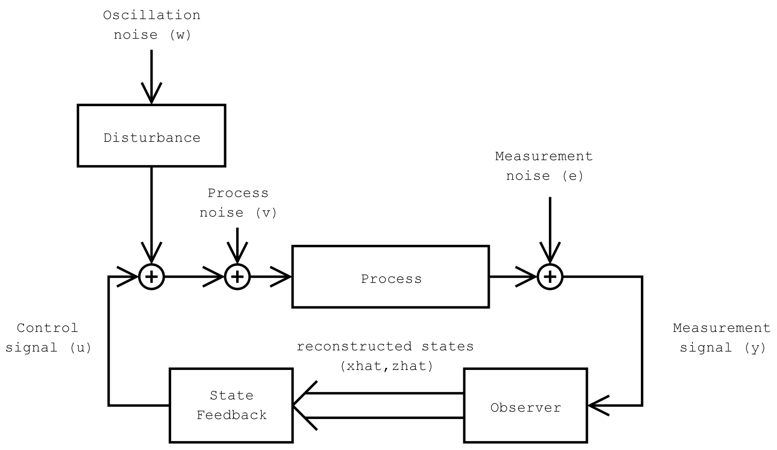

Figure 7: Model of the system with oscillative disturbance.

where

$$\Phi_e = \begin{bmatrix} \Phi & \Gamma C_o \\ 0 & \Phi_o \end{bmatrix},\ \Gamma_e = \begin{bmatrix} \Gamma \\ 0 \end{bmatrix},\ \Gamma_w = \begin{bmatrix} 0 \\ \Gamma_o \end{bmatrix} \text{ and } C_e = \begin{bmatrix} C & C_o \end{bmatrix}.$$

This adds a new parameter to the design of the Kalman filter, namely the variance of the noise w driving the disc eccentricity, relative to the variances of the process and measurement noises v and e, respectively.

2.2.2 Focus servo

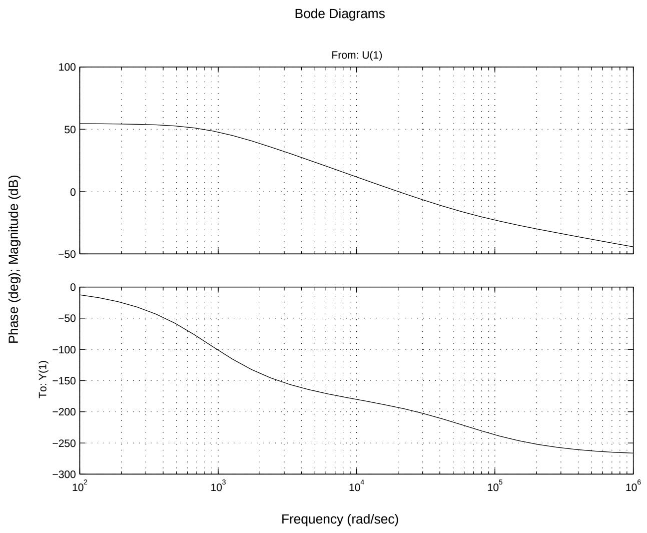

Figure 8: Bode plot of the lens process in the axial direction.

The servo for the focus process is designed using the same methods as described for the radial servo in Section 2.2.1. The bode plot of the process can be seen in Figure 8.

2.2.3 Sledge servo

The sledge is where the laser and lens (a.k.a. the pickup) are mounted, and it can be moved by an electrical motor across the disc. The lens needs to be positioned within about one hundred tracks of the track currently being read from on the disc, and thus the sledge needs to slowly follow the lens as it follows a track. This is done by using the output from the radial controller (u_{rad}) as input to a slow PI controller that tries to make this signal equal to zero by moving the sledge. When u_{rad} has a large low-frequency component this means the lens is bending sideways to compensate for a non-ideal position of the sledge.

In practice, the sledge suffers from a lot of friction and will therefore move in a very jerky fashion. As a matter of fact, feeding the sledge motor with a small square wave overlayed onto the control signal gives a more reliable mode of operation. This will periodically snatch the sledge out of situations where it is temporarily stuck, preventing the controller from building up too much force.

2.2.4 Track jumping

For a DVD player to be able to quickly locate the correct video sequence, audio track or data file on a disc, it is not sufficient just to be able to sequentially read data from the beginning of the disc to the end. Instead, by moving the lens sideways, perpendicular to the tracks, it is possible to very quickly jump to a new position on the disc. This is known as *Random Access*, and is one of the many advantages a DVD disc has over e.g. a VHS cassette.

The algorithm for jumps to adjacent tracks[1] (short jumps), is described below:

1. The controller is switched to another one, where the speed of the lens movements is more heavily punished so that it can handle a step in the reference signal. The observer is also modified to better estimate the states with less information from the input signals.
2. A new set-point is set for RE, i.e. the lens is told to move *away* from the track (in the direction of the jump).
3. When the lens has moved away a bit from the track, the controller is switched to run in open loop[2], since the system has entered its non-linear area. To successfully be able to control the lens during this period it is important that the model sufficiently describes the state of the system (e.g. its current eccentricity).
4. As soon as the new track starts appearing under the lens, the controller is switched back to closed loop. At the same time, the states of the controller

[1]Physical tracks, not logical (e.g. audio) tracks.
[2]The observer is told to disregard the measurements.

is reset so that it appears to the observer as if the new track is the same as the old.

5. If several tracks should be jumped, goto 2.
6. Switch back to the normal controller.

To do long jumps, say over several hundred tracks or so where the lens cannot see the new position from its current position, the sledge has to be moved. This can be done by turning off the radial controller completely (the focus controller can still be running, though) and moving the sledge as close to the new position as possible, given the available accuracy of the sledge and its electrical motor. Then the radial controller is turned back on, and the pickup's new position on the disc can be read from the sector information and compared to the position we want to find. However, the sledge can only move the pickup whithin about one hundred tracks of the target position, and some short jumping (described above) will most likely be necessary to find the exact position.

2.3 Media

2.3.1 Disc parameters

The DVD media is an important part of the whole system and its properties can have significant impact on the actual performance of the control system. The discs have several physical parameters which affect how the disc performs as a part of the complete DVD system. The DVD specification puts limits on these parameters, and it is *AudioDev's* business plan to provide disc manufacturers with equipment to measure all these parameters so that they can achieve compliance with the DVD specifications. Some important parameters are summarised in table 3.

Name	*Description*
Radial Noise	Remaining noise in the *RE*-signal when following a track using a reference servo.
Jitter	Small variations in the lengths of individual lands and pits, compared to their ideal lengths. If the jitter gets to large, the probability of errors in the extracted data increases.
Assymetry	Differences between the average signal-level of short lands/pits compared to long lands/pits. If this difference gets too big the data cannot be reconstructed satisfactory.

Table 3: Examples of some disc parameters that can be measured by *AudioDev's* products.

2.3.2 Influence on the control system

From the controller's point of view, one important property of a specific disc is how it affects the gain of the process, i.e. how much light that are being reflected back to the sensors from the surface. These differences in gain can be quite large; especially between single- and dual-layered DVD's, where the dual-layered disc only reflects about half of the light compared to its single-layered counterpart. Even different areas of the same disc can have significant differences in gain. To manage this, the gains of the radial- and focus-processes are estimated at certain points during runtime by injecting sinus-waves with low amplitudes at sensitive frequencies of each closed loop. Then the real gain of each control loop can be measured and adjusted accordingly. The disc, as modelled, does not have any dynamics of its own, so with the exception of the gain our process will be the same irrespective of the actual media.

The processes are however exposed to disturbances and noise originating from individual characteristics of each disc, and this highly affects how well the control system will be able to follow a track and extract data. First and foremost, the eccentricity of the disc as well as its tendency to wobble adds a low frequency sinusoidal disturbance to both the radial- and focus-process which needs to be addressed by the servos. Furthermore, the signal to noise ratio (SNR) of the measurements used for control varies somewhat between discs, and although a controller can be designed that works good on most available discs, it has proven to be quite difficult to cover all cases since some discs do not follow the specifications very well.

3 Material

3.1 Toolchain

An implementation of the controllers above, including short track jumping but no long jumps, exists as a Simulink model. Its design is specified in a number of MATLAB scripts. This implementation can be compiled using Real Time Workshop into a program executable in the dSPACE environment. The dSPACE environment consists of an expansion card for an ordinary PC, containing a standard PowerPC processor and several A/D- and D/A-converters, as well as software for creating and running custom user interfaces connected to the real-time tasks executing on the PowerPC processor.

The toolchain used for the development of the controller is explained in detail below:

- The design of the different controllers is specified in several MATLAB M-files, which needs to be executed prior to opening the Simulink model.
- The implementation is done as a Simulink model, with only discrete states and using a singletasking fixed-step solver. By utilizing special dSPACE blocks, for the A/D- and D/A-converters, the controller's input and output signals can be connected to the real process' sensors and actuators.

- Using Real Time Workshop, standard C-code can be generated from the Simulink model, which can further be compiled into a PowerPC executable for the dSPACE processor board. Double precision floating-point calculations are used by default in Simulink and also in the generated executable; and it is in fact not possible to change that easily to e.g. fixed point calculations instead.
- The generated application can be loaded into the real-time system either manually from the dSPACE Workbench[3], or automatically as a part of the build process in MATLAB.
- A custom user interface can be designed that lets the user change parameters, control the flow of execution and also study signals during execution in what can be described as virtual oscilloscopes.

4 Problem formulation

The problem is to evaluate some modifications of the current controller that would allow it to be implemented using less chip-space. The modifications should be such that the properties and the performance of the current design are retained. The main approach is to target the current, overly accurate, numerical representation of states and coefficients in the controller so that a reduction of the ALU's size is possible.

The controller to be used as a starting point is available in the MATLAB/dSPACE-environment described above and it can be executed in real-time connected to the real DVD drive via analog I/O. This setup allows for much more rapid development iterations and much higher flexibility than i.e. reprogramming of an FPGA would. However, some care has to be taken to make sure that the results will be the same as when the controller is later re-implemented in an FPGA, e.g. the dSPACE setup will add a delay of almost one full sample to the system, which will not be present when the controller is running in an FPGA.

This work will focus on analyzing the radial-servo and how it can be implemented more efficiently, since that is one part of the system which is known to need high numerical accuracy in the current implementation. The reason for this will be explained below.

5 Analysis

5.1 Prerequisites

The first thing that needs to be changed with the current model is to make it use fixed-point calculations instead. Then the wordlength can be introduced as a parameter; or rather, the *resolution* to be used for calculations in a potential FPGA implementation. Resolution is defined as the smallest non-zero

[3]A Windows program for managing tasks on the PowerPC processor, as well as creating and running graphical user interfaces on the host computer connected to these tasks.

magnitude representable and it depends on the number of bits that are used to represent the fractional part of a real number. If n fractional bits are used, the resolution is $1/2^n$. The *accuracy* is then defined as the maximum difference between a real value and its representation. The accuracy is always equal to the resolution divided by two for fixed point numbers.

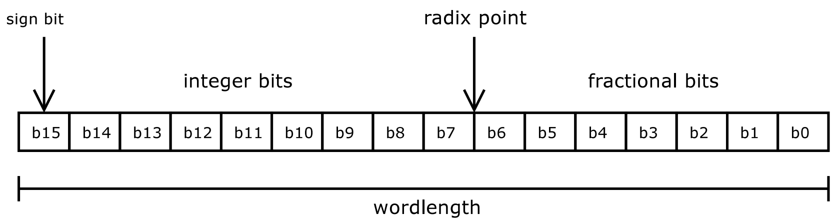

Figure 9: Example of a fixed point word with an integer part and a fractional part.

Figure 9 shows an example of a 16-bit fixed-point word. The bits are divided into one sign bit, 8 integer bits and 7 fractional bits. This gives a resolution of $1/2^7 = 1/128 = 0.0078125$, i.e. words of this particular structure can represent numbers that are integer multiples of 0.0078125. The range of the numbers that can be represented is, of course, limited. In the case of signed arithmethics it is common to use a scheme called "two's complement" for dealing with negative numbers. The range for the example above will then be $\left(-2^9, 2^9 - 1/2^7\right) = (-512, 511.9921875)$.

The minimum wordlength required to do the calculations of a given system depends both on the selected resolution and on the required range to cover all possible state values. The resolution (i.e. the number of fractional bits) is the natural design parameter, and the required range can for example be obtained by monitoring the maximum state values during normal operation of a system.

Unfortunately, as mentioned before, Simulink only works with floating point arithmetics and it is therefore not possible to introduce these modifications using standard Simulink blocks. The solution is to extend the capabilities of Simulink using so-called S-functions. These can be implemented in C, thus presenting the possibility of having sufficiently fine-grained control of the calculations such that the conditions inside a custom ALU can be simulated. Keep in mind that when doing a custom hardware implementation using e.g. FPGAs, the wordlength of the ALU can be selected as any arbitrary integer, not just the standard lengths of powers of two.

The implementation of these modifications are discussed in detail in Section 6.

5.2 Naive approach

The naive approach would be to use the new implementation of the original controller, where the resolution is available as a design parameter, and study its behaviour when operating using different resolutions. The results of this experiment are discussed in Section 7.1.

5.3 Multirate observer

The controller for the radial servo contains a Kalman observer for the disc eccentricity. This is modelled as a poorly damped oscillative disturbance at the rotation frequency (10-20 Hz), which is very low compared to the sample rate of 60 kHz. This leads to a system with poles very near the unit circle, needing high numerical precision for accurate representation. The reason for this is that when a slow dynamic process is sampled with a high sample rate, the dynamics do not change very much between occasional samples but rather on a much longer time scale. So in practice, the states stay almost identical between fast sample hits (the system matrix can be thought of as being approximately equal to the identity matrix), only changing with very small relative amounts each time. The current controller, for example, has time for about 4300 samples as the disc makes one revolution. In the real world, this means that only the least significant bits of these states are changing on such a fast scale, the very bits that are the target for elimination. For our purposes this is likely to be a major bottleneck in limiting the overall required wordlength. To tackle this, a multirate approach is suggested. By dividing the controller into subsystems operating at different sample rates, more in line with their individual dynamic properties, we get a total system which hopefully can be implemented satisfactory with lower numerical precision than before.

The observer for the slow disturbance states, z, can be isolated from the extended observer for x_e outlined in Section 2.2.1 as

$$\hat{z}(k+1) = \Phi_o \hat{z}(k) + K_2 \overbrace{(y(k) - C_e \hat{x}_e(k))}^{\varepsilon(k)} \tag{11}$$

using the notations introduced in that section, with the addition that K_2 is the part of the Kalman filter derived for (10) that updates only the disturbance states. A model of the system where the observer is split in two parts is included in Figure 10.

The system (11) can easily be resampled using MATLAB's `d2d` command. With the original sample rate, the system-matrix for the slow observer is

$$\Phi_o = \begin{pmatrix} 0.9995 & 0.0015 \\ -0.0015 & 0.9995 \end{pmatrix}$$

and it is very close to the identity matrix. If instead the system is downsampled 50 times, the system-matrix wil be

$$\Phi_o = \begin{pmatrix} 0.9727 & 0.0714 \\ -0.00714 & 0.9727 \end{pmatrix}$$

which has nicer numerical properties.

If the sample time for updating the slow system is chosen as a multiple of the fast sample time, we will get a linear and time invariant system at the common sample instances. Exactly what this multiple should be chosen as is a tradeoff between lower accuracy in the reconstruction of the states, leading to noisier control and perhaps difficulties when doing track jumps, and not needing as many bits in the implementation. Intuitively, the slow system should be

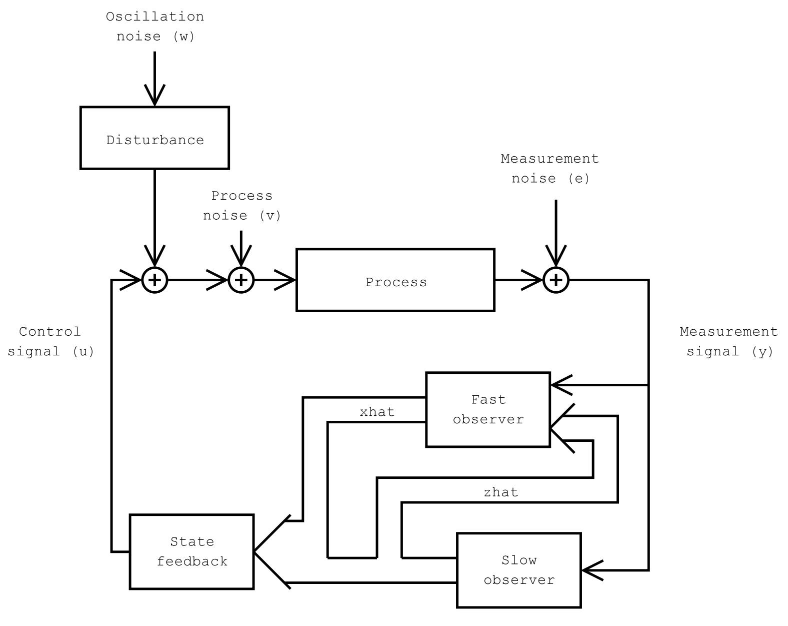

Figure 10: Model of the system with a split observer.

resampled such that its poles become equally sensitive to numerics as the faster subsystem's poles.

The results of experimenting with different resamplings of the slow observer, and how it affects the possibility of lowering the number of bits to represent it, are discussed in Section 7.2.

5.4 Extended multirate observer

When the observer for the low frequency oscillative disturbance is downsampled, valuable information is lost from the measurements in between the slow samples, and experiments showed that performance was not nearly as good as for the original controller when the slow system was downsampled more than about 50 times, even when high numerical accuracy is used. A few ideas on how to modify the observer, with the goal of reclaiming some of the performance lost due to wasted information, is therefore presented.

With the motivation that the dynamics of the slow observer behaves almost as I between fast samples, the following reasoning can be made. At the slow sample instances, the recursion of the states is made with a numerically appropriate system matrix, as explained previously. However, instead of only using a single measurement as input to the system at each slow sample, the sum of all measurements since the last slow sample could be used.

$$\hat{z}(k+N) = \Phi_o \hat{z}(k) + K_2(\varepsilon(k) + \varepsilon(k+1) + \cdots + \varepsilon(k+N-1)) \tag{12}$$

This implies that K_2 should not be part of the resampling, but instead kept in the original version designed for the single-rate observer. The notation used above means that the system is N times slower than the original rate, and that at each slow sample instance, it moves forward the equivalent of N fast steps.

Initial experiments using this approach indicated that the observer gets unstable when downsampling the slow system more than about 10 times. To verify this, a time-invariant description of the combined observer must be obtained. This can be done by only considering the system variables at the common sample instances, but including the fast states one time for each fast sample in a slow period. The state-vector at the common sample instances will then look like

$$\hat{x}_m(k) = \begin{bmatrix} \hat{x}(k-N+1) \\ \hat{x}(k-N+2) \\ \vdots \\ \hat{x}(k) \\ \hat{z}(k) \end{bmatrix} \tag{13}$$

i.e. at each slow sample $[\ldots, k, k+N, k+2N, \ldots]$, $\hat{x}_m$ will contain the N most recent fast updates of the states $\hat{x}$, plus of course $\hat{z}$. If the fast observer for x is described as

$$\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + K_1\varepsilon(k)$$

and the slow observer for z as (12), the dynamics of the combined multirate observer can be described by the following system matrix.

$$\begin{bmatrix} 0 & \cdots & 0 & (\Phi - K_1C) & -K_1C_o \\ 0 & \cdots & 0 & (\Phi - K_1C)^2 & -(\Phi - K_1C + I)K_1C_o \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & (\Phi - K_1C)^N & -((\Phi - K_1C)^{(N-1)} + \cdots + I)K_1C_o \\ -K_2C & \cdots & -K_2C & -K_2C & \Phi_o - NK_2C_o \end{bmatrix}$$

This matrix is actually stable for N up to at least several hundreds (it has not been verified that it is stable for all N). The reason why the original experiments showed instability is not known, but it is very likely that some programming error caused this. It did however initiate a search for other solutions based on applying Kalman theory directly for the multirate system instead of relying on the approximation above. Doing so would guarantee the stability of the observer, it was believed.

So, the question is whether it is possible to make optimal use (in the Kalman sense) of the information available in the measurements. Intuitively, one might suspect that measurements closer to a new slow sample are more valuable than measurements occuring in the beginning of a period. MATLAB's implementation of the Kalman algorithm only handles single-rate systems however, and unfortunately it is not as easy as to just derive two seperate filters for each subsystem independently, since the algorithm requires that all noise that acts on the system being observed is white. Thus the filter has to be derived for the system as a single unit.

One approach, which unfortunately did not work as the author initially hoped, was to use a similar state description as (13) to formulate a Kalman observer for the system that describes the multirate equivalent of (10) at the common sample instances. This system could look something like

$$\left\{\begin{array}{rcl} x_m(k+N) & = & \Phi_m x_m(k) + \Gamma_m u(k) + \Gamma_m v(k) + \Gamma_{mw} w(k) \\ y_m(k) & = & C_m x_m(k) + e(k) \end{array}\right. \tag{14}$$

where $u(k)$, $v(k)$ and $w(k)$ now act as multiple input signals to the system, and y_m act as multiple output signals. That is, $u(k)$ above is really

$$\left[\begin{array}{cccc} u(k-N+1) & u(k-N+2) & \cdots & u(k) \end{array}\right]$$

and so forth for the other signals. When using MATLAB's `kalman` command, it was expected to return a matrix K, where the last row would contain the filter to be used for updating the slow states z from all measurements (which is exactly what is being sought for); and the other rows would describe how to update the fast states (possibly differently) at each fast sample between two slow samples. As the reader might suspect, the update laws for the fast states is not possible to implement since they will not be causal. The obvious flaw is that there is no way to tell the `kalman` command that we want to update the fast states *during* a slow sample using only the current measurement, and alas, the rows of K tell us to also use measurements not only from the past but also from the future (which would, obviously, have improved on the estimates).

Instead of trying to find a completely new optimal Kalman filter for both the fast and the slow system, it would perhaps be sufficient to find an update law for an estimation of the slow system that uses all measurements optimally; given that an observer for the fast system already existed. This solution would also have the advantage of not requiring an update law for the fast observer that varies periodically between slow samples, as the truly optimal solution suggests. The observer for the single-rate system (10) could for instance be used to obtain update laws for the fast states. If the Kalman algorithm could be modified to *not* produce a filter for the fast system, but instead use the ones provided and only seek a solution for the slow system, then a formulation like (14) could be used to find a solution. This route has not been pursued further due to lack of time and capabilities, so it is not clear whether it is possible to do such modifications of the Kalman algorithm or not. Also, as new experiments showed that performance was very satisfactory using only simple accumulation of measurements for the updates, the need of other solutions diminished.

5.5 Performance

One problem that arises from the problem formulation is how to decide if a controller is good enough. Since the overall design of the controller will not be modified, but merely the implementation, its behaviour is expected to be more or less the same. The ramifications of lowering the resolution of how coefficients and states are represented, is that poles and zeros will move slightly from their nominal positions, and also that noise will be added to the system originating from truncation of states and signals. The downsampling in the multirate observers will in itself also add some noise to the system. By studying the variance of the measurements of e.g. the *RE* signal, comparisons can be made between different controllers. To get more easily comparable quantities, a non-squared measurement may be preferred, like the standard deviation or the absolute value of a high-pass filtered *RE* (to remove offsets in the signal), filtered through a low-pass filter. The latter was used during earlier development with the dSPACE setup, and was kept for measuring performance in the experiments presented in this work.

It is somewhat risky to make comparisons of these performance measurements between different experiments, or even in the same experiment if it covers more than a small part of a single disc, since local properties of the disc will affect these numbers. This is a major reason to introduce the resolution of calculations as a run-time parameter that can be changed during online control, since it would then be much easier to directly see where larger quantization steps correlate to degradation in performance.

Another thing that needs to be verified for modified controllers is their abilities to handle track jumps.

6 Implementation

6.1 S-functions

S-functions can be included seamlessly into Simulink models and are linked to the target executables generated by Real Time Workshop during the build process. S-functions have the following basic features:

- They can have an arbitrary number of input parameters that can be used for passing system matrices, sample times etc to the S-function.
- They can run at multiple sample rates, so that a single S-function can be used to update a multirate system.
- They can have an arbitrary number of input and output signals.

When using dSPACE one can also take advantage of the possibility of directly accessing global variables declared in the S-functions from whithin the dSPACE Workbench at run-time. This can for example be used to control the flow of execution in the system.

6.2 Fixed-point arithmetics

The current ALU uses 32-bit operands and has a 64-bit internal accumulator. This means that the ALU stores intermediate results with double precision compared to the operands during, for example, a calculation of a scalar product. When accessing the final result, only one truncation is made to 32 bits instead of one truncation per multiplication, yielding some gain in precision. It would be desirable that the new implementation could simulate the current behaviour for verification purposes, so 32 bits defines the upper limit of what the new implementation should be able to handle. To imitate these conditions in C, we would actually need to work with 64-bit integers, since there is no way to access the high word of the result when multiplying two integers (it is not possible to detect overflows either). Even if the target processor has a 32-bit architecture (like the PowerPC on the dSPACE board), the compiler sometime provides primitives for 64-bit integers (often called *long long int* on 32-bit architectures). This, unfortunately, is not the case with the compiler that is bundled with dSPACE (the compiler for the host system can manage it however, which lead to some confusion during development). Instead, the following alternatives are possible:

- Write a custom multiply-and-accumulate routine in assembler for the target processer. When doing calculations on this low level, the high word is, of course, available after a multiplication has executed. The advantages of this solution are:
  - This is the only solution that could cover all cases exactly as the FPGA does today, since 64-bit integers are not provided by the C compiler.
  - A pure integer implementation would probably be faster than the MATLAB-generated floating point code.

  And the disadvatages:
  - Would require that the author learned PowerPC assembler.
  - Decreased flexibility, e.g. the code would not any longer be compilable for the host (i386) computer in debugging purposes.
  - Increased complexity, i.e. it would probably be harder to understand and maintain the code.
- Only use standard (32-bit) integers, and truncate at each multiplication. Pros:
  - Very easy to implement.
  - Fast (see above).

  Cons:
  - Truncation errors would be different (and larger) than in the FPGA, counteracting the whole purpose of doing a custom implementation to resemble the operations of the FPGA. Although this difference would arguably be quite insignificant, it would be nice to have an implementation that works *exactly* as the one being simulated.
- Use standard (32-bit) integers, but only allow the maximum of 16 bits for the representation of states and coefficients etc., and reserve the other half of the word for storing intermediate results as explained above. The pros are:
  - Easy to implement.
  - Fast.
  - Works as intended, but only up to 16 bits.

Cons:

  - Even though the goal is to reduce the wordlength of the variables in the controller, it is perhaps a bit optimistic to expect that such a large reduction would be possible without difficulties. And even if it were, it would certainly be nice to be able to compare the behaviour with that of a controller using additional bits.

- Use 64-bit floating point (IEEE) variables and make sure they are truncated at the right positions during the flow of execution to perfectly simulate the behaviour of integers. Actually, since the exponent would not be of any use, but only the mantissa, only 53 bits would be available to simulate integers. However, even if this solution were to be used to calculate the scalar product of 32-bit integers, a large number of terms would be needed before the difference between a 53-bit and a 64-bit accumulator would have any impact on the final result. The pros of this solution is:

  - Has the correct behaviour for integers up to 32 bits wide, if the number of terms in scalar products is limited.

  And the cons:

  - Slow, even slower than regular floating point calculations since explicit truncations are needed frequently. Also, since the shift operations, `<<` and `>>`, are not valid for floating point variables, these must be replaced with expensive multiplications and divisions.

To mimic the current behaviour of the ALU, either the first or the last solution must be used. Since the assembler alternative is not very appealing, the last approach was the first attempt. The implementation of that solution consists of the three macros

```
#define L(x,fb) ((real_T)((int)((x)*(1<<(fb)))))
#define U(x,fb) ((real_T)((int)((x)/(1<<(fb)))))
#define O(x,fb) ((real_T)(x)/(1<<(fb)))
```

and they are used to perform calculations in the following way:

- A real signal `x` is truncated to the requested resolution of `fb` bits by using the macro `L(x,fb)`. This function multiplies the real number `x` so that the requested resolution of `fb` fractional bits is contained in the integer part of the real number. Then it is truncated by casting the real number to an integer. After that, the now truncated integer is casted back to a real number, which will be the type of the expression.

- When doing multiplications, the product is truncated by using the macro `U(x,fb)`. This macro cancels the `fb` least significant bits by means of a division, a cast to an integer, and then back to a real number again.

- A properly scaled real value can be obtained by using `O(x,fb)`.

When calculating a scalar product, the sum of several multiplications can be truncated by single call to the `U` macro, in accordance with how accumulations are done in the real ALU.

6.3 Design

Two different approaches are available for introducing the concept of fixed-point arithmetics into the existing Simulink controller:

- Create a very general Simulink block that works like the ordinary LTI-system block available in the control systems toolbox, but having the resolution of the calculations as an additional parameter. It would then be fairly straightforward to replace parts of the current model with instances of this new block.
- Create a highly specialized block which encapsulates all parts of e.g. the radial controller as a single S-function, also with the resolution as a parameter, and then connect it directly to the inputs and outputs of the process through the A/D- and D/A-blocks of the model.

The first alternative certainly looks appealing as it potentially allows us to keep the current model as is, and only replace the various LTI systems in the controller with instances of this new block. This was actually the first approach for a new implementation, but it had some problems that led to the need of a specialized implementation of the whole radial controller instead. The main problem is that the arithmetic operations inside the S-functions take longer time to execute than ordinary floating point operations (for implementation reasons discussed in the previous subsection) and experiments showed that it would not be possible to meet the deadlines at the current sample rate by using such small blocks throughout the controller. In fact, the original controller was already pushing the limits by using almost the whole sample period for calculations, so there were not much margins to begin with.

By observing that Simulink always performes all calculations at all times, even if a certain code path is inactive, it was obvious that the S-functions needed to cover larger parts of the system so that the logic involved in e.g. trackjumping could be taken into account during execution, to not do more work than necessary at each sample. The trackjumping algorithm is a good example, since that involves a switch to another controller during track jumps. In Simulink, calculations involving both controllers are made at all times, but of course only the output from one controller is used at any one time. This can be explicitly coded in an S-function so that unnecessary expensive operations can be avoided.

Another advantage of reimplementing the whole radial-controller in C is that the original Simulink model contains several bulky workarounds for dealing with various sequencial actions (mostly involving the trackjumping algorithm) which would be much easier to implement using a standard procedural language.

Since the experiments presented in this theses only cover the radial-servo, the other blocks of the total system is kept in their original versions to not complicate the implementation. The structure of the overall controller can be seen in Figure 11. The S-function for the radial-controller can be seen in the middle of the diagram.

The reimplemented radial-controller has the following features:

- All system matrices as well as the sample time and the ratio between the

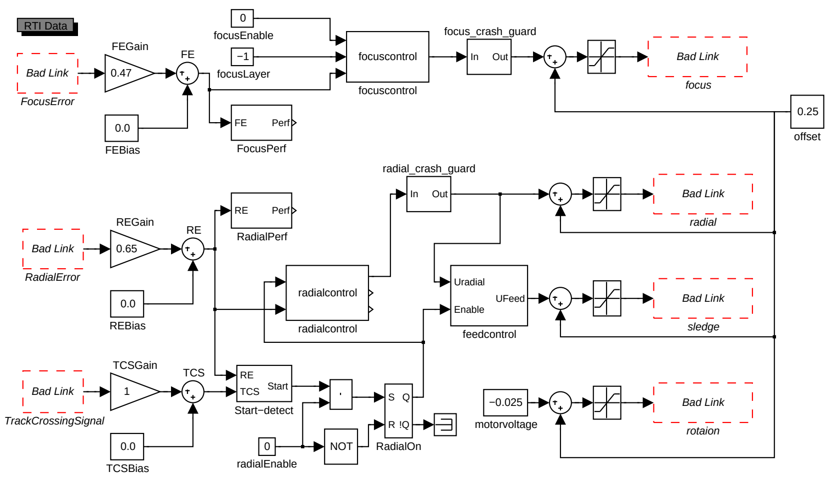

Figure 11: Simulink model of the control system.

slow and the fast observer are input parameters to the S-function.

- Track jumps are controlled through global variables.
- The resolution to use can be changed at run-time through a global variable. All system matrices are truncated to all possible resolutions during startup, so online changes of resolution takes almost no extra time.
- The maximum value of the states can be monitored through a global variable.
- When in multirate mode, the controller can be switched between accumulating and not accumulating measurements for updating the slow observer. This is also controlled through a global variable.

7 Results

7.1 Naive approach

By looking at the performance of *RE* while varying the numerical accuracy during online control, the data presented in Figure 12 has been aquired. In the top subplot, the performance measurement discussed in Section 5.5 is plotted as a function of time. In the middle subplot is the number of fractional bits used. This is the parameter that is changed manually during the simulation to see how lower numerical accuracy affects the performance. To get an idea of the total wordlength actually required, the maximum number of integer bits used up to a given time is plotted in the last subplot. Note that these parameters (i.e. the number of fractional and integer bits) are global to the whole radial servo, and are thus used for *all* coefficients and states in that part of the system.

As can be seen, the performance seems to stay rather steady as the number of fractional bits drop to about 12 or 13 (see the arrow in the top subplot). At 11

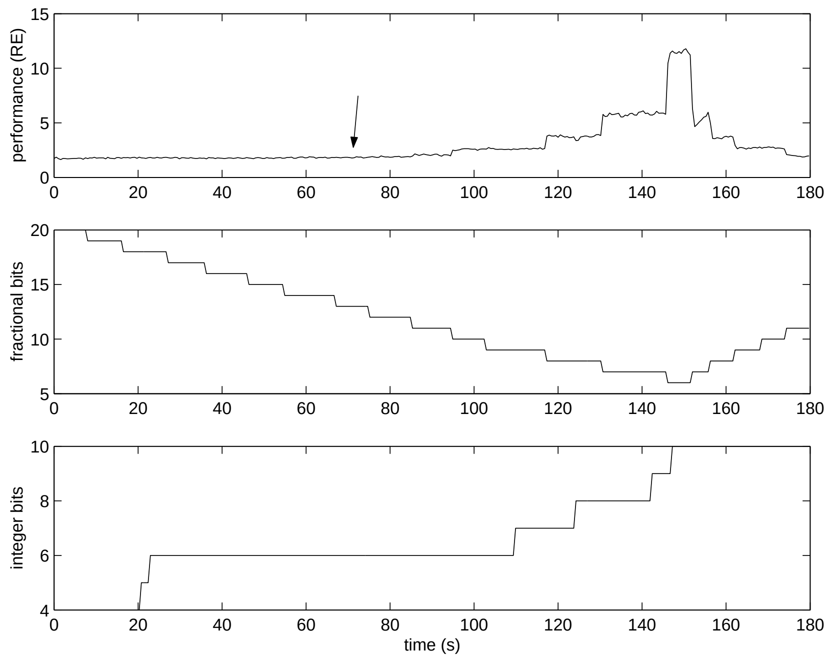

Figure 12: Performance of original radial servo with varying numerical accuracy.

bits and lower, noticable degradation of the performance can be seen, however the radial servo stays on track even when going as low as 6 fractional bits. At this point however, the number of integer bits required to hold the state values are dramatically increased as can be seen in the last subplot.

To sum up, it seems as if a wordlength of about 20 bits would be enough for this particular experiment, and further numerical accuracy does not seem to add any extra gain in performance. However, this experiment only covers about three minutes of track-following of a single disc, and it is likely that more bits is needed to get acceptable performance when e.g. doing track-jumps or trying to catch a track. Discs with other properties, i.e. more noise, large scratches etc., may also need more accuracy.

7.2 Multirate observer

The experiment performed for the original controller has been repeated for when the slow observer is downsampled different multiples of the original sample rate. Figures 13-17 shows the behaviour for some different cases. The plots are structured in the same way as described in the previous subsection. The arrows indicate where the author thinks is the lowest resolution before quantization effects can be seen in the performance.

It is a bit difficult to detect exactly where a decrease in numerical accuracy correlates to a degradation in performance, but it seems to hover around 11-12 fractional bits for all cases actually. Also the number of integer bits required

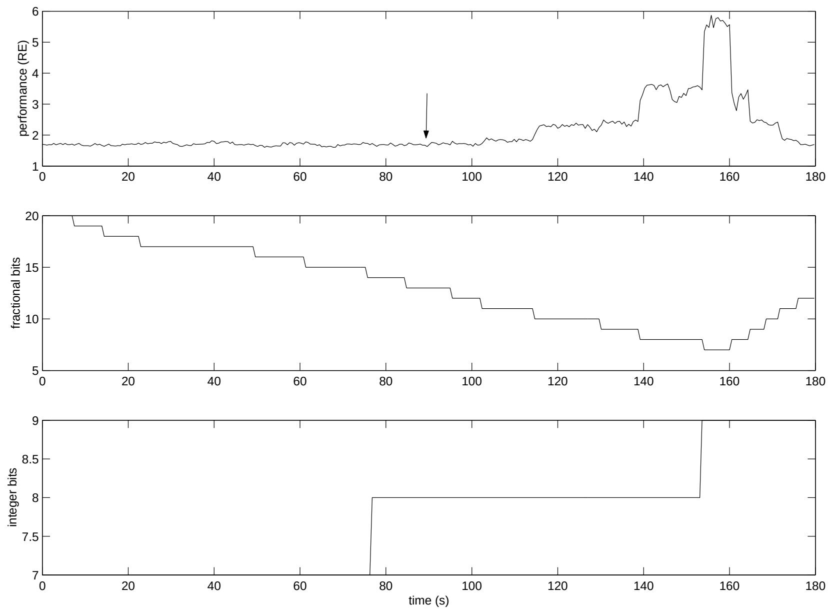

Figure 13: Performance of the radial servo when the slow observer is downsampled 10 times, with varying numerical accuracy.

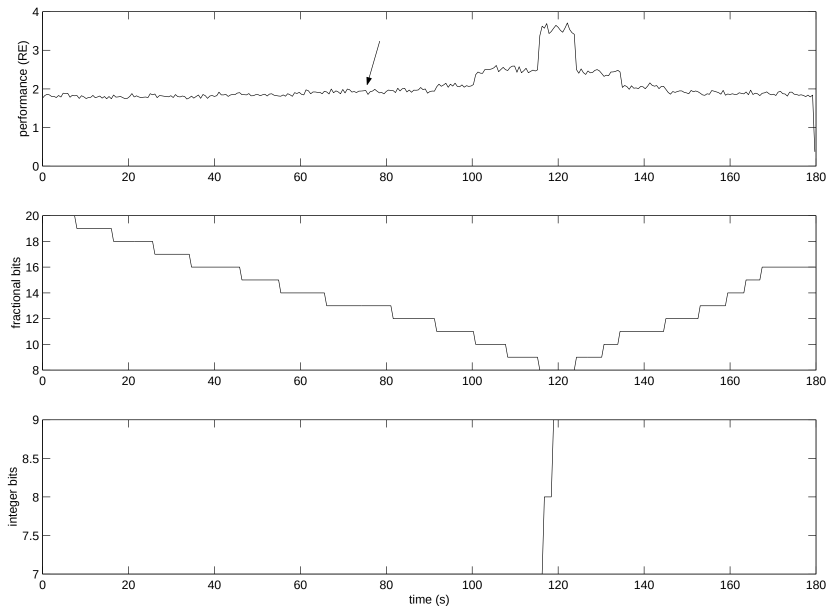

Figure 14: Performance of the radial servo when the slow observer is downsampled 20 times, with varying numerical accuracy.

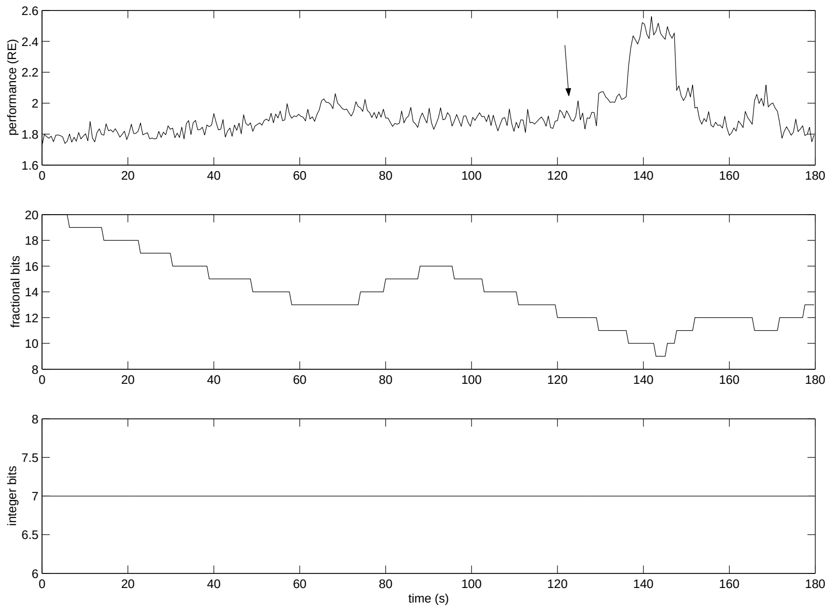

Figure 15: Performance of the radial servo when the slow observer is downsampled 30 times, with varying numerical accuracy.

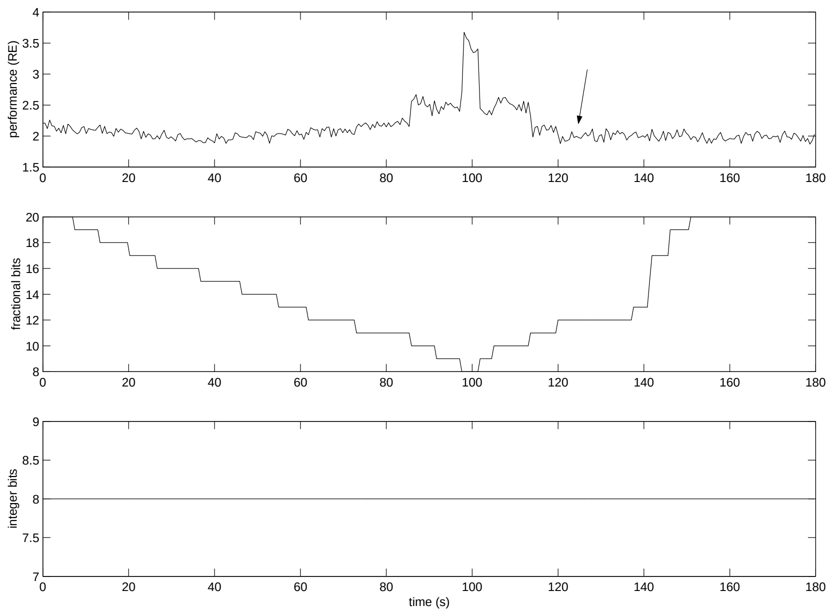

Figure 16: Performance of the radial servo when the slow observer is downsampled 40 times, with varying numerical accuracy.

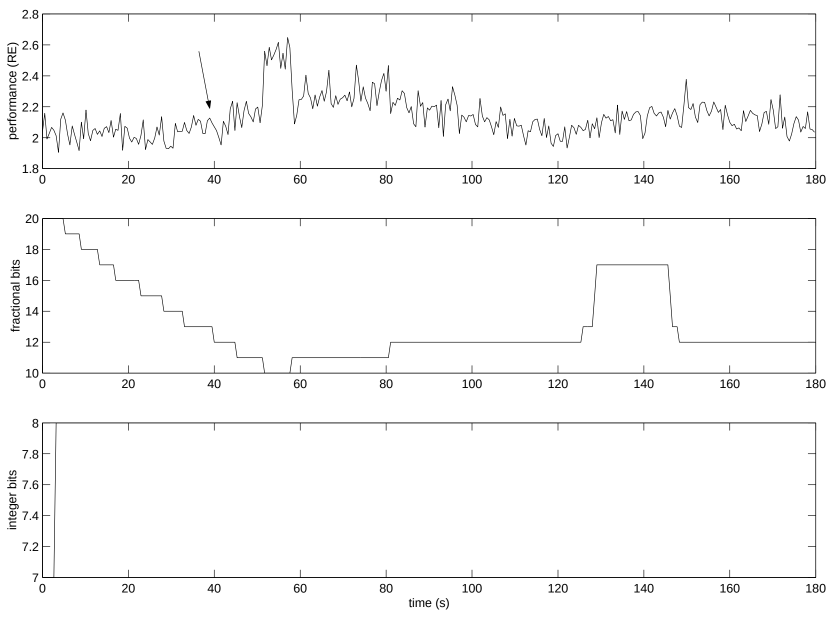

Figure 17: Performance of the radial servo when the slow observer is downsampled 50 times, with varying numerical accuracy.

seems to stay at around 7 or 8. Note that although these different experiments has been conducted on the same disc at approximately the same position, it is probably dangerous to assume that the physical conditions are the same in all cases, and thus the performance measurement should not be considered absolute between different plots. Even considering this, it is obvious that the performance has degraded as the slow observer gets further and further downsampled (most noticable in the last experiment), even if high numerical accuracy is retained. This is of course not acceptable for a candidate to a replacement controller, since it is basically a requirement that it does not have noticably poorer performance.

The degraded performance when trying to use larger downsamplings can probably be blamed at a too poor reconstruction of the oscillative disturbance states. To verify this, plots of these states during actual control has been included for some different cases. When doing these experiments, the numerical accuracy was locked at a relatively high level of 16 fractional bits, to isolate effects from the downsampling itself. Figures 18, 19 and 20 show how the states of the slow observer varies over time when following a track, for some different cases. The state in the top subplot of each figure corresponds to the position of the disturbance and the bottom subplot show its velocity.

As expected, these states are very poorly reconstructed when downsampling this much, and it certainly shows in the overall performance. This behaviour is a bit unfortunate, since it would actually seem quite reasonable to use a downsampling of about 100-200 times; this would mean the observer updated about 15-30 times during a full period of the oscillation. The reason why the reconstruction does not work better at these rates is that the downsampled observer only uses the *current* measurement for updating its states, discarding all

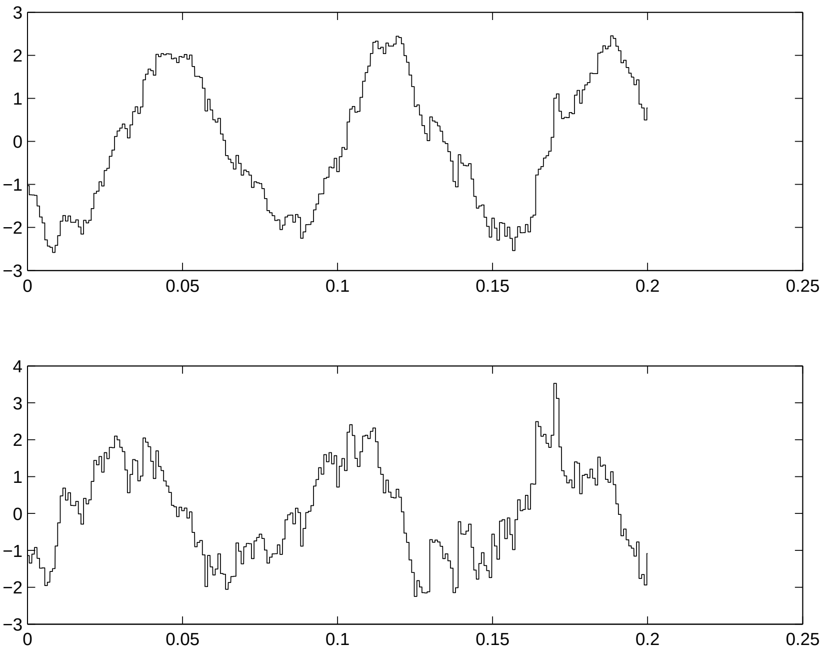

Figure 18: Reconstruction of the oscillative disturbance states, when the observer is downsampled 50 times. Uses 16 fractional bits.

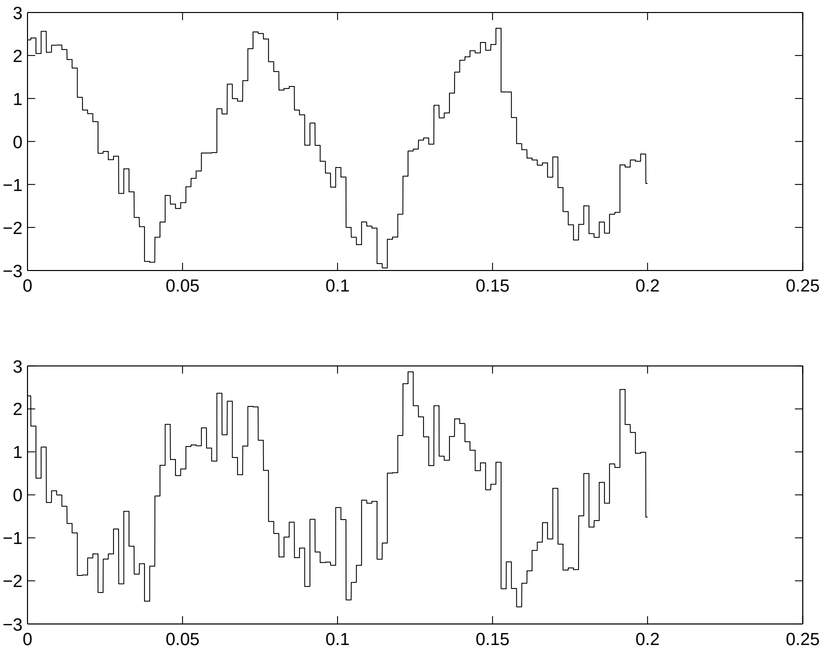

Figure 19: Reconstruction of the oscillative disturbance states, when the observer is downsampled 100 times. Uses 16 fractional bits.

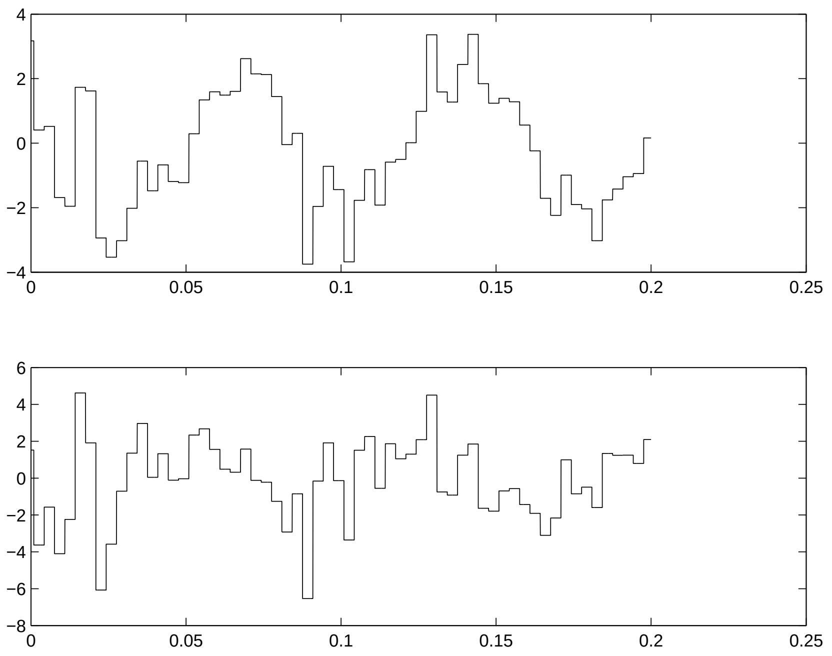
Figure 20: Reconstruction of the oscillative disturbance states, when the observer is downsampled 200 times. Uses 16 fractional bits.

measurements since the last slow sample. Since the measurements are sampled at a faster rate than the observer for the disturbance in these cases, aliasing will occur that distorts the information, hence the distortion of the reconstructed states. As can be seen in the next section, it is possible to achieve much better performance, even at large downsamplings, if all measurements are used instead.

7.3 Extended multirate observer

As explained in Section 5.4, the measurements between slow samples can be saved, and the slow observer can use the sum of these measurements to update its states through the original Kalman filter. This is effectively a low-pass filter on the measurements and the simplicity of this approach compared to other possible low-pass filters, is of course appealing (it does not impose any extra requirements on the need of numerical accuracy, for example). Experiments have been made under similar conditions as the ones in the previous section, displaying the reconstruction of the two oscillative disturbance states for some different downsamplings.

By direct comparsion of Figures 19 and 21, and Figures 20 and 22, it is obvious that using the latter approach yields significantly better reconstructions. Even when going as far as downsampling 300 times (Figure 23) the states look at least as good as when downsampling only 50 times using the original multirate approach (Figure 18).

To see how the quality of the reconstruction of the slow states depends on nu-

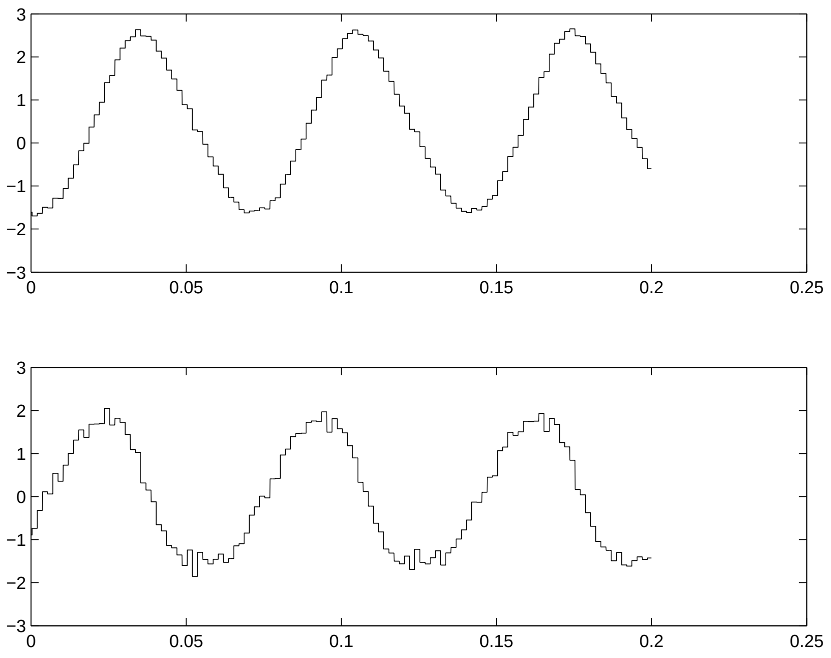

Figure 21: Reconstruction of the oscillative disturbance states, when the observer is downsampled 100 times and using accumulated measurements. Uses 16 fractional bits.

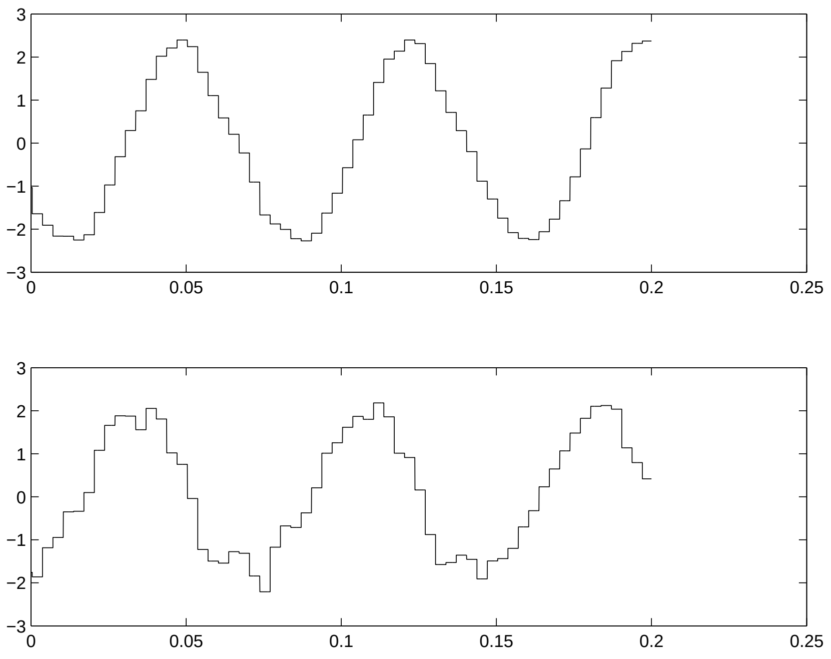

Figure 22: Reconstruction of the oscillative disturbance states, when the observer is downsampled 200 times and using accumulated measurements. Uses 16 fractional bits.

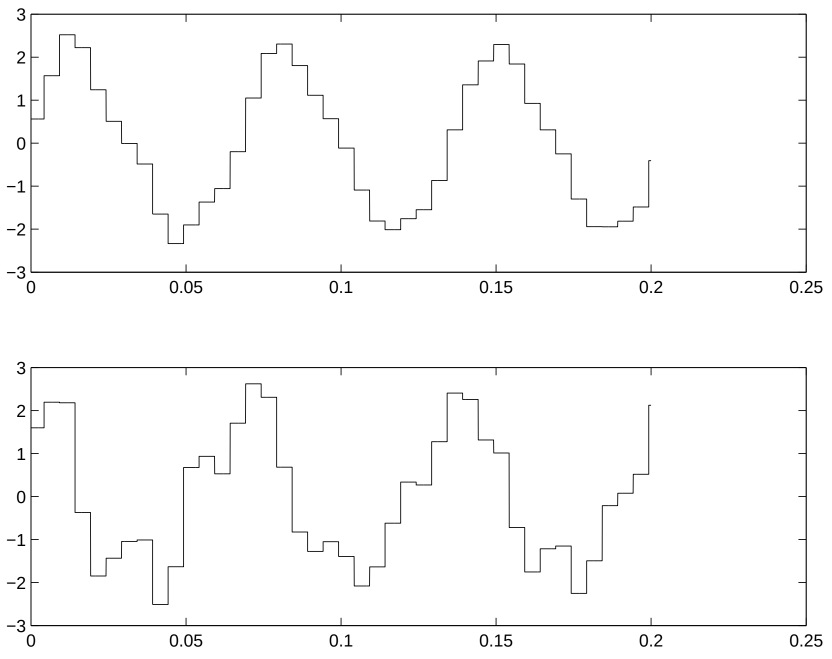

Figure 23: Reconstruction of the oscillative disturbance states, when the observer is downsampled 300 times and using accumulated measurements. Uses 16 fractional bits.

merical accuracy when using the extended multirate observer, some experiments were made with different parameters.

The reconstructions get worse as resolution decreases, as can be seen in Figures 24-27 when comparing 10- and 12-bit resolutions. However, these plots are not enough to draw any further conclusions on what is enough to get reasonable performance from the controller, so new performance tests are made with the extended multirate observer.

Figures 28 and 29 show the performance for two different cases while varying the resolution during three minutes of track-following. The performance starts dropping at 12 fractional bits and lower. As can be seen in the plots for the number of integer bits used, the large downsampling that have been made possible with the extended multirate observer has paid off. The controller seems to manage with only one integer bit, which results in a total wordlength of as low as 13. It is not easy to deduce from these experiments if the controller where the slow observer is downsampled 200 times performs worse than when it is only downsampled 100 times.

7.4 Track-jumping

Another thing that has been used to check if the reconstruction of the oscillative disturbance states is good enough, is to do track jumps. During track jumps, the states of the observer (both the slow and the fast) are particularly important

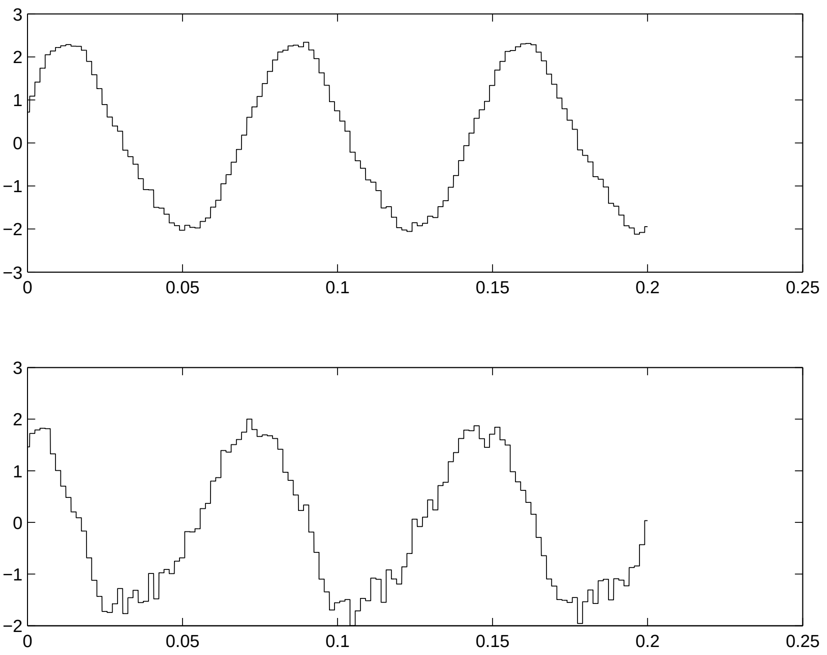

Figure 24: Reconstruction of the oscillative disturbance states, when the observer is downsampled 100 times and using accumulated measurements. Uses 12 fractional bits.

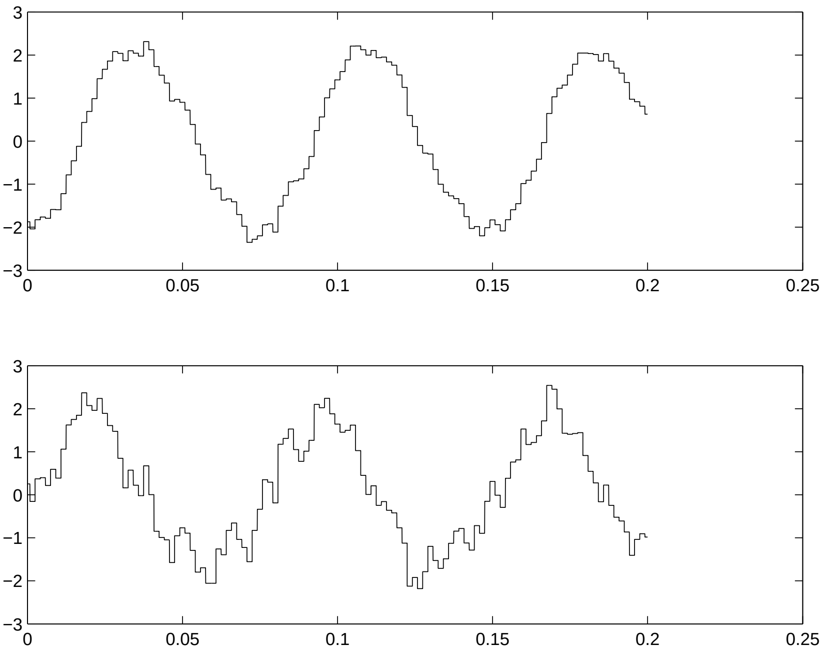

Figure 25: Reconstruction of the oscillative disturbance states, when the observer is downsampled 100 times and using accumulated measurements. Uses 10 fractional bits.

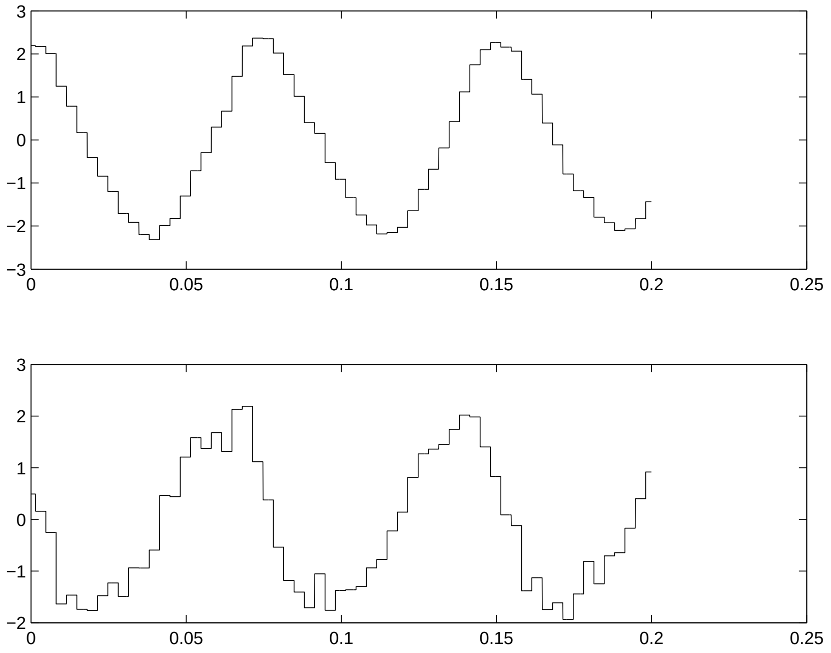

Figure 26: Reconstruction of the oscillative disturbance states, when the observer is downsampled 200 times and using accumulated measurements. Uses 12 fractional bits.

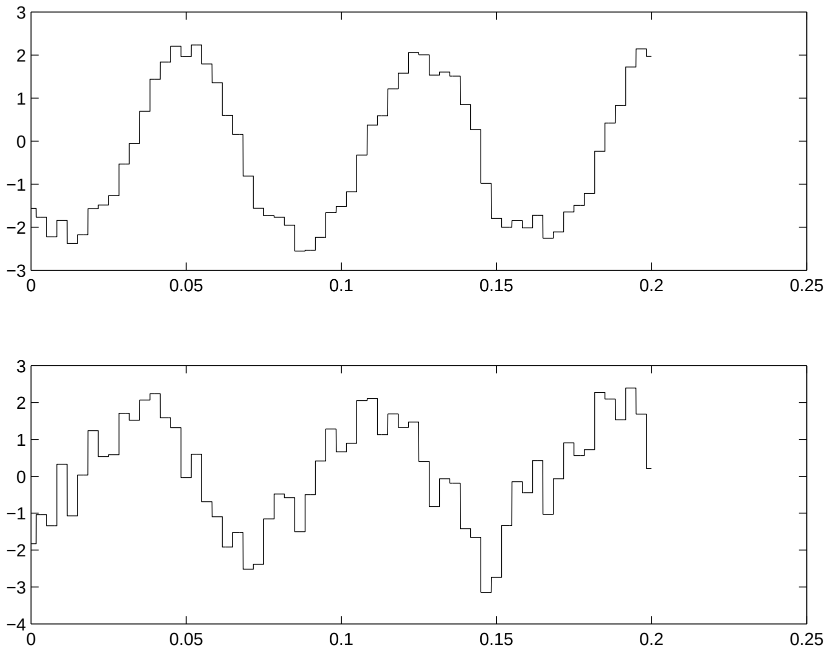

Figure 27: Reconstruction of the oscillative disturbance states, when the observer is downsampled 200 times and using accumulated measurements. Uses 10 fractional bits.

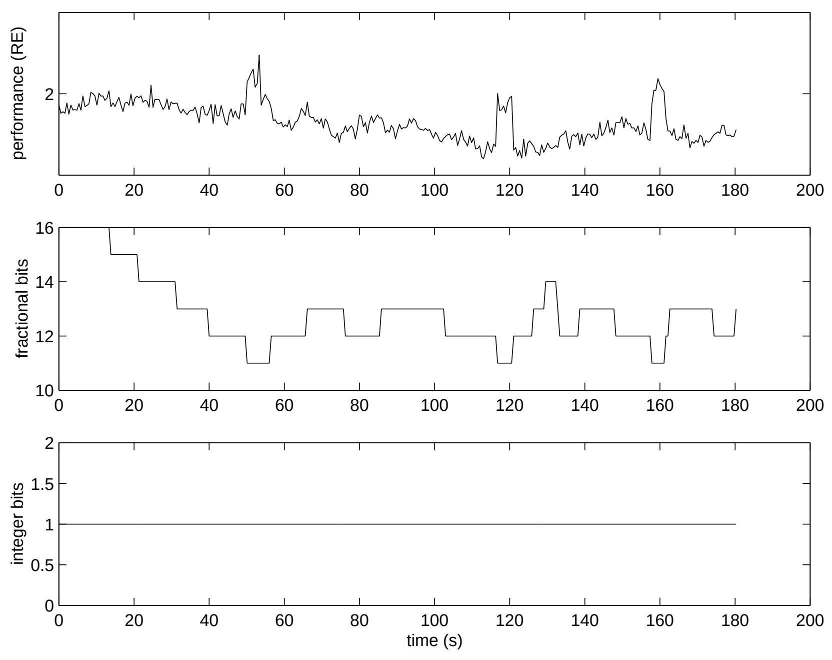

Figure 28: Performance of the radial servo when the slow observer is downsampled 100 times and using accumulated measurements, with varying numerical accuracy.

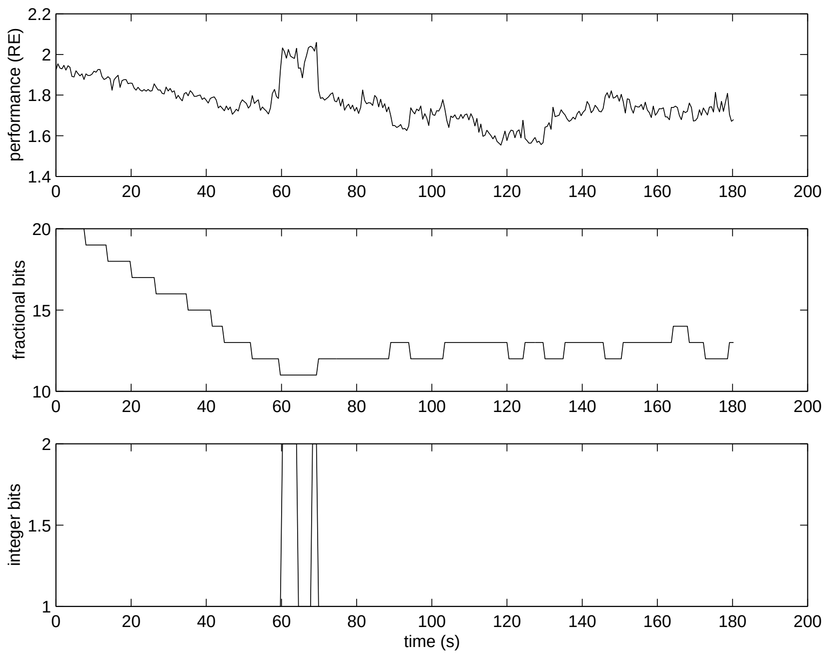

Figure 29: Performance of the radial servo when the slow observer is downsampled 200 times and using accumulated measurements, with varying numerical accuracy.

since they contain the only information available to the controller when it runs in open loop between the tracks.

The plots in Figure 30 show the typical appearance of a reconstructed radial error $y = C_e \hat{x}_e$ and the real radial error in the same diagram during a jump over 20 tracks. These signals are of course supposed to be very similar, but when entering the non-linear area between two tracks, the reconstructed signal will be different from the real.

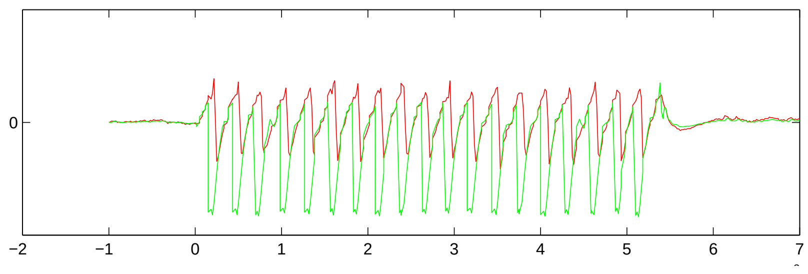

Figure 30: RE (red) and its reconstruction (green) during a trackjump, using 200 times downsampling for the slow observer and 16 fractional bits.

To see how many integer bits that are needed for the observer states when doing track-jumps, as opposed to just following a track when as low as one bit was enough in some cases, multiple repeated track-jumps have been made in both directions. These experiments showed that at least 4 integer bits were required to do the jumps, both when using a downsampling of 100 and 200 times for the slow observer. The number of fractional bits used for these experiments was in the range of 11-13.

These non-exhaustive tests seem to indicate that track-jumps work just as good as with the original controller when the slow observer is downsampled up to 200 times, and when the controller uses a total wordlength of as low as 16. Occasionally jumps fail though, both with the original controller and with the newer ones discussed here. This could very well depend on missing fine-tuning of the track-jumping algorithm. It has not been shown that the success-rate for jumps is lower (or higher) than previously for any controller discussed here.

8 Conclusions

The radial-servo has been reimplemented to facilitate experiments where parts of the controller are running at different rates, using fixed point calculations to simulate the conditions of a custom ALU. Experiments have shown that the observer for the slow dynamic process can be run at a much lower rate, and using a smaller wordlength than before, with retained overall performance of the controller.

A good choice seems to be to downsample the observer for the slow oscillative disturbance about 100 times, and use accumulation of the measurements for updating the estimates. This makes the slow observer run about 30 times faster than the frequency of the disturbance, which is much more appropriate than before. Using this design, the controller has been shown to work indistinguishable from the original controller at wordlengths going as low as 16 bits. However, the tests are not at all very exhaustive, and the experiments that have been made only show the behaviour for some very specific cases, and does not guarantee anything for the general case. Some margins must most likely be present in a real product, so aiming at a wordlength lower than 20 bits for the ALU might be too unrealistic.

9 Suggestions for further work

The ideas used to optimize the radial-servo are expected to be easily appliable on the focus-servo as well, since the processes have many similarities. Also every other calculation currently performed in the control system of the drive must be verified to work with a lower wordlength before the hardware can be changed. This should not be too troublesome, since the major bottleneck has been removed with the multirate controller.

Also worth noting is that the controller, as implemented here, expects that all signals and states have about the same dynamic range in the controller to fully utilize the selected wordlength. This design decision was made to not introduce a lot of degrees of freedom (i.e. not having to tune the dynamic range on a per-variable basis), and instead rely on that the choice of gains are made such that states and signals will end up in approximately the same range. The current design do indeed put signals and states in roughly the same area (i.e. there will not be any states that are several orders of magnitude larger than others) but the gains are by no means fine-tuned to make optimal use of the selected wordlength, so it is possible that the required minimum wordlength can be trimmed further by looking into this matter.

References

[1] Åström, Karl J. and Wittenmark, Björn, *Computer Controlled Systems, Theory And Design*, Prentice Hall, 1997.

[2] Parker, Dana J. and Starrett, Robert A., *CD-ROM Professional's CD-Recordable Handbook*, Pemberton Press, 1996.

[3] Sergio Bittanti, Fabio Dell'Orto, Andrea Di Carlo and Sergio M. Savaresi, *Notch filtering and multirate control for radial tracking in high-speed DVD-players*, IEEE Transactions on Consumer Electronics, Vol. 48, No. 1, February 2002.

[4] The MathWorks, *Simulink Model-Based and System-Based Design, Writing S-Functions*

[5] dSPACE GmbH, *Real-Time Interface, Implementation Guide*, March 1999.

[6] Proakis John G. and Manolakis Dimitris G., *Digital Signal Processing - Principles, Algorithms, and Applications*, Prentice Hall International, 1996.