The Banshee Multivariable Workstation: A Tool for Disk Drive Servo Research [*][†]

Daniel Y. Abramovitch
Hewlett-Packard Laboratories
1501 Page Mill Rd., MS: 2U
Palo Alto, CA 94304

Abstract

The servo design problems for both magnetic and optical drives have become more complicated as performance requirements for these drives get pushed higher. In order to easily try sophisticated control algorithms on these problems, a tool that makes the path between design and implementation an easy one would be useful. The Banshee Multivariable Workstation (BMW) is such a tool. The Banshee Multivariable Workstation is a Vectra/DSP based tool for doing Multi-Input, Multi-Output (MIMO) servo work. The hardware consists of a floating point DSP board (TMS320C30 based Banshee) residing in a 386/486 based MS-DOS machine (an HP Vectra). These are both interfaced to an HP3563A Control Systems Analyzer, an instrument designed to make measurements of both analog and digital control loops. The key feature of this package is that the DSP board is cleanly coupled to Matlab, a CAD program for servo/signal processing work that runs on a wide variety of platforms. This structure allows instruments designed for Single-Input, Single-Output (SISO) systems, such as the HP3563A to be used for MIMO systems. Furthermore, this allows for "running servo experiments from Matlab" and makes it easy to try out sophisticated controller designs with a few keystrokes.

[*]Presented at the 1992 ASME Winter Annual Meeting
[†]To be published in *Advances in Information Storage Systems, ASME Press*

This capability has stirred interest in researchers from both academic and industrial backgrounds. Examples of how this has been applied to disk drive research at HP Labs will be presented.

1 Introduction

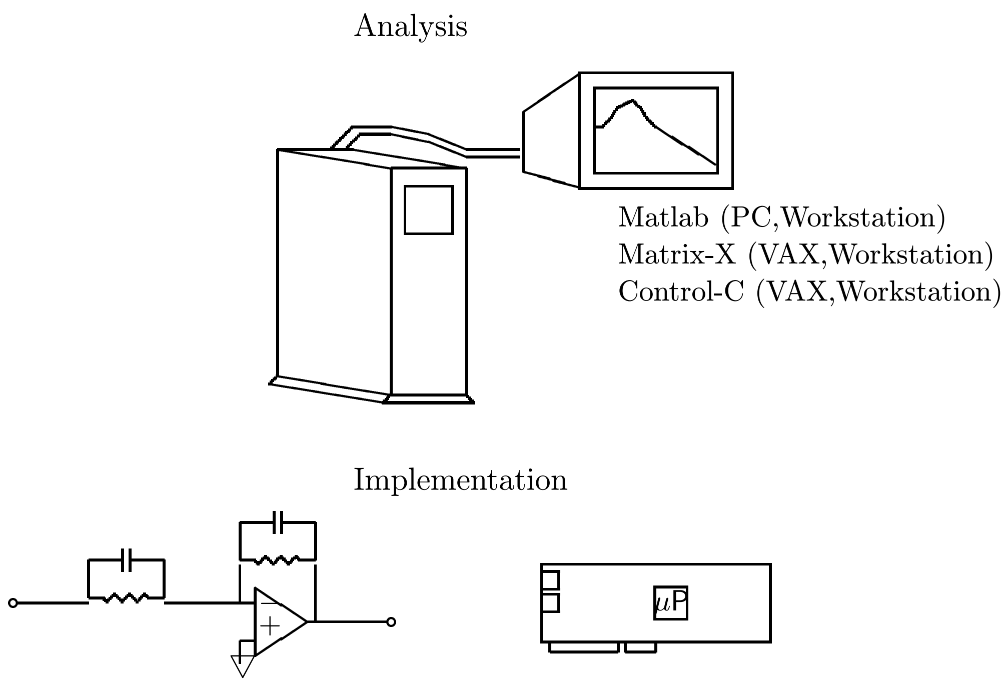


Figure 1: A conceptual view of control analysis and implementation tools. Note that there is little connection betwen all of these areas.

There is typically a very large gap between the academic and industrial control problems. In a textbook control problem one starts with a parametric model where some features of the problem *i.e.* some parameters may not be known. In addition there may be a nonlinearity of known character and certain noise properties are assumed. The objective from this point is to design a controller to give "good" performance along some metric. For the industrial control problem the starting point is a set of electrical and mechanical parts that are accompanied by some nominal parametric models. These are used to create a nominal controller. From this point the system can be measured and an improved controller can be generated. This process is iterated until either the system meets

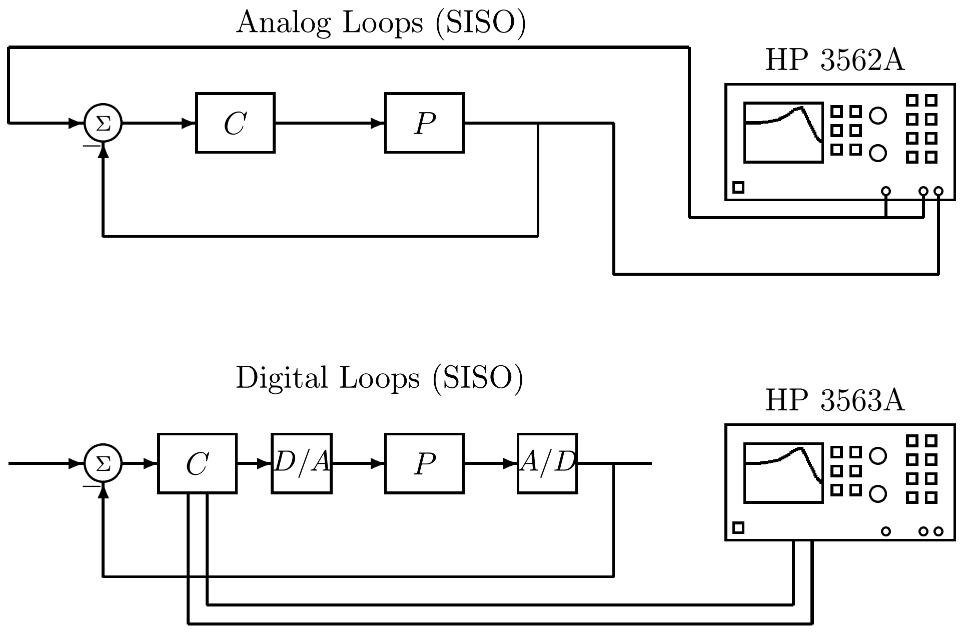


Figure 2: Dynamics Measurements: An example of how the HP 3562A can be used for making analog meausrements and how the HP 3563A can be used for making digital measurements.

its performance requirements or time and cost constraints dictate that the system has met its performance and cost requirements.

A key difference with the textbook problem is that the measurements are not necessarily (or even often) the parametric, time-domain measurements so prevalent in the on line identification literature [1, 2, 3]. More often, nonparametric frequency domain methods are used to obtain a frequency response function (FRF) from a given system input to a given system output [4, 5, 6]. Often, control design is done strictly in the frequency domain without reducing the measurement to a parametric model [7, 8, 9]. However, in order to use the sophisticated control algorithms and CAD programs now available, or in order to deal with multivariable[1] problems in a graceful way, a parametric model is essential. This can be obtained by curve fitting a transfer function to the frequency response function [10, 11, 12, 13, 14]. This process itself is imperfect and is one of the main difficulties in obtaining good parametric models of industrial control problems [15, 16].

[1]The term *multivariable* is synonymous with MIMO (Multi-Input, Multi-Output) in control and signal processing.

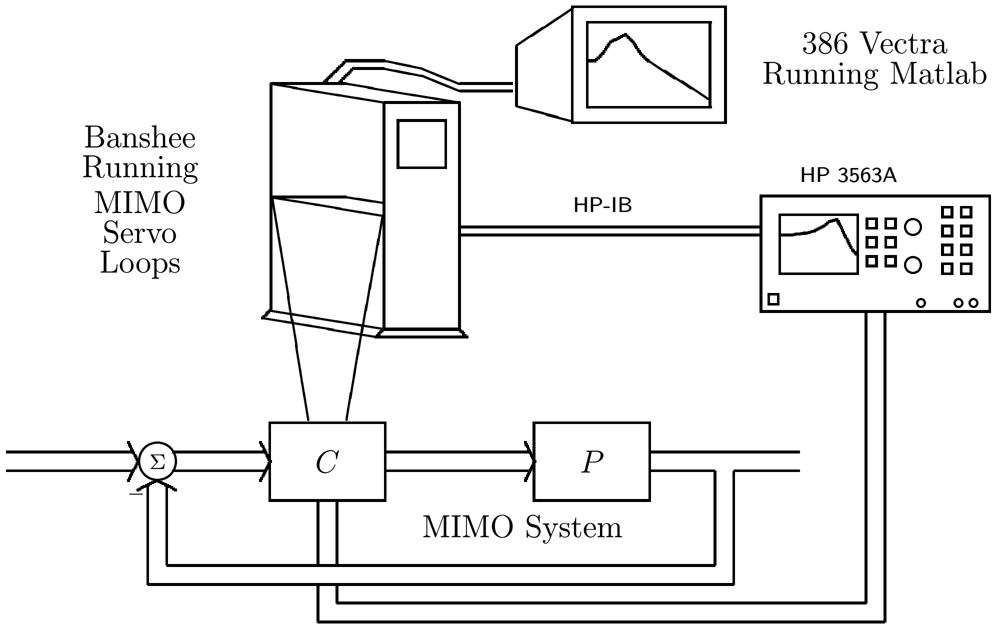


Figure 3: Banshee Multivariable Workstation

From the implementation point of view, it is difficult to try a variety of new control designs when each design involves recoding assembly language. This works against the high sampling rates required by high performance magnetic and optical drives which mandate tight assembly language loops on fast DSP chips. In contrast, many low performance magnetic drives are controlled by moderate performance microcontrollers. While the lower performance specifications of the drives allow for slower sampling rates, the lower performance of the microcontroller CPU typically means that there is even less CPU bandwidth available per sample. Furthermore, the performance of both DSP chips and microcontrollers improves considerably when they use on chip memory exclusively. This severely limits the size of the binary code that can be loaded. The speed and space requirements mandate that code be written in tight assembly language loops. From the perspective of experimenting with new controller designs or even new controller structures, this makes things quite difficult. Even when some analysis is done in Matlab, the designs tend to be kept simple so that they can be easily coded in DSP or microcontroller assembly language. This also limits entry of those researchers who are quite familiar with sophisticated algorithms, but who do not wish to spend time on assembly language programming. Thus, the gap that exists in the modeling problem between what is typically the academic and typically the industrial perspective is reenforced in the implementation area.

Another major issue is exactly how non-parametric measurements of dynamic systems are made. Instruments which perform dynamics measurements, called dynamics analyzers, yield frequency response functions [4, 5, 17]. These instruments are essentially spectrum analyzers which are designed to work well in the mechanical frequency ranges (0 Hz–200 kHz). Many systems, including disk drives, cannot be measured in open-loop. They must have some nominal control loop around them so measurements are made from this closed-loop system. The analyzers used in this paper have a source channel and two input channels. Typically, two types of measurements are made: "two wire" measurements of a closed-loop response and "three wire" measurements of an "open-loop" response. In the case of a two wire measurement, the analyzer source is injected at a junction point of the system and fed into the first analyzer channel. A second signal is measured from a chosen spot in the control loop and this is fed into the second analyzer channel. In a three wire measurement, the input to the first channel is not the analyzer source, but some different measured signal. The advantage of a three wire measurement is that open loop measurements can be made of a closed-loop system. The disadvantage is that any noise sources in the loop between the first analyzer channel and the second analyzer channel will bias the measured frequency response function. In the two wire case, the closed-loop measurements must be "opened" by performing loop mathematics on the measured frequency response function. This operation is subject to finite word length effects, since these instruments usually perform mathematics with fixed point DSP chips. This can be a problem in places where the closed-loop response is close to 1 (typically at low frequency).

The instruments that can make and unwrap closed-loop measurements are all single-input, single-output (SISO) devices. Examples of this type of device are the HP 3562A Dynamic Signal Analyzer (for analog measurements) or the HP 3563A Control Systems Analyzer (for digital, analog, or mixed measurements), shown in Figure 2. Since the HP 3563A has a superset of the other's functionality, it will be the device referred to in the sequel.[2]

Measurements with the HP 3563A Control Systems Analyzer (CSA) are done as follows. Analog control systems are measured via BNC connectors on the front panel of the instrument while digital measurements can be made by connecting a series of digital pods directly to registers or buses of the digital system under test. Quite a few signal processing features are included in the instrument and

[2]The HP 3563A is referred to as a Control Systems Analyzer rather than a dynamics analyzer because its digital capabilities are motivated by the desire to make measurements of digital control loops.

it can be connected to other computers via an HP-IB[3] port. Thus, the possibility of programming the instrument remotely exists and in fact examples in BASIC are given in the CSA documentation. Currently, the CSA can only handle Single-Input, Single-Output control systems.

A dramatic improvement in the time it takes to go from measurement to analysis to design to implementation and back to measurement can be achieved by fusing the best features of the above tools. The pieces of a solution currently exist. Those pieces are listed below:

- There are good standard CAD software packages for MIMO analysis and design *e.g.* Matlab, Matrix-X, Control-C.
- There are good floating point DSP chips on the market *e.g.* TI's TMS320C30, AT&T's DSP32C, Motorolla's 96000.
- There are good instruments for measuring frequency response functions *e.g.* HP 3563A.

There is one more realization that motivates a solution: for complicated control problems, a servo designer wants to spend all their mental energy working within the framework of the CAD program with which they do their system modeling and control design.

The solution presented here can be summed up in one phrase: *run the DSP board from inside of Matlab*. The advantages of this philosophy should become apparent throughout the paper, but among them are:

- Extremely rapid prototyping of designs. It is literally a few keystrokes from the point where Matlab gives you a particular servo design to the point where that design is running on the drive hardware.
- The system intelligence is where it should be, namely in the CAD software. All that is asked of the DSP is rapid implementation of general filtering schemes. This expands the ability of the workstation with each improvement of Matlab or new Matlab script that is created. As there is a continual stream of new scripts (called *m-files*) being produced both by the creators of Matlab (*The Mathworks*) and by third party developers (including quite a few academic researchers), this feature makes the workstation extensible.

[3]Also known as the IEEE 488 Standard Interface Bus. This is a commonly used bus for connecting instruments and computers.

- The processing capabilities of measurement devices, such as dynamics analyzers can be expanded by transferring measurements up to Matlab and processing them there. Also, finite word length effects are minimized since the operations in Matlab are done in floating point.

The tool, pictured in Figure 3 and called the *Banshee Multivariable Workstation*, makes use of a *Banshee* floating point DSP board interfaced to *Matlab*. The DSP board resides in 386/486 based MS-DOS machine (in this case an HP Vectra). This was chosen due to the fact that both Matlab and a wide variety of DSP boards were available for this platform. The floating point DSP eliminates many of the scaling problems faced in SISO control loops on fixed point DSPs. These problems would only be compounded in a MIMO control system. Matlab has analysis and design capabilities for MIMO control. In between lies a menu driven interface program that allows the DSP board to be run out of Matlab. This combination makes practical the implementation of sophisticated MIMO servo designs with relatively few keystrokes. Finally, this is coupled with an interface[4] to the CSA to allow for non-parametric identification of MIMO systems. This interface allows for direct digital measurements of feedback control loops implemented with the Banshee by the CSA. Furthermore the interface with Matlab allows the CSA to be used to make closed-loop measurements of Multi-Input, Multi-Output (MIMO) loops which can then be opened (using matrix waveform math). Finally, the interface allows the CSA to be essentially run from Matlab by means of a menu-driven interface program. The interface to Matlab greatly enhances the data collection and analysis capabilities of the CSA.

Physically, the Banshee is a DSP board made by Atlanta Signal Processors, Inc. (ASPI) based on a single TI TMS320C30 floating point DSP chip running at 16.67 MHz (a 60 nsec instruction cycle). It has been augmented by the addition of a custom analog IO board which has 4 A/D channels and 4 D/A channels[5], so it can implement the compensator, C, portion of the MIMO servo loop. It is connected directly to the digital pods of the HP 3563A for making digital measurements. It communicates to an interface program running on the Vectra via a data exchange register and dual ported RAM. The menu driven interface program communicates to the HP 3563A via HP-IB and can be run out of Matlab.

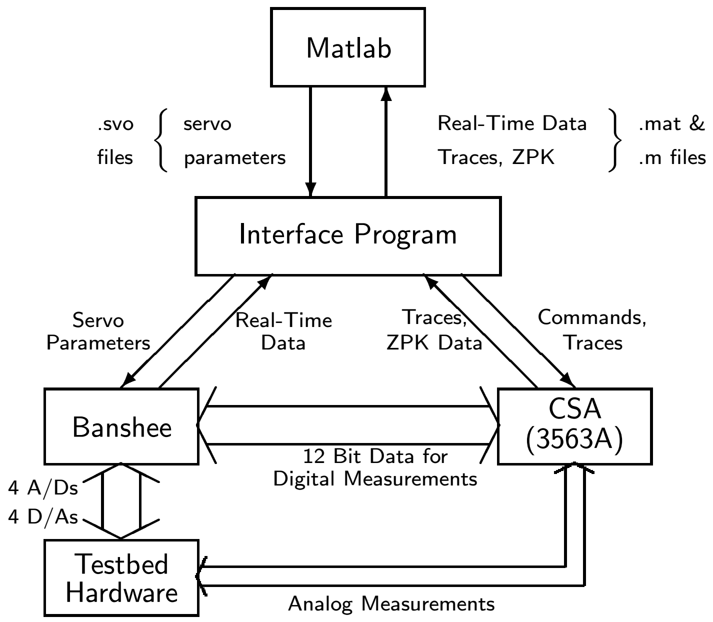
The remainder of this paper will be as follows: Section 2 will give a general description of how the Banshee Multivariable Workstation (BMW) works. Section 3 will discuss its use in servo research for magneto optic drives. Section 4 will discuss how we propose to use the BMW in magnetic drive

[4]Designed by Carl Taussig of HP Labs.

[5]Designed by Lennie Kiyama of HP Labs.

research. Section 5 will discuss possible future enhancements to this system and Section 6 will give a short summary.

2 How the BMW works



Floating point DSP allows easy implementation of servos designed in Matlab.

.svo file: an ASCII file that contains an entire MIMO servo. Written by Matlab, read by Interface Program.

Key feature of Interface Program: all data structures are dynamic $\Longrightarrow$ flexible.

Figure 4: The Banshee Multivariable Workstation: a functional block diagram.

A functional block diagram of the BMW is shown in Figure 4. The ASPI Banshee System (made by Atlanta Signal Processing, Inc.) is a digital signal processing (DSP) board that is based on the TI TMS320C30 floating point DSP chip. It fits into any IBM AT compatible bus. The host AT-class machine that is used here is an HP Vectra, but any AT class machine could be used. We will use the terms host and Vectra interchangeably from this point on. There are several methods to communicate between the host and the Banshee.

Matlab is one of a class of interactive, programmable software environments in which the fundamental data structure is a complex valued matrix [18]. (Other examples include *Matrix-X* and *Control-C*.) A key differentiator between Matlab and the other programs is that Matlab is compact enough to run effectively on DOS computers, while the others tend to be cumbersome on anything less than a 80386-class machine.

As mentioned earlier, the CSA is a dynamics analyzer for making measurements of both analog and digital control systems. Quite a few signal processing features are included in the instrument and it can be connected to other computers via an HP-IB port [5].

2.1 MIMO Servos at the Banshee

In order to preserve much of Matlab's flexibility, a decision was made to implement compensators in one of several general structures, *e.g.* polynomial form IIR or state space, where the size and parameters of the structures could be easily modified during operation by the Interface Program. These few general structure could then be coded efficiently in the DSP assembly language so as to minimize the overhead of the general structure's flexibility. This combination of flexibility and speed was made considerably easier to implement by the floating point nature of the TMS320C30, which eliminates most of the need for scaling.

Currently compensators at the Banshee are implemented as IIR digital filters. Unlike filters in fixed point DSPs, there are far fewer scaling problems in floating point. Thus, typical scaling techniques, such as factoring numerator and denominator polynomials into biquad sections, can be eliminated. This allows direct implementation of the transfer function form of compensators designed by Matlab. State space compensator designs may also be implemented by converting them to polynomial form filters before downloading them. The BMW has *m-file* scripts which create polynomial form IIR *.svo* files from Matlab state space design. (Direct implementation of MIMO state space compensators is possible but has not been coded yet.)

2.2 The Interface Program

The Banshee runs asynchronously from the Host and only needs to interface with it when there is a message passed down from the Interface Program. The Host runs two programs, Matlab and the Interface Program as shown in Figure 4. The Interface Program coordinates the interaction of the Banshee and the CSA, runs the CSA, directs the Banshee's real-time data collection, transfers complete MIMO servo files (*.svo* files) to the Banshee, and controls the interaction with Matlab. The next few sections will discuss these.

2.3 The Host/Matlab Interface

The interface between Matlab and the Interface Program can be done either by memory partition swapping (using PC-Matlab) or by issuing a shell escape from either AT or 386 Matlab. In both cases, all transfer of data between the two is through special file types: *.m*, *.mat*, and *.svo*. The *.m* and *.mat* files are standard file types for Matlab. The *.svo* format was especially created to allow Matlab to pass MIMO servos down to the Banshee through the Interface Program.

The Interface Program is designed to be able to pause its operation while Matlab runs and resume its operation with another shell escape from Matlab. To do this the Interface Program keeps a set of variables that contain the state of the Banshee. These are saved into a set of disk files, as well as any other dynamic data structures that are currently in memory. When the Interface Program resumes, it scans the directory for these files. If they exist, then their contents are restored into memory and the program continues.[6] If they are not present, then the Banshee is assumed to be reset and data variables are initialized. With this scheme, the Interface Program can be considered simply a routine run from Matlab. The data analysis and servo design is all done in Matlab, and a few key strokes lead to the implementation of a design at the Banshee.

2.4 Real-Time Data Collection

Real-time data collection is very flexible. The user has the ability to specify both the names of the variables to be monitored and the number of samples for which data collection will be done. Any memory location (and hence any variable) may be monitored by declaring a pointer to this datum and making it global. This second level of indirection allows monitoring of variables for which the physical location may change during the data collection. The Interface Program produces a table of all the variables available for monitoring which the user can choose from in a menu fashion. The only other information that the user need specify is whether a particular variable is a C30 floating point or a long integer. In the former case, the variable will have to be converted to IEEE floating point. The total size of a data run can not exceed the amount of free RAM which can be dynamically allocated[7]. Currently, this does not include any memory above 640K. This limits the number of records of a given size that can be taken. Doing this dynamically leaves the maximum

[6]This essentially uses the computer disk as a stack, where the Banshee state is "pushed" upon exit and "popped" upon re-entry.

[7]Known as the heap.

amount of memory free for other operations such as loading *.svo* files or interfacing with the HP 3563A. Finally, the data gathered on any run can be saved into a *.m* or *.mat* file for Matlab's use.

Data collection works as follows. A record is set up which tells the Banshee which variables to store at each time step. At the start of data collection, control of communication goes from the Host to the Banshee, which continues to run the control loop and stores the data record for that time step in the Banshee's Dual Ported RAM. This signals the Host, which unloads the data into a buffer in system (DOS) RAM and sends a handshake back to the Banshee. At a sample rate of 25 kHz, there has been no discernible performance effect of storing 10 variables per time step.

2.5 *.svo* MIMO Servo Files

The mechanism for having Matlab download MIMO servos to the Banshee is also quite flexible. The data transfer is done through a special file format, called a *.svo* file. The key to the *.svo* file is that it allows a dynamic data structure holding the entire servo to be allocated as it is being read. While this may seem a minor issue at first glance, it is quite important given that Matlab can design servos in either transfer function (IIR filter) form or state space form. Each of these forms may be implemented in either form on the DSP. Any useful file format must allow the Interface Program to create the needed data structure as it reads the file. Furthermore, the structure should be flexible as the number of plant inputs and outputs may vary for any given problem. Currently, both transfer function forms and state space forms have been specified for *.svo* files. This file structure leaves open the possibility for many other servo structures. The key is that all can be written easily by Matlab and read quickly by the Interface Program. The dynamic allocation is essential to the flexibility of this file structure since it allows controllers of different orders to be constructed.

2.6 Interface to Dynamics Analyzer

The BMW expands the capability of the CSA to Multi-Input, Multi-Output systems. This is accomplished by transferring data up to Matlab and performing matrix waveform mathematics there. Data can then be transferred back to the CSA to make use of the signal processing capabilities there. Of course, the signal processing algorithms in the CSA can be ported to Matlab thereby eliminating the need for transferring the data back.

The Interface Program sets up the CSA measurements and collects measurement data at the

Host. The Host in turn can save the data into *.m* or *.mat* files for use in Matlab. Furthermore, *.mat* files that contain the appropriate elements can be downloaded to the CSA so that the waveform processing routines on the CSA can be used.

Communication between the Host and the CSA are done over the HP-IB. The measurements can be analog or digital. Both measurement traces and zero-pole-gain (ZPK) tables (obtained from curve fits) can be uploaded to the host. The traces contain binary floating point data which must be converted to the IEEE format when they are uploaded. Likewise, when a *.mat* file version of a trace is downloaded, its floating point format must be converted back. Special data structures have been created that contain all the information in a trace or in a ZPK table.

In order to do digital measurements, the digital pods of the CSA are interfaced to the Banshee. Furthermore the Banshee is programmed to inject and extract signals into the appropriate servo loops. The Interface Program coordinates the Banshee with the CSA. When analog measurements are being done, the Banshee is not involved. The Interface Program disconnects the Banshee and signals are brought in and out through the front panel BNC connectors of the CSA.

When making digital measurements, the timing of signals becomes quite important. In digital measurement mode, the CSA expects a time delay of at least one sample between the signal it puts on the digital bus and the return signal that it receives. Thus, in the clocking scheme used here, the signal that the Banshee writes to the CSA is paired with the signal that was read from the CSA on the previous clock cycle. Furthermore, the Banshee can output a control signal to the D/As with either a full or fractional sample delay between it and the previous A/D read. While it is possible to measure a frequency response when the compensator has a fractional sample delay, this delay must be removed before doing a curve fit. The sample delay must be correctly known in order to get a reasonable curve fit [13]. In the fractional sample delay case, the delay may change with different compensators or from one loop to another, so the delay has to be measured for each input-output pair. A simpler solution, from the measurement perspective, is to use a full sample delay in the compensators which will be used in the measurement. Although, this would not be used in a final compensator, due to the loss in phase margin, it does make the loop unwrapping and curve fit much simpler for the MIMO case. This is what has been used in Figures 9 and 10.

3 Using the BMW for Magneto Optic Drive Servo Research

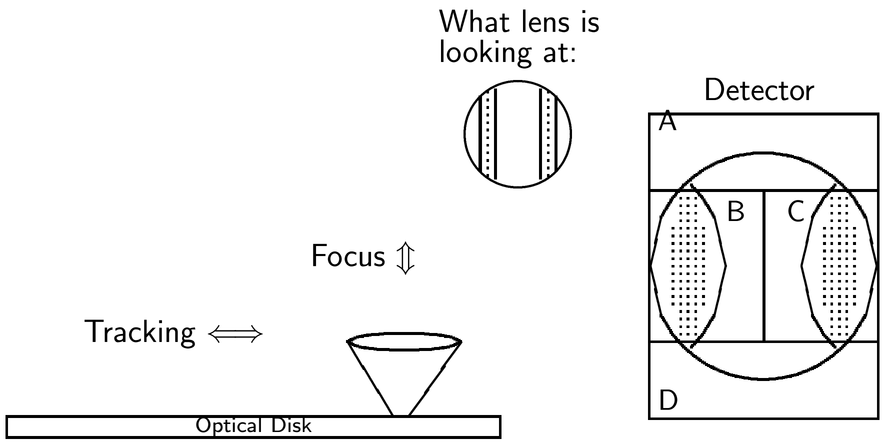


Figure 5: The magneto-optic drive servo problem

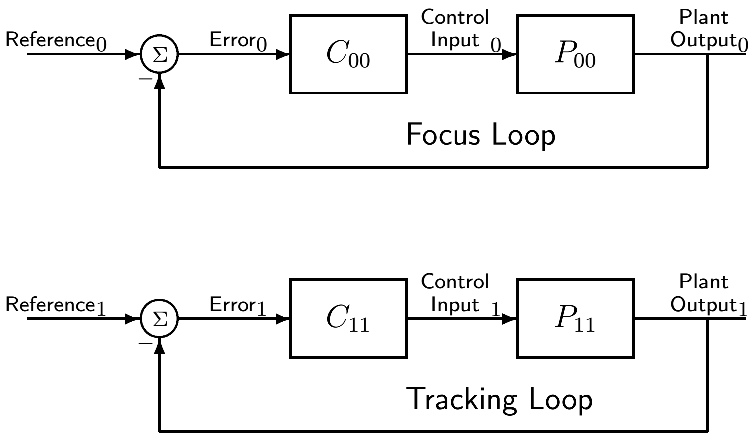


Figure 6: Focus and tracking as separate 2 Single-Input, Single-Output (SISO) systems

Figure 5 shows the optical drive servo problem. An optical drive operates in the far field of the lens, so it must both focus and track (as opposed to only tracking for magnetic drives – which are near field devices). The optical disk is grooved, which forms a diffraction pattern on a segmented detector. There are many possible servo detection schemes [19, 20], but most (on grooved media) depend upon detector alignment with the grooves. An example of a detection method is shown in Figure 5 and discussed below. By proper alignment of the detector with the grooves we can form

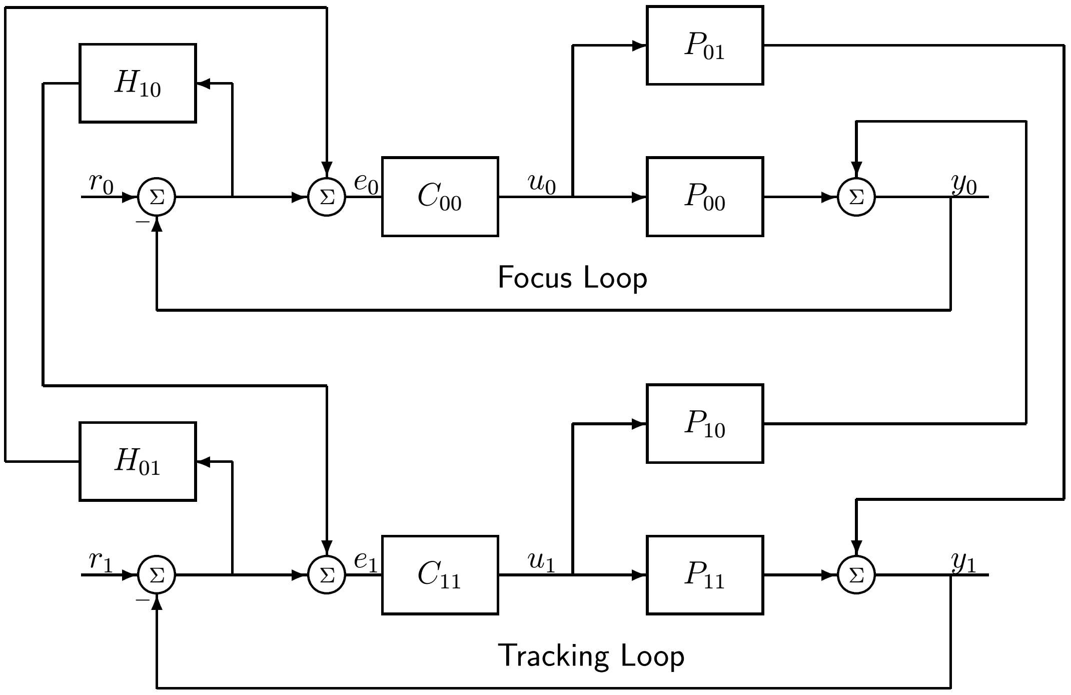


Figure 7: SISO control loops with cross coupling added.

a focus error signal (FES) and a tracking error signal (TES) that are orthogonal to each other.

$$\text{FES} = \frac{\text{A + D - k(B + C)}}{\text{A + B + C + D}} \tag{1}$$

$$\text{TES} = \frac{\text{A + B - (C + D)}}{\text{A + B + C + D}} \tag{2}$$

The normal way of looking at focus and tracking is that they are completely decoupled as shown in Figure 6. Here P_{00} is the focus portion of the actuator, C_{00} is the focus compensator, P_{11} is the tracking portion of the actuator, and C_{11} is the tracking compensator.

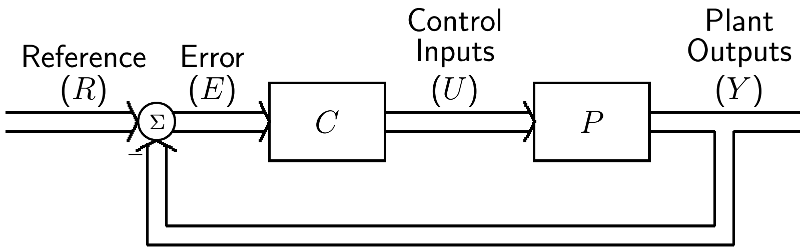


Figure 8: MIMO analysis is simplified by using vector signals and matrix transfer functions.

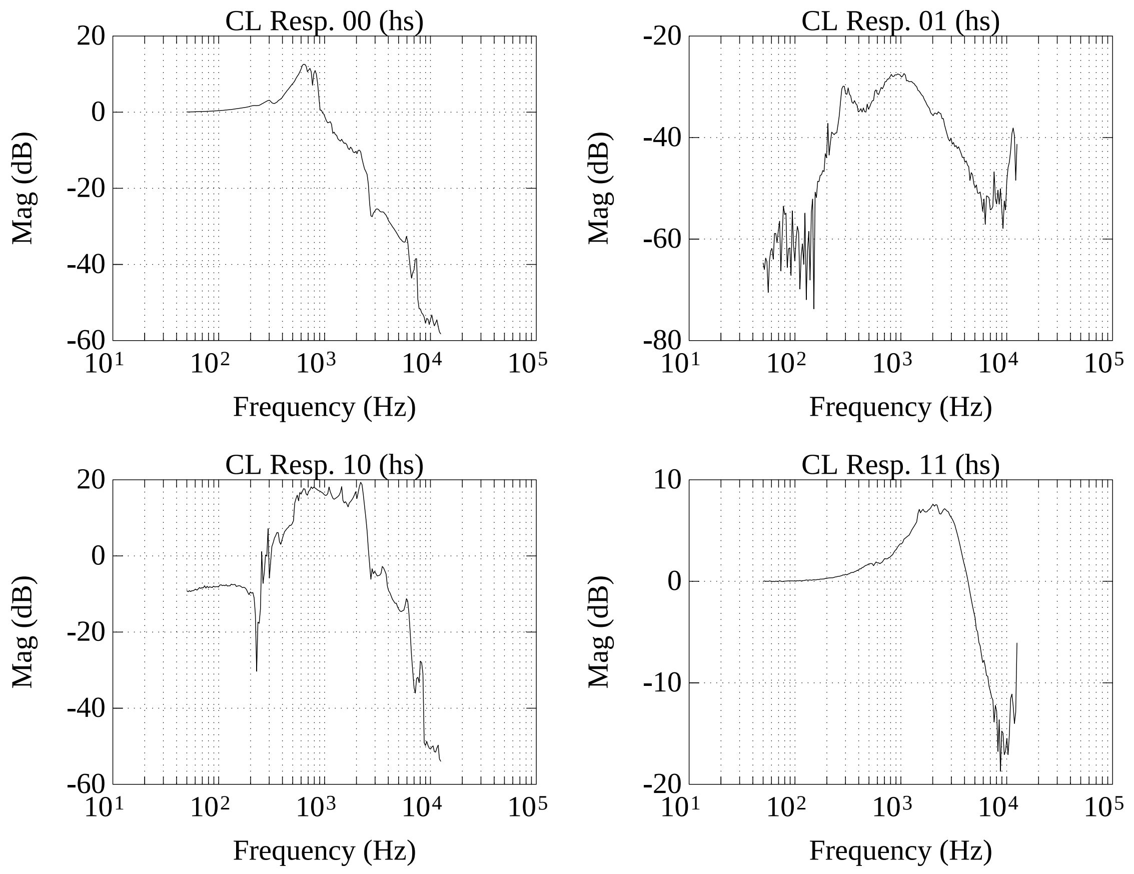


Figure 9: Closed-loop magnitude response for MO servo system.

Rotary actuation changes the skew angle, making it impossible to decouple the system by simple optical alignment. When the coupling is modeled as in Figure 7, the problem can seem intractable. Here the $P_{ij}, i \neq j$, can be thought of as mechanical coupling. The $H_{ij}, i \neq j$, can be thought of as sensor or optical coupling. A major conceptual simplification is seen in Figure 8. The key here is switching from thin lines to thick lines – also known as going from a scalar to a vector representation. C and P represent transfer function or frequency response function *matrices*. Note that the sensor module, H, has been lumped with P here.

As stated earlier many systems can only be measured while in a feedback control loop. Thus, closed-loop measurements are made using some nominal compensator. The measurement is then opened to get at the open-loop frequency response function, PC and finally to P. From here, a higher performance compensator can be designed. On an optical drive, these are the things we can measure (either continuous or discrete time):

$$\begin{aligned} T &= (I+PC)^{-1}PC = PC(I+PC)^{-1} && \text{Transfer function from } R \text{ to } Y \\ S &= (I+PC)^{-1} && \text{Transfer function from } R \text{ to } E \\ C & && \text{By direct measurement or computation} \end{aligned} \quad (3)$$

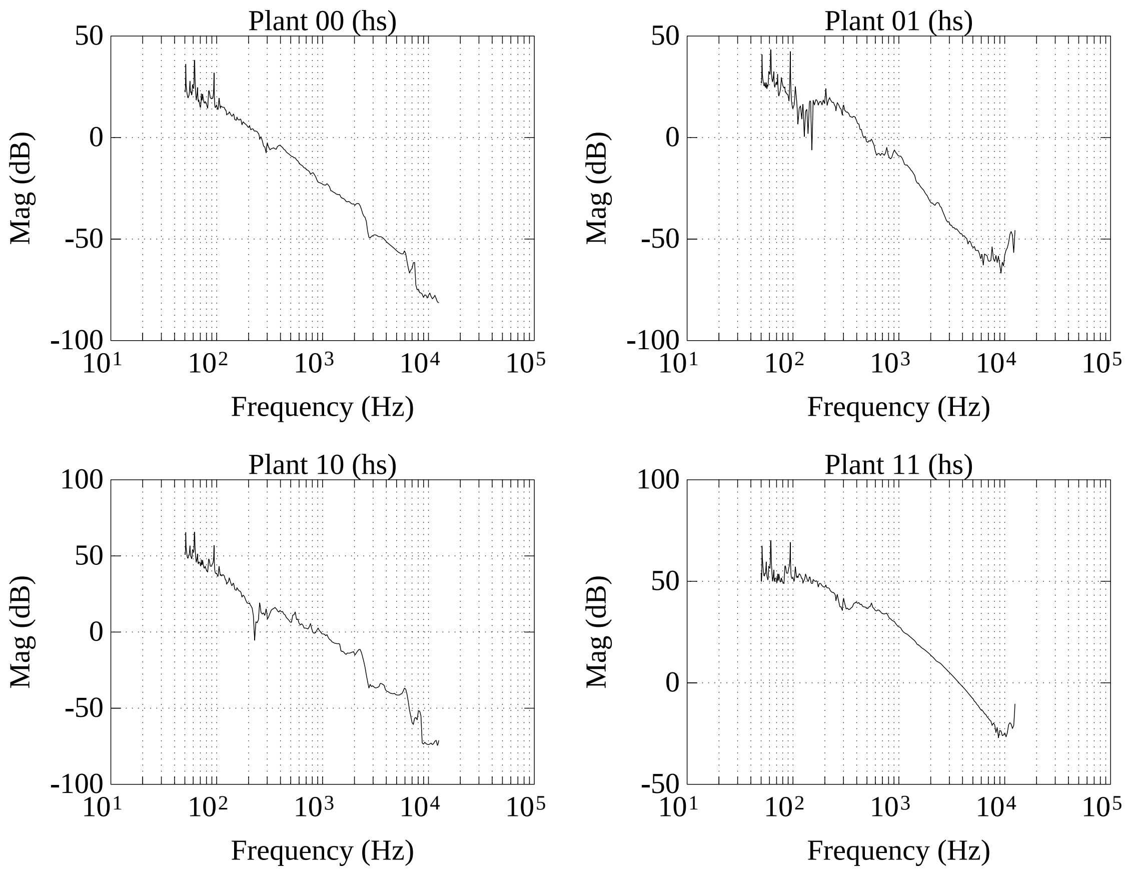


Figure 10: Plant magnitude response for MO servo system.

where T, S, C, and P are transfer function matrices and R, E, U, and Y are signal vectors. To get at P we unwrap these using *matrix* waveform math:

$$PC = S^{-1} - I \quad \text{or} \tag{4}$$

$$PC = T(I-T)^{-1} = (I-T)^{-1}T. \tag{5}$$

and factor out C. Note that although the compensator was synthesised in Matlab and therefore should be known, a measurement of C provides a check that the DSP is in fact implementing the design properly. If the frequency response function of C is not measured, then it must be synthesised with the same frequency axis as the measurements of T or S. This ensures that the matrix waveform math can be done correctly.

An example of this can be seen in Figures 9 and 10. Figure 9 shows the four closed-loop frequency response function measurement of an optical actuator. The compensator is implemented as a pair of third order IIR filters in a diagonal configuration ($C_{01} = 0$ and $C_{10} = 0$) which are sampling data at 25 kHz. As above, the 00 loop is from focus reference to focus output, the 01 loop is from track reference to focus output, the 10 loop is from focus reference to track output and the 11 loop is from track reference to track output. A key difference between SISO and MIMO measurements is apparent from these figures. Note that even though the open-loop plant frequency response function from tracking actuator input to focus actuator output (loop 01) is the same order of magnitude as the open-loop plant focus input to focus output frequency response function (loop 00), the closed-loop responses are considerably different. These measurements were made digitally by the CSA working with the BMW. Each input/output pair was done individually, the results were saved to *.mat* files, and these were uploaded to Matlab. Furthermore the compensators were measured open-loop and these measurements were also uploaded to Matlab. By applying Equation 5 in Matlab, the MIMO closed-loop frequency response function, T, can be unwrapped and the MIMO compensator frequency response function, C, can be factored out, yielding the MIMO plant frequency response function, P. The magnitude response of the MIMO plant is shown in Figure 10. Curve fitting can be done up in Matlab, or each of the individual frequency response functions can be downloaded into the CSA to use its curve fitter. Finally, the zero, poles, and gain (ZPK data) can be uploaded back into Matlab.

4 Using the BMW for Magnetic Servo Work

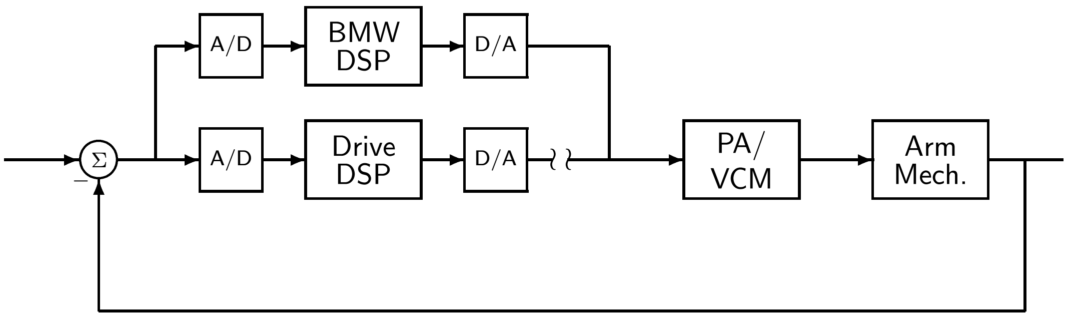


Figure 11: The Banshee DSP can be thought of as being "clipped" into an existing drive's servo loop as shown.

An example of how the BMW could be used in magnetic disk drive research is shown in Figure 11. Conceptually, this is accomplished by clipping the output lead of the drive's D/A converter it is possible for the Drive DSP to continue to do all the non servo loop specific things that it normally does (*e.g.* spindle motor control, handshaking with the microcontroller). In practice, there is a little more that has to be done to allow the drive DSP to hand off the actuator control to the

Banshee. Furthermore, functions such as seek algorithms, reading the track number, and position error signal (PES) decoding may have to be recoded in the Banshee. However, the large memory space and fast floating point calculations of the Banshee causes these functions to have less of an effect on the Banshee DSP than on a typical drive DSP. By recoding only those functions that are necessary, the Banshee DSP is left predominantly free to concentrate on servo algorithms. It should be fairly clear that the rapid prototyping, system modeling, and data collection capabilities of the BMW discussed earlier are just as applicable for this problem.

5 Future Improvements

There are a few improvements that could greatly upgrade the convenience of the system. First, a migration from running out of MS-DOS to MS-Windows would greatly improve the data handling capabilities. This depends upon a Windows version of Matlab being available (it is now) and porting the Interface Program to MS-Windows. Secondly, the Banshee code can be augmented to include state-space implementation. This should be fairly straightforward. Finally, porting more of the CSA functionality to Matlab *m-files* and increasing the flexibility of the Matlab code would help. These improvements could also be considered as the steps to turning the BMW into a product, rather than simply being a research tool. If based on an "open systems" approach, it could be extended to use next generation measurement instruments, other DSP architectures, other CAD programs, and even run on different computer architectures.

6 Summary

The BMW is built on some very simple yet powerful concepts, namely:

- run the DSP board from Matlab,
- leverage existing hardware and software, and
- blur distinctions between instrumentation, implementation, and analysis.

While they may seem like simple, intuitive concepts, they are not easy to maintain when trying to actually make something work. However, holding true to these ideas has allowed the BMW to do things far beyond the sum of its individual components. As this paper has attempted to show, it allows a researcher working on disk drive servos to try some novel designs with a minimum amount of pain.

7 Acknowledgements

The author would like to thank Carl Taussig and Lennie Kiyama, both from HP Labs, for their contributions to the BMW. Carl created the interface between the BMW and the HP 3563A. Lennie designed the new A/D board used on the BMW. Much of the current usefulness of the BMW is due to their efforts.

References

[1] L. Ljung, *System Identification: Theory for the User.* Prentice-Hall Information and System Sciences Series, Englewood Cliffs, New Jersey 07632: Prentice-Hall, 1987.

[2] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification.* MIT Press Series in Signal Processing, Optimization, and Control, Cambridge, Mass 02142: MIT Press, 1983.

[3] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control.* Information and Systems Science Series, Englewood Cliffs, N.J. 07632: Prentice-Hall, 1984.

[4] Hewlett-Packard, *Control System Development Using Dynamic Signal Analyzers: Application Note 243-2*, 1984.

[5] Hewlett-Packard, *HP 3563A Control Systems Analyzer*, 1990.

[6] J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures.* New York, NY: John Wiley & Sons, second ed., 1986.

[7] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems.* Menlo Park, California: Addison-Wesley, second ed., 1991.

[8] K. Ogata, *Modern Control Engineering.* Prentice-Hall Instrumentation and Controls Series, Englewood Cliffs, New Jersey: Prentice-Hall, 1970.

[9] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems.* Menlo Park, California: Addison-Wesley, second ed., 1990.

[10] E. Levy, "Complex-curve fitting," *IRE Transactions on Automatic Control*, vol. AC-4, pp. 37–43, 1959.

[11] J. L. Adcock, "Curve fitter for pole-zero analysis," *Hewlett-Packard Journal*, vol. 38, pp. 33–37, January 1987.

[12] Hewlett-Packard, *Curve Fitting in the HP 3562A*, product note hp 3562a-3 ed., 1989.

[13] Hewlett-Packard, *z-Domain Curve Fitting in the HP 3563A Analyzer*, hp 3563a-1 product note ed., 1989.

[14] R. L. Dailey and M. S. Lukich, "MIMO transfer function curve fitting using chebyshev polynomials." Presented at the SIAM 35th Anniversary Meeting, Denver, CO, October 1987.

[15] H. Vold and A. Meløe, "Pase errors in complex mode structural modification," *Sound and Vibration*, vol. 26, pp. 32–34, June 1992.

[16] M. D. Sidman, F. E. DeAngelis, and G. C. Verghese, "Parametric system identification on logarithmic frequency response data," *IEEE Transactions on Automatic Control*, vol. 36, pp. 1065–1070, September 1991.

[17] Brüel & Kjær, *Multichannel Analysis System Type 3550*, 1991.

[18] C. Moler, J. Little, S. Bangert, and S. Kleiman, *PC-MATLAB User's Guide.* The MathWorks, Inc., Sherborn, MA, Nov 1985. Version 2.0.

[19] G. Bouwhuis, J. Braat, A. Huijser, J. Pasman, G. van Rosmalen, and K. S. Immink, *Principles of Optical Disc Systems.* Bristol, England and Boston, MA: Adam Hilger, Ltd., 1985. ISBN 0-85274-785-3, out of print.

[20] A. B. Marchant, *Optical Recording: A Technical Overview.* Reading, MA: Addison-Wesley Longman, Inc., 1990. ISBN 0-201-76247-1.