Implementation of an Adaptive Repetitive Compensation Algorithm for Tracking-following Disk File Servos

Chi Chan, Graduate Student

Advisor: Professor Roberto Horowitz

Computer Mechanics Laboratory
Department of Mechanical Engineering
University of California at Berkeley
Berkeley, CA 94720

1 June 1997

Abstract

This report concludes a successful implementation of an adaptive repeatable disturbance compensation algorithm for tracking-following servos on a JTS 3.25 disk drive. The theoretical background, experimental setup, computational concerns, and method of implementation are discussed in detail in this report. Experimental data from the use of the adaptive algorithm for the repeatable disturbance compensation is presented.

Contents

1. Introduction

In business and everyday life, huge volumes of information -- audio and visual -- are being digitized, and such information that has traditionally been stored on analog media is now commonly stored on hard disk drives. Multimedia applications, the downloading of images and video from the Internet, and video-on-demand are just a few examples of the growing demand on storage capacity. In addition, the demand on reduction in thickness of portable disk files leads to using fewer disks inside the drives, but still maintaining the same capacity. As a result, the data areal density of fixed disk file systems increases, and so does the effect of disturbance on the actuator servo systems.

Two major functions are provided by the servos system in disk drives: track-seeking and track-following. The track-seeking servo attempts to move the head from one track to another in minimum time, whereas the track-following servo must keep the head positions over the center of a selected track as precisely as possible during the read/write process. Unfortunately, the data tracks are not perfectly concentric and the radial variations which take place as the disk rotates are referred to runout. Two types of track runout affect the track-following performance: non-repeatable and repeatable runout. The former is caused by such factors as random oscillations in HDA and external disturbances, whereas the latter is mainly due to the lack of eccentricity of the spindle-bearing-disk systems.

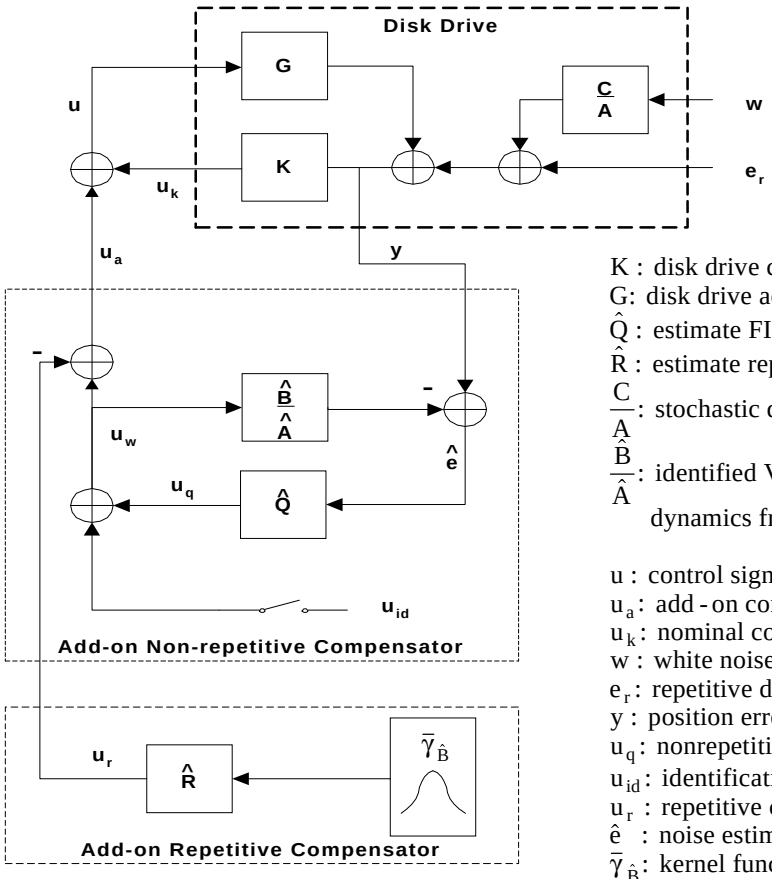
Since different units in a product line have different actuator dynamics and repeatable runout, a fixed compensator designed for all units may not be able to achieve the growing demand on storage capacity and track density. Besides, disk files, particularly those operate in portable computers, are subject to different vibration, noise and thermal environments, so more robust and adaptable servo control schemes becomes more popular. There have been several successful implementations of adaptive repetitive control systems to compensate for the repeatable runout in computer disk file servos. Most of these schemes require some a-priori knowledge of the actuator dynamics in order to guarantee stability. Implementations of adaptive non-repetitive control systems are less common. Recently, an adaptive system for disk file servos was introduced by (Horowitz, Sat & Li) which simultaneously identifies the actuator-controller dynamics and compensates for the repeatable runout. This is achieved by incorporating a learning repetitive control algorithm (Messner et al., 1993) in the control structure formulated in (Horowitz and Li, 1996). The purpose of this report is to present a successful implementation of such adaptive repetitive compensation scheme on a track-following servo of a disk file, which confirms the repeatable-runout rejection of the adaptive control scheme.

2. Theory

The formulation of the adaptive control structure on a disk drive used in this report was developed by (Horowitz and Li, 1996; Horowitz *et al*., 1996; Horowitz, Sat and Li, 1996). Readers with particular interests should refer to these paper for further details. For the sake of this report, fundamental ideas and formulae of the repetitive control structure are cited in this section.

2.1 Systems Formulation

The overall closed-loop dynamics is modeled as a SISO discrete time stochastic system. The system consists of odel of disk drive, a repetitive compensator, and a non-repetitive compensator. Figure 1 shows the overall block diagram of the disk drive track-following servo connected with the add-on repetitive and non-repetitive compensators. The block diagram contains three loops, the VCM-controller loop inside the disk drive, the non-repetitive compensator loop, and the compensator-to-drive loop. We may think the disk drive as the plant for which the add-on compensators try to control, whereas the internal controller K controls the VCM actuator itself.



K : disk drive controller
G: disk drive actuator (VCM)
$\hat{Q}$: estimate FIR filter
$\hat{R}$: estimate repetitive filter
$\frac{C}{A}$: stochastic disturbance model
$\frac{\hat{B}}{\hat{A}}$: identified VCM - controller closed - loop dynamics from u_a to y

u : control signal entering the VCM
u_a: add - on control signal
u_k: nominal control signal
w : white noise
e_r: repetitive disturbance
y : position error signal
u_q: nonrepetitive control signal
u_{id}: identification input signal
u_r : repetitive control signal
$\hat{e}$: noise estimate
$\bar{\gamma}_{\hat{R}}$: kernel function

Figure 1. Block Diagram of the Disk Drive and Add-on Compensators

As the figure shows, the add-on compensators manipulate the sampled PES *y(k)* and injects an auxiliary compensating signal $u_a(k)$. This compensating signal $u_a(k)$ is composed of three components: a white noise $u_{id}(k)$ which is used for an initial system identification phase, a nonrepetitive control signal $u_q(k)$, and a feedforward repetitive control signal $u_r(k)$. The sampled PES in discrete time domain can be described by the following static equation,

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})} u_a(k) + \frac{C(q^{-1})}{A(q^{-1})} w(k) + e_r(k) \quad (1)$$

where $u_a(k)$ is the auxiliary control input, $\frac{B(q^{-1})}{A(q^{-1})}$ is the discrete closed-loop transfer function from u_a to y using the unit delay operator q^{-1}, w(k) is a white noise, $\frac{C(q^{-1})}{A(q^{-1})}$ is the stochastic disturbance model, and $e_r(k)$ is a repeatable disturbance given by

$$e_r(k) = e_r(k-N) \quad (2)$$

The order of the polynomials $A(q^{-1})$, $B(q^{-1})$, and $C(q^{-1})$ are n_a, n_b, and n_c, respectively. The closed-loop transfer function from $u_a(k)$ to y(k) has the same denominator as the stochastic disturbance model because u_a and w both enter the same loop at different locations. In the system described above, the following assumptions are made: a) $q^{na}A(q^{-1})$ and $q^{nc}C(q^{-1})$ are Hurwitz, b) upper bounds for n_a, n_b, and n_c are known, c) the zero-delay term in $B(q^{-1})$ is assumed to be zero, and d) the period of $e_r(k)$ is known to be N.

The control structure in figure 1 consists of three control blocks, a nominal controller K, an add-on non-repetitive compensator, and an add-on repetitive compensator. Basically, the idea is to extract the repetitive and non-repeatable disturbance information from the PES and compensate for them. Both the add-on repetitive and non-repetitive compensators require an identification of the plant from u_a to y to generate the estimation errors ε & , respectively, which is then used to update the control signals. Consider the repetitive control algorithm. By expressing the polynomials A(q-1), B(q-1), and C(q-1) as $(1 + a_1q^{-1} + ... + a_{na}q^{-na})$, $(b_1q^{-1} + \ldots + b_{nb}q^{-nb})$, and $(1 + c_1q^{-1} + \ldots + c_{nc}q^{-nc})$, Eq. (1) is rewritten as follows,

$$\begin{aligned} y(k) &= \psi(k)^T\theta + A(q^{-1})e_r(k) + w(k) \\ \theta &= [a_1 \ldots a_{na},\ b_1 \ldots b_{nb},\ c_1 \ldots c_{nc}]^T \\ \psi(k) &= [-y(k-1) \ldots -y(k-n_a),\ u_a(k-1) \ldots u_a(k-n_b),\ w(k-1) \ldots w(k-n_c)]^T \\ u_a(k) &= u_{id}(k) + u_q(k) + u_r(k) = u_w(k) + u_r(k) \end{aligned} \quad (3)$$

where the auxiliary control signal $u_a(k)$ is separated into its stochastic $u_w(k)$ and repetitive $u_r(k)$ components. To obtain the estimation error $\varepsilon(k)$, an a-priori PES estimate $\hat{y}(k)$ is defined as follows,

$$\begin{aligned}
\hat{y}(k) &= \phi_w(k)^T \hat{\theta}(k-1) + j(k) \\
\hat{\theta}(k) &= [\hat{a}_1(k) \ldots \hat{a}_{\hat{n}a}(k),\ \hat{b}_1(k) \ldots \hat{b}_{\hat{n}b}(k),\ \hat{c}_1(k) \ldots \hat{c}_{\hat{n}c}(k)]^T \\
\phi_w(k) &= [-y(k-1) \ldots -y(k-\hat{n}_a),\ u_w(k-1) \ldots u_w(k-\hat{n}_b),\ \varepsilon(k-1) \ldots \varepsilon(k-\hat{n}_c)]^T \\
\varepsilon(k) &= y(k) - \hat{y}(k), \qquad u_w(k) = u_{id}(k) + u_q(k)
\end{aligned} \tag{4}$$

where $\hat{\theta}(k)$ is the estimate of the parameter vector θ. By updating the PES estimate with the stochastic control input u_w, the repetitive control input u_r can cancel the repeatable runout from the estimation error $\varepsilon(k)$ once converges, leaving only the stochastic disturbance $w(k)$. This allows the repetitive control input $u_r(k)$ to converge to the correct value at which the repeatable disturbance is cancelled because updating of $u_r(k)$ is based on the estimation error $\varepsilon(k)$. To show this, obtain the estimation error dynamics by subtracting Eq (4) from Eq. (3),

$$\begin{aligned}
\varepsilon(k) &= \frac{1}{C(q^{-1})}\phi(k)^T(\theta - \hat{\theta}(k)) + w(k) + \frac{1}{C(q^{-1})}[A(q^{-1})e_r(k) + \hat{B}(q^{-1})u_r(k) - j(k)] \\
\phi(k) &= [-y(k-1) \ldots -y(k-\hat{n}_a),\ u(k-1) \ldots u(k-\hat{n}_b),\ \varepsilon(k-1) \ldots \varepsilon(k-\hat{n}_c)]^T
\end{aligned} \tag{5}$$

The term $j(k)$ is shown to converge to zero in (Horowitz, Sat and Li, 1996) and the difference $\theta - \hat{\theta}(k)$ is assumed to be negligible. Therefore, the determining term in (5) are the repeatable disturbance and the filtered repetitive compensating input. If the repetitive control input u_r is updated such that $u_r(k)$ converges once ε is zero, $u_r(k)$ will converge to the values at which $A(q^{-1})e_r(k) = -\hat{B}(q^{-1})u_r(k)$. These converged $u_r(k)$, which repeat for every N'th samples, thus yields an effective compensation of the repeatable disturbance $A(q^{-1})e_r(k)$.

2.2 Repetitive Control Scheme

The term " repetitive control " refers to methods used to compensate for periodic disturbances which affect dynamical systems. Particularly, discrete periodic disturbances can be described as r(k) = r(k-N), where N is the period. The term " learning control " is also sometimes used. There are many mechanical systems which are subjects to periodic disturbances for which the period is known, most notably rotating machinery such as lathes, printing presses, and computer disk drives. Repeatable disturbances also occur in automated milling and machining and in the operation of robots as well as many other manufacturing applications.

All repetitive controllers require a model of the disturbance in order to generate a cancellation signal. Repetitive controllers can be classified as being either " Internal Model" based or " External Model " based controllers. In Internal Model controllers, a model of the disturbances signals is included in the basic feedback loop. The disturbance model is simply a periodic signal generator. The advantages of this type of algorithm are that it is linear, making analysis easier, and that convergence is very rapid. Complete cancellation can occur in a finite number of cycles. The disadvantage is that frequency response of the system alters because additional dynamics are added. For example, placing the disturbance model inside the basic feedback loop raises the loop gain at the disturbance frequencies, but reduces it at frequencies between the disturbance frequencies. Consequently, while disturbance rejection is improved at the disturbance frequencies, it is degraded at the intermediate frequencies. Besides, robustness to noise and unmodelled dynaimcs is reduced, and the system can become unstable. External Model based controllers do not change the loop gain once converges, and can be more robust to noise because non-linear adaptive estimation of the periodic disturbance can be used. Further, placing the controllers outside the loop make add-on controllers possible, which allow minimum modifications to the disk drive being tested.

The repetitive control scheme used is called the integral equation approach, originally introduced in (Messner et al., 1991). The central idea behind the approach is that the repeatable runout can be represented as a linear combination of basis functions (or basis in discrete time) for the runout happening every rotation of the disk. Given the Kahunen-Loeve expansion of

$$e_r'(t) = A(q^{-1})e_r(t) = \sum_{i=1}^{j} r_i \gamma_i(t) \tag{6}$$

where $A(q^{-1})e_r(t)$ is defined as $e_r'(t)$, which is also repetitive with the same period t_n as the repeatable disturbance $e_r(t)$, $\gamma_i(t)$'s are the basis functions of $e_r'(t)$, and r_i 's are the corresponding coefficients. Because $e_r'(t)$ repeats for every period t_n given by

$$e_r'(t) = e_r'(t-t_n) \tag{7}$$

we thus can derive the sufficient condition, $\gamma_i(t) = \gamma_i(t - t_n)$. One possible choice of the basis functions is the Fourier expansion of $e_r'(t)$. We may think the basis functions, $\cos(w_1Tk) \ldots \cos(w_{n/2}Tk), \ldots \sin(w_1Tk) \ldots \sin(w_{n/2}Tk)$ in the case of Fourier expansion, as some frequency components. The more cos and sin functions are used, the more repeatable disturbance of higher harmonics can be compensated. In discrete time domain, we may rewrite (1) as

$$e_r'(k) = A(q^{-1})e_r(k) = \sum_{i=1}^{j} r_i \gamma_i(k) \tag{8}$$

or in vector form,

$$[e_r'(k), e_r'(k\text{-}1), ..., e_r'(k\text{-}N\text{+}1)] = [r_1, r_2, ..., r_j] \begin{bmatrix} \gamma_1(k) & .. & \gamma_1(k-N+1) \\ .. & .. & .. \\ \gamma_j(k) & .. & \gamma_j(k-N+1) \end{bmatrix}$$

$$= R^T [\bar{\gamma}(k) \quad .. \quad \bar{\gamma}(k-N+1)] = R^T H(k) \qquad (9)$$

where $\gamma_i(k) = \gamma_i(k - N)$, $H(k) = H(k\text{-}N)$ with N being the number of samples in one rotation of the disk, $R = [r_1, r_2, ..., r_j]^T$, $H(k) = \begin{bmatrix} \gamma_1(k) & .. & \gamma_1(k-N+1) \\ .. & .. & .. \\ \gamma_j(k) & .. & \gamma_j(k-N+1) \end{bmatrix}$, and $\bar{\gamma}$'s are the column vectors of H. Because the rank of H(k) is smaller than or equal to N, and H(k) is the range space of $[e_r'(k), e_r'(k\text{-}1), ..., e_r'(k\text{-}N\text{+}1)]$, we can assume j = N, leaving N independent row vectors without loss of generality. In this case, the same function of length N, but of different time delay is used to construct the independent row vectors $[\gamma_i(k), ..., \gamma_i(k\text{-}N\text{+}1)]$. In other words, a function of length N is used to construct the column vector $\bar{\gamma}(k)$, and it is shifted as the sampling time index *k* increases. This simplifies the implementation algorithm and enhances the computation speed, especially when implementing on a DSP because circular addressing can be used. Particularly, we can think of a function of N elements being shifted to the left as going down the matrix H(k). For example, let the function have four elements: a, b, c, and d.

Then

$$H(k) = \begin{bmatrix} a & b & c & d \\ b & c & d & a \\ c & d & a & b \\ d & a & b & c \end{bmatrix} \qquad (10)$$

By describing the repeatable disturbance as in Eq. (8), the auxiliary repetitive control law can be defined by

$$u_r(k) = -\hat{R}^T(k)\, \bar{\gamma}_{\hat{B}}(k) \qquad (11)$$

where $\hat{R}(k)$ is the estimate of the coefficient vector R defined in Eq. (9) above and

$$\bar{\gamma}(k) = \hat{B}(q^{-1})\bar{\gamma}_{\hat{B}}(k) \qquad (12)$$

Convergence of the estimated coefficient vector $\hat{R}(k)$ yields an accurate representation of the repeatable disturbance *$e_r'(k)$*. By substituting the control law (10) into the PES static equation (3), it is straight forward that the convergence of the estimate vectors $\hat{R}(k)$ and $\hat{B}(k)$ results in an effective compensation of the repeatable disturbance. Here we assume

that $\hat{B}(k)\hat{R}(k)\bar{\gamma}_{\hat{B}}(k) = \hat{R}(k)\hat{B}(k)\bar{\gamma}_{\hat{B}}(k)$ after $\hat{R}(k)$ and $\hat{B}(k)$ have converged. Also note that B/A does not have to be minimum phase, and B is not required to be stable. In Eq (11), we define a function $\bar{\gamma}_{\hat{B}}(k)$ to obtain the control law, and $\bar{\gamma}(k)$ is thus defined in Eq (12) using $\bar{\gamma}_{\hat{B}}(k)$. As long as $\bar{\gamma}(k)$, $\bar{\gamma}(k-1)$, ..., $\bar{\gamma}(k-N+1)$ are still linear independent, the repeatable disturbance $e'_r(k)$ can be represented by these linear independent vectors as shown in Eq (9), and R can be estimated by $\hat{R}(k)$. Therefore, the vectors $\bar{\gamma}_{\hat{B}}(k)$, $\bar{\gamma}_{\hat{B}}(k-1)$, ... $\bar{\gamma}_{\hat{B}}(k-N+1)$ must be chosen such that $\bar{\gamma}(k)$, $\bar{\gamma}(k-1)$, ..., $\bar{\gamma}(k-N+1)$ are linear independent.

Recall that in the estimation error dynamics the filtered repetitive control input $\hat{B}(q^{-1})u_r(k)$ is used such that $\varepsilon(k)$ converges to zero as the filtered input converges to $e_r'(k)$. The convergence to $e_r'(k)$ of the filtered input thus can be derived using the following equations,

$$\begin{aligned} \hat{R}(k) &= \hat{R}(k-1) + \Lambda(k)\varepsilon(k) \\ \Lambda(k) &= \frac{G(k)\bar{\gamma}(k)}{1+\bar{\gamma}^T(k)G(k)\bar{\gamma}(k)} \end{aligned} \tag{13}$$

where G(k) is a decaying scalar adaptation gain, the ℓ_2 norm of $\bar{\gamma}(k)$ in the denominator of $\Lambda(k)$ can be calculated as a constant for computation purpose, and $\bar{\gamma}(k)$ in the numerator of $\Lambda(k)$ can be treated as a frequency weighing function of the estimation error $\varepsilon(k)$.

By using this value of $\hat{R}(k)$ in the control law given by Eq. (11) and feeding back the control input $u_r(k)$ into the estimation error equation (5), the estimation error differential equation can be formed. Assuming that the basis function $\bar{\gamma}(k)$ is bounded and $\hat{B}(k)$ has converged before the control output $u_r(k)$ is applied, the proof showing the system is stable using hyperstability is particularly straightforward and can be found in (Tomizuka and Kempf).

3. Experimental Setup

The experimental setup consists of a JTS 3¼" Nordic III disk drive, an interface board, a 486-66 PC with TMS320C30 DSP, a HP3562A dynamics spectrum analyzer, and a HPE3630A power supply. Figure 2 below shows the schematic of the experimental setup.

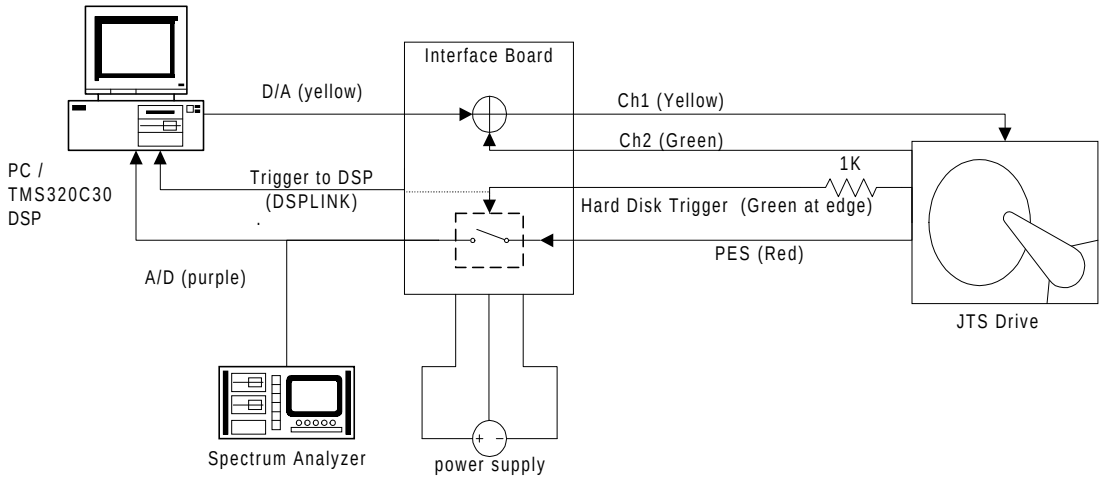


Figure 2. Experimental Setup for the Implementation of the Add-on Compensators.

Basically, the JTS drive is the plant the add-on compensators try to control (please refer to figure 1 for the overall block diagram). Referring to the notation used in figure 1, Ch1 gives the control signal from the hard disk controller u_k, whereas the red wire outputs the position error signal y. The add-on compensating algorithm is implemented on a TMS320C30 DSP, which is installed in a 486-66PC. In order for the DSP to manipulate the sampled PES and append its output to the drive, an interface board as described below must be built.

3.1 The JTS Drive

The JTS 3¼" Nordic III hard drive has a rotation speed of 4103.2 rpm (i.e. ≈ 68.39 Hz) and capacity of 810 MB. The sampling period is 152.32 μs, and thus, each track has 96 servo sectors (i.e. each rotation has 96 sample of PES). The number of servo sectors on track implies how often the repeatable runout repeats. Particularly, the repeatable runout can be described by, $e_r(k) = e_r(k-N)$, where N is the number of servo sector per track and is 96 in this case. The typical value of PES using the nominal controller is about (2.74 ± 0.150) V, whereas the nominal controlling signal is about 50 mV. Because the minimum count of the DSP A/D is 0.0916 mV, which is relatively small comparing to the signal noise of PES, further magnification of PES to enhance the

revolution is not necessary. However, the signal-to-noise ratio of the nominal control signal is so small that shielding is particularly important. Figure 3 shows the block diagram of the closed-loop servo-controller system inside the drive. The closed-loop system basically consists of a voice coil motor (VCM), a micro-controller to implement the nominal control scheme, a digital-to-analog converter (DAC) to convert the digital control signal from the micro-controller to control voltage, a pulse-width modulus to represent the digital PES measured by the head as duty-cycle-weighted square waves, and a low-pass filter working as an integrator to convert the duty cycle of the PWM to discrete voltage level.

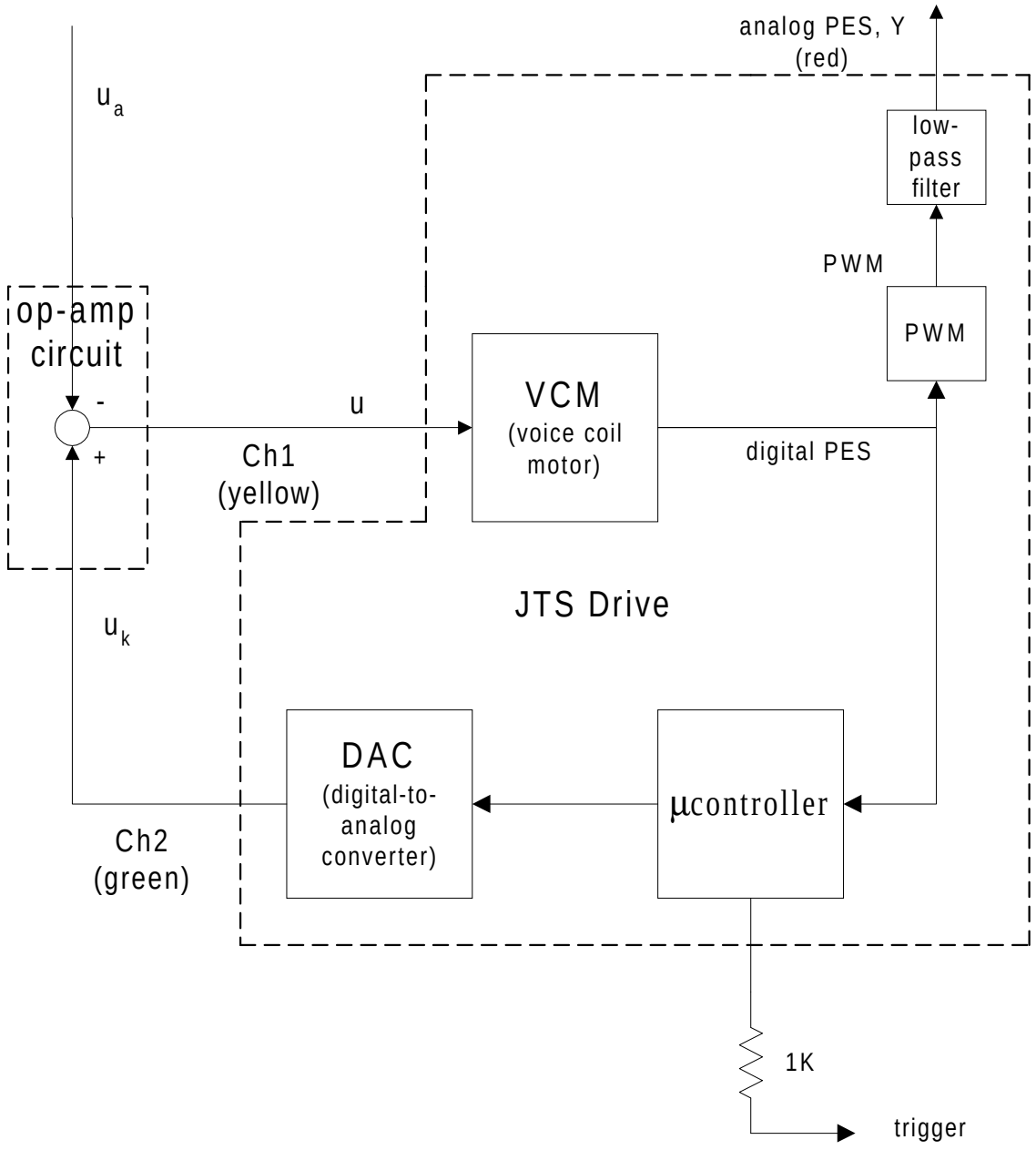


Figure 3. Block diagram of the Closed-Loop Servo-Controller System inside the Drive

The JTS disk drive is modified so that the PES can be measured, and additional signal can be appended to the internal embedded controller output signal. Physically, there are four wires pull out from the drive. PES can be measured on the red wire. The green wire is connected to CH2 which outputs the internal controller signal. The control signal, which may be appended with an additional compensating signal, must be fed back to CH1 (yellow wire) for re-calibration. In other words, CH2 must be connected to CH1 directly or indirectly in order to close the servo loop, and they are not connected inside the hard disk. The green wire at the edge of the disk drive outputs the triggering signal, whereas the black wire provides grounding. Since the trigger edge is too sharp to electro-magnetically interference the sensitive control signal from CH2, whose magnitude is about 50mV at most, shielding of the trigger signal is not enough. A resistor of 1KΩ is connected to the end of the trigger wire (green) to form a low-pass filter together with the capacitance inside the coaxial cable in order to smoothen the trigger edge.

Referring to figure 1, it is obviously that the hard disk, including the VCM and the controller, is the plant the add-on controllers trying to compensate. So the closed-loop dynamics form u_a to y must be known. Figure 4 shows the closed-loop frequency response of this system measured by a spectrum analysis with its sweeping-sin-wave feature. Besides, the repeatable runout for one cycle (96 samples) is recorded and is shown in figure 5.

The analog PES the JTS drive output is a low-passed PWM. The micro-controller inside the drive uses digital PES. To facilitate testing and measurement, the digital PES is converted to analog signal in the form of pulse-width-modulus (PWM) and then passed through a low-pass filter. All the conversion and low-pass filtering are implemented inside the drive and the only accessible analog PES is the low-passed PWM. To maintain the phase lag as minimum as possible, the corner frequency of the low-pass filter cannot not be too low. As a result, the analog PES is contaminated with triangular waves of the PWM frequency (refer to figure 8), and the amplitude of the triangular waves is at least 2/3 of the PES. Therefore, the analog PES must be further notch-filtered at the frequency of the PWM by the interface board.

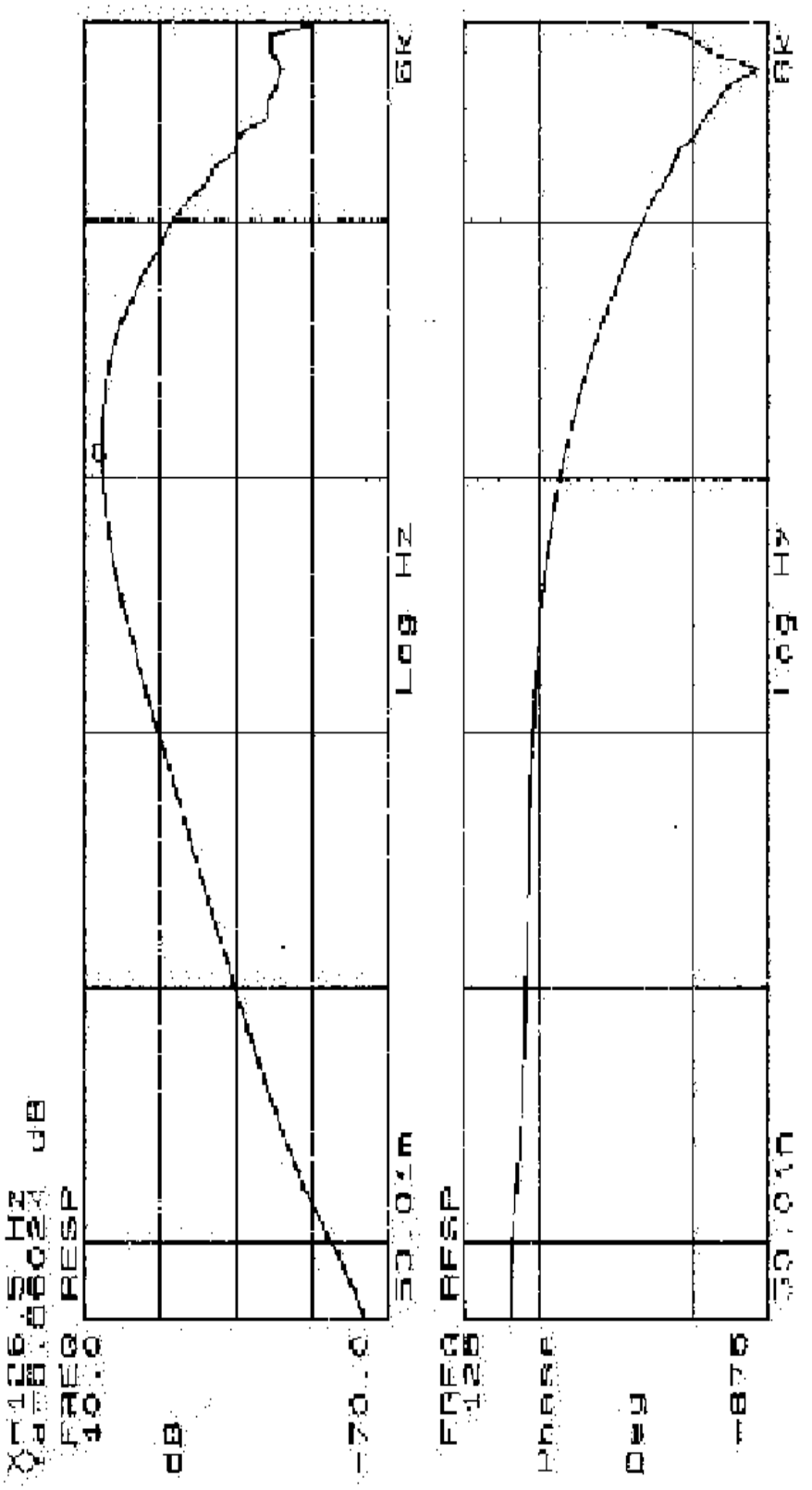


Figure 4. Closed Loop Frequency Response of the Actual Servo-controller System from $_{a}$uto y.

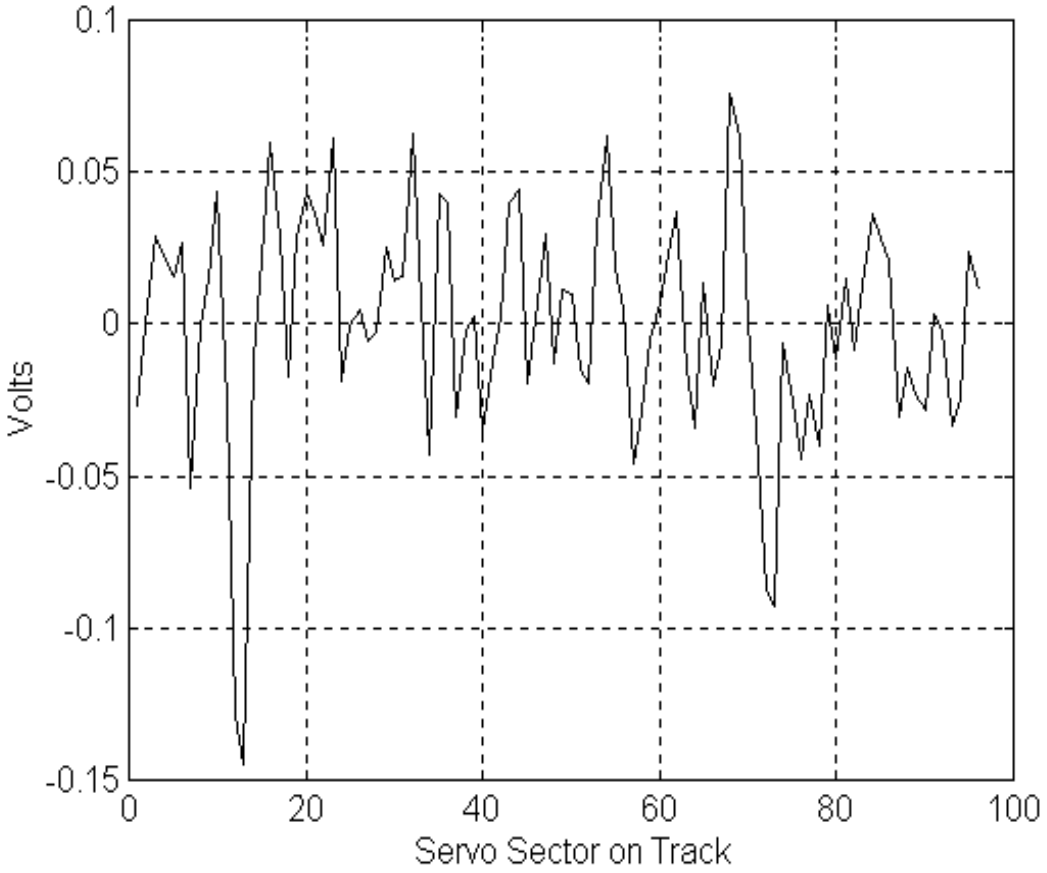


Figure 5. Repeatable runout for One Cycle of Rotation

3.2 The Interface Board

The interface board has two electrical subsystems, a control signal summer and a PES decoder. The control signal summer is used to append the control signal of the add-on compensator and any additional noise to the control signal of the hard disk controller, whereas the PES decoder provides notch filtering, low-pass filtering, and zero-order holding of the analog PES from the drive. Figure 6 shows the connections of the interface board to the DSP and to the drive. Addition noise can be appended to the control signal in order to evaluate the performance of the non-repetitive compensator. The power supply of the interface board should be larger than ±5V in order to operate the digital components.

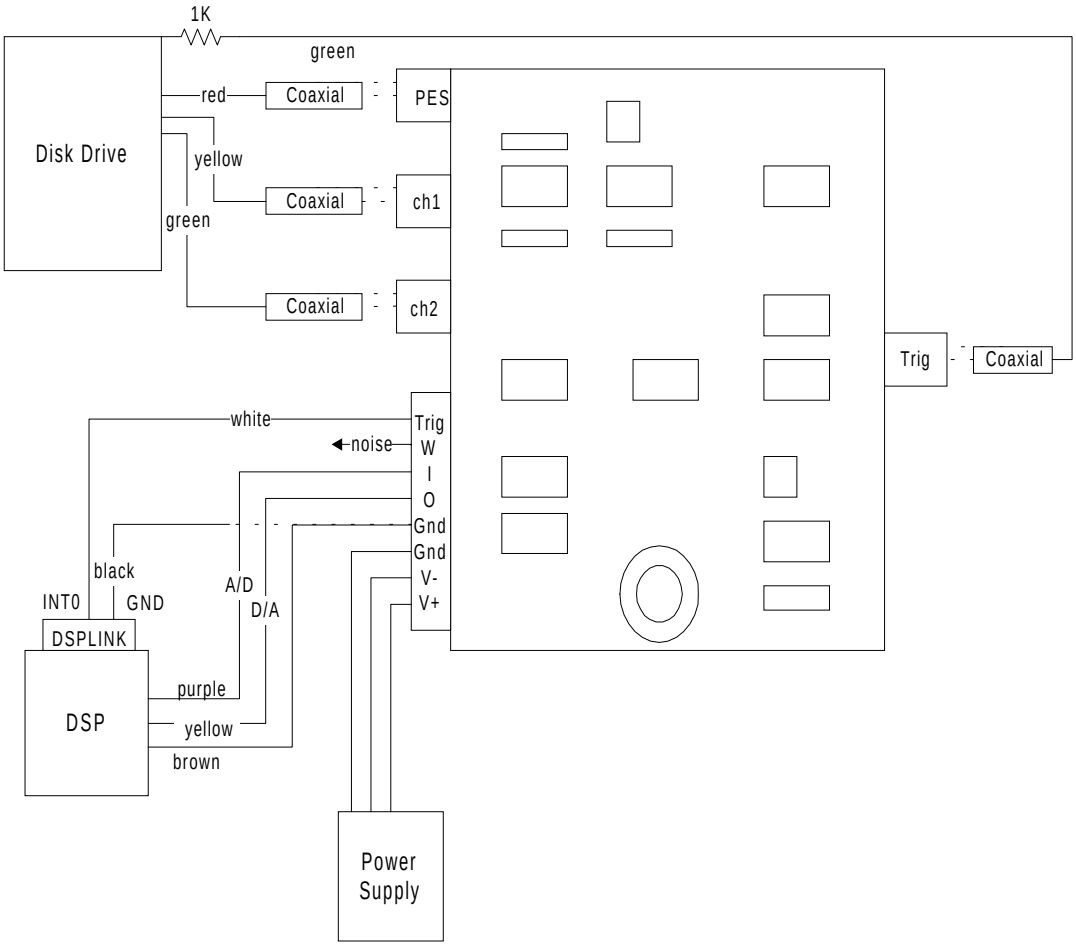


Figure 6. The Interface Board and Connections.

3.2.1 Control Signal Summer

In order to append the control of the add-on compensator to the that of the hard disk controller, an op-amp circuit similar to that shown in figure 7 must be used. The circuit in the figure consists of a differentiator, a common-mode filter, an ac-coupler, a summing amplifier, an inverting amplifier, and several voltage followers.

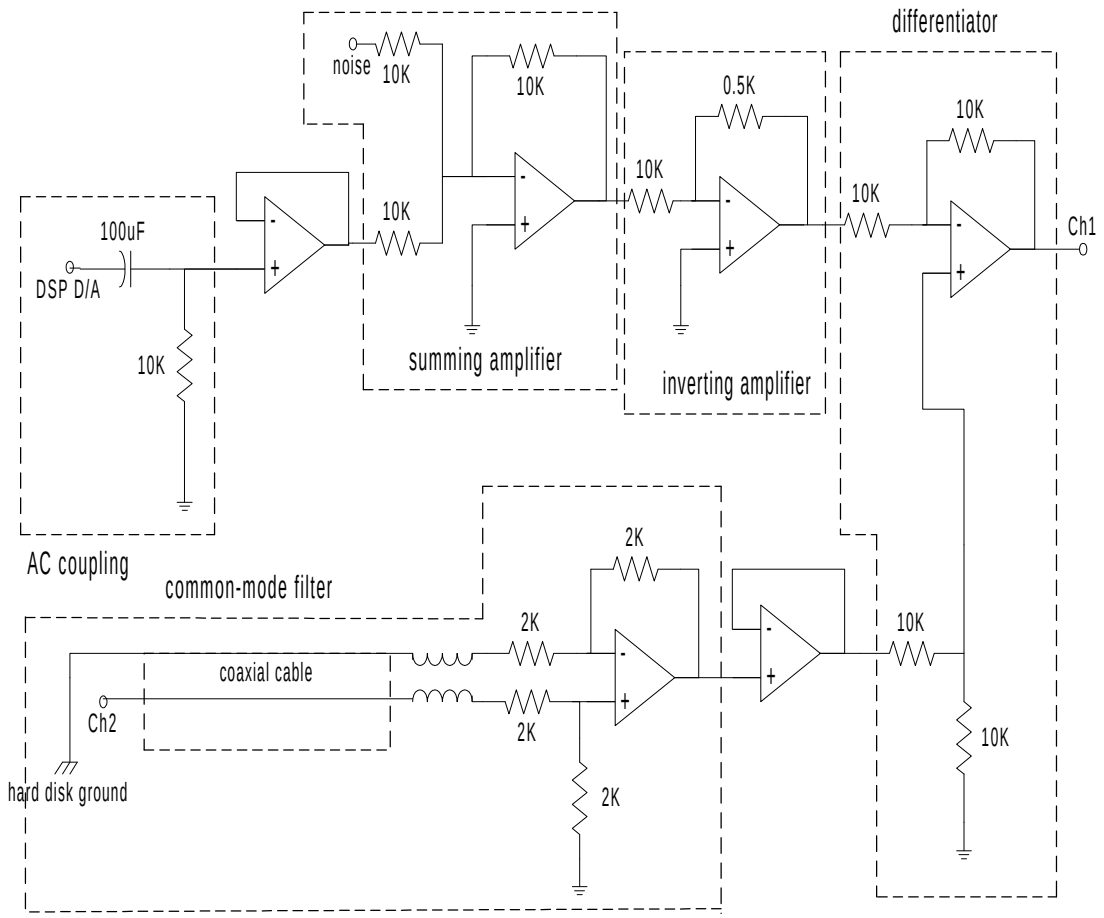


Figure 7. Schematic of the Control Signal Summer.

The major component is the differentiator which appends the add-on control signal to the hard disk control signal by the equation,

$$Ch1 = Ch2 - DSP\ D/A \tag{14}$$

The common-mode filter ensures the common grounding between the disk drive and the differentiator. Using a wire to common ground the differentiator and the disk drive is not effective to the control signal from ch2 because the signal from ch2 is very small in magnitude, about 30mV at most. Also, the distance between the differentiator and the hard drive is significantly long. For such a small signal, the inductance of the long ground wire can create enough impedance to separate both ground. Therefore, the common-mode filter is used to common ground the differentiator with the hard disk so that noise contamination to the signal from ch2 at the differentiator can be reduced. The ac-coupler is a critical component to avoid biasing the calibration of the JTS drive's VCM. Without it, any dc offset of the control signal can increase the low-frequency variance of the PES. Once the calibration is biased, the variance of PES changes even thought the dc-offset is set to zero. In that case, the hard disk must be reboot. In order to

evaluate the performance of the non-repetitive compensator, color noise is added to the control signal through the summing amplifier. Also, due to the finite revolution of the DSP D/A, the inverting amplifier is used to further diminish the DSP D/A signal by a factor of 20 in order to enhance the revolution of the add-on control signal at the expense of the signal range. Therefore, the overall input-to-output relation of the control signal summer excluding the ac-coupling is given by,

$$\text{Ch1} = \text{Ch2} - \frac{\text{DSP D / A} + \text{white noise}}{20} \tag{15}$$

3.2.2 PES Decoder

The other function of the interface board is to filter and zero-order hold the analog PES from the drive. The signal P1 in figure 8 shows a typical PES output from the drive. The PES (P1) is generated by low passing a PWM whose duty cycle represents the digital PES used in the micro-controller. To avoid phase lag, the corner frequency of the low-pass filter is designed to be as high as possible, and as a result, P1 is contaminated with triangle waves at the PWM frequency. So, P1 must be notch filtered at the frequency of the triangle wave as shown in figure 8. A low-pass filter is also placed in front of the notch filter in order to filter any noise that may induce to the cable. By optimizing between the noise attenuation and phase lag at the sampling frequency, the phase lag of the overall filter (low-pass + notch) achieves to 13° at 6kHz. Besides, only the steady state of the filtered PES (P2) in each sampling period has the error information which can be manipulated by the add-on controller. Therefore, an one-shot is used to delay the edge of the triggering signal from the drive for about 90μs, after which P2 has reached the steady state. A zero-order hold then samples the steady state value of P2 and holds it after 14μs, leaving 48μs for the DSP to update the add-on control signal to the drive. Due to the fast low-triggering INT0 on the DSP, which detects the external trigger to the ISR, a low of duration longer than 1μs can create more than one external trigger. So a third one-shot must be used to create a short low with duration of 280ns to trigger the ISR. Figure 9 shows the schematic of the PES decoder which filters P1 and zero-order holds P2.

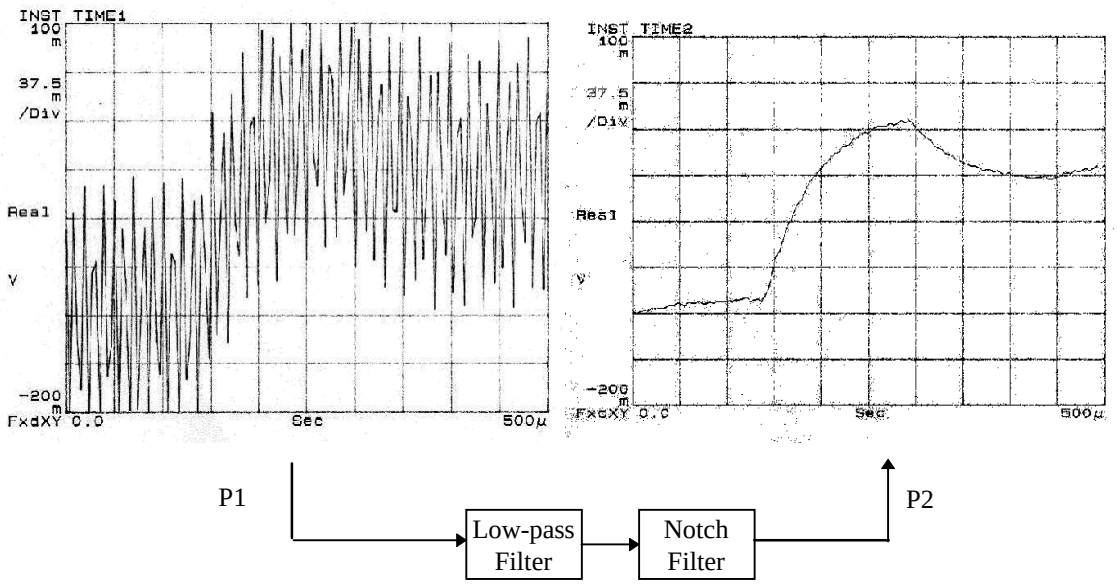


Figure 8. PES from the Drive and its Filtering.

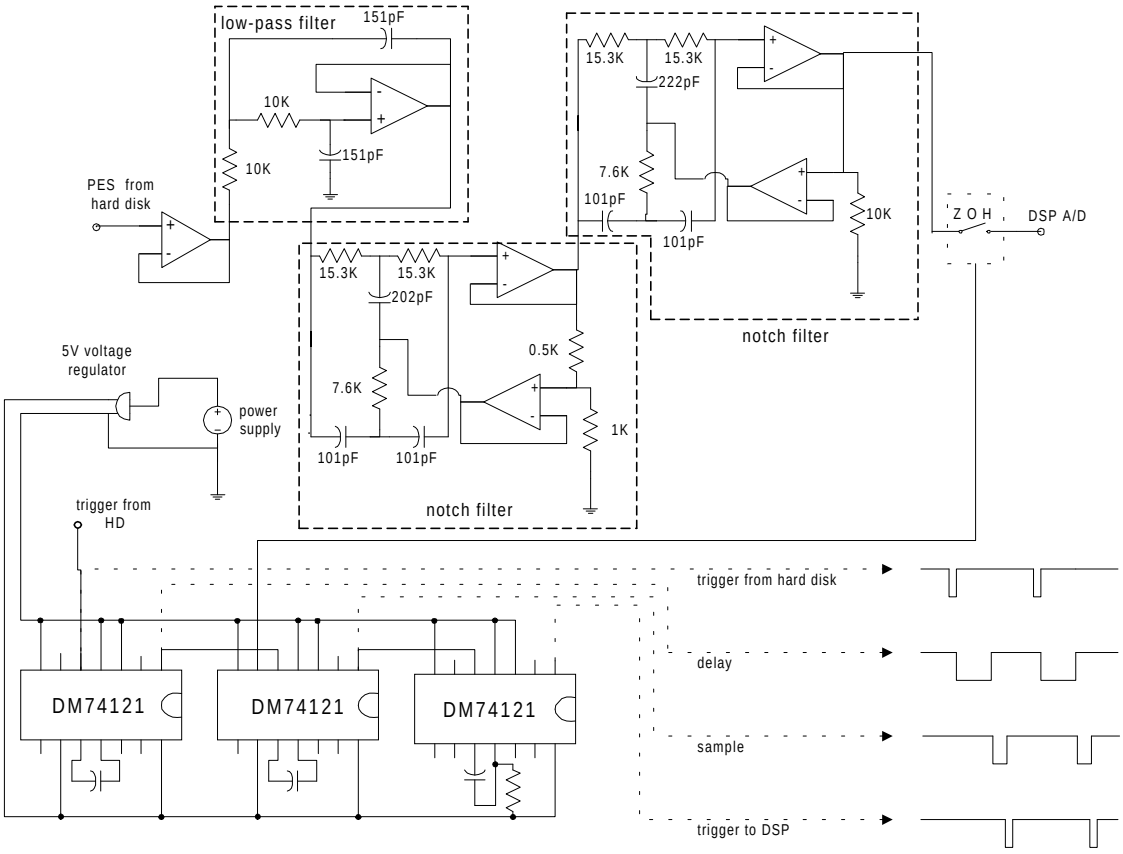


Figure 9. Schematic of the PES Decoder.

3.3 TMS320C30 Digital Signal Processor

The TMS320C30 is a digital signal microprocessor produced by Texas Instruments. The C30 is the third generation of TMS320 family of microprocessor. The TMS320C30 is a 32-bit floating-point microprocessor capable of executing up to 16.7 MIPS (million instructions per second) and 33 MELOPS (million floating-point operations per second). The instruction time of the microprocessor is 60ns.

For our applications, it is simpler to control the C30 by using the SDS30 Debug Monitor. The SDS30 is an easy to use Debug Monitor. Some of the functions of the SDS30 Debug Monitor include: downloading a complied assembly language program into the microprocessor's memory, displaying the instructions of the program that was downloaded, providing the ability to put breaks on the code to be able to monitor the contents of the registers and of the memory at different stages of the program, and transferring data from the specified addresses of the C30 memory to a data file. All these characteristics facilitate the debugging process. With respect to the memory maps, our board only has area A-bank0, area B-bank3, RAM0, and RAM1. Other area of memory are used for memory mapped devices, such as analog I/O and DSPLINK (digital ports). Before discussing the assembly program, additional details about the analog interfaces, links, and DSPLINK should be explained.

3.3.1 Analog Interfaces

The TMS320C30 system board has two channels, each of them with A/D and D/A converters. The A/D and D/A of the same channel share the same address, and the converters of different channels are located at the following addresses:

Channel A, A/D & D/A at address 804000h
Channel B, A/D & D/A at address 804001h

Each channel uses the 16 MSB (most significant bits) of the register at the given address, and data is represent in 16-bit 2's complement format (16-bit integer). To output a voltage, you must write the corresponding fixed-point number to the desired channel (804000h or 804001h) and write a dummy to 804008h to generate a conversion trigger. Two other options are available to generate a conversion trigger: through timer 1 and by an external trigger. To read a voltage from the A/D, a conversion trigger is generated, and the corresponding value can be read at the 16 MSB of the desired channel. Both A/D and D/A processes complete in 6μs after the conversion trigger is generated. Since the A/D's and D/A's use the same register for output and for input. With no processor intervention, the A/D value will be shifted into the register, with the previous register contents being simultaneously shifted to the D/A for every conversion trigger. Therefore, if no value is written to the register, the analog input signal on each channel will, by default, be echoed directly to the corresponding analog output channel, with one conversion trigger delay. To avoid echoing the A/D value through the D/A in the

following conversion trigger, the previous D/A value can be written to the channel register. Section ?? describes one possible solution to this problem. Besides, the value coming through the A/D must be read before a value is written to be sent out through the D/A in order to avoid overwriting the newly A/D converted voltage.

The D/A can output a voltage between +3 and -3 volts. The conversion factor from a fixed-point 32-bit number to a voltage at the output of the D/A is thus

$$32767*16^4 \text{ decimal} / 3 \text{ volts}$$

This number automatically writes the desired number to the 16 MSB. To convert the value read by the A/D's to volts, the 32-bit number read should be right shifted by 16 bits.

For the experiments that were performed in the implementation of an adaptive repetitive control for disk file servos, only channel B was used to input and output signals. Once trigger happens at DSPLINK's INT0, PES on the A/D of channel B is taken into the DSP by generating a software conversion trigger at 804008h. Then the digital PES is manipulated by the repetitive algorithm to update the control signal, which is given out to the D/A of channel B. Since the value on channel B's register (804001h) from the previous sample is echoed through the D/A when triggering the A/D conversions, the control signal from the previous sample must be written to the register before reading the A/D in order to maintain the voltage at the D/A. Further discussions of how the control signal is updated can be found in section 4, Assembly Programs.

3.3.2 Links

The links are small jumpers that are used on the TMS320C30 system board to select different options by making hardware connections. For the experiments that were performed in the implementation of an adaptive repetitive control for disk file servos, link #2 was not connected, link #10 was placed at position b, and link #12 was, placed at position a.

3.3.3 DSPLINK

DSPLINK is a bi-directional digital expansion interface that allows digital input and output. By using DSPLINK, the construction of memory mapped digital ports is greatly simplified. The digital ports can be mapped between addresses 800000h and 801FFFh. Only the 16MSB at those addresses can be accessed. DSPLINK connector is the physical connector with the address lines, the data lines, and other related signals such as w/r, reset, INT 0/1, and clock. The address and data lines only get activated when a read or write operation is performed at an address between the 800000h and 801FFFh range. Therefore, when building a hardware device to interface with the DSPLINK, the designer need not to decode all the address lines, since memory operations outside this range will not interface with the DSPLINK connector. It can be thought of as having a

separate bus for DSPLINK. For the experiments that were performed in the implementation of an adaptive repetitive control for disk file servos, only INT0 and digital ground are connected to the interface board.

4.0 **Simulated Results**

4.1 Simulation of the Disturbance Models and Plant

The objective of the add-on compensators is to minimize the variance of the PES. The mean of the PES is not important in the aspect of the add-on compensators because the disk drive internal controller is able to minimize the steady state error. So we can assume that the PES measured from the disk drive is zero mean. To obtain the variance of the PES, we can measure the spectrum magnitude/density of the PES. Figure 10 shows the spectrum magnitude of the nominal PES used in the simulations. The spectrum diagram shown consists of the repetitive and non-repeatable runout. Since the drive spans at 68.39 Hz, the repeatable runout has the spectrum magnitude at the multiples of this frequency (i.e. 68.39 Hz, 138.8 Hz, 208.2 Hz and etc.) The spectrum at these frequencies is simulated by using a record of the repeatable runout in one rotation of the drive (refer to figure 5). The spectrum magnitude at the other frequencies are due to the non-repeatable runout. The spectrum is thus formulated by modeling the stochastic disturbance model $\frac{\hat{C}(k)}{\hat{A}(k)}$, which is then used to filter some white noise, whose spectrum magnitude is 1 at all frequencies.

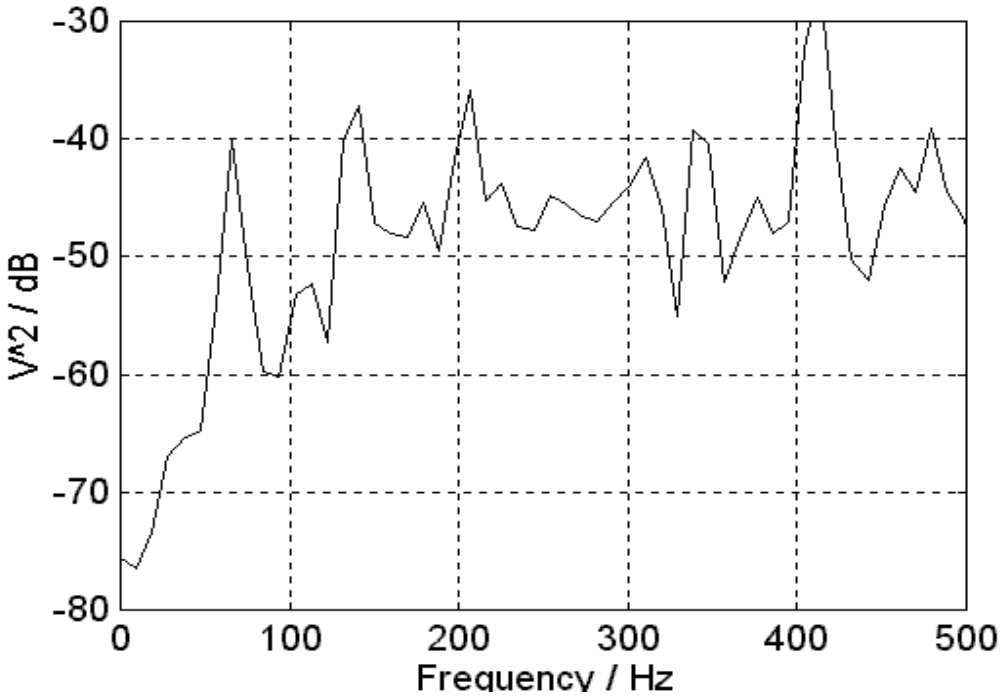


Figure 10. Simulated Spectrum Magnitude of the Nominal PES

Even though the VCM-actuator dynamics is simply a second-order spring-mass system, the controller-actuator closed-loop system is not. As shown in figure 11, it is

formulated as a seventh-order SISO system for simulation purposes. In order to minimize computation time and memory usage in the implementations, the controller-actuator closed-loop system is identified as a fourth-order transfer function in the simulation and experiment. Using the recursive least square algorithm, the identified fourth-order transfer function of the plant is

$$G'_{uy} = \frac{0.016z^{-1} - 0.036z^{-2} + 0.176z^{-3}}{1 - 1.4192z^{-1} + 0.4045z^{-2} + 0.1115z^{-3}} \quad (16)$$

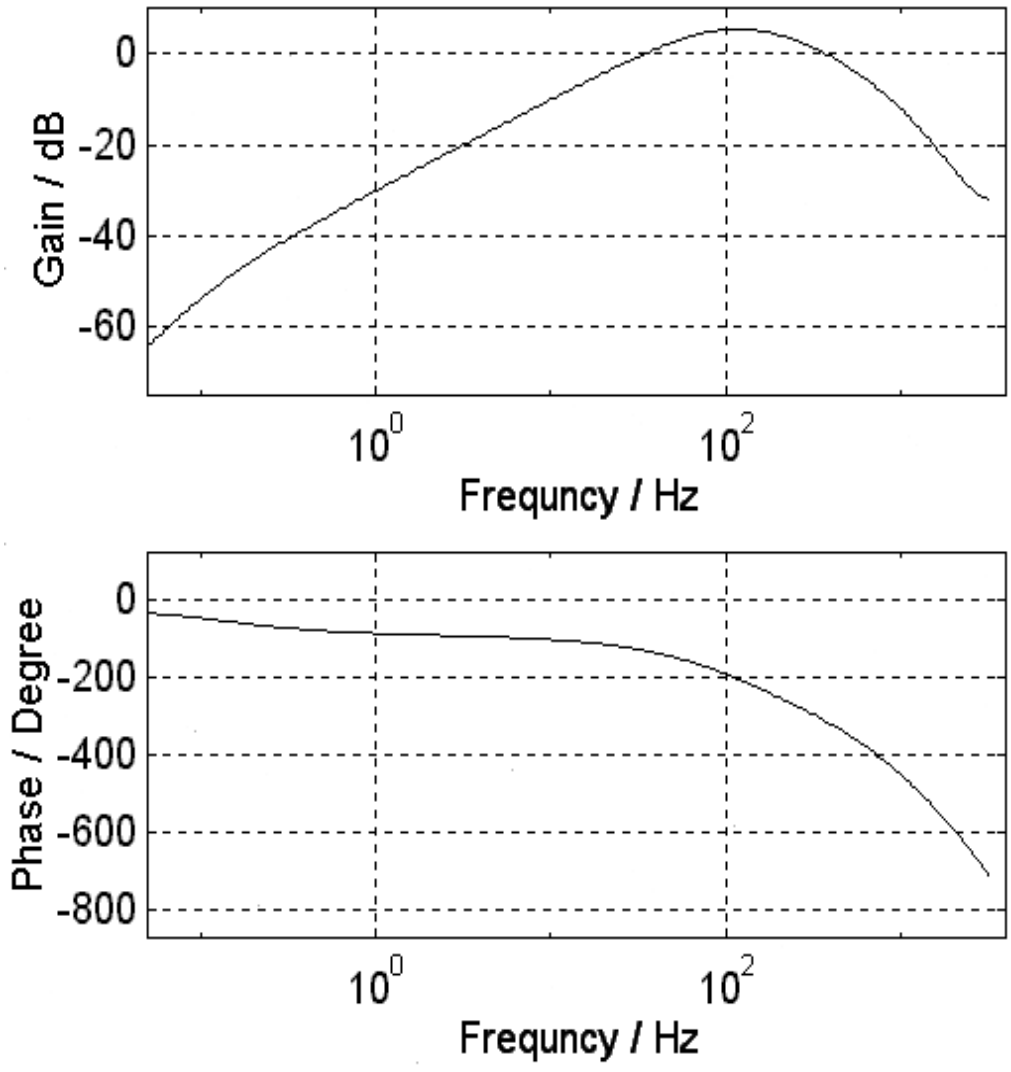


Figure 11. Discrete Bode Plot of the Simulated Closed Loop Transfer Function from the Add-on control Signal to PES with Nyquist frequency at 3283 Hz.

$$Guy = \frac{0.0023z^{-1} - 0.0017z^{-2} - 0.1219z^{-3} + 0.2167z^{-4} - 0.0719z^{-5} - 0.0234z^{-6}}{1 - 3.6534z^{-1} + 5.2527z^{-2} - 3.7650z^{-3} + 1.4089z^{-4} - 0.2620z^{-5} + 0.0189z^{-6}} \quad (17)$$

4.2 Kernel Function and Performance

Given the plant by G_{uy} and the identified plant by G'_{uy}, the repetitive control scheme is simulated to compensate for the repeatable runout. In the simulations and experiment, the kernel function $\bar{\gamma}_{\hat{B}}(k)$ is defined by,

$$\bar{\gamma}_{\hat{B}}(k) = [f(0,\ k),\ f(1,\ k),\ ...,\ f(95,\ k)]^T \tag{18}$$

where f(n, k) is a Gaussian function given by,

$$f(n,k) = \begin{cases} \dfrac{1}{\sqrt{2\sigma\pi}} \exp\left(\dfrac{-x^2}{2\sigma^2}\right) & \text{and} \quad x = 3\sigma\dfrac{n}{N} \quad \text{for} \left|\text{remainder}\left(\dfrac{n-k}{N}\right)\right| < N \\ 0 & \text{for} \left|\text{remainder}\left(\dfrac{n-k}{N}\right)\right| \geq N \end{cases} \tag{19}$$

where σ is 10. Each cycle of the kernel function can be interpreted as a function consisting of a zero and a non-zero vector (kernel = [non-zero vector ^{T}, zero vector ^{T}] ^{T}). The non-zero vector is defined by sampling the Gaussian function,

$$g(x) = \frac{1}{\sqrt{2\sigma\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \quad |x| < 3\sigma \tag{20}$$

by N-1 data points. Since each cycle of the kernel function has a fix number of data points (96 for the JTS drive). Increasing the number of samples on g(x), *N*, thus decreases the length of the zero component. As a result, the function g(x) spread out over 2N-1 samples, flattening the kernel function. Therefore, N defines the flatness of the kernel function. Recall from section 2 that the kernel vector has two functions. One is to provide the basis to estimate the repeatable disturbance e_r, and second is to frequency-weigh the estimation error ε. Eq(13) shows that H(k) is symmetrical, so the basis in time domain is also given by $\bar{\gamma}_{\hat{B}}(k - k_1)$, where k_1 indicates any shifting of the kernel function. Obviously, the kernel function $\bar{\gamma}_{\hat{B}}(k)$ gives the frequency content of the basis. Note that H(k) still has full rank or, in other words, all the rows γ_i are linear independent though the basis are the same in frequency domain. Besides, from Eq (13), $\bar{\gamma}(k)$ is used as a fundamental formulation of $\hat{R}(k)$, so we may think it as a frequency weighing function of the estimation error. The reason why a Gaussian function is used as the kernel is thus because it looks much like a low-pass filter in the frequency domain, giving a low frequency weighing on the estimation error and providing basis of a frequency continuum for the estimation of the repeatable disturbance. Figure 12 shows the two kernel functions of different flatness, and figure 13 shows their frequency content.

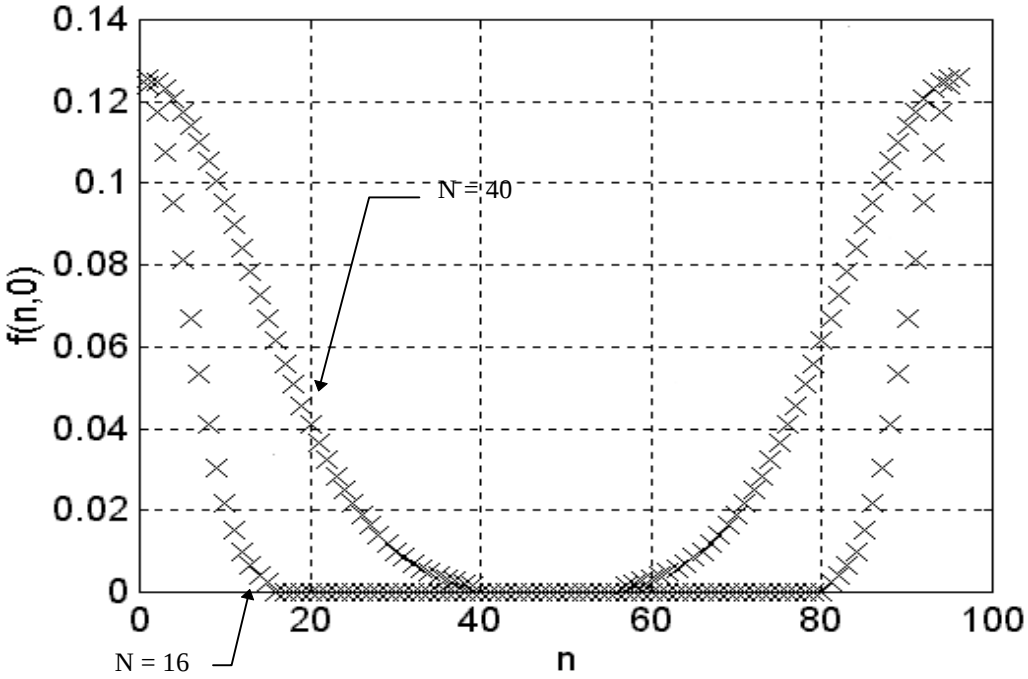


Figure 12. Two Kernel Functions $\bar{\gamma}_{\hat{B}}(0)$ of Different N

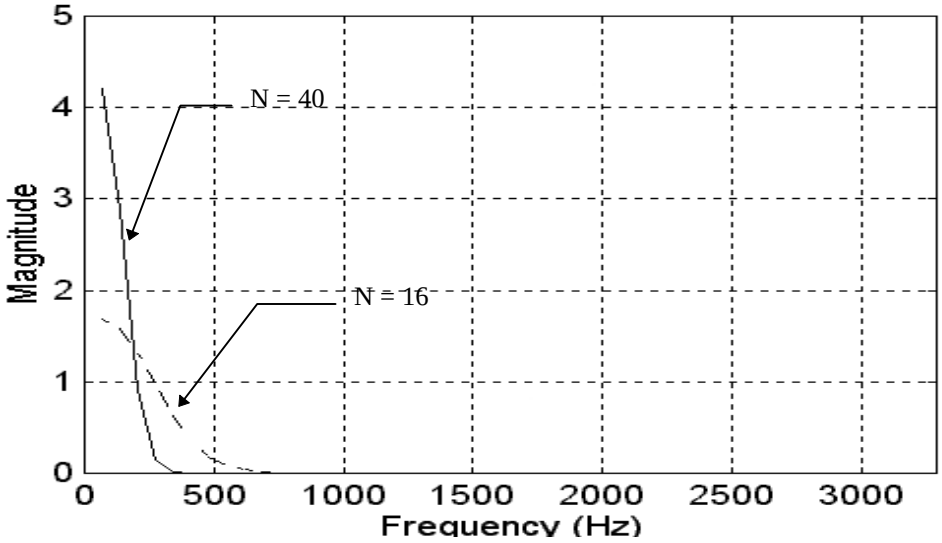


Figure 13. The Kernel Functions in Frequency Domain

It illustrates that the flatter the kernel is (larger N), the less high frequency content it has. Therefore, reducing N results in compensation for the repeatable runout of higher harmonic frequencies because the basis has more high frequency content, and the

estimation error is more highly frequency-weighed. Figure 14 and figure 15 show the simulated spectrum magnitude of the PES with and without the adaptive repetitive compensation using kernel functions of different N (flatness). By comparing these figures, we can find that more of the harmonic frequencies (up to the fifth) can be compensated using the kernel function of N = 16 while using the kernel of N = 40 can only compensate for the first two harmonics. It is also of our interests to show the convergence of $\hat{R}(k)$. Because $\hat{R}(k)$ has 96 terms, only several terms are shown here. Figure 15 shows the convergence of the 11th, 21st, 31st, 41st, 51st, 61st, 71st, and 81st terms of $\hat{R}(k)$ for the first 1100 samples, and figure 16 shows the convergent $\hat{R}(k)$ using the kernel of N = 16.

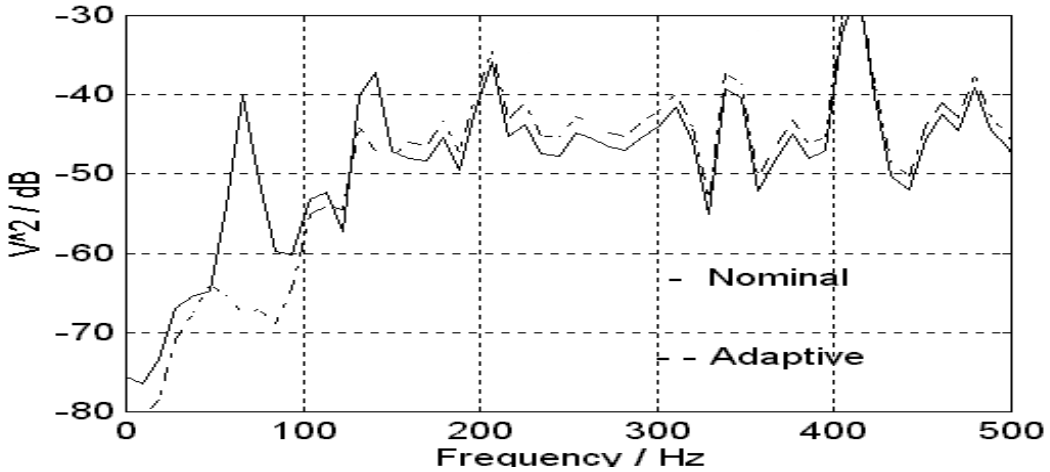


Figure 13. Simulated Spectrum Magnitude of Nominal and Adaptive PES using Kernel Function of N = 40

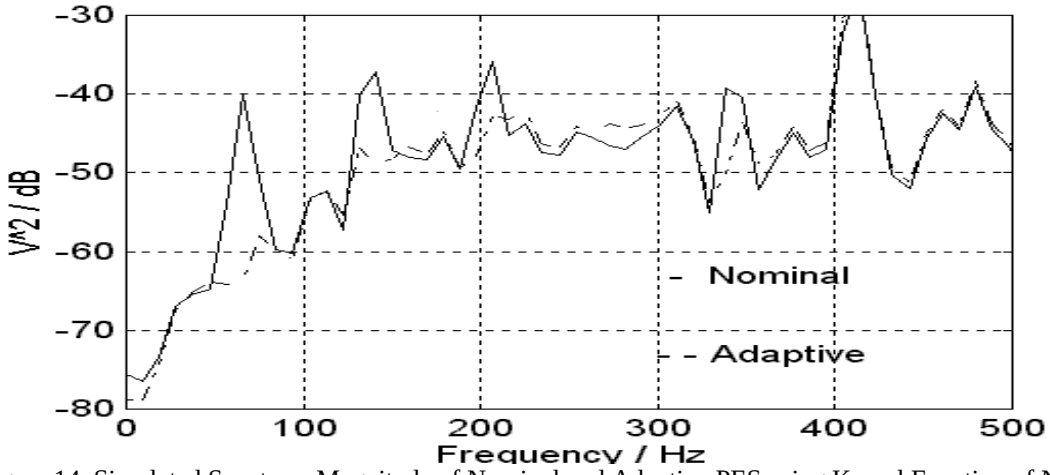


Figure 14. Simulated Spectrum Magnitude of Nominal and Adaptive PES using Kernel Function of N = 16

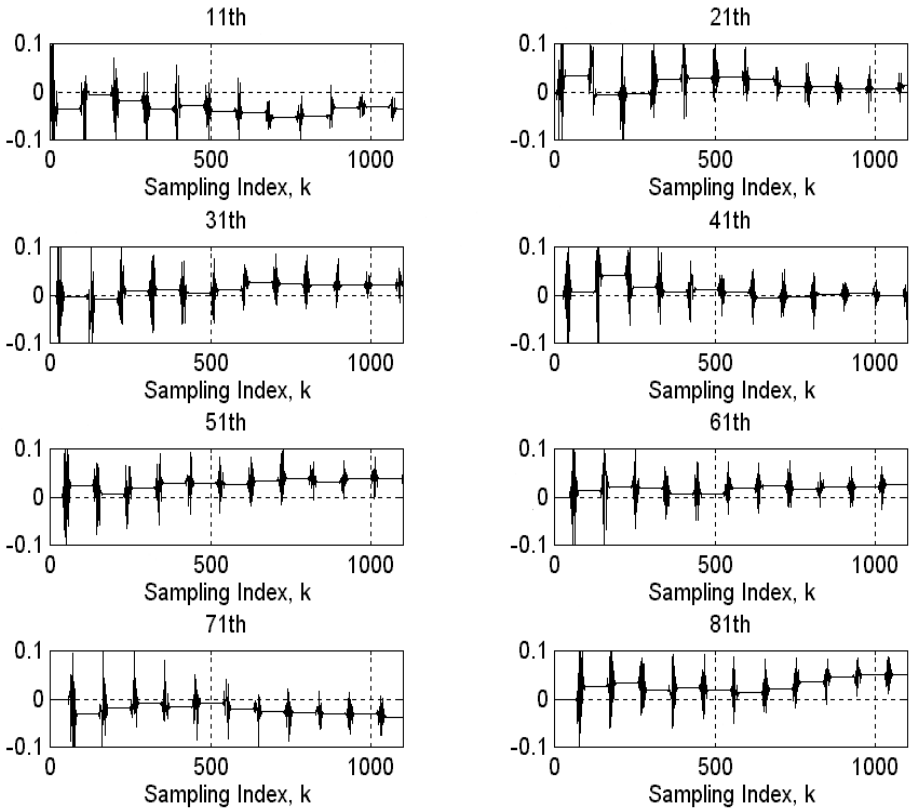


Figure 15. Convergence of the estimation $\hat{R}(k)$ in the first 1100 samples using the Kernel of N =16

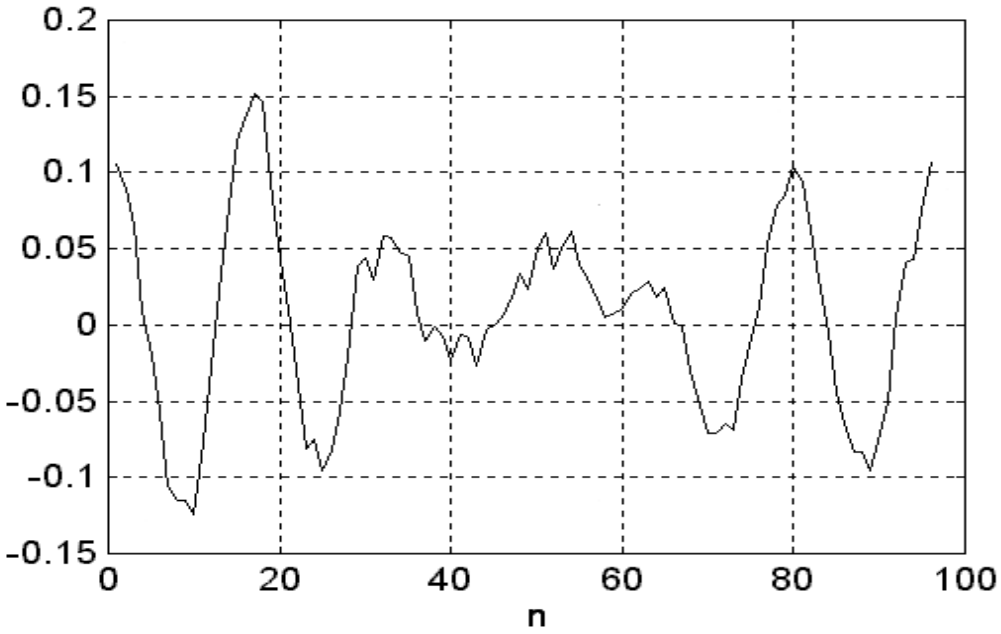


Figure 16. Convergent $\hat{R}(k)$ using the Kernel function of N =16

5.0 Assembly Program

The add-on compensators are implemented on a TMS320C30 DSP. To enhance the computation efficiency, the algorithm is written in assembly language. The control algorithm has two phases, identification phase and control phase. The former identifies the plant parameters of the hard disk closed-loop system from u_a to y using the recursive least square algorithm, whereas the latter compensates for the repetitive and non-repeatable runout using the adaptive repetitive algorithm described in section 2 and the adaptive non-repetitive scheme given in (Horowitz and Li, 1996; Horowitz *et al*., 1996). So the identification phase must be carried out to obtain the convergent plant parameters before the control phase can compute the auxiliary control outputs.

5.1 Implementation Method

The DSP's INT0 is set to 'external triggering' in order to synchronize the DSP with the hard disk. Once a trigger occurs on the hard disk, which is connected to INT0 through the interface board, the interrupt service routine of the assembly program starts, suspending any job in process. The program has two interrupt service routines, one for the identification phase and one for the control phase. In the identification ISR, the PES is read through the A/D of channel B, and a pseudo binary random control output is sent to the disk drive through channel B's D/A. θ and ϕ are then updated in the remaining of the program. In the control ISR, the A/D of channel B is read, and the auxiliary control outputs (u_q and u_r) in channel B's D/A are updated using the PES just input from channel B's A/D. To allow the control outputs to be updated as soon as a new PES is available, the computations involved in between reading from and writing to channel B's register must be minimized. Therefore, the repetitive control laws in Eq. 11 is rewritten as,

$$\begin{aligned} u_r(k) &= -\left(\hat{R}(k-1) + \frac{G(k)\bar{\gamma}(k)}{1 + \bar{\gamma}^T(k)G(k)\bar{\gamma}(k)}\varepsilon(k)\right)\bar{\gamma}_{\hat{B}}(k) \\ &= -\left(\underbrace{\hat{R}(k-1)\,\bar{\gamma}_{\hat{B}}(k)}_{REP1} + \underbrace{\frac{G(k)\bar{\gamma}(k)\bar{\gamma}_{\hat{B}}(k)}{1 + G(k)|\bar{\gamma}|^2}}_{REP2}\varepsilon(k)\right) \end{aligned} \qquad (21)$$

where $\bar{\gamma}^T(k)\bar{\gamma}(k)$ is computed as a constant $|\bar{\gamma}|^2$ omitting the discrete time index *k*. With Eq (21), the terms REP1 and REP2 can be pre-computed, and the control output $u_r(k)$ can be updated as soon as the estimation error $\varepsilon(k)$ is available. Besides, referring to Eq (13), $\hat{R}(k-1)$ is updated using $\varepsilon(k-1)$. So when implementing the algorithm, $\hat{R}(k)$ is updated and then REP1 is computed. In section 2, $\bar{\gamma}(k)$ and $\bar{\gamma}_{\hat{B}}(k)$ is down-shifted given by Eq (10) as k increases. In the assembly program, however, $\hat{R}(k)$ is up-shifted in each sampling period instead. For a $\hat{R}(k)$ of four terms, a, b, c, and d, $\hat{R}(k)$ is up-shifted as below,

$$\hat{R}(k) = \begin{bmatrix} c \\ d \\ a \\ b \end{bmatrix} \qquad \hat{R}(k-1) = \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix} \qquad \hat{R}(k-2) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Therefore, updating of $\hat{R}(k)$ includes up-shifting $\hat{R}(k-1)$ and adding $\varepsilon(k)$ to it.

By taking advantages of the circular addressing feature provided by C30, the pointer to $\hat{R}(k)$ is shifted downward. In the experiment carried out for the repetitive compensator, the following control law is also tested,

$$\begin{aligned} u_r(k) &= -\hat{R}(k-1)\bar{\gamma}_{\hat{B}}(k) \\ &= -\left(\hat{R}(k-2) + \frac{G(k-1)\bar{\gamma}(k-1)}{1 + \bar{\gamma}^T(k-1)G(k-1)\bar{\gamma}(k-1)}\varepsilon(k-1)\right)\bar{\gamma}_{\hat{B}}(k) \qquad (22) \\ &= -\left(\underbrace{\hat{R}(k-2)\,\bar{\gamma}_{\hat{B}}(k)}_{\text{REP1}} + \underbrace{\frac{G(k-1)\bar{\gamma}(k-1)\bar{\gamma}_{\hat{B}}(k)}{1 + G(k-1)|\bar{\gamma}|^2}\varepsilon(k-1)}_{\text{REP2}}\right) \end{aligned}$$

In this case, $\hat{R}(k-2)$ is updated using $\varepsilon(k-2)$, and $\hat{R}(k-2)$ is up-shifted by 2 steps with respect to $\bar{\gamma}_{\hat{B}}(k)$. Therefore, REP1 is computed before $\hat{R}(k)$ is updated using the $\varepsilon(k)$ in that sampling period. Since REP1 is computed first, $\hat{R}(k-2)$ must be up-shifted by 2 steps before computing REP1 in order to match each elements of $\hat{R}(k-2)$ with $\bar{\gamma}_{\hat{B}}(k)$. To convert the algorithm back to that shown in Eq (21), three changes have to be made. First, change the estimation error used to update $u_r(k)$ from $\varepsilon(k-1)$ to $\varepsilon(k)$. This can be found in ISR2. Second, change $\bar{\gamma}(k-1)\bar{\gamma}_{\hat{B}}(k)$ to $\bar{\gamma}(k)\bar{\gamma}_{\hat{B}}(k)$ by not shifting FGAU when computing GAU_FGAU in the subroutine GAU_DFGAU. Third, compute REP1 as $\hat{R}(k-1)$ by not up-shifting $\hat{R}(k)$ by 2 steps in the subroutine REP and updating $\hat{R}(k)$ before calculating REP1.

5.2 Program Flow

Before looking to the flow of the program, we must explain the following variables and arrays to be used in the program.

Table 1. Variables and Arrays used in the Assembly Program

GAU	It is the kernel function, $\bar{\gamma}_{\hat{B}}$. In this case, we use a Gaussian function given by,

	$\bar{\gamma}_{\hat{B}}(x) = \begin{cases} \frac{1}{\sqrt{2\sigma\pi}} \exp(\frac{-9x^2}{2N^2}) & \text{for } \left\lvert \text{remainder}(\frac{x}{96}) \right\rvert < N \\ 0 & \text{for } \left\lvert \text{remainder}(\frac{x}{96}) \right\rvert \geq N \end{cases}$ where σ is 10, x are all integers from 0 to 95, and N is a number of the user's choice. Note that the arrangement of GAU array is circular such that x can also be interpreted as all negative integers from -95 to 0. The function above is basically identical to Eq (19) with N being the number of samples used for the non-zero elements. So, the larger N is, the flatter GAU is. The function above consists of an array of N's non-zero elements, an array of zeros, and an array of (N-1)'s non-zero elements. To minimize the computation time and memory usage, GAU only stores the non-zero arrays. This breaks GAU into two parts, GAU1 and GAU2. A typical GAU is shown in figure 17. GAU has three pointers, PT1_GAU, PT2_GAU, and PT3_GAU. The addresses to which these pointers are pointing is also shown in figure 17.
FGAU	It stores the B-filtered kernel function, $\bar{\gamma}$. Before $\hat{B}$ is identified, it has a structure similar to GAU. FGAU consists of two parts, FGAU1 and FGAU2. When filtered by $\hat{B}$, the elements at the larger addresses are treated as the latest information (i.e. FGAU = [FGAU(k-95), FGAU(k-94), …, FGAU(k)]^T). Before filtered, FGAU1 has three components. It has (NB-1)'s previous values of GAU to be filtered by $\hat{B}$ to obtain the first element of the filtered FGAU, GAU1, and (NB-1)'s zeros to store the (NB-1)'s non-zero elements of the filtered FGAU at the end of GAU1. This structure facilitates the filtering process. Figure 18 shows a typical FGAU before and after it is filtered by $\hat{B}$. Similarly, FGAU has three pointers, PT1_FGAU, PT2_FGAU, and PT3_FGAU.
RHAT	It stores the estimation $\hat{R}(k)$. In the implementation of the repetitive algorithm, instead of down-shifting $\bar{\gamma}_{\hat{B}}$ and $\bar{\gamma}$, $\hat{R}(k)$ is up-shifted. By taking advantages of the circular buffering feature provided by C30, the pointer to RHAT (i.e. PTU_RHAT) is shifted downward instead of moving around the elements of RHAT. RHAT has another pointer called PT_RHAT, which is fixed and is used to show the convergence of several elements of RHAT array.
REP1	It stores the result of REP1 as defined in Eqs. (21) and (22)
REP2	It stores the result of REP2 as defined in Eqs. (21) and (22)
k_DECAY	It is the inverse of the adaptation gain *1/G(k)* defined in Eqs (21) and (22). To guarantee convergence of $\hat{R}(k)$, G(k) must decay to 0 as k -> ∞. This is made possible by incrementing k_DECAY.

k_DECAYR	It is G(k) defined in Eqs (21) and (22).
k_INC	It is the increment of k_DECAY.
GAU_NORM	It stores $\|GAU\|^2$
DATA	It stores the first 1100 samples of eight elements of RHAT array
GAU_FGAU	$\bar{\gamma}_{\hat{B}}(k)\bar{\gamma}(k-1)$
KAPPA	$KAPPA = \frac{k\_DECAYR}{1 + k\_DECAYR * GAU\_NORM}$

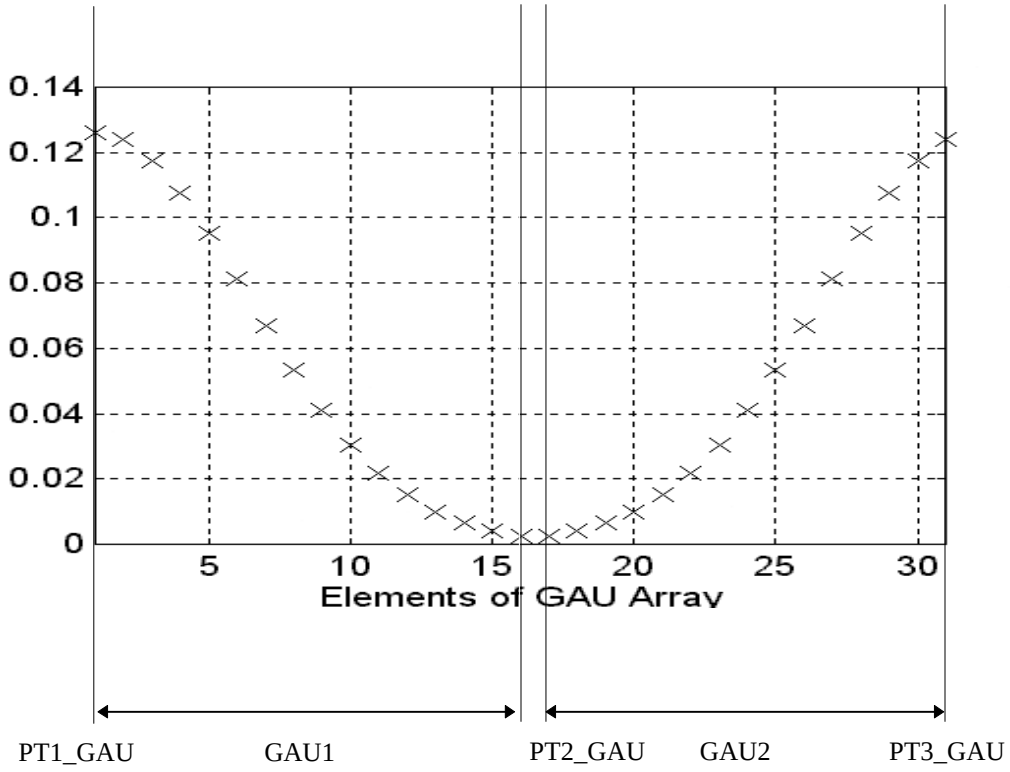


Figure 17. A Typical Structure of GAU Array and its Pointers

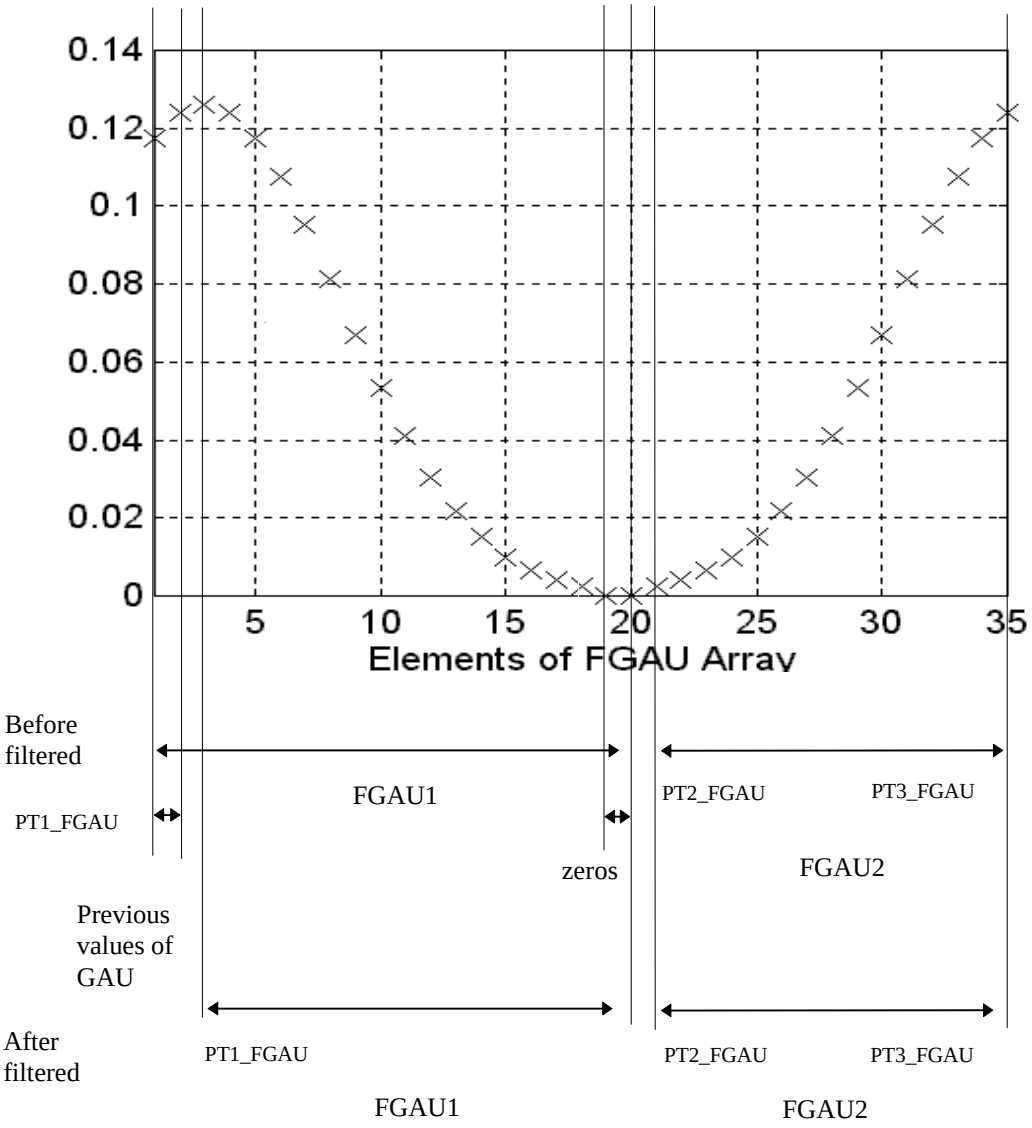


Figure 18. A Typical Structure of FGAU Array and its Pointers before and after filtered

With the variables and arrays defined above, we can now look at the flow chart shown in figure19. As mentioned above, ISR1 is used in the identification phase obtain the plant parameters $\frac{\hat{B}(k)}{\hat{A}(k)}$, where $\hat{B}$ and $\hat{A}$ are both fourth-order polynomial of the delay operator q^{-1}. ISR2 is the heart of the program. It contains the adaptive algorithm to compensate for the repetitive and non-repeatable disturbances. So, the details of ISR2 is shown separately in figure 20.

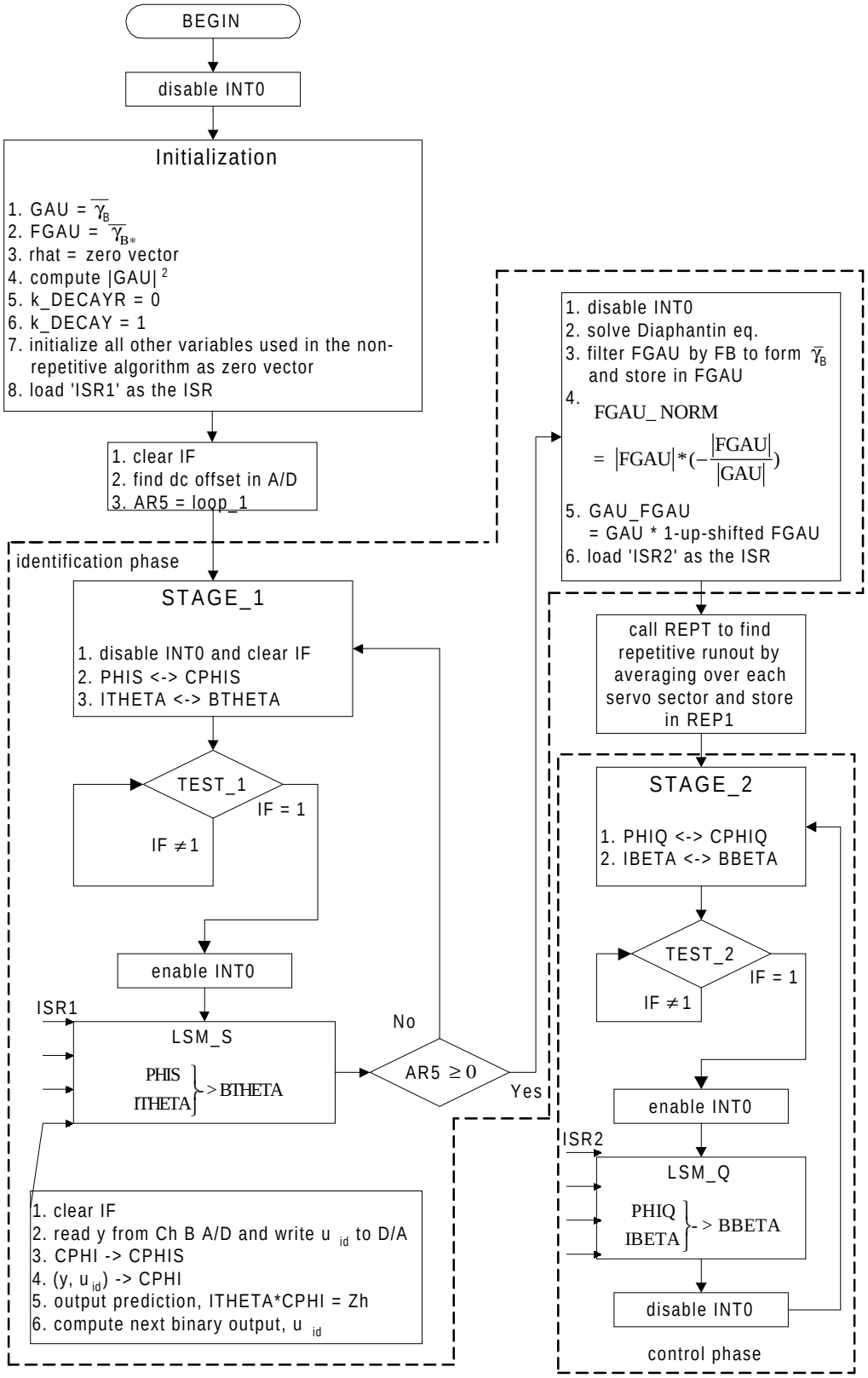


Figure 19. The Flow Diagram of the Adaptive Control Scheme

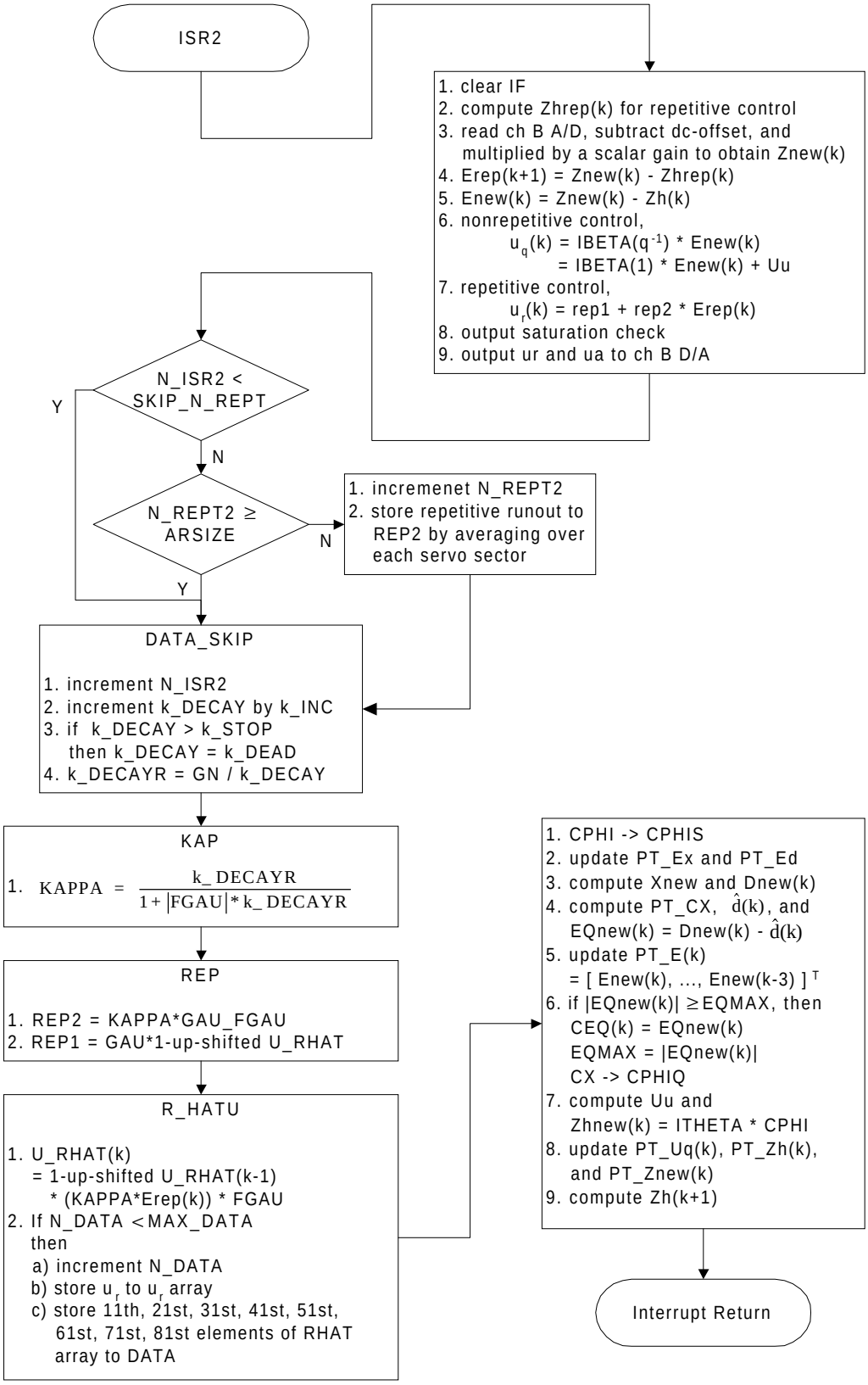


Figure 20. The Flow Diagram of ISR2

Figure 19 shows that once the interrupt INT0 is enabled, the ISR can interrupt the program flow whenever an external trigger occurs. This enables the multirating update of the plant parameter vector θ and the adaptive non-repetitive FIR filter Q, which usually requires longer computation time. When the subroutines LSM_S and LSM_Q update θ and Q, the ISR can interrupt the program once an external trigger occurs in order to keep up the I/O update of the DSP with the disk drive. To ensure the correct values of θ and Q are used by the ISR, up-sampled θ and Q are used. Unfortunately, the repetitive algorithm cannot use the multirating updating scheme and must update all the related variables and arrays, such as RHAT, REP1, and REP2, in one sampling period. Therefore, INT0 should be disable when updating these variables and arrays.

There are three assembly programs, ardemo, rdemo, and ademo. All these programs are similar; they all have two phases, identification phase and control phase. The difference is that different program turns on different compensators. The first program implements both the adaptive repetitive and non-repetitive compensators. The second one implements only the repetitive compensator, and the last one only turns on the non-repetitive compensator. All these programs call three library files, 'mfilt.lib', 'mmisc.lib', mmcal2.lib'. The 'mfilt.lib' library file has five FIR filters, including 'mfilt5', which filters the kernel function by $\hat{B}$. The 'mmisc.lib' library file has several functions, including prbs2, invf, mdup, and exp. The function 'prbs2' generates the pseudo random binary sequence used as the control input in the identification phase. The function 'invf' inverts the register content in R1 and output the result back to R1. No argument is required when calling 'invf'. The function 'mdup' copies all the elements of array 1 to array 2 using the circular addressing feature. The function is called as, 'mdup pointer_array1 pointer_array2 array_size array_size-2'. The function 'exp' calculates the exponential function of the first argument and output the result to the second argument. Only ten terms of the power series is used in the calculation of 'exp'. The last library file 'mmcal2.lib' computes matrix multiplication. 'mv' multiplies a matrix with a column vector. 'vm' multiplies a row vector with matrix, and 'vv' computes the dot product of two vectors.

6.0 Experimental Results

The spectrum magnitude of the PES by applying the adaptive repetitive algorithm is shown in figure 21. Table 2 summarizes the parameters used in the assembly program. It shows a 10-dB attenuation on the fundamental frequency (68.39 Hz) and a 5-dB attenuation on the second and third harmonics (138.8 Hz and 208.2 Hz, respectively). Figure 22 shows the convergence of the 11th, 21st, 31st, 41st, 51st, 61st, 71st, and 81st terms of $\hat{R}(k)$ for the first 1100 samples, and figure 23 shows the convergent $\hat{R}(k)$. The control output u_r for the first 1100 samples is also shown in figure 24.

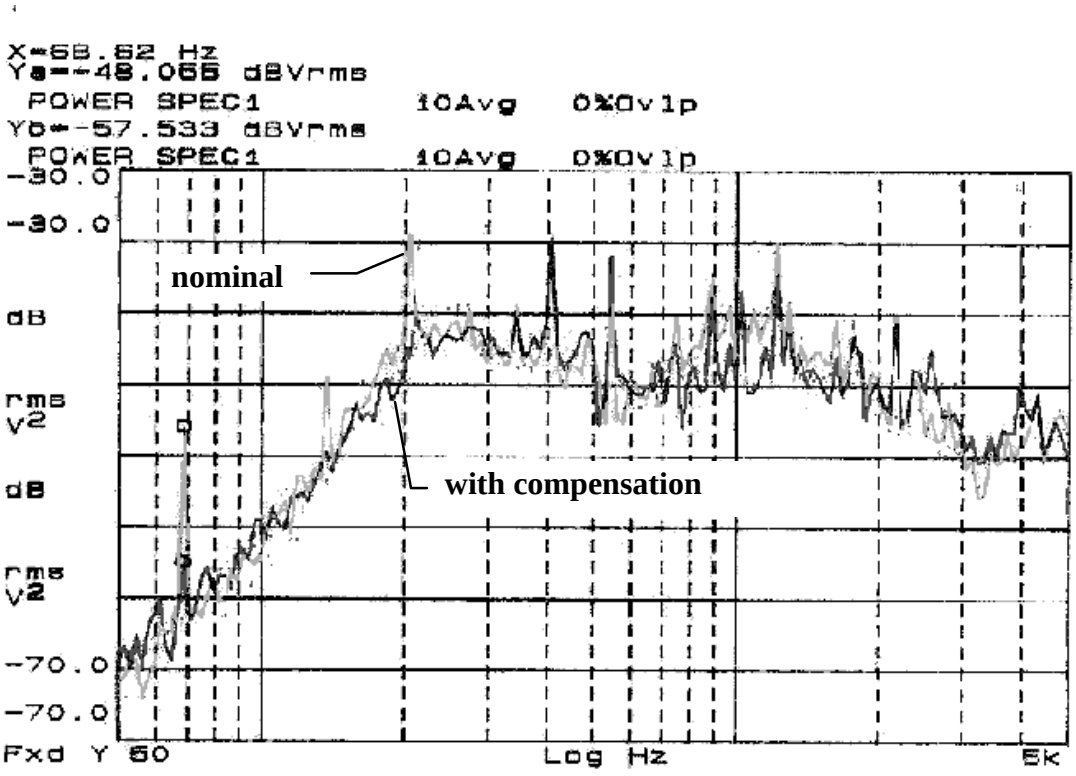


Figure 21. Experimental Spectrum Magnitude of PES with and without repetitive compensation.

Table 2. Repetitive Algorithm Parameters

Gn	550	k_DEAD	1000000
k_INC	0.3	k_STOP	100000
N	16	σ	10

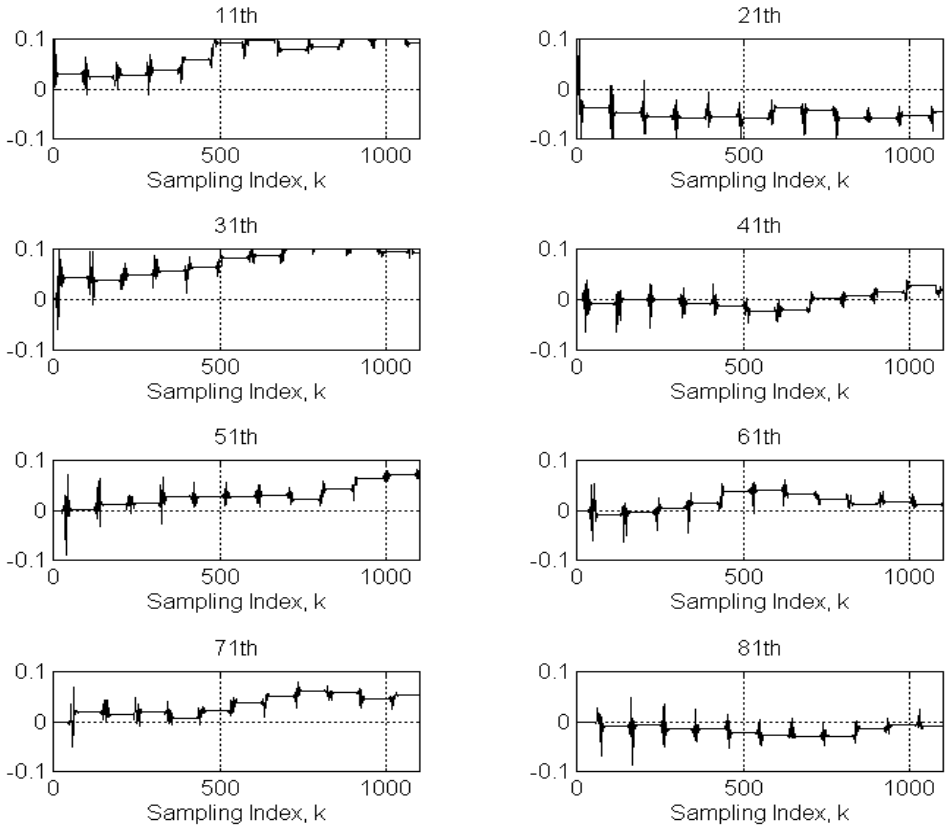


Figure 22. Convergence of the estimation $\hat{R}(k)$ in the first 1100 samples using the Kernel of N =16

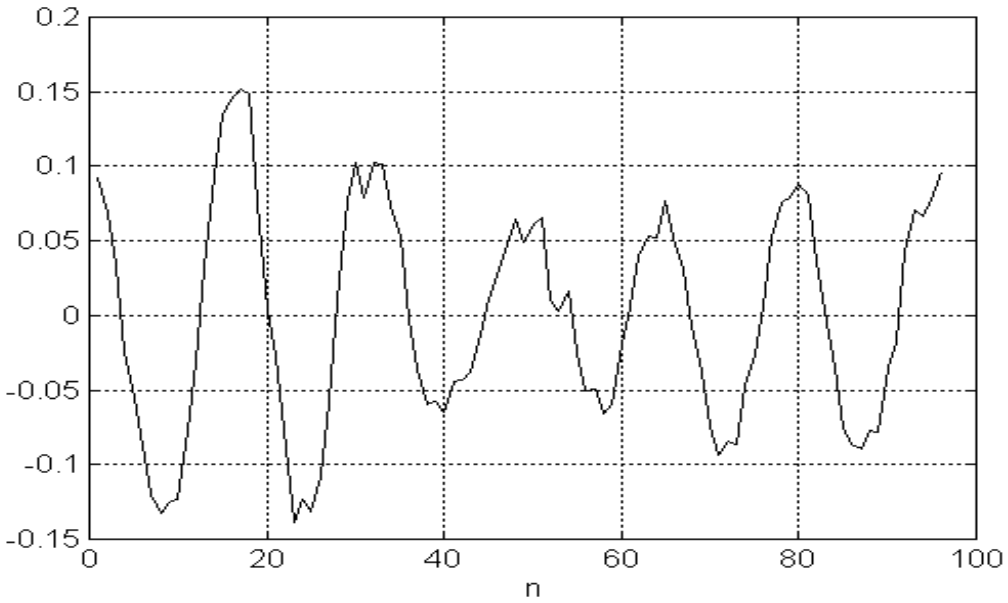


Figure 23. Convergent $\hat{R}(k)$ using the Kernel function of N =16

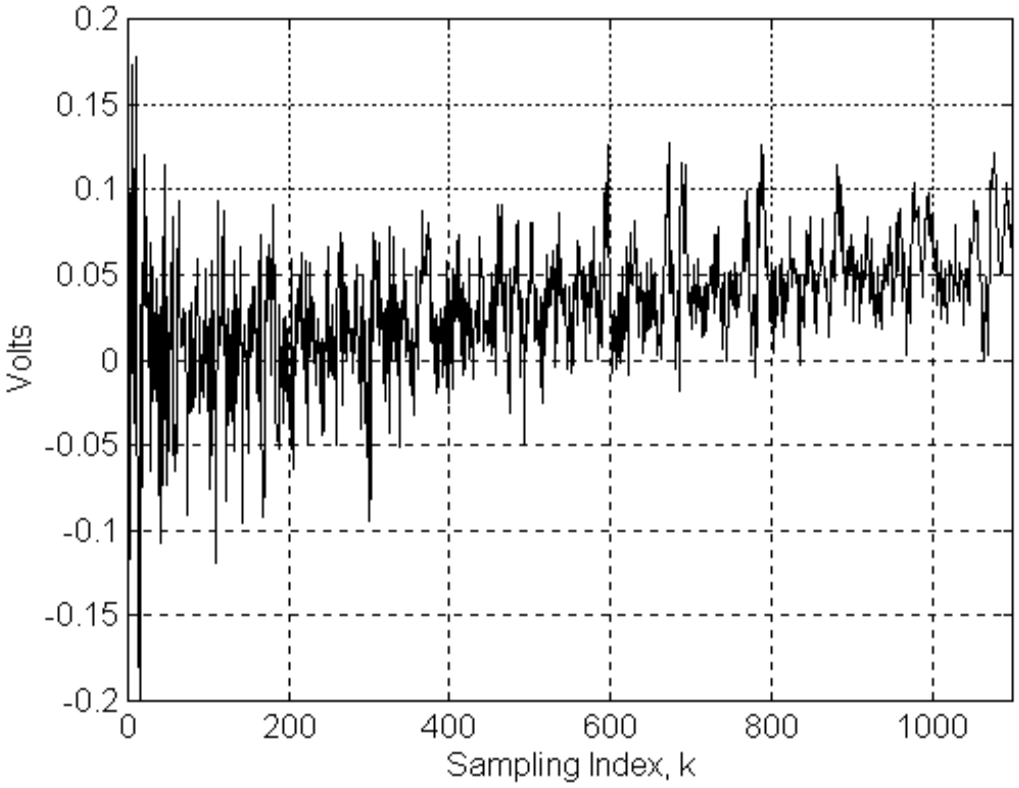


Figure 24. Control Output u_r in the First 1100 Samples

7.0 **Conclusion**

The integral equation approach is successfully formulated and tested on the JTS 3.25 drive. By using a kernel function of N = 16 as shown in figure 12, experimental results confirm that the repeatable runout of the first three harmonic frequencies is compensated, and the power magnitude of the nominal and compensated PES is shown in figure 21. The estimated repeatable disturbance parameters $\hat{R}(k)$ is also shown in figures 22 and 23. These are consistent with the simulation results, which indicates the compensation of the first third harmonic frequencies.

References

[1] M. Tomizuka, K. Chew, and W. Yang. Disturbance rejection through an external model. *ASME Journal of Dynamic Systems, Measurement and Control,* 112(4):559-564, December 1990.

[2] W. Messner and C Kempf. A comparison of four discrete-time repetitive control algorithms. December 1991

[3] R. Horowitz, S. Pannu, and B. Li. Adaptive track following servos for disk files. 1996

[4] R. Horowitz and B. Li. Design and implementation of adpative non-repetitive track-following disk file servos. 1996

[5] W. Messner, R. Horowitz, W. W. Kao, and M. Boals. A new adaptive learning rule. *IEEE Transaction on Automatic Control*, 36(2):188-197, February 1991.

[6] K. Chew and M. Tomizuka. Digital control of repetitive errors in disk-drive systems. *IEEE Control Systems Magazine*, 10(1):16-20, January 1990.