DESIGN TOOLS FOR PULSE-FREQUENCY-MODULATED CONTROL SYSTEMS: ERROR ANALYSIS AND LIMIT-CYCLE PREDICTION

by

Jake J. Abbott

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

The University of Utah

December 2001

ABSTRACT

The human nervous system uses pulse frequency modulation (PFM) to transmit information. In PFM, a continuous signal is converted into a pulse stream with a frequency that is proportional to the magnitude of the continuous signal. In an effort to create electromechanical prostheses that better approximate human behavior, an experimental neuroprosthetic arm has been designed to use the PFM signals obtained directly from nerves for control. Control system design and analysis tools are needed for systems containing PFM signals, which are poorly understood from a controls-engineering perspective. This research gives qualitative and quantitative insight into the behavior of PFM control systems.

This thesis is divided into four parts. First, three methods of pulse frequency modulation that have previously been proposed are compared and found to be equivalent for the control of a neuroprosthesis. Second, three methods of pulse frequency demodulation (PFD) are considered, and the errors encountered with each method are compared for frequencies relevant to the control of a neuroprosthesis. Unlike the PFM methods considered, the PFD methods are not equivalent, and some methods are obviously better choices than others. Third, a graphical limit-cycle prediction method is developed for PFM control systems. This method uses a tabular frequency-dependent describing function, and is shown to be accurate for many systems. Finally, the wrist of the Experimental Neural Arm is modeled, and the limit cycles seen in experiments with

the wrist are compared to those predicted by the graphical limit-cycle predictor. The predictor works well with the actual neuroprosthesis.

CONTENTS

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank the Center for Engineering Design for generously funding this research. I would like to thank Dr. Sanford Meek for giving me guidance when I needed it, but freedom to choose the direction from which to approach this problem. I would like to thank Mark Colton for help with a lot of little things that can quickly add up. Finally, I would like to thank my wife Katie for her love and patience.

1. INTRODUCTION

The Utah Arm 2 is an electromechanical prosthetic arm that is currently controlled using electromyographic (EMG) signals measured from the surface of the skin. These EMG signals arise from the electrical activity in the muscles below the skin. The Utah Arm 2 has been modified to use electrical signals obtained directly from nerves, using sensors developed by Dr. Ken Horch's lab in the Department of Bioengineering at the University of Utah. This modified arm will be referred to as the Experimental Neural Arm. Control of the Experimental Neural Arm with nerve signals would be more natural than control using EMG signals because the nerves used for control would be the same nerves that would control a real arm, and this would theoretically make the performance of the artificial arm approach that of a real arm.

The human body is not fully understood, and interfacing electromechanical prostheses with it requires techniques that are not commonly used in control systems engineering. The human nervous system uses what can be approximated as pulse frequency modulation (PFM) to transmit information through nerves. A PFM signal is a sequence of pulses of nearly uniform amplitude and very short duration whose frequency carries the signal's data. When pulse frequency modulating a continuous signal, information about the original signal is necessarily lost due to the discretized nature of the PFM signal; nothing is known about any changes in the continuous signal until the occurrence of a new pulse.

Tools are needed to help analyze and design control systems containing PFM signals, specifically the Experimental Neural Arm. It is desirable to understand if the system is stable or not. It is also desirable to understand the transient and steady-state behavior of the system. Ideally there would be tools, like those available in classical and nonlinear controls, that would assist in the analysis and design of systems containing PFM signals.

PFM signals occur in the human nervous system because of the creation and propagation of action potentials [1]. However, PFM signals are rarely used in engineering applications because they are very inefficient; much of the information contained in a continuous signal is lost during the modulation process. Pulse frequency modulators are also highly nonlinear, and are therefore not mathematically well defined or understood. PFM signals are very insensitive to noise, but this seems to be their only positive attribute.

A model of pulse frequency modulation in the human nervous system is needed for two purposes. First, because experimental time with real amputees is very rare, a model of an amputee is needed to help design new Neural Arms and arm controllers. Second, to feed back information into the nervous system through afferent nerves, it is necessary to send information in a form that the brain understands.

Several methods of pulse frequency modulating a signal have been proposed over the years [2-7]. When creating a model that includes the pulse frequency modulation of the nervous system, it is not obvious which is the best method to choose. In Chapter 2, four different pulse frequency modulation methods that have been proposed are compared to one another; these methods are Integral PFM, Neural PFM (these two

methods fall under a larger PFM class known as ΣPFM [2]), voltage-to-frequency conversion [3-5], and Unified States Sample & Hold [6]. All of the methods, with the exception of Neural PFM, are only subtly different from one another, and can be considered equivalent for the control of a neuroprothesis. The problems encountered when pulse frequency modulating a signal using a digital computer are also discussed in Chapter 2.

Using PFM signals from thousands of nerves to control a motor-driven artificial arm is impractical, due to the difficulty and invasiveness of implanting sensors in nerve endings. For this reason, it is necessary to demodulate as few as one PFM signal for control of the arm. Demodulating a PFM signal to recreate the original signal, which is assumed to be continuous, poses interesting problems. The demodulation of a PFM signal can be accomplished in many ways, each of which has problems from a control system design perspective.

Most PFM methods are equivalent to one another, but the various methods of pulse frequency demodulating (PFD) are distinct, each having advantages and disadvantages. In Chapter 3, five different pulse frequency demodulation methods are compared to one another; these methods are period measurement, first-order low-pass filtering, second-order low-pass filtering, finite-impulse-response filtering, and counting pulses in a fixed-time window. The advantages and disadvantages of each method are discussed. The errors encountered using each method are quantified and compared; this includes the errors encountered when using a digital computer to demodulate a PFM signal.

Because pulse frequency modulators and demodulators are not time-invariant, traditional describing function techniques cannot be applied to systems containing PFM. In Chapter 4, a method is developed to graphically predict the existence of limit cycles in systems containing pulse frequency modulation and demodulation; the method also predicts the amplitude and frequency of the limit cycle, if it exists. This method is based on a tabular describing function, and it works well when compared to Simulink simulations.

The graphical limit-cycle predictor of Chapter 4 works well when compared to simulations, but it is desirable to prove the validity of the method with a real system. In Chapter 5, a model of the Experimental Neural Arm wrist is created. The interface of the Experimental Neural Arm wrist to an amputee is simulated using two computers communicating to each other using only PFM signals; one computer acts as a wrist controller, while the other computer simulates an amputee. The wrist model is then used to validate the graphical limit-cycle predictor of Chapter 4 by comparing predicted limit cycles to actual limit cycles seen in the wrist. The predictions match the limit cycles seen in the wrist well.

In Chapter 6, some possible future-work topics that could use the results of this research are presented.

2. COMPARISON OF PULSE FREQUENCY MODULATION METHODS

Engineers have been proposing methods of pulse frequency modulation (PFM) for nearly forty years [2-7], and in many cases the purpose was to model the human nervous system. When modeling a system that contains PFM elements, it is not obvious which PFM method is best to use. Every PFM method is very nonlinear and complicated to analyze in anything but the most basic scenarios. Every PFM method uses integration in some form, which leads to a low-pass filtering behavior of all PFM methods. The purpose of this chapter is not to redo the work that has been previously done on PFM, but rather to compare the various available methods, and to show that for all practical purposes many methods are equivalent to each other and can be used when modeling PFM with no loss of generality. The three methods of PFM considered here are Sigma Pulse Frequency Modulation [2], voltage-to-frequency conversion [3-5], and Unified States Sample and Hold [6].

The first method of PFM considered is Sigma Pulse Frequency Modulation (ΣPFM) [2]. The most widely investigated method of PFM is integral pulse frequency modulation (IPFM), which is a subclass of ΣPFM. IPFM is mathematically straightforward and easy to understand. Another class of ΣPFM is neural pulse frequency modulation, which may better represent the way the nervous system works than IPFM.

A voltage-to-frequency (V/F) converter [3-5] is a common electrical circuit that basically behaves like IPFM. Because it is a physical circuit, a V/F converter does not

behave ideally like the mathematical expressions of other PFM methods, but it can actually be implemented in an analog circuit.

The Unified Steps Sample and Hold method of PFM [6] uses a highly nonlinear, but continuous, mathematical function to replace the discontinuities in IPFM, allowing easier closed-form analysis of PFM systems. Simulink simulations of all three PFM methods are found in Appendix A.

2.1 Sigma Pulse Frequency Modulation

The PFM method known as ΣPFM was first proposed by Pavlidis and Jury [1]. ΣPFM is a very general pulse frequency modulator, encompassing IPFM and NPFM. The equations for ΣPFM are:

$$\frac{dp}{dt} = x - r\,\mathrm{sgn}(p)\delta(|p| - r) - g(p) \tag{1}$$

$$y = \mathrm{sgn}(p)\delta(|p| - r) \tag{2}$$

$x \equiv$ Modulator Input

$y \equiv$ Output Pulse Stream

$p \equiv$ Integral of $x - g(p)$

$g(p) \equiv$ Any Function of p

$r \equiv$ Threshold Value of Integral

$\mathrm{sgn}() \equiv$ signum function

$\delta \equiv$ Unit Impulse

$r\mathrm{sgn}(p)\delta(|p|\text{-}r) \equiv$ Integrator Reset

A unit-area ideal impulse occurs when the magnitude of p reaches the threshold r. At the occurrence of a pulse, the integral p is reset to zero. The sign of the output pulse is the same as the sign of p when $|p|$ reaches r, allowing for negative pulses. This is known as double-signed PFM.

If the domain of x is known, it is possible to bias x such that p is never decreasing. This results in only positive pulses, and is known as single-signed PFM.

2.1.1 Integral Pulse Frequency Modulation

IPFM is the most widely investigated PFM method, probably because it is the simplest. IPFM is a special cased of ΣPFM where $g(p) = 0$ in Eq. (1). In this case, p is simply the integral of the input x. When this integral reaches a threshold r, a pulse is emitted at the output, and the integral is reset to zero.

The output pulses to a step input of x_0 will have a pulse frequency f in Hertz and a period between pulses T in seconds given by:

$$f = \frac{x_0}{r} \tag{3}$$

$$T = \frac{r}{x_0} \tag{4}$$

If the integral p is not zero at the occurrence of the step input, the initial pulse period will differ from T, but the pulse period will be equal to T for all time thereafter.

Figure 1 shows an IPFM pulse output to a 1-Hz unit-amplitude squarewave input. The pulses have been reduced to a unit height for graphical purposes, but true IPFM actually outputs ideal unit impulses (infinite height, zero width). For this figure, $x_0 = +/-1$, $r = 1$. Equation (3) and Eq. (4) give $f = 10$ Hz and $T = 0.1$ seconds, respectively. Also,

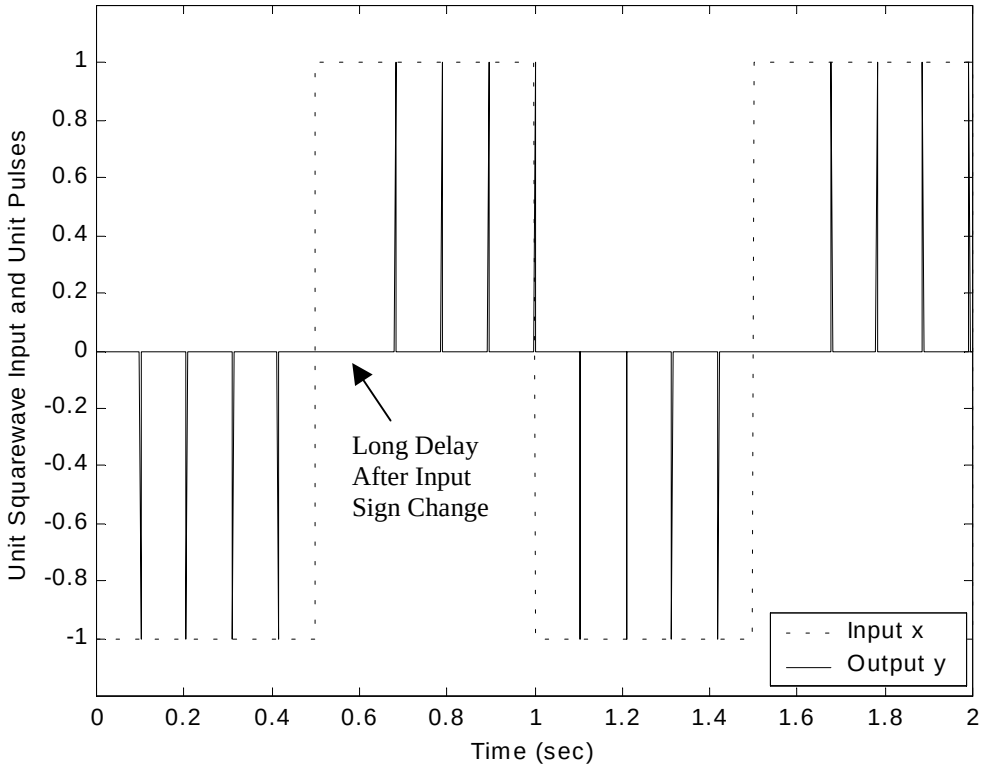


Fig. 1 1-Hz Squarewave Input and Output Pulses for IPFM with Threshold r = 0.1

at start-up, $p = 0$. Because $p(0) = 0$, a time delay of T seconds exists before the first pulse is emitted. The input transition from –1 to 1 shows a delay longer than T seconds between the input transition and the first positive pulse. This is due to the negative value of p at the time of the input transition. In general, the first pulse occurring after an input step from a negative to a positive value (or vice versa) has a delay between T and $2T$ seconds.

Figure 2 shows an IPFM pulse output to a 1-Hz 0.5-amplitude squarewave input that has been biased by 1.5. The pulses have again been reduced to a unit height for graphical purposes. This is an example of single-signed IPFM, because the integral p is never decreasing, and no negative pulse is ever emitted. Using Eq. (3) and Eq. (4) gives f

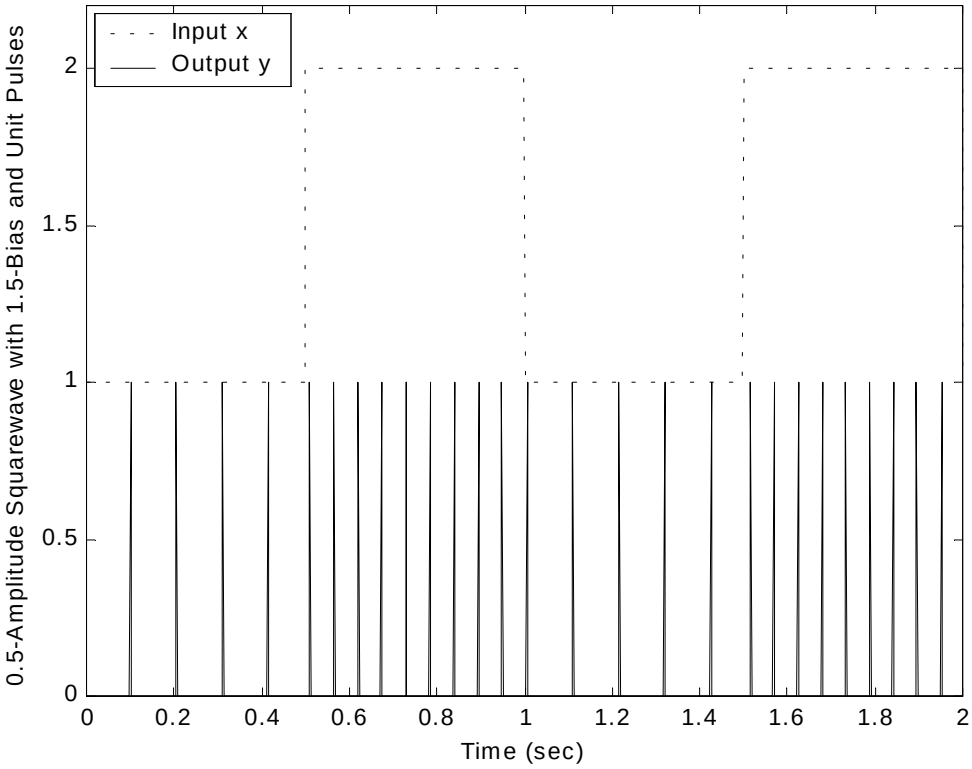


Fig. 2 1-Hz Squarewave Input with Bias and Output Pulses for IPFM with Threshold r = 0.1

= 10 Hz and $T = 0.1$ seconds when $x_0 = 1$, and $f = 20$ Hz and $T = 0.05$ seconds when $x_0 = 2$. In this single-signed scheme, the time delay between an input change and the next pulse is less than or equal to the new value of T. This gives one pulse period that is at a transitional value somewhere between the previous and subsequent values of T.

2.1.2 Neural Pulse Frequency Modulation

Neural PFM (NPFM), also known as relaxation PFM, is a special cased of ΣPFM [2] where $g(p) = cp$ in Eq. (1), and c is a constant. If Eq. (1) is analyzed just after the emission of a pulse, it becomes:

$$\frac{dp}{dt} = x - cp \tag{5}$$

which can be written in Laplace domain as:

$$p(s) = \frac{1}{s+c} x(s) \tag{6}$$

Neural PFM acts as a first-order low-pass filter with a time constant and a DC gain of 1/*c* between the modulator input and the output *p*. The input needs to be at least *c* times larger than the threshold *r* for the modulator to ever emit a pulse. The fundamental reason for using NPFM, rather than IPFM, is that for small inputs no pulses are emitted. This lends to steady-state errors, but eliminates sustained oscillations in a closed-loop system, which may be a desirable trade-off.

The output pulses to a step input of x_0 have a pulse frequency *f* in Hertz and a period between pulses *T* in seconds given by:

$$f = \frac{c}{\ln\left(\frac{x_0}{x_0 - cr}\right)} \tag{7}$$

$$T = \frac{1}{c} \ln\left(\frac{x_0}{x_0 - cr}\right) \tag{8}$$

These equations are obviously more nonlinear than those of IPFM.

Figure 3 shows Eq. (7) for various values of *c*, again for constant inputs. The value of *c* is determined by looking at the value of *x/r* when the frequency breaks away from zero. For example, the frequency becomes nonzero for the plot where *c* = 2 at the point (2,0). The pulse frequency becomes more nonlinearly related to the input when the

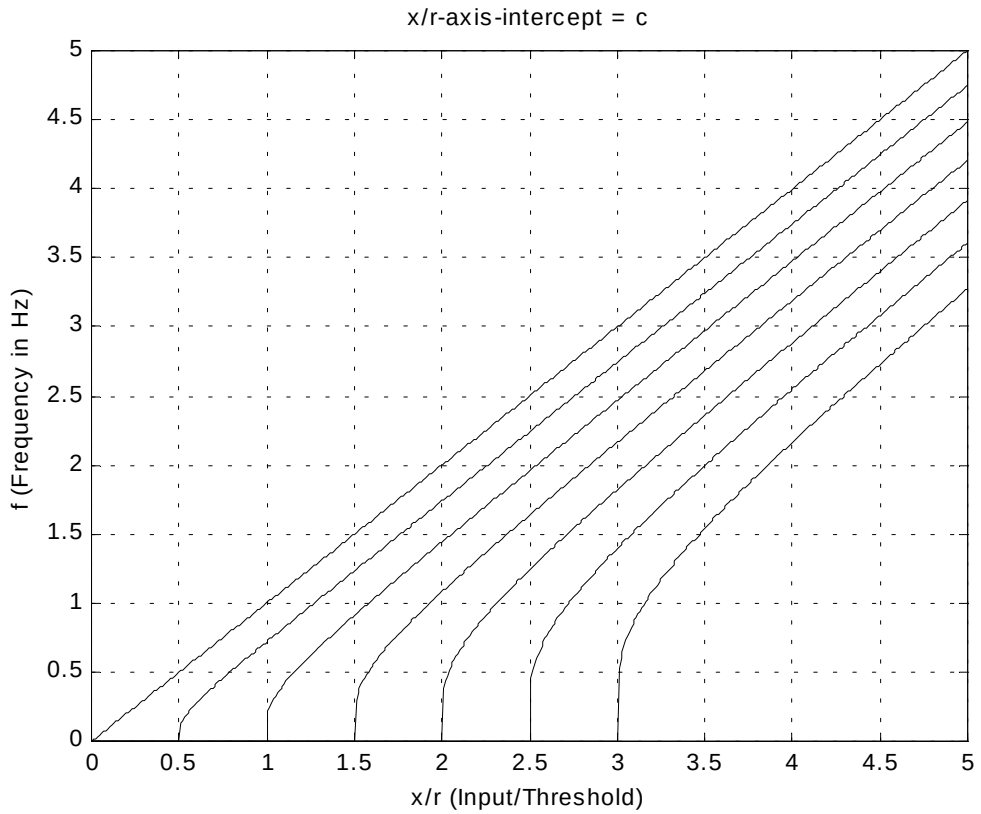


Fig. 3 NPFM Frequency vs. *x*/*r*

input value is near the threshold value, and as *c* increases. Note that IPFM is achieved when $c = 0$.

2.2 Voltage-to-Frequency Converter

A voltage-to-frequency (V/F) converter [3-5] is a practical circuit used to implement PFM. This circuit is often referred to as a voltage-controlled oscillator (VCO), but when used as a pulse frequency modulator, the actual description of the circuit used is a V/F converter, not a VCO [3-4]. There is only a subtle difference between the two circuits; the output of a VCO can be any waveform (squarewave, sinusoid, etc.) with a frequency proportional to the input voltage, while a V/F converter specifically outputs pulses with a frequency proportional to the input voltage [3].

VCOs and V/F converters are circuits that many engineers are familiar with – even those engineers with no experience with PFM. While a pulse frequency modulator would probably never be modeled as a V/F converter in a simulation, it is important to show that a V/F converter is equivalent to other PFM methods, if for nothing else, than to give engineers something they are familiar with when considering PFM methods.

A V/F converter is not as simple as ΣPFM, and the mathematical equations describing it are not as elegant, but the circuit can actually be implemented. There are no infinitely-high, zero-width pulses required in a V/F converter; there are no operations that must take place infinitely fast either.

There is no universally-recognized definition of a V/F converter, but most examples have similar components. Some form of op-amp integrator is used. Also, there is some method of switching between the input voltage being modulated and some negative voltage used to reset the integrator at the occurrence of a pulse.

Figure 4 shows a V/F converter circuit that is an adaptation of Fig. 1.36 in Nack [5]. The V/F converter in Fig. 4 uses elements found in [4] and in [5]. It behaves very similar to that of [5], but has a more-linear relationship between the input voltage and the output pulse frequency.

The V/F converter of Fig. 4 contains an op-amp integrator, an op-amp comparator, a timer, and a digital buffer. The input voltage being modulated is labeled V_i, the negative voltage used to reset the integrator is labeled $-V_r$, and the integrator output voltage is labeled V_o. The comparator's noninverting input is grounded, and its output is low when the inverting-input voltage is higher than ground. A constant positive V_i causes V_o to have a constant negative slope, decreasing until $V_o = 0$, at which point the

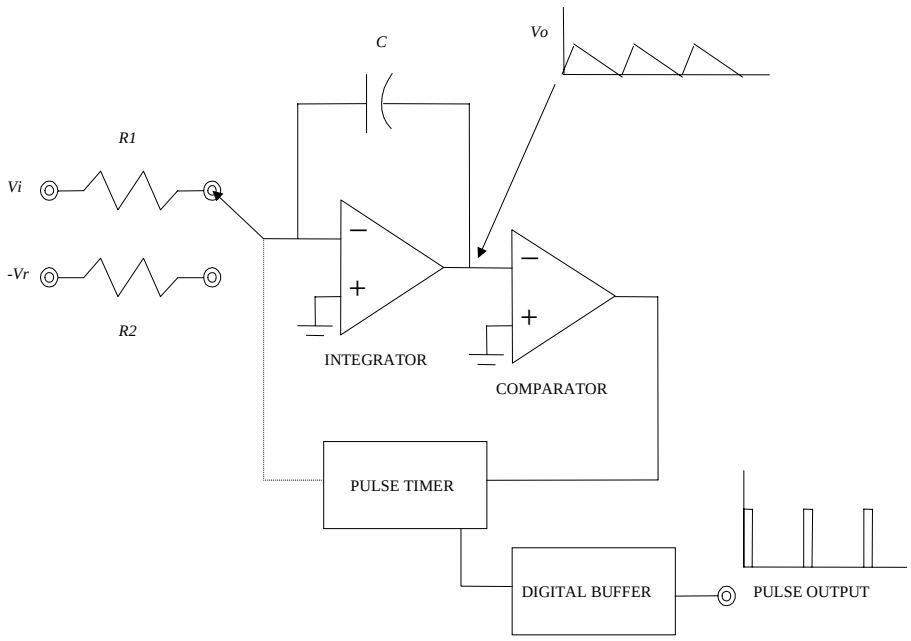


Fig. 4 V/F Converter

output of the comparator goes high. This triggers the pulse timer, which is a standard timer circuit with an output that stays high for τ seconds. As long as the timer output is high, the input is switched to the negative reset voltage, and the V/F converter's output voltage goes high for τ seconds, which is the duration of an output pulse.

Figure 4 includes a characteristic plot of V_o as a function of time. Assuming the various voltages in the system are grounded before system start-up, a pulse is emitted at start-up. This pulse emitted at start-up is really the only practical difference between the V/F converter and IPFM. For a constant input, the output pulse frequency in Hertz is given by:

$$f = \frac{R_2}{R_1 \tau V_r} V_i \tag{9}$$

This frequency equation assumes that the pulse duration τ is negligible compared with the pulse period; this is a false assumption if the parameters of Eq. (9) are not chosen carefully.

It may be noted that the capacitor value has no effect on the pulse frequency, but it does affect the maximum voltage that V_o reaches during its charging and discharging cycle. This is important during practical implementation of the circuit. Also note that this circuit only works for single-signed operation, where V_i is never negative.

2.3 Unified States Sample and Hold

The Unified States Sample and Hold (USSH) method was first proposed by Frank and Turski [6]. The USSH method is an integral scheme that uses a so-called serraphile function, which is a continuous function that approximates a saw-tooth function. The serraphile function is defined as:

$$ser(\alpha) = \lim_{\rho \to 1^-} \frac{2}{\pi} \tan^{-1}\left(\frac{\rho \sin(\alpha)}{1 + \rho \cos(\alpha)} \right) \tag{10}$$

Figure 5 shows how the serraphile function approaches a saw-tooth function as ρ approaches 1.

For the first step of the USSH method, the signal to be modulated, *e*, is integrated and multiplied by a gain *b*. In Laplace domain:

$$p(s) = \frac{b}{s} e(s) \tag{11}$$

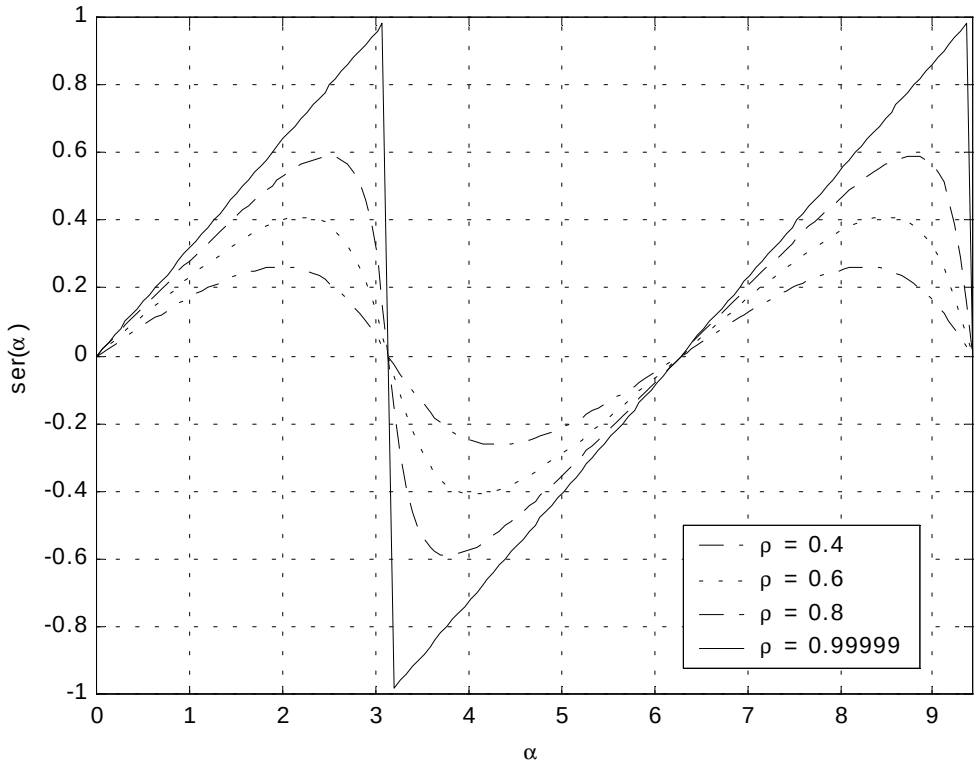


Fig. 5 Serraphile Function

Next, the integrated input is run through Frank and Turski's USSH, utilizing the serraphile function:

$$q = p - \frac{1}{2} ser(2\pi p) \tag{12}$$

The value of q is a constant throughout the linear regions of the serraphile function. At the quickly changing region of the serraphile function (the saw-tooth), the value of q quickly changes to a new value, and is then constant for the next gently sloping region of the serraphile function. The final step in the USSH method involves differentiating q. In Laplace domain:

$$u(s) = sq(s)G(s) \tag{13}$$

Here *u* is the output pulse stream, and *G(s)* is the Laplace form of the shape of the desired pulse *g(t)*. For the purposes of this thesis, *G(s)* = 1 which gives a pure impulse as the desired pulse shape.

The output pulses to a step input of e_0 have a pulse frequency *f* in Hertz and a period between pulses *T* in seconds given by:

$$f = be_0 \tag{14}$$

$$T = \frac{1}{be_0} \tag{15}$$

These frequency and period values are valid for a constant input, but the shape of the serraphile function makes the first pulse come with a delay of only half of the steady-state period:

$$T_0 = \frac{T}{2} \tag{16}$$

Figure 6 shows a USSH pulse output to a 1-Hz unit-amplitude squarewave input. The pulses have been reduced to a unit height for graphical purposes, but true USSH actually outputs pulses with height and width that are a function of ρ (Eq. (10)) and $G(s)$ (Eq. (13)). For this figure, e = +/-1, b = 10. Equations (14) and (15) give f = 10 Hz and T = 0.1 seconds, respectively. Also, at start-up, p = 0. Because $p(0) = 0$, a time delay of T_0 = 0.05 seconds exists before the first pulse is emitted.

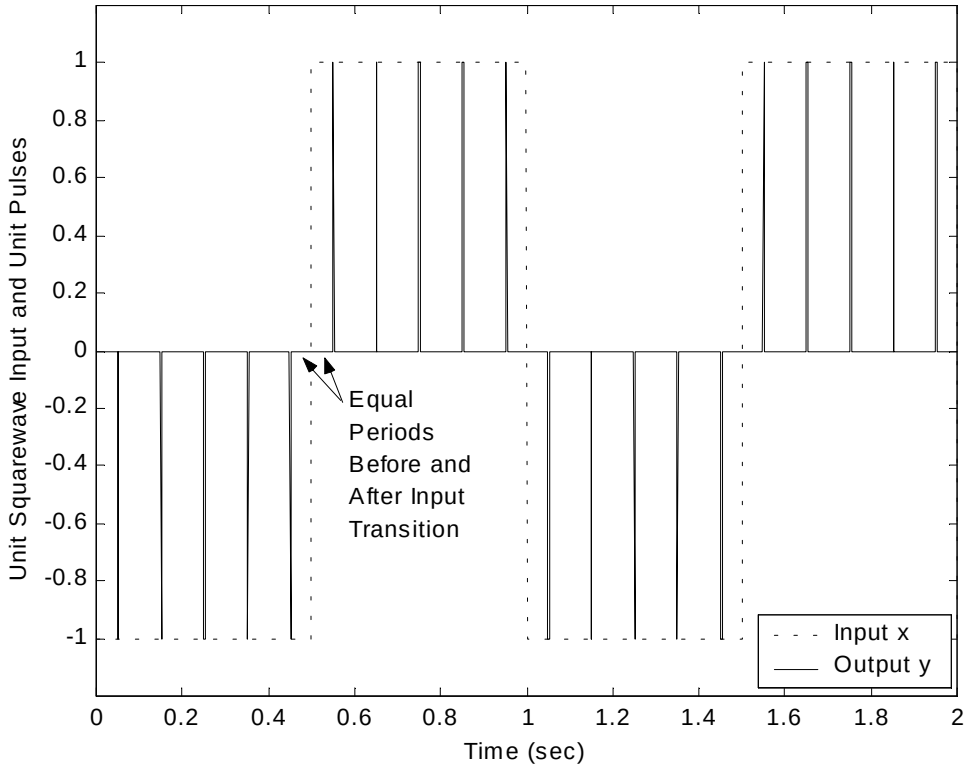


Fig. 6 1-Hz Squarewave Input and Output Pulses for USSH with Gain b = 10

The input transition from –1 to 1 shows an identical period between the last negative pulse and the input transition, and between the input transition and the first positive pulse. This behavior is due to the nature of the serraphile function. If p is increasing, the serraphile function in Fig. 5 is analyzed from left to right. If p is decreasing, Fig. 5 is analyzed from right to left. With a squarewave input, whatever time has elapsed since passing a serraphile "tooth" in one direction will be exactly matched when backtracking in the serraphile function. In Fig. 6 it appears that the delay before and after an input transition may be the same as T_0, but this is coincidental. In general, the first pulse occurring after an input step from a negative to a positive value (or vice versa) has a delay less than T seconds.

Figure 7 shows a USSH pulse output to a 1-Hz 0.5-amplitude squarewave input that has been biased by 1.5. The pulses have again been reduced to a unit height for graphical purposes. This is an example of single-signed USSH, because the integral p is never decreasing, and no negative pulse is ever emitted. Using Eqs. (14) and (15) gives f = 10 Hz and T = 0.1 seconds when e_0 = 1, and f = 20 Hz and T = 0.05 seconds when e_0 = 2. In this single-signed scheme, the USSH behaves like single-signed IPFM after the initial period T_0.

One detail about the USSH method that should be noted is that, if the input has a DC value, the integral p will grow to infinity. Practically this could lead to overflow problems. The flexibility of software would probably make this problem solvable, but no

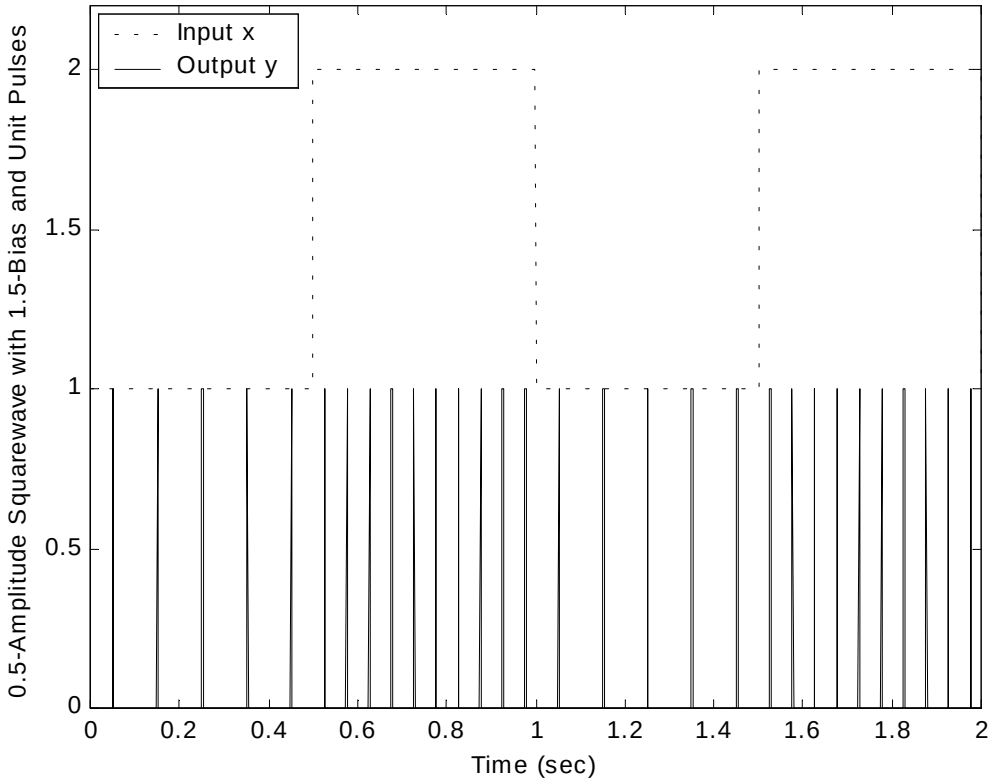


Fig. 7 1-Hz Squarewave Input with Bias and Output Pulses for USSH with Gain b = 10

solution will be sought here.

2.4 Pulse Frequency Modulation Method Equivalency

Various methods of PFM have been introduced, each with its own strengths and weaknesses. ΣPFM is very mathematically defined. It is also relatively easy to implement on a digital computer, where resetting an integral is not difficult to do. A V/F converter is not as elegantly-defined mathematically as other PFM methods, but it is a common circuit with which engineers are familiar. The USSH method needs no form of integrator reset, but certain variables could grow to infinity if a USSH is implemented in a real-time controller. The greatest benefit of the USSH method is its complete lack of discontinuities.

Each PFM method has different behavior at start-up, and each PFM method will behave differently at very high frequencies. The various PFM methods, with the exception of NPFM, will behave equivalently in DC and low-frequency situations, after a brief discrepancy at start-up. Table 1 shows the system parameters of the three equivalent PFM methods. If a modulation constant k_M is defined as the gain between a constant modulator input signal and the resulting constant pulse frequency in Hertz, then for IPFM

$$k_M = \frac{1}{r} \tag{17}$$

for a V/F converter

$$k_M = \frac{R_2}{R_1 \tau V_r} \tag{18}$$

Table 1 Equivalent Parameters of Equivalent PFM Methods

	IPFM	V/F Converter	USSH
Input Variable	x	V_i	e
Frequency f	$\frac{x}{r}$	$\frac{R_2}{R_1 \tau V_r} V_i$	be
Pulse Height	∞	Logic HI	∞
Pulse Duration	0	τ	0
First Pulse Delay	T	0	$\frac{T}{2}$
Constants	Threshold r	Pulse-Timer Duration τ Reset Voltage V_r	Integrator Gain b

and for USSH

$$k_M = b \tag{19}$$

Practically, for modeling and control of the Experimental Neural Arm, any of these PFM methods will work, with the modulation constant being the only important factor.

NPFM is not equivalent to the other three PFM methods considered here; it is much more nonlinear. Pavlidis and Jury [2] claim that NPFM better approximates the way the nervous system works than does IPFM, but current information about the nervous system does not seem to suggest that NPFM models the nervous system any better than does IPFM. Current information suggests a logarithmic relationship between the modulator input and the pulse frequency [8], but NPFM does not have this behavior.

It does not seem at this time that using NPFM to model the nervous system gives any additional benefit over the three integral PFM schemes.

2.5 Effect of Discrete Sampling on Pulse Frequency Modulation

This section deals with pulse frequency modulation using a digital computer with a fixed, known sampling rate. As with any digital-to-analog signal conversion, the faster the computer's sampling rate is relative to the signal's frequency, the better the computer can accurately represent the signal. Only DC signals will be considered while trying to quantify errors due to digital modulation because any other signals become far too complex, but the results translate well to low frequency signals because the errors found here are instantaneous errors that are based on the instantaneous desired pulse frequency.

Let f and T be the desired output pulse frequency in Hertz and the desired output pulse period in seconds of the signal being modulated, let f_m and T_m be the actual modulator output pulse frequency and period, and let f_s and T_s be the computer's sampling frequency and period. Figure 8 shows how a desired pulse period is necessarily extended due to the discrete nature of the computer. Every output pulse occurs at a computer sample. For any instantaneous desired pulse period T, the first actual output pulse always occurs at a computer sample, and the computer then measures time forward from this point. Because of the causal nature of digital pulse frequency modulation, the second actual output pulse always occurs at the computer sample that follows when the desired output pulse should occur to give a period T.

The relationships characterized in Fig. 8 are

$$T + T_s > T_m > T \tag{20}$$

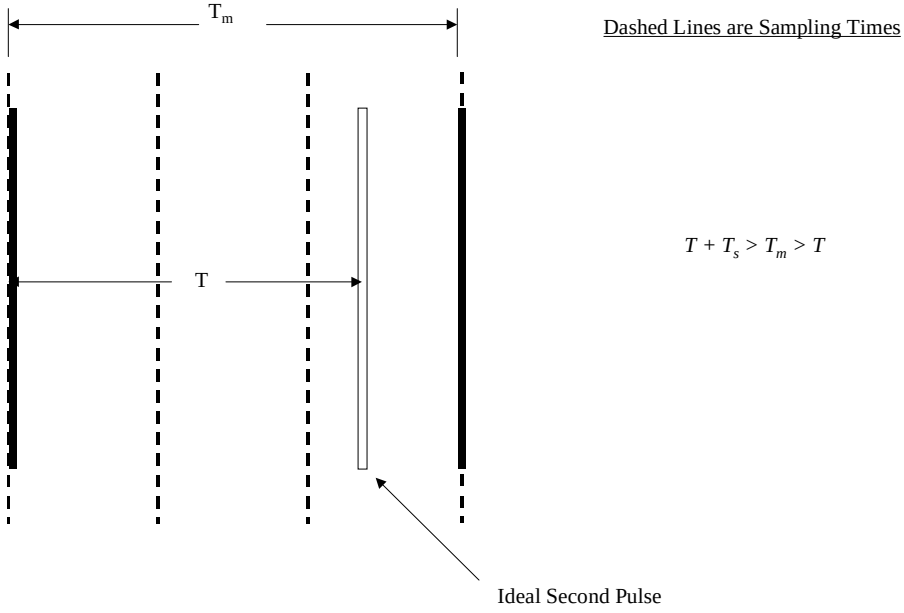


Fig. 8 Pulse Frequency Modulation with Discrete Samples

$$T_s n > T > T_s(n-1) \Rightarrow T_m = T_s n \tag{21}$$

where n is a positive integer. Dividing Eq. (21) by T_s and rearranging for frequency rather than period gives:

$$n > \frac{f_s}{f} > n-1 \Rightarrow \frac{f_s}{f_m} = n \tag{22}$$

Let an error in the modulator output pulse frequency be defined as:

$$\varepsilon_m = \frac{f - f_m}{f} \tag{23}$$

Notice that an underestimate in the modulator frequency will cause a positive error (the modulator frequency is always an underestimate). Figure 9 shows the error of Eq. (23) as

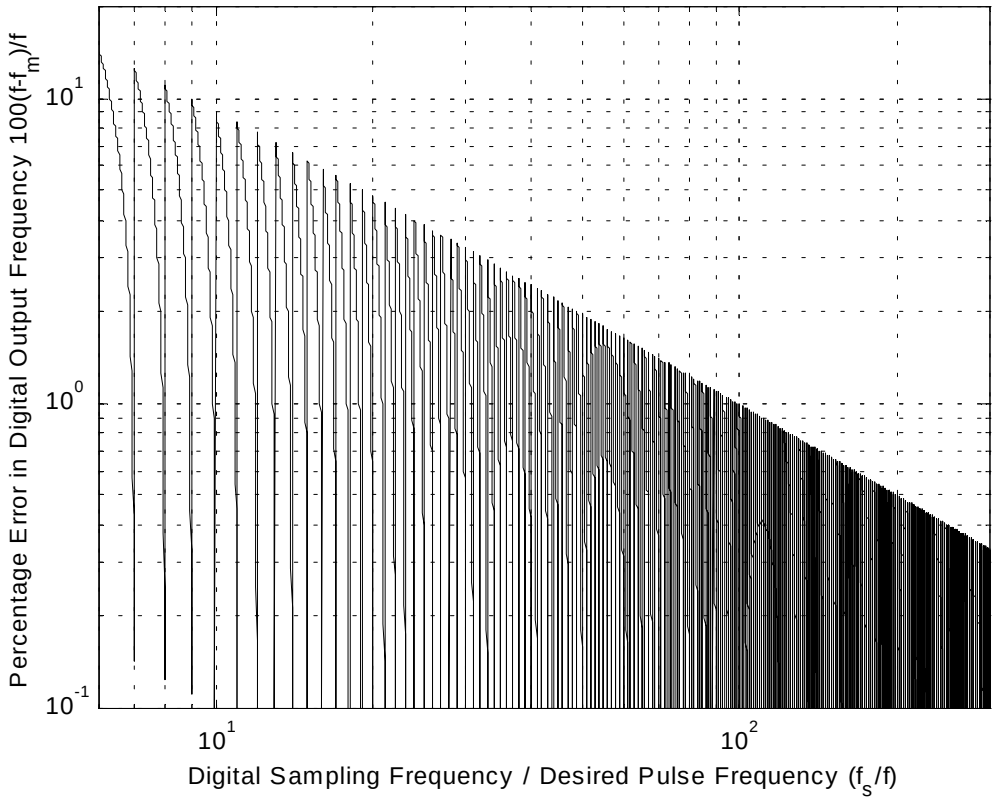


Fig. 9 Error in Digital Output Frequency vs. Normalized Desired Output Frequency

a percentage of desired frequency *f,* using the relation of Eq. (23). The actual error goes to zero at points, but they are not shown on a log-log plot; this is unimportant, because only the high errors are of any concern. The straight line made by the top of the plot should be used as an error envelope for a given ratio f_s/f.

Figure 10 shows how digital PFM will work on a computer with a 3000-Hz sampling rate. The output pulse frequency is always lower than the desired pulse frequency, and the error grows as the desired frequency increases. Figure 10 is characteristic of how the normalized errors of Fig. 9 will appear with any fixed sampling rate.

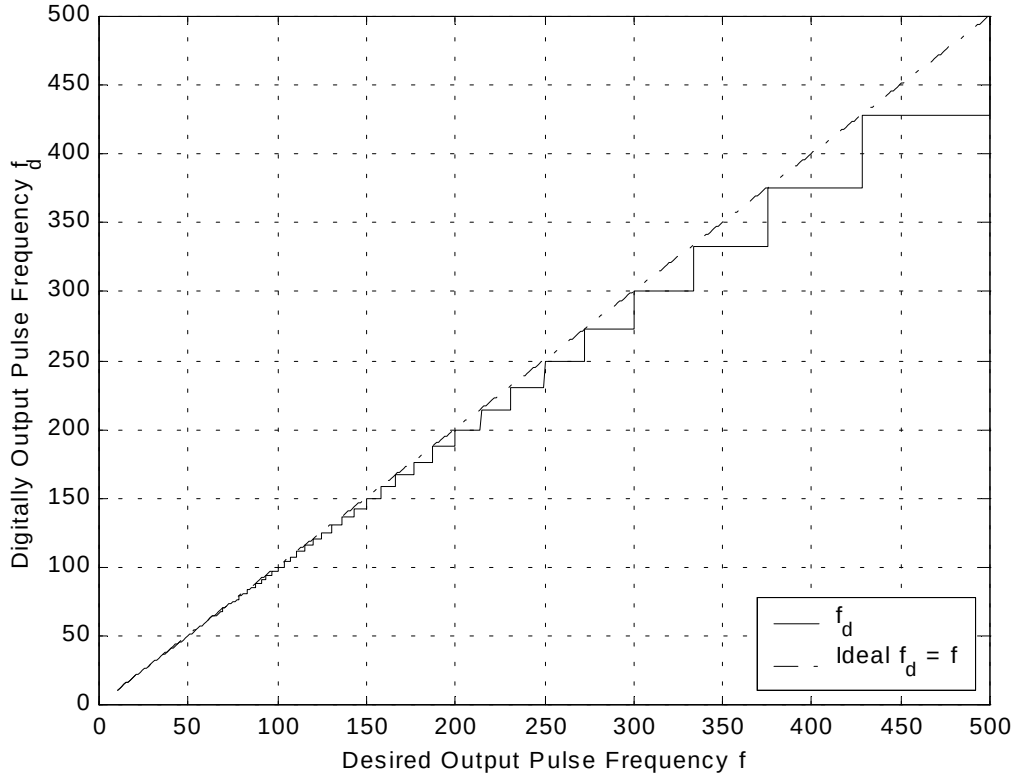


Fig. 10 Digital Output Frequency vs. Desired Output Frequency for f_s = 3000 Hz

2.6 Parallel-Path Single-Signed Pulse Frequency Modulation/Demodulation

Li and Jones [7] proposed the idea of parallel-path single-signed pulse frequency modulation/demodulation (PPSSPFMD) as a way to transmit a double-signed signal when only positive pulses may be transmitted, such as in the nervous system. PPSSPFMD is different from single-signed PFM, which biases the original double-signed input signal so that the input to the modulator is always positive. Single-signed PFM results in a stream of positive pulses, but knowledge of the bias must be known to demodulate the pulse stream. Also, as can be seen in Fig. 2 and Fig. 7, the pulse-frequency behavior is not symmetric for biased signals; the portions of the signal with

low magnitudes have a low frequency, and hence a long delay (see Chapter 3).

Figure 11 shows a PPSSPFMD setup. The nonlinearities that precede the pulse frequency modulators allow the positive portion of the input signal to pass through the upper pulse frequency modulator and demodulator (demodulators are covered in Chapter (3)), and allow the negative portion to pass through the lower pulse frequency modulator and demodulator after being multiplied by a gain of –1. The bottom demodulator output is then multiplied by –1 again and summed with the top demodulator output to give the reconstructed input signal. The nonlinearities preceding the modulators act as either/or switches in this setup, but in general they may weigh the portion of the input going through each path.

The PPSSPFMD setup is used to model the control of a joint in the human body. Because the nervous system can only transmit pulses of one sign, and because muscles can only apply forces in contraction, it takes two sets of muscles pulling in opposite directions to control the movement of a joint. An example is the biceps and triceps opposing each other to control the elbow joint. PPSSPFMD is a cumbersome acronym, but it will be repeated frequently enough in this thesis to warrant its existence.

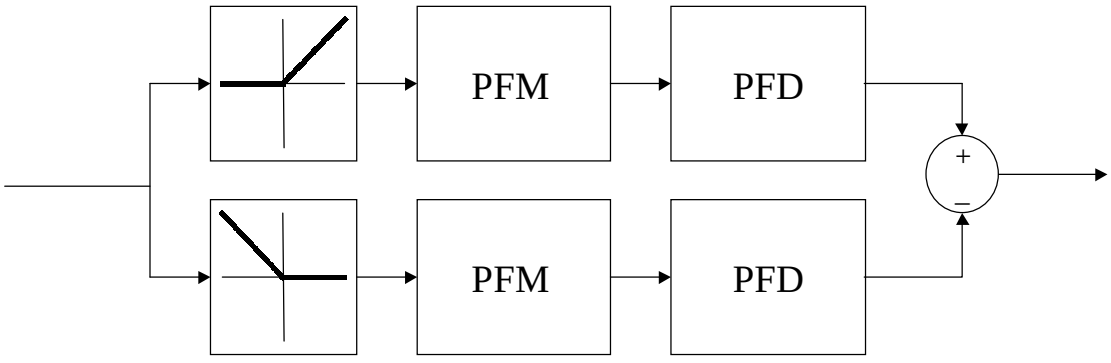


Fig. 11 Parallel-Path Single-Signed Pulse Frequency Modulator/Demodulator

3. COMPARISON OF PULSE FREQUENCY DEMODULATION METHODS

Pulse frequency demodulation (PFD) is even more integral to the control of the Experimental Neural Arm than is PFM. It is conceivable that the arm could be run open-loop (with only visual feedback), which requires no pulse frequency modulation, but PFD is always needed to decode the signals coming from the nervous system.

Many PFM methods were found to be basically equivalent to one another in Chapter 2, but the various ways to demodulate a PFM signal are very different from one another, and the advantages and disadvantages of each become very evident when they are qualitatively and quantitatively compared.

Period measurement is the first pulse frequency demodulation (PFD) method considered. This method reacts quickly when demodulating a high pulse frequency signal, but with relatively large errors, and it reacts slowly when demodulating a low pulse frequency signal, but with relatively small errors.

Low-pass filtering is the next PFD method considered. It is proposed as a way to mimic the way the nervous system demodulates a PFM signal. This method has an adjustable constant delay, but the error in the demodulated frequency grows when this delay is reduced. The error also grows as the pulse frequency decreases. Three types of low-pass filtering are considered: first-order low-pass filtering, second-order low-pass filtering, and finite-impulse-response filtering.

The final PFD method considered measures the number of pulses that occur during a fixed-time sampling window, and approximates the pulse period by dividing the sampling window by the number of pulses. The primary advantage of using this PFD method is that the sampling time is fixed, so traditional discrete control system design techniques can be employed. The primary disadvantage of this method is that the time delay is unnecessarily large when the pulse frequency is high. This PFD method has relatively large errors when demodulating low-frequency signals and relatively small errors when demodulating high-frequency signals.

Because the ultimate goal of this thesis is to help control an artificial arm with the human nervous system, only the PPSSPFMD system of Section 2.6 will be considered. From a demodulation perspective, this means demodulating two separate single-signed PFM signals, and then summing the effect of these two signals, so this chapter will only deal with the demodulation of single-signed pulse streams.

3.1 Period Measurement

Probably the most basic way of determining pulse frequency is by, at the occurrence of each pulse, measuring the period between the current pulse and the previous pulse, and then updating the demodulated frequency as the inverse of this period. The demodulated frequency only changes at the occurrence of a pulse.

This method updates the demodulated frequency at every pulse occurrence with the exception of the first pulse. Because no pulse has come before the first pulse, it acts as a marker that will not be used until the second pulse comes. This acts as a time delay of one pulse period in the demodulated frequency. This time delay is in addition to any time delay coming from the PFM method used. For a step input, the IPFM method gives

an additional delay of one pulse period, the USSH method gives an additional delay of one-half of one period, and the V/F converter gives no additional delay. When demodulating single-signed pulse streams, these delays are only seen at start-up.

3.1.1 Idealized Period Measurement

Figure 12 shows a stream of pulses along with its demodulated frequency. This pulse stream is initially at a frequency of 0.5 Hz, and steps up to a 1-Hz signal. The demodulated signal steps up at the 9-second mark, but the original signal that was pulse frequency modulated stepped up at the 8-second mark to create this pulse stream. This shows the time delay due to demodulation of 1 second (the new pulse period).

Figure 13 shows another stream of pulses with its demodulated frequency. This

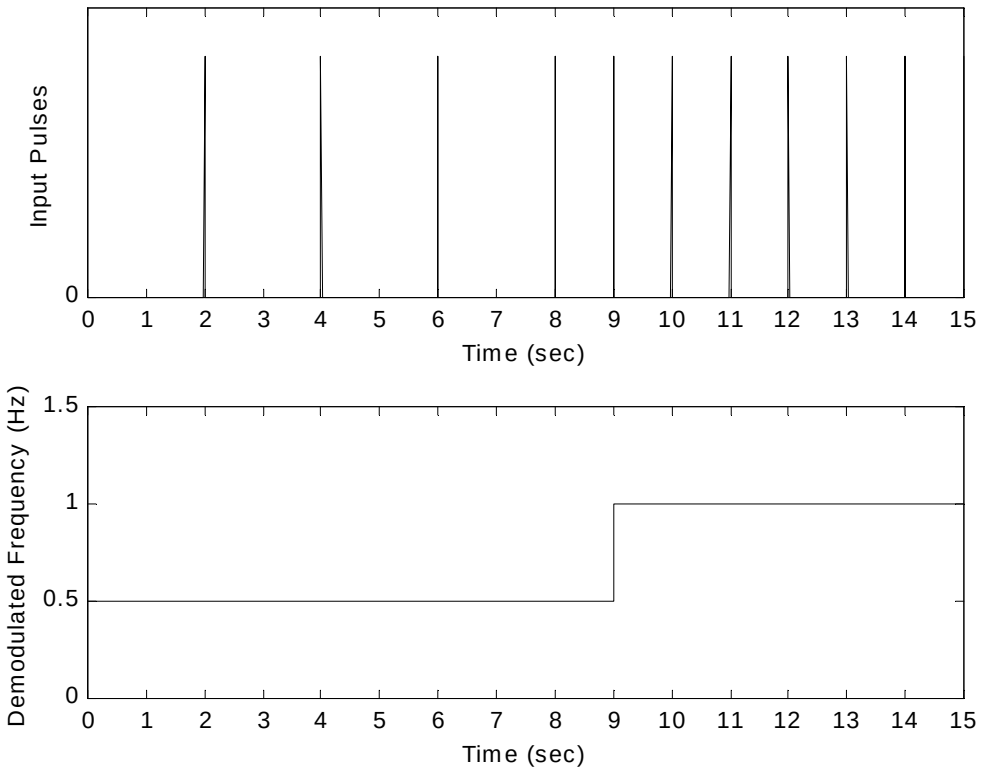


Fig. 12 Demodulation by Pulse Period Measurement for Input Step Up

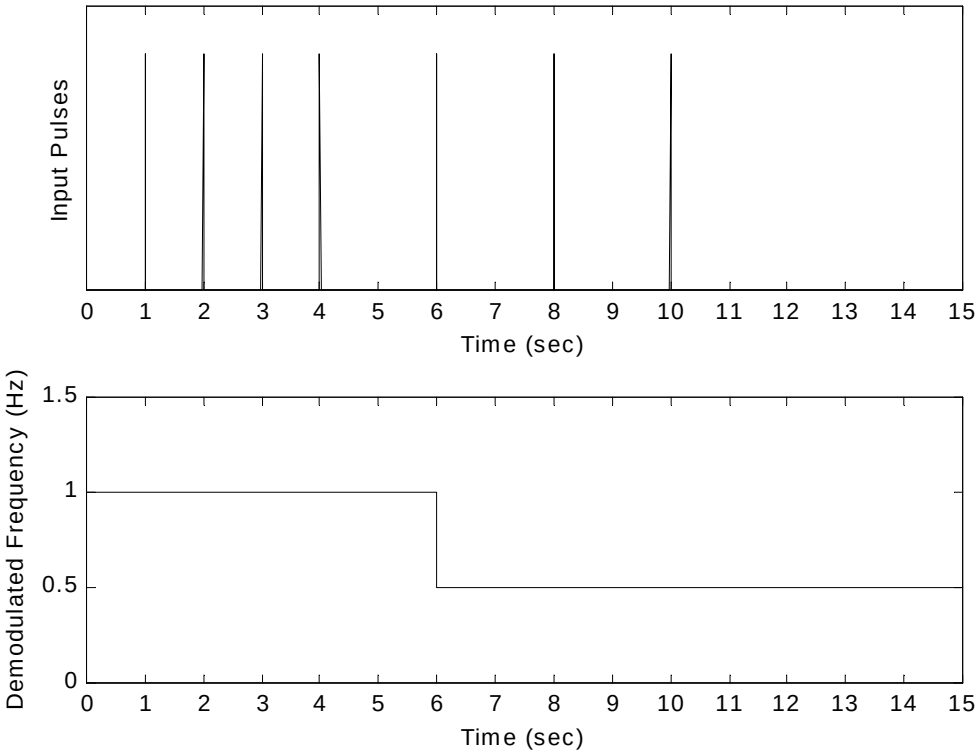


Fig. 13 Demodulation by Pulse Period Measurement for Input Step Down

pulse stream is initially at a frequency of 1 Hz, and steps down to a 0.5-Hz signal. The demodulated signal steps down at the 6-second mark, but the original signal that was pulse frequency modulated stepped down at the 4-second mark to create this pulse stream. The time delay is longer when stepping-down in pulse frequency, but the real problem with this method of PFD is what occurs after the pulse at the 10-second mark. No pulse occurs after this pulse, but the demodulated frequency never changes because the next pulse never comes. By the 12-second mark, the pulse frequency is obviously smaller than 0.5 Hz, but the demodulation method does not account for this.

There are two primary methods of dealing with the problem that is encountered when stepping down in pulse frequency. One of these methods is a relaxation method,

which uses knowledge of the previous pulse period to demodulate a signal in a more intelligent way. With the relaxation method, if a time *t* has passed since the last pulse, and *t* is greater than the previous pulse period, then the demodulated frequency in Hertz is:

$$t > T_{prev} \Rightarrow f_d = \frac{1}{t} \tag{24}$$

Figure 14 shows how Fig. 13 would look using period measurement with relaxation as the PFD method. The demodulated frequency will never reach zero with relaxation, but will only decay to an asymptote at zero. Practically, this may lead to steady-state errors, but may also be too negligible (very close to zero) to affect the performance of a real

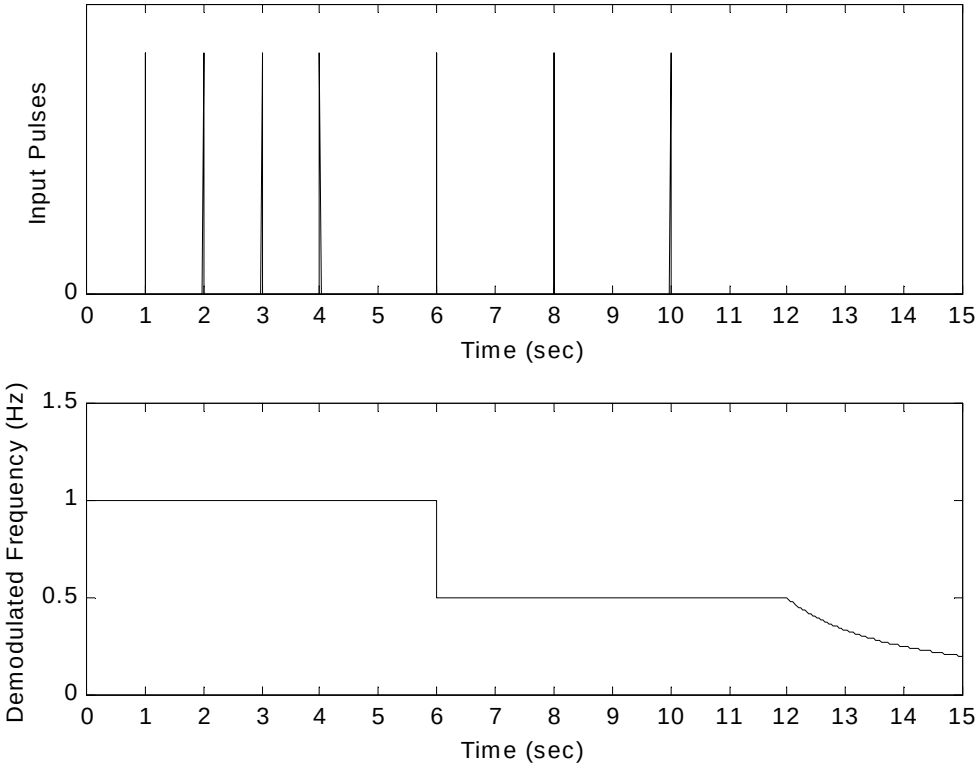


Fig. 14 Demodulation by Pulse Period Measurement with Relaxation for Input Step Down

system.

The other method used to correct the problem encountered when stepping down in pulse frequency is a deadband method. With the deadband method, all pulse frequencies smaller than a designated deadband frequency will be demodulated as a zero; more practically, if a time *t* passes, since the last pulse, that is greater than the designated deadband period T_{dead} (the inverse of the deadband frequency), then the demodulated frequency is set to zero:

$$t > T_{dead} \Rightarrow f_d = 0 \tag{25}$$

Figure 15 shows how Fig. 13 would look using period measurement with a deadband of 0.4 Hz.

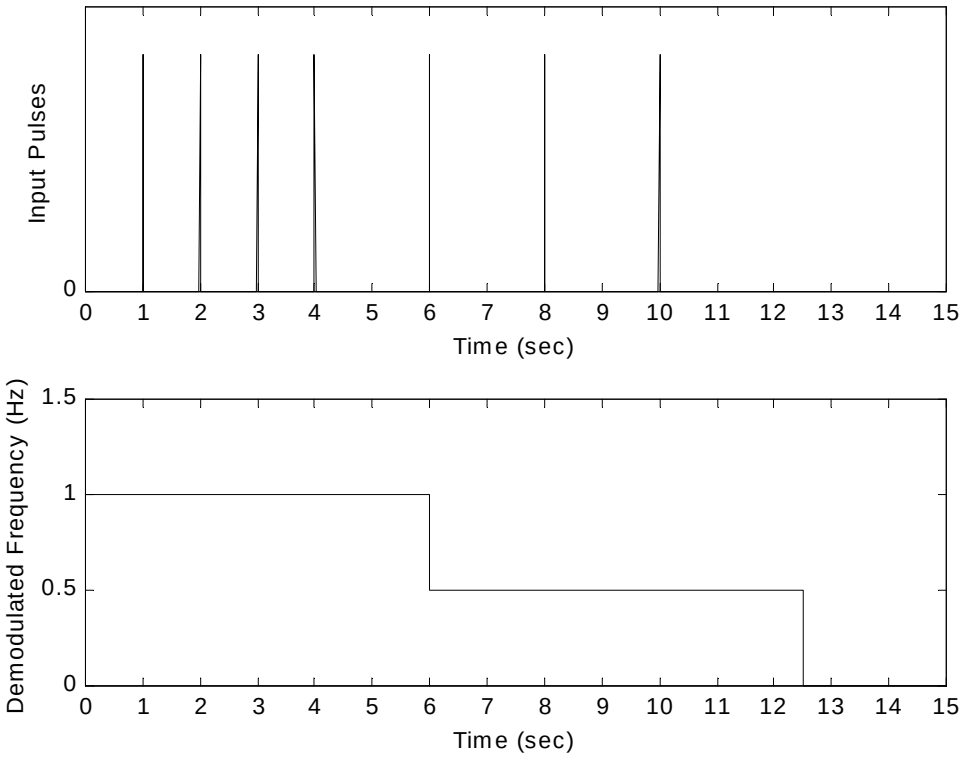


Fig. 15 Demodulation by Pulse Period Measurement with 0.4-Hz Deadband for Input Step Down

A combination of relaxation and deadband together may be the best option. This method would work especially well when stepping down from a large pulse frequency to a very small or zero pulse frequency; here the relaxation would react quickly, while the deadband would eventually drive the demodulated frequency to zero.

For period measurement PFD, knowledge of the modulation constant (see Chapter 2, Eqs. (17) through (19)) is needed to reconstruct the original modulated signal. If x is the original signal that was pulse frequency modulated, then let the reconstructed signal after demodulation, x_d, be defined as:

$$x_d = \frac{f_d}{k_M} \tag{26}$$

The reconstructed signal will only be absolutely correct for scenarios where the input to the modulator is a constant. For any varying input signal, the reconstructed signal will be only be an approximation of the original, but this is unavoidable due to the filtering nature of the integrators in the pulse frequency modulators and due to the discretized nature of the PFM signal.

3.1.2 Effect of Discrete Sampling on Period Measurement

This section deals with demodulation of a PFM signal using a digital computer with a fixed, known sampling rate. As with any analog-to-digital signal conversion, the faster the computer's sampling rate is relative to the signal's frequency, the better the computer can accurately reconstruct the signal. Only DC signals will be considered while trying to quantify errors due to digital demodulation because any other signals become far too complex, but the results translate well to low frequency signals because

the errors found here are instantaneous errors that are based on the instantaneous pulse frequency.

Let f and T be the actual pulse frequency in Hertz and the actual pulse period in seconds of the pulse stream being demodulated, let f_d and T_d be the demodulated frequency and period, and let f_s and T_s be the computer's sampling frequency and period. A pulse is always detected late (it is impossible to detect early); this is the nature of analog-to-digital conversion. Pulses are always detected late by a period Δ, where:

$$T_s \geq \Delta > 0 \tag{27}$$

Figure 16 shows the worst-case scenarios that would cause the largest errors in the demodulated period. From this figure, the value of the demodulated period will fall somewhere in the range:

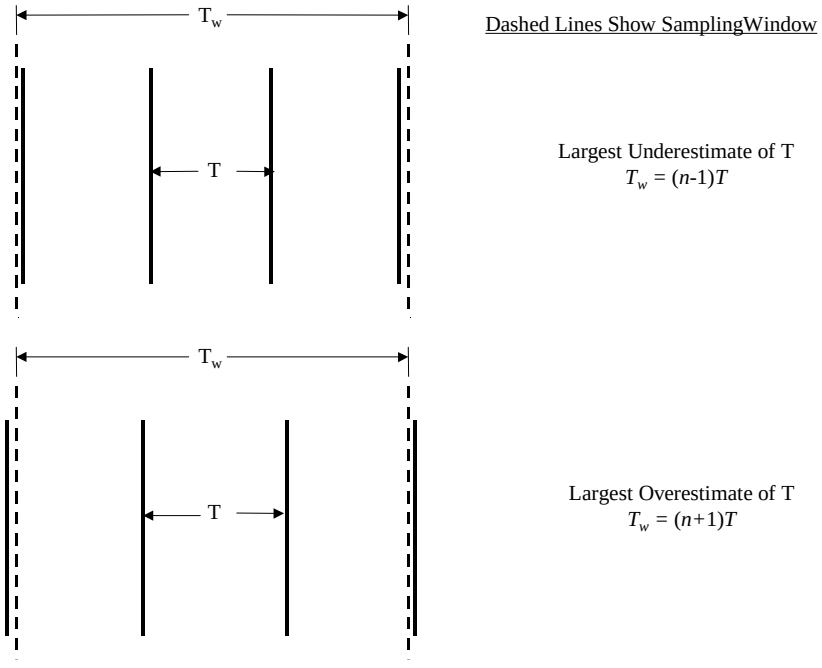


Fig. 16 Period-Measurement PFD with Discrete Samples

$$T + T_s > T_d > T - T_s \tag{28}$$

Each pulse is detected late, so for any given period measurement it is equally likely that the period will be overestimated or underestimated.

Note that an underestimate in period leads to an overestimate in frequency, and vice verse. If an error in the demodulated frequency ε_d is defined as:

$$\varepsilon_d = \frac{f_d - f}{f} \tag{29}$$

Using Eq. (28), ε_d will be found inside an error envelope with

$$\varepsilon_{over} \geq \varepsilon_d \geq \varepsilon_{under} \tag{30}$$

where, after some manipulation, the maximum possible overestimate and underestimate in the measured frequency are found as

$$\varepsilon_{over} = 1 - \frac{\frac{f_s}{f}}{\frac{f_s}{f} + 1} \tag{31}$$

$$\varepsilon_{under} = 1 - \frac{\frac{f_s}{f}}{\frac{f_s}{f} - 1} \tag{32}$$

These equations have been normalized to look at the error as a function of the relationship between sampling frequency and pulse frequency. Figure 17 plots the

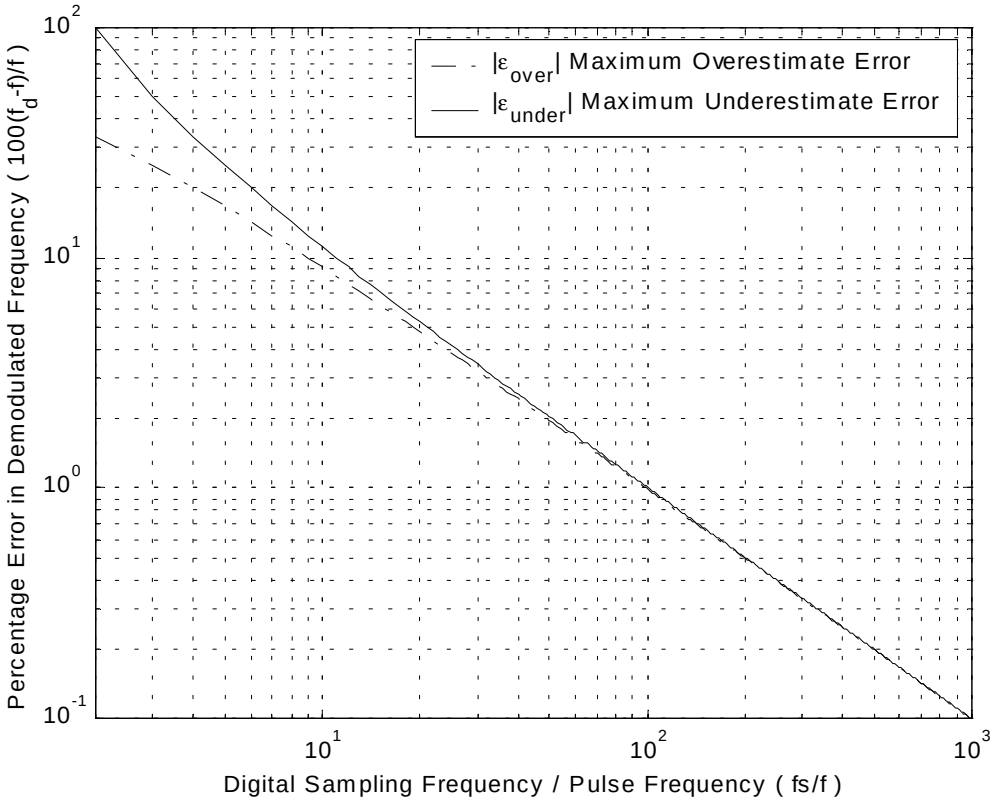


Fig. 17 Error in Period-Measurement PFD vs. Normalized Frequency

magnitudes of the error envelopes given in Eqs. (31) and (32) as a percentage of the pulse frequency.

Laboratory experiments show a pulse frequency range in the nervous system in the range of approximately 10-200 Hz [8]. Figure 18 shows the error envelopes for a 3000-Hz sampling rate; the frequency could be overestimated by as much as 6.3% or underestimated by as much as 7.1% when measuring a 200-Hz pulse stream, or could be overestimated or underestimated by as much as 0.33% when measuring a 10-Hz pulse stream, when using a computer with a 3000-Hz sampling rate.

A Simulink simulation of pulse measurement PFD is given in Appendix B. This simulation uses a 10-Hz deadband, and can demodulate single- or double-signed signals.

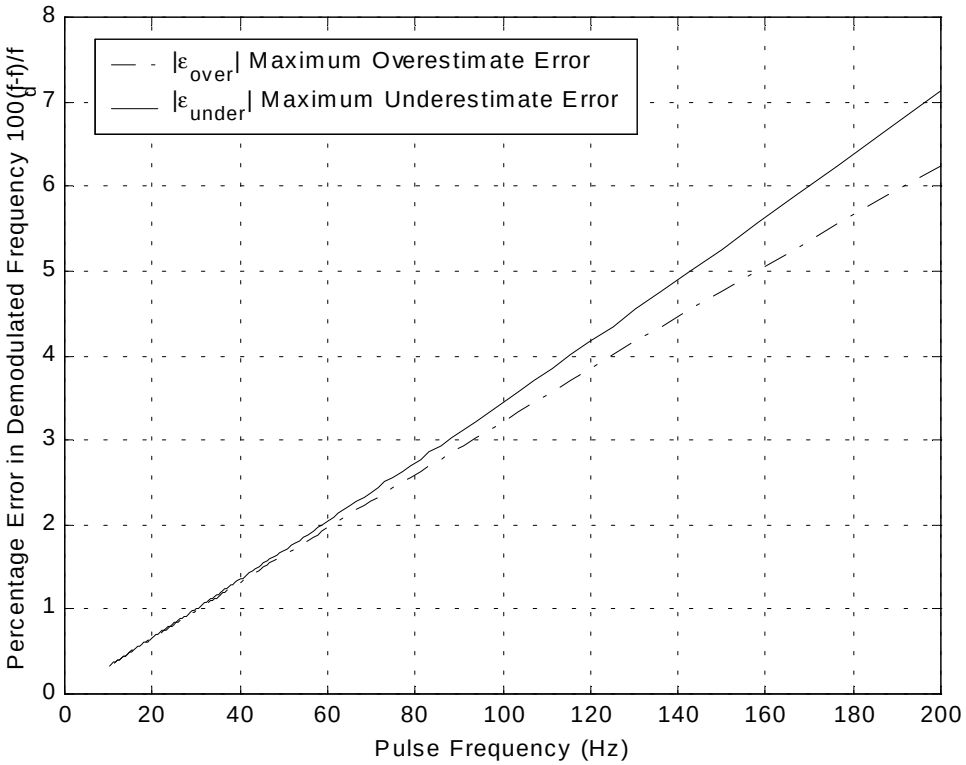


Fig. 18 Error in Period-Measurement PFD vs. Frequency for a 3000-Hz Sampling Rate

3.2 Low-Pass Filtering of Pulses for Pulse Frequency Demodulation

The human nervous system seems to use a type of low-pass filtering as its PFD method. Each electrical pulse causes a muscle twitch. If these twitches come close enough together, they result in an aggregate muscle movement [9]. In an effort to mimic the way the human body works, it is desirable to investigate low-pass filtering as a PFD method.

It seems possible that a pulse stream may be demodulated with a filter for some type of PFD. Each pulse would create an impulse response in the filter, which is an instantaneous increase in the filtered signal followed by a decay to zero. If a second pulse occurs before the first pulse decays away, the convolution of the two pulses will

create a net filter response. This behavior seems to mimic the twitch response of a muscle.

Three types of low-pass filtering are considered here: first-order low-pass filtering, second-order low-pass filtering, and finite-impulse-response filtering.

3.2.1 First-Order Low-Pass Filtering

Consider a first-order low-pass filter as pulse frequency demodulator, in Laplace domain:

$$y(s) = \frac{\beta}{s+\alpha}\delta(s) \tag{33}$$

where δ is the incoming pulse stream to be modulated, with a pulse frequency f and pulse period T, and y is the demodulator output. The output is not labeled as the demodulated frequency f_d because there is no indication of how the output of the filter is related to the input pulse frequency. Figure 19 shows the response of the filter of Eq. (33) when $\alpha = \beta = 50$ and $f = 100$ Hz. The time constant τ of the filter (the inverse of α) is 0.02 seconds. It is seen from this plot that the output y has an aggregate response of a step input to the low-pass filter with a time constant of what appears to be 0.02 seconds and with a "steady-state" value that jitters, but appears to be centered around 100, which is the value of f. Changing the DC gain of the filter, β/α, would only scale the response linearly. These properties seem encouraging for use of a low-pass filter for PFD.

The impulse response of the filter of Eq. (33) is:

$$y_\delta(t) = \beta e^{-\alpha t} \tag{34}$$

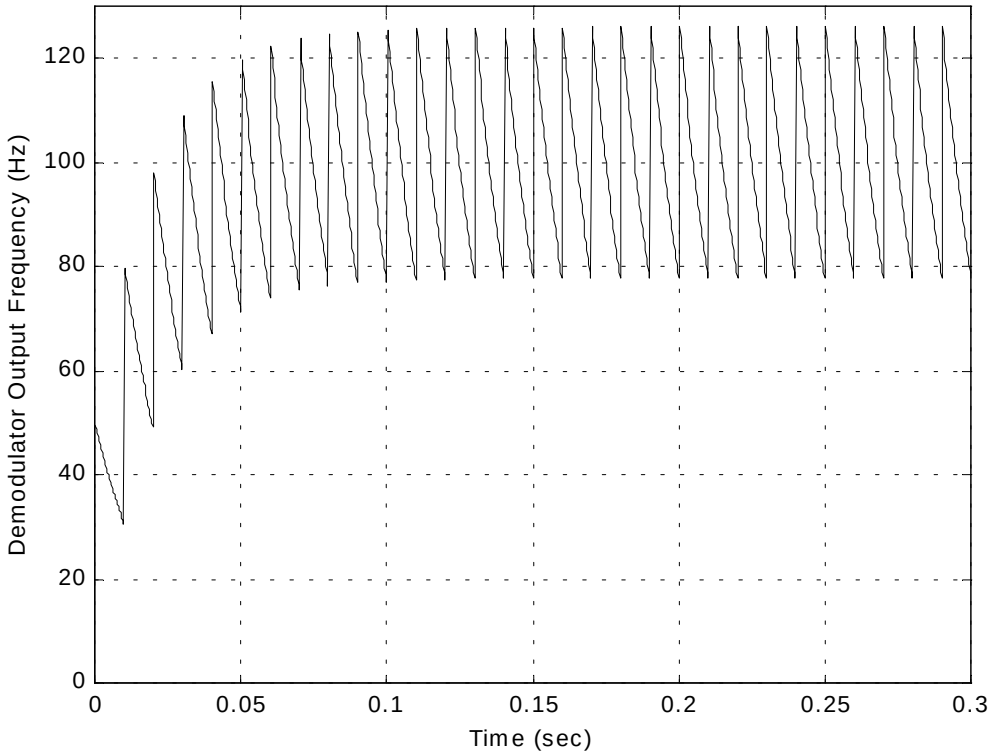


Fig. 19 Response of Unity-Gain First-Order Filter with $5/\alpha = 0.1$ Seconds to a 100-Hz Pulse Frequency

which means that each impulse causes an instantaneous increase of β in the output as is seen in Fig. 19. The total response y at a time $\Delta \leq T$ since the last pulse, due to all previous pulses, is given by:

$$y(\Delta) = \beta e^{-\alpha\Delta} + \beta e^{-\alpha(\Delta+T)} + \beta e^{-\alpha(\Delta+2T)} + \ldots \tag{35}$$

$$y(\Delta) = \beta e^{-\alpha\Delta}(1 + e^{-\alpha T} + e^{-2\alpha T} + \ldots) \tag{36}$$

$$y(\Delta) = \beta e^{-\alpha\Delta}\left(\sum_{k=0}^{\infty} e^{-k\alpha T}\right) \tag{37}$$

If $\alpha T > 0$, then $y(\Delta)$ can be written as:

$$y(\Delta) = \beta e^{-\alpha\Delta}\left(\frac{e^{\alpha T}}{e^{\alpha T} - 1}\right) \tag{38}$$

Equation (38) assumes that pulses have been coming at a constant frequency for an infinite amount of time. Remembering the nature of exponential decay, only the pulses occurring in the past five time constants (using a 99% settling time) have any measurable effect on the total response. This means that Eq. (38) is always an approximation, but it is a good approximation after five time constants past the last change in input frequency.

Keeping in mind the shape of the plot in Fig. 19, define the highest value of y in the "steady-state" (occurring just at an impulse) as y_h, and define the lowest value of y in the "steady-state" (occurring just before an impulse) as y_l. Using Eq. (38):

$$\Delta = 0 \Rightarrow y_h = \frac{\beta e^{\alpha T}}{e^{\alpha T} - 1} \tag{39}$$

$$\Delta = T \Rightarrow y_l = \frac{\beta}{e^{\alpha T} - 1} \tag{40}$$

Are $y_h \geq f$ and $y_l \leq f$ always true statements for a unity-gain ($\beta = \alpha$) first-order filter? First look at the statement about y_h:

$$y_h \geq f \Rightarrow \frac{y_h}{f} \geq 1 \Rightarrow y_h T \geq 1 \Rightarrow \frac{T\alpha e^{\alpha T}}{e^{\alpha T} - 1} \geq 1 \tag{41}$$

Because $\alpha T > 0$, the last statement of Eq. (41) can be written as:

$$(\alpha T - 1)e^{\alpha T} \geq -1 \tag{42}$$

This statement is always true for $\alpha T \geq 0$, therefore the statement $y_h \geq f$ is always true with the assumptions given previously. With a similar methodology, it can easily be shown that $y_l \leq f$.

Do $y_h \rightarrow f$ and $y_l \rightarrow f$ as $f \rightarrow \infty$? If so, $y_h/f \rightarrow 1$ and $y_l/f \rightarrow 1$ as $f \rightarrow \infty$:

$$\lim_{f\to\infty}\left(\frac{y_h}{f}\right) = \lim_{f\to\infty}\left(\frac{\left(\frac{\alpha e^{\alpha T}}{e^{\alpha T}-1}\right)}{f}\right) = \lim_{T\to 0}\left(\frac{T\alpha e^{\alpha T}}{e^{\alpha T}-1}\right) = \frac{0}{0} \tag{43}$$

Use l'Hopital's Rule:

$$\lim_{T\to 0}\left(\frac{\alpha^2 T e^{\alpha T} + \alpha e^{\alpha T}}{\alpha e^{\alpha T}}\right) = \lim_{T\to 0}(\alpha T + 1) = 1 \tag{44}$$

Therefore $y_h \rightarrow f$ as $f \rightarrow \infty$. With a similar methodology, it can easily be shown that $y_l \rightarrow f$ as $f \rightarrow \infty$.

Figure 20 shows the maximum errors, defined in Eq. (29), due to the output highs and lows of Eqs. (39) and (40). Notice that Eqs. (39) and (40), and Figure 20, match what is seen in Fig. 19 well, predicting the saw-toothed oscillation between approximately 77 and 127 Hz.

Because the demodulation error increases proportionally with α/f, low-pass filter demodulation works better for high- rather than low-pulse-frequency signals, and works better when the filter's time constant is longer rather than shorter.

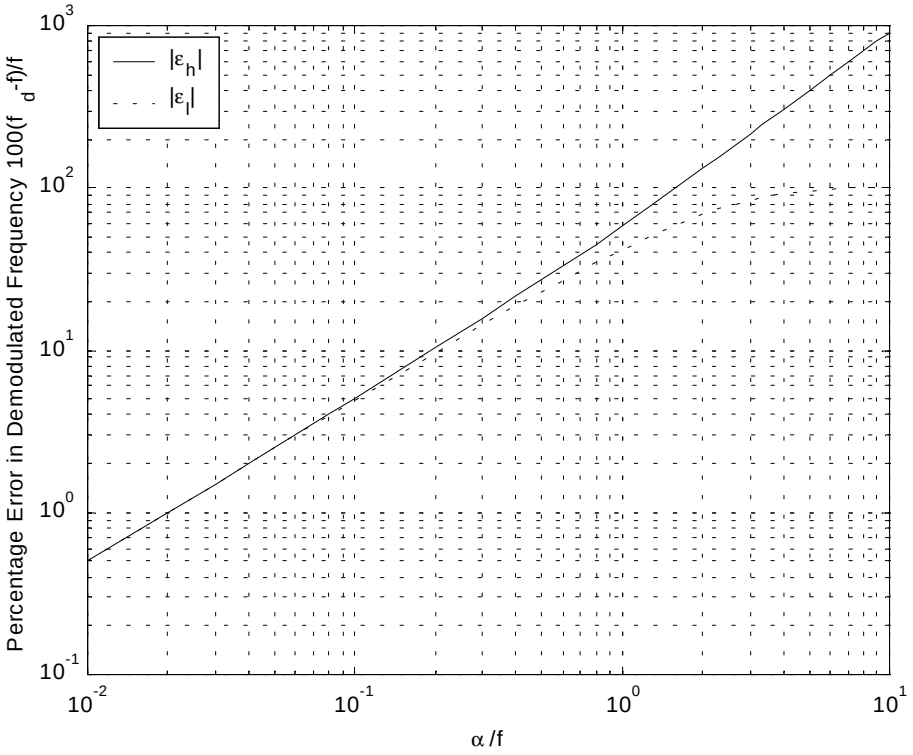


Fig. 20 Error in Unity-Gain First-Order Low-Pass Filter vs. Normalized Frequency

Figure 21 shows the maximum errors for a first-order filter with $5/\alpha = 0.1$ seconds. For this filter, the error can be as high as 400% when demodulating a 10-Hz signal, and is still large when demodulating a 200-Hz signal.

3.2.2 Second-Order Low-Pass Filtering

A first-order low-pass unity-gain filter can be used as a pulse frequency demodulator. The output of the demodulator jitters around the input frequency value with a known error in the "steady-state," and it reaches the "steady-state" based on the time constant of the filter. A low-pass filter of higher order may give more desirable results by smoothing out the response seen in Fig. 19.

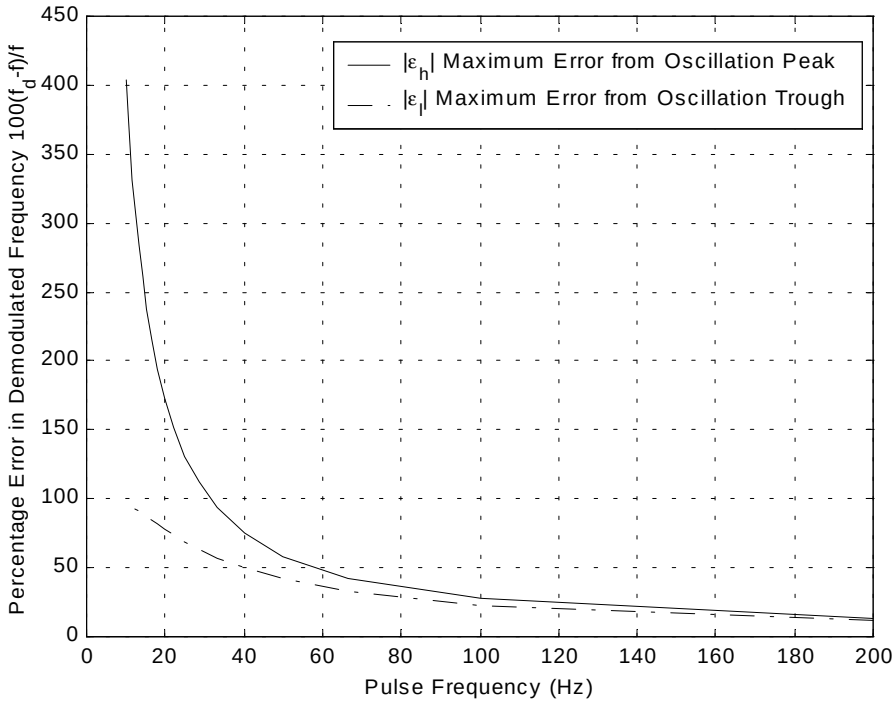


Fig. 21 Error in First-Order Filtering PFD vs. Frequency with $5/\alpha = 0.1$ Second

Consider a second-order low-pass unity-gain filter as demodulator, in Laplace domain:

$$f_d(s) = \frac{\alpha^2}{(s+\alpha)^2}\delta(s) \tag{45}$$

where δ is the incoming pulse stream to be modulated, with a pulse frequency f and pulse period T, and y is the demodulator output. The output is labeled as the demodulated frequency f_d, unlike in Eq. (33), because it was proven in Section 3.2.1 that the filter output is the best value to use for f_d for a unity-gain filter. For the same reason, the β in the numerator of the first-order filter of Eq. (33) has been changed in Eq. (45) to force the

filter to have unity-gain. Figure 22 shows the response of the filter of Eq. (45) when α = 76.4 and f = 100 Hz. This response is much smoother than that of the first-order filter.

The impulse response of the filter in Eq. (45) is given by:

$$y_\delta(t) = \alpha^2 t e^{-\alpha t} \tag{46}$$

which starts a zero, climbs up to a peak value, and then decays away to zero, all with no discontinuities. The peak value of the impulse response occurs at $t = 1/\alpha$ seconds, which is the time constant of the first-order filter, but is a time-to-peak of the second-order filter. Let the time for the impulse response of a second-order filter to decay be measured relative to the value of the response at the peak ($t = 1/\alpha$ seconds). The time for the

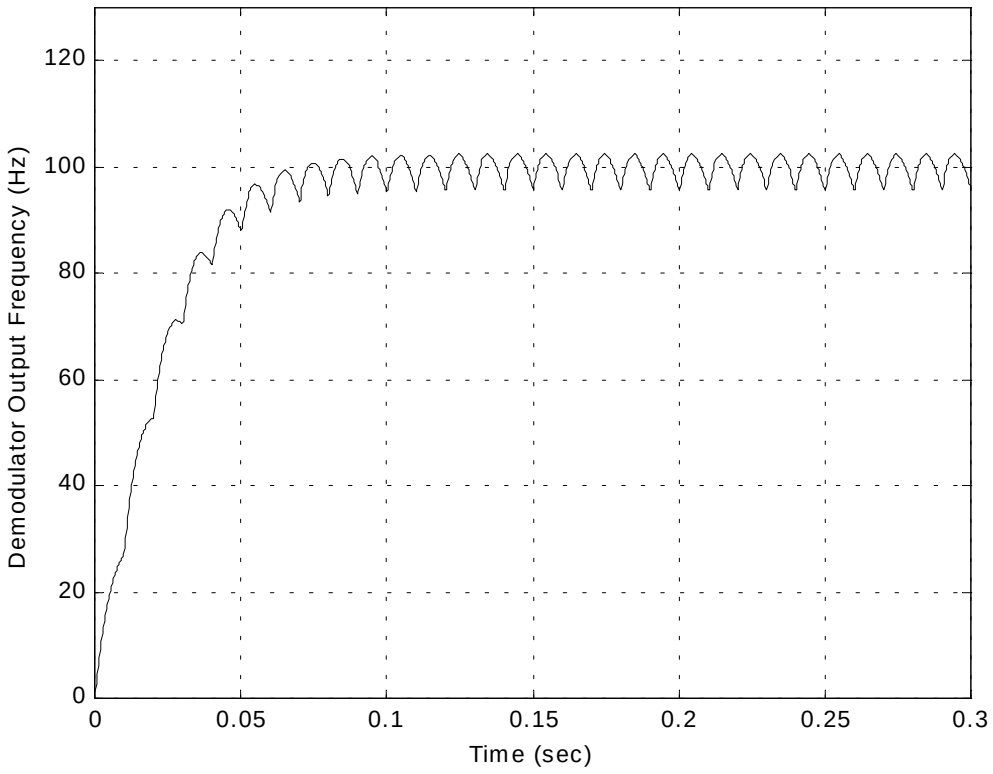


Fig. 22 Response of Unity-Gain Second-Order Filter with 7.64/α = 0.1 Seconds to a 100-Hz Pulse Frequency

impulse response to decay 95%, 98%, and 99% of the way to zero are found to be:

$$t_{95\%} = \frac{5.74}{\alpha} \tag{47}$$

$$t_{98\%} = \frac{6.83}{\alpha} \tag{48}$$

$$t_{99\%} = \frac{7.64}{\alpha} \tag{49}$$

The second-order response appears much smoother than the first-order response, but Eqs. (47) through (49) show that having the same α is not a satisfactory method to compare the first- and second-order responses, but rather α should be chosen to make the two filters have the same speed of response.

The total response y at a time $\Delta \le T$ since the last pulse, due to all previous pulses, is given by:

$$f_d(\Delta) = \alpha^2 \Delta e^{-\alpha\Delta} + \alpha^2 (\Delta + T) e^{-\alpha(\Delta+T)} + \alpha^2 (\Delta + 2T) e^{-\alpha(\Delta+2T)} + \ldots \tag{50}$$

$$f_d(\Delta) = \alpha^2 e^{-\alpha\Delta} (\Delta + (\Delta + T) e^{-\alpha T} + (\Delta + 2T) e^{-2\alpha T} + \ldots) \tag{51}$$

$$f_d(\Delta) = \alpha^2 e^{-\alpha\Delta} (\Delta + \Delta e^{-\alpha T} + \Delta e^{-2\alpha T} + \ldots + T e^{-\alpha T} + 2T e^{-2\alpha T} + \ldots) \tag{52}$$

$$f_d(\Delta) = \alpha^2 e^{-\alpha\Delta} \left(\Delta \left(\sum_{k=0}^{\infty} e^{-k\alpha T} \right) + T \left(\sum_{k=0}^{\infty} k e^{-k\alpha T} \right) \right) \tag{53}$$

$$f_d(\Delta) = \alpha^2 e^{-\alpha\Delta} \left(\frac{\Delta e^{\alpha T}}{e^{\alpha T} - 1} + \frac{T e^{\alpha T}}{(e^{\alpha T} - 1)^2} \right) \tag{54}$$

Equation (54) assumes that pulses have been coming at a constant frequency for an infinite amount of time, but only the pulses occurring in the past $7.64/\alpha$ seconds have any measurable effect on the total response, using a 99% settling time. This means that Eq. (54) is always an approximation, but it is a good approximation after $7.64/\alpha$ seconds past the last change in input frequency.

Keeping in mind the shape of the plot in Fig. 22, define the highest value of f_d in the "steady-state" (occurring at an impulse response peak) as f_{dh}, and define the lowest value of f_d in the "steady-state" (occurring at an impulse) as f_{dl}. Using Eq. (54):

$$\Delta = 0, T \Rightarrow f_{dl} = \frac{\alpha^2 T e^{\alpha T}}{(e^{\alpha T} - 1)^2} \tag{55}$$

To find f_{dh}, differentiate Eq. (54) with respect to Δ and set equal to zero. This gives the value of Δ at the peak:

$$\Delta_{peak} = \frac{1}{\alpha} - \frac{T}{e^{\alpha T} - 1} \tag{56}$$

Substituting Eq. (56) into Eq. (54) gives the value of f_{dh}, which is not given in closed form here, due to its complexity.

$$f_{dh} = f_d(\Delta_{peak}) \tag{57}$$

Figure 23 shows the maximum errors due to the output highs and lows of Eqs. (55) and (57). Notice that Figure 23 matches what is seen in Fig. 22 well, predicting the saw-toothed oscillation between approximately 95 and 103 Hz.

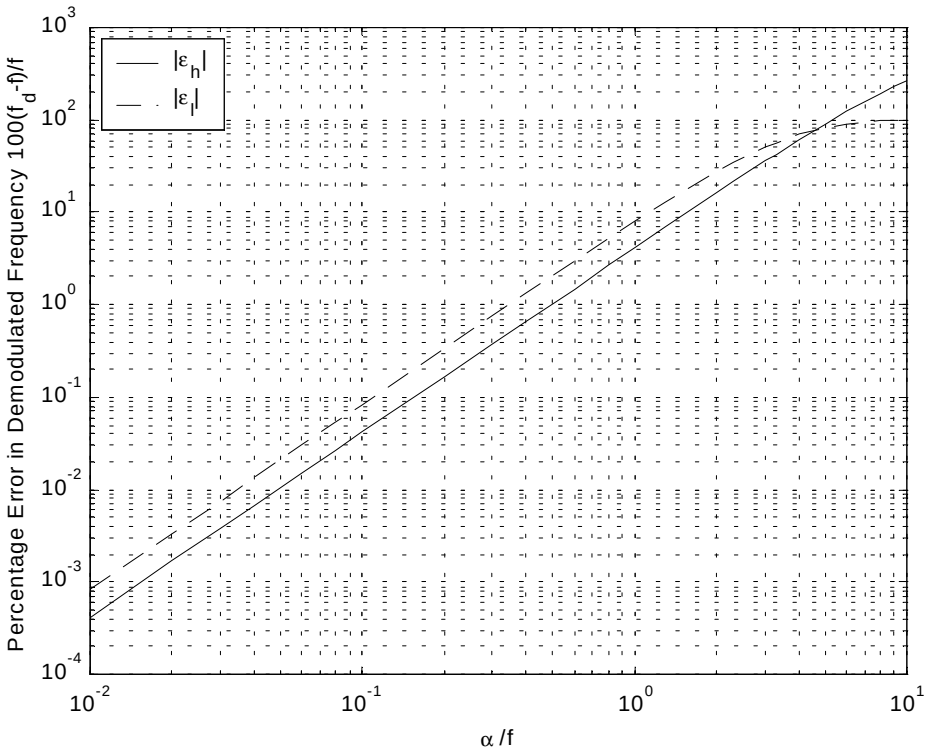


Fig. 23 Error in Unity-Gain Second-Order Low-Pass Filter vs. Normalized Frequency

Figure 24 shows the maximum errors for a second-order filter with $7.64/\alpha = 0.1$ seconds. This filter gives a 180% error when demodulating a 10-Hz signal, compared with 400% with the first-order low-pass filter. The error is also smaller at high frequencies.

3.2.3 Finite-Impulse-Response Filtering

Another proposed filter for PFD is a finite-impulse-response (FIR) filter, purely designed for use on a digital computer. This method of PFD involves storing the last *N* samples of the incoming pulse stream in a stack, where *N* is some constant integer. A pointer cycles through the stack, moving one stack element per computer sample, adding *N* to the stack element if a pulse is detected. Between each computer sample, the entire

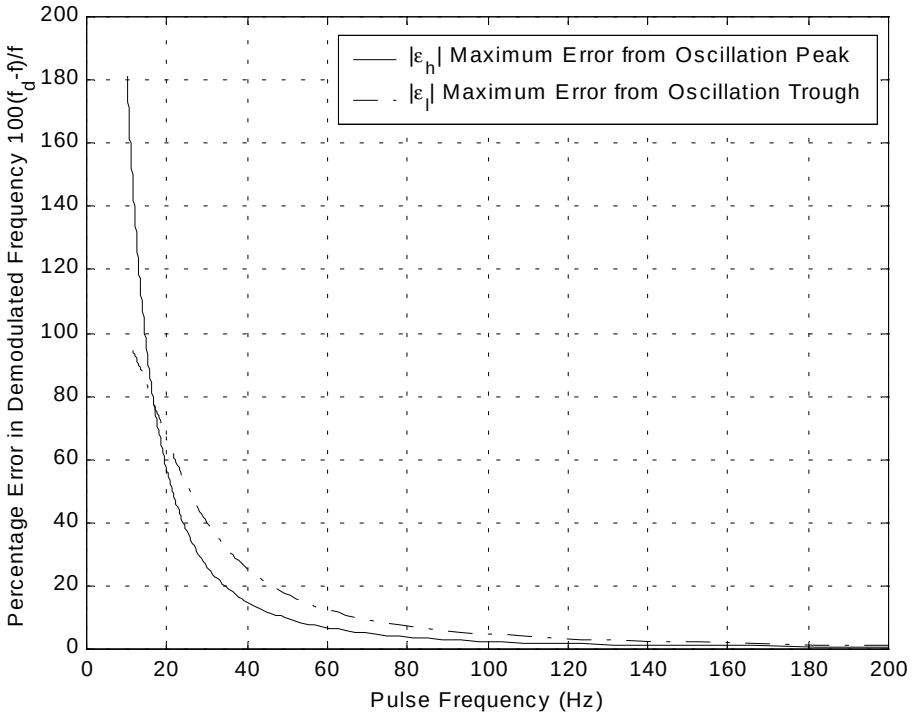


Fig. 24 Error in Second-Order Filtering PFD vs. Frequency for $7.64/\alpha = 0.1$ Seconds

stack is cycled through, and each nonzero element is reduced by 1. The output of the demodulator at each computer sample is the sum of all the stack elements.

This PFD method creates a low-pass filter whose impulse response decays away linearly to zero, rather than exponentially like a first-order low-pass filter. With this method, any pulses occurring more than *N* computer samples ago have no effect on the output, where a low-pass filter theoretically feels the effect of all previous pulses.

Figure 25 shows the response of an FIR filter with a 0.1-second decay to a 100-Hz input frequency. This FIR filter seems to have a similar response as that of the first-order low-pass filter. Note that with an FIR filter the decay between individual pulses is linear, rather than exponential. The same sort of discontinuities are seen in the response, and they have a similar behavior in the "steady-state." Probably the largest difference

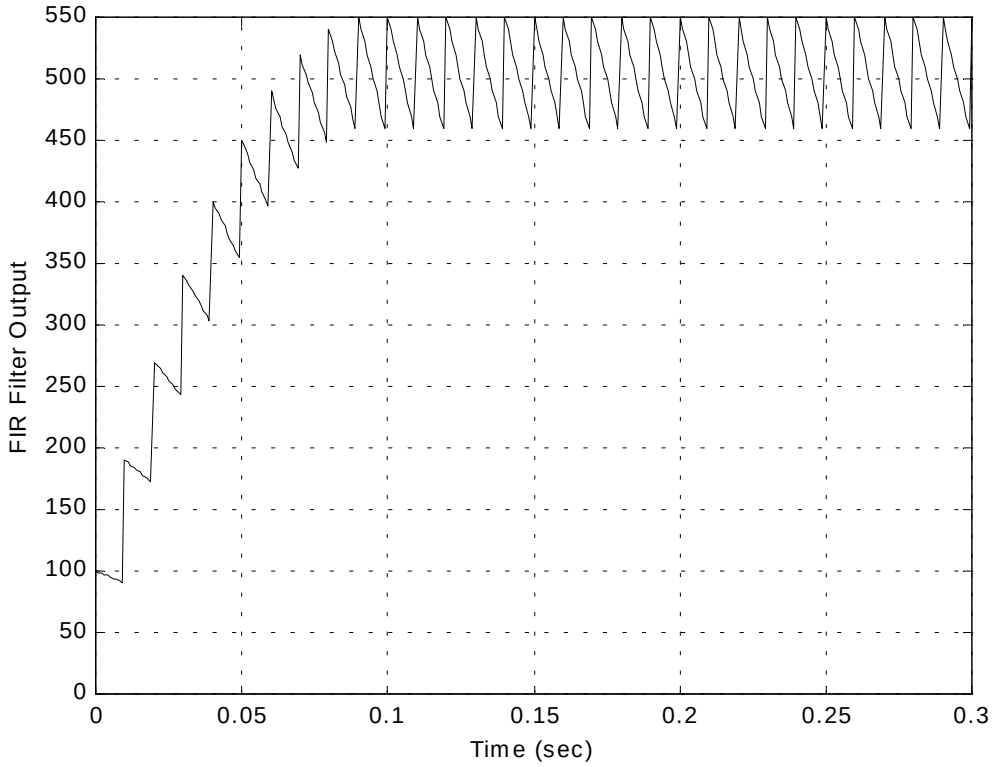


Fig. 25 Response of an FIR Filter with a 0.1-Second Decay to a 100-Hz Pulse Frequency

between the FIR filter and the first-order low-pass filter is that, for a first-order low-pass filter with unity DC gain, the aggregate response approaches a “steady-state” equal to the pulse frequency in Hertz, but the FIR filter does not show this behavior.

Because of the nonlinearity of the FIR filter, a closed-form general solution of the filter’s response to a constant-frequency input pulse stream is not found here. A solution can be found, however, that works for a limited number of input/filter combinations, and this solution can be used as an approximation for other cases. Consider an FIR filter with a n output y to an input pulse stream of constant frequency f and period T. Let the filter have an impulse response that decays away in Δ seconds, with:

$$\Delta = NT_s \tag{58}$$

where N is the number of computer samples that it takes for an impulse response to decay to zero, and T_s is the computer sampling period. Let the input pulse period be defined as:

$$T = nT_s \tag{59}$$

where n is an integer. This restricts the possible input frequencies, but this is a necessity of the approximation. The FIR filter has an aggregate response, which was found empirically, given by:

$$y = 0.5\frac{\Delta^2}{T_s}\left(1+\frac{T_s}{\Delta}\right)f \tag{60}$$

So, for a given FIR filter, y is linearly proportional to f, and Eq. (60) can be written as:

$$y = k_{FIR} f \tag{61}$$

Figure 26 shows the same response of Fig. 25, but the constant of Eqs. (60) and (61) is accounted for. Now the aggregate filter output is the demodulated frequency.

Equation (60) accurately predicts the mean value between the saw-toothed oscillation highs and lows in the "steady-state." These output highs and lows are found by the equations:

$$y_h = (k_h+1)N - \sum_{i=1}^{k_h} in \tag{62}$$

$$y_h = (k_h+1)N - \frac{nk_h}{2}(k_h+1) \tag{63}$$

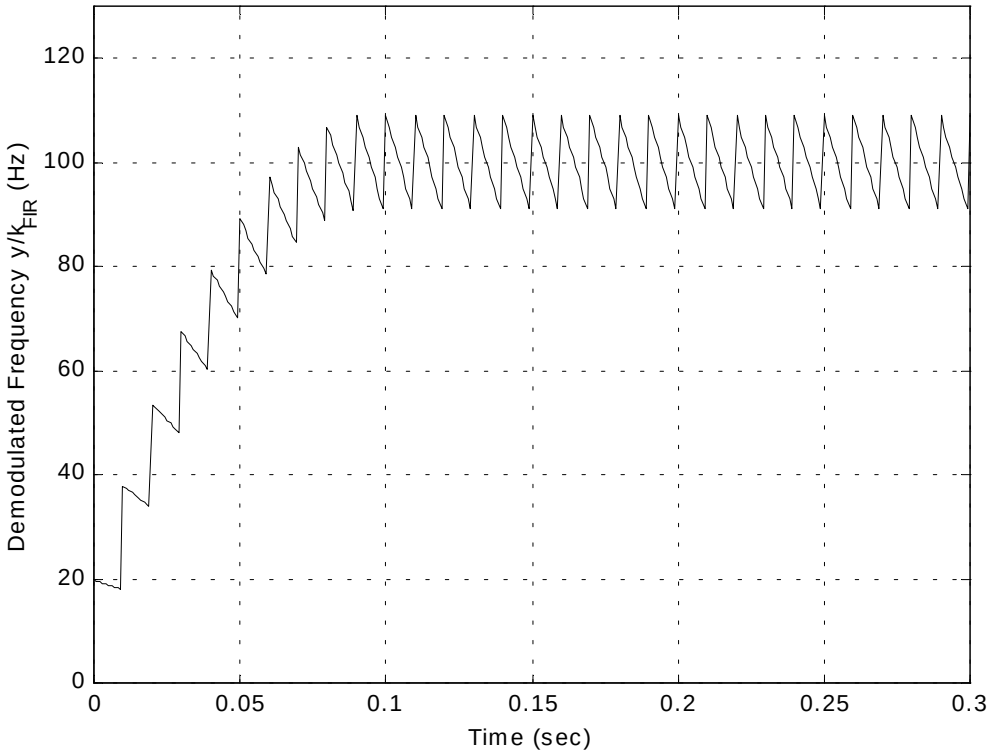


Fig. 26 FIR Filter PFD with a 0.1-Second Decay to a 100-Hz Pulse Frequency

$$k_h = floor\left(\frac{N}{n}\right) \tag{64}$$

$$y_l = (N+1)k_l - \sum_{i=1}^{k_l} in \tag{65}$$

$$y_l = (N+1)k_l - \frac{nk_l}{2}(k_l+1) \tag{66}$$

$$k_l = floor\left(\frac{N+1}{n}\right) \tag{67}$$

The *floor*() function returns the closest integer to the quantity in the parenthesis, in the direction of negative infinity.

Using the definition of error from Eq. (29), the errors due to the "steady-state" highs and lows of Eqs. (63) and (66) are plotted in Fig. 27; this plot is specifically for an FIR filter with a 3000-Hz sampling rate and a 0.1-second impulse decay.

3.3 Fixed-Time Sampling Window

Another way to demodulate a PFM signal is by counting the number of pulses that occur in a fixed-time sampling window. This method has the advantage of using traditional digital design techniques, because the fixed-time sampling window becomes the sampling period. If the sampling window is T_w seconds, and N pulses occur in a given window, the demodulated period is defined as:

$$T_d = \frac{T_w}{N} \tag{68}$$

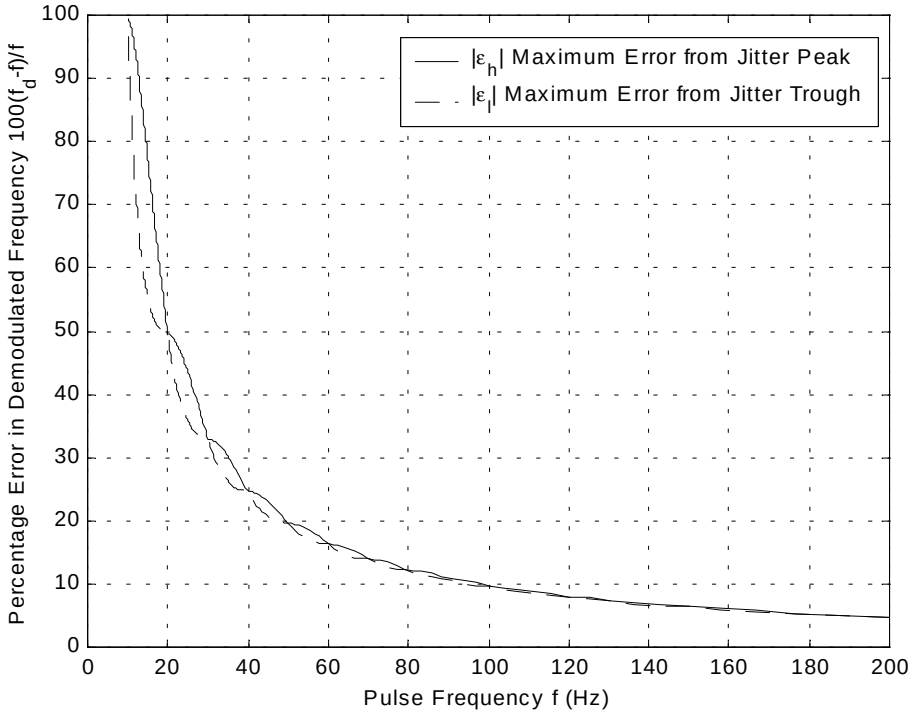


Fig. 27 Error Band for a 3000-Hz FIR Filter with a 0.1-Second Decay

This gives a demodulated frequency of:

$$f_d = Nf_w \tag{69}$$

Figure 28 shows the worst-case demodulation scenarios that can occur using Eq. (68). In the portion of the figure showing the largest underestimate of *T*, pulses occur just inside the edges of the sampling window. The window is divided into three equal parts, but four pulses occur in the window, so Eq. (68) will divide T_w into four parts, which underestimates the period. In the portion of the figure showing the largest overestimate of *T*, pulses occur just outside the edges of the sampling window. The window is still divided into three equal parts, but only two pulses occur in the window, so Eq. (68) will divide the T_w into two parts, which overestimates the period.

Figure 28 and Eq. (68) put the demodulated pulse period in the range:

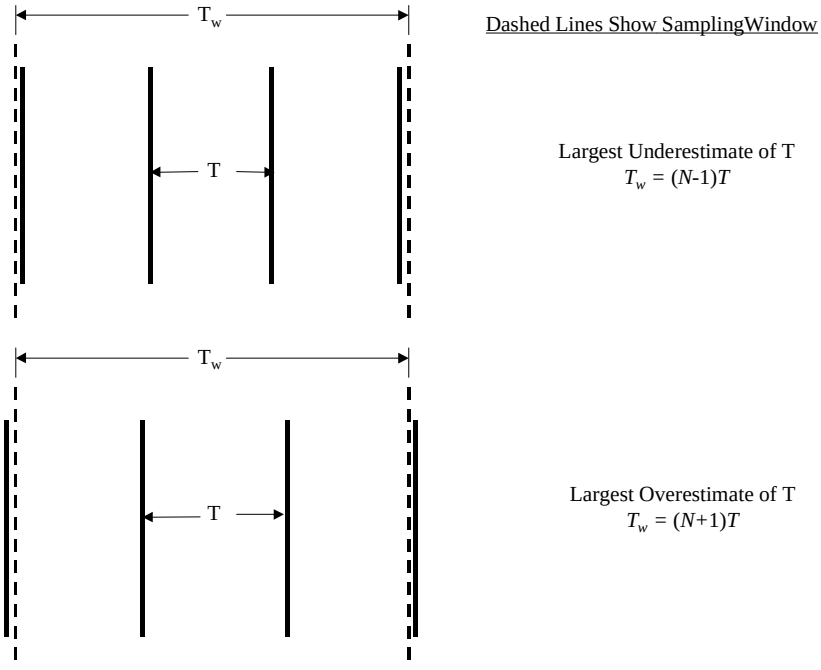


Fig. 28 Fixed-Time Sampling Window

$$\left(\frac{N+1}{N}\right)T \geq T_d \geq \left(\frac{N-1}{N}\right)T \tag{70}$$

If an error in the demodulated frequency is defined as in Eq. (29), a little manipulation of Eqs. (29) and (68) yields a bounded error:

$$\frac{T}{T_w} \geq \varepsilon_d \geq -\frac{T}{T_w} \tag{71}$$

$$\frac{f_w}{f} \geq \varepsilon_d \geq -\frac{f_w}{f} \tag{72}$$

Th error of Eq. (29) is given as a fraction of f, so a window frequency of 5 Hertz and a pulse frequency of 50 Hertz would lead to an error in the demodulated frequency of as much as ±10% of f. This method only works if $f > f_w$. If $f \leq f_w$, the demodulated frequency will jitter between zero (when no pulses occur in a window) and f_w (when one pulse occurs in a window).

Fixed-Time Sampling Window PFD has large errors when demodulating small pulse frequencies, and small errors when demodulating large pulse frequencies; the opposite of the behavior seen with period measurement PFD.

3.4 Error Comparison of Pulse Frequency Demodulation Methods

To compare the various PFD methods, it is necessary the find some way to compare equivalent demodulators, because the behavior of each demodulator is fundamentally different from the others. The method used here will be to compare demodulators with the same settling time. Because the lowest pulse frequency seen in

the nervous system is approximately 10 Hz [8], the pulse-period measurement method will have a deadband of 10 Hz, giving a settling time of 0.1 seconds (worst-case). The fixed-time sampling window of T_w = 0.1 seconds will also be used. For the two continuous low-pass filters, α will be chosen to give a 99% settling time of 0.1 seconds. For the FIR filter, Δ is chosen to give a decay of 0.1 seconds. Table 2 quantifies the errors from these PFD methods.

It appears from Table 2 that pulse measurement may be the best overall method. It has low errors across applicable frequencies, and an added benefit of small time delays at high frequencies, when it is most important. Second-order low-pass filtering is probably the second-best method, but the FIR filter and fixed-time window have lower errors at low frequencies than the second-order low-pass filter.

While the human body demodulates incoming pulses through some form of low-pass filtering, the sheer number of nerves transmitting information in parallel and

Table 2 Comparison of Errors for PFD Methods with 0.1-Second Settling Times

PFD Method	Settling Time Parameter	f = 10 Hz ε_d	f = 100 Hz ε_d	f = 200 Hz ε_d
Pulse Measurement	Deadband	0.0033	0.034	0.071
First-Order Low-Pass Filter	$\frac{5}{\alpha}$	4.0	0.27	0.13
Second-Order Low-Pass Filter	$\frac{7.64}{\alpha}$	1.8	0.047	0.012
FIR Filter	Δ	1	0.09	0.04
Fixed Sampling Window	T_w	1	0.1	0.05

asynchronously from one another would give an aggregate response that filters out the bumpy response of any one nerve. This method works well in the human nervous system, but it does not work as well if single PFM signals are being demodulated.

It should be noted that every PFD method in this chapter is evaluated with a noise-free pulse stream. Here, noise would manifest itself as extra pulses. The period-measurement PFD method appears to be the method that gives the lowest errors, but this method is the most sensitive to extra pulses, which are demodulated as a very high frequency for a very short duration. In practice, the period-measurement PFD method would have to be followed by a filter. The low-pass filtering PFD methods are relatively insensitive to extra pulses. This noise consideration is discussed further in Chapter 6.

The difference in behavior between the first-order low-pass filter and the second-order low-pass filter is very large. By continuing to concatenate unity-gain first-order filters, it may be possible to create a high-order low-pass filter demodulator with very desirable characteristics.

4. GRAPHICAL LIMIT-CYCLE PREDICTION

The describing function is a common way to predict the existence of limit cycles in a nonlinear system. The describing function of a nonlinearity is its equivalent gain and phase-lag to a sinusoidal input of amplitude A, where the gain and phase-lag are a function of A.

Pulse frequency modulators and demodulators are time-varying, meaning they are not only amplitude dependent, but also frequency dependent. It would be desirable to be able to predict the existence of limit cycles in systems containing pulse frequency modulation. Developing a describing function for PFM systems has been attempted before with limited success. Dymkov [10] attempted a describing function for a double-signed IPFM, and Li and Jones [7] attempted a describing function for PPSSPFMD. The results of these are so mathematically complicated that they cannot be easily applied to any real application where pulse frequency modulators and demodulators are just a part of a larger system.

4.1 Description of Graphical Method (Simple Loop)

Consider the simple loop of Fig. 29. This loop contains a nonlinear describing function that is dependent on the amplitude and frequency of its input sinusoid, in series with a linear element, connected with negative unity feedback. To look for the existence of a self-sustained limit cycle, it is standard practice to set the input of the system to zero

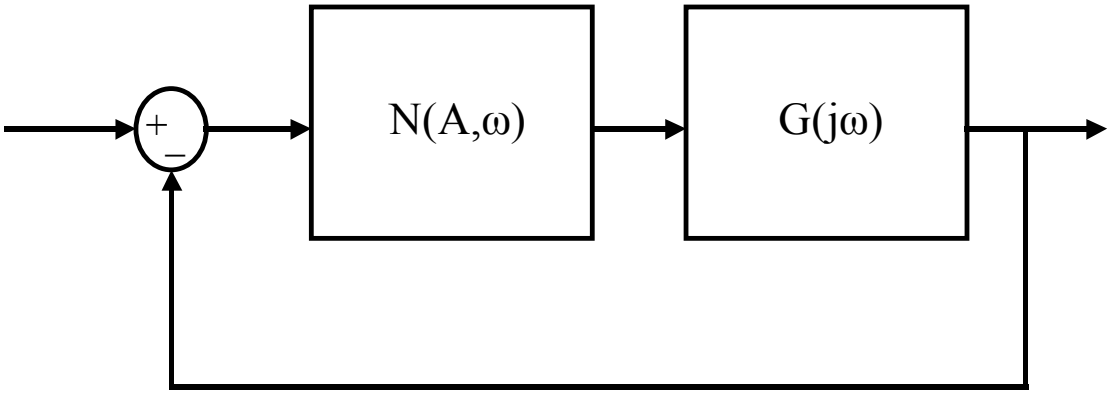


Fig. 29 Simple Loop with Frequency-Dependent Describing Function

and then disturb the system. From describing function analysis [11], a limit cycle will occur if:

$$G(j\omega) = -\frac{1}{N(A,\omega)} \tag{73}$$

For a describing function that is only a function of amplitude, limit-cycle prediction involves finding the intersection of two loci of points – that of the Nyquist plot of the linear plant *G*, and that of the negative inverse of the describing function *N*. The amplitude of the limit cycle is taken from the describing function at the intersection, and the frequency of the limit cycle is taken from the Nyquist plot of the linear plant at the intersection.

Because the describing function in Fig. 29 is a function of amplitude and frequency, an intersection of the Nyquist plot of the linear plant and the negative inverse describing function will indicate a limit cycle only if the frequency taken from the describing function matches the frequency of the Nyquist plot at the point of intersection.

Figure 30 shows a possible way to think about this frequency-dependent describing function analysis. The Nyquist plot of the linear plant is plotted as before. Remember that the Nyquist plot is a locus of points, where each point represents the gain and phase of the plant for a sinusoidal input of a given frequency, but the amplitude of the sinusoid does not matter. For the describing function, each constant-frequency negative inverse describing function is plotted as a locus of points. Each of these constant-frequency loci acts as an independent negative inverse describing function that is only dependent on amplitude. There are infinitely many of these constant-frequency loci, because there are infinitely many frequencies, but practically only some *n* loci need be plotted based on the resolution needed and the bandwidth of the system. A limit cycle can occur only if the negative inverse describing function for a constant frequency ω_n crosses the Nyquist plot of the linear plant at the point representing the frequency ω_n. If

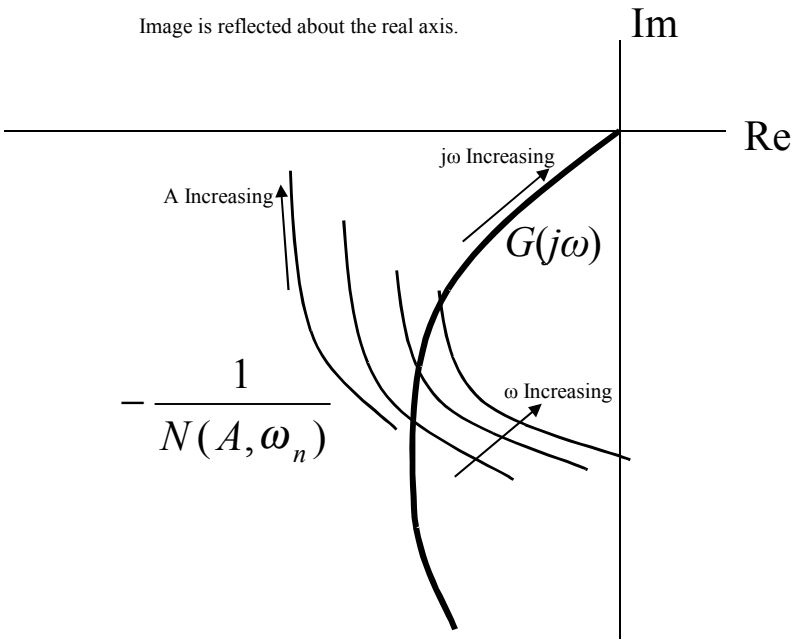


Fig. 30 Nyquist Plot for Frequency-Dependent Describing Function

the intersection does not have a common frequency, the limit cycle will not sustain itself. If there is a common-frequency intersection, this frequency is the frequency of the limit cycle, and the amplitude of the limit cycle is taken from the negative inverse describing function locus for the intersection frequency.

If no intersection that would create a self-sustaining limit-cycle is possible, system stability can be determined. If every point of the negative inverse describing function loci are encircled by the Nyquist plot of the linear plant, the system is unstable. If no points are encircled, the system is stable.

This path of reasoning can now be applied to pulse frequency modulated systems; the nonlinearity just needs to be explicitly defined as a pulse frequency modulation/demodulation pair. The input to the modulator is the continuous output of the plant, multiplied by a gain of –1. The output of the demodulator is the reconstructed signal that may or may not represent the modulator input well after going through PFM and PFD. This reconstructed signal will be a delayed, discretized, and distorted version of its former self. To apply the amplitude and frequency dependent describing function method shown in Fig. 30, the equivalent gain and phase-lag of the PFM/PFD pair is developed in the next section.

4.2 Tabular Describing Function with Post-Filtering Method

To apply the graphical limit cycle prediction method described of Section 4.1 to systems containing pulse frequency modulation, it is first necessary to determine which type of PFM and PFD methods to use. Because the ultimate goal of this thesis is to develop tools for use with the control of the Experimental Neural Arm, the PPSSPFMD setup of Section 2.6 will be used here. Figure 29 becomes Fig. 31 when PPSSPFMD is

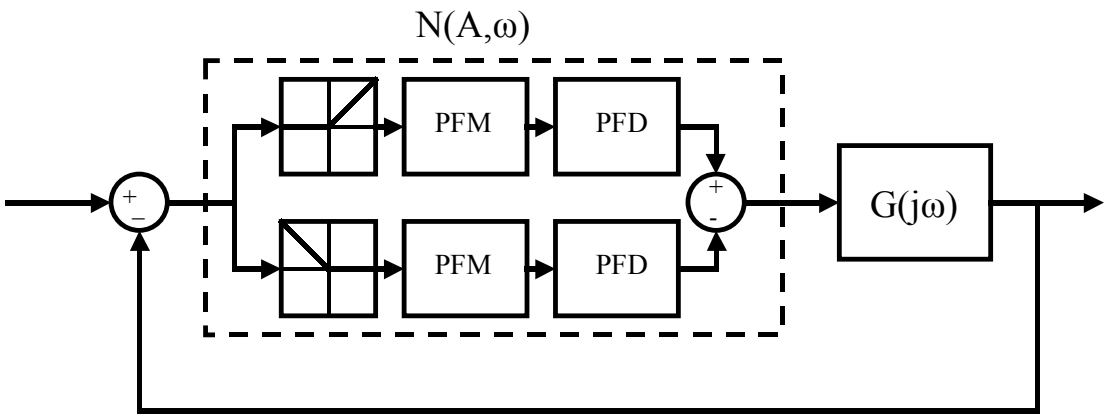


Fig. 31 Simple Loop with PPSSPFMD Describing Function

used. The conclusion of Section 2.4 is that all integration methods that have a linear input-to-frequency relationship are equivalent. For this reason, the USSH method was chosen here, because its delay at startup is halfway between the IPFM method and the V/F converter. Remember, though, that this start up behavior is only seen at start up with the single-signed PFM being used here, and that the other two PFM methods would have worked just as well. The conclusion of Section 3.4 is that period measurement is probably the best overall PFD method, though this is a debatable point. Its low errors at low frequencies should help prevent jitter, and its small time delays at high frequencies should help with tracking and stability.

Now that a model has been chosen, the equivalent describing function of the PPSSPFMD nonlinearity can be found using a method that will be referred to as the post-filtering method. The post-filtering setup is shown in Fig. 32. The PPSSPFMD setup modulates and demodulates an input sinusoid x of amplitude A and frequency ω (rad/sec):

$$x = A\sin(\omega t) \tag{74}$$

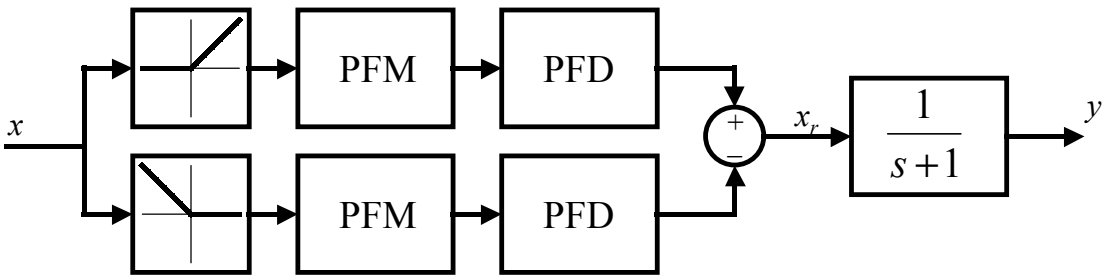


Fig. 32 Post-Filtering Method

The input to the low-pass filter is a reconstructed version of the input to the PPSSPFMD x_r, and the output of the filter y can be approximated as a sinusoid with a new amplitude and phase-lag:

$$y = A_y \sin(\omega t - \phi) \tag{75}$$

For a given sinusoidal frequency, the gain k_F and the phase-lag ϕ_F of the filter are known quantities. Because they are known, the equivalent gain and phase-lag of the PPSSPFMD nonlinearity defined above can be calculated with the equations:

$$k_N = \frac{A_y}{A k_F} \tag{76}$$

$$\phi_N = \phi - \phi_F \tag{77}$$

Because the input to the filter is not purely sinusoidal, the output is not either. To find the best sinusoidal approximation of the output, a three-step curve fit is employed. First, the amplitude of the output is approximated by its highest peak. Second, using the amplitude approximation, the best choice for ϕ is found with a least-squares method. Third, using the phase-lag ϕ, the best choice for A is found by matching the RMS values

of the output and the sinusoidal approximations. This three-step curve fit should give a good sinusoidal approximation to a quasi-sinusoidal signal. The Simulink post-filtering setup and associated MATLAB scripts are given in Appendix C.

This post-filtering method is only valid when the plant acts as a low-pass filter, which is a valid assumption for mechanical plants. This is a limitation of the method, but it is the same limitation found in traditional describing function analysis, where only the lowest-frequency component of a Fourier analysis is considered.

Using the post-filtering method, the equivalent gain and phase-lag (in degrees) of the PPSSPFMD were found, and are listed in Table 3 and Table 4, respectively. The data were taken at various frequencies, and at various values of input amplitude multiplied by modulation constant Ak_M (it can easily be seen that it is the combined value of these two variables that matters, not either of them alone). The "NaN"s in the upper-right corner of the tables stand for "not a number" (in MATLAB form), and they represent a case where the output of the PPSSPFMD nonlinearity is not periodic, so no limit cycle is possible at that combination of amplitude, modulation constant, and frequency. This characteristic is expected with a combination of low amplitude and high frequency, considering the integrating nature of the pulse frequency modulators.

The data in Table 3 and Table 4 could be fit into a describing-function equation, but they can easily be used in tabular form by interpolating between the table values. The columns of Table 3 and Table 4, along with interpolated "columns" which represent frequencies between the data taken, can be plotted as separate loci of points to create a plot like that of Fig. 30. More data could be taken, if it was deemed necessary, at higher or lower frequencies, or at higher values of Ak_M. No data need be taken for lower values

Table 3 PPSSPFMD Equivalent Gain

Amplitude X Modulation Constant	1 rad/sec	5 rad/sec	10 rad/sec	15 rad/sec	20 rad/sec	25 rad/sec	30 rad/sec	35 rad/sec	40 rad/sec
13	0.79	0.76	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	0.81	0.84	0.86	0.8	NaN	NaN	NaN	NaN	NaN
15	0.87	0.94	0.9	0.84	NaN	NaN	NaN	NaN	NaN
20	0.95	0.98	0.99	1.04	1.29	NaN	NaN	NaN	NaN
25	0.96	0.99	1.03	1.08	1.04	1.16	NaN	NaN	NaN
30	0.99	1	1.04	1.12	1.16	1.18	1.23	NaN	NaN
40	0.99	1	1.04	1.11	1.13	1.16	1.14	1.12	1.14
60	1	1	1.03	1.06	1.04	0.99	0.96	0.94	0.89
80	1	1	1.02	1	0.97	0.92	0.86	0.85	0.76
120	1	1	1	0.97	0.89	0.84	0.72	0.82	0.86
180	1	1	0.99	0.95	0.85	0.74	0.65	0.55	0.55
240	0.99	0.99	0.98	0.93	0.83	0.78	0.69	0.63	0.61
300	0.99	0.99	0.98	0.92	0.83	0.72	0.64	0.55	0.43
360	0.99	0.98	0.95	0.9	0.81	0.75	0.6	0.77	0.59
420	0.98	0.97	0.94	0.89	0.8	0.74	0.59	0.51	0.63
500	0.94	0.94	0.91	0.86	0.78	0.68	0.62	0.6	0.72
600	0.8	0.8	0.77	0.73	0.67	0.56	0.47	0.51	0.6
800	0.62	0.61	0.59	0.55	0.49	0.45	0.36	0.48	0.43
1000	0.53	0.53	0.51	0.47	0.43	0.36	0.38	0.49	0.51

Table 4 PPSSPFMD Equivalent Phase-Lag (degrees)

Amplitude X Modulation Constant	1 rad/sec	5 rad/sec	10 rad/sec	15 rad/sec	20 rad/sec	25 rad/sec	30 rad/sec	35 rad/sec	40 rad/sec
13	6	28	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	6	27	54	70	NaN	NaN	NaN	NaN	NaN
15	5	27	51	74	NaN	NaN	NaN	NaN	NaN
20	5	24	45	72	87	NaN	NaN	NaN	NaN
25	5	21	42	63	85	103	NaN	NaN	NaN
30	5	20	39	61	81	96	118	NaN	NaN
40	5	17	36	55	75	91	111	131	148
60	4	14	30	49	66	83	102	117	140
80	4	11	26	43	59	74	96	107	133
120	3	9	21	36	48	61	77	86	101
180	3	6	17	31	38	47	58	58	52
240	3	6	15	26	32	39	45	45	32
300	2	6	13	24	26	33	35	33	24
360	2	5	12	19	25	30	31	21	20
420	2	5	11	18	23	29	27	20	17
500	2	4	10	17	21	25	24	18	15
600	1	5	11	17	22	25	25	18	15
800	1	5	11	18	25	28	25	15	16
1000	0	5	12	18	24	28	26	14	16

of Ak_M at frequencies where a "NaN" is given; this would just result in more "NaN"s.

The data could also be retaken with a different post-filter to try to account for the bandwidth of the linear plant being used; this would create a need for three-dimensional interpolation (Ak_M, frequency, bandwidth). A few simulations were run with a unity-gain low-pass post-filter with a bandwidth of 50 rad/sec, and the equivalent gain and phase-lag of the PPSSPFMD nonlinearity changed very little, especially considering the intention of the data is for use with a graphical method, where great precision is not needed.

The PPSSPFMD nonlinearity considered in the section used USSH as its PFM method, and period measurement with a 10-Hz deadband as its PFD method, so the tabular describing function data of Table 3 and Table 4 relate directly to this setup. No other types of PFM or PFD will be considered here, but the post-filter method could be used to recreate Table 3 and Table 4 using any combination of modulator and demodulator.

4.3 Simple-Loop Examples with Simulation Comparisons

An algorithm can be used with the data in Table 3 and Table 4 to graphically predict limit cycles in the simple loop of Fig. 31. The algorithm is performed by the MATLAB script "Limit_Predictor.m," found in Appendix D. In the MATLAB code, the linear plant is defined as "SYS." To best understand the algorithm used for limit-cycle prediction, consider an example; see Fig. 33, which is the plot created by "Limit_Predictor.m." A second-order linear plant with two poles at –10 rad/sec and a DC gain of 5 is used in this example, and the modulation constant is 20. The plant was chosen because it is a simple system with a bandwidth near that of the Experimental Neural Arm. The choice of modulation constant was arbitrary.

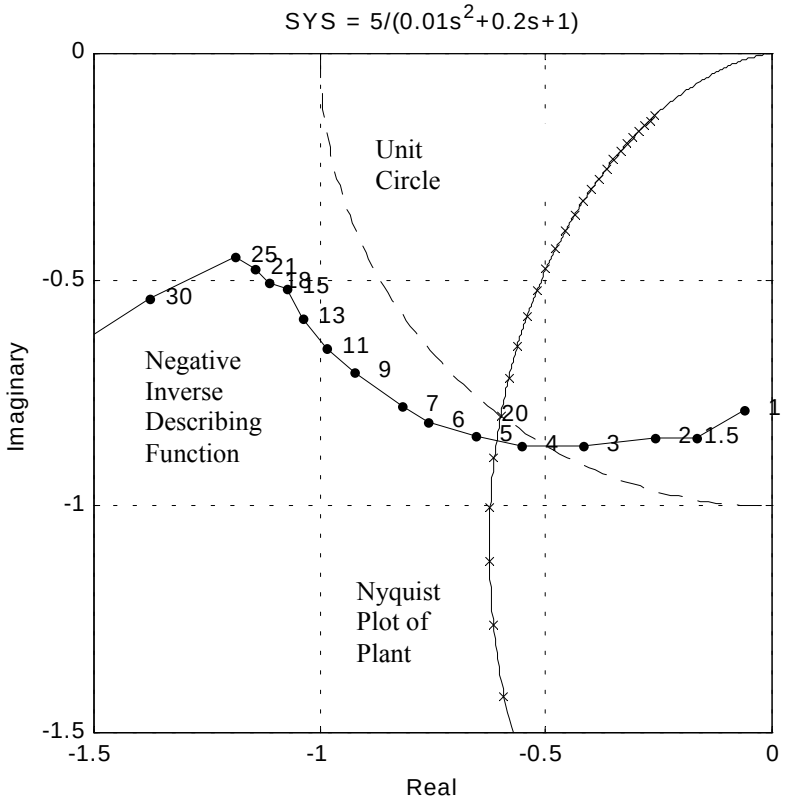


Fig. 33 Graphical Limit-Cycle Predictor

The solid line with "x" markers is the Nyquist plot of the linear plant, and each "x" marks an integer frequency value; remember the frequency increases as the Nyquist plot goes toward the origin. The dashed line is the unit circle. Because the gains in Table 3 are centered around 1 (ideally they would be 1), the algorithm uses the integer frequency on the Nyquist plot that is closest to the unit circle as a first guess at the limit-cycle frequency. This frequency is also labeled for the duration of the algorithm; notice on the plot in Fig. 33 that a "20" is next to the closest "x" to the unit circle, indicating a frequency of 20 rad/sec. To determine the values of the other "x"s, simply count away from 20 by integer values. The remaining solid line is the amplitude-dependent negative inverse describing function of the PPSSPFMD nonlinearity if only limit cycles of 20

rad/sec are considered. The numbers to the right of the dot markers on this line represent different limit-cycle amplitudes (the values of Ak_M used from the table are divided by k_M before being printed to the screen).

If the negative inverse describing function crossed the Nyquist plot at 20 rad/sec, it could be concluded that a limit-cycle would occur with a frequency of 20 rad/sec, and with an amplitude obtained from describing function locus. The describing function does not cross at 20 rad/sec though, and an iteration is required. The software prompts for a new frequency, and 19.5 is entered because the Nyquist plot is crossed between the "x"s representing 19 and 20 rad/sec. The software now plots the negative inverse describing function plot if only limit cycles with a 19.5-rad/sec frequency are considered. This plot still crosses the Nyquist plot at 19.5 rad/sec, so it can now be concluded that a limit cycle will occur with a frequency of 19.5 rad/sec and an amplitude of 4.5, which was taken from the describing function locus. This example took only one iteration to complete, and most problems are just as simple.

To check the validity of these results, compare the graphical limit-cycle prediction with a Simulink simulation of the system, which is given as Fig. 34. Because pulse frequency modulation and demodulation are time-varying, no true limit cycle is ever achieved; the magnitudes of the peaks vary, and the instantaneous frequency varies also. Pavlidis and Jury [2] described this behavior as a "limit annulus," because a phase-plane plot of the plant output and its time derivative shows a response that always falls within a distorted doughnut shape (not the true definition of an annulus), never growing too large or decaying too small, but also never falling into a periodic orbit. Figure 35 shows the same data in Fig. 34, but in phase-plane form, illustrating the "limit annulus." This being

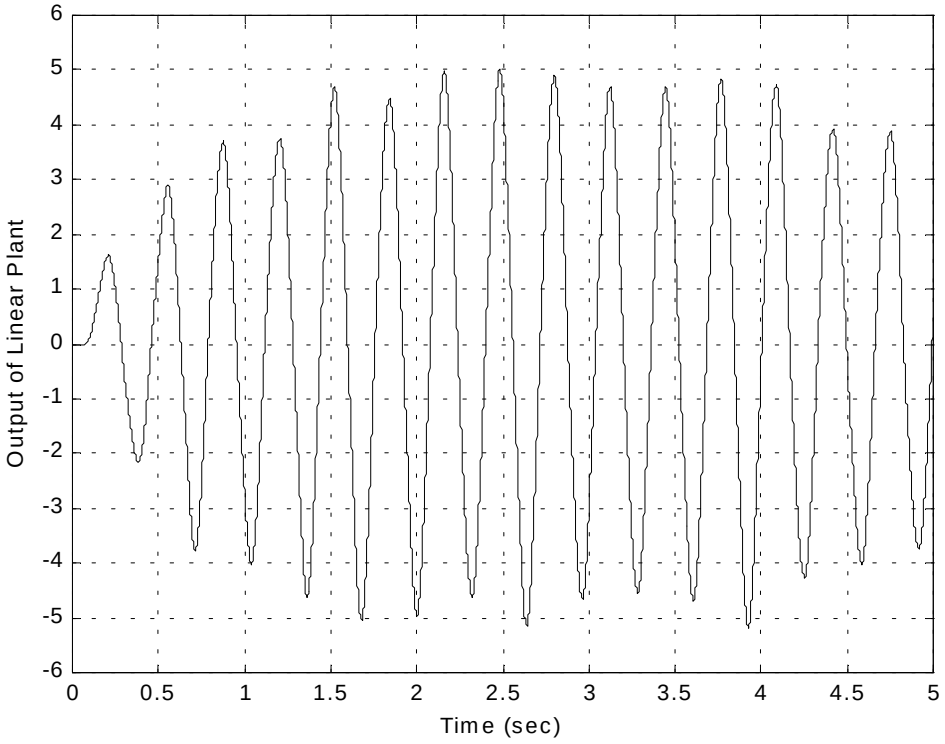


Fig. 34 Limit Cycle in Simulation of Simple Loop

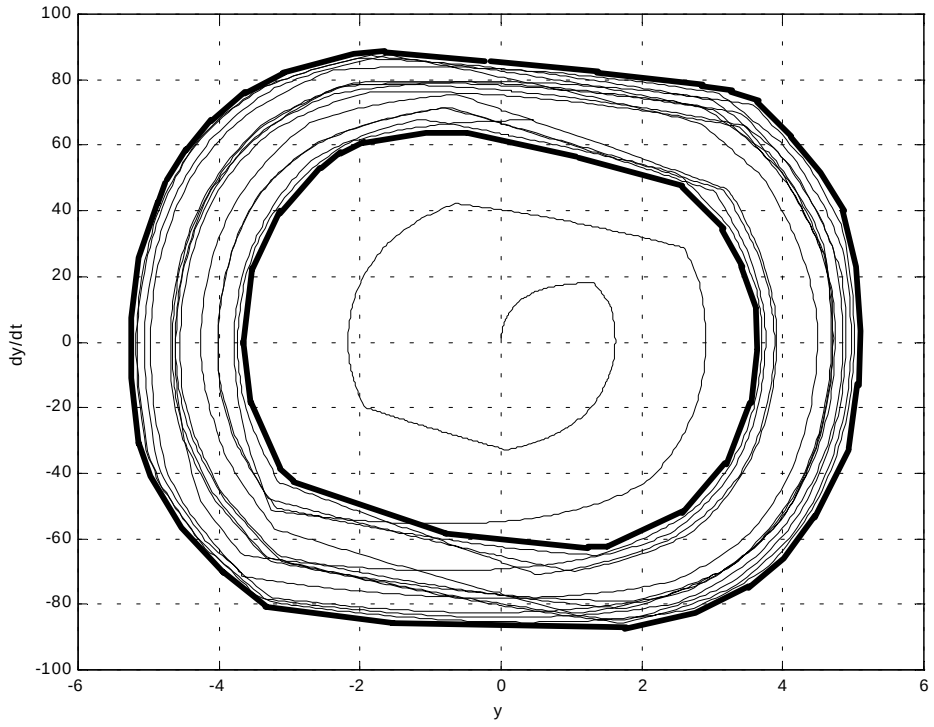


Fig. 35 Phase-Plane Plot Showing Limit Annulus (In Bold)

said, the limit-cycle amplitude and frequency obtained from the graphical predictor are only approximations of the behavior of the output. For this example, the graphical predictor predicted an amplitude of 4.5, and the simulation shows peaks in the range of about 4-5. The graphical predictor predicts a frequency of 19.5 rad/sec, and the simulation shows frequecies between about 19 and 20 rad/sec. For this example, the graphical predictor was nearly perfect, but this is not always the case; the prediction is accurate enough, though, to understand the behavior of the system being analyzed.

Table 5 contains comparisons of predicted limit-cycle amplitudes and frequencies with simulated limit-cycle amplitudes and frequencies. For this simple loop, the graphical limit-cycle predictor is very accurate.

Table 5 Comparison of Graphically-Predicted and Simulated Limit Cycles for Simple Loop ($k_M = 20$)

SYS	Simulation Amplitude	Simulation Frequency (rad/sec)	Predicted Amplitude	Predicted Frequency (rad/sec)
$\frac{3.5}{0.01s^2+0.2s+1}$	2	16	2.2	16.9
$\frac{5}{0.01s^2+0.2s+1}$	4.5	19.5	4.5	19.5
$\frac{15}{0.01s^2+0.2s+1}$	14	29	14.3	29.8
$\frac{100}{s^2+11s+10}$	None	None	None	None
$\frac{200}{s^2+11s+10}$	2.5	13	2.8	12.9
$\frac{400}{s^2+11s+10}$	8.5	17	10	17.6

4.4 Limit-Cycle Prediction Algorithm for Complex Loops

The tabular describing function was validated in the previous section as an accurate depiction of a PPSSPFMD nonlinearity, but the simple loop that was analyzed is unrealistically simple for use with any real applications, and was only used to develop the method. Consider the feedback loop of Fig. 36; this loop is more complex than that of Fig. 31, and more suited to real applications. This loop has a forward-path PPSSPFMD nonlinearity and a linear plant $G(j\omega)$ like the simple loop, but it also contains a feedback PPSSPFMD nonlinearity and a linear transfer function $H(j\omega)$, the input of which is the error of the system. The form of this loop is not important; the algorithm developed here is applicable to even more complex loops. The algorithm will be given explicitly for the loop of Fig. 36, but the method is easily extrapolated. The MATLAB script implementing this algorithm is "Limit_Predictor2.m," given in Appendix D.

Like before, the Nyquist plot of the linear plant $G(j\omega)$ is plotted, but now the line that previously represented the negative inverse describing function of the PPSSPFMD represents the negative inverse describing function of the concatenation of all the remaining elements in the loop. The algorithm iterates through one frequency at a time, with the initial guess chosen as before. Regardless of the elements in the loop, the

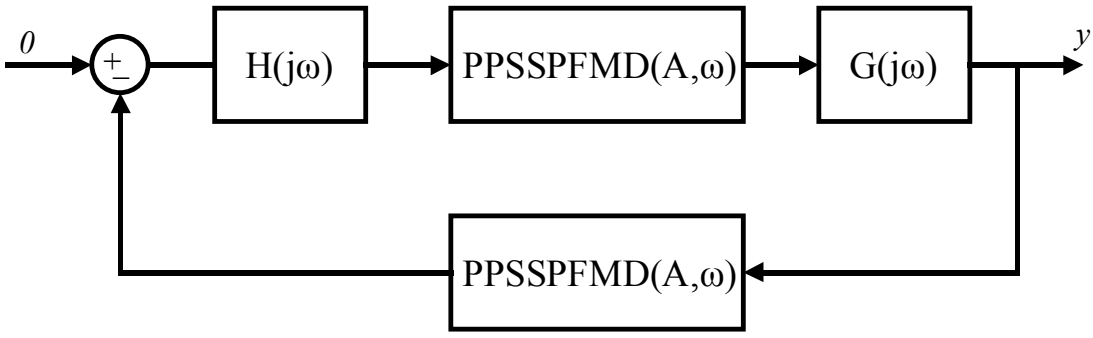


Fig. 36 Complicated Loop with PPSSPFMD Describing Functions

algorithm starts at the output y of the linear plant $G(j\omega)$, and continues around the loop in the clockwise direction. Assume the output y has the form:

$$y = A\sin(\omega t) \tag{78}$$

The gain K_{fb} and the phase-lag ϕ_{fb} of the feedback PPSSPFMD are found from Table 3 and Table 4 for the amplitude-times-modulation-constant Ak_M and the frequency ω. Now, the gain K_H and the phase-lag ϕ_H of the linear element $H(j\omega)$ are found for the frequency ω. The output of the linear plant $H(j\omega)$ has the form (approximately):

$$u = AK_{fb}K_H \sin(\omega t - \phi_{fb} - \phi_H) \tag{79}$$

The gain K_f and the phase-lag ϕ_f of the forward-path PPSSPFMD are now found from Table 3 and Table 4 for the amplitude-times-modulation-constant $AK_{fb}K_Hk_M$ and the frequency ω. The concatenated describing function has a total gain K_{total} and a total phase-lag ϕ_{total} of:

$$K_{total} = K_{fb}K_HK_f \tag{80}$$

$$\phi_{total} = \phi_{fb} + \phi_H + \phi_f \tag{81}$$

The final step is to plot a point at a distance $1/K_{total}$ away from the origin and ϕ_{total} degrees away from the real axis, and label the point with the amplitude A. This is repeated for different values of A to create the concatenated-negative-inverse-describing-function locus.

Notice that the negative feedback is never explicitly considered. It is implicit in the method that a single gain of –1 occur somewhere in between two of the concatenated elements.

4.5 Complex-Loop Examples with Simulation Comparisons

The algorithm for complex loops presented in Section 4.4 is validated here by comparing results of Simulink simulations to the graphical limit-cycle predictor. The algorithm is performed by the MATLAB script "Limit_Predictor2.m," found in Appendix D. In the MATLAB code, the linear transfer function and linear plants $H(j\omega)$ and $G(j\omega)$ of Fig. 36 are defined as "SYS1" and "SYS2", respectively.

For an example, consider a situation where "SYS1" is defined as 200/(s + 200) in Laplace form, and "SYS2" is defined as $400/(s^2 + 11s + 10)$. "SYS2" was chosen because it is a simple system that has approximately the same bandwidth of the Experimental Neural Arm. "SYS1" was arbitrarily chosen as a simple filter. After iterating through different frequency values, following the same method described in Section 4.3, the graphical limit-cycle predictor code "Limit_Predictor2.m" results in the plot given as Fig. 37. The limit-cycle predictor predicts a limit-cycle frequency of approximately 14.7 rad/sec, and a limit-cycle amplitude of approximately 30. Figure 38 shows the Simulink simulation of the same system. The simulation shows limit-cycle peaks occurring mostly in the range of 28-32, but occasionally dipping down as low as 24. This matches the prediction of 30 well. The simulation shows a limit-cycle frequency of approximately 14 rad/sec; this also matches the prediction of 14.7 rad/sec well.

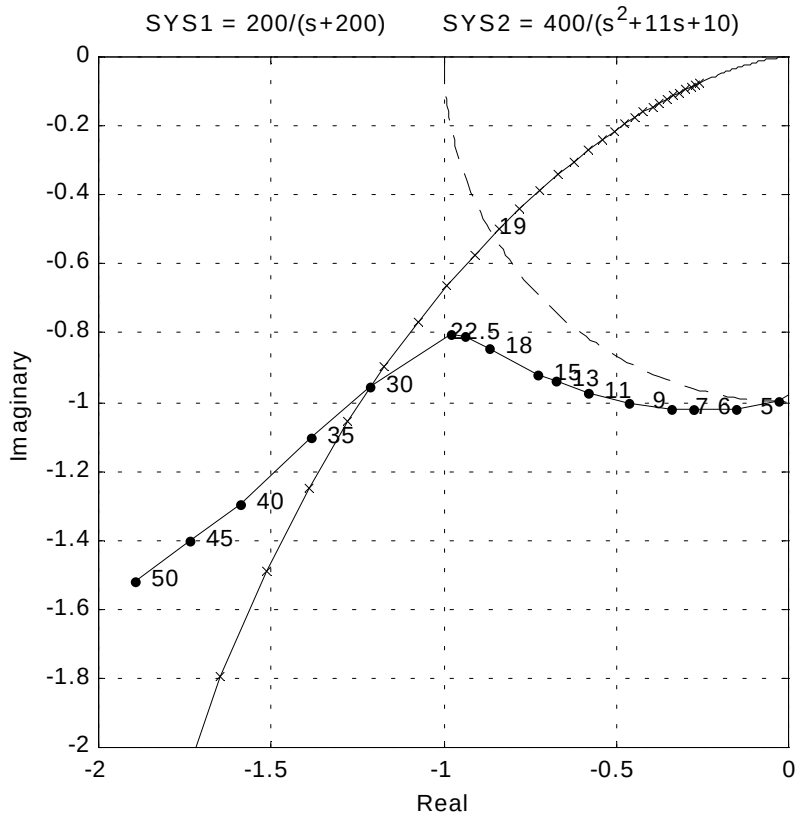


Fig. 37 Graphical Limit-Cycle Predictor for Complex Loop

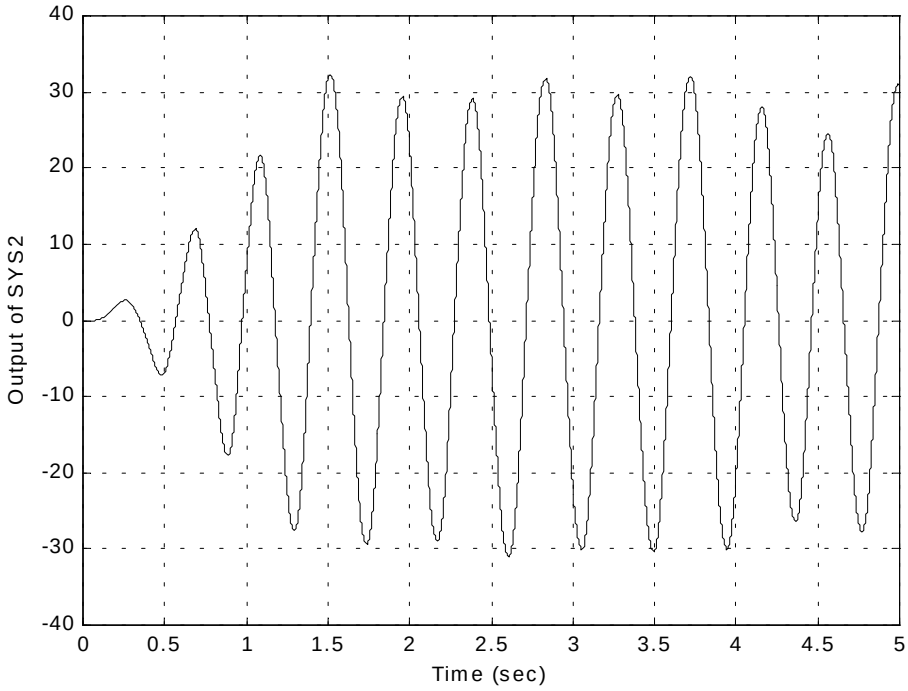


Fig. 38 Limit Cycle in Simulation of Complicated Loop

Table 6 contains comparisons of predicted limit-cycle amplitudes and frequencies with simulated limit-cycle amplitudes and frequencies. For this complex loop, the graphical limit-cycle predictor is still very accurate.

Table 6 Comparison of Graphically-Predicted and Simulated Limit Cycles for Complex Loop ($k_M = 20$)

SYS1 SYS2	Simulation Amplitude	Simulation Frequency (rad/sec)	Predicted Amplitude	Predicted Frequency (rad/sec)
$\frac{200}{s+200}$ $\frac{100}{s^2+11s+10}$	1.7	8	2.1	8.1
$\frac{200}{s+200}$ $\frac{200}{s^2+11s+10}$	8	11	10.7	11.9
$\frac{200}{s+200}$ $\frac{400}{s^2+11s+10}$	30	14	30	14.7
$\frac{5}{s+5}$ $\frac{50}{s^2+11s+10}$	None	None	None	None
$\frac{5}{s+5}$ $\frac{100}{s^2+11s+10}$	7.7	6	7	5.7
$\frac{5}{s+5}$ $\frac{200}{s^2+11s+10}$	40	6	38	6.4

5. GRAPHICAL LIMIT-CYCLE PREDICTION WITH EXPERIMENTAL NEURAL ARM WRIST

A graphical limit-cycle predictor method was developed in Chapter 4, but it was verified only with Simulink models. Simulink models can be very informative regarding basic system behaviors, but they still fail to fully embrace the complexity of a real system. The model used will necessarily have errors that are unaccounted for. One problem that occurs when pulse frequency modulating and demodulating a signal in a Simulink model is that everything occurs on a common computer sample, which synchronizes a normally asychronous PFM signal with its demodulator. This eliminates the demodulation error of Section 3.1.2, but does account for the modulation error of Section 2.5. This is not an accurate depiction of real systems containing PFM elements.

It is very desirable to validate the graphical limit-cycle predictor method with a real system. Because the impetus of this thesis is to create control system design tools for use with the Experimental Neural Arm, the method will be validated with the wrist of the Experimental Neural Arm. The Experimental Neural Arm is shown as Fig. 39.

A wrist model, in the form needed for the limit-cycle prediction algorithm, is developed here. The experimental setup is explained. The graphical limit-cycle predictor works well for many experiments, but there are cases when the prediction loses accuracy which will be discussed.

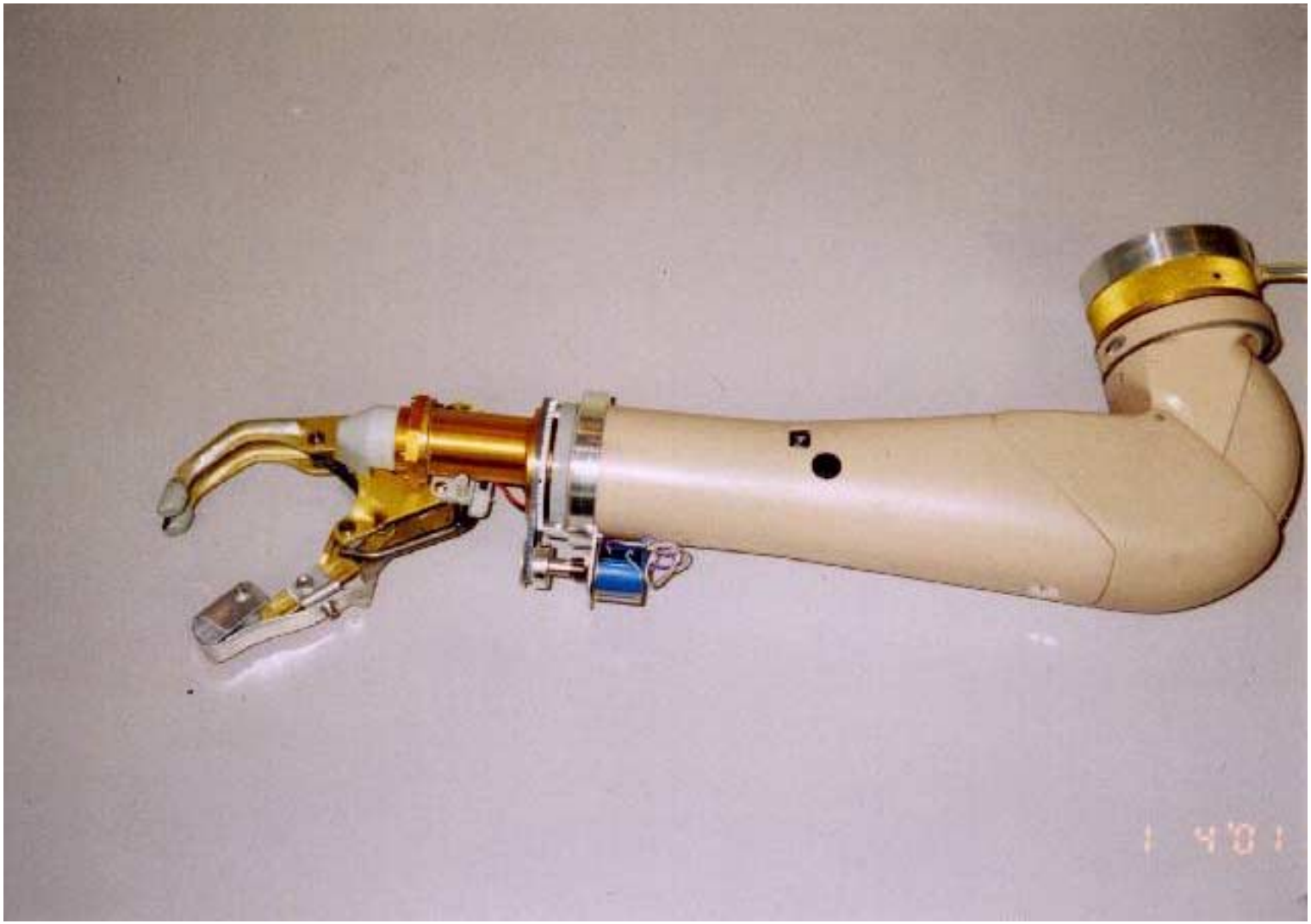


Fig. 39 Experimental Neural Arm

5.1 Human/Arm System Model

The loop of Fig. 36 may still be too simplistic for real systems, but as was stated in Section 4.4, the graphical-limit-cycle algorithm is easily expanded to more complicated loops. Consider the system of Fig. 40, which is a possible model of the Experimental Neural Arm connected to an amputee. This loop contains afferent and efferent time delays, plus a linear compensator $C(j\omega)$. Here the transfer function $H(j\omega)$ represents a model of the brain's function; Gossett et al. [12] use a PI controller to model the brain in a similar loop. There are multiple ways to use the graphical limit-cycle predictor with this system. One possible method would lump the controller and plant into one transfer function for the Nyquist plot, and would lump the two time delays with the transfer function $H(j\omega)$, using Pade approximants [13] for the time delays. Another possible method would plot the Nyquist plot of the plant $G(j\omega)$, and every other element in the loop would be lumped with the concatenated describing function with the algorithm described in Section 4.4. It should be noted though, that a time delay of Δ seconds can be written as a pure phase lag of $\omega\Delta$ radians [13]. Because the limit-cycle-predictor algorithm turns every element in the loop into an equivalent gain and phase-lag, a time delay is very easily handled with the algorithm without the need for a Pade

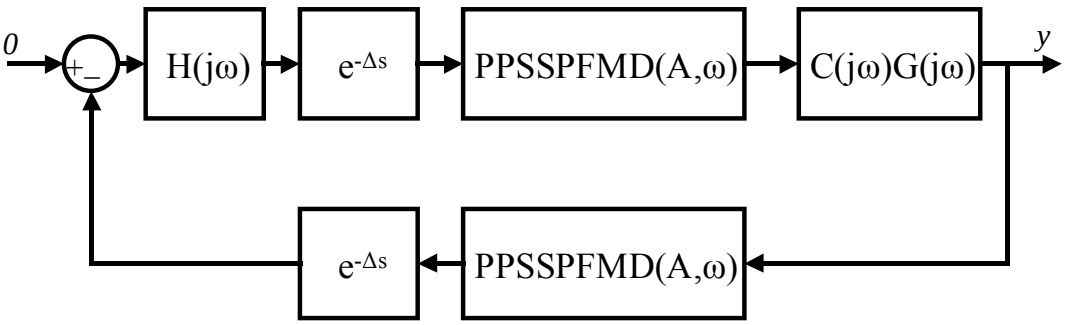


Fig. 40 Model of Human Connected to Experimental Neural Arm

approximant.

5.2 Wrist Model

To use the graphical limit-cycle predictor algorithm of Section 4.4, there is great freedom regarding the elements in the loop, but one constraint is that a linear plant model is needed. It is this linear plant of which the Nyquist plot is plotted in the algorithm. The model of the plant may contain nonlinearities, but they must be given in series with some linear model; the nonlinearities can then be included in the concatenated describing function for the loop. Many common nonlinearities can be modeled in this way, including saturations and deadbands.

This being said, a model of Experimental Neural Arm wrist is needed. The wrist model must take the form of a linear transfer function that may be preceded, in series, by nonlinearities of any form.

Some wrist properties from Fukuyama [14] are used here without independent validation. The wrist has a deadband do to stick-slip friction, whereby no voltage input with a magnitude less than 0.24 volts causes any movement in the wrist. Also, the model of the wrist should have a voltage-to-position transfer function with two poles (one negative and one at the origin) and no zeros.

To obtain a transfer function of the wrist, a Hewlett-Packard 3562A Dynamic Signal Analyzer was used. This signal analyzer drives the plant being modeled by a sinusoid of adjustable amplitude (volts) and frequency (Hertz). The output of the plant is fed back to the signal analyzer (volts). The instrument sweeps through many frequencies, determining the frequency-dependant gains of the plant for a preset range of frequencies.

The signal analyzer fits a transfer function to the data taken based on user-determined number of poles and zeros.

Attempts at obtaining an open-loop model were unsuccessful; the nonsymmetrical properties of the wrist caused it to drift towards its mechanical stops during open-loop testing. For this reason, the closed-loop model of the wrist was found, and the open-loop wrist model was derived from it. The form of the open- and closed-loop models are shown as Fig. 41, where $W(j\omega)$ is the open-loop wrist model, $W_{CL}(j\omega)$ is the closed-loop wrist model, x is the driving input sinusoid (volts) to the closed-loop system, e is the error signal (volts) actually being applied to the wrist, and y is the wrist potentiometer output (volts). For linear plants, where the gain is only dependent on ω, deriving the open-loop model from the closed-loop model is perfectly valid. The deadband in the wrist creates an amplitude dependency; sinusoids of small amplitude are greatly affected by the

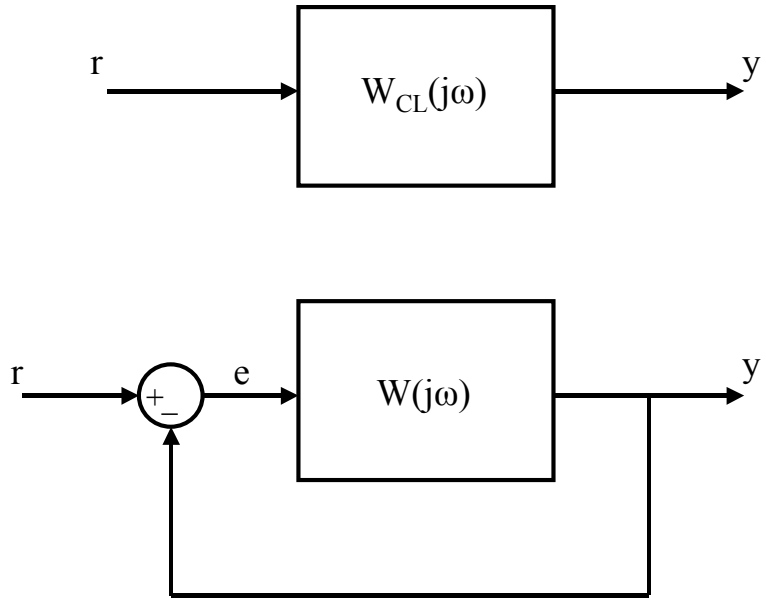


Fig. 41 Open- and Closed-Loop Wrist Models

deadband, where sinusoids of large amplitude are barely affected. The validity of the open-loop derivation is investigated later in this chapter.

$W(j\omega)$ and $W_{CL}(j\omega)$ are of the form:

$$W(j\omega) = \frac{K}{(s+\sigma_1)(s+\sigma_2)} \tag{82}$$

$$W_{CL}(j\omega) = \frac{K}{(s+\sigma_{CL1})(s+\sigma_{CL2})} = \frac{K}{s^2 + (\sigma_1+\sigma_2)s + (\sigma_1\sigma_2 + K)} \tag{83}$$

The closed-loop poles are found with the signal analyzer, and Eq. (83) is used to derive the open-loop poles. It is necessary to find the amplitude dependence of the wrist model. The A/D card currently running the Experimental Neural Arm has a saturation of ±0.9 volts, so no amplitude above that value will be considered (it will be accounted for with a saturation later). The deadband of ±0.24 volts means that no amplitude below this need be considered. Five different closed-loop models were obtained for five different amplitudes of the sinusoidal input r – 0.3, 0.4, 0.5, 0.7, and 0.9 volts:

$$W_{CL0.3}(j\omega) = \frac{0.0405}{(s+0.589)(s+0.238)} \tag{84}$$

$$W_{CL0.4}(j\omega) = \frac{0.137}{(s+0.877)(s+0.216)} \tag{85}$$

$$W_{CL0.5}(j\omega) = \frac{0.261}{(s+1.10)(s+0.203)} \tag{86}$$

$$W_{CL0.7}(j\omega) = \frac{0.263}{(s+1.17)(s+0.205)} \tag{87}$$

$$W_{CL0.9}(j\omega) = \frac{0.271}{(s+1.33)(s+0.185)} \tag{88}$$

These models obtained from the signal analyzer have the poles given in Hertz, but the DC gain is correct. After changing the poles to units of rad/sec, and changing *K* to keep the DC gain the same, the open-loop models are found using Eq. (83). The open-loop model for a 0.9-volt amplitude input is found to be:

$$W_{0.9}(j\omega) = \frac{10.7}{(s+9.61)(s-0.107)} \tag{89}$$

Both poles are nonzero, as is the case for all five open-loop models, but the known form of the wrist model is:

$$W(j\omega) = \frac{K}{s(s+\sigma)} \tag{90}$$

The form of the model in Eq. (89) is simply due to numerical errors in the signal analyzer. To force one of the poles to zero to match the form of the ideal open-loop wrist model, a root-locus equivalency method is used. Figure 42 shows how the two open-loop poles can be changed to one pole at the origin and one negative pole by matching the oscillatory portion of the root-locus. The amplitude-dependent wrist models in the form of Eq. (90), and with the pole in units of rad/sec, become:

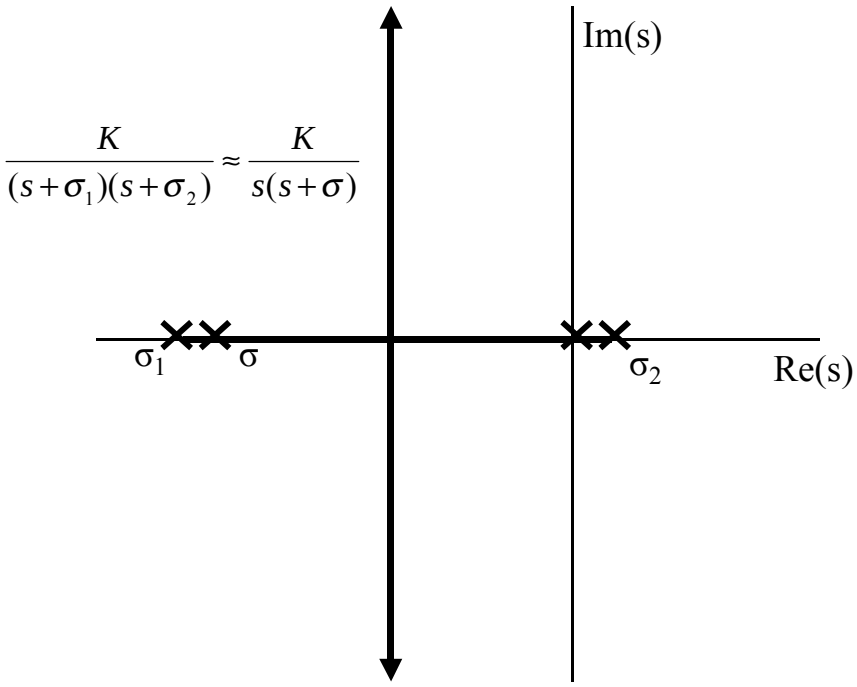


Fig. 42 Root-Locus Equivalency Method

$$W_{0.3}(j\omega) = \frac{1.60}{s(s+5.20)} \tag{91}$$

$$W_{0.4}(j\omega) = \frac{5.41}{s(s+6.87)} \tag{92}$$

$$W_{0.5}(j\omega) = \frac{10.3}{s(s+8.20)} \tag{93}$$

$$W_{0.7}(j\omega) = \frac{10.4}{s(s+8.65)} \tag{94}$$

$$W_{0.9}(j\omega) = \frac{10.7}{s(s+9.50)} \tag{95}$$

The model is very dependent on amplitude; for small amplitudes the gain of the wrist transfer function is appears effectively smaller, and the pole appears slower.

As was previously mentioned, deriving the open-loop wrist model from the closed-loop model is always valid if the plant is only dependent of ω, but the wrist is dependent on amplitude as well. Refer to Fig. 41; the input amplitudes used in the wrist models of Eqs. (91) through (95) are the amplitudes of r, but the amplitude of e is what is needed to develop an amplitude-dependent open-loop wrist model. If the amplitudes of r and e are nearly the same, deriving the amplitude-dependent open-loop model from the amplitude-dependent closed-loop model is valid; otherwise it is not. The transfer function between r and e is, in Laplace form:

$$e(s) = \frac{1}{1 + W(s)} r(s) \tag{96}$$

Figure 43 shows the Bode plot of the transfer function of Eq. (96). Plots are shown for the 0.3-volt and the 0.9-volt wrist models. These plots indicate that for frequencies greater than about 1 rad/sec r and e have the same magnitude, and the model is valid. If, when using these wrist models with the graphical limit-cycle predictor, the algorithm predicts a limit-cycle frequency less than 1 rad/sec, then the results could not be trusted.

For use with the limit-cycle predictor algorithm, the wrist needs to be modeled as a linear plant preceded by a nonlinearity. One method to incorporate Eqs. (91) through (95) into one model is to use the 0.9-volt wrist model (the model least affected by the

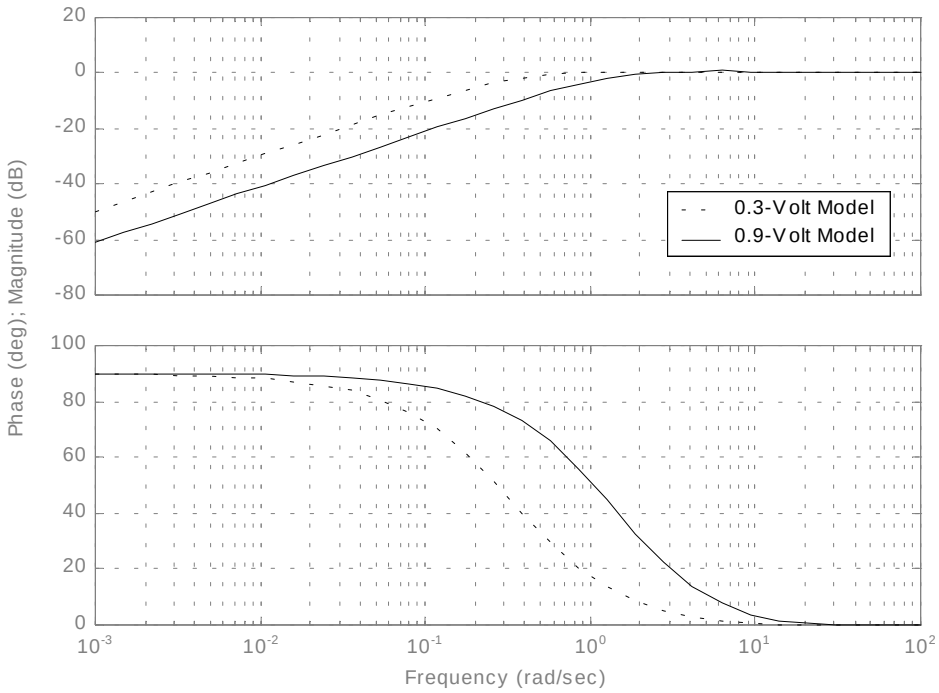


Fig. 43 Bode Plot to Check Validity of Derived Open-Loop Models

deadband) as the linear plant, which is preceded by an amplitude-dependent nonlinearity. The linear wrist, preceded by its amplitude-dependent nonlinearity, are:

$$\frac{k(A)(s+9.50)}{10.7(s+p(A))}\cdot\frac{10.7}{s(s+9.50)} \tag{97}$$

Here $k(A)$ and $p(A)$ are the amplitude-dependent gain and pole of the nonlinearity, respectively. The nonlinearity accounts for the relative effect of the deadband during oscillations of varying amplitudes. Figure 44 plots the gains (numerators) of Eqs. (91) through (95), along with a third-order polynomial curve fit, which is found to be:

$$k(A) = 108.4A^3 - 245.86A^2 + 183.81A - 34.668 \tag{98}$$

Figure 45 plots the poles of Eqs. (91) through (95), along with a third-order polynomial

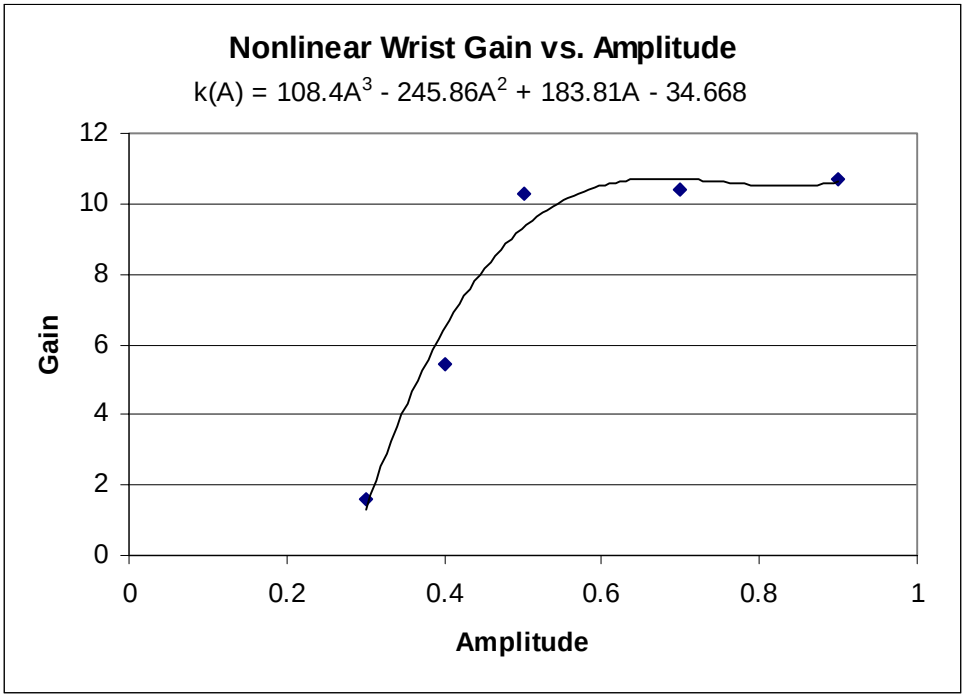


Fig. 44 Curve Fit of Wrist Model Gain

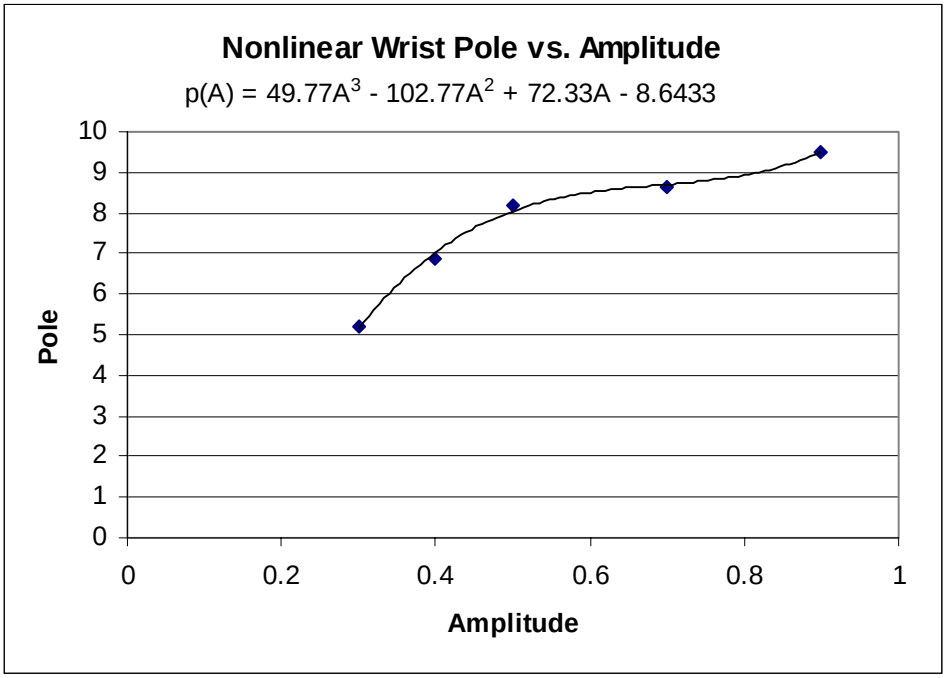


Fig. 45 Curve Fit of Wrist Model Pole

curve fit, which is found to be:

$$p(A) = 49.77A^3 - 102.77A^2 + 72.33A - 8.6433 \tag{99}$$

This amplitude-dependent nonlinearity can easily be incorporated into the graphical limit-cycle predictor algorithm.

The effects of the deadband in the wrist were present during all of the modeling, so no deadband describing function needs to precede the wrist model. No limit cycles can occur, though, that would create a sinusoidal input to the wrist model with an amplitude less than 0.24 volts; this needs to be accounted for in the algorithm. The saturation was never encountered because the highest amplitude input considered in modeling was not bigger than 0.9 volts. A hard-saturation describing function will need to precede the wrist model in the algorithm. This hard-saturation describing function, from Khalil [11], is:

$$A \le 0.9 \Rightarrow \Psi(A) = 1 \tag{100}$$

$$A > 0.9 \Rightarrow \Psi(A) = \frac{2}{\pi}\left[\sin^{-1}\left(\frac{0.9}{A}\right) + \frac{0.9}{A}\sqrt{1-\left(\frac{0.9}{A}\right)^2}\right] \tag{101}$$

The final wrist model for use with the graphical limit-cycle predictor is given in Fig. 46. The linear plant is the last element of the wrist model. This linear plant is preceded, from left to right, by a deadband, by a saturation, and by the nonlinear wrist model of Eq. (97). Notice the deadband is not a describing function; it has no effective gain that is a function of amplitude, but rather acts to only allow limit cycles with an

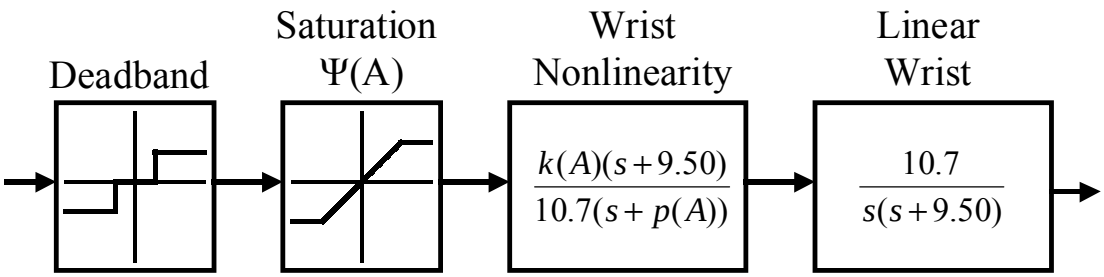


Fig. 46 Final Wrist Model

amplitude greater than 0.24 volts to pass through. The saturation is a hard-saturation describing function defined in Eqs. (100) and (101).

5.3 Two-Computer Amputee Simulation

Because amputees are not easily obtained for experiment, two computers are used to simulate the loop of Fig. 40. One computer controls the wrist, and one computer simulates an amputee. For the experiments here, the time delay Δ was left at zero, meaning no time delay was explicitly added in software. The setup of the two computers is shown in Fig. 47. Two "efferent" lines and two "afferent" are used, giving a total of four channels of PFM signals between the two computers, and no other method of communication between them. This is most likely the way that the wrist of the Experimental Neural Arm will be connected to an amputee.

The brain model, labeled "PI," is a proportional-plus-integral controller. The wrist controller, labeled "PV," is a proportional-plus-velocity-feedback controller. A PI controller may or may not be an accurate model of the brain, but the purpose here is not to accurately model the brain, but rather to verify the graphical limit-cycle predictor of Chapter 4.

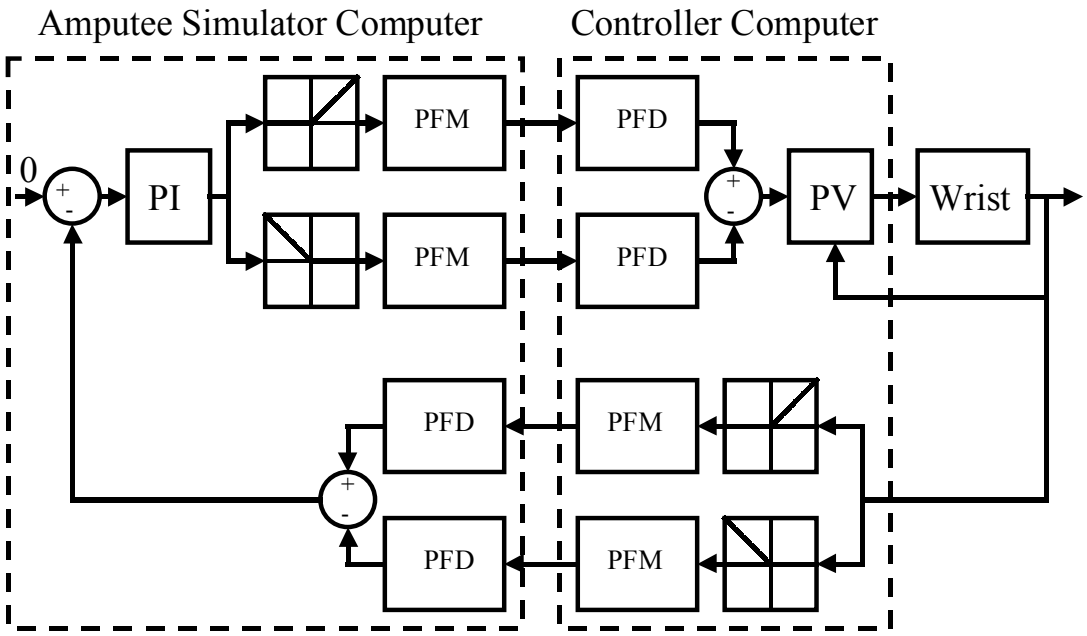


Fig. 47 Two-Computer Setup to Simulate Human Controlling the Experimental Neural Arm Wrist

The PFM method used on both computers is IPFM. The PFD method used in both computers is period measurement with a 10-Hz deadband. Both computers have a sampling rate of 3000 Hz. The software used was written in C++, and was largely an adaptation of software created by Mark Colton [15].

5.4 Graphical Limit-Cycle Prediction vs. Experimental Results

The graphical limit-cycle predictor algorithm of Chapter 4 is ready to be tested on a real system. The MATLAB code "Limit_Predictor2.m" was modified to include the wrist model of Fig. 46, and is given as "Limit_Predictor3.m" in Appendix D. Because of the desired form of the wrist model in the limit-cycle predictor algorithm, the PV controller implemented in the actual system is modeled as a PD controller in the algorithm. There is a subtle difference between the two, and the effect of this modeling choice will be discussed later.

Many experiments were performed with the two-computer setup of Fig. 47, with various gains for the PI and PV controllers. All experiments ran for approximately 20 seconds before data was taken to ensure the transient effects of start-up were gone. A few experimental data sets are compared to the graphical limit-cycle predictions of the systems. They were chosen to reflect the strengths and weaknesses of the graphical limit-cycle predictor.

Consider a system with a "brain" PI controller with gains $K_{p1} = 10$ and $K_i = 5$, with a PV controller with $K_{p2} = 20$ and $K_v = 0$, and with a modulation constant of $k_M = 200$. Figure 48 shows the experimental data of this system after a small disturbance. The frequency of the limit cycle is approximately 9.2 rad/sec, and the amplitude of the limit cycle falls in the range of 0.07-0.18 volts.

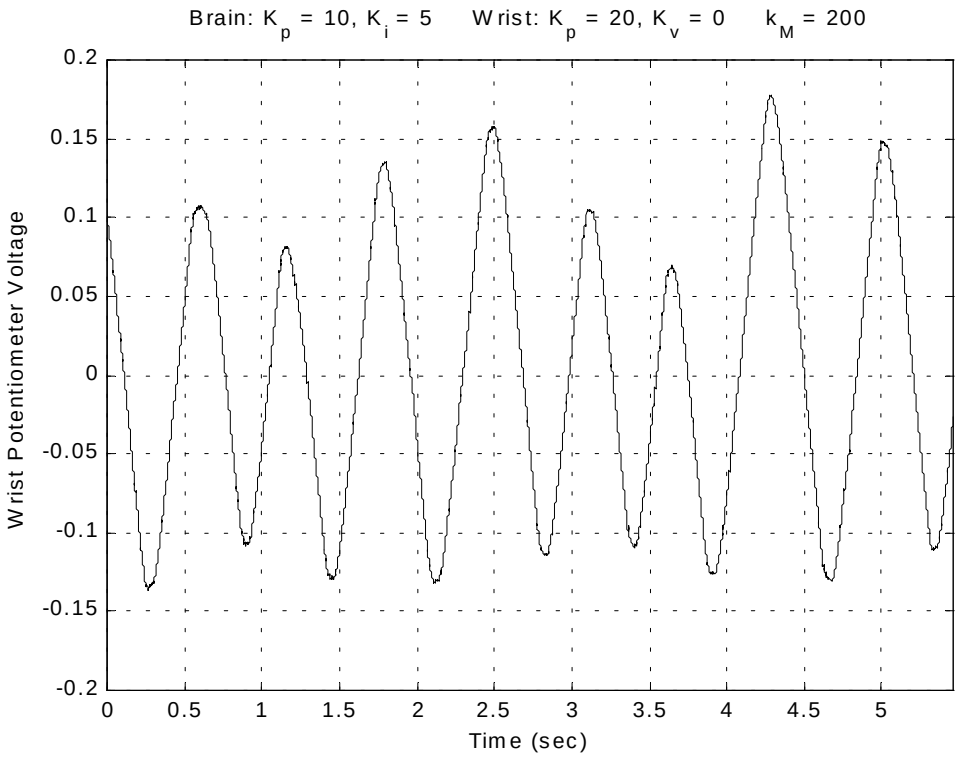


Fig. 48 Limit Cycle of Wrist Experiment #1

Figure 49 shows the graphical limit-cycle predictor after iterating frequencies until a match was found. A frequency of 9.3 rad/sec and an amplitude of 0.12 volts is predicted. For this example, the graphical predictor predicts the behavior of the real system well. This level of accuracy is characteristic of a large number of experiments. No other examples will be given where the prediction is highly accurate, because more can be learned from the cases where the prediction is less precise.

Consider a system with a "brain" PI controller with gains $K_{p1} = 1$ and $K_i = 7$, with a PV controller with $K_{p2} = 10$ and $K_v = 0$, and with a modulation constant of $k_M = 200$. Figure 50 shows the experimental data of this system. The frequency of the limit cycle is approximately 2.2 rad/sec, and the amplitude of the limit cycle is approximately 0.7 volts. Figure 51 shows the graphical limit-cycle predictor after iterating frequencies until a match was found. A frequency of 2.9 rad/sec and an amplitude of 0.45 volts are

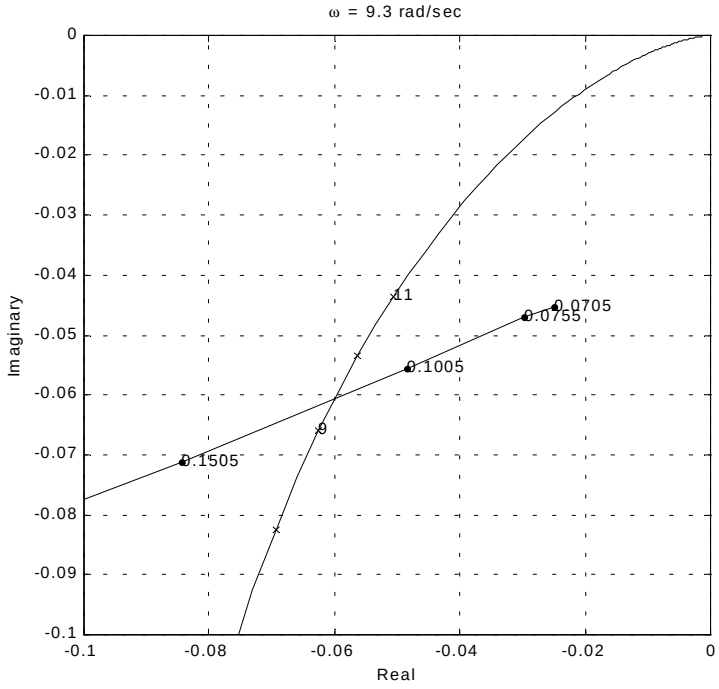


Fig. 49 Graphical Limit-Cycle Prediction of Wrist Experiment #1

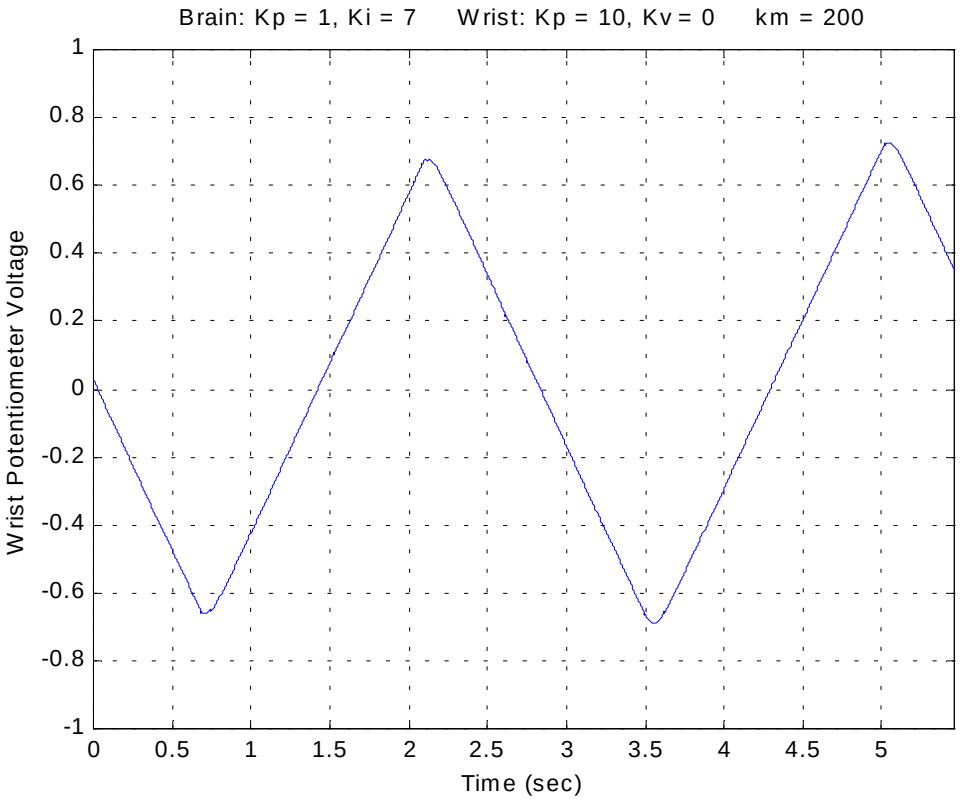


Fig. 50 Limit Cycle of Wrist Experiment #2

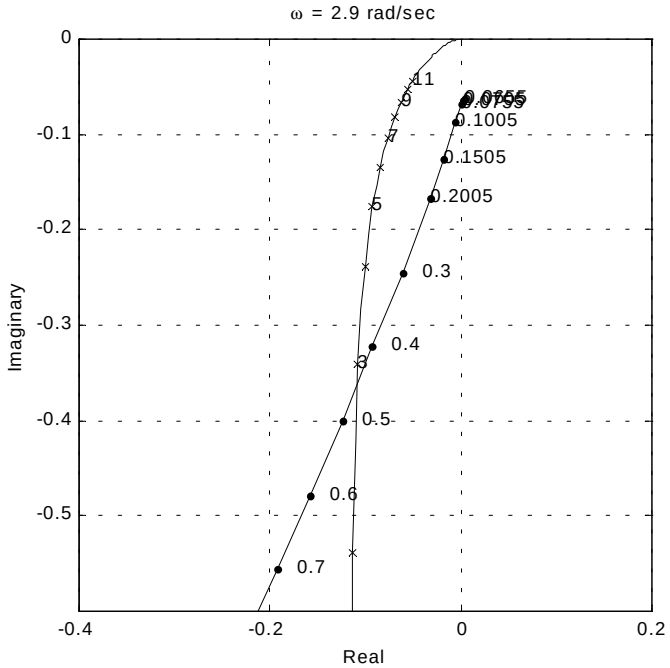


Fig. 51 Graphical Limit-Cycle Prediction for Wrist Experiment #2

predicted.

This prediction is less accurate than the previous case. Notice how the response of the wrist deviates from a true sinusoid. This is due to the input saturation, which can be seen in the long periods where the velocity is at a maximum, almost creating a triangle wave. Because the tabular describing function for the PPSSPFMD was created using sinusoidal inputs, the method does not predict limit-cycle amplitudes as accurately when the signals deviate from sinusoids. This is also the case when using traditional describing functions, which are also low-frequency sinusoidal approximations. Do note that the predicted frequency is still fairly accurate. This is because, regardless of the wave shape at any point in the loop, every element in the loop experiences some signal with a common periodic nature if the limit cycle is self-sustaining.

Also, notice how close to parallel the Nyquist plot and the negative inverse describing function are in this case. As in traditional describing function analysis, the less perpendicular the intersection, the less accurate. This is due to the sensitivity of the method when the lines are close to parallel; a small change in the position of the lines causes a large change in the point of intersection. Figure 52 shows the graphical limit-cycle predictor for a frequency of 2.2 rad/sec. Notice how close the amplitude of 0.7 volts is to intersecting with the Nyquist plot. In general, the more perpendicular the locus is to the Nyquist plot at the point of intersection, the more robust the prediction. This problem, along with the saturation problem previously mentioned, are problems that occur with traditional describing functions as well, and do not reflect poorly on the graphical limit-cycle predictor for PFM systems.

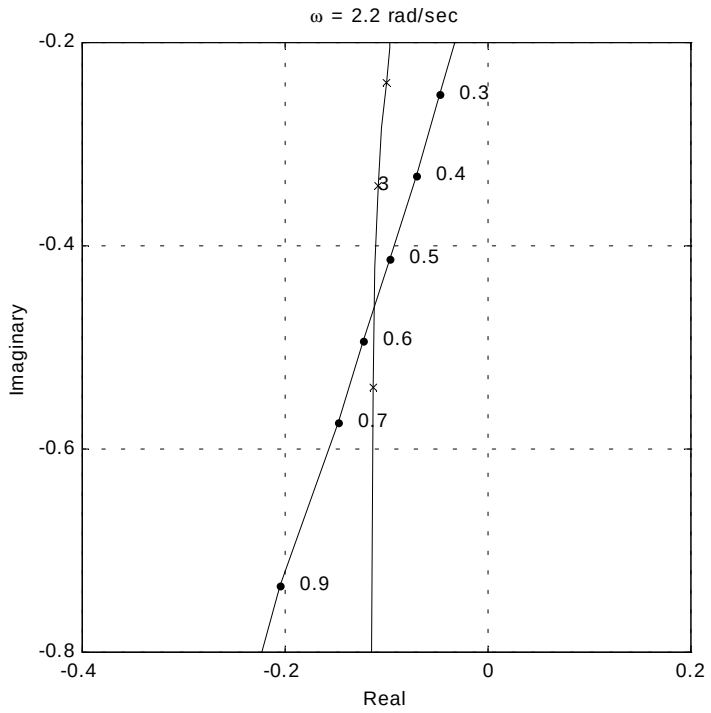


Fig. 52 Sensitivity of Graphical Limit-Cycle Prediction of Wrist Experiment #2

For the final example, consider a system with a "brain" PI controller with gains $K_{p1} = 5$ and $K_i = 5$, with a PV controller with $K_{p2} = 10$ and $K_v = 1.5$, and with a modulation constant of $k_M = 200$. Figure 53 shows the experimental data of this system. The frequency of the limit cycle is approximately 10 rad/sec, but it varies enough to see the change in frequency with the naked eye. The amplitude of the limit cycle falls in the range of 0.07-0.17 volts. Figure 54 shows the graphical limit-cycle predictor after iterating frequencies until a match was found. A frequency of 13.1 rad/sec and an amplitude of 0.074 volts are predicted.

The predicted amplitude is on the low end of the amplitudes seen, but still falls within the correct range of possible values. The prediction of the frequency, however, seems to be too high. This appears to be caused by replacing the actual PV controller

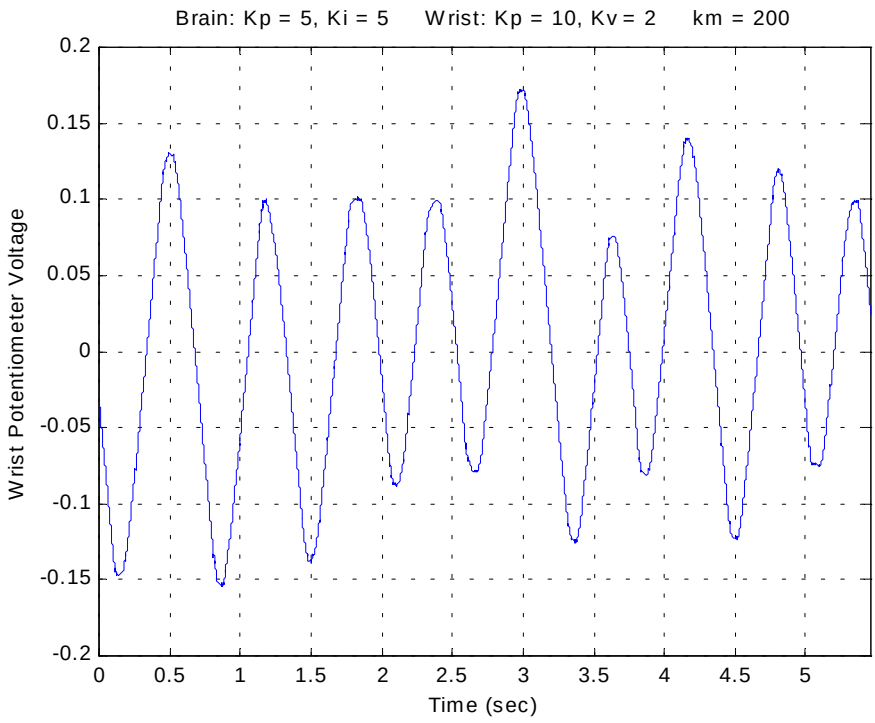


Fig. 53 Limit Cycle of Wrist Experiment #3

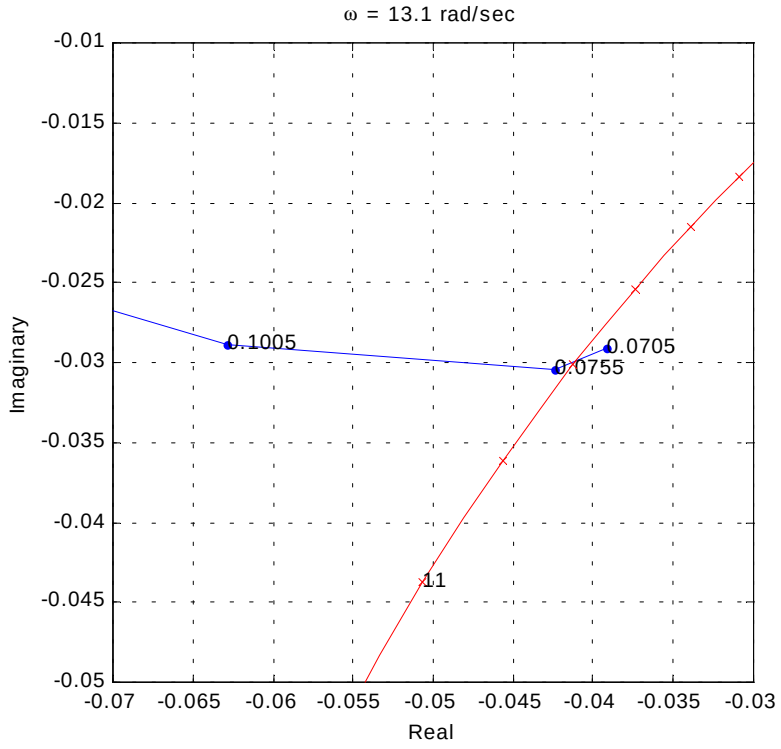


Fig. 54 Graphical Limit-Cycle Prediction of Wrist Experiment #3

with a PD controller in the limit-cycle predictor algorithm. Using velocity feedback stabilizes a system slightly better than using a PD controller, even though their behaviors are similar in many ways. This is why the real system is more damped (it has a lower frequency) than the PD approximation of the system.

This modeling problem occurs because the four-element wrist model of Fig. 46 prohibits using velocity feedback in the limit-cycle predictor algorithm. The velocity feedback would have to be fed back to the first wrist element of Fig. 46 (the deadband), but the algorithm, as it is currently posed, allows for elements only in series. This problem is caused both by having a highly nonlinear wrist, and also by using an algorithm that requires a loop structure that is not always valid.

In practice, velocity feedback is not needed in control of the Experimental Neural Arm, because the friction in the arm creates enough dampening. If only proportional feedback is used, the problems of this last example will not be seen.

Table 7 shows limit-cycle amplitudes and frequencies seen in experiments with the wrist compared to the predicted limit-cycle amplitudes and frequencies. The table gives values for different combinations of K_{p1} and K_i (from the amputee brain simulator), and K_{p2} and K_v (from the wrist controller). The "limit annulus" behavior is seen in the real wrist even more than in simulations, because the wrist does not have symmetric properties; the experimental values given in the table show a range of amplitudes and frequencies when the range is large.

Table 7 Comparison of Graphically-Predicted and Experimental Limit Cycles in Experimental Neural Arm Wrist ($k_M = 200$)

Brain and Wrist Gains	Experimental Amplitude (volts)	Experimental Frequency (rad/sec)	Predicted Amplitude (volts)	Predicted Frequency (rad/sec)
$K_{p1} = 1$ $K_i = 0$ $K_{p2} = 10$ $K_v = 0$	0.1-0.15	9.1	0.14	7.3
$K_{p1} = 1$ $K_i = 5$ $K_{p2} = 10$ $K_v = 0$	0.24-0.31	5	0.27	0.47
$K_{p1} = 10$ $K_i = 5$ $K_{p2} = 20$ $K_v = 0$	0.07-0.15	9.2	0.12	9.2
$K_{p1} = 1$ $K_i = 7$ $K_{p2} = 10$ $K_v = 0$	0.7	2.2	0.45	2.9
$K_{p1} = 5$ $K_i = 5$ $K_{p2} = 10$ $K_v = 1.5$	0.07-0.17	8.3-11.9	0.074	13.1

6. FUTURE WORK

The purpose of this thesis is to develop tools for control system analysis and design of systems containing pulse frequency modulation. The primary intended application of this thesis is for use with the Experimental Neural Arm, but no effort is made here to develop more accurate models of the human central nervous system. For the work of this thesis to be applied accurately, it is necessary to develop better models of the human nervous system.

6.1 Develop Better Model of Human/Arm System

The model of the human central nervous system interfaced with an artificial arm, seen in Fig. 40 is an adaptation of that of Gossett et al. [12], although they did not include pulse frequency modulation in the model. This model uses position error in the forward loop, and position in the feedback loop. It is entirely possible that this is not an accurate model of a human interfacing with an artificial arm.

Stein [16] suggests that three different types of motoneurons transmit three different types of information: alpha motoneurons transmit force information, static gamma motoneurons transmit position information, and dynamic gamma motoneurons transmit velocity information. It is very important to know which type of nerve is being used to control the Experimental Neural Arm, but any of the three could theoretically be used for control.

6.2 Consider Logarithmic Pulse Frequency Modulation and Demodulation

From experiments with amputees, it appears there may be a logarithmic relationship between the input pulse frequency to an afferent nerve and the "continuous" signal sensed by the amputee [8]. Pulse frequency modulation in the nervous system would then have the inverse relationship between input signal and efferent output pulse frequency. This logarithmic model has not been quantified yet; it is simply a verbal description given by amputees.

The tabular describing function of the PPSSPFMD could easily be recalculated using the post-filter method of Section 4.2, by first calculating the logarithm of the signal before passing it through the PPSSPFMD.

6.3 Consider Demodulation of "Noisy" PFM Signals

The PFD methods of Chapter 3 were analyzed only for the demodulation of noise-free PFM signals. In this context, "noise" is the occurrence of an extraneous pulse or a missing pulse. Period measurement was determined to be the best overall PFD method with a fast sampling time, but this method would be the most sensitive to noisy PFM signals. The low-pass filtering methods would be relatively insensitive to noisy PFM signals.

It would be nice to know the level of noise that is needed before the errors encountered in period measurement equal those encountered in low-pass filtering. It would first be necessary to quantify this "pulse noise." If the point when errors are the same was known, the best PFD method could be chosen based on the level of noise in the PFM signal.

6.4 Include Stiffness Control

It is possible that the rectifying nonlinearities of the PPSSPFMD setup of Fig. 11 should be remodeled. In the human body, two opposing muscles may be flexed at the same time, not to move the joint, but rather to increase the stiffness of the joint. This indicates that the "either/or" behavior of the rectifying nonlinearities used in this thesis may need to be remodeled to allow activity in both the upper and lower paths of the PPSSPFMD simultaneously.

Once better models of the rectifying nonlinearities are found, it may be possible to recreate the stiffness control of the human body when pulses are seen simultaneously from the nerves of opposing muscles. This could possibly be accomplished by increasing controller gains in this situation.

6.5 Use Limit-Cycle Matching to Determine Human Parameters

The model of the human nervous system interfaced with an artificial arm, seen in Fig. 40, has many modeled elements, each of which is subject to error. There is a model of the brain (this could easily be oversimplified), there is a model of the time delays in the nervous system (these could change from person to person, or even due to body chemistry), there are models of the PFM and PFD methods used by the human body (problems with these have been discussed previously), and finally there is the model of the electromechanical prosthesis.

The easiest element to get an accurate model of is the prosthesis; traditional controls engineering can be used here. It is also reasonable to assume the time delays of the nerves can be modeled accurately, and data are available currently to build such a model. As the remaining element models are improved upon, it may be possible to use

the limit-cycle prediction algorithm to determine the remaining human parameters. For example, if a reasonable model of the human PPSSPFMD was found, the only remaining element is the model of the brain. By matching predicted limit-cycle amplitudes and frequencies with those seen in a prosthetic connected to an amputee, a functional brain model could be developed.

6.6 Use Error Envelopes for H-Infinity Design

The maximum errors due to pulse frequency modulation and demodulation were found in Chapters 2 and 3. It may be possible to use these maximum errors for H-Infinity design of control systems with pulse frequency modulation. It should be noted that the time delays due to pulse frequency modulation and demodulation would also need to be accounted for in some way.

7. CONCLUSIONS

This thesis successfully developed new tools for control system analysis and design of systems containing pulse frequency modulation.

Three methods of pulse frequency modulation (IPFM, V/F converters, and USSH) were found to be equivalent, after a brief discrepancy at start-up, when used in a single-signed scheme. When modeling PFM, none of these three methods is superior to the others. The problems encountered when pulse frequency modulating a signal with a digital computer were also quantified, showing large errors at high pulse frequencies.

Five methods of pulse frequency demodulation (period measurement, first-order low-pass filtering, second-order low-pass filtering, finite-impulse-response filtering, and fixed-time window) were analyzed and the errors encountered with each method were quantified and compared. Period measurement was determined to be the best overall method of PFD, but this is only when considering noise-free PFM signals; the presence of noise could possible result in low-pass filtering being a better PFD choice.

A method was created to obtain the equivalent gain and phase-lag of any pulse frequency modulation/demodulation system, creating a frequency-dependent tabular describing function. An algorithm was developed that uses the tabular describing function to graphically predict the existence (and amplitude and frequency if they exist) of limit cycles in systems containing pulse frequency modulation. The graphical predictions were compared to simulations and were found to be very accurate.

Two computers were used to simulate the interaction between an amputee and the Experimental Neural Arm; one computer simulated the amputee, and the other directly controlled the arm. The two computers communicated to each other only by parallel-path single-signed pulse frequency modulation, much like a real amputee would control the arm.

A nonlinear model of the Experimental Neural Arm wrist was created in the form needed by the graphical limit-cycle prediction algorithm. Graphical limit-cycle predictions were compared to actual limit cycles seen in the wrist being controlled by the two-computer setup. The predictions matched the limit cycles seen in the wrist well.

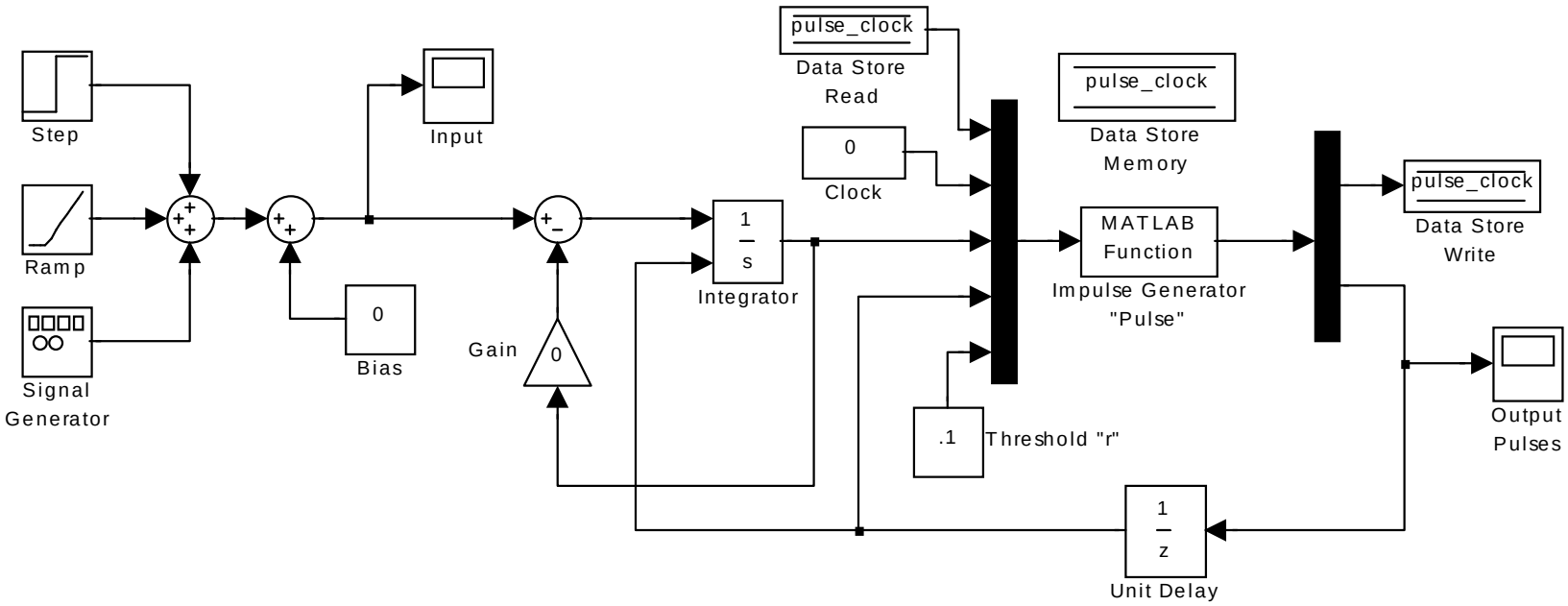
APPENDIX A

PULSE FREQUENCY MODULATION SIMULINK MODELS

AND MATLAB SCRIPTS

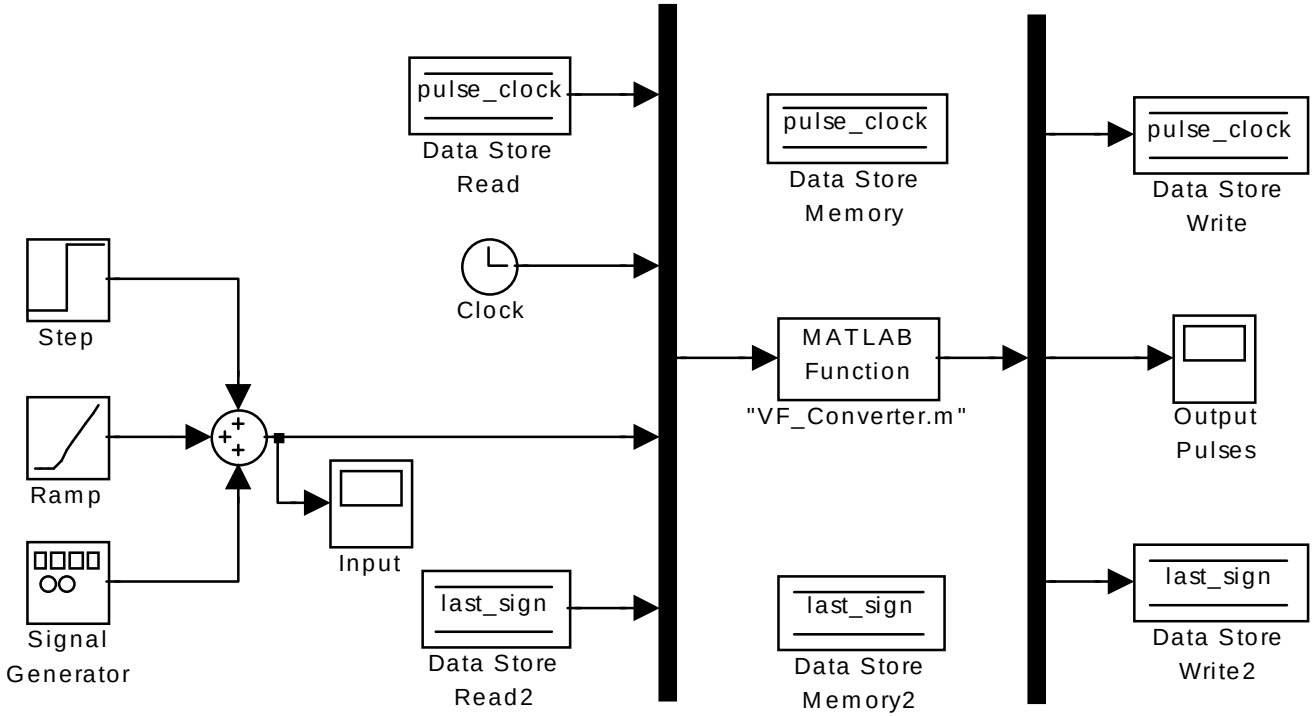
Sigma Pulse Frequency Modulator

This is a reconfigurable pulse frequency modulator. It can convert any analog signal into a PFM signal. Setting "Bias" creates single-signed PFM. Setting "Gain" on the integral feedback loop creates Neural PRM. With no bias or feedback gain, the system is a double-signed Integral Pulse Frequency Modulator.



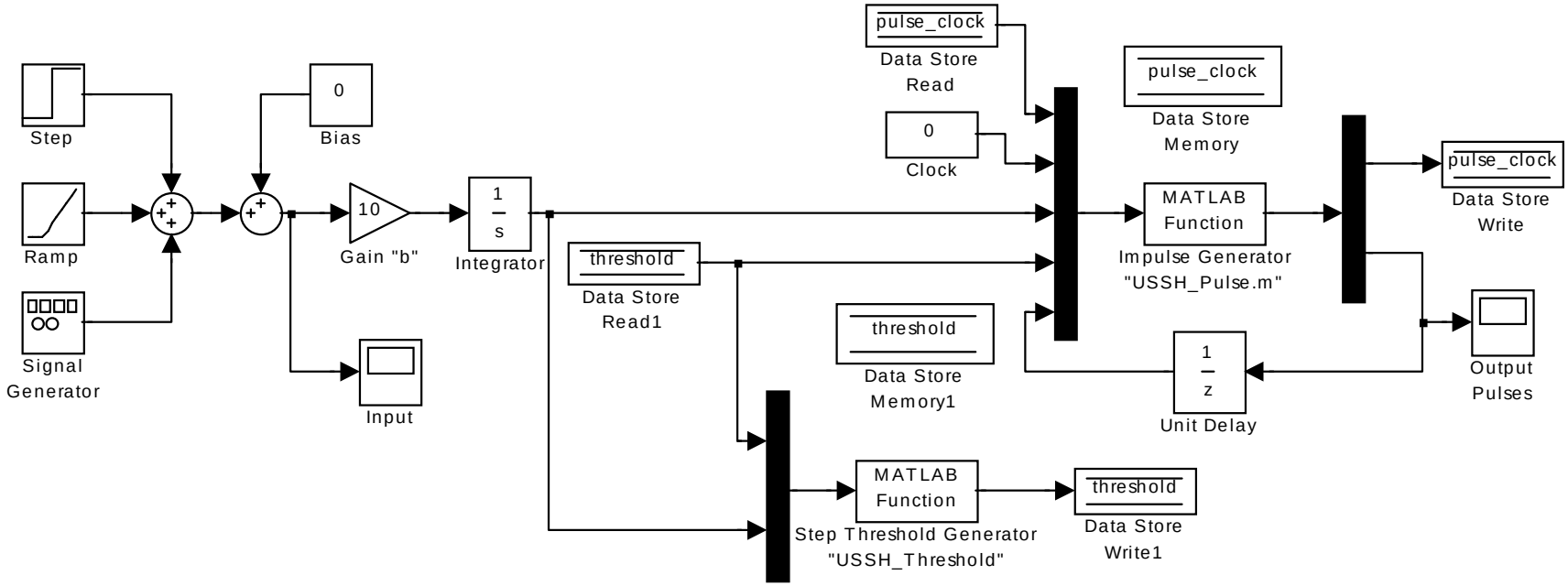
Voltage-to-Frequency Converter

This V/F converter is not implemented like a physical circuit. It is basically an IPFM scheme, but a pulse is emitted at start-up.



Unified Steps Sample & Hold

This is the Unified Steps Sample & Hold method of PFM. The serraphile function is not used here; a "moving threshold" method, which is presented in the original paper and is equivalent, is used instead.



```
% Pulse.m

% For use with the Sigma Pulse Frequency Modulator Simulink model.
% This function delivers a pulse of a magnitude defined below when the
% input crosses the threshold value.  It can be used for double-signed PFM.

function demux = Pulse(mux)

mag = 1;  % Magnitude of pulse.
pulse_clock = mux(1);
clock = mux(2);
input = mux(3);
last_out = mux(4);
threshold = mux(5);

if and(pulse_clock == 0, and(input < threshold, input > -threshold))
   out_pulse = 0;
   pulse_clock = 0;
elseif and(pulse_clock == 0, input >= threshold)
   out_pulse = mag;
   pulse_clock = clock;
elseif and(pulse_clock == 0, input <= -threshold)
   out_pulse = -mag;
   pulse_clock = clock;
else
   out_pulse = 0;
   pulse_clock = 0;
end

demux(1) = pulse_clock;
demux(2) = out_pulse;
```

```
% USSH_Pulse.m

% This function delivers a positive pulse of a magnitude % % defined below when the
input crosses the threshold value, % and a negative pulse when the input crosses
% (threshold - 1).  To be used with Unified States Sample & % Hold Simulink model.

function demux = USSH_Pulse(mux)

mag = 1; % Magnitude of pulse.
pulse_clock = mux(1);
clock = mux(2);
input = mux(3);
threshold = mux(4);
last_out = mux(5);

if and(pulse_clock == 0, and(input < threshold, input > -threshold))
   out_pulse = 0;
   pulse_clock = 0;
elseif and(pulse_clock == 0, input >= threshold)
   out_pulse = mag;
   pulse_clock = clock;
elseif and(pulse_clock == 0, input <= (threshold-1))
   out_pulse = -mag;
   pulse_clock = clock;
else
   out_pulse = 0;
   pulse_clock = 0;
end

demux(1) = pulse_clock;
demux(2) = out_pulse;
```

```
% USSH_Threshold.m

% This function generates the new threshold value in the
% Unified Steps Sample & Hold PFM method.

function threshold = USSH_Threshold(mux)

threshold = mux(1);
input = mux(2);

if and(input < threshold, input > (threshold-1))
   threshold = threshold;
elseif input >= threshold
   threshold = threshold+1;
else
   threshold = threshold-1;
end
```

```
% VF_Converter.m

% This function is voltage-to-frequency converter.  It delivers
% pulses at a frequency that is a function of the input signals magnitude.
% A pulse is given off at the beginning of the simulation.

function demux = VF_Converter(mux)

mag = 1; % Magnitude of pulse.
pulse_clock = mux(1);
clock = mux(2);
input = mux(3);
last_sign = mux(4);

freq = i2f_func(input);
period = 1/freq;
time_since = clock-pulse_clock;

if and(clock > 0, and(abs(freq) > eps, pulse_clock == 0))
   out_pulse = mag*sign(freq);
   pulse_clock = clock;
   last_sign = sign(freq);
elseif and(pulse_clock ~= 0, and(sign(freq) == last_sign, time_since >= abs(period)))
   out_pulse = mag*last_sign;
   pulse_clock = clock;
   last_sign = last_sign;
elseif and(sign(freq) ~= last_sign, pulse_clock ~= 0)
   out_pulse = mag*sign(freq);
   pulse_clock = clock;
   last_sign = sign(freq);
elseif and(abs(freq) <= eps, pulse_clock == 0)
   out_pulse = 0;
   pulse_clock = 0;
   last_sign = 0;
else
   out_pulse = 0;
   pulse_clock = pulse_clock;
   last_sign = last_sign;
end

demux(1) = pulse_clock;
demux(2) = out_pulse;
demux(3) = last_sign;
```

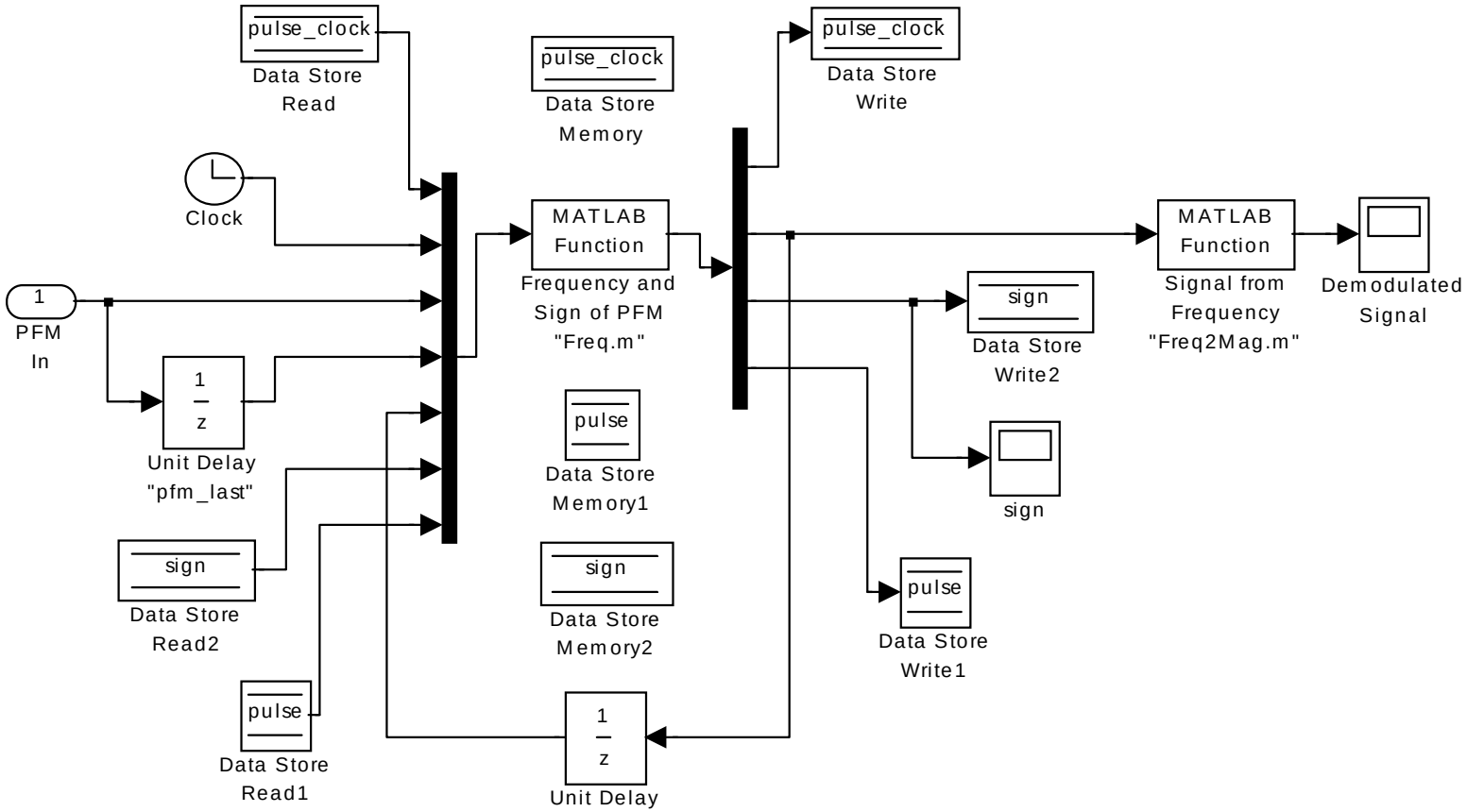
```
% i2f_func.m

% This is the relationship between magnitude of input and frequency of output for
% the V/F converter.

function freq = i2f_func(input)

k = 20; % Modulation Constant
freq = k*input;
```

APPENDIX B

PERIOD MEASUREMENT PFD SIMULINK MODEL

AND MATLAB SCRIPTS

Period Measurement Pulse Frequency Demodulator



```
% Freq.m

% This function delivers the frequency of the PFM signal.
% This program can demodulate double-signed PFM signals.

function demux = freq(mux)

pulse_clock = mux(1);
clock = mux(2);
pfm = mux(3);
pfm_last = mux(4);
freq_last = mux(5);
sign_last = mux(6);
pulse = mux(7);
diff = pfm-pfm_last;

if and(pulse_clock == 0, abs(diff) <= eps)
   pulse_clock = 0;
   freq = 0;
   pulse_sign = 0;
   pulse = 0;
elseif and(abs(diff) > eps, pulse_clock == 0)
   pulse_clock = clock;
   freq = 0;
   pulse_sign = sign(diff);
   pulse = 1;
elseif and(and(abs(diff) > eps, pulse == 0), ...
      and(pulse_clock ~= 0, sign(diff) == sign_last))
   freq = 1/(clock-pulse_clock)*sign(diff);
   pulse_clock = clock;
   pulse_sign = sign(diff);
   pulse = 1;
elseif and(and(abs(diff) > eps, pulse == 0), ...
      and(pulse_clock ~= 0, sign(diff) ~= sign_last))
   freq = 0;
   pulse_clock = clock;
   pulse_sign = sign(diff);
   pulse = 1;
else
   temp = 1/(clock-pulse_clock);
   freq = min(abs(freq_last),temp)*sign_last;
   pulse_sign = sign_last;
   pulse = 0;
end

demux(1) = pulse_clock;
```

```
demux(2) = freq;
demux(3) = pulse_sign;
demux(4) = pulse;
```

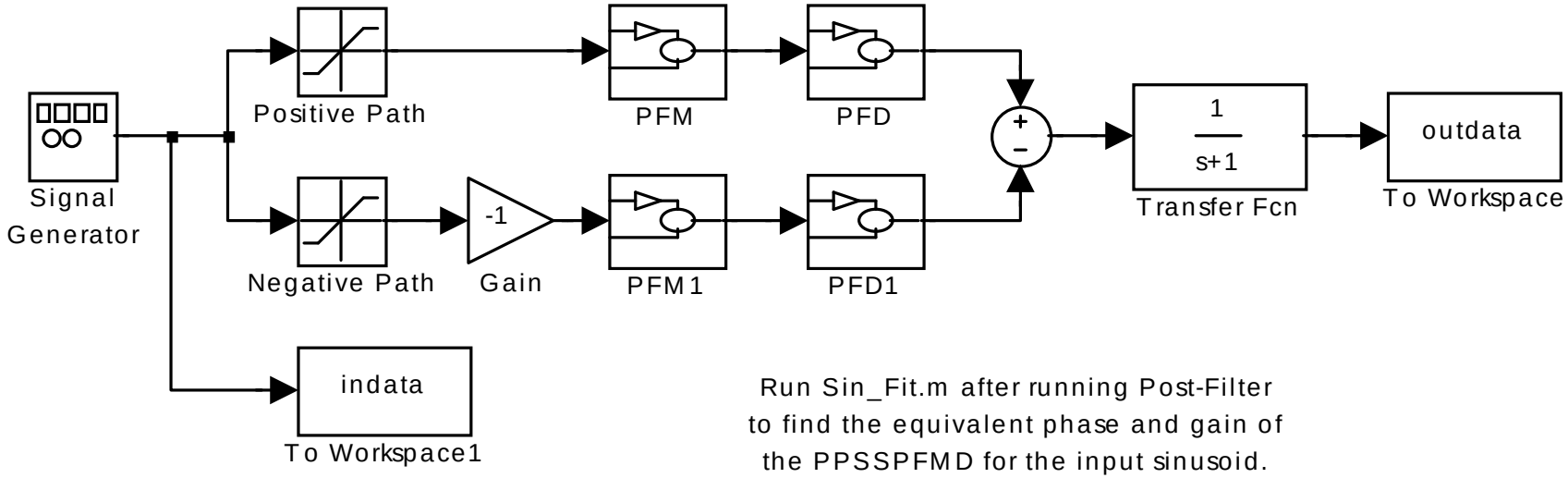
```
% Freq2Mag.m

% Uses modulation contant to reconstruct modulated signal
% from demodulated frequency.

function mag = Freq2Mag(freq)

k = 20; % Modulation Constant
mag = freq/k;
```

APPENDIX C

POST-FILTERING METHOD SIMULINK MODEL

AND MATLAB SCRIPT

Post-Filter



Run Sin_Fit.m after running Post-Filter to find the equivalent phase and gain of the PPSSPFMD for the input sinusoid.

```
% Sin_Fit.m
% Jake Abbott
%
% Runs phase_shift.m and gain_fit.m to find
% the best fit of a sinusiod to data.
%
% [total_phase,phase,gain] = Sin_Fit(indata,outdata,tout,omega,phi_max)
%
% "total_phase" is the phase lag between the input and the output
% "phase" is the phase lag of the PPSSPFMD
% "gain" is the gain of the PPSSPFMD
% indata is the reference sinusoid
% outdata is the data to be fit to a sinusoid
% tout is the time data
% omega is the frequency of the reference sinusoid (and the curve fit)
% phi_max is the largest possible phase lag considered, in degrees

function [total_phase,phase,gain] = Sin_Fit(indata,outdata,tout,omega,phi_max)

% Model of post-filter.
num = 1;
den = [1 1];
plant = tf(num,den);
[bode_gain,bode_phase] = bode(plant,omega);

% Only use second half of data to eliminate transient behavior.
N = round(length(indata)/2);
indata2 = indata(N:length(indata));
outdata2 = outdata(N:length(outdata));
tout2 = tout(N:length(tout));

% Find best sinusoidal fit.
indata_mag = max(indata2);
outdata_mag = max(outdata2);
total_gain = outdata_mag/indata_mag;
total_phase = phase_shift(indata2,outdata2,tout2,total_gain,omega,phi_max);
fit1 = outdata_mag*sin(omega*tout2+total_phase*pi/180);
RMS_fit1 = RMS(fit1);
RMS_outdata = RMS(outdata2);
fit_gain = RMS_outdata/RMS_fit1;
fit = fit1*fit_gain;
phase = total_phase-bode_phase;
gain = total_gain*fit_gain/bode_gain;

figure(1); clf;
plot(tout2,indata2,tout2,outdata2,tout2,fit)
```

```
% RMS.m
% Jake Abbott
%
% Calculates the RMS value of the input signal.

function RMS_value = RMS(signal)

sum = 0;
for i = 1:length(signal),
   sum = sum + signal(i)^2;
end

RMS_value = sqrt(sum/length(signal));
```

```
% phase_shift.m
% Jake Abbott
%
% Finds the best-fit sinusoid that is a pure phase-lag of a reference
% sinusoid, using a least-squares curve fit.  Output is the phase-lag
% in degrees.  Time spacing is 0.001 seconds.
%
% phase = phase_shift(indata,outdata,t,gain,omega,phi_max)
% "indata" is the reference sinusoid
% "outdata" is the data to be fit to a sinusoid
% "t" is the time vector
% "gain" is the gain used when finding best phase shift
% "omega" is the frequency of the reference sinusoid (and the curve fit)
% "phi_max" is the largest possible phase lag considered, in degrees

function phase = phase_shift(indata,outdata,t,gain,omega,phi_max)

phi = 0:1:phi_max;
phi = -phi*pi/180;
Sphi = zeros(size(phi));

for i = 1:length(phi),
  S = 0;
  for j = 1:length(indata),
    S = S + 2*outdata(j)/gain*cos(omega*t(j)+phi(i))- ...
      2*sin(omega*t(j)+phi(i))*cos(omega*t(j)+phi(i));
  end
  Sphi(i) = S;
end

[Sphi_min,index] = min(abs(Sphi));
phase = phi(index)*180/pi;
S_phase = Sphi_min;
```

APPENDIX D

GRAPHICAL LIMIT-CYCLE PREDICTION MATLAB SCRIPTS

```
% Limit_Predictor.m
% Jake Abbott
%
% Limit cycle predictor Pulse Frequency Modulated setup given below.
%
%                +---------------+     +------+
%      --->O--->| PPSSPFMD |--->| SYS  |---+--->
%           -|      +---------------+     +------+    |
%            |                                        |
%            +----------------------------------------+
%
% The Pulse Frequency Modulator can be any integral scheme.  The Pulse
% Frequency Demodulator measures the period between pulses, and uses a
% frequency deadband of 10-Hz (any pulse period greater than 0.1-sec
% sets the demodulated frequency to zero).
%
% The system "SYS" should be defined in the workspace, in transfer
% function notation, before running limit_predictor.m.  The user is
% prompted to input the Modulation Constant.
%
% The frequency closest to the unit circle is marked, and x's mark
% integer frequencies.  If the system nyquist plot (red) encirlces the
% point (-1,0), then the feedback system is unstable.  If the system
% nyquist plot does not intersect the blue line, then the feedback
% system has no limit cycle.
%
% The user is prompted to iterate the frequency line (blue) being plotted.
% Once the frequency line being plotted matches the frequency of the
% nyquist plot at the intersection, that is the limit cycle frequency.
% The amplitude of the limit cycle is pulled directly from the blue line.

Tabular_PPSSPFMD; % Load tabular describing function

% Modulation constant
kMf = input('Input Modulation Constant  ');

% Frequencies of nyquist plot
w = [0.01:0.01:0.1 0.2:0.1:50 50:300];
[mag,phase] = bode(SYS,w);
[temp,index] = min(abs(mag-ones(size(mag))));
cross_freq = w(index)
% Integer frequencies of x's on nyquist plot
w1 = 1:300;
[mag1,phase1] = bode(SYS,w1);
[temp,index1] = min(abs(mag1-ones(size(mag1))));
cross_freq1 = w1(index1);
```

```
% Create nyquist plot of plant
[Re,Im] = nyquist(SYS,w);
Re1 = zeros(length(Re),1);
Im1 = zeros(length(Im),1);
for k = 1:length(w),
     Re1(k) = Re(1,1,k);
     Im1(k) = Im(1,1,k);
end
[Re,Im] = nyquist(SYS,w1);
Re2 = zeros(length(Re),1);
Im2 = zeros(length(Im),1);
for k = 1:length(w1),
     Re2(k) = Re(1,1,k);
     Im2(k) = Im(1,1,k);
end

% Amp_kM that will be plotted
Amp = [15 20 30 40 60 80 100 120 140 180 220 260 300 360 ...
      420 500 600 800 1000];
realimag = zeros(length(Amp),1);
omega = cross_freq;
while omega > 0,
  for k = 1:length(Amp),
    gain = interp2(Omega,Amp_kM,Gain,omega,Amp(k));
    R = 1/(gain);
    phase = interp2(Omega,Amp_kM,Phase_Lag,omega,Amp(k));
    realimag(k) = -R*cos(phase)-R*i*sin(phase);
  end
  figure(1); clf; plot(realimag);
  figure(1); hold on; plot(realimag,'.');
  text(real(realimag),imag(realimag),num2str(Amp'/kMf));

  % Plot nyquist plot of plant
  figure(1); hold on; plot(Re1,Im1,'r');
    figure(1); hold on; plot(Re2(1:2*index1),Im2(1:2*index1),'rx');
    text(Re2(index1),Im2(index1),int2str(cross_freq1));

  % Create unit circles
    Real = zeros(91,1);
    Imag = zeros(91,1);
    for k = 1:91,
        Real(k) = -cos((k-1)*pi/180);
        Imag(k) = -sin((k-1)*pi/180);
  end
  figure(1); hold on;
```

```
    plot(Real,Imag,'g--');

    axis([-2.5 0 -2.5 0]);
    axis square; grid
  xlabel('Real'); ylabel('Imaginary')

  omega = input('Input New Frequency (Enter 0 to Quit)  ');
end
```

```
% Limit_Predictor2.m
% Jake Abbott
%
% Limit cycle predictor Pulse Frequency Modulated setup given below.
%
%                +-------+    +---------------+    +--------+
%     --->O--->| SYS1 |--->| PPSSPFMD |--->| SYS2  |----+--->
%          -|     +-------+    +---------------+    +--------+     |
%           |                                                      |
%           |                +---------------+                     |
%           +----------------| PPSSPFMD |<--------------------+
%                            +---------------+
%
% The Pulse Frequency Modulator can be any integral scheme.  The Pulse
% Frequency Demodulator measures the period between pulses, and uses a
% frequency deadband of 10-Hz (any pulse period greater than 0.1-sec
% sets the demodulated frequency to zero).
%
% The systems SYS1 and SYS2 should be defined in the workspace, in transfer
% function notation, before running Limit_Predictor2.m.  The user is
% prompted to input the Modulation Constant, which is the same for both
% PPSSPFMD blocks.  The two PPSSPFMD blocks and SYS1 are combinded and
% shown as a blue line, while the nyquist plot of SYS2 is shown as a red line.
%
% The frequency closest to the unit circle is marked, and x's mark integer
% frequencies.  If the plant (SYS2) nyquist plot (red) encirlces the point
% (-1,0), then the feedback system is unstable.  If the plant nyquist plot
% does not intersect the blue line, then the feedback system has no limit
% cycle.
%
% The user is prompted to iterate the frequency line (blue) being plotted.
% Once the frequency line being plotted matches the frequency of the
% nyquist plot at the intersection, that is the limit cycle frequency.
% The amplitude of the limit cycle is pulled directly from the blue line.

Tabular_PPSSPFMD; % Load tabular describing function

% Modulation constant
kM = input('Input Modulation Constant  ');

% Frequencies for SYS1 bode and SYS2 nyquist plot
w = [0.01 0.1:0.1:1 2:0.5:49 50:2:300];
mag1 = zeros(size(w)); mag2 = zeros(size(w));
phase1 = zeros(size(w)); phase2 = zeros(size(w));
for j = 1:length(w),
    [mag1(j),phase1(j)] = bode(SYS1,w(j));
```

```
  [mag2(j),phase2(j)] = bode(SYS2,w(j));
end
phase1 = phase1*pi/180;
phase2 = phase2*pi/180;
[temp,index] = min(abs(mag2-ones(size(mag2))));
cross_freq = w(index)
% Integer frequencies of x's on plant nyquist plot
w_int = 1:300;
[mag2int,phase2int] = bode(SYS2,w_int);
[temp,index_int] = min(abs(mag2int-ones(size(mag2int))));
cross_freq_int = w_int(index_int);

% Create nyquist plot of plant
[Re,Im] = nyquist(SYS2,w);
Re2 = zeros(length(Re),1);
Im2 = zeros(length(Im),1);
for k = 1:length(w),
    Re2(k) = Re(1,1,k);
    Im2(k) = Im(1,1,k);
end
[Re,Im] = nyquist(SYS2,w_int);
Re2_int = zeros(length(Re),1);
Im2_int = zeros(length(Im),1);
for k = 1:length(w_int),
    Re2_int(k) = Re(1,1,k);
    Im2_int(k) = Im(1,1,k);
end

% Amp_kM that will be plotted
Amp = [20 30 40 60 80 100 120 140 180 220 260 300 360 450 600 ...
    700 800 900 1000];
realimag = zeros(length(Amp),1);
omega = cross_freq;
while omega > 0,
  for k = 1:length(Amp),
    gain_a = interp2(Omega,Amp_kM,Gain,omega,Amp(k));
    gain_b = interp1(w,mag1,omega);
    phase_a = interp2(Omega,Amp_kM,Phase_Lag,omega,Amp(k));
    phase_b = interp1(w,phase1,omega);
    temp = gain_a*gain_b*Amp(k);
    if and(gain_a < 1000 , gain_a > -1000) % Check for NaN
      gain_c = interp2(Omega,Amp_kM,Gain,omega,temp);
      phase_c = interp2(Omega,Amp_kM,Phase_Lag,omega,temp);
    else
      gain_c = NaN;
      phase_c = NaN;
```

```
    end
    gain = gain_a*gain_b*gain_c;
    R = 1/(gain);
    phase = phase_a+phase_c-phase_b;
    realimag(k) = -R*cos(phase)-R*i*sin(phase);
  end
  figure(1); clf; plot(realimag);
  hold on; plot(realimag,'.');
  text(real(realimag),imag(realimag),num2str(Amp'/kM));

  % Plot nyquist plot of plant
  figure(1); hold on; plot(Re2,Im2,'r');
    plot(Re2_int(1:2*index_int),Im2_int(1:2*index_int),'rx');
  text(Re2_int(index_int),Im2_int(index_int),int2str(cross_freq_int));

  % Create unit circles
    Real = zeros(91,1);
    Imag = zeros(91,1);
    for k = 1:91,
        Real(k) = -cos((k-1)*pi/180);
        Imag(k) = -sin((k-1)*pi/180);
  end
  figure(1); hold on;
    plot(Real,Imag,'g--');

    axis([-3 0 -3 0]);
    axis square; grid
  xlabel('Real'); ylabel('Imaginary')

  omega = input('Input New Frequency (Enter 0 to Quit)  ');
end
```

```
% Limit_Predictor3.m
% Jake Abbott
%
% Limit cycle predictor Pulse Frequency Modulated setup given below.
% For use with Experimental Neural Arm Wrist #3.
%
%               +-------+    +---------------+    +------+    +-------+
%     --->O--->| SYS1 |--->| PPSSPFMD |--->|  NL  |--->| SYS2 |----+--->
%          -|     +-------+    +---------------+    +------+    +-------+     |
%           |                                                                  |
%           |                   +--------------+                               |
%           +-------------------| PPSSPFMD |<------------------------------+
%                               +--------------+
%
% The Pulse Frequency Modulator can be any integral scheme.  The Pulse
% Frequency Demodulator measures the period between pulses, and uses a
% frequency deadband of 10-Hz (any pulse period greater than 0.1-sec
% sets the demodulated frequency to zero).
%
% The systems SYS1 and SYS2 should be defined in the workspace, in transfer
% function notation, before running Limit_Predictor3.m.  The user is
% prompted to input the Modulation Constant, which is the same for both
% PPSSPFMD blocks.  The two PPSSPFMD blocks, SYS1, and NL are combinded and
% shown as a blue line, while the nyquist plot of SYS2 is shown as a red line.
%
% The frequency closest to the unit circle is marked, and x's mark integer
% frequencies.  If the plant (SYS2) nyquist plot (red) encirlces the point
% (-1,0), then the feedback system is unstable.  If the plant nyquist plot
% does not intersect the blue line, then the feedback system has no limit
% cycle.
%
% The user is prompted to iterate the frequency line (blue) being plotted.
% Once the frequency line being plotted matches the frequency of the
% nyquist plot at the intersection, that is the limit cycle frequency.
% The amplitude of the limit cycle is pulled directly from the blue line.

kp1 = 5;
ki = 5;
kp2 = 10;
kv = 1.5;

Tabular_PPSSPFMD; % Load tabular describing function

% Modulation constant
kM = input('Input Modulation Constant  ');
```

```
num = [kp1 ki];
den = [1 0];
SYS1 = tf(num,den);

num = 10.70;
den = [1 9.50 0];
SYS2 = tf(num,den);

% Frequencies for SYS1 bode and SYS2 nyquist plot
w = [0.01 0.1:0.1:1 2:0.5:49 50:2:300];
mag1 = zeros(size(w)); mag2 = zeros(size(w));
phase1 = zeros(size(w)); phase2 = zeros(size(w));
for j = 1:length(w),
     [mag1(j),phase1(j)] = bode(SYS1,w(j));
   [mag2(j),phase2(j)] = bode(SYS2,w(j));
end
phase1 = phase1*pi/180;
phase2 = phase2*pi/180;
[temp,index] = min(abs(mag2-ones(size(mag2))));
cross_freq = w(index)
% Integer frequencies of x's on plant nyquist plot
w_int = [0.1:0.1:0.9 1:300];
[mag2int,phase2int] = bode(SYS2,w_int);
[temp,index_int] = min(abs(mag2int-ones(size(mag2int))));
cross_freq_int = w_int(index_int);

% Create nyquist plot of plant
[Re,Im] = nyquist(SYS2,w);
Re2 = zeros(length(Re),1);
Im2 = zeros(length(Im),1);
for k = 1:length(w),
     Re2(k) = Re(1,1,k);
     Im2(k) = Im(1,1,k);
end
[Re,Im] = nyquist(SYS2,w_int);
Re2_int = zeros(length(Re),1);
Im2_int = zeros(length(Im),1);
for k = 1:length(w_int),
     Re2_int(k) = Re(1,1,k);
     Im2_int(k) = Im(1,1,k);
end

SYS_PD = tf([kv kp2],1);

% Amp_kM that will be plotted
Amp = [13.1 14.1 15.1 20.1 30.1 40.1 60 80 100 120 140 180 240 300 450 600 ...
```

```
    700 800 900 1000];
realimag = zeros(length(Amp),1);
omega = cross_freq;
while omega > 0,
  for k = 1:length(Amp),
    % Include Feedback PFM/PFD
    gain_a = interp2(Omega,Amp_kM,Gain,omega,Amp(k));
    phase_a = interp2(Omega,Amp_kM,Phase_Lag,omega,Amp(k));
    % Include SYS1
    [gain_b,phase_b] = bode(SYS1,omega);
    phase_b = phase_b*pi/180;
    % Include Forward Path PFM/PFD
    if and(gain_a < 1000 , gain_a > -1000) % Check for NaN
      temp = gain_a*gain_b*Amp(k);
      gain_c = interp2(Omega,Amp_kM,Gain,omega,temp);
      phase_c = interp2(Omega,Amp_kM,Phase_Lag,omega,temp);
    else
      gain_c = NaN;
      phase_c = NaN;
    end
    % Include PD Controller
    [gain_f,phase_f] = bode(SYS_PD,omega);
    phase_f = phase_f*pi/180;
    % Include Saturation with Deadband
    if and(gain_c < 1000 , gain_c > -1000) % Check for NaN
        A = gain_a*gain_b*gain_c*gain_f*Amp(k)/kM;
        if (A <= 0.237)
        gain_d = NaN;
        elseif (A > 0.9)
        gain_d = 2/pi*(asin(0.9/A)+0.9/A*sqrt(1-(0.9/A)^2));
        else
        gain_d = 1;
      end
    else
      gain_d = NaN;
    end
    % Include Wrist Nonlinearity
    if and(gain_d < 1000 , gain_d > -1000) % Check for NaN
        A = A*gain_d;
        k_A = 108.4*A^3-245.86*A^2+183.81*A-34.668;
        p_A = 49.77*A^3-102.77*A^2+72.33*A-8.6433;
        num3 = k_A*[1 9.50]; den3 = 10.70*[1 p_A];
        SYS3 = tf(num3,den3);
      [gain_e,phase_e] = bode(SYS3,omega);
      phase_e = phase_e*pi/180;
    else
```

```
        gain_e = NaN;
        phase_e = NaN;
      end
      gain = gain_a*gain_b*gain_c*gain_d*gain_e*gain_f;
      R = 1/(gain);
      phase = phase_a-phase_b+phase_c-phase_e-phase_f;
      realimag(k) = -R*cos(phase)-R*i*sin(phase);
    end
    figure(1); clf; plot(realimag);
    hold on; plot(realimag,'.');
    text(real(realimag),imag(realimag),num2str(Amp'/kM));

    % Plot nyquist plot of plant
    figure(1); hold on; plot(Re2,Im2,'r');
    plot(Re2_int(1:3*index_int),Im2_int(1:3*index_int),'rx');
    for k = index_int:2:2*index_int,
      temp = k-index_int;
      text(Re2_int(k),Im2_int(k),int2str(cross_freq_int+temp));
    end

    % Create unit circles
      Real = zeros(91,1);
      Imag = zeros(91,1);
      for k = 1:91,
          Real(k) = -cos((k-1)*pi/180);
          Imag(k) = -sin((k-1)*pi/180);
    end
    figure(1); hold on;
      plot(Real,Imag,'g--');

      axis([-1 1 -1 0]);
      axis square; grid
    xlabel('Real'); ylabel('Imaginary')

    omega = input('Input New Frequency (Enter 0 to Quit)  ');
  end
```

```
% Tabular_PPSSPFMD.m

% Run this script to load the tabular parallel-path single-signed
% pulse frequency modulation/demodulation describing function.

% Frequencies and amplitudes of data
Omega = [1 5 10 15 20 25 30 35 40]'; % (rad/sec)
Amp_kM = [13 14 15 20 25 30 40 60 80 120 180 240 300 360 420 500 600 800 1000];

% Phase lag data in degrees
Phase_Lag = [6 28 NaN NaN NaN NaN NaN NaN NaN
6 27 54 70 NaN NaN NaN NaN NaN
5   27  51  74  NaN   NaN   NaN   NaN   NaN
5   24  45  72  87  NaN   NaN   NaN   NaN
5   21  42  63  85  103    NaN   NaN   NaN
5   20  39  61  81  96  118    NaN   NaN
5   17  36  55  75  91  111    131    148
4   14  30  49  66  83  102    117    140
4   11  26  43  59  74  96  107    133
3   9   21  36  48  61  77  86  101
3   6   17  31  38  47  58  58  52
3   6   15  26  32  39  45  45  32
2   6   13  24  26  33  35  33  24
2   5   12  19  25  30  31  21  20
2   5   11  18  23  29  27  20  17
2   4   10  17  21  25  24  18  15
1   5   11  17  22  25  25  18  15
1   5   11  18  25  28  25  15  16
0   5   12  18  24  28  26  14  16];
Phase_Lag = Phase_Lag*pi/180; %Convert to radians

% Gain data
Gain = [0.79 0.76 NaN NaN NaN NaN NaN NaN NaN
0.81 0.84 0.86 0.8 NaN NaN NaN NaN NaN
0.87    0.94    0.9 0.84    NaN   NaN   NaN   NaN   NaN
0.95    0.98    0.99    1.04    1.29    NaN   NaN   NaN   NaN
0.96    0.99    1.03    1.08    1.04    1.16    NaN   NaN   NaN
0.99    1   1.04    1.12    1.16    1.18    1.23    NaN   NaN
0.99    1   1.04    1.11    1.13    1.16    1.14    1.12    1.14
1   1   1.03    1.06    1.04    0.99    0.96    0.94    0.89
1   1   1.02    1   0.97    0.92    0.86    0.85    0.76
1   1   1   0.97    0.89    0.84    0.72    0.82    0.86
1   1   0.99    0.95    0.85    0.74    0.65    0.55    0.55
0.99    0.99    0.98    0.93    0.83    0.78    0.69    0.63    0.61
0.99    0.99    0.98    0.92    0.83    0.72    0.64    0.55    0.43
0.99    0.98    0.95    0.9 0.81    0.75    0.6 0.77    0.59
```

```
0.98  0.97  0.94  0.89  0.8 0.74  0.59  0.51  0.63
0.94  0.94  0.91  0.86  0.78  0.68  0.62  0.6 0.72
0.8 0.8 0.77  0.73  0.67  0.56  0.47  0.51  0.6
0.62  0.61  0.59  0.55  0.49  0.45  0.36  0.48  0.43
0.53  0.53  0.51  0.47  0.43  0.36  0.38  0.49  0.51];
```

REFERENCES

[1] Peretto, P., 1992, *An Introduction to the Modeling of Neural Networks*, Cambridge University Press, Chapter 2.

[2] Pavlidis, T., and Jury, E. I., 1965, "Analysis of a New Class of Pulse-Frequency Modulated Feedback Systems," *IEEE Transactions on Automatic Control*, Vol. AC-10, pp. 35-42.

[3] Horowitz, P., and Hill, W., 1989, *The Art of Electronics*, Cambridge University Press, Second Edition, Chapters 4 and 9.

[4] Horn, D. T., 1987, *Oscillators Simplified, with 61 Projects*, Tab Books Inc., Blue Ridge Summit, PA, Chapter 7.

[5] Nack, J., 1988, "Amplifiers," *Interfacing Sensors to the IBM PC*, W. J. Tompkins et al., eds., Prentice-Hall P T R, Englewood Cliffs, NJ, pp. 30-33.

[6] Frank, P. M., and Turski, K. K., 1985, "Design of Pulse Frequency Modulated (PFM) Control Systems," *Applied Digital Control*, S. G. Tzafestas, ed., Elsevier Science Publishers B.V., North-Holland, pp. 225-252.

[7] Li, C. C., and Jones, R. W., 1963, "Integral Pulse Frequency Modulated Control Systems," *Proc. 2nd Congress IFAC*, pp. 186-195.

[8] Lawrence, S., 2000, Personal Correspondence, University of Utah, Salt Lake City, UT.

[9] Chaffin, D. B., Andersson, G. B. J., and Martin, B. J., 1999, *Occupational Biomechanics*, John Wiley & Sons, Inc., New York, Third Edition, Chapter 2.

[10] Dymkov, V. I., 1967, "Periodic States in Pulse-Frequency Systems," *Automatica*, 11, pp. 1708-1714.

[11] Khalil, H. K., 1996, *Nonlinear Systems*, Prentice-Hall, Upper Saddle River, NJ, Second Edition, Chapter 10.

[12] Gossett, J. H., Clymer, B. D., and Hemami, H., 1994, "Long and Short Delay Feedback on One-Link Nonlinear Forearm with Coactivation," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24(9), pp. 1317-1327.

[13] Franklin, G. F., Powell, J. D., and Emami-Naeini, A., 1994, *Feedback Control of Dynamic Systems,* Addison-Wesley, Reading, MA, Third Edition, Chapter 6.

[14] Fukuyama, A., 2001, "Mathematical Modeling of the Utah Artificial Arm," Master's Thesis, University of Utah, Salt Lake City, UT.

[15] Colton, M. B., 2001, "An Experimental NeuroElectric Prosthetic Arm Control System," Master's Thesis, University of Utah, Salt Lake City, UT.

[16] Stein, R. B., 1982, "What Muscle Variable(s) Does the Nervous System Control in Limb Movements?," *The Behavioral and Brain Sciences,* 5, pp. 535-577.