بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيمِ

In the Name of Allah,
the Compassionate, the Merciful,

THE EFFECTS OF ANTI-ALIASING FILTERS ON SYSTEM IDENTIFICATION

A thesis presented in partial fulfilment of the requirements for the degree of Master of Technology in Production Technology Department at Massey University.

Mohammad Ali Sadr-Nia

1992

CONTENTS

ACKNOWLEDGEMENTS

I find no appropriate words to express my deepest sense of gratitude to almighty God, whose blessing did not let me deviate from the right direction even through trials and tribulations.

I would like to thank my chief supervisors, Dr R. I. Chaplin and Dr H. C. Bakker for their excellent guidance and close supervision during the course of the work and preparation of this thesis.

I wish to thank Mr. I. Noell and all of the other staff members of Production Technology Department for their advice and encouragement.

I am particularly grateful to the Scholarship Dept. of Ministry of Culture and Higher Education of Islamic Republic of Iran for their award of a full fee scholarship and for their excellent assistance.

My special gratitude is to my wife who gave me great support and assistance.

AIMS and OBJECTIVES of the PROJECT

Keywords: System identification, Simulation, Anti-aliasing filtering.

In order to satisfy the Nyquest criterion for sampling, signals must be band limited. This is usually achieved by using low pass analog filters which must be placed before sampling (these filters are called anti-aliasing filters). These filters have some effect on the identification of systems. The aim of this project was to determine these effects.

The objectives were to:

1) Choose systems with different natural frequencies
2) Simulate these systems
3) Apply a PRBS (Pseudo Random Binary Sequence) input and log the output
4) Sample this data
5) Transfer sampled data to MATLAB
6) Find the best model using the MATLAB identification toolbox
7) Simulate the filters
8) Pass the output data through these filters

9) Repeat (5) and (6)

10) Simulate additive measurement noise

11) Add noise to the output of the system

12) Repeat (8) and (9)

13) Compare the real data with model data

14) Calculate the error criterion (J) for each case

Finally, a suitable parameter was to be identified which could be used to design effective anti-aliasing filters.

SUMMARY

Research was conducted to determine the effect of anti-aliasing filters on the identification of dynamic systems. Systems were simulated in the continuous simulation package ESL. The system response to a PRBS (Pseudo Random Binary Sequence) was recorded. Simulated noise was added and passed through a number of simulated analog filters. The systems were identified using the MATLAB identification toolbox.

Two standard filters (Butterworth and Chebychev) were used with cut-off frequencies between ω_S (natural frequency of the system) and 20 times ω_S.

Results showed that carefully designed filters could improve the performance of the identification algorithm in the presence of non-white high frequency additive noise. However for noise free measurements the filters degraded the performance of identification algorithms. This performance could be observed in the identified models steady state error, overshoot and settling time when subject to a step input.

In the experiments performed, the lowest order (and in one case second order) filters with cut-off frequency of $\omega_n \cong 5\omega_S$, gave the best results.

CHAPTER 1

INTRODUCTION and LITERATURE REVIEW

In order to study the effects of anti-aliasing filters on the accuracy of model identification it is necessary to have an understanding of the following subjects:

I. Systems and Models

II. Identification

III. Signal Processing

IV. Simulation

These subjects are introduced in this chapter.

1.1 SYSTEMS AND MODELS

A *system* is defined as "a wide range of more or less complex objects, whose behavior we are interested in studying, affecting, or controlling" (Lennart Ljung, 1983). Some examples we could mention are:

• *An armature controlled dc-motor*: Where we would like to control speed of the motor using the input voltage applied to the armature.

• *An inverting amplifier circuit* : Where we would like to control the output voltage by changing the resistance.

• *A paper machine*: Where we would like to control the quality of the paper by changing consistency, temperature, speed, etc.

• *A telephone communication channel*: Where we would like to design a filter for the receiver to produce high voice quality.

• *A time series of data* (e.g., sales, unemployment, or rainfall figures): Where we would like to predict the future in order to act properly now.

A *Model* is defined as the knowledge of the properties of a system. The model may be given in any one of several different forms, for example:

• *Mental or intuitive models*: Knowledge of the system's behaviour is summarized in a person's mind. Like a driver's model of an automobile.

• G*raphic models*: Properties of the system are summarized in a graph or in a table. An example could be a Bode diagram for the frequency response of a linear system.

• *Mathematical models*: There are sometimes mathematical relationships between variables, like Kirchhoff's law of voltages around a loop.

For many purposes only mental models are required, but for complex design problems mathematical model are necessary.

1.1.1 Obtaining A Model Of A System

There are two ways of building a mathematical model of a system.

(1) *physical modeling*: By looking inside the system to find the physical laws that govern the system's behaviour, a mathematical model can be constructed.

(2) *Identification*: Often because of incomplete of the knowledge of the system, direct modeling may not be possible. Furthermore, physical modeling can be quite time-consuming and may produce an unnecessarily complex model. In such cases, signals produced by the system can be used to construct a model. This procedure is called *identification.*

1.1.2 Model Classification

System models can be classified according to the type of equations used to describe them. First we can distinguish between *continuous*, *discrete*, and *hybrid* systems. A continuous system is one for which variables change continuously with respect to time. In a discrete-time system, variables change only at distinct (finite) instants of time. In some physical applications, both discrete-time and continuous variables may exist; thus, they are termed *hybrid* systems.

In some systems dependent variables are functions of more than one variable and will have *partial derivatives* instead of an *ordinary derivative*.

Models can also be distinguished by whether they have *distributed-parameters* or *lumped-parameters*. In a distributed-parameter model, the dynamic behaviour is described in terms of *partial differential equations*, for example the expressions for voltage and current at all points along a transmission line. A *lumped-prameter model* is characterized at only a finite number of points and uses ordinary differential equations. For a physical inductor, a lumped model representation includes only a resistance, R, connected in series with an equivalent single inductance, L. When dealing with a lumped-parameter model, one may discretize in time, resulting in a discrete-time model that is described by a *difference equation.*

A further distinction between models is linear and nonlinear. *Linear systems* are those satisfying the following conditions: (a) multiplying system input by a constant results in multiplication of its output by the same constant, and (b) the system response to a number of inputs applied together (simultaneously) is the sum of individual responses when each input is applied individually. *Nonlinear systems* are the ones for which these two conditions do not hold.

Another classification is into deterministic and stochastic models. In a *stochastic model*, the relationships are described in terms of probabilities only whereas in a *deterministic* all relationships are certain (if known).

As a summary, models can be classified in 6 different ways depending on the system properties and model representations.

Property	Axis Range
time	dynamic, steady state
space	distributed, lumped, non distributed
signal type	continuous, hybrid, discrete
statistical nature	deterministic, stochastic
mathematical nature	linear, nonlinear, chaotic
model representation	differential, difference, behavioural

1.2 SYSTEM IDENTIFICATION

System identification is referred to as "the determination of a mathematical model for a system or a process by observing its input-output relationships" (Hsia, 1977). Using input-output data we find the parameters of a system model.

Over the last decade great progress has been made in system identification methods. This has been helped by the need to design better control systems. This is especially true where there are time-varying parameters in a plant and its environment. Here, adaptive systems are needed to maintain optimal performance.

Recently, there has been very much progress toward the application of system identification techniques to physiological, biomedical, ecological, transportation and sociological problems. In addition, the availability of modern theory and complex computational algorithms has caused the fast growth of system identification technology.

1.2.1 Formulation And Classification Of The System Identification

Shown in figure 1.1 is a system with input and output and the method of identification problem. The system model to be found is the mathematical equation that relates the input to the output at all times.

In order to obtain such a model, a variety of inputs fed into the system. Having observed the responses, the input-output data are then processed to determine the model. On the basis of the amount of prior knowledge about the system, the identification can be classified into two categories.

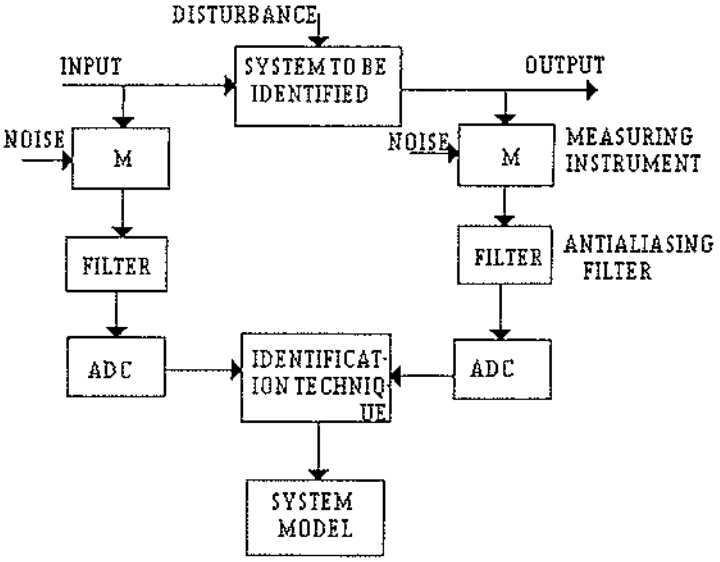


Fig. 1.1 Block Diagram Representation of the System Identification Problem.

1. The complete identification problem: this means that nothing is known about the basic properties of the system before hand. This type of the problem is also referred to as a *black box* problem.

2. Partial identification problem: in this category, some basic characteristics of the system, such as linearity, bandwidth, and so on, are assumed to be known. However, the order of the equations or the values of some coefficients are unknown. This situation is also called a *gray box* problem.

Fortunately, in practice, the majority of engineering systems are of the latter type. In many cases, the structure of the system is known and only a set of parameters in the model equation is left to be determined. Thus the modeling problem is reduced to that of *parameter identification.*

From the viewpoint of system theory, the determination of the unknown parameters from the exact input-output data is possible. In reality, however, the input-output data are corrupted by measurement noise and the determination of system parameters is essentially a statistical-estimation problem.

The procedures for carrying out system identification can be divided into the following steps:

1. Specify and parameterize a class of mathematical models that represents the system to be identified.

2. Apply an appropriately chosen test signal (PRBS was used in the experimental phase of this project) to the system and record the input-output data. If the system is in continuous operation and a test signal is not permitted, then we must use the normal operating data for identification.

3. Perform the parameter identification to select the model that best fits the statistical data.

4. Perform a validation test (In the experimental phase of this project real data was compared with model data with the same input) to see if the model chosen adequately represents the system with respect to final identification objectives.

5. If the validation test is passed, the procedure ends. Otherwise, another class of model must be selected and steps (2) through (4) performed until a validated model is obtained (Hsia, 1977).

To give a better feeling for the role identification plays in applications two examples are considered.

EXAMPLE 1. (Prediction of Power Demand) The demand for electricity from a power system changes over time. These changes in somewhat predictable with time of

day and over the week, month, or year. Efficient electricity production needs good predictions of the load some hours ahead.

Prediction of the power demand needs a model of its random component. This random component itself may depend on circumstances, e.g., the weather, that may vary with time. Therefore using a predictor that *adapts* itself is desirable.

The above is an example of *adaptive prediction.* Sometimes the predictions themselves may be of interest as the following example demonstrates.

EXAMPLE 2. (Digital Transmission of Speech) The transmission of speech over a communication channel is increasingly done digitally. The transmission line has limited capacity so efficient use is important. If the *next sample* is predicted at both the transmitter and the receiver, one need transmit only the difference between the actual and the predicted value (*prediction error*). Because of the prediction error is smaller than the signal itself, it needs fewer bits to transmit. This technique is known as *predictive coding* in communication theory. Prediction of the next value depends on the character of the transmitted signal. For speech, this character varies with the different sounds being pronounced. Efficient use of the predictive encoding requires the model used by the predictor to be adaptive.

The area of *adaptive control* is concerned with the study and design of controllers and regulators that adjust to varying properties of the controlled object. This is currently a very active research area (Lennart Ljung, 1983).

1.2.2 Parameter Estimation Methodology

There are a number of parameter estimation techniques that have been applied to the identification problem. They include the methods of maximum likelihood, least squares, cross-correlation, instrumental variable, and stochastic approximation.

In all estimation techniques, the optimal model is found by minimising some error criterion. Error criterion can be defined in many ways leading to many possible optimal

models. Some error criteria use; difference between parameter estimates and the true values (*parameter error*), the difference between the system output and the model output for similar inputs (*output error*), or, the difference between the model equation and the measured input and output data (*equation error*).

There are two modes in which identification can be done. In *off-line identification,* the input-output data is first observed and then model parameters are estimated based from the complte data set. In *on-line identification*, the parameter estimates are recursively calculated at every sampling interval from the new data and used to correct and update the existing estimate. This is termed recursive identification, real-time identification, adaptive algorithm, or sequential estimation.

There are numerous system identification methods, both off-line and on-line. One method for classifying them is:

I. Classical Methods:(mostly off-line)
   (a) Frequency Response Identification
   (b) Impulse response identification by deconvolution
   (c) Step response identification
   (d) Identification from correlation functions
   (e) Identification using spectral density functions
   (f) Pseudo random binary sequences (PRBS)

lll. Model Adjustment Techniques:
   (a) Least-squares (recursive)
   (b) Generalized least squares (recursive)
   (c) Instrumental variables
   (d) Maximum likelihood (recursive)

1.2.3 Classical Methods Of System Identification

A number of classical methods will now be described. They are *classical* only in that they have been in use for longer than the *modern* techniques.

1.2.3.1 Frequency Response Method

The frequency response method is based upon the Bode diagram of frequency response.

In this method, sine-waves with different frequencies are used as inputs and the steady-state output is observed. Both the magnitude ratio and the phase shift between the output and input are measured. Let G(s) be the transfer function of the system, then G(jω) is the frequency response, i.e.,

$$G(j\omega)=M(\omega).e^{j\phi(\omega)}=\frac{Y(j\omega)}{X(j\omega)} \qquad (1.01)$$

where M is the ratio of the magnitudes, and ϕ is the phase shift between the output and the input.

The plot of $M(\omega)$ and $\phi(\omega)$ against ω(log scale) can then be used to estimate the various break-frequencies (poles and zeros) of the transfer function (figure1.2).

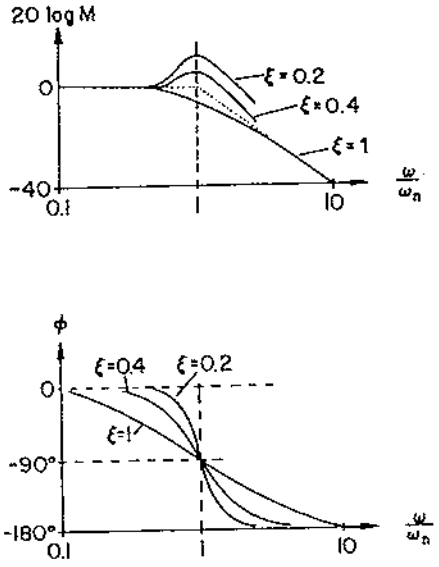


Fig. 1.2 Frequency response curves for a second-order system given by $G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$.

After Sinha and Kuszta, 1983.

1.2.3.2 Identification From Step Response

The simplest input for identification is a step input; produced by suddenly switching off (or on) an input voltage (or current), or by suddenly opening (or closing) an input valve, etc. An ideal step is physically impossible to construct, but the approximation is close if the rise-time of the step input is much shorter than the period of the highest frequency in the system.

If the system model is of the first-order, only two pieces of information are required: (i) the steady-state response to the step input and (ii) the time constant.

For a second-order system model (with two poles and no zero), there are two situations: (1) when the two poles are real and (2) when the poles are a complex conjugate. To find these from measurements of (a) steady-state response, (b) maximum overshoot, (c) time required to reach the first-peak, and (d) time required to reach 50% of the steady-state value(for overdamped systems) can be easily derived.

Consider the underdamped second-order system described by the transfer function

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \qquad \text{for } \zeta<1 \qquad (1.02)$$

The response of this system to a unit step is given by

$$c(t) = 1 - \frac{1}{\beta} e^{-\zeta\omega_n t} \sin(\omega_n \beta t + \theta) \tag{1.03}$$

where

$$\beta = \sqrt{1-\zeta^2} \qquad \text{and} \qquad \theta = \tan^{-1}\frac{\beta}{\zeta}$$

It can easily be shown that the maximum response is given by

$$M_{pt} = 1 + e^{-\zeta\pi/\beta} \tag{1.04}$$

and it occurs at

$$T_p = \frac{\pi}{\beta\omega_n} \tag{1.05}$$

Hence, by measuring M_{pt}, the value of the damping ratio, ζ, can be calculated from equation (1.04). Finally equation (1.05) can be used to determine the undamped natural frequency, ω_n. See Fig. 1.3. (Dorf, 1980).

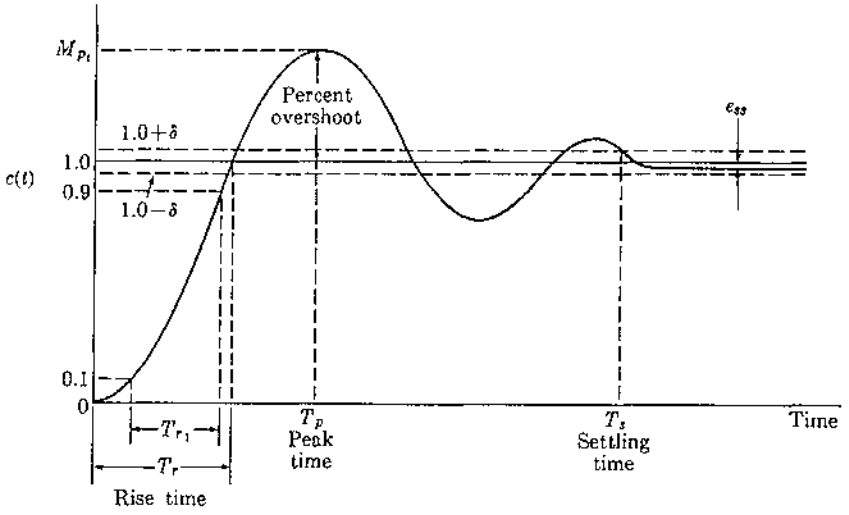


Fig. 1.3. Step response of a control system (Eq. 1.3). After Dorf, 1980.

1.2.3.4 Correlation Method

The techniques of step input or sinusoid input are applicable only to the identification of linear systems. In addition, response testing with these input functions is not always practical because of the existence of system noise. Therefore we need techniques that

can be used to identify both linear and nonlinear systems which will not be affected by noise.

The correlation method is based on applying a random input to the process, and takes a simpler form if the input is a white noise input.

The input-output relationship for a linear time invariant system may be written as

$$y(t)=\int_0^\infty w(t-\tau)x(\tau)d\tau = \int_0^\infty w(\tau)x(t-\tau)d\tau \qquad (1.06)$$

The cross-correlation between the input and output is obtained as

$$\begin{aligned}\phi_{yx}(\theta) &= E[y(t).x(t-\theta)], \text{ assuming stationariness}\\ &= E[\int_0^\infty w(\tau)x(t-\tau)d\tau.x(t-\theta)]\\ &= \int_0^\infty w(\tau).E[x(t-\tau)x(t-\theta)]d\tau\\ &= \int_0^\infty w(\tau)\phi_{xx}(\theta-\tau)d\tau \qquad (1.07)\end{aligned}$$

For the particular case when the input x(t) is white noise, we have

$$\phi_{xx}(\theta-\tau) = \delta(\theta-\tau), \qquad (1.08)$$

where $\delta(t-k)$ represents the unit impulse or delta function occurring at t=k.

Hence, for this case,

$$\begin{aligned}\phi_{yx}(\theta) &= \int_0^\infty w(\tau).\delta(\theta-\tau)d\tau\\ &= w(\theta) \qquad (1.09)\end{aligned}$$

Thus $\phi_{yx}(\theta)$ is the same as the impulse response of the system at $t=\theta$.

1.2.3.5 Spectral Density Functions

In the frequency domain, the response of a linear system is characterised by the frequency response function $H(j\omega)$. This function is the Fourier transform of the impulse response h(t). For deterministic signals, the Fourier transforms of input and output, $X(j\omega)$ and $Y(j\omega)$ respectively, are related by

$$Y(j\omega) = H(j\omega).X(j\omega)$$

The amplitude gain at any frequency ω, defined as the ratio (output amplitude) / (input amplitude), is $|H(j\omega)|$. At the same frequency, since power is proportional to (amplitude)^2, the power gain, defined as the ratio (output power) / (input power) is $|H(j\omega)|^2$. For systems with real parameters, $H(-j\omega)$ is the complex conjugate of $H(j\omega)$. Hence $|H(-j\omega)|$ is identical to $|H(j\omega)|$, and the power gain is thus an even function of frequency.

If the input to this system has a power spectrum $\phi_{xx}(\omega)$ then the power spectrum $\phi_{yy}(\omega)$ of the output signal y(t) is given by

$$\phi_{yy}(\omega) = H(-j\omega)H(j\omega)\phi_{xx}(\omega) = |H(j\omega)|^2\phi_{xx}(\omega)$$

For system identification purpose, knowing $\phi_{xx}(\omega)$ and $\phi_{yy}(\omega)$, $|H(j\omega)|$ can be found, and thus the magnitude curve of the Bode plot can be drawn (See Schwarzenbach and Gill, 1984).

1.2.3.6 Pseudo Random Binary Sequences (PRBS)

One of the most interesting and useful signals for system identification work is a pseudo random binary sequence (PRBS). This is a practical white noise signal that can easily be generated by digital circuit, or digital computer. The PRBS is a periodic sequence that takes on only two values. The times at which transition can occur are multiples of a specified time interval, Δt, and the state for any succeeding interval is nearly independent of the state in any preceding interval.

An example of such a signal is shown in figure 1.4. This signal has a periodic autocorrelation function shown in figure 1.5. We see that the autocorrelation function closely approximates the delta function of an ideal white noise. The approximation can be adjusted by changing N and Δt.

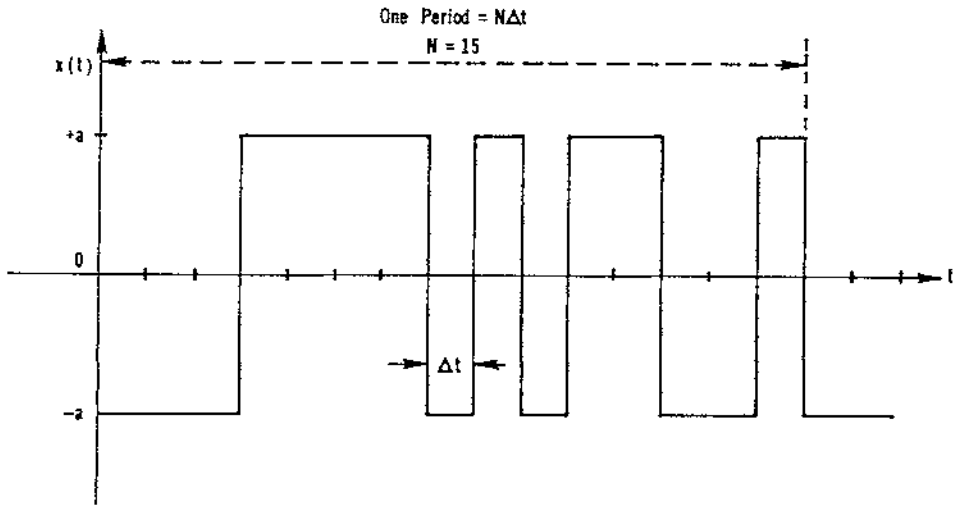


Fig. 1.4 A 15-bit Pseudo-random Binary Sequence (PRBS). After Hsia, 1977.

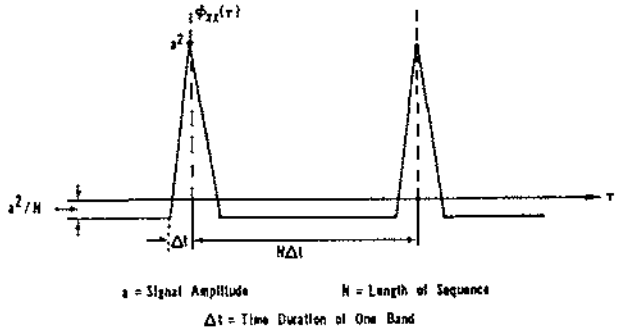


Fig. 1.5 Autocorrelation Function of PRBS. After Hsia 1977.

A computer program to generate the PRBS was written to use in the experimental phase of this project (see appendix 1).

Because of the very small perturbations, the PRBS signal can be applied for testing a system under operating conditions. This technique possesses some advantages over other techniques like (a) since the signal is periodic, a short recording time and minimal computational are required, (b) the method is highly immune to noise.

1.2.4 Off-Line Methods for System Identification

In this chapter will be discussed one off-line method (Least-squares) for estimating the parameters of a linear model from the input-output data of a single-input single-output system and interested reader is referred to Sinha and Kuszta, 1983 to find more

detail about both this technique and the Instrumental variables method (Both are used in the experimental phase of this project). It will be assumed that the order of the model is known a priori, and that equispaced samples of the input-output data are available.

1.2.4.1 Estimation of the parameters of a Discrete-Time Model from Noise-Free Input-Output Data

Consider the single-input single-output system shown in Fig. 1.6. Using z-transforms, the input-output relationship is given by

$$\frac{X(z)}{U(z)} = H(z) = \frac{a_0+a_1z^{-1}+\ldots+a_mz^{-m}}{1+b_1z^{-1}+\ldots+b_nz^{-n}} \tag{1.10}$$

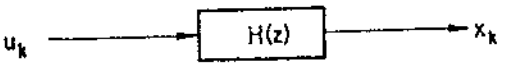


Fig. 1.6

In the form of a difference equation, equation (1.10) is written as below,

$$x_k = \sum_{i=0}^{m} a_i u_{k-i} - \sum_{i=1}^{n} b_i x_{k-i} \tag{1.11}$$

where

$$x_i \triangleq x(iT)$$
$$u_i \triangleq u(iT) \qquad i=1,2,\ldots$$

Thus, our problem is the determination of the parameters a_0 , a_1 , ... ,a_m ,b_1 , ... ,b_n from the input-output data.

Collecting the various sets of x_i and u_i, equation (1.11) may be concatenated to give the following matrix equation

$$\begin{bmatrix} u_k & u_{k-1} & \cdots & u_{k-m} & -x_{k-1} & -x_{k-2} & \cdots & -x_{k-n} \\ u_{k+1} & u_k & \cdots & u_{k-m+1} & -x_k & -x_{k-1} & \cdots & -x_{k-n+1} \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \cdot \\ u_{k+p-1} & u_{k+p-2} & \cdots & u_{k+p-m-1} & -x_{k+p-2} & -x_{k+p-3} & \cdots & -x_{k+p-n-1} \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_m \\ b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix} = \begin{bmatrix} x_k \\ x_{k+1} \\ \cdot \\ \cdot \\ \cdot \\ x_{k+p-1} \end{bmatrix} \tag{1.12}$$

or

$$\mathbf{A}_k' \theta = \mathbf{x}_k \tag{1.13}$$

where

$$\mathbf{A}_k' = \begin{bmatrix} u_k & u_{k-1} & \cdots & u_{k-m} & -x_{k-1} & -x_{k-2} & \cdots & -x_{k-n} \\ u_{k+1} & u_k & \cdots & u_{k-m+1} & -x_k & -x_{k-1} & \cdots & -x_{k-n+1} \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \cdot \\ u_{k+p-1} & u_{k+p-2} & \cdots & u_{k+p-m-1} & -x_{k+p-2} & -x_{k+p-3} & \cdots & -x_{k+p-n-1} \end{bmatrix} \tag{1.14}$$

$$\theta = \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_m \\ b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix} \triangleq \text{parameter vector} \tag{1.15}$$

and

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ x_{k+1} \\ \cdot \\ \cdot \\ \cdot \\ x_{k+p-1} \end{bmatrix} \triangleq \text{concatenated output vector} \tag{1.16}$$

It may be noted that if $\mathbf{A}_k'$ is a square nonsingular matrix (i.e., p=m + n +1 and det $\mathbf{A}_k' \neq$ 0), then one may obtain the parameter vector simply as

$$\theta = (\mathbf{A}_k')^{-1} \mathbf{x}_k \tag{1.17}$$

1.2.4.2 Weighted Least-Squares Estimates Of Parameters From Noise-Contaminated Data

The result derived in the previous section is of theoretical interest only, since the measurements are always contaminated with noise. In such practical situations, one may model the system as shown in Figure 1.7, where the measured output is shown as

$$y_i = x_i + n_i \tag{1.18}$$

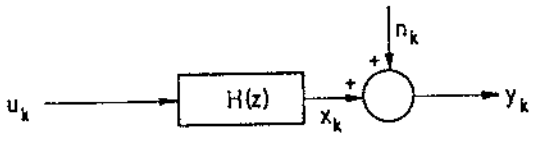


Fig. 1.7

It is assumed that $\{n_i\}$ is a white Gaussian noise sequence.

Substituting equation (1.18) into (1.11), we obtain

$$y_k = \sum_{i=0}^{m} a_i u_{k-i} - \sum_{i=1}^{n} b_i y_{k-i} + v_k = \phi_k^T \theta + v_k \tag{1.19}$$

where

$$\phi_k^T = [\ u_k \quad u_{k-1} \quad \cdots \quad u_{k-m} \quad -y_{k-1} \quad -y_{k-2} \quad \cdots \quad -y_{k-n}\ ] \tag{1.19a}$$

and

$$v_k = n_k + \sum_{i=1}^{n} b_i n_{k-i} \tag{1.20}$$

are called the output or equation errors.

Equation (1.19) may be concatenated, as before, to give

$$A_p \theta = y_p - v_p \tag{1.21}$$

where

$$\mathbf{A}_p = \begin{bmatrix} u_k & u_{k-1} & \cdots & u_{k-m} & -y_{k-1} & -y_{k-2} & \cdots & -y_{k-n} \\ u_{k+1} & u_k & \cdots & u_{k-m+1} & -y_k & -y_{k-1} & \cdots & -y_{k-n+1} \\ . & . & & . & . & . & & . \\ u_{k+p-1} & u_{k+p-2} & \cdots & u_{k+p-m-1} & -y_{k+p-2} & -y_{k+p-3} & \cdots & -y_{k+p-n-1} \end{bmatrix}$$

$\triangleq$ concatenated observation matrix (1.22)

and

$$\mathbf{y}_p = \begin{bmatrix} y_k \\ y_{k+1} \\ . \\ . \\ . \\ y_{k+p-1} \end{bmatrix} \triangleq \text{concatenated measurement vector} \qquad (1.23)$$

Because the presence of noise, it is now needed more than (m+n+1) equations to estimate the parameter vector from equation (1.21); i.e., now

$$p > m+n+1 \qquad (1.24)$$

Let us say the estimate of θ, based on the p sets of input-output data as $\hat{\theta}_p$. If we assume that $\hat{\theta}_p$ is the optimal estimate of the parameter vector, then the optimal estimate of the output vector, $\hat{\mathbf{y}}_p$, would be written

$$\hat{\mathbf{y}}_p = \mathbf{A}_p \hat{\theta}_p \qquad (1.25)$$

On the base of the minimisation a performance index, J, (Sinha and Kuszta, 1983 pp. 29-33) we have

$$\hat{\theta}_p = (\mathbf{A}_p^T \mathbf{A}_p)^{-1} \mathbf{A}_p^T \mathbf{y}_p \qquad (1.26)$$

The estimate given by equation (1.26) is called the *least squares estimate.*

A more general form is

$$\hat{\theta}_p = (\mathbf{A}_p^T \mathbf{w} \mathbf{A}_p)^{-1} \mathbf{A}_p^T \mathbf{w} \mathbf{y}_p \qquad (1.27)$$

where **w** is positive definite symmetric matrix. Note that making **w** = **I**, changes equation (1.27) into equation (1.26). The least squares solution is a special case of the present solution, which is called the *weighted least squares* solution. A number of important questions about equation (1.27) are now considered.

1.2.4.3 Conditions for the Existence of the Weighted Least-Squares Solution

First of all, we must know when a solution to this equation exists, as it requires the inversion of the matrix $\mathbf{A}_p^T \mathbf{w} \mathbf{A}_p$. In general, the matrix $\mathbf{A}_p^T \mathbf{A}_p$ will be nonsingular if the input sequence satisfies one of the following conditions:

(i) $\{u_i\}$ is a random sequence

(ii) $\{u_i\}$ is a pseudo random binary sequence(PRBS)

1.3 FILTERING

In many applications it is of interest to change relative amplitudes of the frequency components in signal or perhaps eliminate some frequency components entirely, a process referred to as *filtering*. Since the spectrum of output for linear systems is that of the input multiplied by the frequency response of the system, filtering can be accomplished using of such systems with an appropriately chosen frequency response. This represents one of the very important applications of linear time-invariant systems.

1.3.1 Butterworth And Chebychev Filters

For a given number of poles, and hence a given degree of filter, the Butterworth response provides a passband magnitude characteristic that is as flat as possible near $\omega=0$, at the expense of a slow transition from the passband to the stopband region. Because of this, Butterworth response is sometimes referred to as *maximally-flat*.

Figure 1.8 shows a set of Butterworth response curves, normalized to a 3db cut-off frequency of 1 rad s^{-1}. Here n is the order of the filter, which is equal to the number of poles in the transfer function. The higher the order of the filter the closer the magnitude response comes to the ideal box-car (brick-wall) characteristic.

In many applications some ripples in the passband can be tolerated. By allowing the passband magnitude response to ripple in a controlled way, the Chebychev filter trades

off the flatness of the passband response for a greater rate of cut-off in the transition region. A Chebychev filter is specified in terms of its number of poles (order) and the magnitude of its passband ripple.

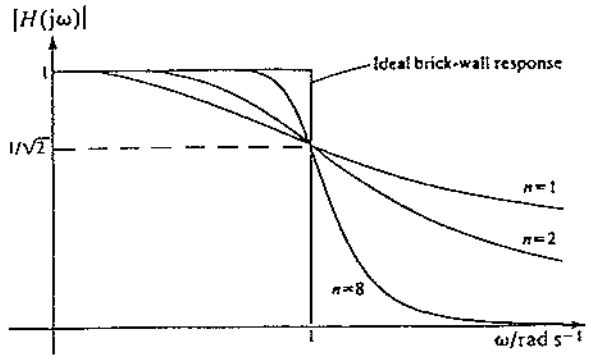


Fig. 1.8 Normalized Butterworth magnitude response curves

Figure 1.9 compares the response of a seventh-order Chebychev filter, with a passband ripple of 1.5 db, with that of seventh-order Butterworth filter.

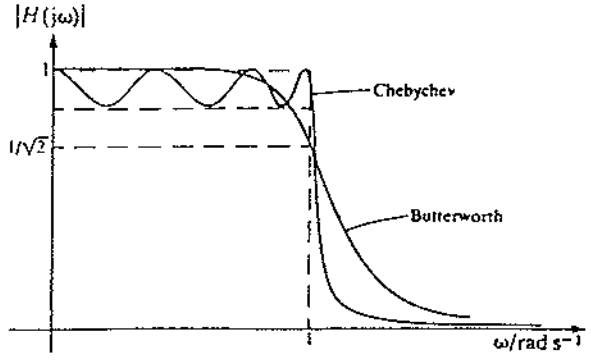


Fig 1.9 Comparison of Butterworth and Chebychev magnitude responses for n=7.

These analog lowpass filters are commonly employed as *anti-aliasing filters*, that are applied to continuous-time signals before analog-to-digital conversion. They are also suitable as interpolation filters to convert pulses, having amplitudes proportional to the value of the elements in the discrete time sequence, into signals that are continuous in time.

1.3.2 Sampling

Sampling theorem discusses about certain conditions which samples of a continuous-time signal must have to be recoverable.

In moving pictures, which consist of a time sequence of individual frames, when these samples are viewed at a sufficiently fast rate we get an accurate representation of the original continuously moving scene.

Much of the importance of the sampling theorem also lies in its role as a bridge between continuous-time signals and discrete-time signals.

1.3.3 Nyquist Criterion (Sampling Theorem)

When sampling a continuous-time signal $c_A(t)$ to produce the sequence$\{c_A(nT_s)\}$, we want to ensure that all the information in the original signal is held in the samples. To determine the condition under which there is no information loss, let us consider $c_A(t)$ to have a bandlimited spectrum, or one for which

$$C_A(j\Omega) = 0 \quad \text{for} \quad |\Omega| > \Omega_M \tag{1.34}$$

as shown in Fig. 1.10(a).

When $c_A(t)$ is sampled with sampling period T_s, then the spectrum of the sampled signal $C_s(j\Omega)$ is the periodic extension of $C_A(j\Omega)$ with period $2\pi/T_s$, as shown in Fig. 1.10(b). The form of $C_s(j\Omega)$ in the frequency range $[-\pi/T_s, \pi/T_s]$ is identical to $C_A(j\Omega)$ if

$$\pi/T_s > \Omega_M \qquad \text{or} \qquad T_s < \pi/\Omega_M \tag{1.35}$$

In this case, there is no overlap in the spectral components.

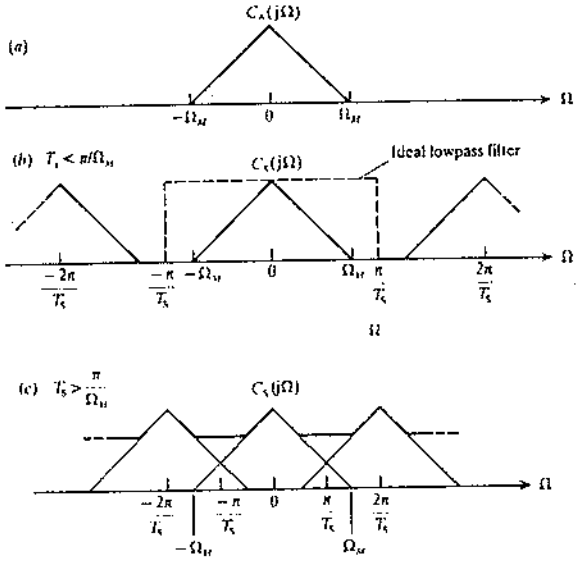


Fig. 1.10 Relationship between continuous-time spectrum of a signal and the spectrum of the discrete-time sequence obtained by sampling the signal with sampling period T_s. (a) Original spectrum of continuous-time signal; (b) spectrum of sampled sequence when $\Omega_M < \pi/ T_s$ (c) spectrum of sampled sequence when $\Omega_M > \pi/ T_s$. The latter case illustrates the aliasing error. After Kuc,1988.

If T_s is chosen to be greater than π/ Ω_M, spectral overlap occurs in the periodic extension, and the form of $C_s(j\Omega)$ in the range $-\pi/ T_s \leq \Omega < \pi/ T_s$ is then no longer similar to $C_A(j\Omega)$ as shown in Fig. 1.10(c). This overlap, caused by sampling at too low a rate, produces an irretrievable error in the spectral values, called *aliasing*. In other words aliasing refers to the fact that high-frequency components of a time function are *folded back* and appear as a low-frequency components if sampling rate is too slow. The true spectral shape is irretrievable since many different $C_A(j\Omega)$ functions can produce the same $C_s(j\Omega)$. Two possible candidates are shown in Fig. 1.11.

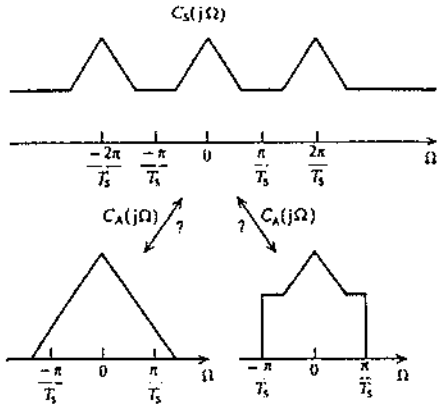


Fig. 1.11 Two candidates for the continuous-time spectrum when aliasing occurs. After Kuc,1988.

A useful sampling measure is the *sampling rate*, $f_s = 1/T_s$. Recalling that $\Omega = 2\pi f$, and defining f_M as the highest frequency component in the signal, then no spectral overlap will occur if

$$f_s > 2f_M \tag{1.36}$$

To prevent aliasing error, more than two samples are required per period of the highest frequency sinusoidal component present in the signal. The smallest sampling rate before aliasing occurs for a particular continuous-time signal is called the *Nyquist rate*.

The effect of aliasing errors is occasionally observed in films of moving cars in which the wheels appear to be turning in the direction opposite to that expected from the motion of the car.

1.3.4 Anti-Aliasing

Practically the frequency range of original continuous-time signal may be larger than the desired information. This commonly happens when a low frequency signal is contaminated by high-frequency noise. If this signal sampled by the Nyquist criterion for the desired analog signal, unwanted high frequency signals would cause aliasing errors to occur.

To prevent aliasing errors caused by these undesired high-frequency signals, an analog lowpass filter, called an *anti-aliasing filter,* must be used. This filter is used before sampling and reduces the power in the analog signal for the frequency range beyond $\Omega = \pi/T_s$. In practice, the spectral magnitude level for $\Omega > \pi/T_s$ should be less than 1% (- 40dB) of the desired signal spectrum to prevent significant aliasing.

For example, suppose an analog signal have power in a large frequency range, but *relevant* information is only in the frequency range $-\Omega_R \leq \Omega \leq \Omega_R$. It is asked to determine the sampling period T_s. The Nyquist criterion tells us that T_s must be less than π/Ω_R. If sampled at this rate, the higher frequency components in the analog signal

will be aliased into the relevant discrete-time signal. The anti-aliasing filter must satisfy two conditions:

Condition 1. The components of the analog signal with frequencies less than Ω_R must be negligibly attenuated by the filter.

Condition 2. The components for $|\Omega| \geq 2\pi/ T_S - \Omega_R$ must be attenuated strongly to prevent aliasing in the relevant range. After sampling, the components in this frequency range fall into the range $-\Omega_R \leq \Omega \leq \Omega_R$ when the periodic extension is formed (Kuc, 1988).

1.4 SIMULATION

Simulation is the process of understanding of the behaviour of a physical system by observing the behaviour of a model of the system. Thus, simulation is considered the science of experimenting with models. There are many purposes why simulation is valuable. For example, simulation is used to check and optimize the design of a system before its construction. Other purposes include analysis, tests of sensitivity, forecasting, safety, man-in-the-loop training and teaching.

1.4.1 Computer Simulation

Computer simulation is the technique of using computers to give, often in great detail, the performance of real systems. The purpose of the study is achieved by observing the model's behaviour under assumptions defined by the experimenter (user). There are some constraints in performing system simulation, for example the cost of model definition, software programming, data collection, etc.

1.4.2 Digital Representation Of Signals

As it is known, the variables for a continuous system have values for every point in time. On the other hand, a digital computer calculates values for the continuous

variables of a system at distinct points. Thus, a digital computer simulation of a continuous system is in fact a discrete-time system.

A digital computer represents variables with a finite number of bits, and accuracy is limited by the value of the least significant bit. In general, this means that the digital equivalent of a continuous signal at a given point in time will be ±1/2 of the value of the least significant bit, which is referred to as quantization error. The number of bits used, and hence the accuracy of the results, is determined by the word length of the digital computer being used for simulation.

Since numerical integration to solve the equation is fundamental, the next two sections will explain numerical integration techniques.

1.4.3 Numerical Integration

As noted before, the digital computer determines values for the continuous signals of the system being simulated by producing a series of discrete values. For example, the continuous function x(t) becomes a sequence of discrete values $x(t_0)$, $x(t_1)$, $x(t_2)$, . . .,$x(t_k)$, $x(t_{k+1})$, . . .,$x(t_n)$. Usually, the time interval between adjacent values is constant and represented by $T=t_{k+1} - t_k$.

Although the discretization error is often critical, the primary source of error in representing a simulation variable x(t) at $t=t_k$ is found in the method used to calculate derivatives, commonly referred to as numerical integration. The root of the problem can be seen by a careful look at Taylor's series expansion x_{i+1} in terms of x_i, which can be written in the form

$$x_{i+1} = x_i + \frac{Tdx_i}{dt} + (T^2/2!)\frac{d^2x_i}{dt^2} + \ldots + (T^n/n!)\frac{d^nx_i}{dt^n} \qquad (1.37)$$

where T is the time interval and dx_i/dt is the derivative of x_i at $t=t_i$.

This series gives the value of x at $t=t_{i+1}$ in terms of x_i and its derivatives. Taylor's series can be used to derive several numerical integration formulas, but more important, it is the criterion used for evaluating almost all numeral integration techniques.

As an example of a numerical integration method, consider the approximation by using only the first two terms of Taylor's series:

$$x_{i+1} = x_i + \frac{Tdx_i}{dt} \tag{1.38}$$

This equation is commonly referred to Euler's method, or the rectangular rule.

In order to illustrate the use of Euler's method to develop a digital computer simulation, consider the first-order differential equation

$$\frac{dx}{dt} + ax(t) = r(t) \tag{1.39}$$

with x(t=0) = 0 and r(t)=1, a unit step input. To develop a discrete equivalent suitable for programming on a digital computer, Eq.(1.39) can be written as below

$$\frac{dx_i}{dt} = -ax_i(t) + r_i(t) \tag{1.40}$$

Use Eq. (1.40) for dx_i/dt in Eq. (1.38), this yield

$$x_{i+1} = x_i - aTx_i + Tr_i \tag{1.41}$$

Note that the continuous system response for a = 1 and r(t) = 1 is an exponential rise, as shown by the solid curve in Figure 1.12 with $x_0 = 0$ (assumed). If a time interval of 2 sec (T=2) is selected, then $x_1= (x_0 - aTx_0 +Tr_0)=2$. Likewise, the value for x_2 is $x_2= (x_1 - aTx_1 +Tr_1) =0$, and in a similar manner, $x_3 = 2$, $x_4 = 0$,. . . The resulting output is as shown (labelled T=2) in Figure 1.12 and is obviously incorrect. Smaller values of T will cause the values of x_i to gradually approach the correct value, x=1.0. Note that T=1 results in output values of 1.0 at all intervals of time. As T becomes smaller , the discrete solution approaches the correct exponential rise shown in Figure 1.12; it is

obvious that the time interval for the digital simulation must be much shorter than the time constant of the system being simulated.

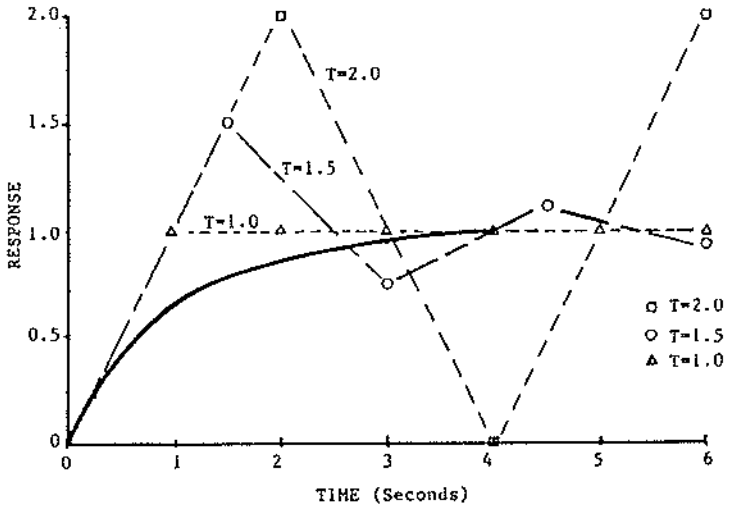


Fig. 1.12 Solutions to Eq. (1.41). After Kheir, 1988

The example problem in this section illustrates that a simple method exists for developing an equivalent difference equation for a differential equation. The example also illustrates the sensitivity of the solution to the size of the time interval T and clearly indicates a need to discuss the relationships between accuracy, computation time, and time interval.

1.4.4 Errors In Numerical Integration

In the example in the preceding section, a little thought would have indicated problems with $T \geq 2$. Since the time-constant for the differential equation in Eq.(1.39) is assumed to be 1 sec, trying to calculate values for larger than the time constant creates problems. Also, problems should have been anticipated from the use of only two terms of Taylor's series expansion because the contributions of the second and all higher order derivatives are missing. Thus Euler's method is referred to as a *first order*

numerical integrated technique. Techniques that include the effect of second order derivatives are referred to as second order methods and, in general, are more accurate than first-order techniques. Note that the inclusion of higher-order derivatives requires additional calculations for each value and will also need more computer memory.

The most commonly algorithm is one of the Runge-Kutta family; 2nd order, 4th order, etc, where the Taylor series coefficient have been modified to improve the accuracy of the truncated series. Other methods are more involved and are usually identified by their originator (e.g., Simpson, Adams, or Milne).

The previous discussion might lead one to believe that smaller and smaller integration (time) intervals will result in improved accuracy . This is not always the case. Note that the derivative is approximated by a difference (x_{i+1} - x_i) and that smaller values of T cause the values of x_i and x_{i+1} to approach each other. Since the digital computer represents variables with a finite binary sequence, each value will have a finite truncation error. If T is small enough so that x_{i+1} and x_i differ only by their truncation error, then (x_{i+1} - x_i) becomes zero and difference equation is no longer correct. Thus, the observation that accuracy improves directly with smaller and smaller time intervals has a limit, see Kheir (1988).

1.4.5 Simulation Software

In this section it will be discussed about some aspects of the software of digital computer simulation. In selecting a computer language, one would consider the following three areas to assess its efficiency: program execution speed, computer memory utilization, and language availability.

1. In viewpoint of *programming languages*, machine language and assembly language are at the lowest level.

Higher-level programming languages (also called general-purpose) allow the programmer to be removed from concerns related to machine operations. Examples of

these general-purpose languages are: BASIC, FORTRAN, COBOL, PASCAL,and ALGOL.

2. High-level *simulation languages* also involve compiler, similar to high-level programming languages, but are specifically used for simulation applications. Most simulation languages require less programming time; moreover, it is simpler to change a model after being written. It is also easier to debug such programs. A unique feature of simulation languages is their basic building blocks. Among the earlier simulation languages are MIDAS, DYSAC, DSL, GASP, MIMIC, DYNAMO, GPSS, SIMULA, CSSL(Continuous System Simulation Language), and CSMP. More recent simulation languages include ACSL (Advanced Continuous Simulation Language), SDL, ESL, SIMNON, SLAM, and SIMAN.

3. In its simplest form, a digital simulation package is a collection of routines (programs to be possibly compiled separately and then included as part of other program(s)). Today's simulation packages have not only full developmented, but, coupled with the available hardware, provide one of the most powerful tools for modeling and simulation activities in an interactive fashion.

The *interactive* (conversational) mode of simulation means that the simulation process on the computer be interrupted for the purpose of asking, or reporting to, the user. Based on the information available from the computer, the human partner decides on what is next to be modified, executed, etc.

Over the past two decades or so, attention has focused on developing simulation packages that are useful in many areas of applications. These packages have been packed on the educational processes and on activities in computer-aided design (CAD), computer-aided manufacturing (CAM), and, in general, in computer-aided engineering (CAE).

Simulation languages usually differ in their logic, construction, flexibility, and ease of usage. These differences include: (a) the basic objective of the language, (b) algorithms for generating random numbers, (g) program initialization , (h) data entry, (i) output

reports and (j) methods for data analysis. Most simulation languages, however provide the following standard capabilities: (a) structured data input, (b) time-advance mechanism, (c) acceptable random-number generators.

Resent years have seen the development of a tremendous number of discrete, continuous, and combined discrete-continuous simulation languages. Among the widely used software is ESL which was used in this project. Interested reader about the characteristics of the other simulation language is referred to Kheir 1988, pp. 681-694.

1.4.5.1 ESL

ESL is an advanced continuous-system simulation language (CSSL) that is being developed under contract from the European Space Agency. ESL is designed to be portable and to run on computers supporting a *FORTRAN 77* environment.

ESL is characterised by its advanced programming concepts. These include:

- Separation of model and experiment.
- Capability of building models from submodels.
- Optionally users may describe systems by a *Graphical* input program, rather than use conventional language form.
- Advanced discontinuity handling.
- Seven integration algorithms including three stiff methods.
- Parallel segmentation.

To implement ESL, an interpreter and a translator version of the language are required. The interpreter translate the user's program into H-code and the translator converts the H-code to FORTRAN-77. The entire system is written in FORTRAN-77, as indicated by J. L. Hay (1989).

ESL was developed on a PRIME 550 computer in the Simulation Laboratory of the University of Salford, England.

CHAPTER 2

MODEL SELECTION and SIMULATION

In order to determine the anti-aliasing filters effect on the identification of different types of systems, the following systems were modelled:

a) A simple second order linear system

b) A nonlinear chemical system

c) A nonlinear, discontinuous switching regulator

These systems covered a wide range of natural frequencies from 1 to 7300 rad s^{-1} and a range of nonlinear behaviour. For the linear system, a discrete model could be obtained analytically and compared directly with the identified models, however in the other two cases only the behaviour of the system and the identified models could be compared.

Each system will now be explained in detail.

2.1 SECOND ORDER LINEAR SYSTEM

This system has a low-pass second-order Butterworth response with a 3-dB cut off frequency at 1 rad s^{-1} =$(1/2\pi)$ Hz. The transfer function of the system is:

$$H(s) = \frac{1}{s^2 + 1.414s + 1}$$

2.2 NONLINEAR REACTION SYSTEM

A continuous stirred tank reactor system was considered in which bromine in solution combines to form bromine gas which escapes. The flow of bromine into the tank is given by:

flow . concentration = $F.C_i$,

and the flow out is F.C.

The reaction rate is given by $R = K_1.C^2$ where K_1 is a constant.

The differential equation governing the system is

$$\frac{dC}{dT} = F.C_i - F.C - K_1.C^2$$

where F = 0.5, K_1 = 1.0 and C_i has a value of 0.5 and is disturbed by a pseudo random binary sequence (PRBS).

2.3 SWITCHED-MODE POWER REGULATOR

This system was chosen (ESL Application Manual) because it exhibits discontinuous nonlinear behaviour with a high natural frequency. A switched-mode power regulator (SMPR) takes as input an un-regulated power supply voltage (V_S) and produces a stabilised output voltage (V_O) with minimal power loss. The level of the output is determined by a reference voltage (V_{ref}).

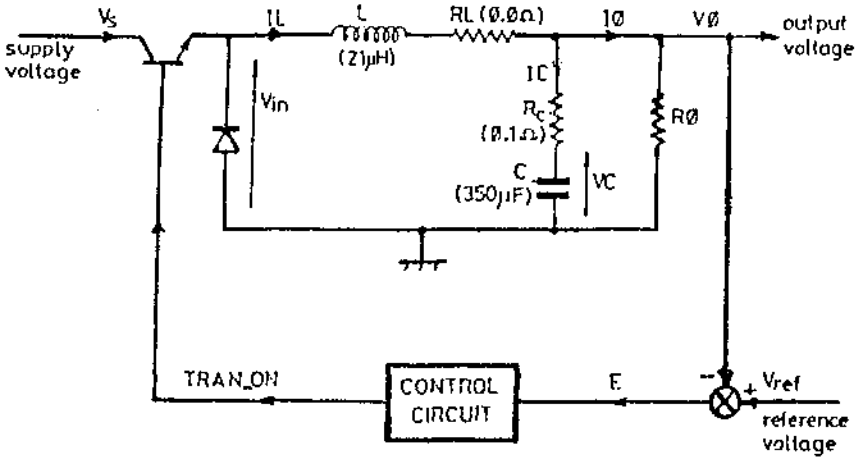


Fig. 2.1 SMPR power circuit

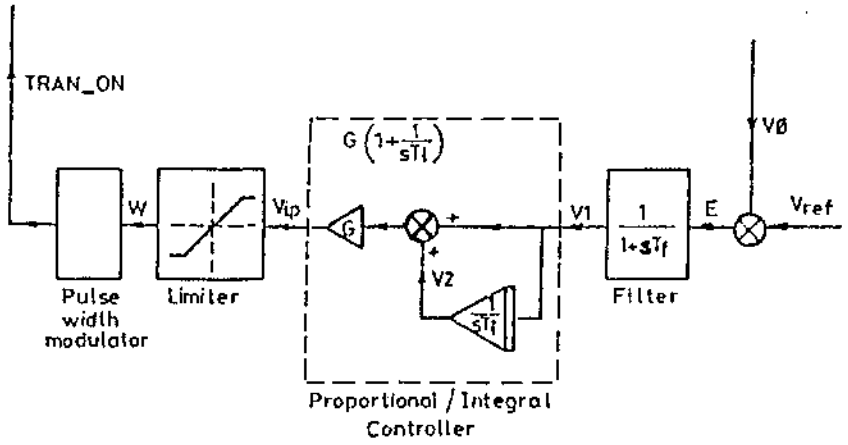


Fig. 2.2 SMPR control circuit. After ESL manual

The system is illustrated in figure 2.1 and 2.2. The equations governing the system are:

Power circuit (Fig 2.1)

$$I_o = \frac{(V_c + I_l * R_c)}{(R_o + R_c)}$$

$$V_o = I_o * R_o$$

$$I_c = I_l - I_o$$

$$V_c' = I_c / C$$

and for the control circuit (Fig. 2.2)

$V_2' = V_1 / T_i$

$V_{ip} = G * (V_1 + V_2)$

$V_1' = \frac{E - V_1}{T_f}$

The parameter values used in the simulation were:

Parameter	Value	Unit
G	1.0	none
V_s	70.0	V
L	21	μH
C	350	μF
R_l	0.0	Ω
R_c	0.1	Ω
R_o	25.0	Ω
T_i	450.0	μs
T_f	20.0	μs
f_o	80	KHz

The pulse width modulator (PWM) with sampling frequency f_o and mark space ratio w was modelled using a ramp generator and threshold detector, see figure 2.3 for a diagram illustrating the PWM operation.

See Appendix B for a more detailed derivation of the model and the derivation of its parameters.

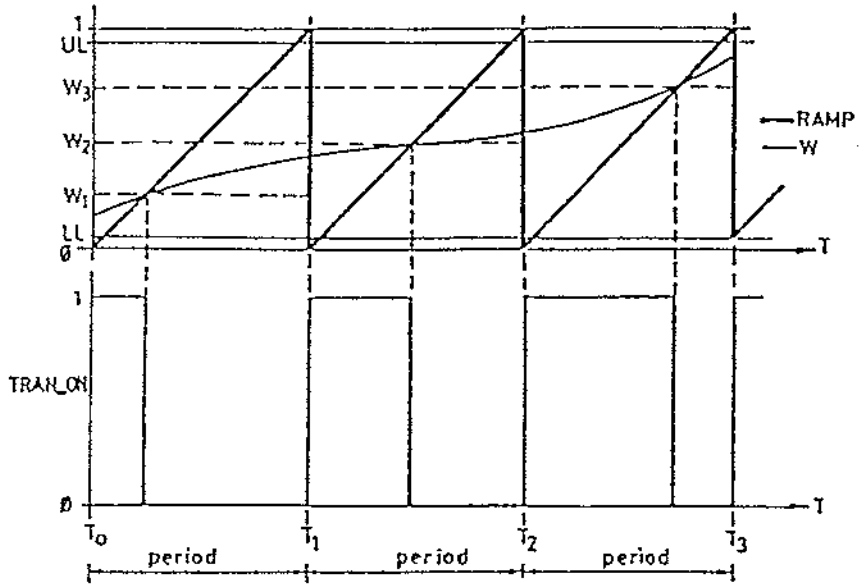


Fig. 2.3 Timing diagram for the PWM

2.4 SIMULATION STRUCTURE

ESL was used as the simulation package in this project. ESL (European Simulation Language) is an advanced continuous system simulation language, which was developed to meet the simulation requirements of the European Space Agency. for more information about ESL, see Literature Review section 1.4.5.1.

Each plant model and the anti-aliasing filter were simulated as ESL submodels (similar to procedures or subroutines). The input perturbations used were the output of a PRBS generator and the additive noise components were obtained from a sinusoid noise model. Both of these were also simulated as submodels.

These submodels were interconnected as shown in figure 2.4 and controlled by an Experiment control section.

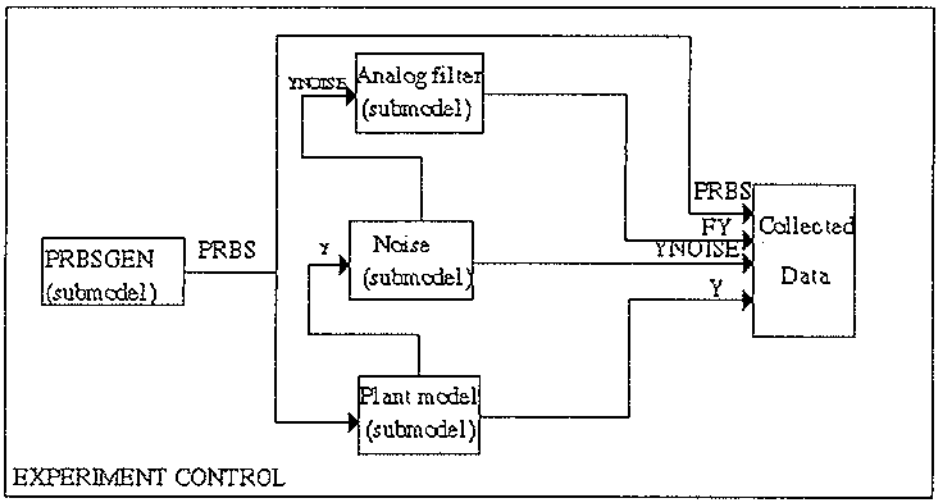


Fig 2.4 General Block Diagram of Simulation

Data recorded were:

Input perturbation	PRBS
Plant model output	Y
Plant model output with measurement noise	YNOISE
Filtered noisy plant output	FY

This data was stored on file for subsequent processing using the MATLAB IDENTIFICATION TOOLBOX. The simulation experiments were carried out at very short time steps (relative to the plant dynamics). In order to ensure that the collected data retained their analog nature, this data was then sampled at a lower rate for use in MATLAB.

2.4.1 Pseudo Random Binary Sequence Generator

A PRBS can be generated in ESL using the inbuilt random number generator to specify the next time the output is to change value. The magnitude and base frequency of the PRBS can be specified using two parameters to the submodel.

See appendix A for the ESL program for the PRBS generator.

2.4.2 Noise Simulation

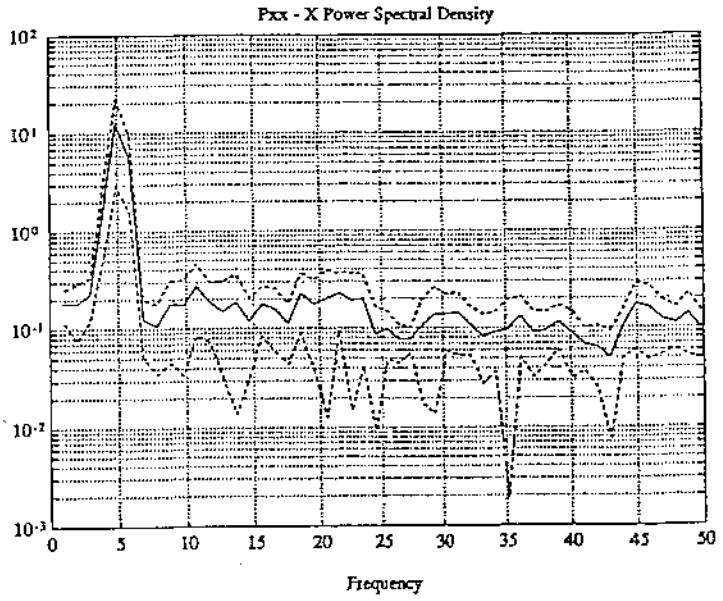
Measurement noise was simulated by adding to the output of each system a signal whose Power Spectral Density (PSD) could be controlled. This was generated using:

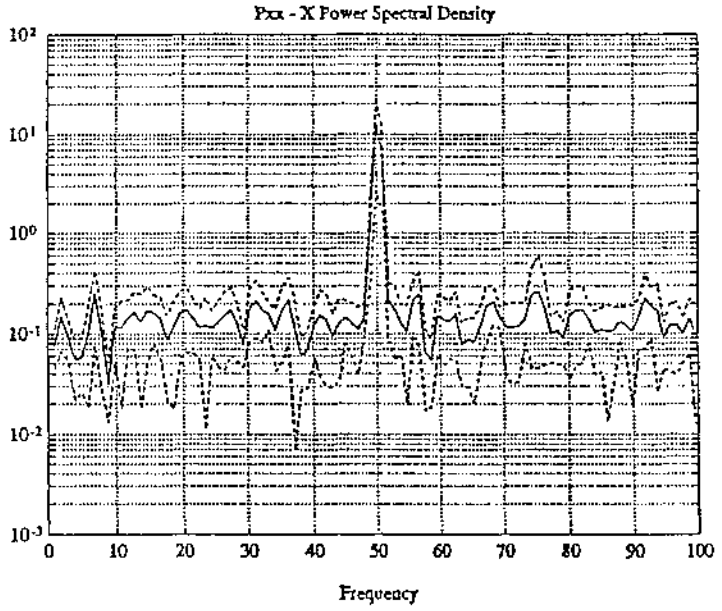
$$N = A_r \sin((2\pi f + \omega_r)t)$$

Where A_r and ω_r are random values whose range can be controlled.

This gives a signal whose PSD has a peak at frequency f, a spread about this frequency governed by ω_r and a magnitude controlled by A_r. See Fig 2.5 for PSD's of the noise signals.

The choice of f and ω_r were dependant upon the system being simulated and were chosen to ensure no overlap occurred in the spectral distribution of the noise and system.





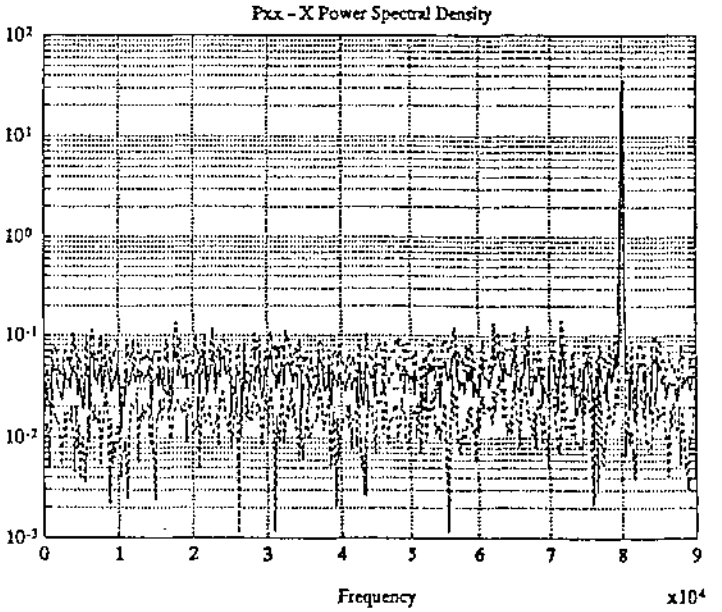


Fig 2.5 PSD of the noise for systems 1, 2 and 3 (top to bottom, respectively).

2.4.3 Filter Submodel

The matrix and vector features of ESL were used to implement the anti-aliasing filters as a common model where the individual filters were characterized by parameters read in from a file.

2.4.4 Linear System Submodel

ESL accepts dynamic models as equations in *natural form.* This means that the transfer function of the linear system model required conversion to natural form before entry to ESL. This was accomplished using the transfer operator. See appendix A for ESL program.

2.4.5 Nonlinear Reaction System Submodel

Prime Notation, explicit multiplication and power operators defined in the ESL manual, were used to change the nonlinear system to ESL program. See appendix A for the ESL program.

2.4.6 Switch Mode Power Regulator (SMPR) Submodel

The discontinuity features of ESL were used to model the pulse width modulator and the limiter. These were used with explicit multiplication defined in ESL manual. See appendix A for the ESL program.

CHAPTER 3

FILTER SELECTION and SIMULATION

For this work two standard filter types were selected; Butterworth and Chebychev. A number of different orders of filter were used with cut-off frequencies to cover the range from below the natural frequency of the system to almost twenty times the systems natural frequency.

The ESL program for the filter uses one submodel to implement a general state space form of the filter and ten submodels to implement each of the Butterworth and Chebychev filters. The filters used were Butterworth and Chebychev filters of 1st, 2nd, 3rd, 4th and 5th order.

The state space form of the filter transfer function was obtained as follows. The general form of a linear, analog filter is:

$$Y(s) = \frac{\sum_{i=0}^{m} a_i s^i}{\sum_{i=0}^{n} b_i s^i} X(s)$$

where $n \geq m$.

By assuming $b_n = 1$, we have

$$Y(s) = \frac{a_m s^m + a_{m-1}s^{m-1} + a_{m-2}s^{m-2} + \ldots + a_2 s^2 + a_1 s + a_0}{s^n + b_{n-1}s^{n-1} + b_{n-2}s^{n-2} + \ldots + b_2 s^2 + b_1 s + b_0} X(s) \qquad (3.1)$$

Let

$$Z(s) = \frac{X(s)}{s^n+b_{n-1}s^{n-1}+b_{n-2}s^{n-2} + \ldots +b_2s^2+b_1s+b_0} \tag{3.2}$$

Then we have,

$$Y(s) = (a_ms^m + a_{m-1}s^{m-1} + a_{m-2}s^{m-2} + \ldots +a_2s^2 +a_1s +a_0)Z(s) \tag{3.3}$$

Extending the numerator terms to n-1 by setting $a_{n-1}=a_{n-2}=\ldots=a_{m+1}=0.0$ we have,

$$y(t) = a_{n-1}\frac{d^{n-1}z}{dt^{n-1}} + a_{n-2}\frac{d^{n-2}z}{dt^{n-2}} +\ldots+ a_m\frac{d^mz}{dt^m} + a_{m-1}\frac{d^{m-1}z}{dt^{m-1}} +\ldots+ a_0z \tag{3.4}$$

Let $w_1=z$, $w_2=\frac{dz}{dt}$, $w_3=\frac{d^2z}{dt^2}$,... ,$w_n=\frac{d^{n-1}z}{dt^{n-1}}$

Therefore we have,

$$\frac{dw_1}{dt}=w_2$$

$$\frac{dw_2}{dt}=w_3$$

.

.

.

$$\frac{dw_{n-1}}{dt}=w_n$$

$$\frac{dw_n}{dt}= x(t) - b_{n-1}w_n - b_{n-2}w_{n-1} - \ldots - b_2w_3 - b_1w_2 - b_0w_1$$

In matrix form, this become:

$$\begin{bmatrix} \frac{dw_1}{dt} \\ \frac{dw_2}{dt} \\ . \\ . \\ . \\ \frac{dw_{n-1}}{dt} \\ \frac{dw_n}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ . & . & . & . & \ldots & . \\ . & . & . & . & \ldots & . \\ 0 & 0 & 0 & 0 & \ldots & 1 \\ -b_0 & -b_1 & -b_2 & \ldots & -b_{n-2} & -b_{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \\ . \\ w_{n-1} \\ w_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ 1 \end{bmatrix} x(t)$$

or

$$\mathbf{w}' = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -b_0 \cdots & -b_{n-1} \end{bmatrix} \mathbf{w} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} x(t)$$

This yields :

$$A = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -b_0 \cdots & -b_{n-1} \end{bmatrix} \qquad B = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$$

and from (3.4):

$$C = [\ a_0\ \ a_1\ \ a_2\ \ \cdots\ \ a_{n-1}\ ] \qquad D = 0.0$$

3.1 SIMULATION OF FILTERS

Tables of common coefficients are obtainable for the transfer functions. The coefficients for the Butterworth and Chebychev filters used are shown overleaf (Fig 3.1). These can be converted to the state space form above (eg. via MATLAB's tf2ss command).

This state space form was implemented in ESL and tested by obtaining the frequency domain responses of the filters and comparing this with the expected response.

Table 1 Coefficients of normalized Butterworth polynomials ($a_0 = a_n = 1$ for all n).

n	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
1									
2	1.4142136								
3	2.0000000	2.0000000							
4	2.6131259	3.4142136	2.6131259						
5	3.2360680	5.2360680	5.2360680	3.2360680					
6	3.8637033	7.4641016	9.1416202	7.4641016	3.8637033				
7	4.4939592	10.0978347	14.5917939	14.5917939	10.0978347	4.4939592			
8	5.1258309	13.1370712	21.8461510	25.6883559	21.8461510	13.1370712	5.1258309		
9	5.7587705	16.5817187	31.1634375	41.9863857	41.9863857	31.1634375	16.5817187	5.7587705	
10	6.3924532	20.4317291	42.8020611	64.8823963	74.2334292	64.8823963	42.8020611	20.4317291	6.3924532

Table 2 Butterworth pole locations.

$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
−1.0000000	−0.7071068 ±j0.7071068	−1.0000000	−0.3826834 ±j0.9238795	−1.0000000	−0.2588190 ±j0.9659258	−1.0000000	−0.1950903 ±j0.9807853	−1.0000000	−0.1564345 ±j0.9876883
		−0.5000000 ±j0.8660254	−0.9238795 ±j0.3826834	−0.3090170 ±j0.9510565	−0.7071068 ±j0.7071068	−0.2225209 ±j0.9749279	−0.5555702 ±j0.8314696	−0.1736482 ±j0.9848078	−0.4539905 ±j0.8910065
				−0.8090170 ±j0.5877852	−0.9659258 ±j0.2588190	−0.6234898 ±j0.7818315	−0.8314696 ±j0.5555702	−0.5000000 ±j0.8660254	−0.7071068 ±j0.7071068
						−0.9009689 ±j0.4338837	−0.9807853 ±j0.1950903	−0.7660444 ±j0.6427876	−0.8910065 ±j0.4539905
								−0.9396926 ±j0.3420201	−0.9876883 ±j0.1564345

Table 3 Coefficients of normalized Chebychev transfer functions (0.5 dB ripple, $a_n = 1$ for all n).

n	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
1	2.8627752									
2	1.5162026	1.4256245								
3	0.7156938	1.5348954	1.2529130							
4	0.3790506	1.0254553	1.7168662	1.1973856						
5	0.1789234	0.7525181	1.3095747	1.9373675	1.1724909					
6	0.0947626	0.4323669	1.1718613	1.5897635	2.1718446	1.1591761				
7	0.0447309	0.2820722	0.7556511	1.6479029	1.8694079	2.4126510	1.1512176			
8	0.0236907	0.1525444	0.5735604	1.1485894	2.1840154	2.1492173	2.6567498	1.1460801		
9	0.0111827	0.0941198	0.3408193	0.9836199	1.6113880	2.7814990	2.4293297	2.9027337	1.1425705	
10	0.0059227	0.0492855	0.2372688	0.6269689	1.5274307	2.1442372	3.4409268	2.7097415	3.1498757	1.1400664

Table 4 Chebychev pole locations (0.5 dB ripple).

$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
−2.8627752	−0.7128122 ±j1.0040425	−0.6264565	−0.1753531 ±j1.0162529	−0.3623196	−0.0776501 ±j1.0084608	−0.2561700	−0.0436201 ±j1.0050021	−0.1984053	−0.0278994 ±j1.0032732
		−0.3132282 ±j1.0219275	−0.4233398 ±j0.4209457	−0.1119629 ±j1.0115574	−0.2121440 ±j0.7382446	−0.0570032 ±j1.006405	−0.1242195 ±j0.8519996	−0.0344527 ±j1.0040040	−0.0809672 ±j0.9050658
				−0.2931227 ±j0.6251768	−0.2897940 ±j0.2702162	−0.1597194 ±j0.8070770	−0.1859076 ±j0.5692879	−0.0992026 ±j0.8829063	−0.1261094 ±j0.7182643
						−0.2308012 ±j0.4478939	−0.2192929 ±j0.1999073	−0.1519873 ±j0.6553170	−0.1589072 ±j0.4611541
								−0.1864400 ±j0.3486869	−0.1761499 ±j0.1589029

Fig. 3.1 Coefficients of normalized Butterworth and Chebychev transfer functions.

CHAPTER 4

EXPERIMENTS

This chapter describes the series of experiments that were designed to observe the effects of including an anti-aliasing filter in the measurement data stream. In order to determine these effects it was necessary to obtain the best identified model of each system. This was achieved by simulating each system and generating noise free outputs for the pseudo random binary input disturbance required by the identification methods (see figure 4.0). These noise free outputs were then sampled (with a sampling rate fast enough to ensure no aliasing) and used to obtain the identified models. It is known [refer to section 1.3.3] that the quality of the model so obtained depends upon the sampling interval so the experiments were repeated at various sampling rates.

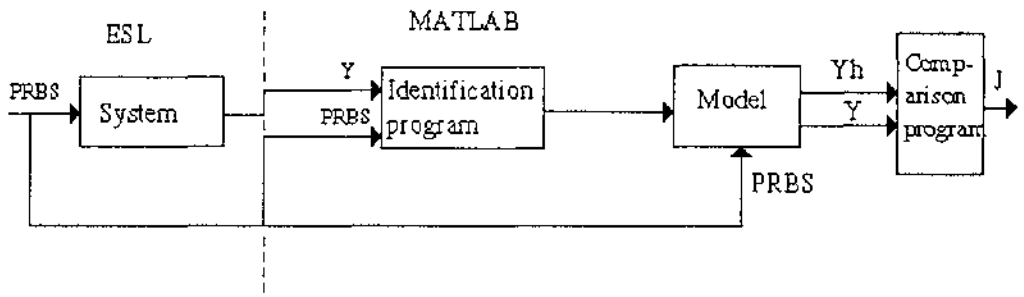


Fig 4.0 Block diagram of the noise-free experiments

The quality of the model was determined by comparing the performance of the model against the simulation using input and output data not used during the identification. An error criterion (J), which compared the real output data with that of the model under the same input (PRBS), was defined as follows:

$$J = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

where Y_i is real data from ESL,

$\hat{Y}_i$ is data from the model,

n is the number of data points (n=100)

J is the error criterion

Two sets of experiments were performed using the anti-aliasing filters. These were;

a) filtered, noise free data

b) filtered data with zero mean and bandwidth limited noise added prior to filtering (in effect measurement noise).

In section 4.4 the methods of the identification used are explained.

4.1 EXPERIMENTS WITHOUT FILTERING

To determine the effect of aliasing errors, an experiment was performed in which the system and noise blocks were included (fig 4.1). The error criterion J was calculated and compared with that obtained from the best model (noise-free).

This experiment would be expected to show that the model affected by aliasing was much worse than that of the noise-free model.

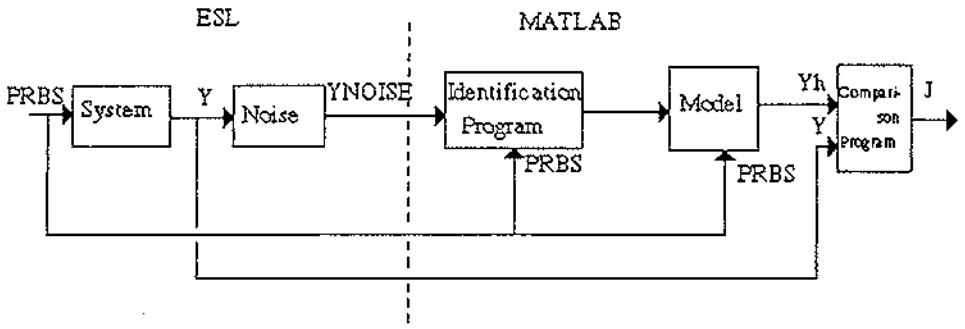


Fig 4.1 Experiments without filtering

4.2 FILTERED NOISE FREE EXPERIMENTS

In this experiment an anti-aliasing filter was included in the system but no noise was added. Butterworth and Chebychev filters of different order and cut-off frequency were used and the error criterion was calculated. The following block diagram shows this step.

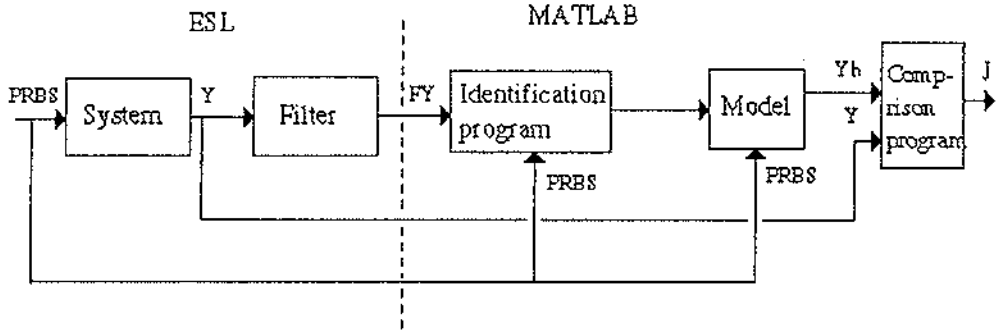


Fig 4.2 Block diagram of the filtered noise free experiments

4.3 FILTERED DATA WITH NOISE EXPERIMENTS

In this experiment noise was added into the system used in section 4.2. Various orders of Butterworth and chebychev filters were used. For each one an initial cut-off frequency (ω_n) below the natural frequency of the system was chosen. The simulation

was run and the error criterion calculated. This was repeated for increasing values of ω_n.

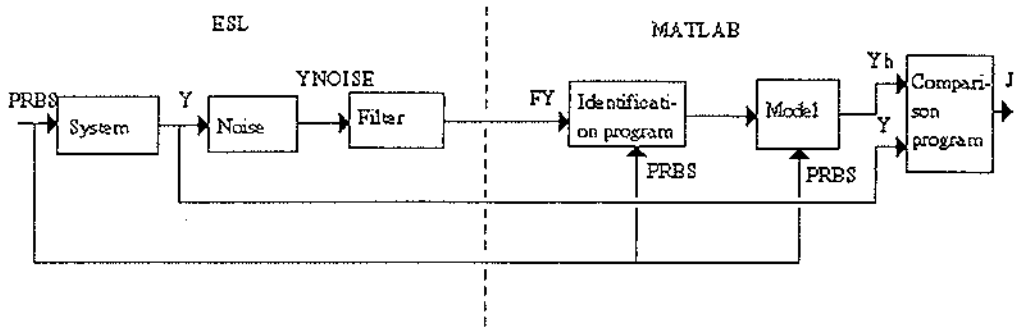


Fig 4.3 Block diagram of the filtered data with noise experiments

4.4 IDENTIFICATION METHODS USED IN EXPERIMENTS

The identification algorithms used were the ARX (Autoregressive eXogeneous variable) and the IV4 method (Instrumental Variables method four) (see MATLAB identification toolbox).

One thousand input-output data points were collected from the process as the input was changed in a random fashion between two levels. The sampling interval is known. Six hundreds data points were used for the identification and one hundred were used to calculate the error criterion, J.

We want to find an ARX model, which is usually written

$$A(q^{-1})y(t) = B(q^{-1})u(t-nk) + e(t)$$

where B and A are polynomials in delay operator q^{-1}:

$$A(q^{-1}) = 1 + a_1q^{-1} + \ldots + a_{na}q^{-na}$$

$$B(q^{-1}) = b_1 + b_2q^{-1} + \ldots + b_{nb}q^{-nb+1}$$

The reason for the term ARX is that the model is a combination of an autoregressive(AR) part, $A(q^{-1})y(t)$, and a control part, $B(q^{-1})u(t)$. The control signal is known as the eXogeneous variable, hence the X.

We want to fit to the data a model of the following form (the best form resulted from experiments) :

$$y(t) + a_1 y(t-T) + a_2 y(t-2T) = b_1 u(t-T) + b_2 u(t-2T)$$

or in the z domain

$$Y(z) + a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) = b_1 z^{-1} U(z) + z^{-2} b_2 U(z)$$

in the form of transfer function we have

$$\frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$$

MATLAB gives the coefficients b_1 , b_2 , a_1 ,and a_2 .

CHAPTER 5

RESULTS and DISCUSSION

This chapter includes, separately, the results for each system studied and its accompanying discussion. The input, output and comparison between real data and model data graphs for the best model and noise only experiment of each system are shown. For each system four graphs are shown in addition to four tables summarising values of the error criterion (J) with filter alone and noise plus filter. Since the transfer function of the first system is known, comparisons between the step response of the various forms of the real continuous system, the analytical discrete system and identified model are shown.

Appendix C shows the effect of the different filters on the identified model.

5.1 SECOND ORDER LINEAR SYSTEM

The transfer function of the system is

$$H(s) = \frac{1}{s^2 + 1.414s + 1} \quad \text{so that} \quad H(s)\big|_{s=0} = 1.0 \qquad (5.1)$$

Using the *zero order hold* transform method (see MATLAB Control System Toolbox page CR-11) and sampling time, 0.3, this can be analytically transformed to:

$$H(z) = \frac{0.039z + 0.0338}{z^2 - 1.5815z + 0.6543} \quad \text{so that} \quad H(z)\big|_{z=1} = 1.0 \qquad (5.2)$$

The closed loop transfer function of (5.1) is:

$$H(s) = \frac{1}{s^2 + 1.414s + 2} \quad \text{so that } H(s)\big|_{s=0} = 0.5 \qquad (5.3)$$

The discrete form corresponding to this is:

$$H(z) = \frac{0.0387z + 0.0336}{z^2 - 1.5098z + 0.6543} \quad \text{so that } H(z)\big|_{z=1} = 0.5003 \qquad (5.4)$$

The best sampling time for this system was found as T=0.3 and the best parameters for the PRBS were found as MAG=0.5 and MAXT=8.0. After substitution of these parameters and then sampling the output of ESL program (STUDY.OUT) with T=0.3, the sampled data were transferred into MATLAB.

The identified model by ARX method was:

$$H(z) = \frac{0.0364z + 0.0298}{z^2 - 1.5883z + 0.6574} \quad \text{so that} \quad H(z)\big|_{z=1} = 0.958 \qquad (5.5)$$

The closed loop of the identified model was:

$$H(z)\big|_{cl} = \frac{H(z)\big|_{op}}{1+H(z)\big|_{op}} \quad \text{or}$$

$$H(z)\big|_{cl} = \frac{0.0364z + 0.0298}{z^2 - 1.5519z + 0.6872} \quad \text{so that } H(z)\big|_{cl\ z=1} = 0.489 \qquad (5.6)$$

From comparison of (5.2) with (5.5) it can be seen they are almost the same.

Fig 5.1 shows the step response of the continuous system (5.1) and its analytical discrete form (5.2) for open loop and closed loop. Fig 5.2 shows the analytical discrete system and identified model for open loop and closed loop.

The identified model was found by transferring one thousand input-output data points from the ESL program on the PC to the DEC-VAX. 600 data points were then selected for building a model. To evaluate how well the model fit the data, a simple test was to run a simulation whereby real input data was fed into the model, and to compare the simulated output with the actual, measured output. For this a portion of the data that was not used to build the model, for example points 900 to 1000 were selected (see Fig 5.3).

It can be seen that the model was quite capable of describing the system, even for data that were not used in calculating the fit. The best model gave a J value of 0.0039.

Fig 5.4 shows the effect of aliasing on the identified model. J value for this identified model is 0.0237 and if one compares it with that of the best model it shows why should use an anti-aliasing filter to remove the effect of the aliased noise. We should, however, be very careful to design the best filter to obtain the lowest J. Table 2 show J's for different cut-off frequencies of the Butterworth filter. For the first order filter with ω_n=5 we have the best response.

To quantify the effect of the filter on the identified model, the step response of the closed loop identified model without filter and with two kinds of filter was found. Fig 5.5 shows the step response of the identified closed loop system without filter and with a Butterworth filter. From the plots, three effects of the filter on the identification can be seen:

I. Increases the steady state error.

II. Increases the overshoot.

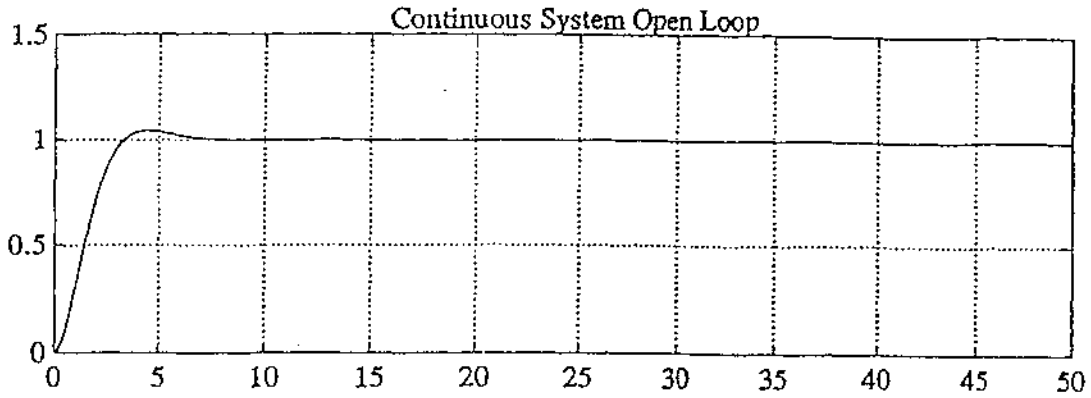
III. Increases the settling time.

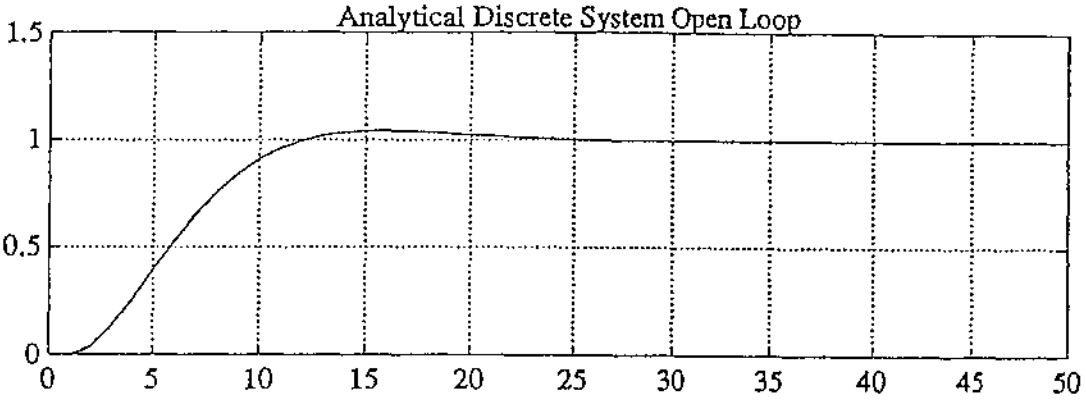
The same results was obtained for Chebychev filters (Fig 5.6). When the cut-off frequency of the filter (ω_n) is less than the natural frequency of the system (ω_s=1) the effect of the filter is much greater. As ω_n of the filter increases, the effect of the filter

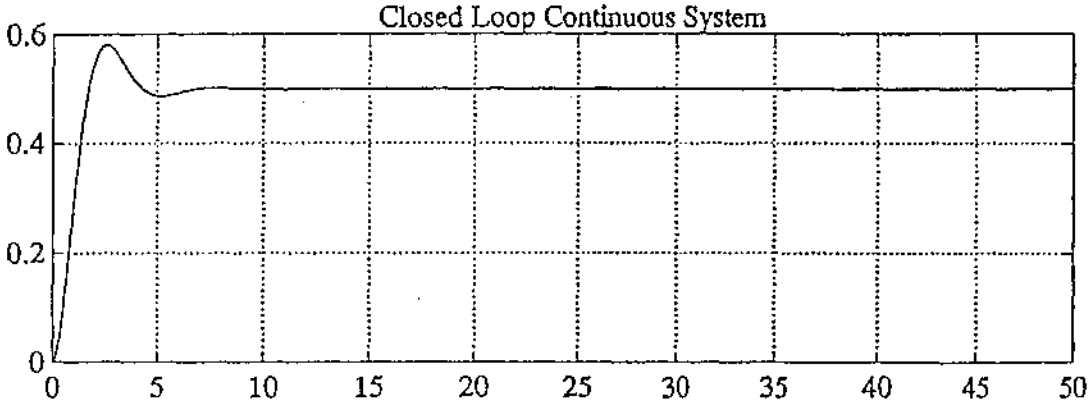
can be seen to decrease. When the degree of the filter goes up, the settling time after a step response increases, especially with a low cut-off frequency.

Fig 5.7 shows the cut-off frequency of the Butterworth filter versus error criterion (J) for different orders of filter. In Fig 5.7, when ω_n increases, J deceases but does not reach that of the best model (J=0.0039). Fig 5.8 shows the cut-off frequency of the Chebychev filter versus J for filters of different order. In Fig 5.8 it can be seen that when ω_n increases, J decreases, but never reaches that of the best model. When the degree of the filter increases the range of J values increases. Beyond 10 times the bandwidth of the system it does not really matter which degree of filter is used. Therefore, from the viewpoint of realization a first order filter is best. The same results are obtained with both identification methods, ARX and IV4.

Noise was now added into the system. The results for the Butterworth and Chebychev filters are shown in tables 2 and 4 respectively. Plots of ω_n versus J are shown in Fig 5.9 and 5.10. The minimum J value was obtained for a filter of order 1.







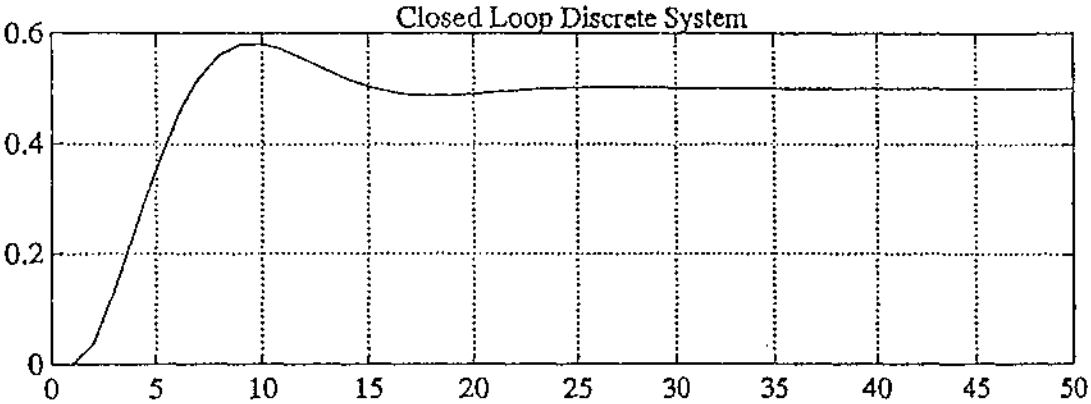


Fig 5.1 Step response of continuous system and its discrete in open loop and closed loop forms.

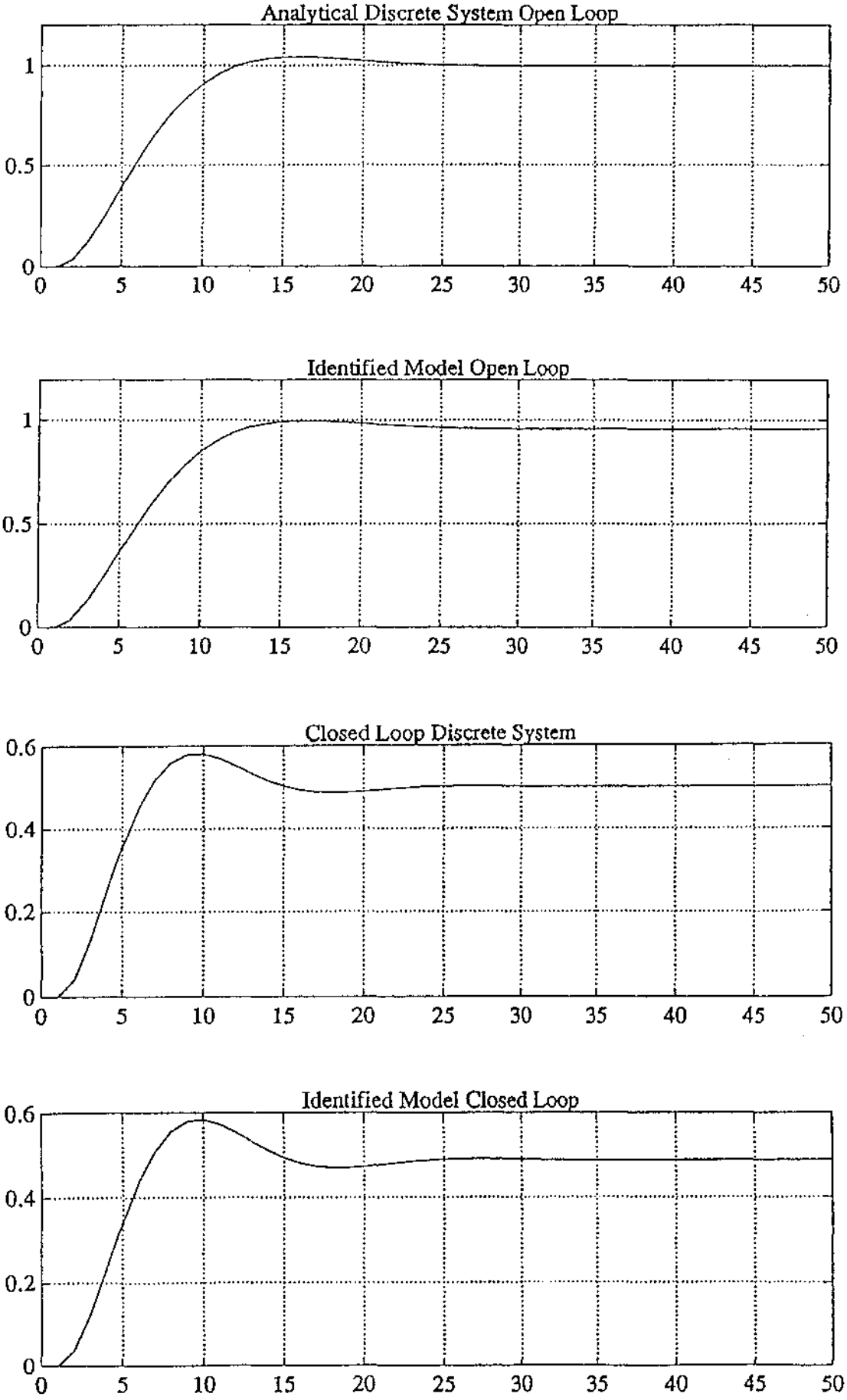


Fig 5.2 Step response of the analytical discrete system and identified model in two forms open loop and closed loop.

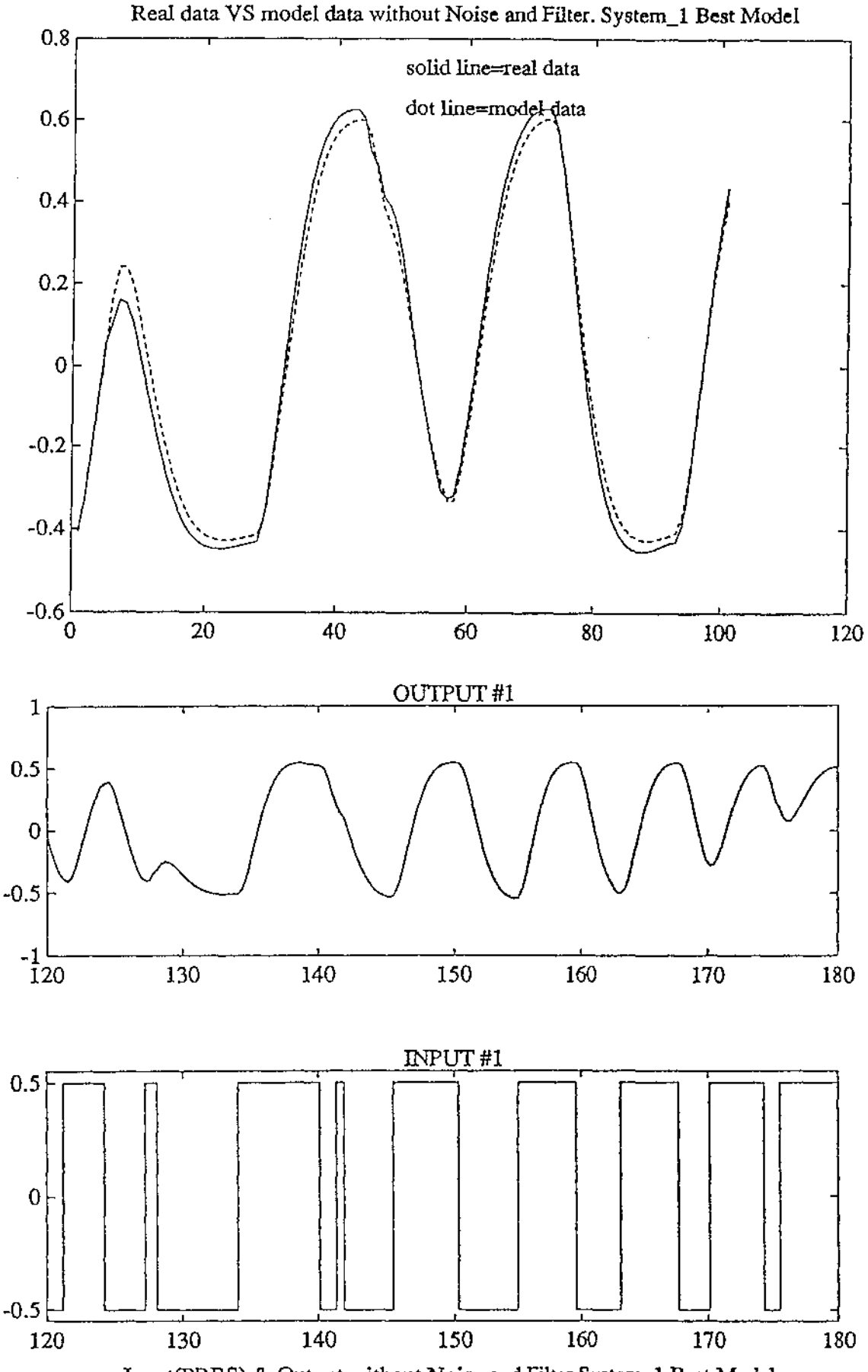


Fig 5.3 Input and output for system 1 and comparison between real data and model data.

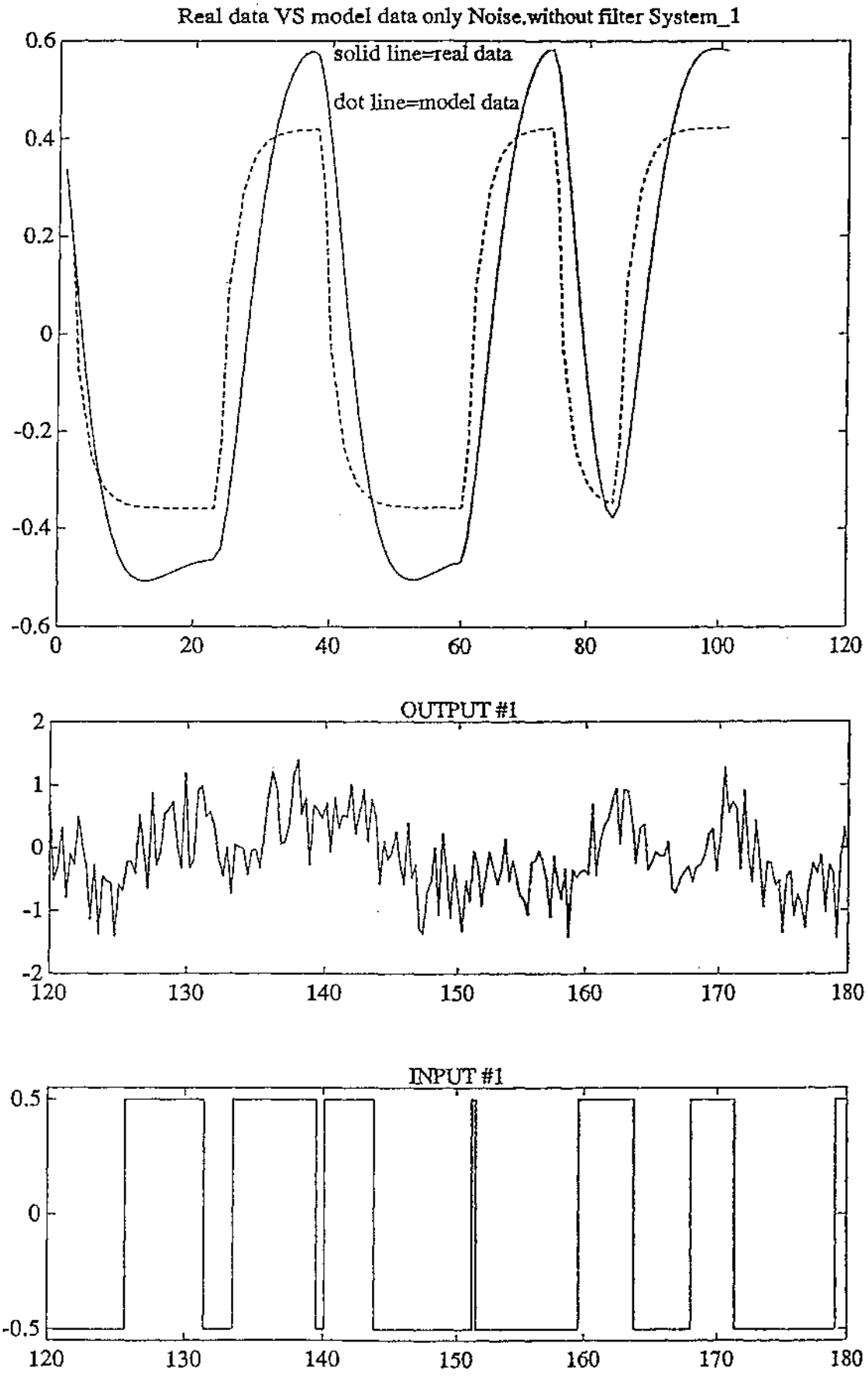


Fig 5.4 The effect of aliasing on the identified model for system 1.

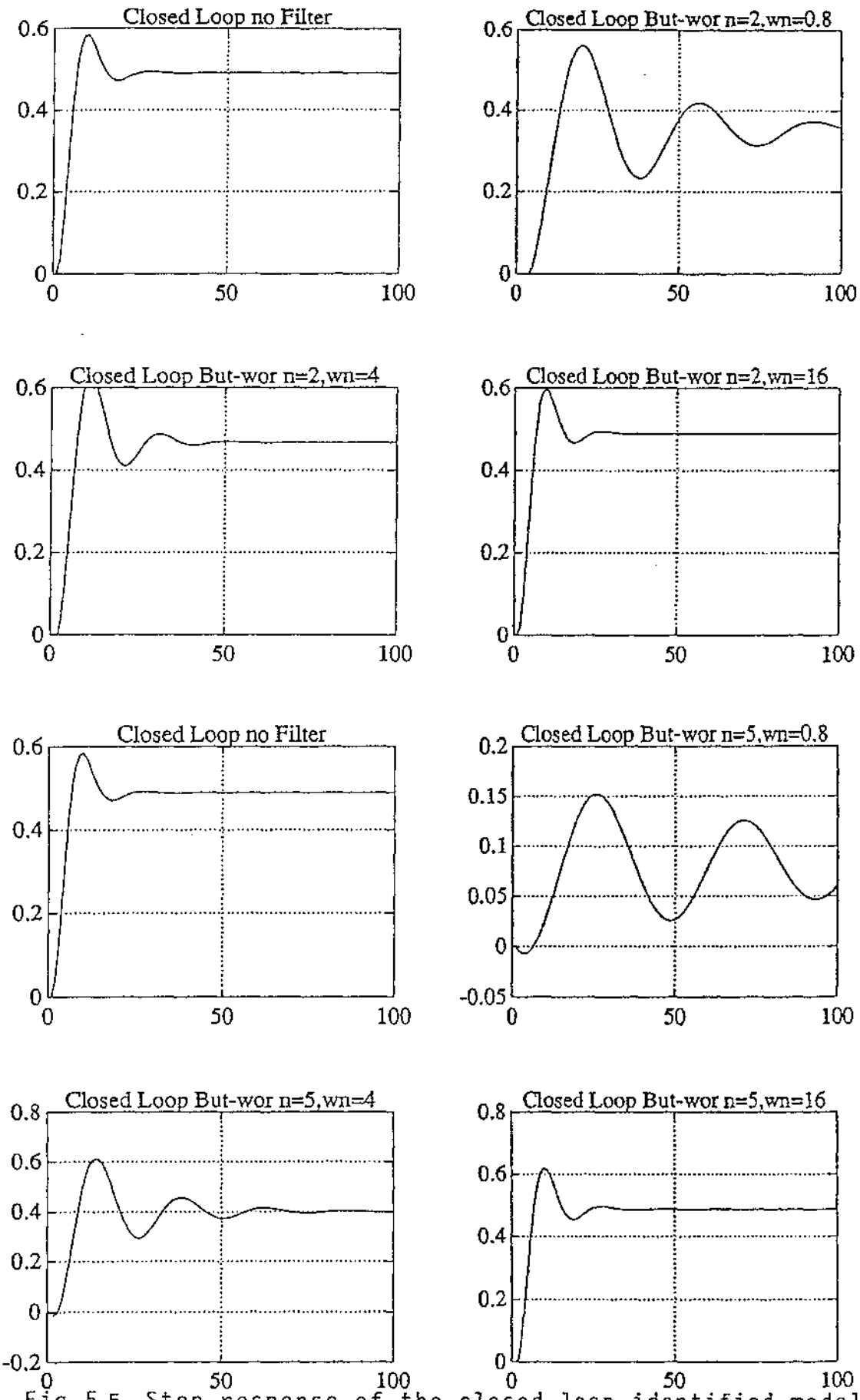


Fig 5.5 Step response of the closed loop identified model without filter and with different Butterworth filters.

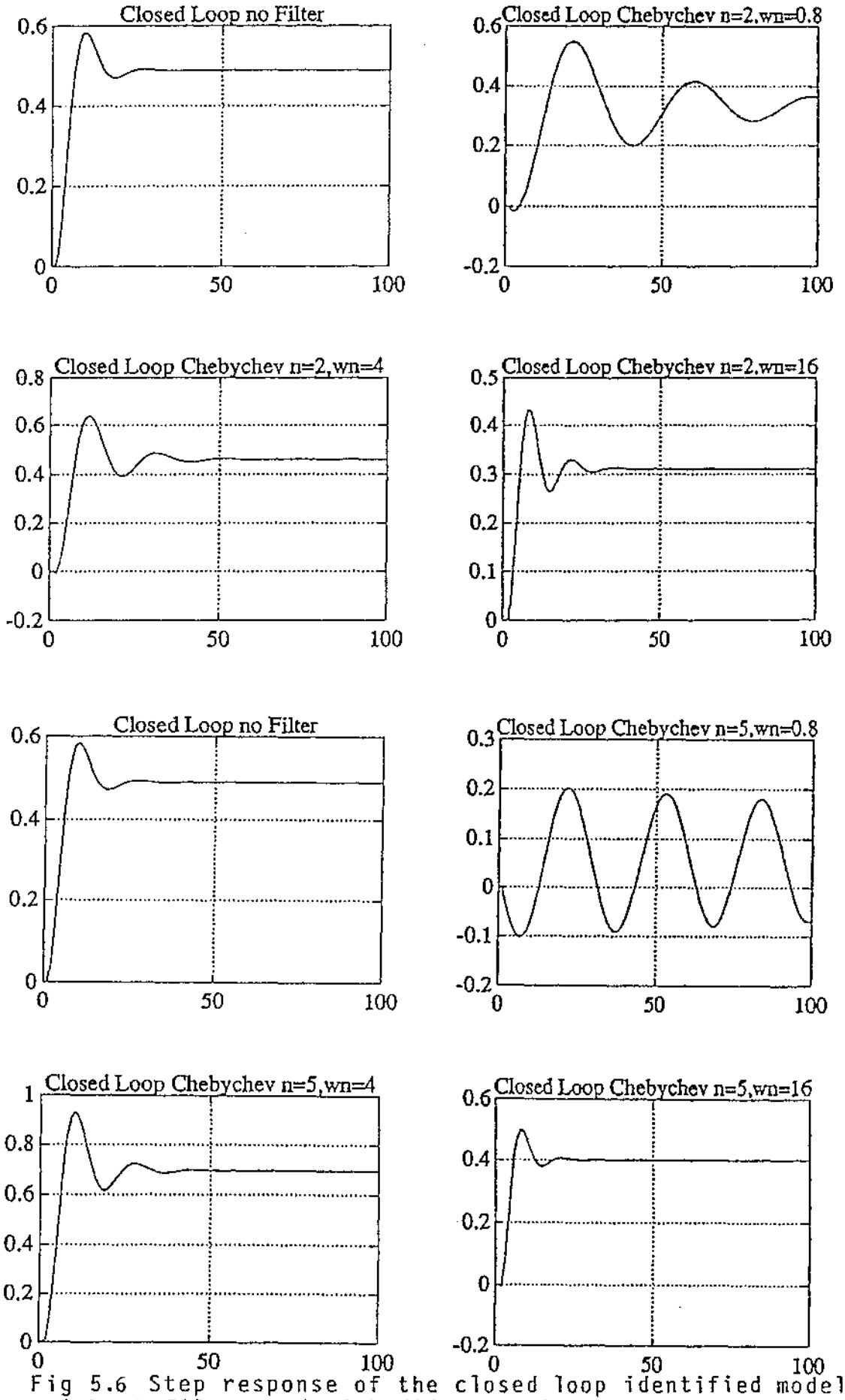


Fig 5.6 Step response of the closed loop identified model without filter and with different Chebychev filters.

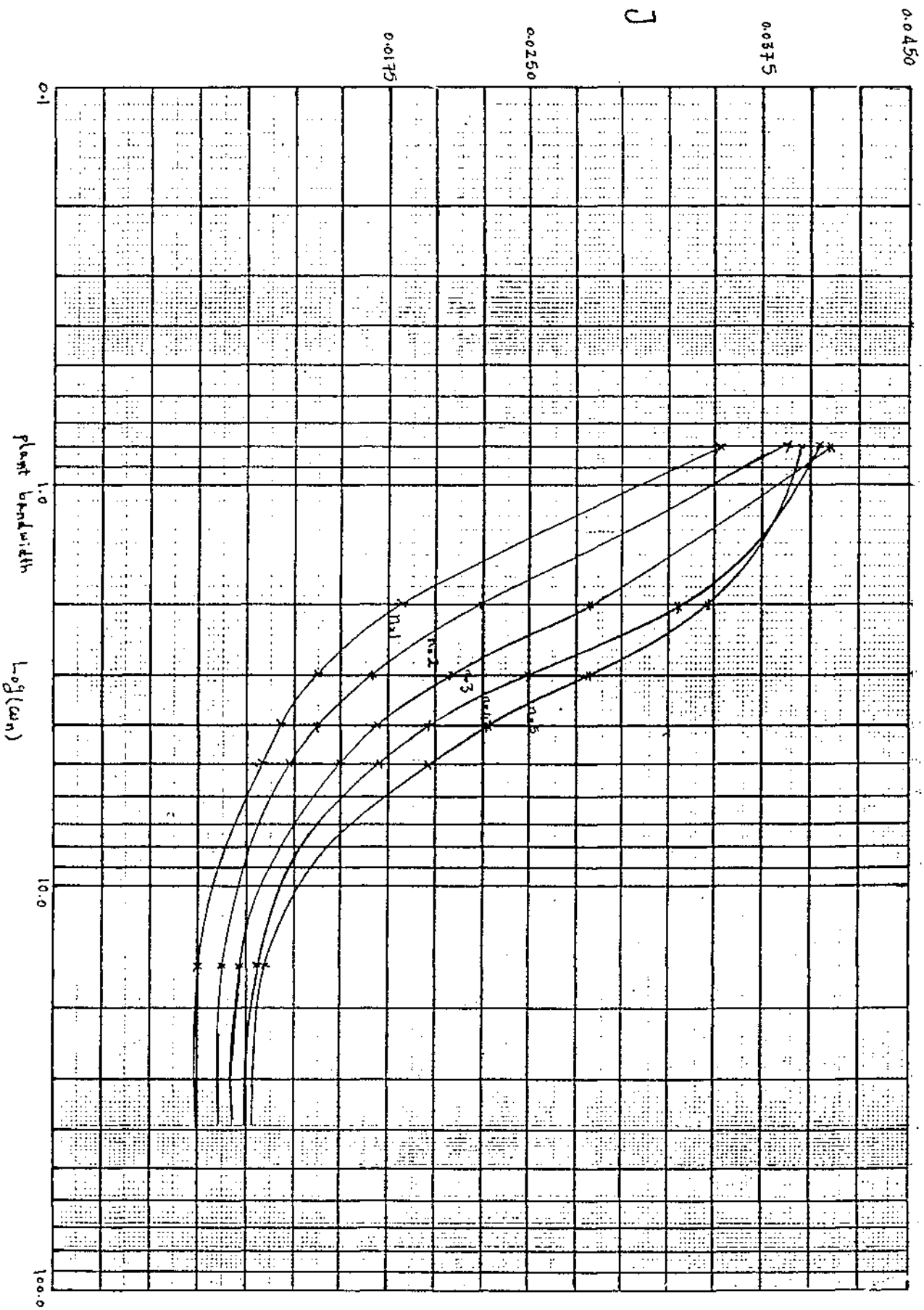


Fig 5.7 "System 1 - Only Butterworth Filter"

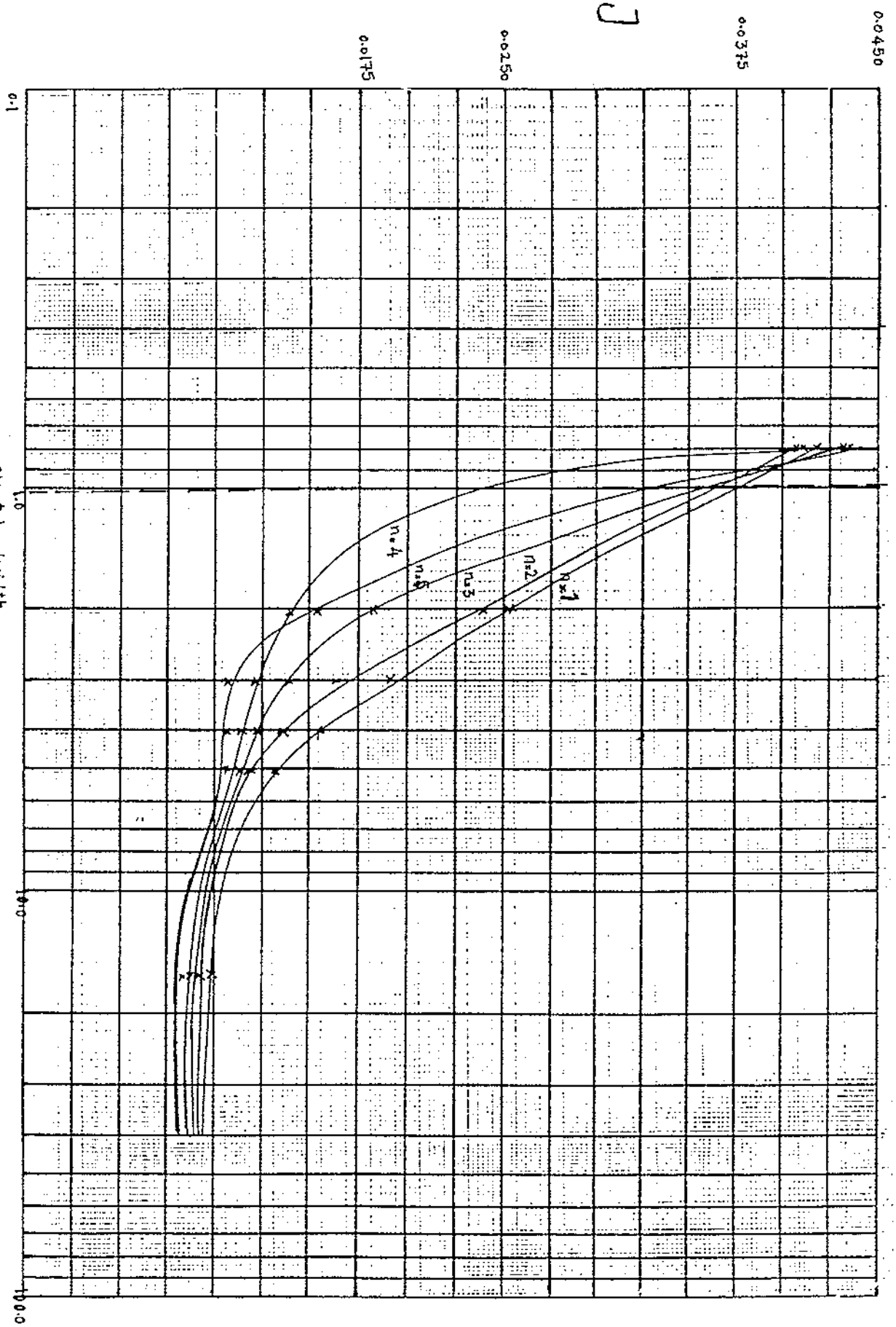


Fig 5.8 "System 1 - only Chebychev Filter"

Fig 5.9 "System 1. Noise and Butterworth Filter"

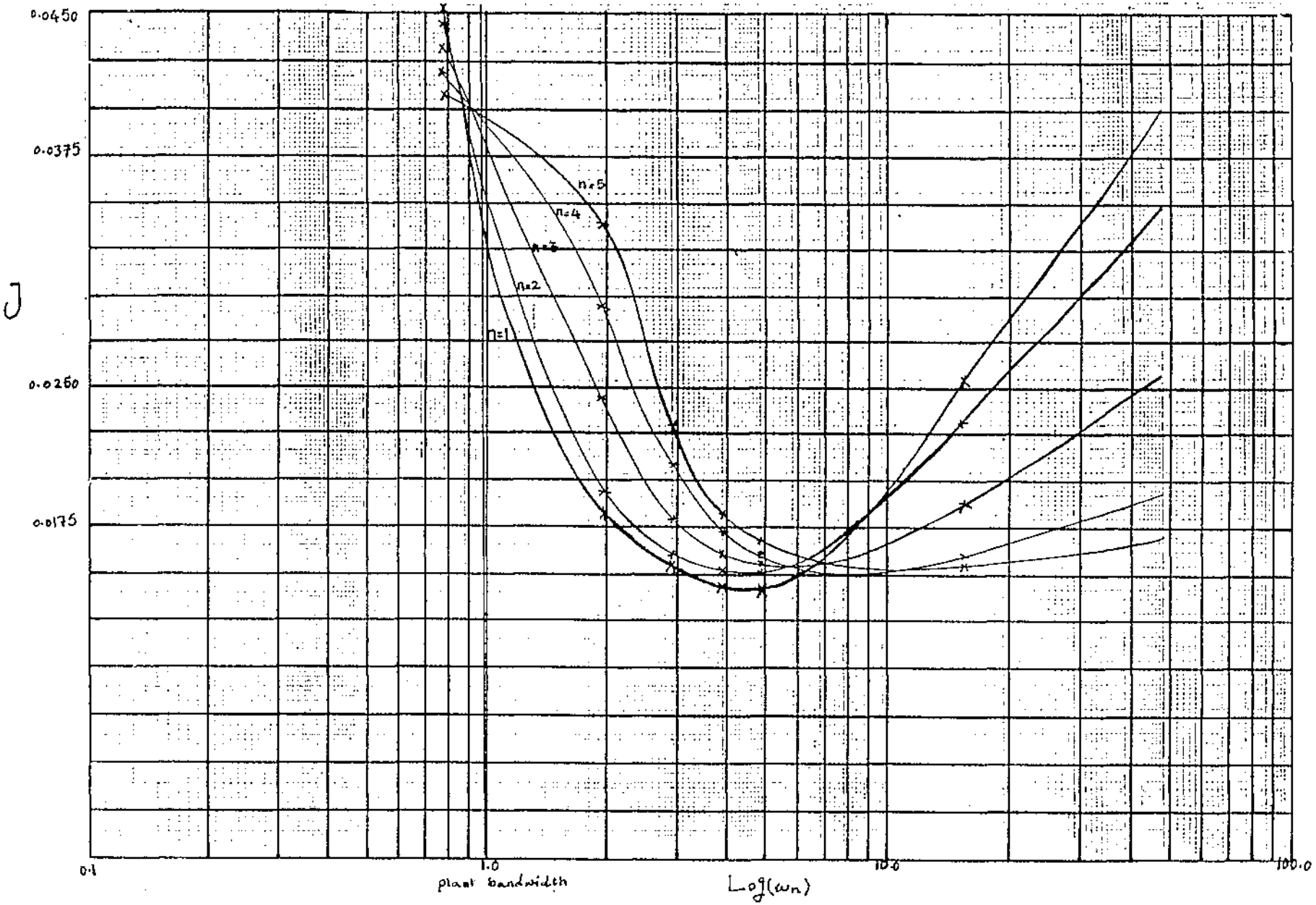


Fig 5.10 "System 1_Noise and Chebychev Filter"

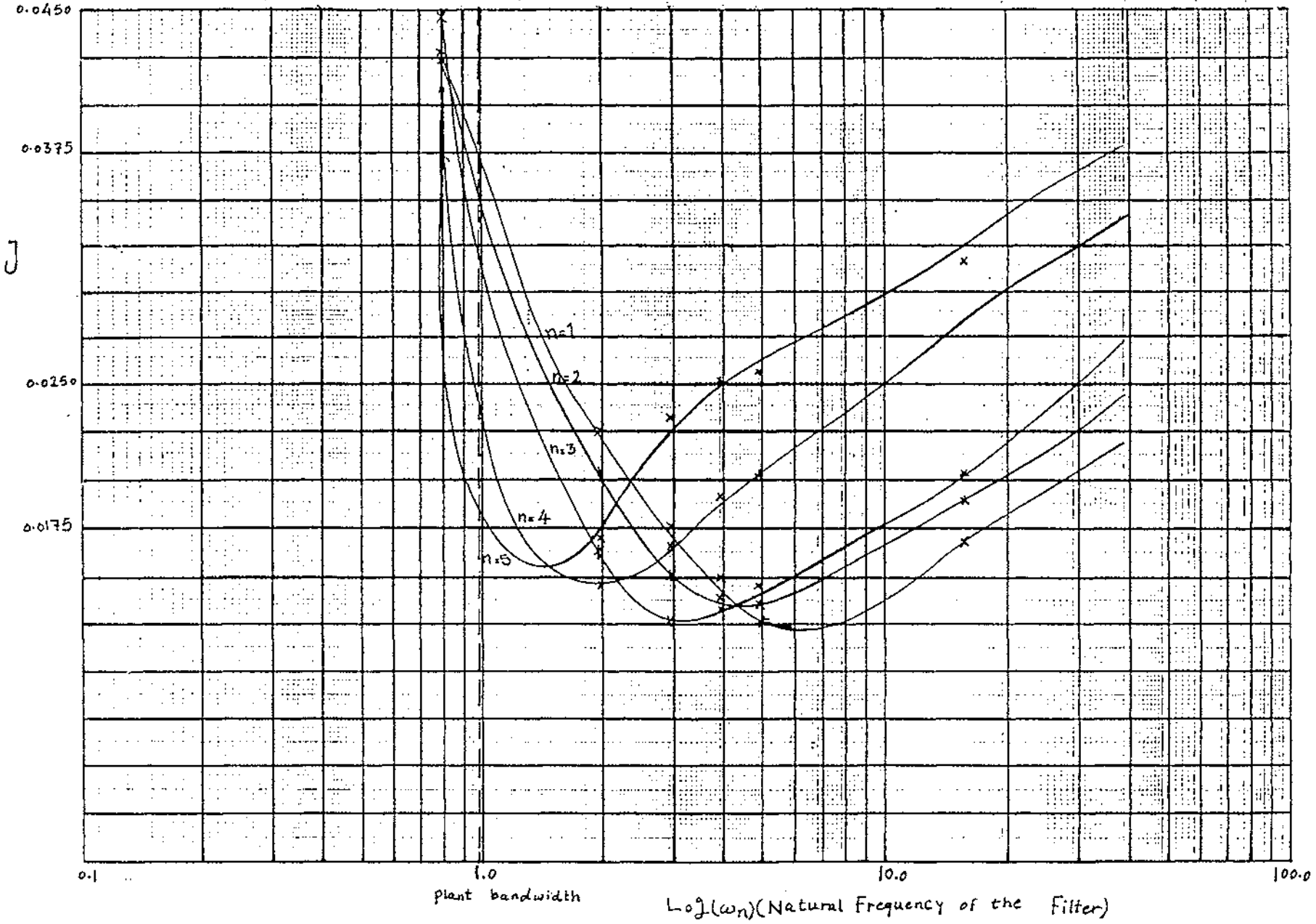


Table 1 System 1_only Buttterworth Filter

FILTER	J
S1B1ω0.8	0.0350
S1B1ω2	0.0175
S1B1ω3	0.0137
S1B1ω4	0.0115
S1B1ω5	0.0107
S1B1ω16	0.0075
S1B2ω0.8	0.0387
S1B2ω2	0.0219
S1B2ω3	0.0157
S1B2ω4	0.0131
S1B2ω5	0.0116
S1B2ω16	0.0087
S1B3ω0.8	0.0404
S1B3ω2	0.0280
S1B3ω3	0.0198
S1B3ω4	0.0160
S1B3ω5	0.0139
S1B3ω16	0.0091
S1B4ω0.8	0.0399
S1B4ω2	0.0327
S1B4ω3	0.0239
S1B4ω4	0.0192
S1B4ω5	0.0164
S1B4ω16	0.0096
S1B5ω0.8	0.0390
S1B5ω2	0.0356
S1B5ω3	0.0276
S1B5ω4	0.0222
S1B5ω5	0.0189
S1B5ω16	0.0102

Table 2 System 1_Noise and Butterworth Filter

FILTER	J
S1NB1ω0.8	0.0450
S1NB1ω2	0.0178
S1NB1ω3	0.0153
S1NB1ω4	0.0142
S1NB1ω5	0.0136
S1NB1ω16*	0.0250
S1NB2ω0.8	0.0442
S1NB2ω2	0.0190
S1NB2ω3	0.0155
S1NB2ω4	0.0146
S1NB2ω5	0.0142
S1NB2ω16	0.0227
S1NB3ω0.8	0.0429
S1NB3ω2	0.0241
S1NB3ω3	0.0165
S1NB3ω4	0.0149
S1NB3ω5	0.0146
S1NB3ω16	0.0181
S1NB4ω0.8	0.0416
S1NB4ω2	0.0291
S1NB4ω3	0.0184
S1NB4ω4	0.0156
S1NB4ω5	0.0149
S1NB4ω16	0.0153
S1NB5ω0.8	0.0402
S1NB5ω2	0.0334
S1NB5ω3	0.0204
S1NB5ω4	0.0169
S1NB5ω5	0.0158
S1NB5ω16	0.0148

* S1NB1ω16 means system 1 with noise and Butterworth filter n=1, ω_n=16.

Table 3 System 1_ only Chebychev Filter

FILTER	J
S1C1ω0.8*	0.0412
S1C1ω2	0.0250
S1C1ω3	0.0187
S1C1ω4	0.0150
S1C1ω5	0.0127
S1C1ω16	0.0092
S1C2ω0.8	0.0402
S1C2ω2	0.0235
S1C2ω3	0.0162
S1C2ω4	0.0132
S1C2ω5	0.0117
S1C2ω16	0.0087
S1C3ω0.8	0.0429
S1C3ω2	0.0179
S1C3ω3	0.0137
S1C3ω4	0.0119
S1C3ω5	0.0109
S1C3ω16	0.0086
S1C4ω0.8	0.0406
S1C4ω2	0.0135
S1C4ω3	0.0119
S1C4ω4	0.0112
S1C4ω5	0.0105
S1C4ω16	0.0086
S1C5ω0.8	0.0430
S1C5ω2	0.0150
S1C5ω3	0.0104
S1C5ω4	0.0104
S1C5ω5	0.0102

Table 4 System1 _ Noise and Chebychev Filter

FILTER	J
S1NC1ω0.8	0.0425
S1NC1ω2	0.0225
S1NC1ω3	0.0175
S1NC1ω4	0.0150
S1NC1ω5	0.0125
S1NC1ω16	0.0170
S1NC2ω0.8	0.0422
S1NC2ω2	0.0204
S1NC2ω3	0.0150
S1NC2ω4	0.0139
S1NC2ω5	0.0137
S1NC2ω16	0.0189
S1NC3ω0.8	0.0449
S1NC3ω2	0.0163
S1NC3ω3	0.0128
S1NC3ω4	0.0137
S1NC3ω5	0.0146
S1NC3ω16	0.0203
S1NC4ω0.8	0.0407
S1NC4ω2	0.0146
S1NC4ω3	0.0167
S1NC4ω4	0.0191
S1NC4ω5	0.0201
S1NC4ω16	0.0316
S1NC5ω0.8	0.0392
S1NC5ω2	0.0169
S1NC5ω3	0.0231
S1NC5ω4	0.0251
S1NC5ω5	0.0255

* S1C1ω0.8 means system 1 with Chebychev filter n=1, ω_n=0.8.

5.1.1 Discussion (System 1)

As the filter cut-off frequency increases, identification of the system becomes easier because the filter passes more of the system bandwidth with less distortion. This is continues until ω_n (cut-off frequency of the filter) reaches 10 times the bandwidth of the system at which point all of the information concerning the system passes through the filter and therefore leaves J constant (Fig 5.11).

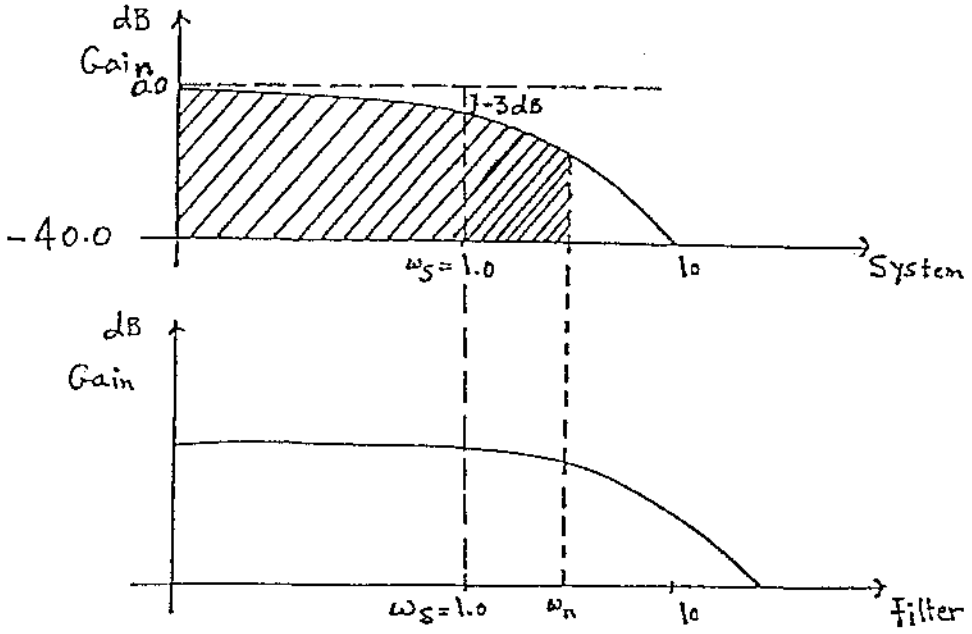


Fig 5.11 Relation between the bandwidth of the system and cut-off frequency of the filter

A higher order filter is seen to increase J because the increasing phase distortions in the pass band of the filter effect the identification (Fig 5.12).

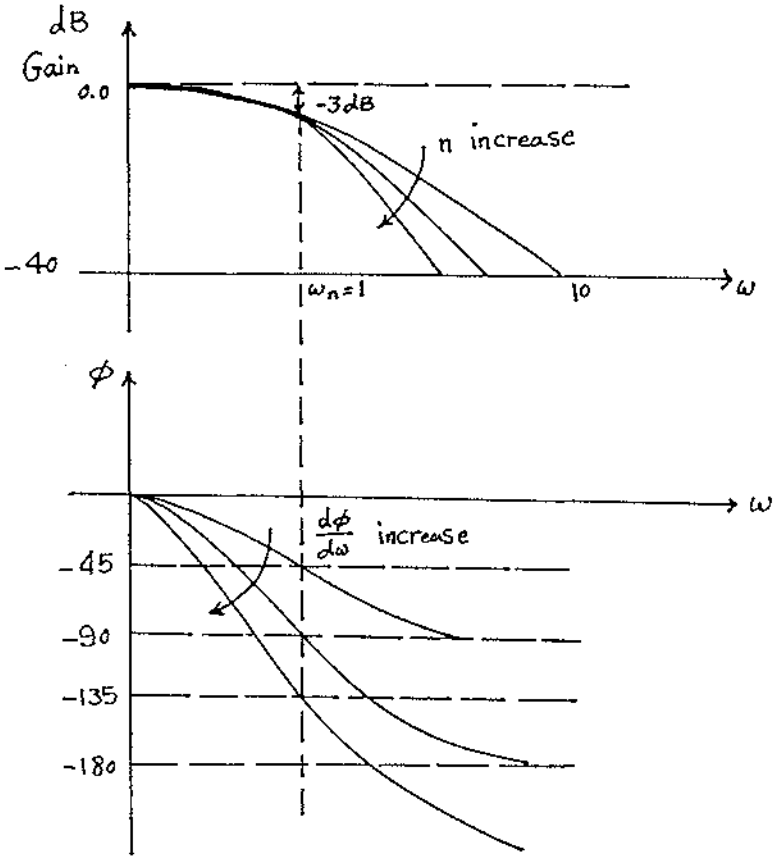


Fig 5.12 The effect of the phase distortions on J

With noise added to the system, J will decrease initially with increasing ω_n and then increase. This is because, initially, the filter passes more information about the system, which the identification algorithm can use to find the best model. Beyond 5 times the plant bandwidth, however the noise also begins to pass through the filter and detrimentally affecting the identification algorithm (Fig 5.13).

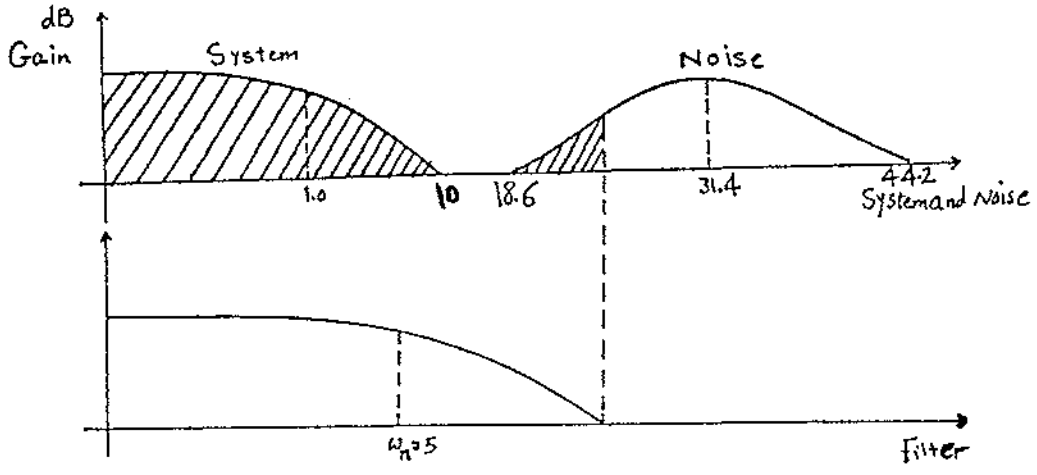


Fig 5.13 Relation between noise added system and filter

5.2 NONLINEAR REACTION SYSTEM

Before experimentation on filters a suitable sampling time and suitable PRBS parameters were needed. Values that produced a sufficient bandwidth of excitation were T=0.1 for the sampling time with MAG=0.5 and MAXT=3 for the PRBS parameters.

The best identified model for this system was:

$$H(z) = \frac{0.0393z - 0.0061}{z^2 - 1.1230z + 0.2038}$$

for which J= 0.0019.

Fig 5.14 shows the input (PRBS), the output and the real / best model comparisons.

Fig 5.15 shows the effect of aliasing on the identified model. The response of the model identified with added noise but no filter shows clearly that high order modes (probably corresponding to the noise) have been identified whereas the low order modes are not identified at all. The value of J for this identified model is 0.0086 that compared with the best model (J=0.0019) it is obvious that we should use an anti-aliasing filter. However, we should be very careful to design a suitable filter. For example in this case only first order Butterworth filter with cut-off frequency 4 rad s^{-1} gives the lowest J (see table 6).

Figs 5.16 and 5.17 are plots of J value against ω_n for Butterworth filters and Chebychev filters respectively. Figs 5.18 and 5.19 show the results when noise was added to the system.

5.2.1 Discussion (System 2)

The response of the system to a step input was obtained by ESL and then the Power Spectral Density of the system was plotted. From this graph the natural frequency of the system was determinated at the half power point ($\omega_s = 4$).

From Figs 5.16 and 5.17 it can be seen that when the filter cut-off frequency increases J decreases because more of the frequency of the system passes through the filter. Beyond 10 times the plant bandwidth ($\omega_n \geq 10*\omega_s$) it does not really matter which filter is chosen. A first order filter is therefore preferred for realisation since all filters will pass the total system bandwidth.

Figs 5.18 and 5.19 show the effect of the filter on identification when noise was added to the system. The value of J increases as ω_n passes beyond 3 times the plant bandwidth since the noise can then pass through to the identification algorithm.

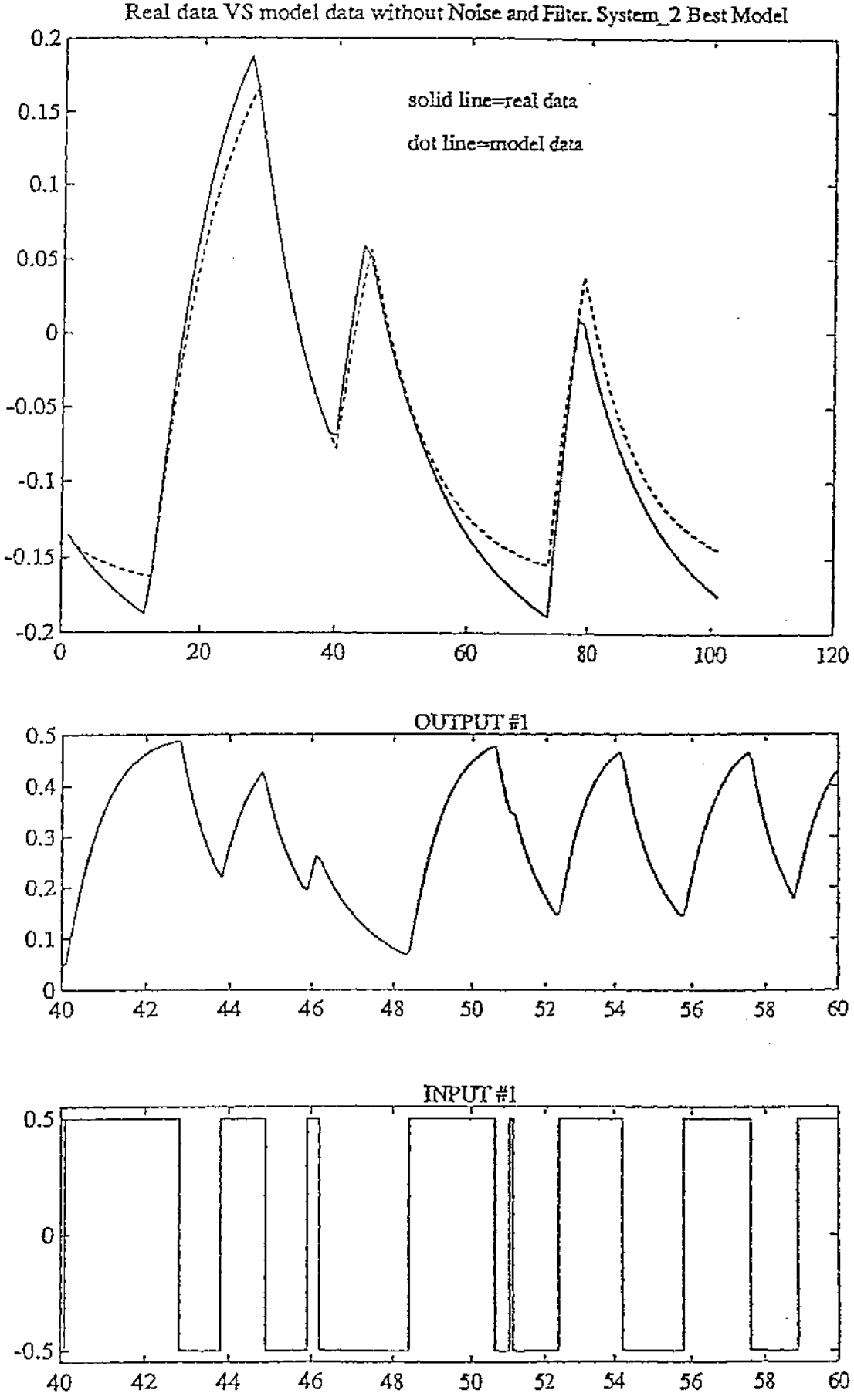


Fig 5.14 Input, Output and Comparison between Real data and Model data for System 2.

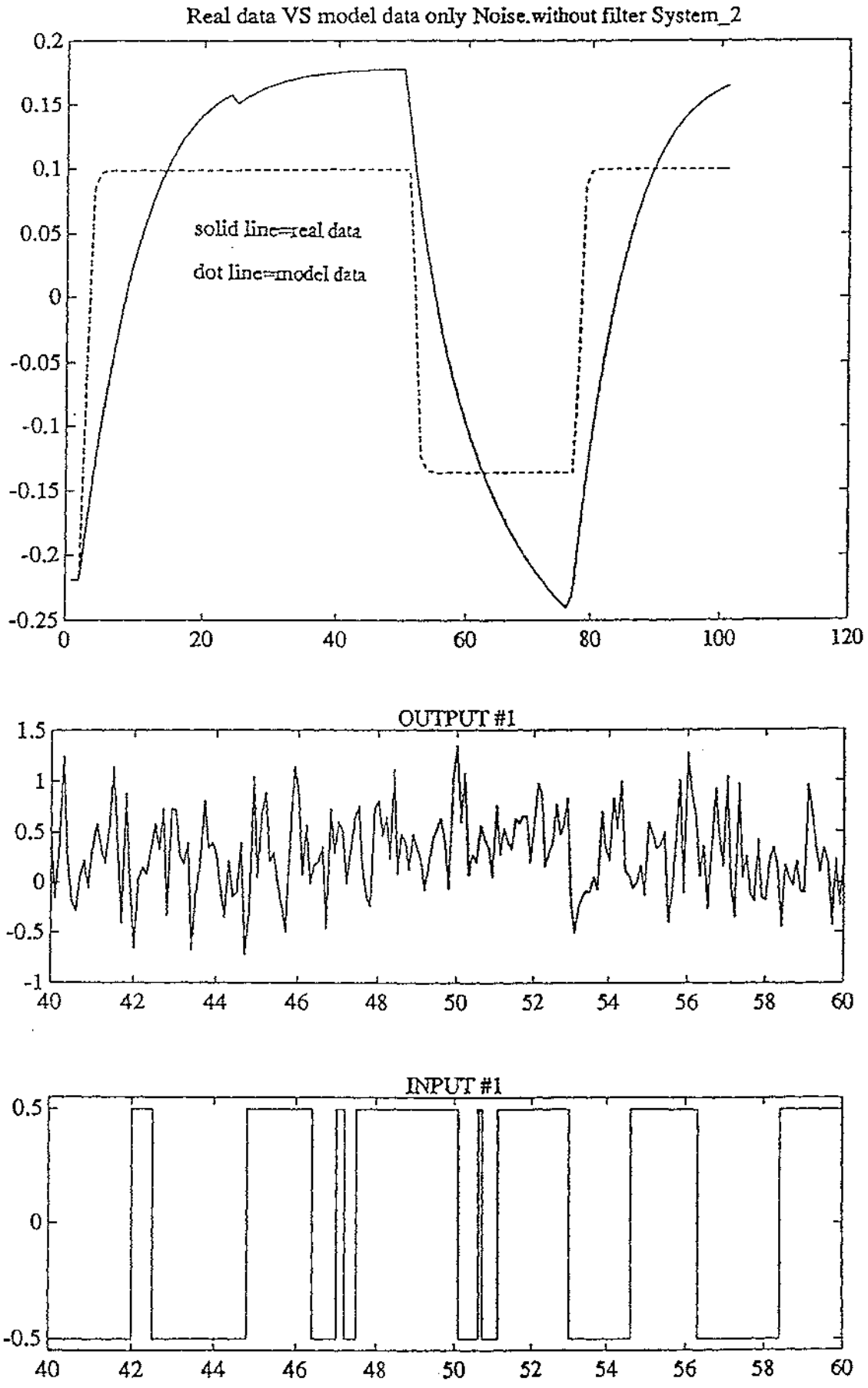


Fig 5.15 The effect of aliasing on the identified model for system 2.

Fig 5.16 "System 2 - only Butterworth Filter"

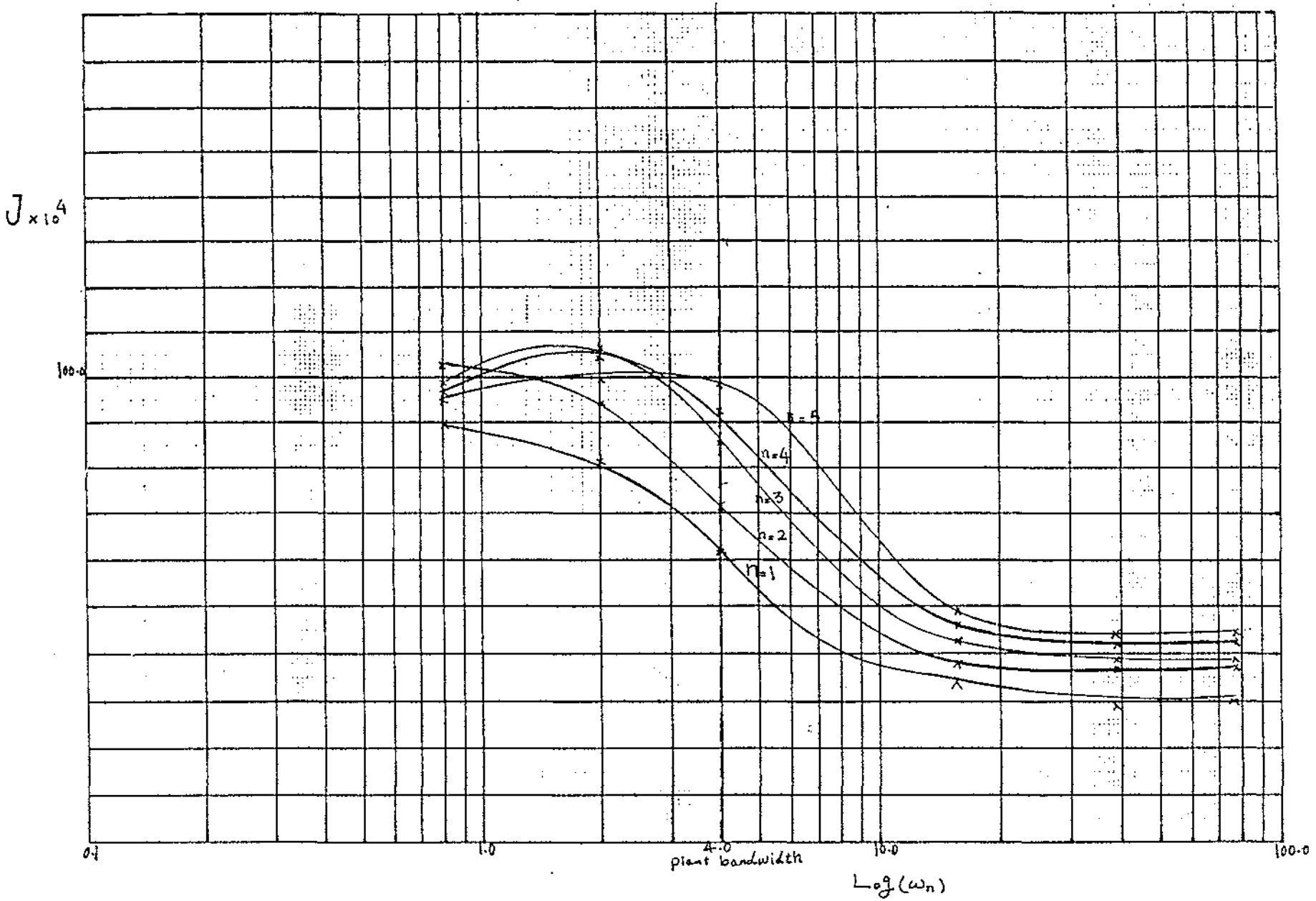
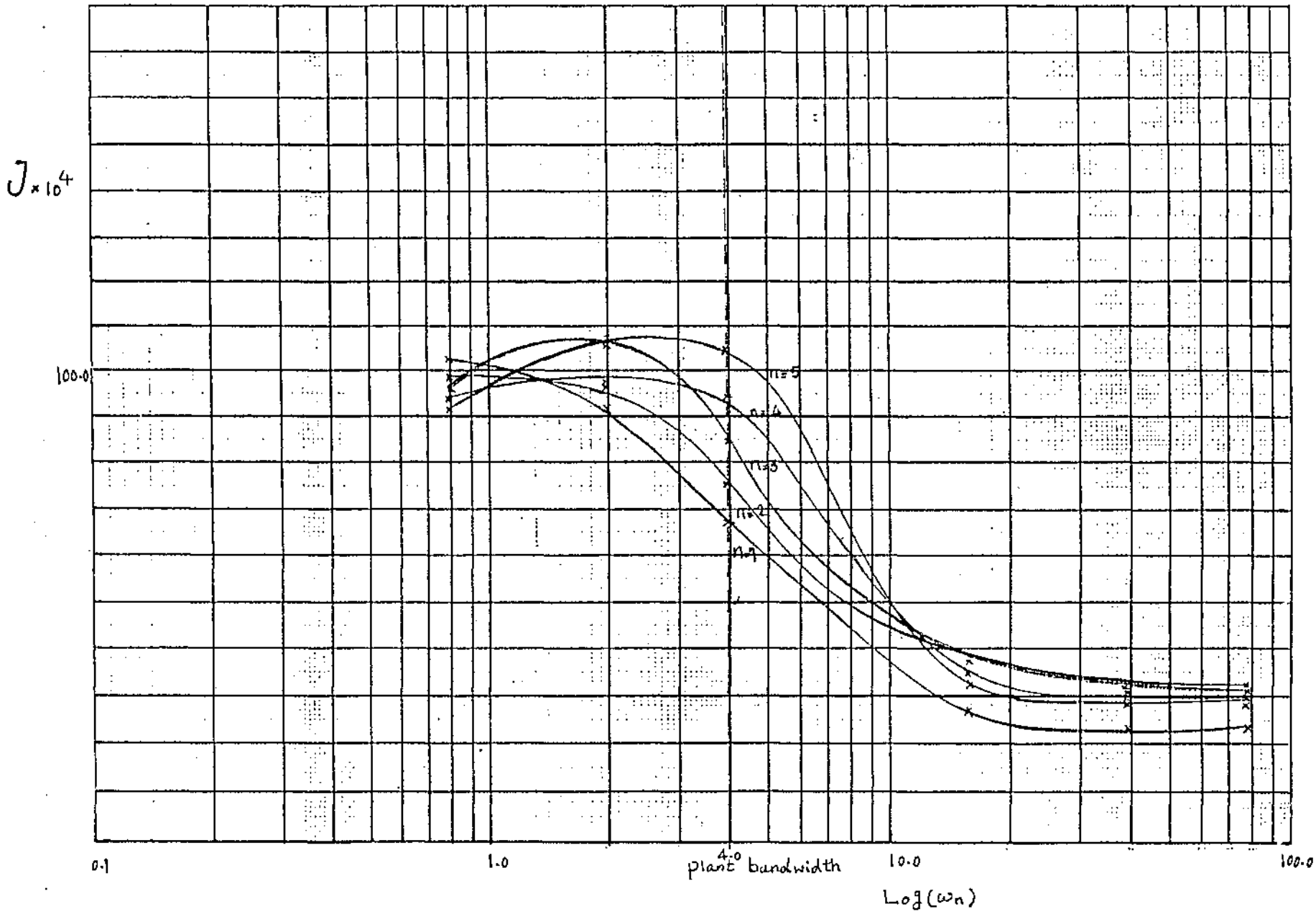


fig 5.17 System 2 only Chebychev Filter



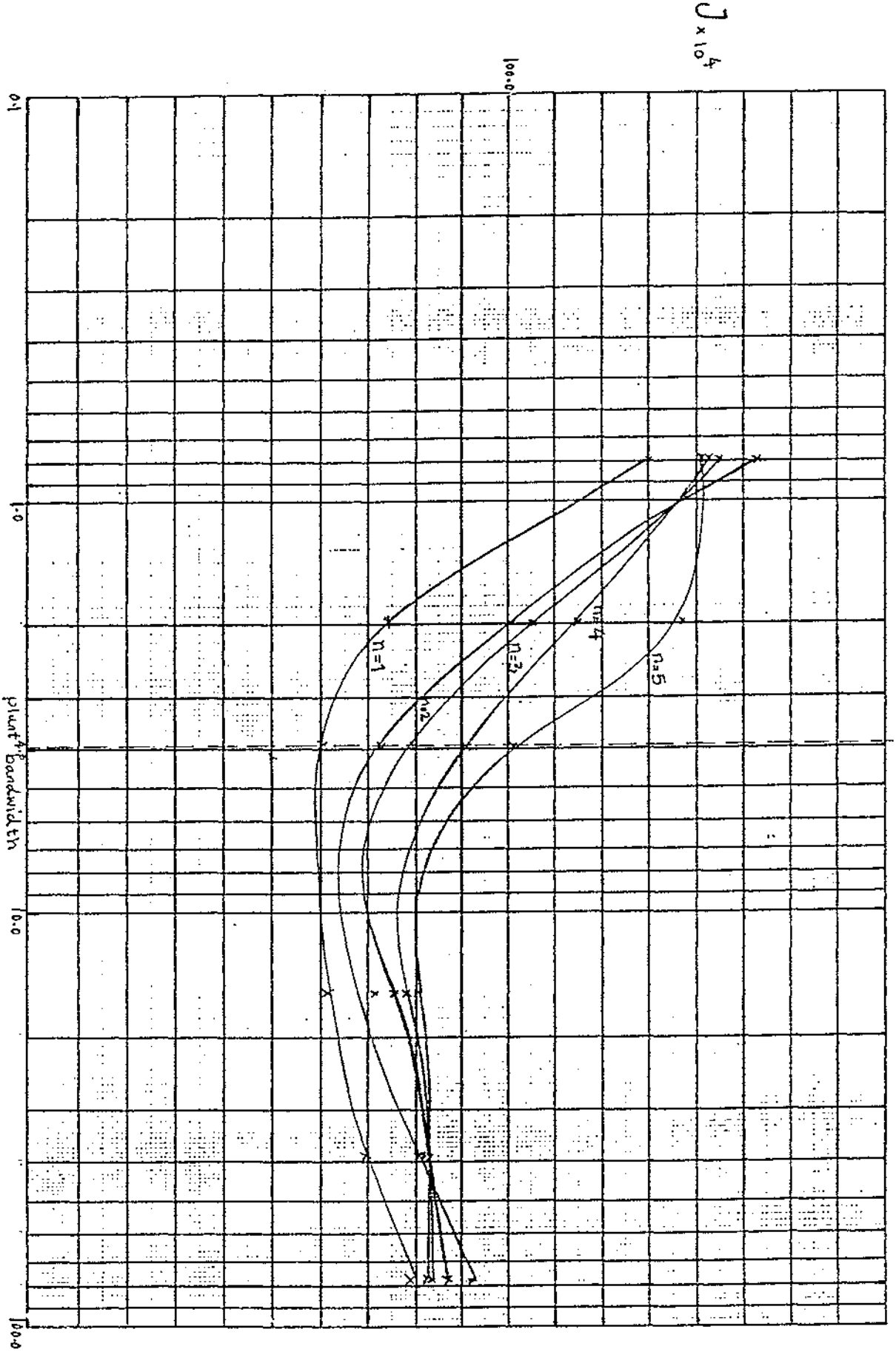


fig 5.18 " System 2_Noise and Butterworth filter "

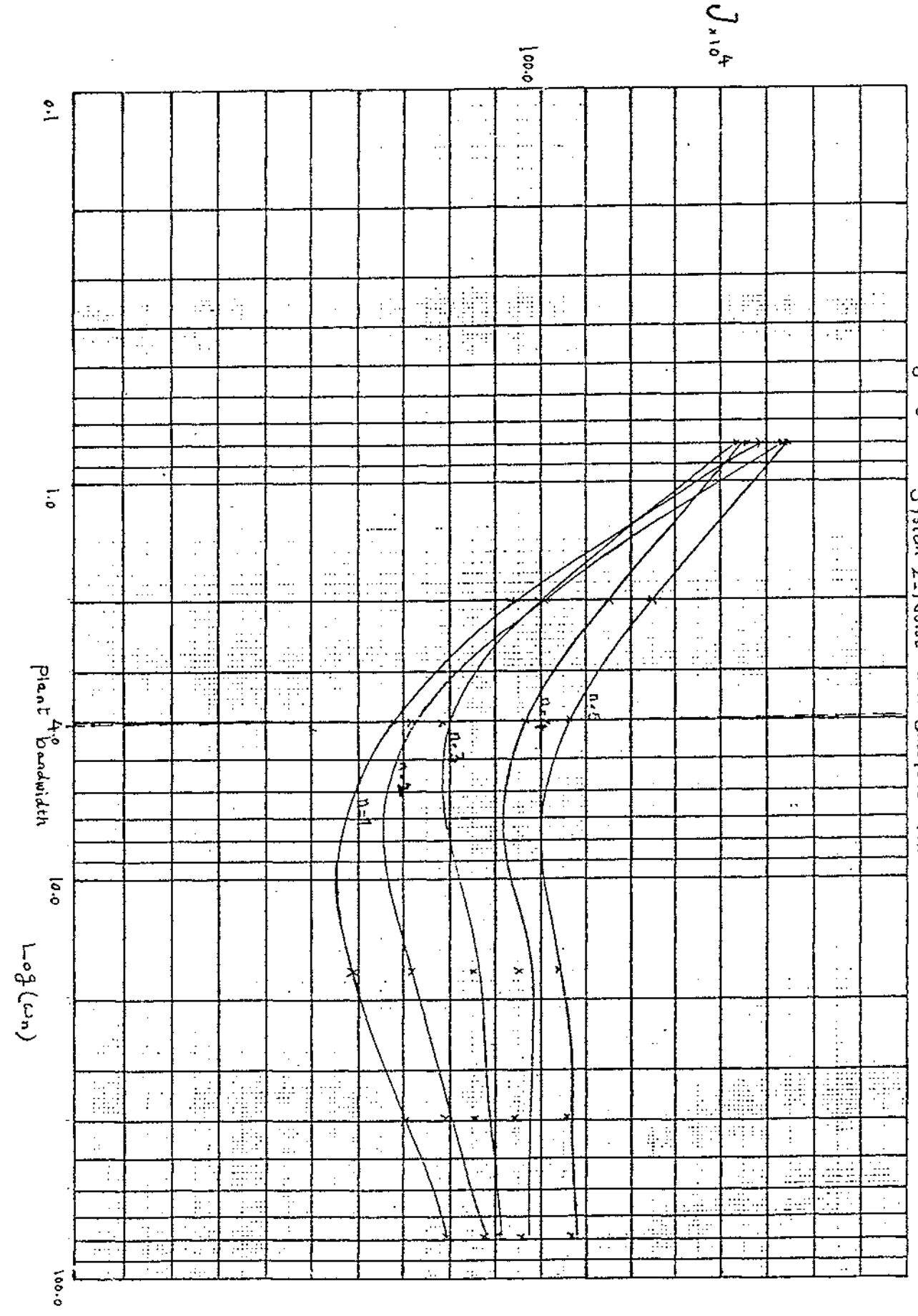


Fig 5.19 "System 2-Noise and Chebycheu Filter"

Table 5 System 2_ only Butterworth Filter

FILTER	J
S2B1ω0.8	0.0088
S2B1ω2	0.0080
S2B1ω4	0.0060
S2B1ω16	0.0034
S2B1ω40	0.0029
S2B1ω80	0.0029
S2B2ω0.8	0.0101
S2B2ω2	0.0093
S2B2ω4	0.0070
S2B2ω16	0.0037
S2B2ω40	0.0036
S2B2ω80	0.0036
S2B3ω0.8	0.0097
S2B3ω2	0.0105
S2B3ω4	0.0084
S2B3ω16	0.0042
S2B3ω40	0.0038
S2B3ω80	0.0038
S2B4ω0.8	0.0095
S2B4ω2	0.0104
S2B4ω4	0.0091
S2B4ω16	0.0045
S2B4ω40	0.0041
S2B4ω80	0.0041
S2B5ω0.8	0.0094
S2B5ω2	0.0098
S2B5ω4	0.0097
S2B5ω16	0.0048
S2B5ω40	0.0043

Table 6 System 2_ Noise and Butterworth Filter

FILTER	J
S2NB1ω0.8*	0.0129
S2NB1ω2	0.0074
S2NB1ω4	0.0060
S2NB1ω16	0.0062
S2NB1ω40	0.0069
S2NB1ω80	0.0078
S2NB2ω0.8	0.0152
S2NB2ω2	0.0100
S2NB2ω4	0.0072
S2NB2ω16	0.0071
S2NB2ω40	0.0080
S2NB2ω80	0.0091
S2NB3ω0.8	0.0144
S2NB3ω2	0.0105
S2NB3ω4	0.0079
S2NB3ω16	0.0075
S2NB3ω40	0.0080
S2NB3ω80	0.0086
S2NB4ω0.8	0.0142
S2NB4ω2	0.0114
S2NB4ω4	0.0090
S2NB4ω16	0.0077
S2NB4ω40	0.0080
S2NB4ω80	0.0083
S2NB5ω0.8	0.0140
S2NB5ω2	0.0136
S2NB5ω4	0.0100
S2NB5ω16	0.0079
S2NB5ω40	0.0081

* S2NB1ω0.8 means system 2 with noise and Butterworth filter n=1, ω_n=0.8.

Table 7 System 2_ only Chebychev Filter

FILTER	J
S2C1ω08	0.0104
S2C1ω2	0.0093
S2C1ω4	0.0069
S2C1ω16	0.0029
S2C1ω40	0.0025
S2C1ω80	0.0025
S2C2ω08	0.0100
S2C2ω2	0.0099
S2C2ω4	0.0077
S2C2ω16	0.0040
S2C2ω40	0.0034
S2C2ω80	0.0034
S2C3ω08	0.0097
S2C3ω2	0.0108
S2C3ω4	0.0086
S2C3ω16	0.0039
S2C3ω40	0.0033
S2C3ω80	0.0034
S2C4ω08	0.0095
S2C4ω2	0.0099
S2C4ω4	0.0096
S2C4ω16	0.0037
S2C4ω40	0.0036
S2C4ω80	0.0036
S2C5ω08	0.0093
S2C5ω2	0.0108
S2C5ω4	0.0106
S2C5ω16	0.0036
S2C5ω40	0.0036
S2C5ω80	0.0036

Table 8 System 2 _ Noise and Chebychev Filter

FILTER	J
S2NC1ω08*	0.0150
S2NC1ω2	0.0095
S2NC1ω4	0.0070
S2NC1ω16	0.0060
S2NC1ω40	0.0071
S2NC1ω80	0.0080
S2NC2ω08	0.0155
S2NC2ω2	0.0101
S2NC2ω4	0.0073
S2NC2ω16	0.0073
S2NC2ω40	0.0080
S2NC2ω80	0.0088
S2NC3ω08	0.0145
S2NC3ω2	0.0102
S2NC3ω4	0.0080
S2NC3ω16	0.0086
S2NC3ω40	0.0086
S2NC3ω80	0.0091
S2NC4ω08	0.0147
S2NC4ω2	0.0116
S2NC4ω4	0.0098
S2NC4ω16	0.0096
S2NC4ω40	0.0095
S2NC4ω80	0.0096
S2NC5ω08	0.0155
S2NC5ω2	0.0126
S2NC5ω4	0.0108
S2NC5ω16	0.0105
S2NC5ω40	0.0106
S2NC5ω80	0.0109

* S2NC1ω0.8 means system 2 with noise and Chebychev filter n=1, ω_n=0.8.

5.3 Switched-Mode Power Regulator

PRBS parameters of MAG=0.5 and MAXT=0.00009 were found to give a suitable excitation spectrum for this system. The best sampling time was found to be T=12.5E-6.

The best identified model was:

$$H(z) = \frac{0.1082z + 0.0579}{z^2 - 0.6603z - 0.0681}$$

with J=0.0071.

Various orders of the identified models were examined, but the second order model above gave the lowest J.

Fig 5.20 shows the system input, system output and real / best model comparisons.

The experiment with additive noise and no filter gave J=0.0110 (Fig 5.21). Comparing this J with that of the best identified model, we can see the effect of the aliasing on the identified model. If we are careful enough to design the best filter (second order ω_n=50000, J=0.0077) we can reduce the effect of aliasing on the identified model.

Because of the discontinuity (internal noise) in this system the results are different from the two previous systems. Figs 5.22 and 5.23 show the results of this system with different orders of the Butterworth and Chebychev filters respectively. Fig 5.24 and 5.25 show the results when noise was added.

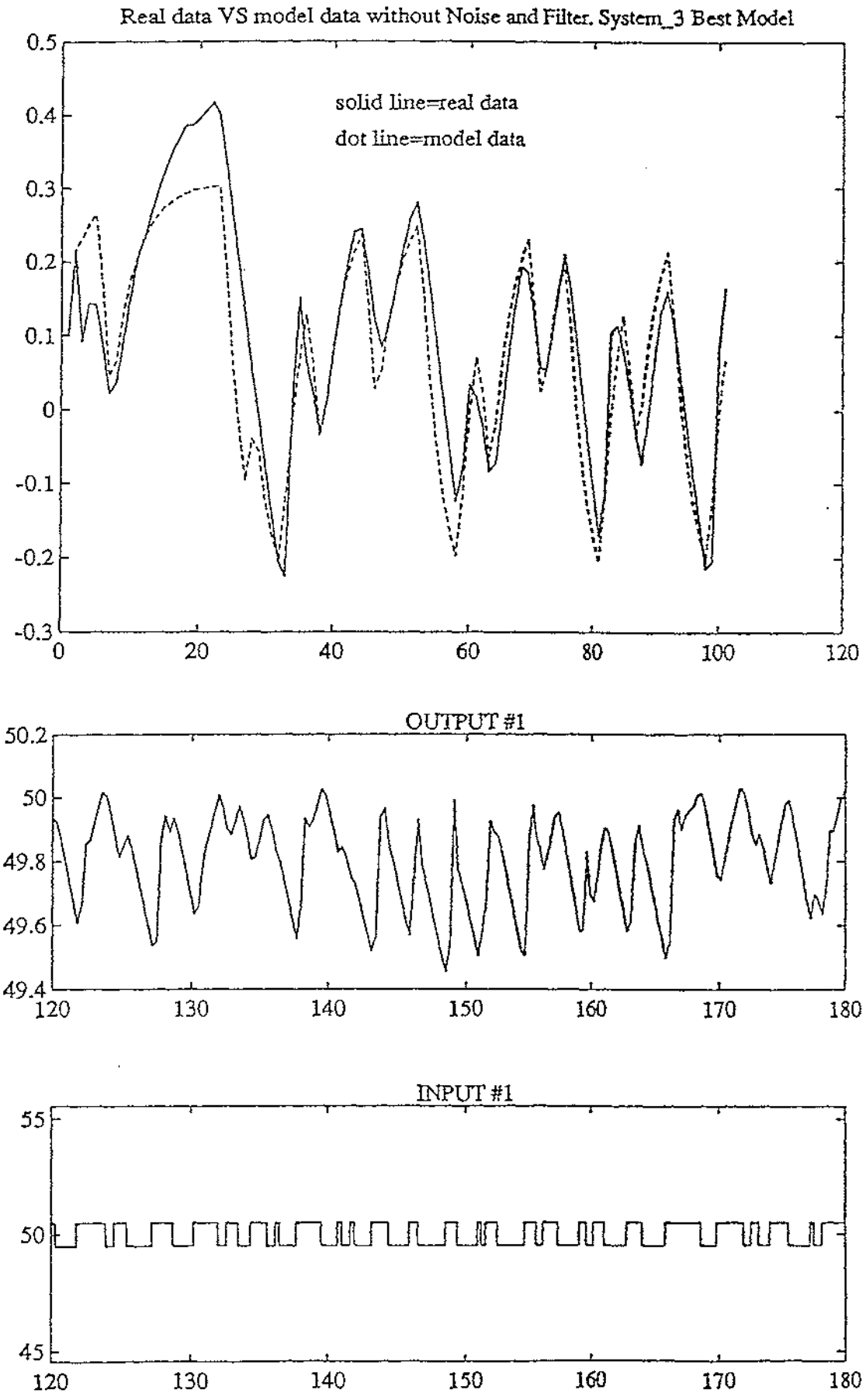
5.3.1 Discussion (System 3)

Figs 5.22 and 5.23 clearly show that, for both Butterworth and Chebychev filters a second order filter produces the best identification results. Again an increase in J is seen for ω_n > 5 times the plant frequency as noise begins to pass through the filter to the identification algorithm.

The system, due to the pulse width modulation and the oscillator has its own internal noise. So the effect of the filter with no measurement noise will therefore not be as good as for systems 1 and 2. The effect of the variable noise on the identification is not easily determined.

As measurement noise is included (Fig 5.24 and 5.25) the identification becomes more sensitive to the filter. It can be seen that the best ω_n is for n=2.

The model determined from the noisy data without filtering appears from fig 5.21 to have given excessive weigh to the high frequency modes of the system. Notice in fig 5.21 for points between 0 to 20 where low frequency components in the real data is significant as is not generated by this model.



Input(PRBS) & Output without Noise and Filter. System_3 Best Model

Fig 5.20 Input, Output and Comparison between Real data and Model data for System 3.

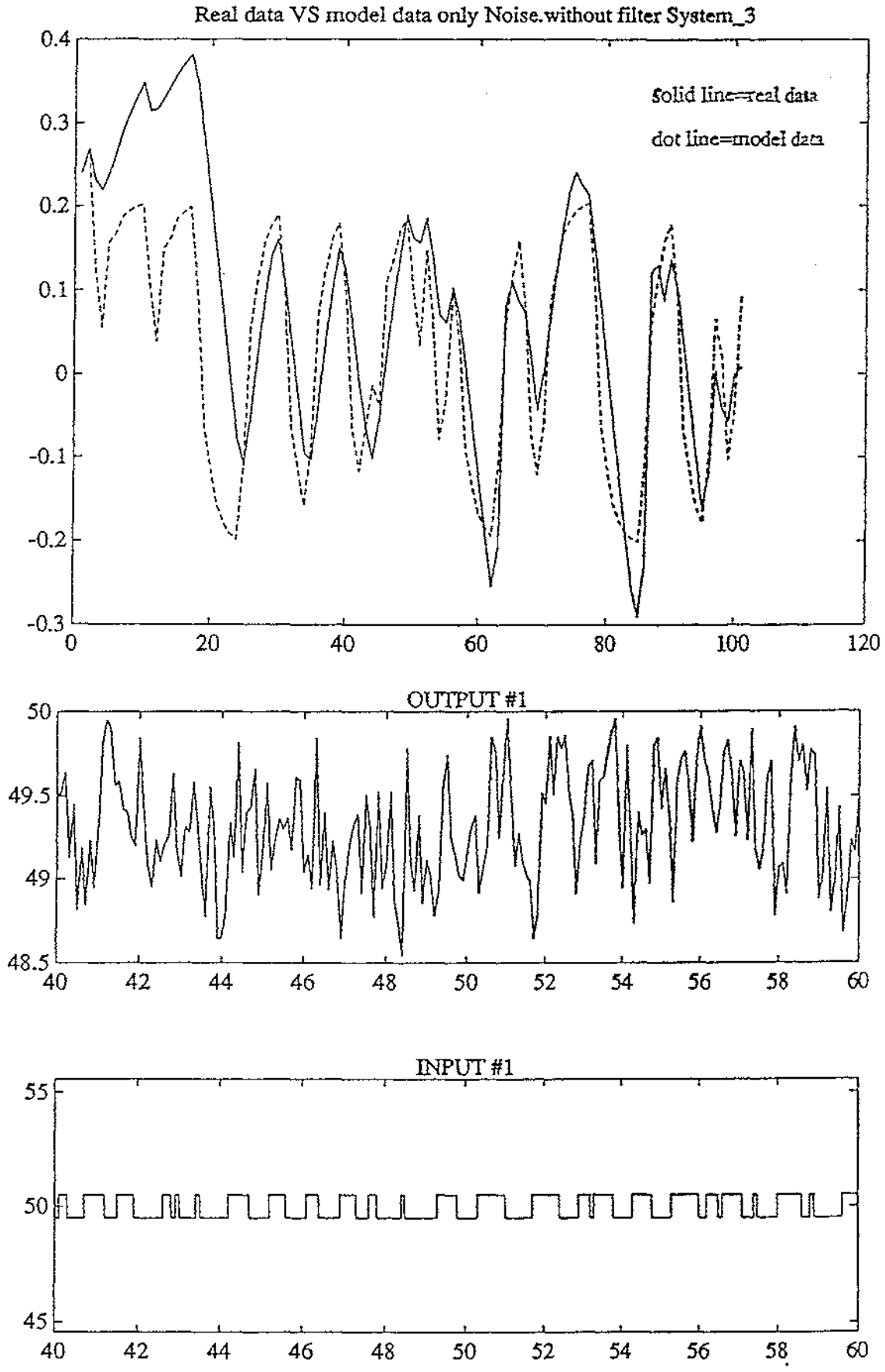


Fig 5.21 The effect of aliasing on the identified model for system 3.

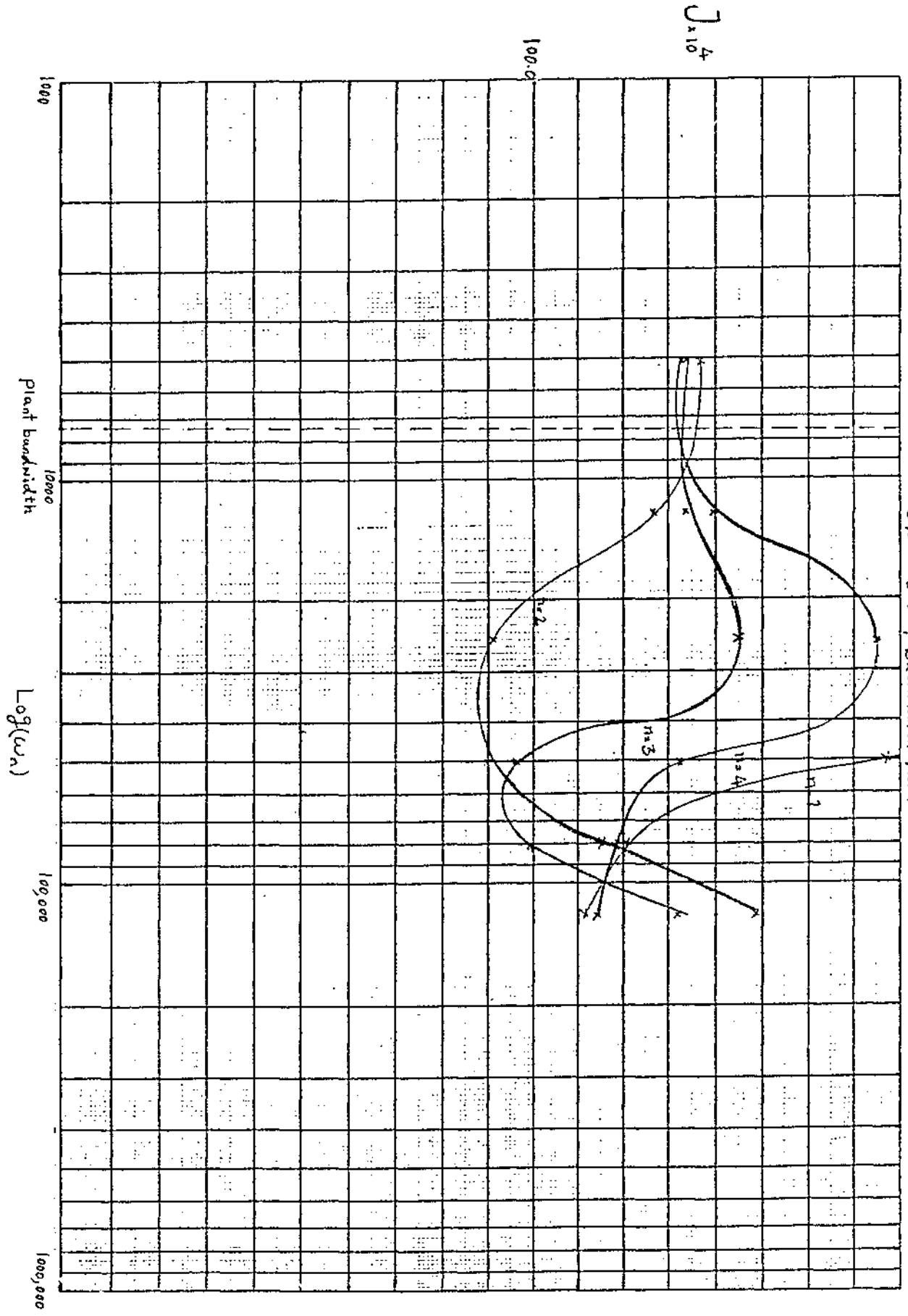


fig 5.22 "System 3-only Butterworth Filter"

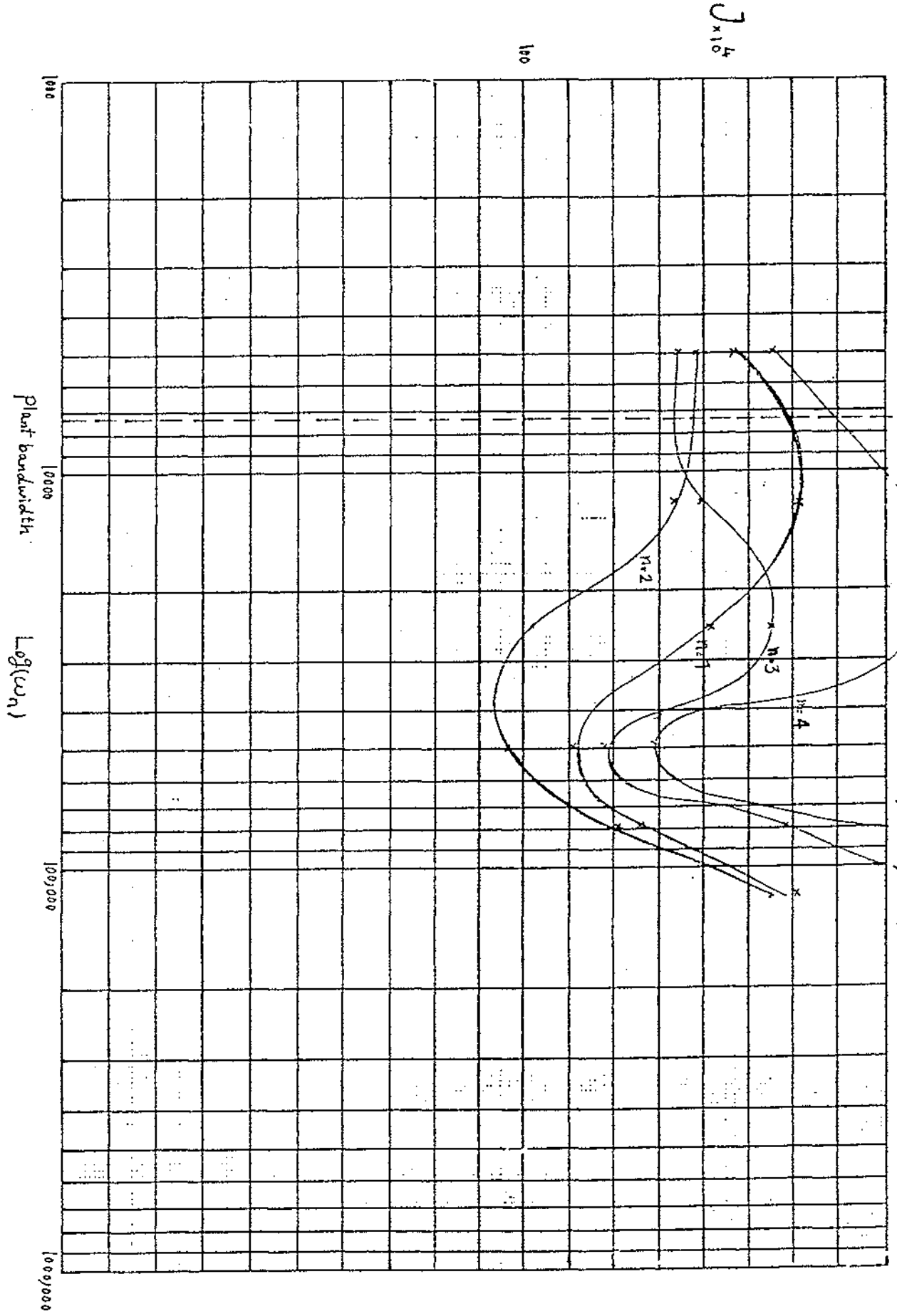


Fig. 5.23 "System 3-only Chebychev Filter"

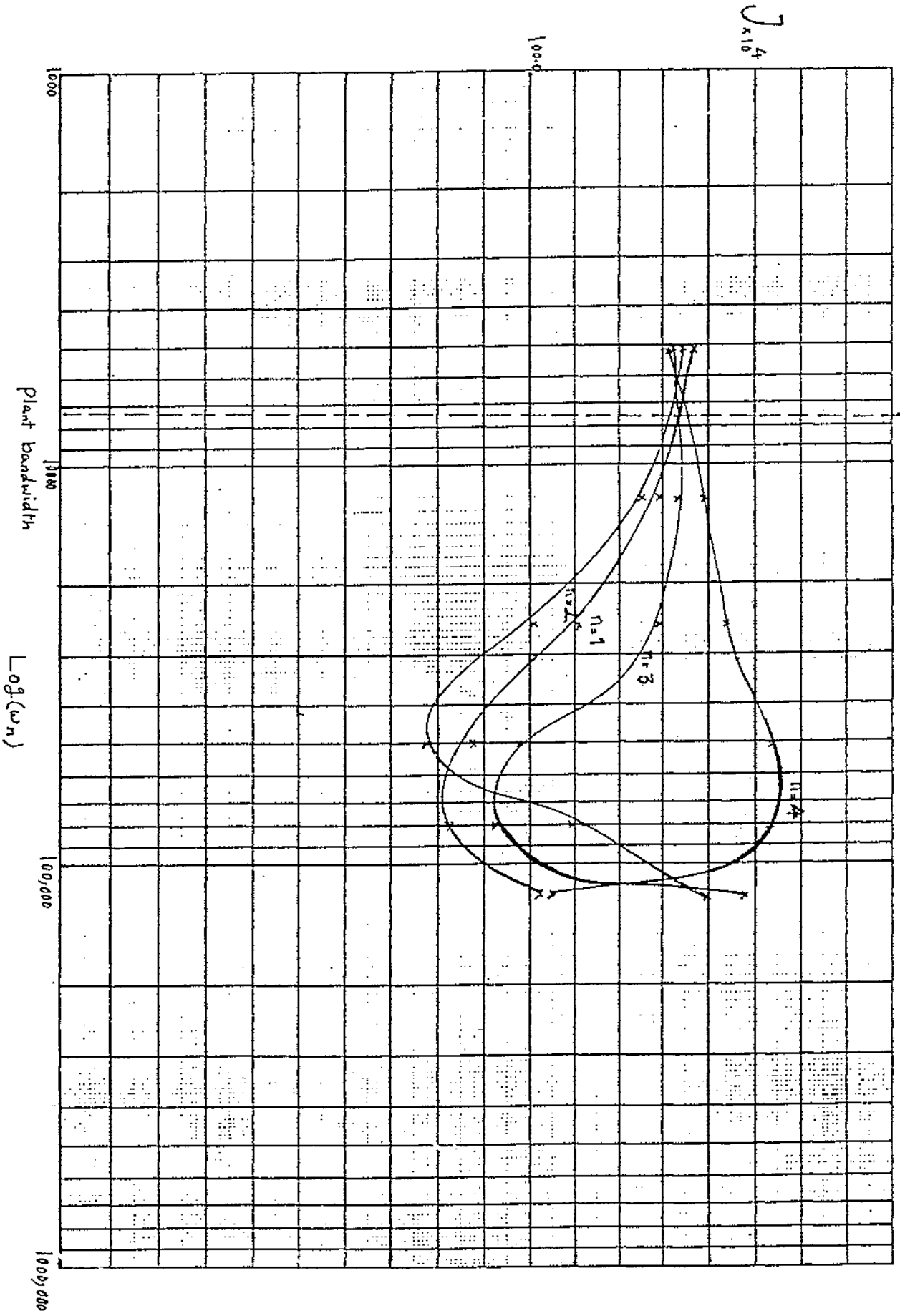


Fig 5.24 "System 3_Noise and Butterworth Filter"

Fig 5.25 "System 3-Noise and Chebychev Filter"

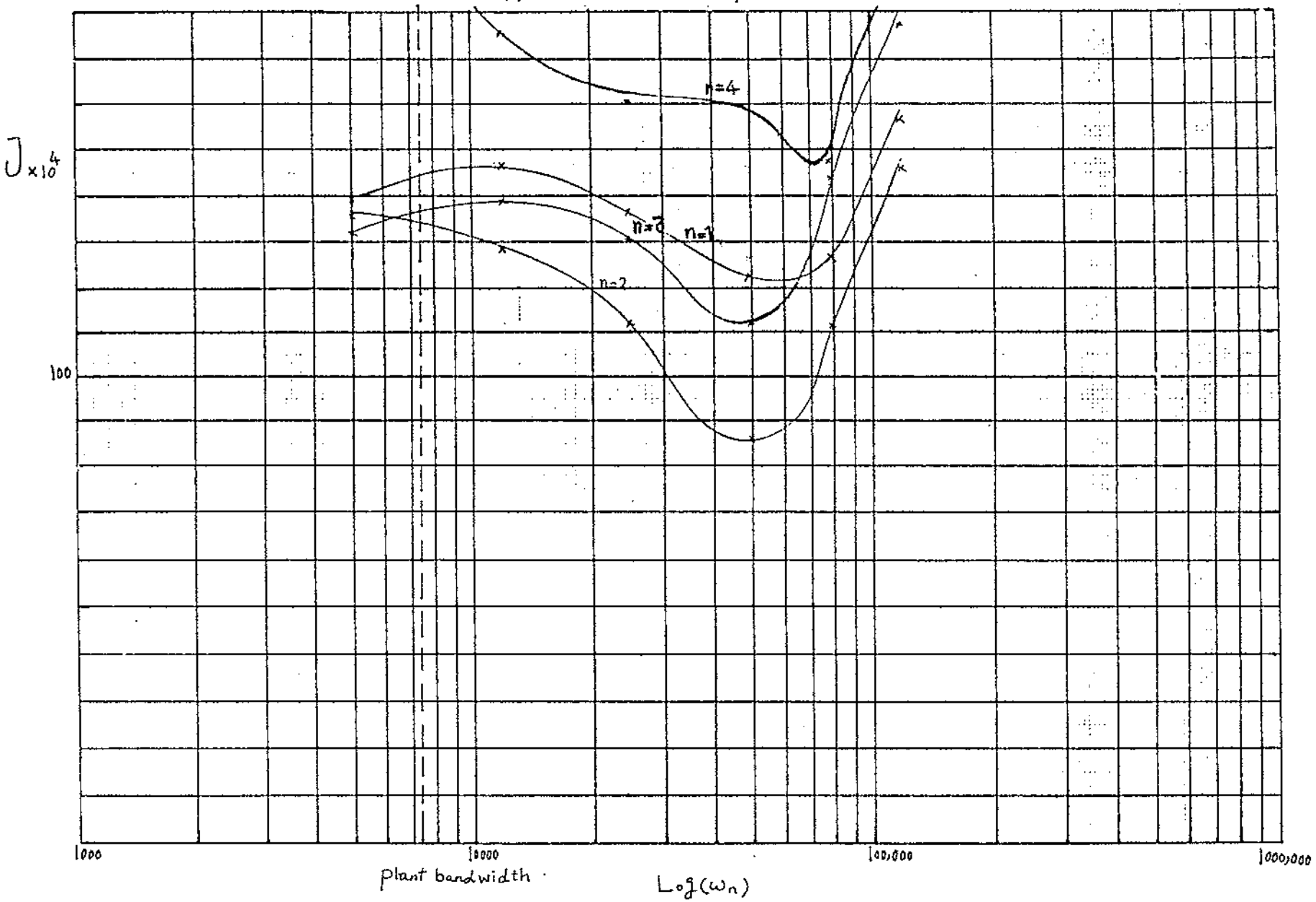


Table 9 System 3_ only Butterworth Filte

FILTER	J
S3B1ω5	0.2305
S3B1ω12	0.1235
S3B1ω25	0.0522
S3B1ω50	0.0195
S3B1ω80	0.0121
S3B1ω120	0.0113
S3B2ω5*	0.0137
S3B2ω12	0.0127
S3B2ω25	0.0092
S3B2ω50	0.0091
S3B2ω80	0.0116
S3B2ω120	0.0149
S3B3ω5	0.0134
S3B3ω12	0.0134
S3B3ω25	0.0145
S3B3ω50	0.0097
S3B3ω80	0.0100
S3B3ω120	0.0132
S3B4ω5	0.0134
S3B4ω12	0.0140
S3B4ω25	0.0175
S3B4ω50	0.0133
S3B4ω80	0.0120
S3B4ω120	0.0115

Table 10 System 3_ Noise and Butterworth Filter

FILTER	J
S3NB1ω5	0.0135
S3NB1ω12	0.0128
S3NB1ω25	0.0110
S3NB1ω50	0.0087
S3NB1ω80	0.0082
S3NB1ω120	0.0103
S3NB2ω5	0.0133
S3NB2ω12	0.0124
S3NB2ω25	0.0100
S3NB2ω50	0.0077
S3NB2ω80	0.0109
S3NB2ω120	0.0139
S3NB3ω5	0.0131
S3NB3ω12	0.0132
S3NB3ω25	0.0128
S3NB3ω50	0.0097
S3NB3ω80	0.0092
S3NB3ω120	0.0147
S3NB4ω5	0.0130
S3NB4ω12	0.0138
S3NB4ω25	0.0143
S3NB4ω50	0.0153
S3NB4ω80	0.0153
S3NB4ω120	0.0105

* S3B2ω5 means system 3 with Butterworth filter n=2 ω_n=5000.

Table 11. System 3_ only Chebychev Filter

FILTER	J
S3C1ω5	0.0145
S3C1ω12	0.0160
S3C1ω25	0.0140
S3C1ω50	0.0110
S3C1ω80	0.0125
S3C1ω120*	0.0158
S3C2ω53	0.0137
S3C2ω123	0.0132
S3C2ω253	0.0101
S3C2ω54	0.0096
S3C2ω84	0.0120
S3C2ω124	0.0154
S3C3ω53	0.0133
S3C3ω123	0.0138
S3C3ω253	0.0154
S3C3ω54	0.0117
S3C3ω84	0.0157
S3C3ω124	0.0190
S3C4ω53	0.0154
S3C4ω123	0.0183
S3C4ω253	0.0363
S3C4ω54	0.0128
S3C4ω84	0.0186
S3C4ω124	0.0225

Table 12. System 3_ Noise and Chebychev Filter

FILTER	J
S3NC1ω5	0.0138
S3NC1ω12	0.0144
S3NC1ω25	0.0135
S3NC1ω50	0.0121
S3NC1ω80	0.0125
S3NC1ω120	0.0155
S3NC2ω53	0.0134
S3NC2ω123	0.0127
S3NC2ω253	0.0111
S3NC2ω54	0.0085
S3NC2ω84	0.0111
S3NC2ω124	0.0144
S3NC3ω53	0.0130
S3NC3ω123	0.0137
S3NC3ω253	0.0129
S3NC3ω54	0.0111
S3NC3ω84	0.0142
S3NC3ω124	0.0176
S3NC4ω53	0.0291
S3NC4ω123	0.0174
S3NC4ω253	0.0159
S3NC4ω54	0.0157
S3NC4ω84	0.0146
S3NC4ω124	0.0205

* S3C1ω120 means system 3 with Chebychev filter n=1 ω_n=120000.

CHAPTER 6

CONCLUSIONS

The results produced by this research have shown that we must use a filter, otherwise we will get models in which the higher frequency modes belonging to the noise are identified rather than the lower frequencies belonging to the system. The following conclusions can also be drawn:

1) Lower order filters are recommended for identification, however, we should be very careful to design the filter with a suitable cut-off frequency to achieve the lowest error criterion (J), otherwise we will get worse results.

2) For filters in the absence of measurement noise, the step response of the closed loop identified model has a larger steady state error, overshoot and settling time. As the cut-off frequency of the filter increases the effect of the filter decreases.

3) When measurement noise is added two cases arise:

a. When $0.8\omega_s \leq \omega_n \leq k\omega_s$ the error criterion (J) decreases with an increase in ω_n. Where ω_n is the cut-off frequency of the filter, ω_s is the natural frequency of the system, k is a constant and depends on the characteristics of the system and possibly the noise spectrum. For the systems simulated k=5 for the linear system, k=3 for the nonlinear system and k=6 for the nonlinear discontinuous system.

b. When $\omega_n > k\omega_s$ J increases with ω_n.

Recommendation for further work include:

i) choosing more test systems

ii) Using variable noise bandwidth to determine the effect of this on the ratio $= \frac{\omega_n}{\omega_s}$

iii) using other identification methods

REFERENCES

Antoniou, A. (1979)
Digital Filters Analysis and Design, McGraw-Hill, New York.

Astrom, K. J. and Eykhoff, P. (1971)
System Identification - A Survey, Automatica, vol. 7, pp. 123-162.

Balakrishnan, A. V. and Thoma, M. (1980)
Identification of Continuous Dynamical Systems, Springer-Verlag, New York.

Distefano, J. (1967)
Feedback and Control Systems, McGraw-Hill, New York.

Dorf, R. C. (1980)
Modern Control Systems, Addison Wesley, Mass.

Emanuel, P. and Leff, E. (1979)
Introduction to Feedback Control Systems, McGraw-Hill, New York.

Godfrey, K. and Jones, P. (1986)
Signal Processing for Control, Springer-Verlag, New York.

Goodwin, G. C. and Sin, K. S. (1984)
Adaptive Filtering Prediction and Control, Prentice-Hall, Englewood Cliffs, N.J.

Hay, J. L. (1989)
ESL Manual, Salford University, Salford M5 4pp.

Hostetter, G. H. and Clement J. (1982)
Design of Feedback Control Systems, Holt, Rinehart and Winston, New York.

Hsia, T. C. (1977)
System Identification, Lexington Books, Lexington, Mass.

Isermann, R. (1989)
Digital Control Systems, Springer-Verlag, New York.

Kheir, N. A. (1988)
Systems Modeling and Computer Simulation, Marcel Dekker, Inc., New York.

Kuc, R. (1988)
Introduction to Digital Signal Processing, McGraw-Hill Co. Singapore.

Ljung, L. and Soderstrom, T (1983)
Theory and Practice of Recursive Identification, MIT.

Middleton, R. H. and Goodwin, G. C. (1990)
Digital Control and Estimation a Unified Approach, Prenticer-Hall, Englewood Cliffs, N.J.

Norton, J. P. (1986)
An Introduction to Identification, Academic Press, London.

Oppenheim, A. V. and Willsky, A. S. (1983)
Signals and Systems, Prentice-Hall, N.J.

Power, H. M. and Simpson, R. J. (1978)
Introduction to Dynamics and Control, McGraw-Hill, UK.

Proakis, J. G. and Manolakis, D. G. (1989)
Introduction to Digital Signal Processing, Macmillan Publishing Company, New York.

Rosoko, J. S. (1972)
Digital simulation of physical systems, Addison Wesley, Reading, MA.

Schwarzenbach, J. and Gill, K. F. (1984)
System Modeling and Control, Edward Arnold, London.

Sinha, N. K. and Kuszta, B. (1983)
Modeling and Identification of Dynamic Systems, Van Nostrand Reinhold Co. New York.

The Open University. (1984)
Analogue and Digital Filters, Great Britain.

Troch, I. (1978)
Simulation of Control Systems, North-Holland Publishing Company, Amsterdam.

Zeines, B. (1972)
Automatic Control Systems, Prentice-Hall, Englewood Cliffs, New Jersey.

APPENDIX A

Listings of the ESL programs for systems 1, 2 and 3 are given here.

ESL PROGRAM FOR SYSTEM 1

```
STUDY
--
INCLUDE "PRBSGEN";
INCLUDE "FILTER";
INCLUDE "LINEAR";
--
PROCEDURE Noise(REAL:FREQ,TIME)RETURN REAL;
REAL:N; N:=SIN((2.0*3.14*FREQ+RAND(1))*TIME);
RETURN N*RAND(1);
END Noise;
--
MODEL DATALOG (REAL:FY);
-------------------------------------------------------------
--ESL program DATALOG
-------------------------------------------------------------
  INCLUDE "FILTERB2";
  REAL :PRBS,Y,YNOISE;
  CONSTANT REAL :MAG/0.5/,MAXT/8.0/,FREQ/5.0/;
  FILE:Outfile;
INITIAL
REWRITE Outfile,"STUDY.OUT";
--
DYNAMIC
 PRBS      := PRBSGEN(MAG,MAXT);
 YNOISE    := Y+Noise(FREQ,T);
 FY,Z      := FILTER(A,B,C,D,YNOISE);
 Y         := LINEAR(PRBS);
--
STEP
  TABULATE Outfile,T,PRBS,Y,YNOISE,FY;
--
END DATALOG;
--
-- Experiment
--
-- Definition of experiment to be carried out on system
 REAL: FY;
-- Logical repeat variable
 LOGICAL: YES;
-- Define integration control parameters
 TSTART := 0.000000E+00; TFIN  := 300.0; CINT  := 1.0; DISERR := 0.100000E-03;
 INTERR := 0.100000E-02; ALGO  := 2; NSTEP  := 10.0; FY:=RAND(-1);
--
 LOOP
-- Invoke model
  DATALOG(FY);
  PRINT "STUDY COMPLETED";
-- Output simulation results
  PRINT "TABULATED RESULTS ARE:";
  TABULATE FY;
  READ "Do you want another run? ",YES;
  TERMINATE NOT YES;
  INTERACT;
 END_LOOP;
--
END_STUDY
```

ESL PROGRAM FOR SYSTEM 2

```
STUDY
--
INCLUDE "PRBSGEN";
INCLUDE "FILTER";
INCLUDE "NLINEAR";
--
PROCEDURE Noise(REAL:FREQ,TIME)RETURN REAL;
REAL:N; N:=SIN((2.0*3.14*FREQ+RAND(1))*TIME);
RETURN N*RAND(1);
END Noise;
--
MODEL DATALOG (REAL:FY);
------------------------------------------------------------
--ESL program DATALOG
------------------------------------------------------------
  INCLUDE "FILTERB2";
  REAL :CS,PRBS,CSNOISE;
  CONSTANT REAL :MAG/0.5/,MAXT/3.0/,FREQ/50.0/;
  FILE:Outfile;
INITIAL
REWRITE Outfile,"STUDY.OUT";
--
DYNAMIC
 PRBS       := PRBSGEN(MAG,MAXT);
 CSNOISE    := CS+Noise(FREQ,T);
 FY,Z       := FILTER(A,B,C,D,CSNOISE);
 CS         :=NLINEAR(PRBS);
--
STEP
  TABULATE Outfile,T,PRBS,CS,CSNOISE,FY;
--
END DATALOG;
--
-- Experiment
--
-- Definition of experiment to be carried out on system
 REAL: FY;
-- Logical repeat variable
 LOGICAL: YES;
-- Define integration control parameters
 TSTART := 0.000000E+00; TFIN  := 100.0; CINT  := 1.0; DISERR := 0.100000E-03;
 INTERR := 0.100000E-02; ALGO  := 2; NSTEP  := 100.0; FY:=RAND(-1);
--
 LOOP
-- Invoke model
  DATALOG(FY);
  PRINT "STUDY COMPLETED";
-- Output simulation results
  PRINT "TABULATED RESULTS ARE:";
  TABULATE FY;
  READ "Do you want another run? ",YES;
  TERMINATE NOT YES;
  INTERACT;
 END_LOOP;
--
END_STUDY
```

ESL PROGRAM FOR SYSTEM 3

```
STUDY
--
INCLUDE "REALPL";
INCLUDE "PICONT";
INCLUDE "LIMIT";
INCLUDE "MODULT";
INCLUDE "PRBSGEN";
INCLUDE "FILTER";
INCLUDE "SMPR";
--
 PROCEDURE Noise(REAL:FREQ,TIME)RETURN REAL;
 REAL:N; N:=SIN((2.0*3.14*FREQ+RAND(1))*TIME);
 RETURN N*RAND(1);
 END Noise;
--
MODEL DATALOG(REAL:FY);
---------------------------------
--ESL program DATALOG
---------------------------------
INCLUDE "FILTERB2";
-- Circuit constants
 CONSTANT REAL: L/2.1E-5/,RL/0.0/,CC/3.5E-4/,Rc/0.1/,FREQ/80000.0/;
 CONSTANT REAL: V1IC/0.0125/,Tf/2.0E-5/,MAG/0.5/,MAXT/0.00009/,R0/25.0/;
 CONSTANT REAL: V2IC/0.50/,Ti/4.5E-4/,G/1.0/,Vs/70.0/;
 CONSTANT REAL: LL/0.05/,UL/0.95/,Vref/50.0/;
 CONSTANT REAL: Td/0.0/,PERIOD/1.25E-5/;
-- Circuit variables
 REAL:V0,I0,Vin,IL,IC,VC,E,V1,Vip,W,PRBS,VrefP,V0NOISE;
FILE: OUTFILE;
-- Logical variables
 LOGICAL: TRAN_ON,IDIODE;
INITIAL
 REWRITE OUTFILE,"STUDY.OUT";
-- Initialse circuit state variables
 VC:= 50.0;
 IL:= 0.0;
DYNAMIC
-- Output current and voltage
 I0    := (VC+IL*Rc)/(R0+Rc);
 V0    := SMPR(I0);
 PRBS  := PRBSGEN(MAG,MAXT);
 VREFP := VREF+PRBS;
 V0NOISE:= V0+NOISE(FREQ,T);
 FY,Z  := FILTER(A,B,C,D,V0NOISE);
-- Capacitor current
 IC:= IL-I0;
-- Error signal
 E:= VrefP-V0;
-- Call to filter submodel
 V1:= REALPL(V1IC,Tf,E);
-- Call to PI controller submodel
 Vip:= PICONT(V2IC,Ti,G,V1);
-- Call to limiter submodel
 W:= LIMIT(LL,UL,Vip);
-- Determine transistor state (call to PWM submodel)
```

```
  TRAN_ON:= MODULT(Td,W,PERIOD);
-- Determine diode state (conducting if IL>=0.0)
  IDIODE:= IL >= 0.0;
-- Determine state of input voltage to LC stage
  Vin:= if TRAN_ON then Vs
        else_if IDIODE then 0.0
        else V0;
-- Current through inductor
  IL':= (Vin-RL*IL-V0)/L;
-- Voltage across capacitor
  VC':= IC/CC;
COMMUNICATION
  PREPARE "SMPR",T,VREFP,V0,V0NOISE,FY;
  TABULATE OUTFILE,T,VREFP,V0,V0NOISE,FY;
--
END DATALOG;
-- EXPERIMENT
-- Define variable circuit parameters
  REAL: FY;
--  Logical repeat variable
LOGICAL:YES;
-- Define integration control parameters
  ALGO:=RK4;   CINT:= 6.25E-6;   NSTEP:= 1.0;   TFIN:=12.5E-3;
  FY:=RAND(-1);
--
LOOP
--Invoke model
DATALOG(FY);
PRINT "STUDY COMPLETED";
--Output simulation results
PRINT "TABULATED RESULTS ARE:";
TABULATE FY;
READ "Do you want another run?",YES;
TERMINATE NOT YES;
INTERACT;
END_LOOP;
--
END_STUDY
```

Submodel LINEAR for system 1

```
SUBMODEL LINEAR(REAL:Y := REAL:PRBS);
-------------------------------------------------------------
-- ESL subprogram LINEAR
-------------------------------------------------------------
DYNAMIC
  Y:=TRANSFER(1/(S**2+1.414*S+1))*PRBS;
STEP
  PREPARE "LINEAR", T,PRBS,Y;
--
END LINEAR;
--
```

Submodel NLINEAR for system 2

```
SUBMODEL NLINEAR(REAL: CS:= REAL: PRBS);
-------------------------------------------------------------
-- ESL subprogram NLINEAR
-------------------------------------------------------------
REAL :F,Ci,K1;
INITIAL
  CS := 0.5;
DYNAMIC
  F  := 0.5;
  Ci := PRBS+0.5;
  K1 := 1.0;
  CS' :=F*Ci-F*CS-K1*CS**2;
STEP
  PREPARE "NLINEAR",T,PRBS,CS;
--
END NLINEAR;
--
```

Submodel SMPR for system 3

```
SUBMODEL SMPR(REAL:V0 := REAL:I0);
-------------------------------------------------------------
--ESL subprogram SMPR
-------------------------------------------------------------
CONSTANT REAL:R0/25.0/;
DYNAMIC
V0 :=I0*R0;
--
END SMPR;
--
```

Submodel MODULT for system 3

```
SUBMODEL MODULT(LOGICAL: Y:= REAL: Td,sig,per);
-----------------------------------------------------------------
-- Logical pulse width modulator which generates a logical
-- pulse train with specified period and a mark-space
-- ratio. An initial delay is permitted, and the initial
-- output may be specified as TRUE or FALSE. The calling
-- sequence is:
--
--   y:= MODULT(Td,sig,per)
--
-- where,
--
-- Td is the time at which the pulse train starts. If
--   Td >= 0.0, y will remain FALSE for Td seconds. If
--   Td < 0.0, pulse train will remain TRUE for
--   (-Td) seconds.
-- sig is the modulating signal in the range [0,1],
-- per is the period of the pulse train in units of T.
--
-- The output is a memory variable.
--
-----------------------------------------------------------------
 REAL: start,ramp;
INITIAL
 if Td > 0.0 then
   Y:= FALSE;
   start:= TSTART+Td-per;
 else_if Td < 0.0 then
   Y:= TRUE;
   start:= TSTART+ABS(Td)-per*sig;
 else
   Y:= TRUE;
   start:= TSTART;
 end_if;
DYNAMIC
 ramp:= (T-start)/per;
 when ramp >= sig then
   Y:= FALSE;
 when ramp >= 1.0 then
   start:= start+per;
   Y:= TRUE;
 end_when;
--
END MODULT;
```

Submodel PICONT for system 3

```
SUBMODEL PICONT(REAL: y:= REAL: IC,TC,K,x);
-------------------------------------------------------------
-- This submodel defines a proportional plus integral (PI)
-- controller. The calling sequence is:
--
--   y:= PICONT(IC,TC,K,x)
--
-- where,
--
-- IC is the integrator initial condition, z(TSTART) = IC,
-- TC is the time constant of the integrator,
-- K is the proportional gain,
-- x is the input variable.
--
-- The differential equations are given by,
--
--   z' = x/TC
--
--   y = K*(x+z)
--
-- and the equivalent Laplace Transform function is,
--
--       y(s)       K
--       ---- = K + ---- .
--       x(s)      s*TC
--
-- The output is an algebraic variable.
--
-------------------------------------------------------------
 REAL: z;
INITIAL
 z := IC;
DYNAMIC
 z' := x/TC;
 y := K*(x+z);
--
END PICONT;
```

Submodel LIMIT for system 3

```
SUBMODEL LIMIT(REAL: y:= REAL: LL,UL,x);
-----------------------------------------------------------------
-- A limiter sets lower and upper limits on the amplitude
-- of an input variable. The calling sequence is:
--
--    y:= LIMIT(LL,UL,x)
--
-- where,
--
-- LL is the lower limit,
-- UL is the upper limit,
-- x is the input variable.
-- N.B. UL > LL.
--
-- y is given a value such that,
--
-- y = x, if LL < x < UL,
-- y = UL, if x >= UL,
-- y = LL, if x <= LL.
--
-- The output is an algebraic variable.
--
-----------------------------------------------------------------
 REAL: range,xnorm;
INITIAL
 if LL >= UL then
   print "**** Error in LIMIT: Limits not consistent";
   STOP;
 end_if;
 range:= UL-LL;
DYNAMIC
 xnorm:= (x-LL)/range;
--
 y:= if xnorm > 1.0 then UL
    else_if xnorm < 0.0 then LL
        else x;
--
END LIMIT;
```

Submodel REALPL for system 3

```
SUBMODEL REALPL(REAL: y:= REAL: IC,P,x);
-----------------------------------------------------------------
-- Generates a real pole transfer function. The calling
-- sequence is:
--
--    y:= REALPL(IC,P,x)
--
-- where,
--
-- IC is the initial condition, y(TSTART) = IC,
-- P is a constant,
-- x is the input variable.
--
-- The differential equation is given by,
--
--    P*y'+y = x
--
-- and the equivalent Laplace Transform function is,
--
--    y(s)      1
--    ---- = -------- ,
--    x(s)   P*s + 1
--
-- The output is a memory variable.
--
-----------------------------------------------------------------
INITIAL
  y:= IC;
DYNAMIC
  y':= (x-y)/P;
--
END REALPL;
```

Submodel PRBS for all systems

```
SUBMODEL PRBSGEN( REAL: PRBS := REAL: MAG,MAXT);
------------------------------------------------------------
-- ESL subprogram PRBSGEN
------------------------------------------------------------
 REAL: NEXT;
 INTEGER: I;
INITIAL
 I := 1;
 PRBS:=MAG;
 NEXT:=RAND(-MAXT);
DYNAMIC
 WHEN T >= NEXT THEN
  NEXT := T + RAND(MAXT);
  PRBS := I*MAG;
  I := -I;
 END_WHEN;
END PRBSGEN;
--
```

Submodel Filter for all systems

```
SUBMODEL FILTER (REAL: FY,Z(*):=REAL: A(*,*),B(*),C(*,*),D(*),INPUT);
------------------------------------------------------------
--ESL subprogram FILTER
------------------------------------------------------------
INITIAL
  Z(4):=0.0;
DYNAMIC
  Z':=A*Z+B*INPUT;
  FY :=C*Z+D*INPUT;
STEP
  PREPARE "FILTER",T,FY,INPUT;
END FILTER;
```

MATLAB Program for Identification

%We select the first 600 data points for building a model. For convenience, the input-output vectors are merged into a matrix:

z=[fy(1:600) prbs(1:600)];

%Let us first take a look at the data, we can select the values between sample numbers 400 and 600 for a closeup, and at the same time obtain correct time scales, with:

idplot(z,400:600,0.3)

% xlabel('Input(PRBS) & Output without noise and filter. System_1 Best Model')

% print

%We can remove the constant levels and make the data zero mean with

z=detrend(z);

%Now let us fit to the data a model of the form:

% y(t)+a1y(t-T)+a2y(t-2T)=b1u(t-T)+b2u(t-2T) (1.1)

%where T is the sampling interval. This model, known as an ARX-model,tries to "explain" or compute the value of the output at time t,given previous values of y and u. The best values of the coefficients a_1, a_2, b_1 and b_2 can be computed with:

th=arx(z,[2 2 1]);

% The numbers in the second argument tell arx to find a model (1.1) with 2 a-parameters, 2 b-parameters and 1 delays. The result is stored in the matrix th in a some what coded form. The sampling interval is stored in normalized form (equal to 1.0) in element th(1,2). To specify the actual sampling interval, enter

th(1,2)=0.3;

There are several ways to display and illustrate the computed model. with

present(th)

% the coefficient values of(1.1) and their estimated standard deviations are presented on the screen.

pause % press any key to continue.

% We can remove the constant levels and make the data zero mean with 'detrend'.

```
%Next, you might ask, can we evaluate how well the model fits the data? A simple test
is to run a simulation whereby real input data is fed into the model,and to compare the
simulated output with the actual, measured output. For this we select a portion of the
data that was not used to build the model, for example from sample 900  to 1000:
yp=detrend(y(900:1000));
up=detrend(prbs(900:1000));
yh=idsim(up,th);
plot([yp yh]),pause
% title('Real data VS model data without noise and filter. System_1 Best Model')
% print
```

MATLAB program for error criterion (J)

```
ydiff=yp(900:1000)-yh;
ydiffsqu=ydiff'*ydiff;
yroot=sqrt(ydiffsqu);
J=yroot/100
```

APPENDIX B

System 3 is explained here in more detail.

2.3 Switched-Mode Power Regulator

2.3.1 Circuit Elements

I. Power Circuit

The power circuit consists of a switch (transistor plus diode), an inductor/capacitor (LC) filter stage and the load resistance R_O. The logical output of the PWM (pulse-width modulator), drives the switch which connects the supply voltage (V_s) to the LC filter.

There are two modes of operation of the SMPR. These depend on the current (I_l) flowing into the LC filter, and are known as the discontinuous and continuous mode. In the discontinuous mode, I_l returns to zero during each period (of the PWM) whereas, in the continuous mode it dose not. The results presented, clearly show the distinction between these two modes of operation.

It is easy to explain the action of the SMPR with reference to the voltage input (V_{in}) to the LC filter which determines I_l. There may be three states during each period of the PWM:

(1) When the transistor is ON it effectively connects the supply voltage to power circuit, ie:

$$V_{in} = V_S$$

(2) During the period when the transistor is ON, the inductor will store energy. When the transistor switches OFF, this stored energy will be transferred by I_l continuing to flow through the *free-wheeling* diode which is forward biased. Hence, V_{in} is the voltage across the diode when conducting and assuming a perfect diode:

$$V_{in} = 0.0$$

(3) The energy stored, and also I_l, will decay to zero and the diode will stop conducting. Then V_{in} is in parallel with, and equal to the sum of the voltage drop across the inductor (ie $R_l * I_l$) and the output voltage (V_O). In this case, however, I_l will be zero and so:

$$V_{in} = V_o$$

The above describes the discontinuous mode of operation, whereas in the continuous mode state (3) does not apply as the system changes directly from state (2) back to state (1).

The current I_l is given by the differential equation :

$$I_l' = (V_{in} - R_l * I_l - V_o) / L$$

The output current (I_o), output voltage (V_o), capacitor current (I_c) and capacitor voltage (V_c) have the following dynamic equations:

$$I_o = \frac{(V_c + I_l * R_c)}{(R_o + R_c)}$$

$$V_o = I_o * R_o$$

$$I_c = I_l - I_o$$

$$V_c' = I_c / C$$

II. Pulse-Width Modulator

The function of the PWM (pulse-width modulator) is to provide the timing pulses to the base of the transistor which in turn controls the state of the transistor. The PWM works by using a ramp timing waveform as shown in Fig 2.3 There are two input signals, namely:

(i) the sampling frequency (fo),

(ii) the mark-space ratio control signal (w).

Note that fo is the output pulse frequency which has a mark-space ratio of w.

The action of the PWM is described as follows:

The slope of the ramp waveform (RAMP) is such that it's value is 0.0 at the start of a period and 1.0 at the end. This is achieved by calculating the ramp as :

RAMP = time/period

where time is the time from the start of the current period.

Initially the output (which may be HIGH or LOW, ie 1 or 0) is set HIGH. As the ramp rises, two separate triggering points are passed. First, when the value of the ramp

becomes equal to the input w, the output becomes LOW. Second, when the ramp becomes equal to 1.0 (which occurs at the end of each period), the output returns to HIGH state and the ramp is reset to 0.0. The process is then repeated. Note, $0.05 \leq w \leq 0.95$ (see limiter).

III. **Limiter**

The mark-space ratio control signal input to the PWM must not be too large or too small. It can be seen from Fig 2.3 that if this signal is greater than 1.0 or less than 0.0 then it will be outside the range of the ramp waveform, and so a limiter is used to keep the signal within this range.

The limits should be close to 0.0 and 1.0 whilst, giving a reasonable tolerance. For this reason a 5% cut-off was chosen giving the upper and lower limits:

$$UL = 0.95 \text{ and } LL = 0.05.$$

IV. **PI Controller**

The controller, shown in Fig 2.2, is a proportional plus integral controller. The equations of the controller are :

$$V_2' = V_1 / T_i \qquad (1)$$

and

$$V_{ip} = G * (V_1 + V_2) \qquad (2)$$

It is assumed supply voltage V_s = 70.0, and output resistance is 25.0 Ohms. The output of the PWM causes the transistor to operate and pass 70 volts to the power circuit when the PWM output is a maximum. It therefore has an effective gain of approximately 70 and the gain G of the PI controller may be chosen to be:

$$G = 1.0$$

The integrating time constant (T_i) is such that the oscillatory action introduced by the LC filter stage in the power circuit is removed. From control theory this can be found using Zeigler-Nichols rules for 'PI' controllers. That is:

$$T_i = T_u / 1.2$$

where

$$T_u = (2.0 * \pi) / \omega_n$$

ω_n is the natural frequency of oscillation .

The natural frequency of the LC filter is given by:

$$\omega_n = \frac{1}{SQRT(L * C * (1 + R_C / R_O))}$$

The values of L, C, R_l and R_C are selected as follows:

$$L = 2.1E\text{-}5 \text{ Henrys} \quad C = 3.5E\text{-}4 \text{ Farrads}$$

$$R_l = 0.0 \text{ Ohms} \quad R_C = 0.1 \text{ Ohms} \quad R_O = 25.0 \text{ Ohms}$$

Substituting values for L, C, R_C and R_O we obtain the value of T_i as:

$T_i = 4.5E\text{-}4$ seconds.

V. **Filter**

The purpose of the filter is to reduce the ripple introduced by the sampling frequency (fo) of the PWM.

The transfer function of the filter is given by:

$$V1(s) = \frac{1}{1+(s*T_f)} * E(s)$$

from which the differential equation is:

$$V1' = \frac{E - V1}{T_f}$$

To achieve satisfactory filtering the filter's break frequency (fb) is given by:

$$fb = fo/10$$

Hence the time constant (T_f) of the filter is:

$$T_f = 1 / (2*\pi*fb)$$

which gives:

$T_f = 2.0E\text{-}5$ seconds.

This gives an attenuation of 0.1 at the ripple frequency (fo = 80 KHZ).

The switched-mode power supply can be represented in ESL by the model DATALOG shown in Appendix A. The power circuit is described by algebraic and differential equations, whilst, the control circuit is described by calls to the submodels.

The dynamic action of the power circuit LC filter stage is described by the equations given above. The state variables V_C and I_l are initialised in the INITIAL region.

The discontinuous mode action of V_{in} is described by means of an ' if ' statement within the DYNAMIC region of the model. Two logical variables are declared, these being TRAN_ON (which is the output of the submodel MODULT and , therefore determines whether the transistor is ON or OFF), and IDIODE (which represents the conducting state of the diode, and is TRUE for forward biasing, ie when $I_l \geq 0.0$).

By means of the ' if ' statement Vin is set to :

$V_{in} = V_S$ if TRAN_ON is TRUE

$V_{in} = 0.0$ if TRAN_ON is FALSE but IDIODE is TRUE

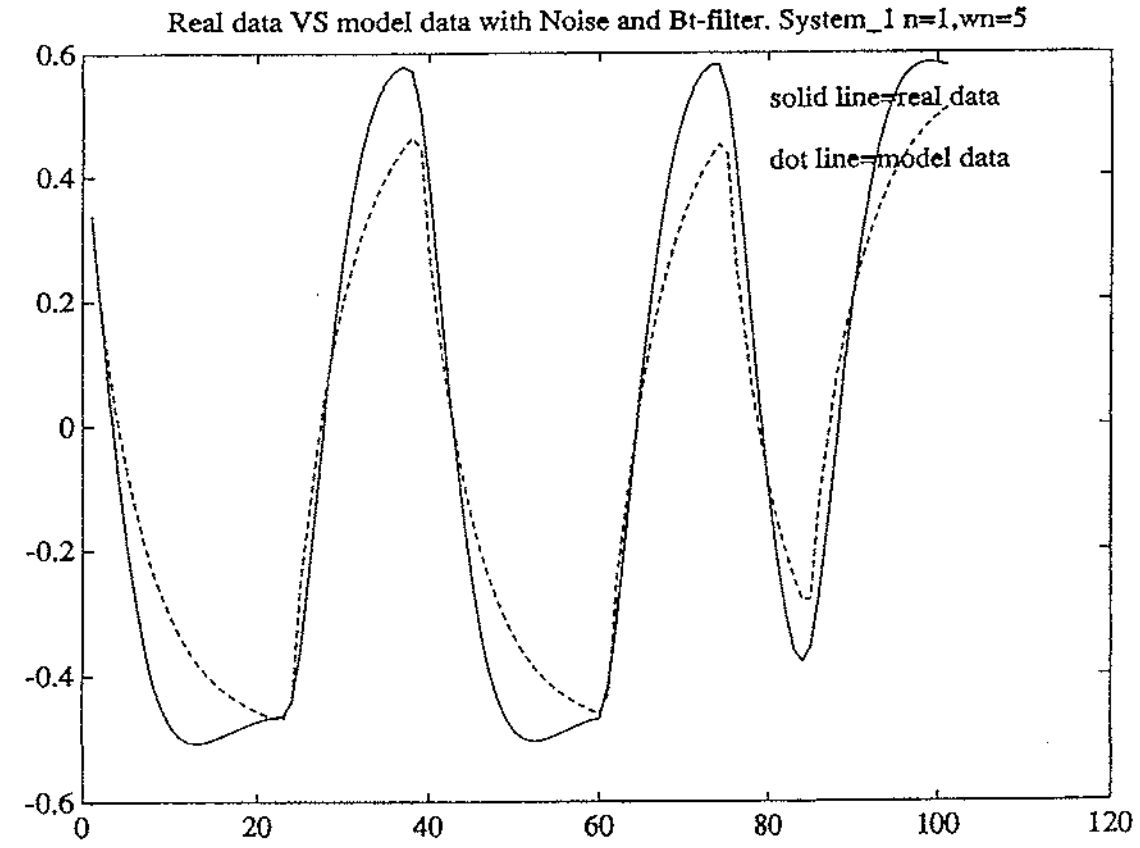
$V_{in} = V_O$ if TRAN_ON is FALSE and IDIODE is FALSE

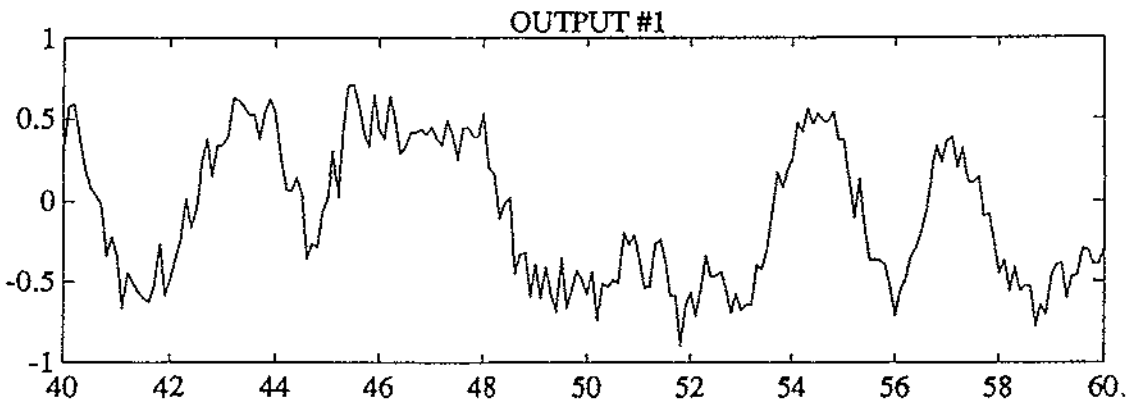
The error voltage (E) is calculated by simple subtraction. The outputs of the blocks in the control circuit are obtained by calls of their respective submodels.

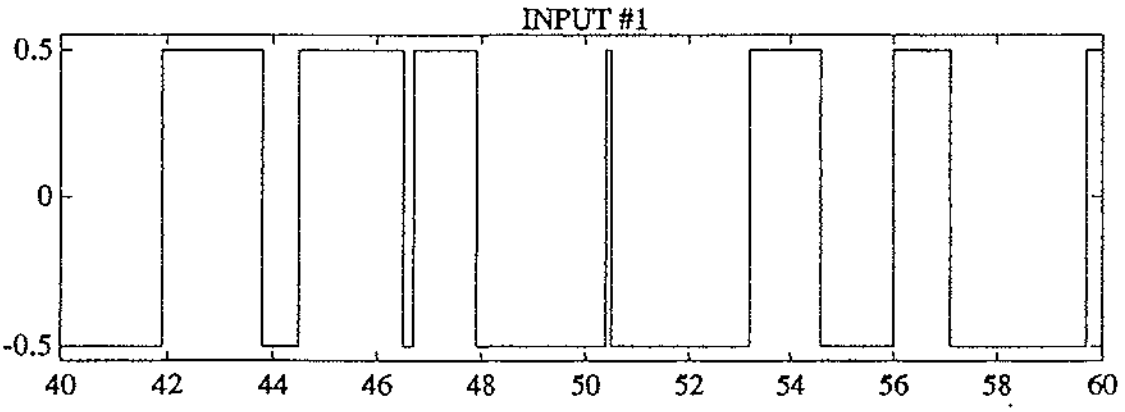
The COMMUNICATION region contains appropriate statements for output and plotting.

APPENDIX C

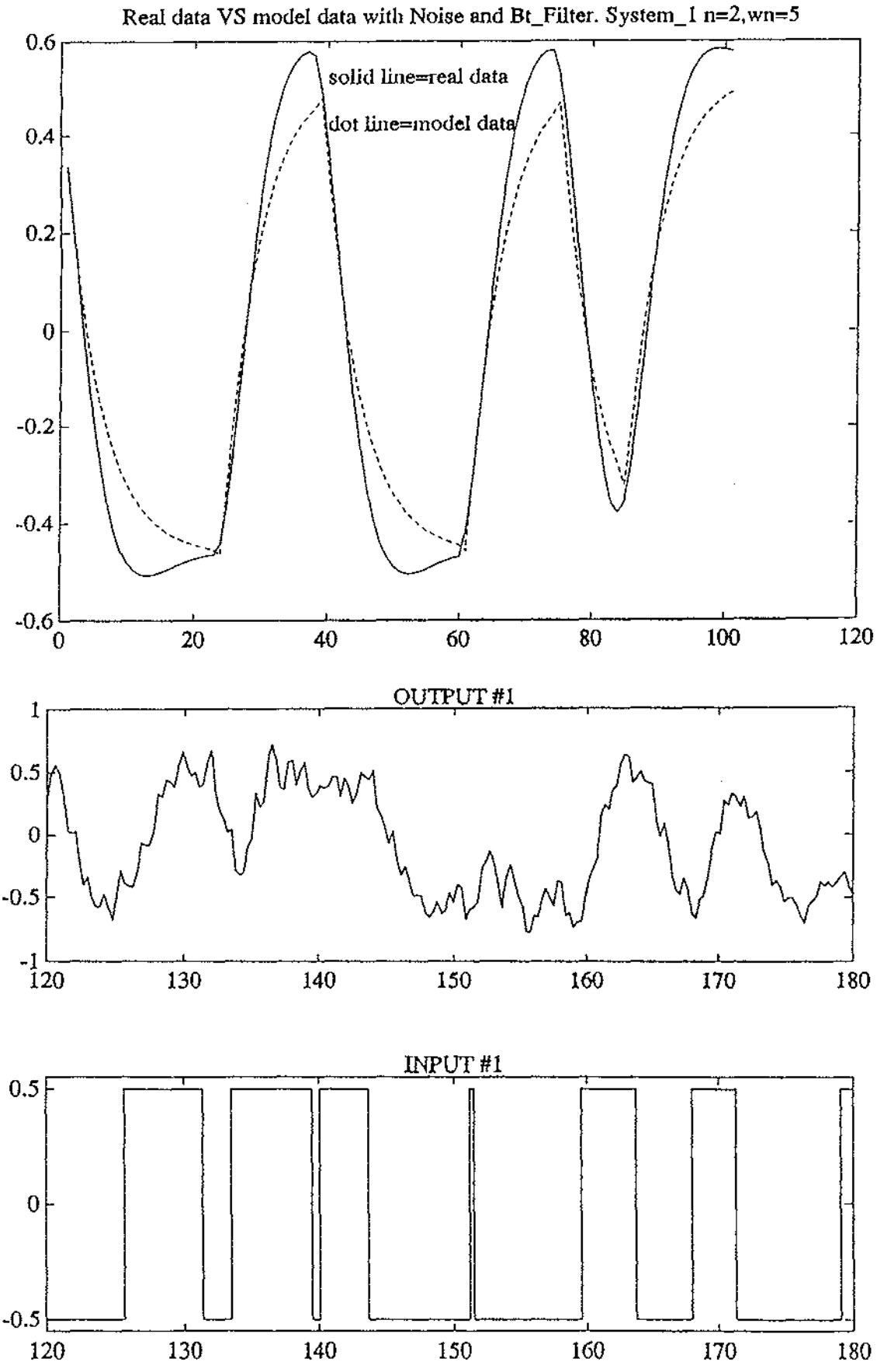
This appendix shows the effect of the different filters on the identified model. The filters were chosen from the best response of each order of the filter for systems 1, 2 and 3 (refer to tables to find J).



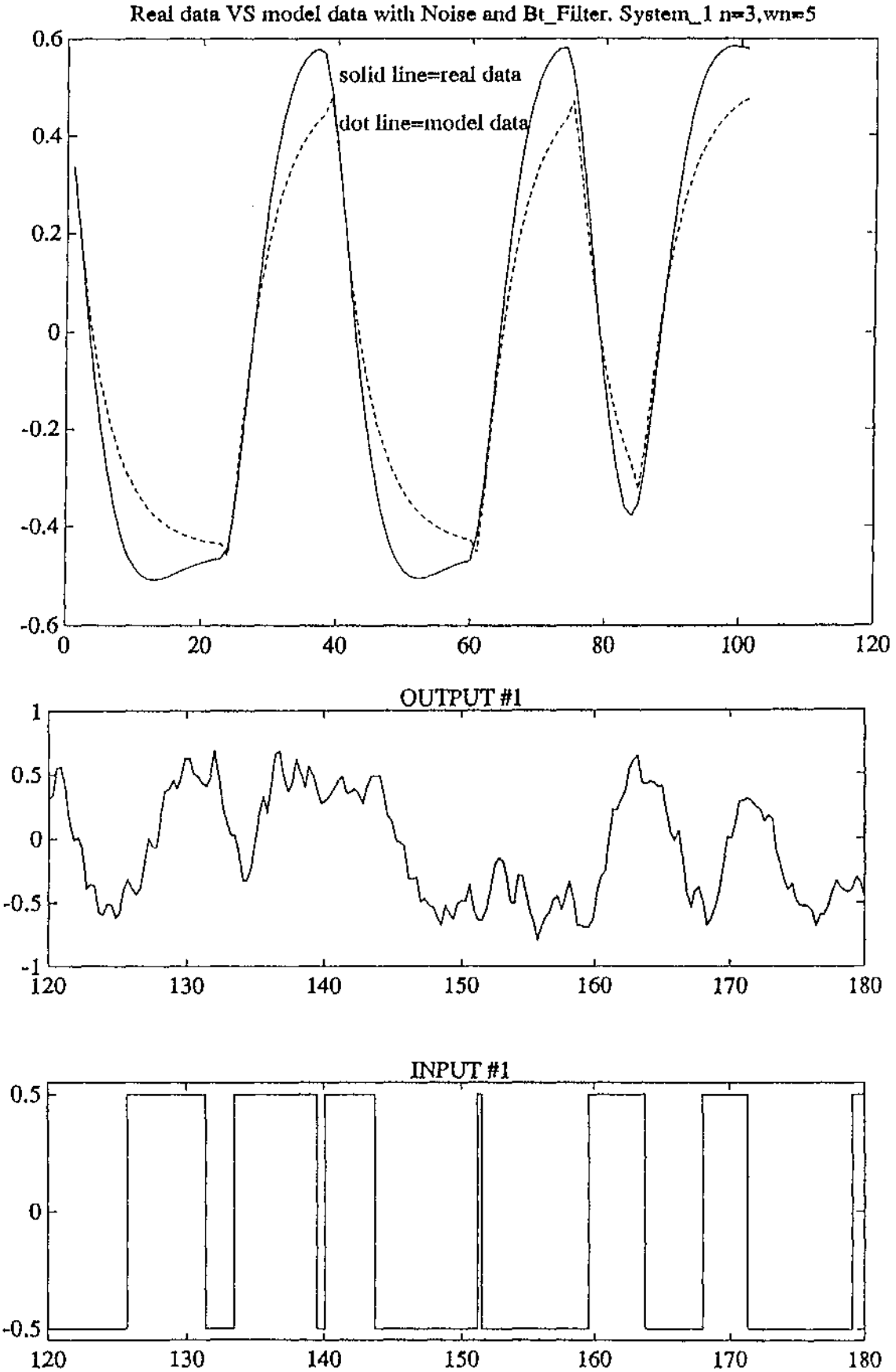




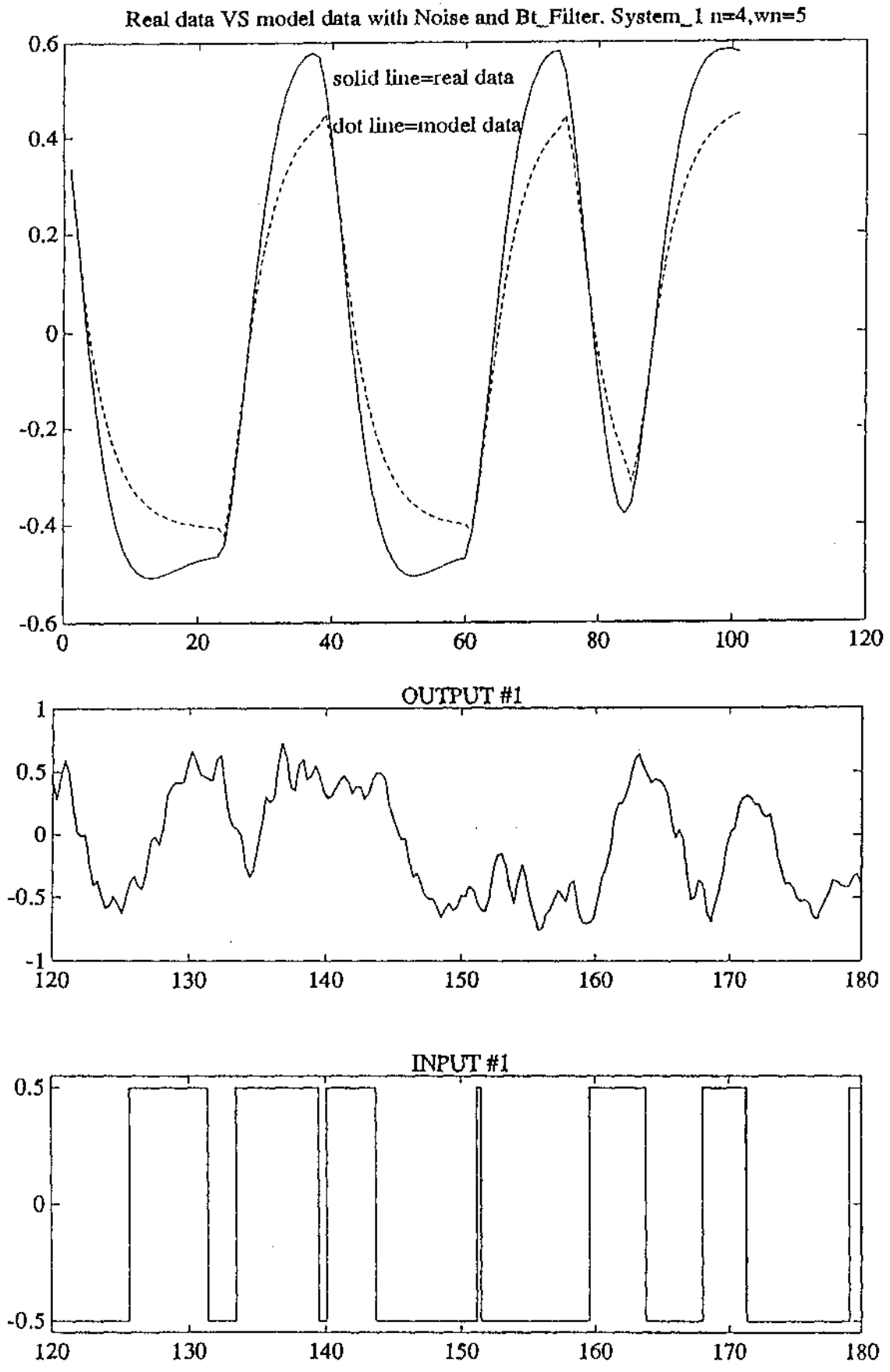
Input(PRBS) & Output with Noise and Bt-filter. System_1 n=1,wn=5



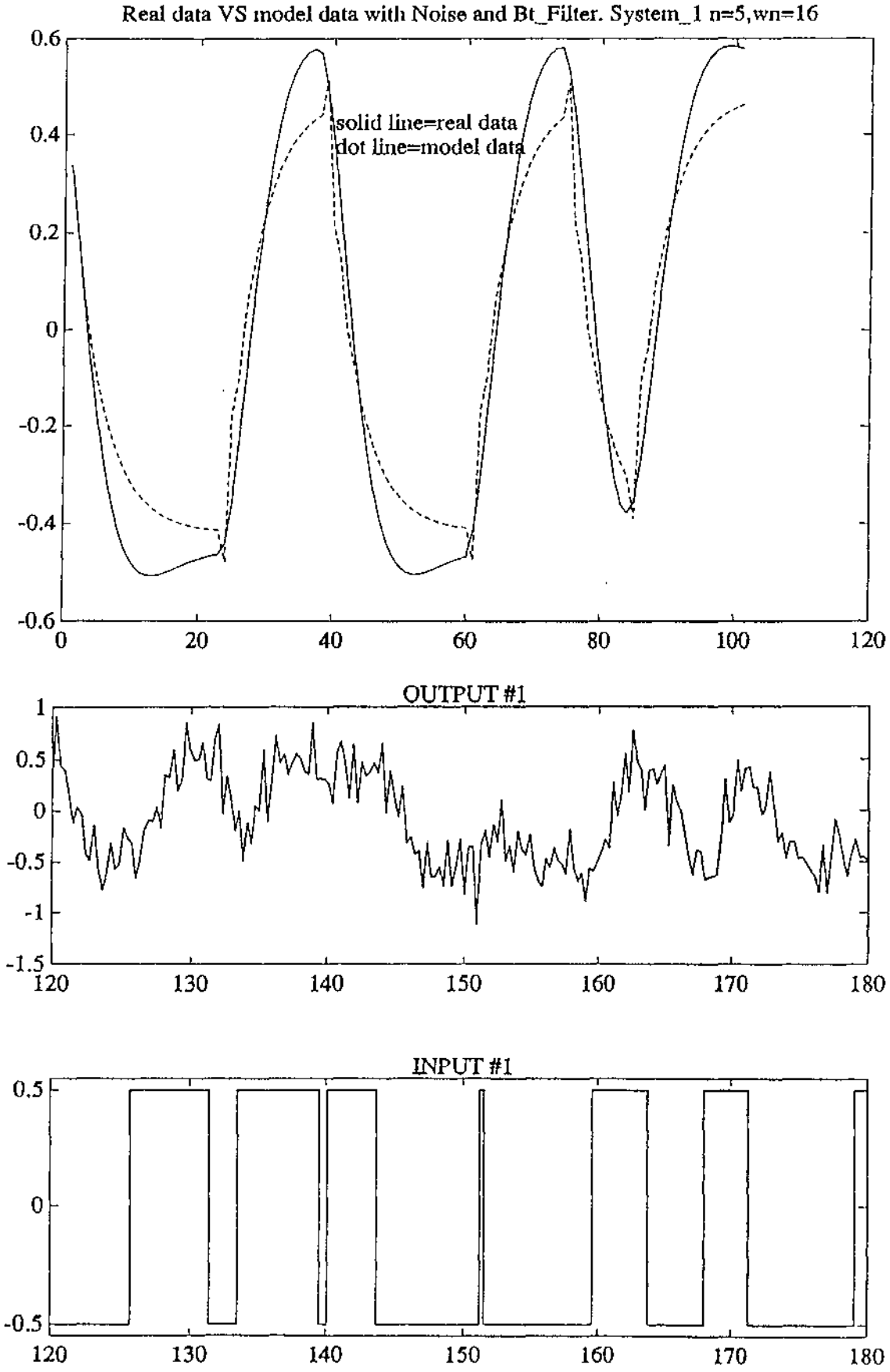
Input(PRBS) & Output with Noise and Bt-Filter. System_1 n=2,wn=5



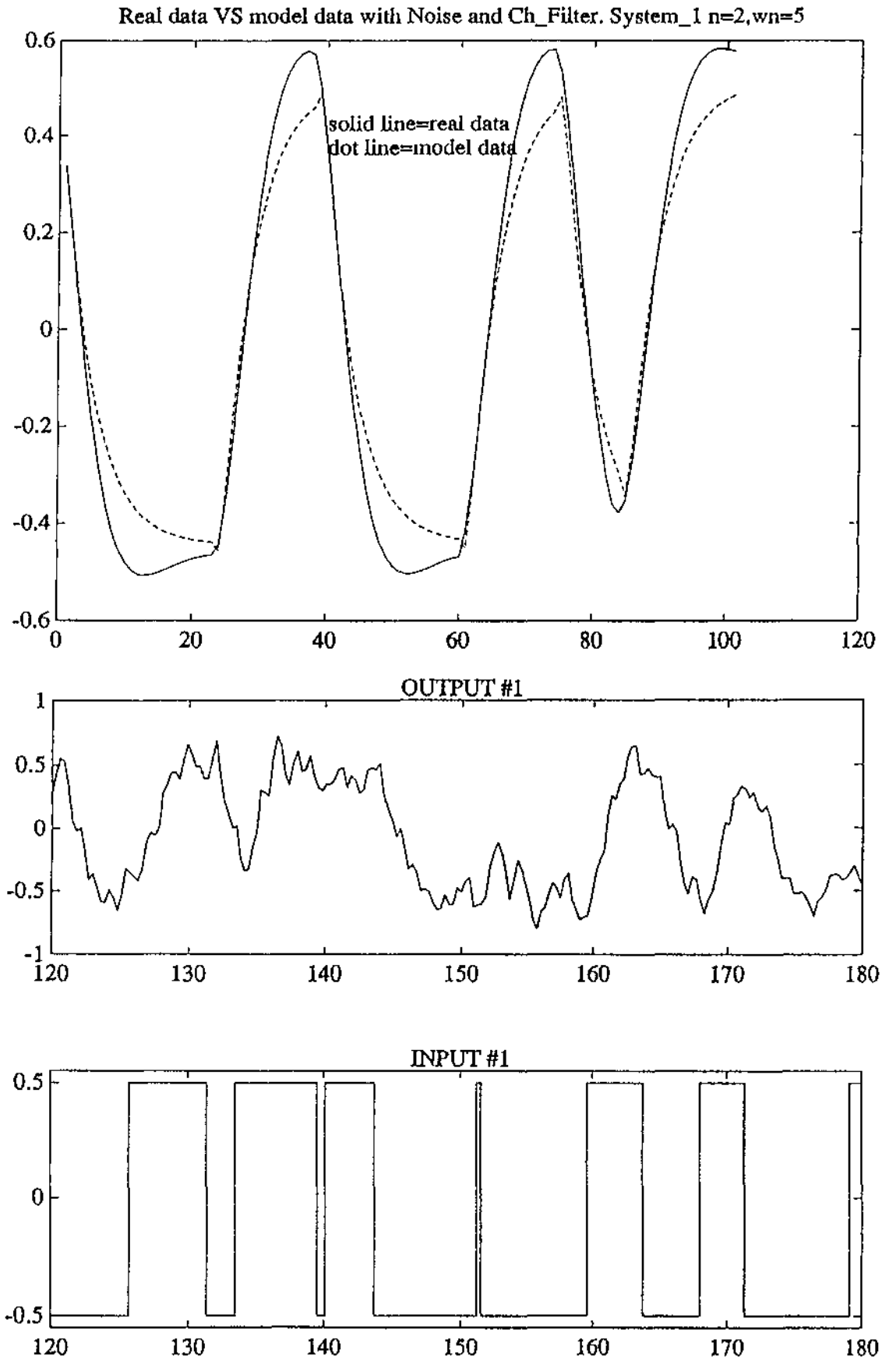
Input(PRBS) & Output with Noise and Bt-Filter. System_1 n=3,wn=5



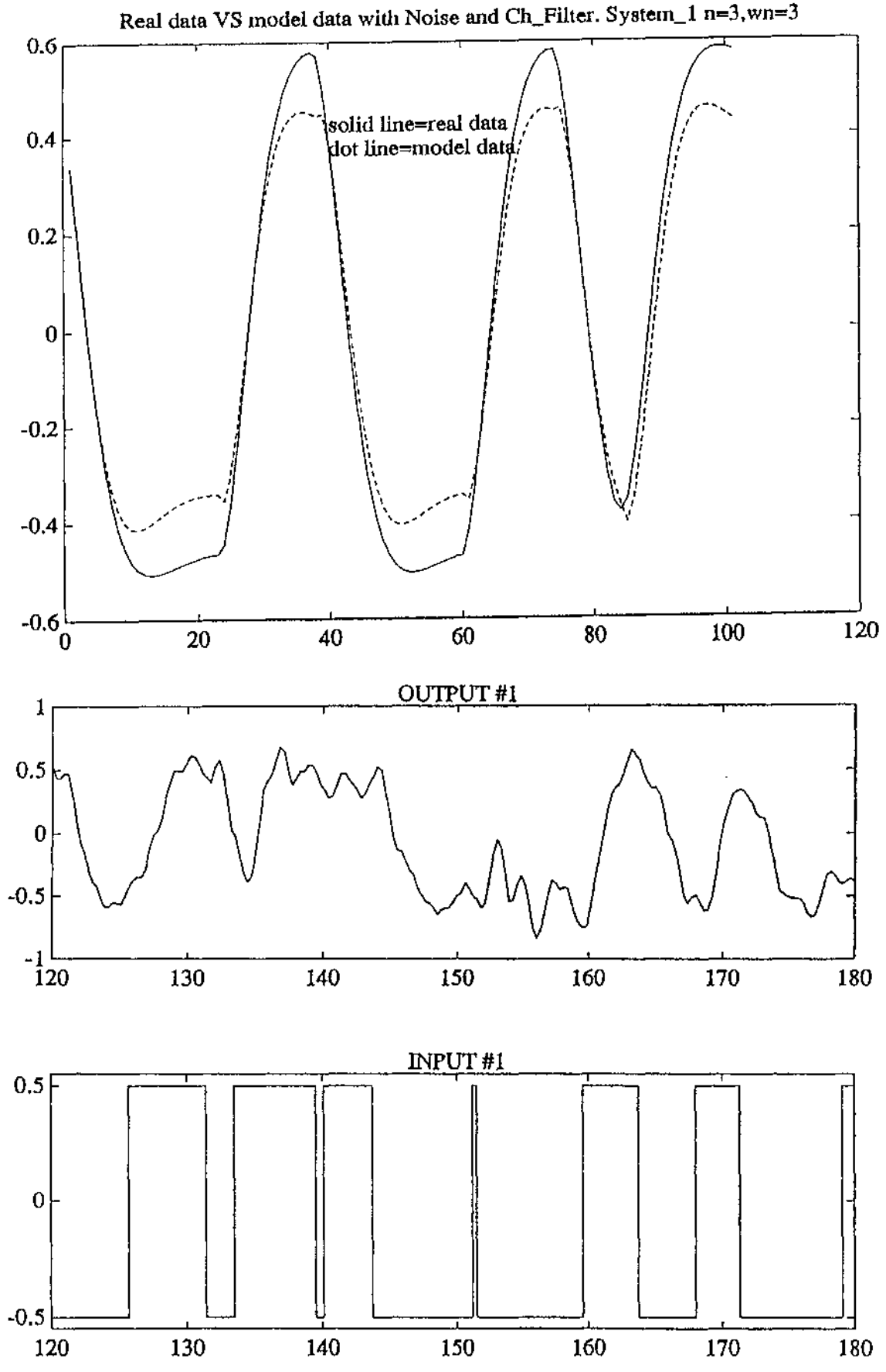
Input(PRBS) & Output with Noise and Bt-Filter. System_1 n=4,wn=5



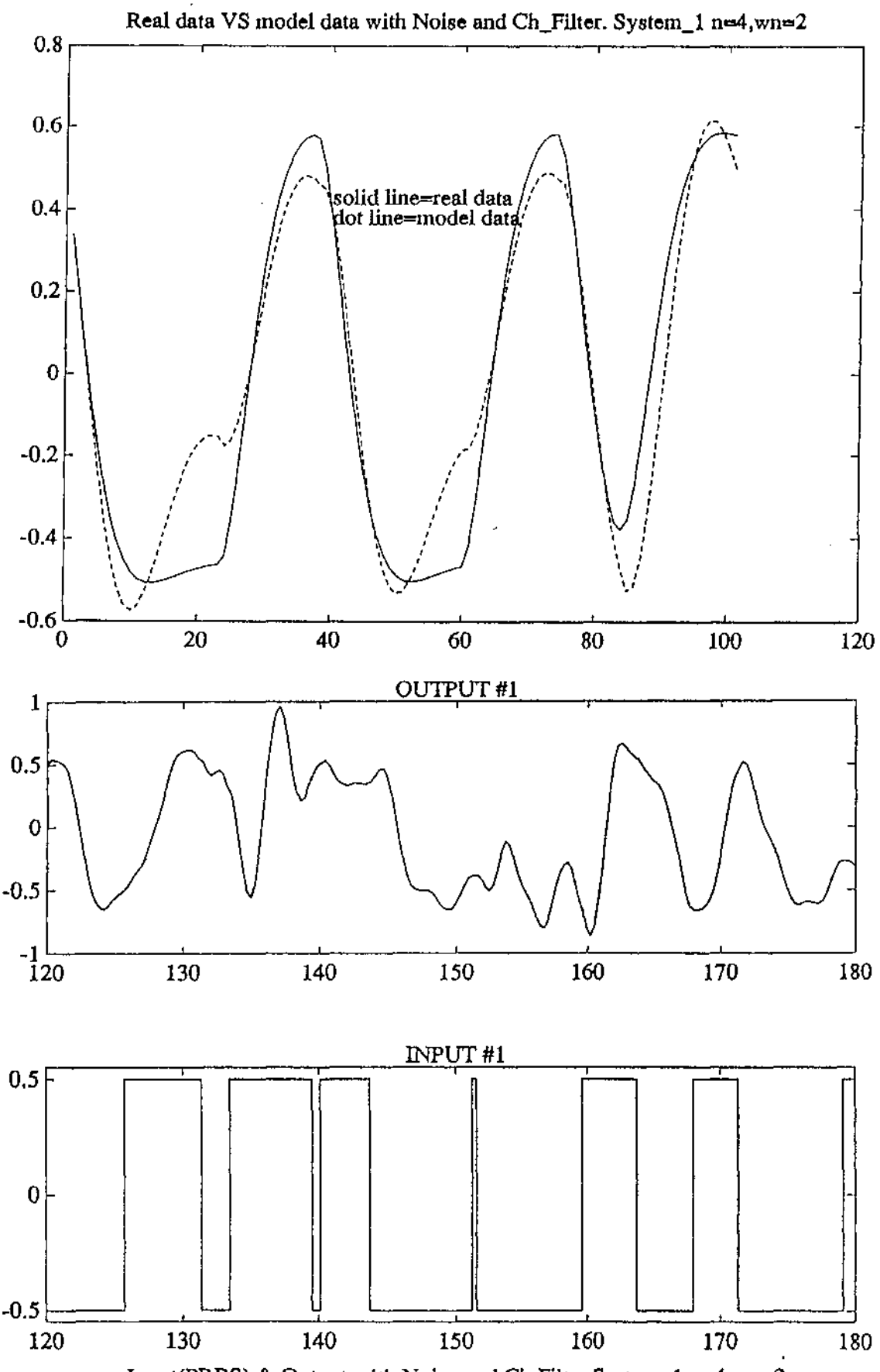
Input(PRBS) & Output with Noise and Bt-Filter. System_1 n=5,wn=16



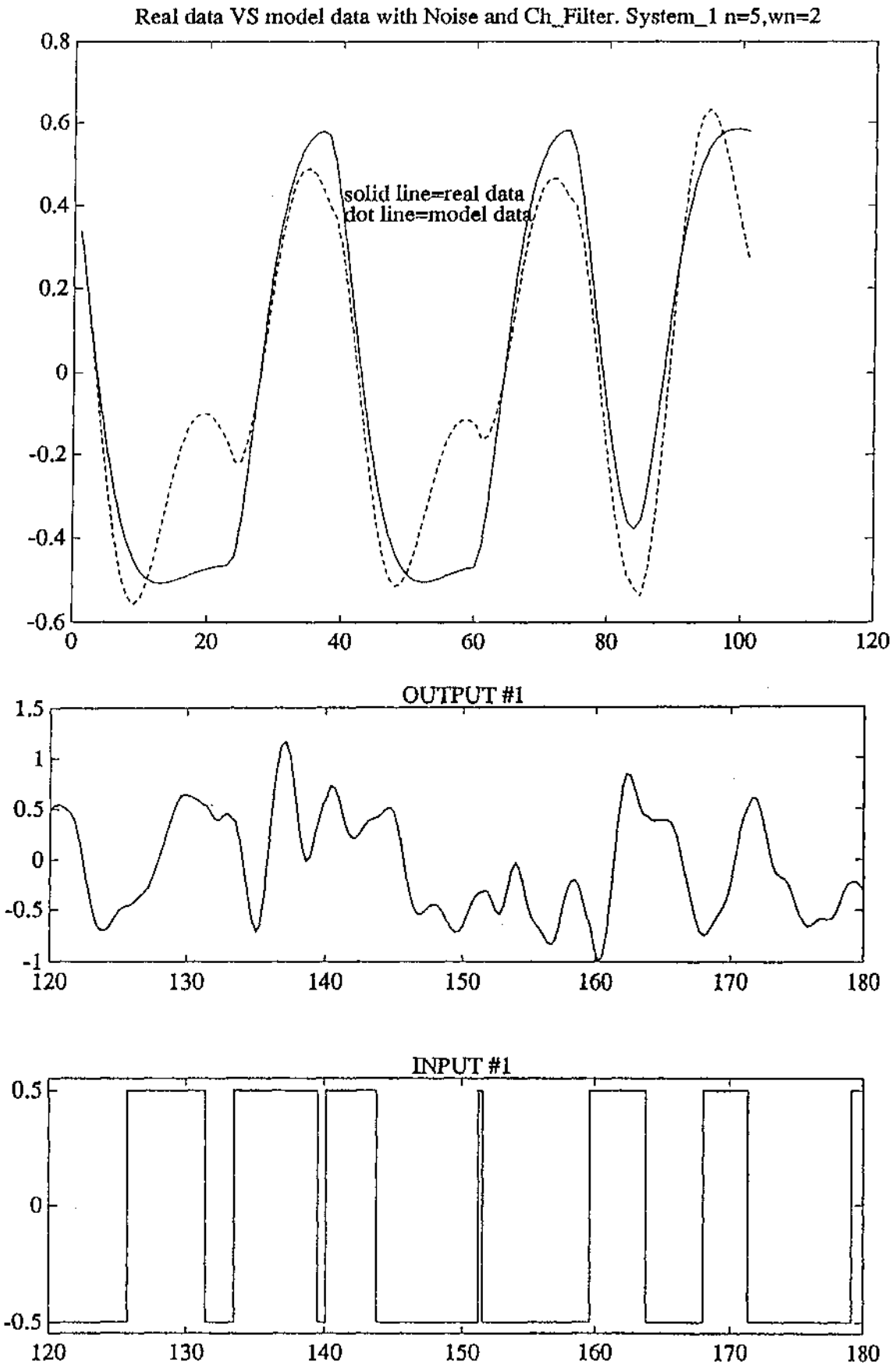
Input(PRBS) & Output with Noise and Ch-Filter. System_1 n=2,wn=5



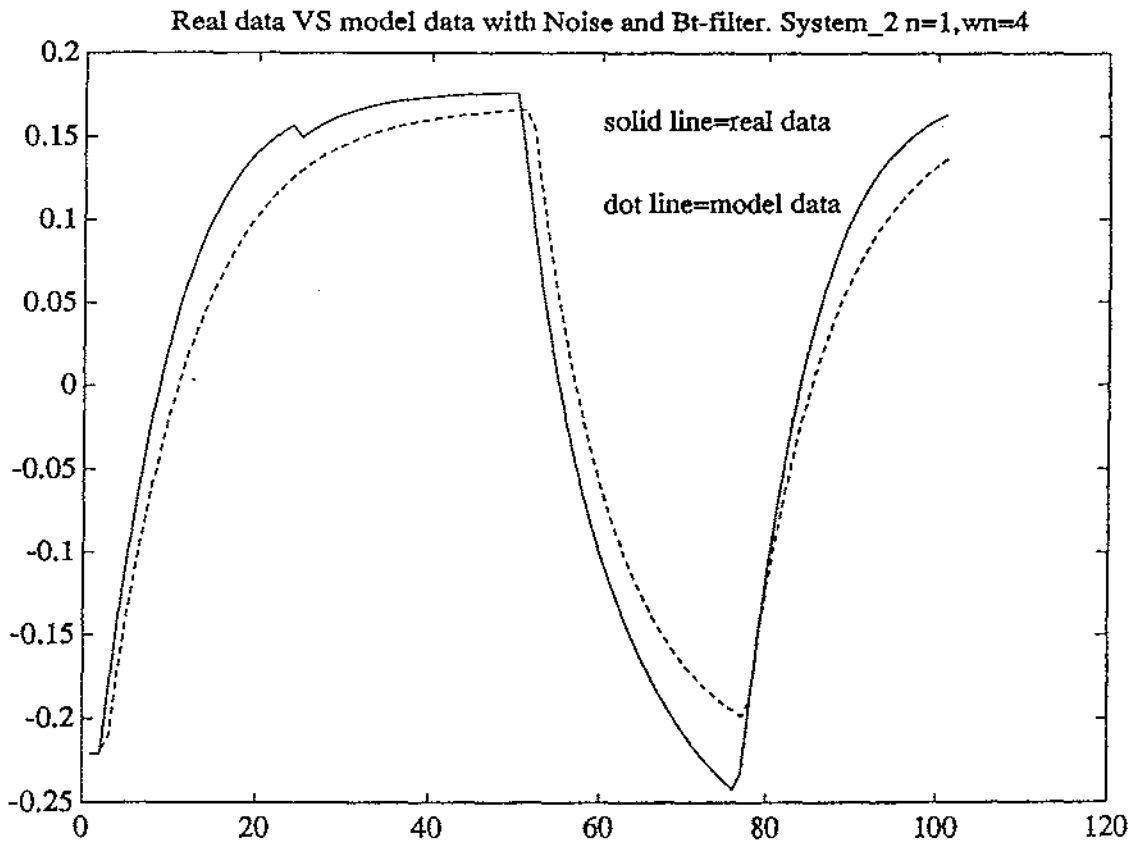
Input(PRBS) & Output with Noise and Ch-Filter. System_1 n=3,wn=3

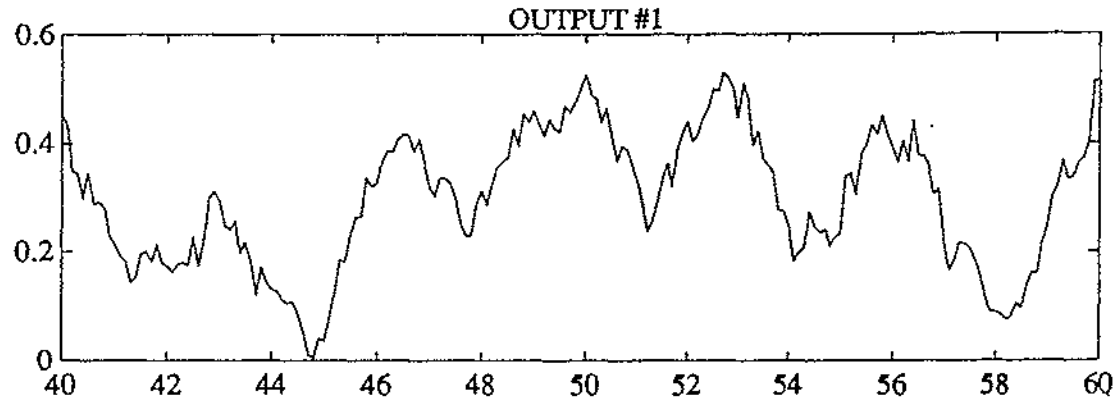


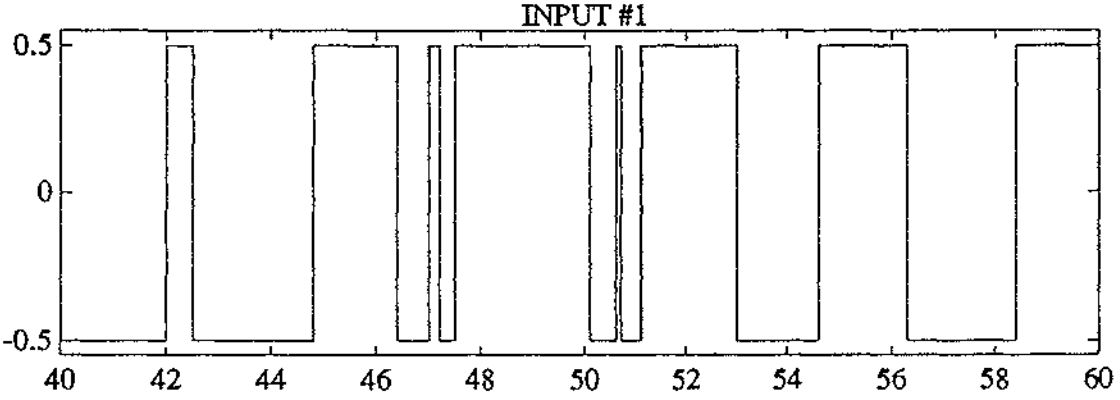
Input(PRBS) & Output with Noise and Ch-Filter.System_1 n=4,wn=2



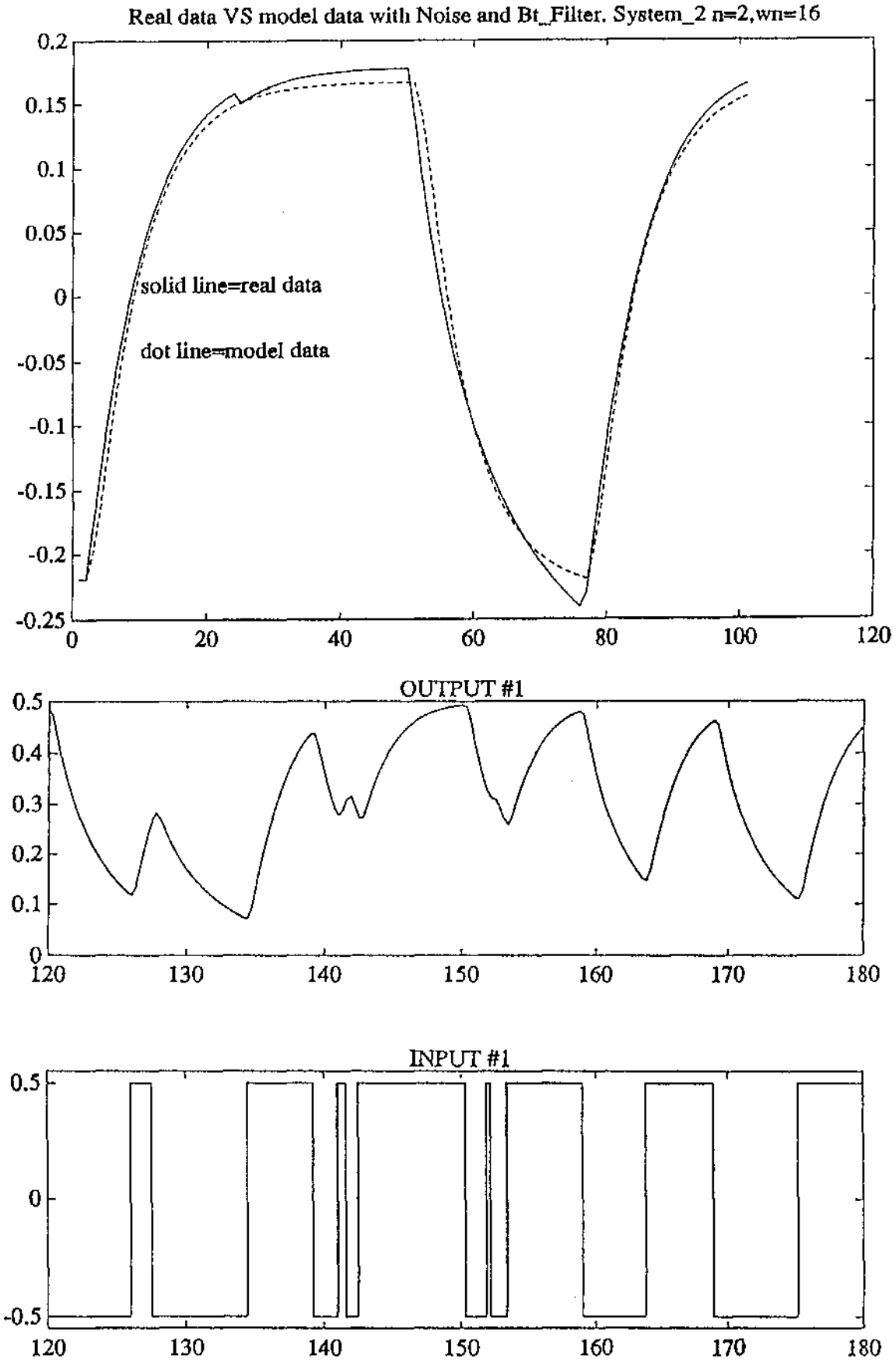
Input(PRBS) & Output with Noise and Ch-Filter.System_1 n=5,wn=2



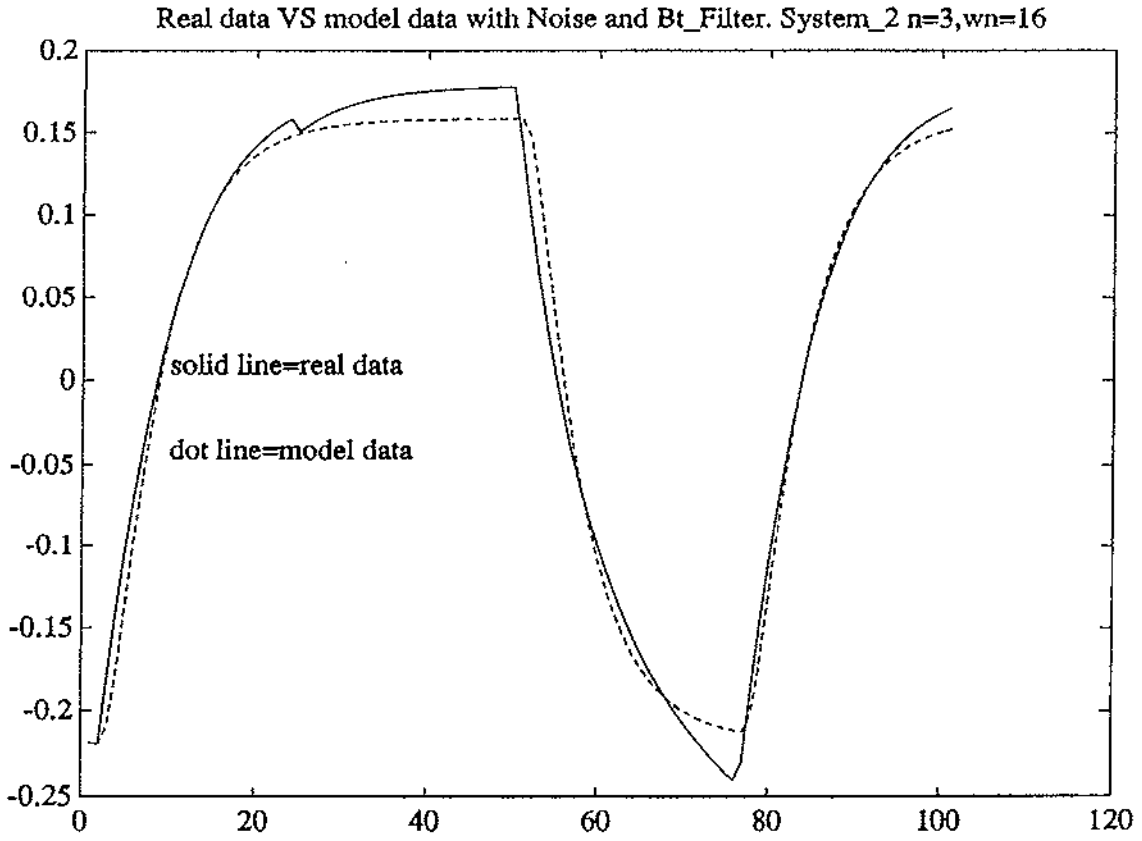


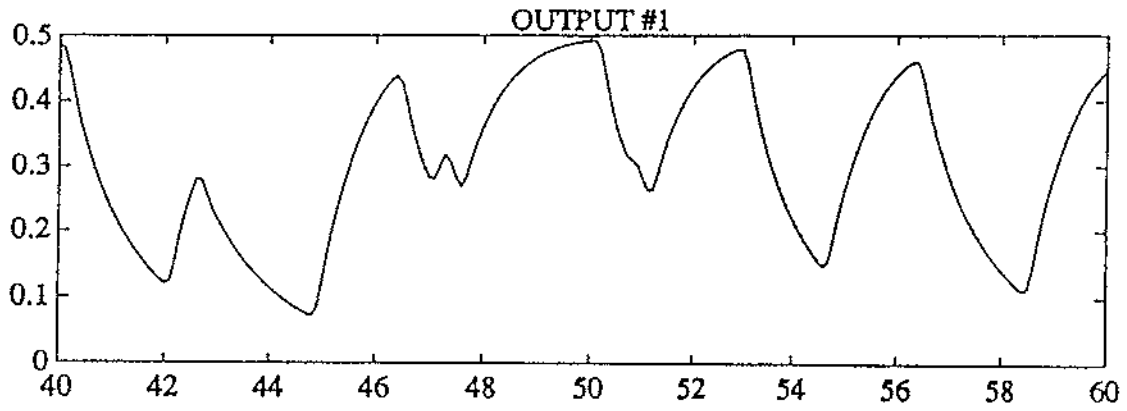


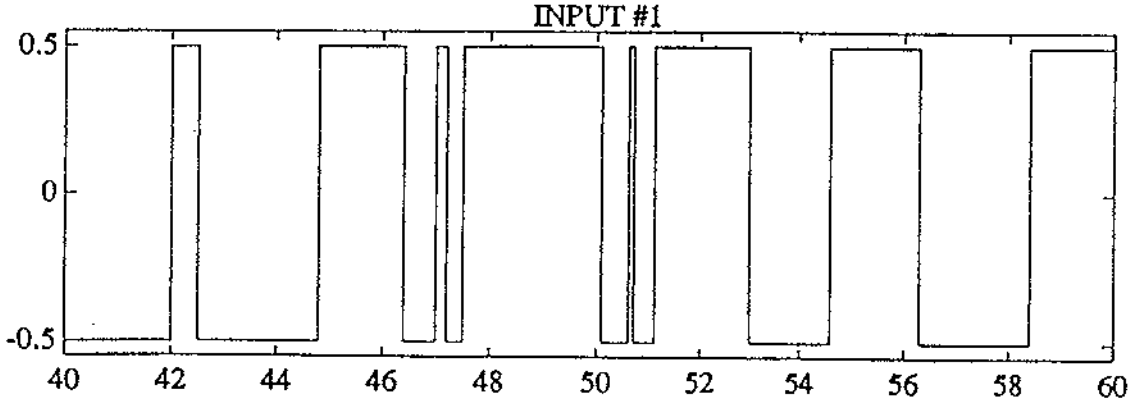
Input(PRBS) & Output with Noise and Bt-filter. System_2 n=1,wn=4



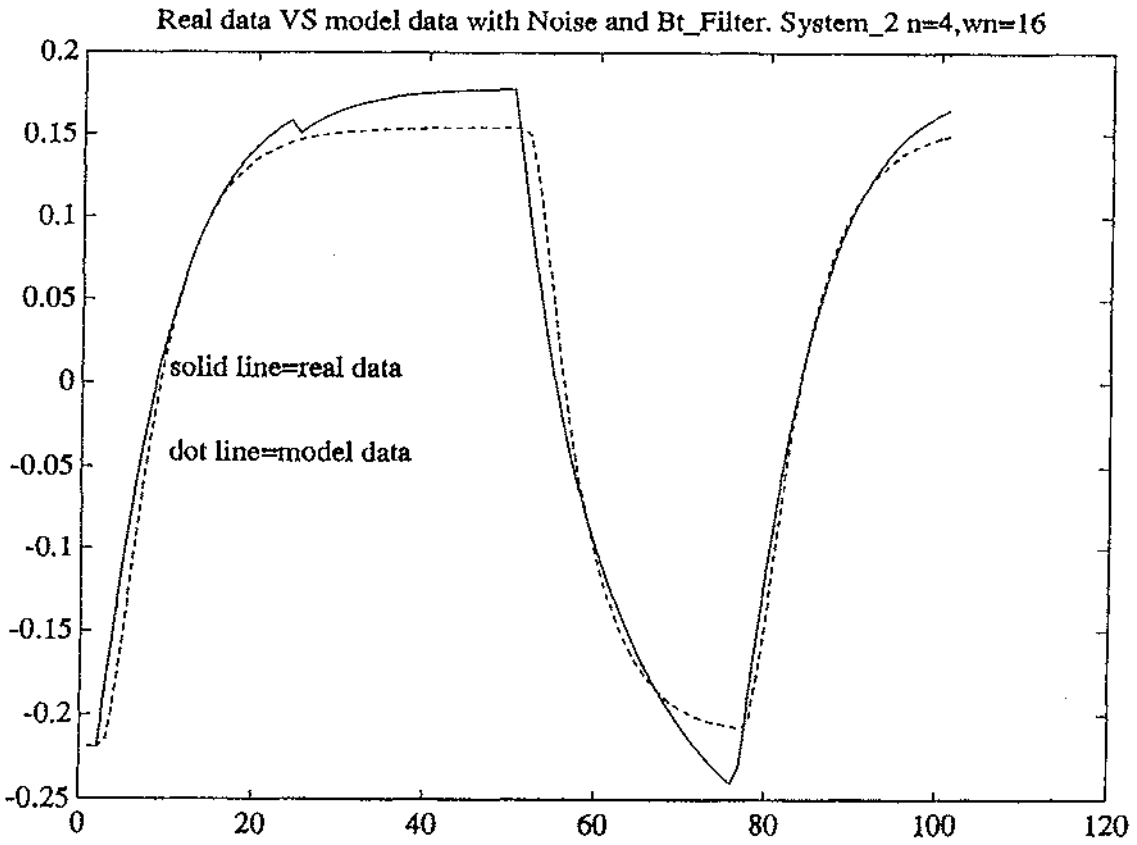
Input(PRBS) & Output with Noise and Bt-Filter.System_2 n=2,wn=16

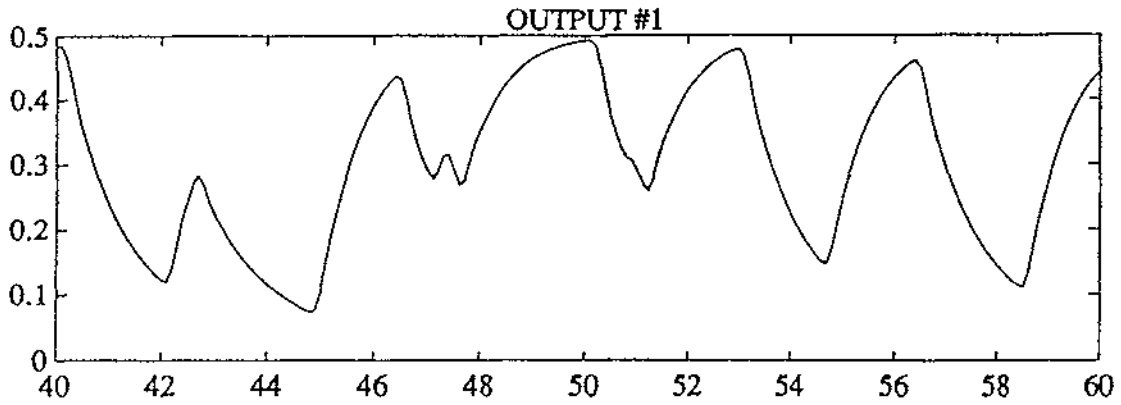


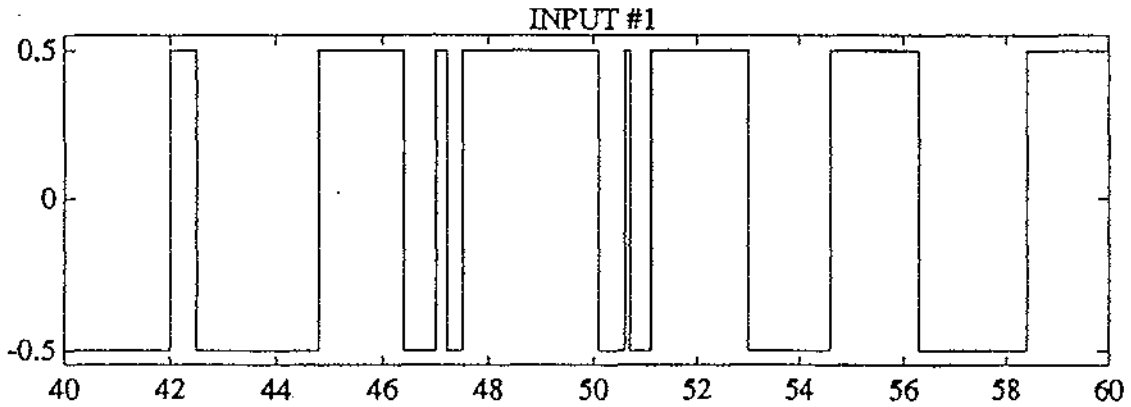




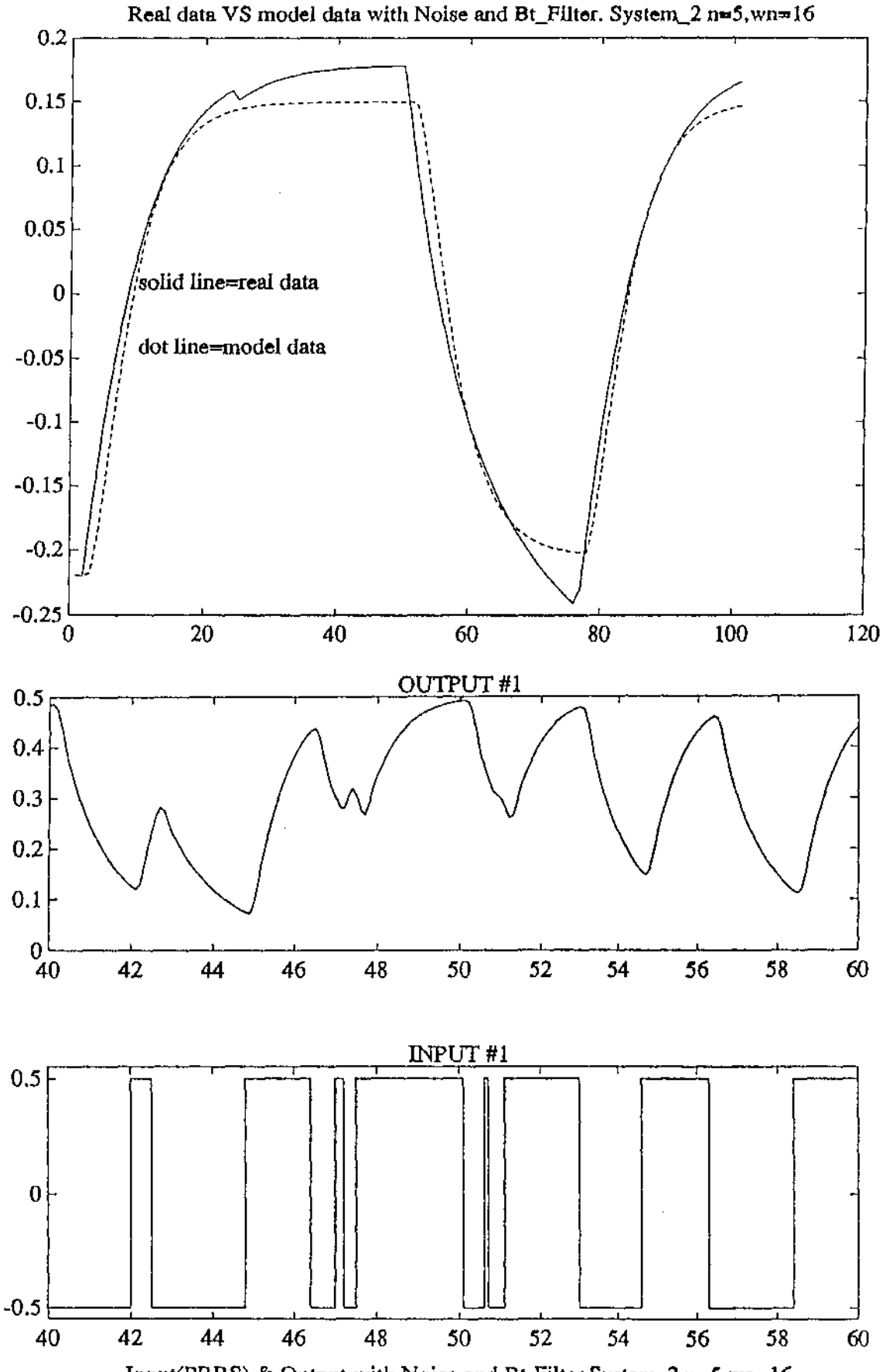
Input(PRBS) & Output with Noise and Bt-Filter.System_2 n=3,wn=16



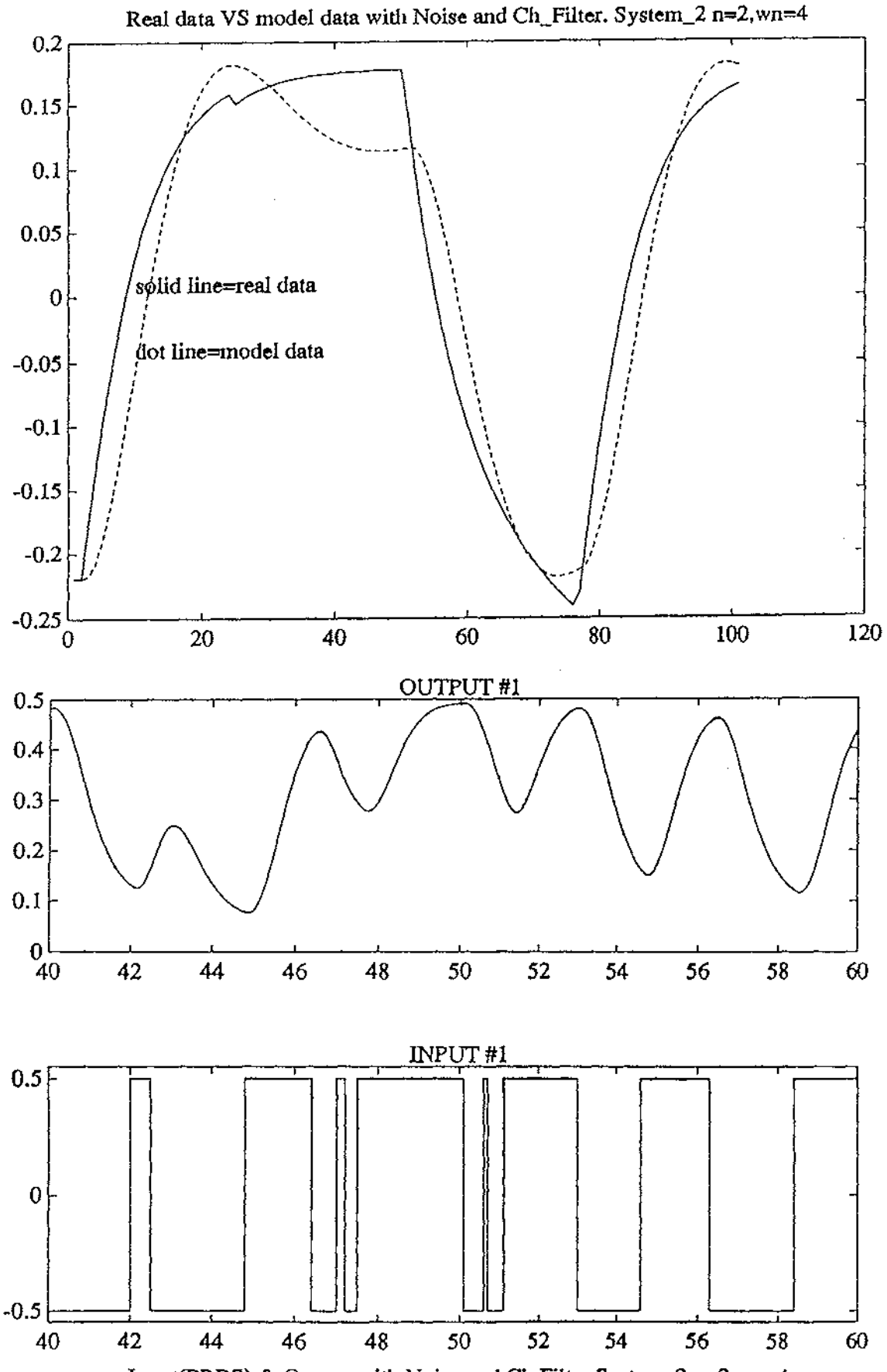




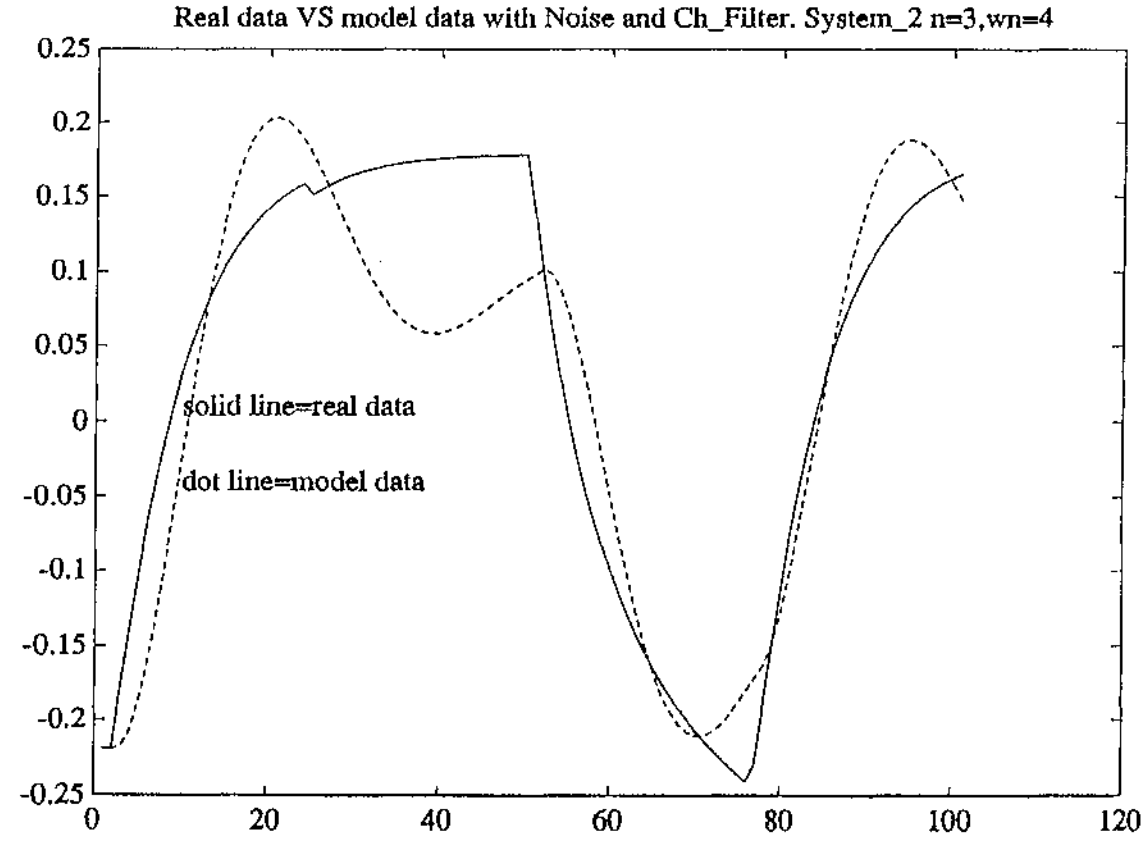
Input(PRBS) & Output with Noise and Bt-Filter.System_2 n=4,wn=16

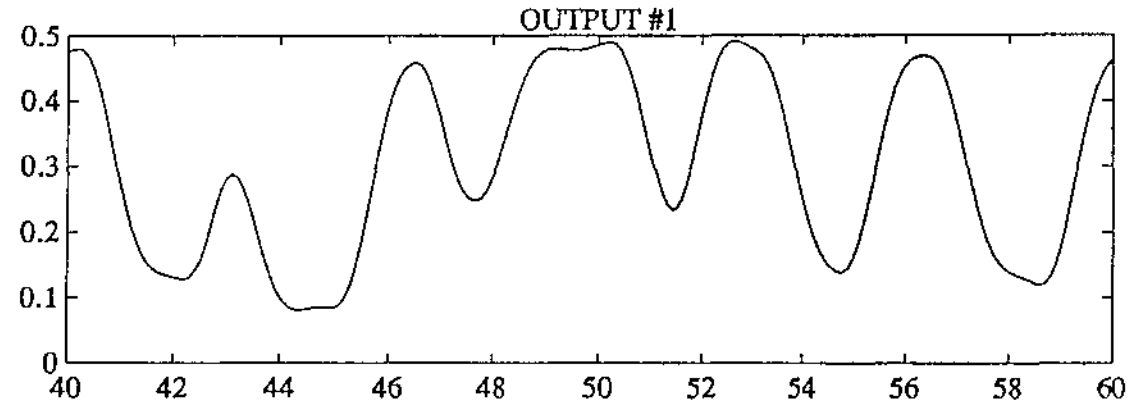


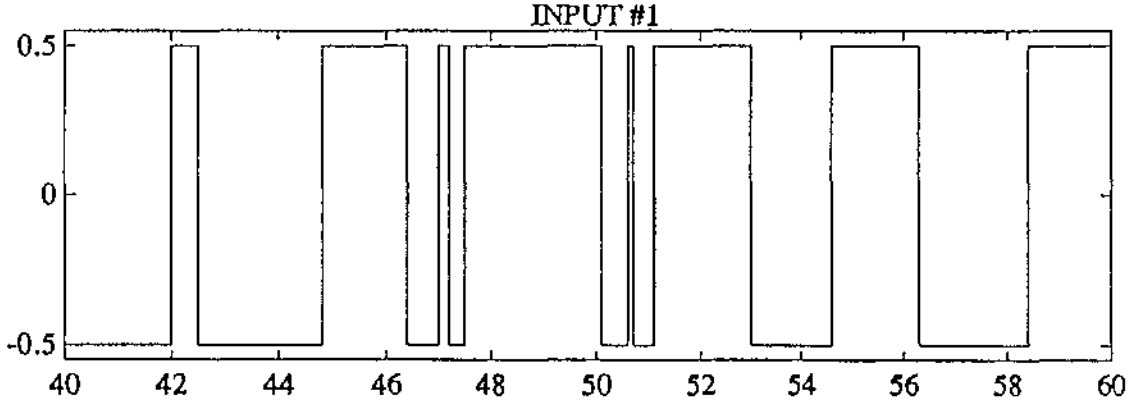
Input(PRBS) & Output with Noise and Bt-Filter.System_2 n=5,wn=16



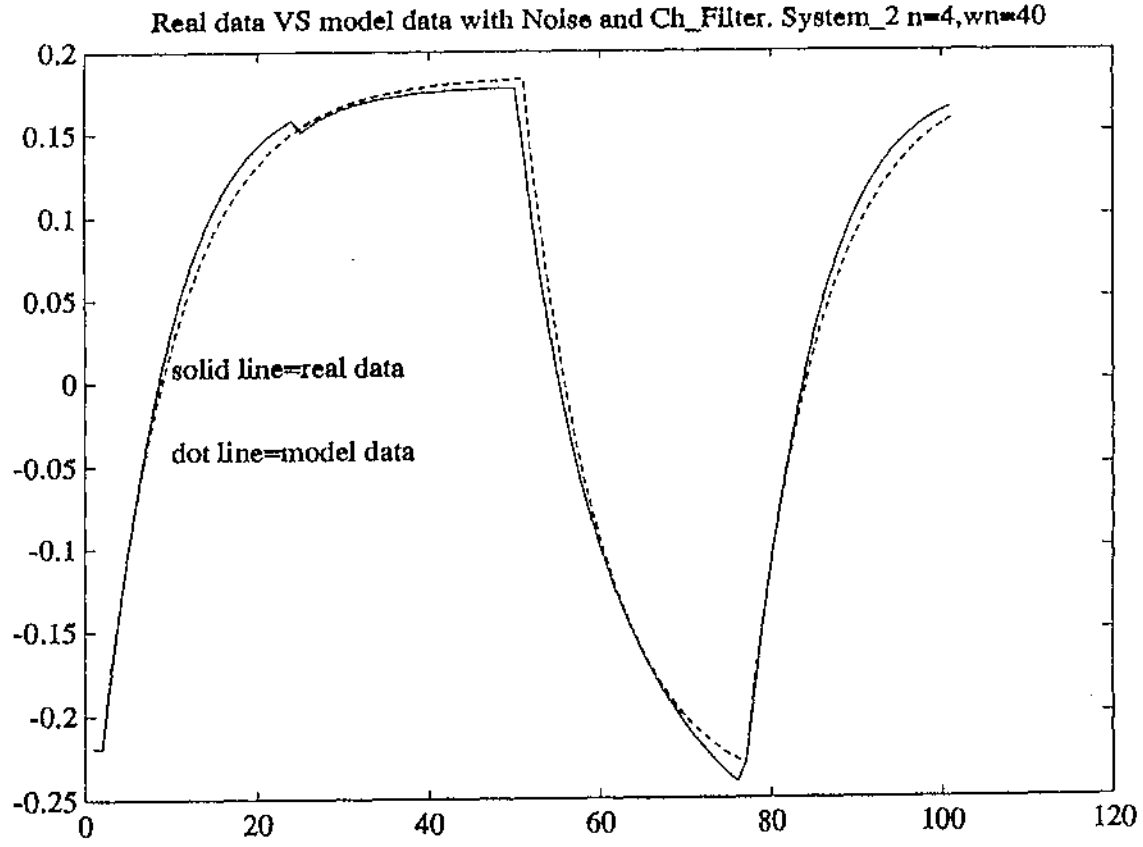
Input(PRBS) & Output with Noise and Ch-Filter.System_2 n=2,wn=4

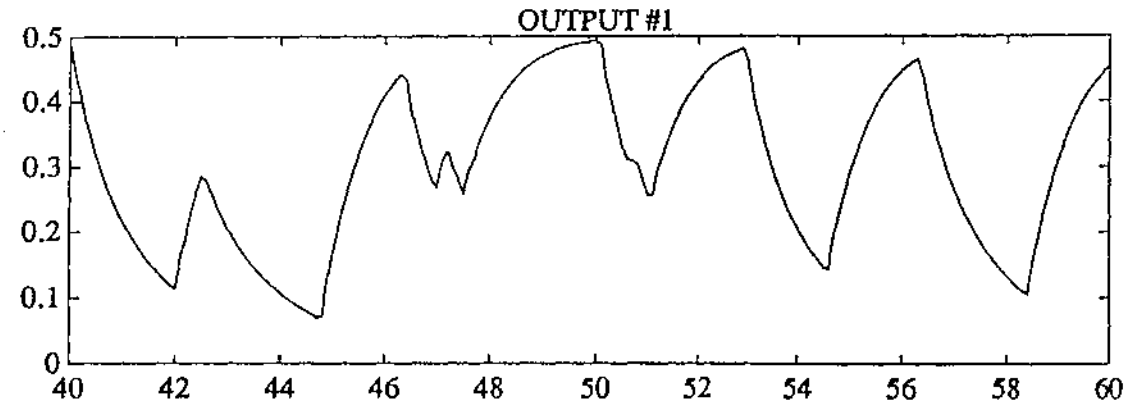


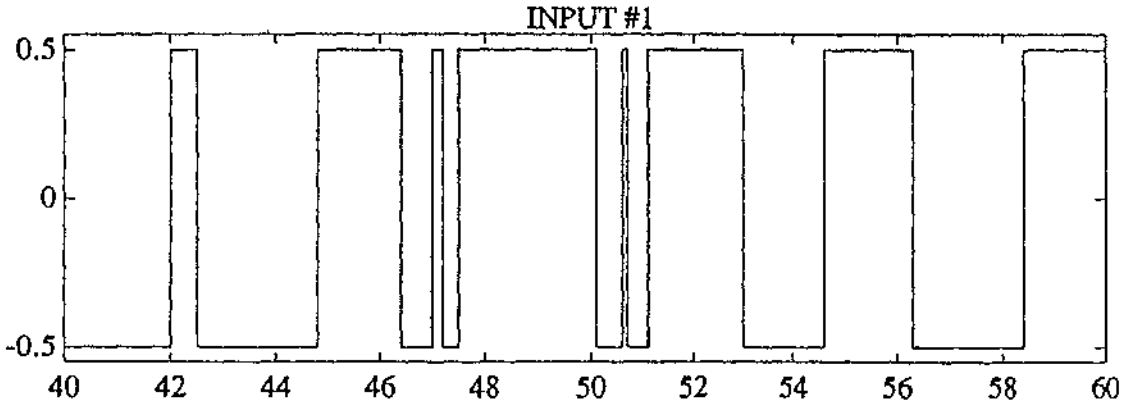




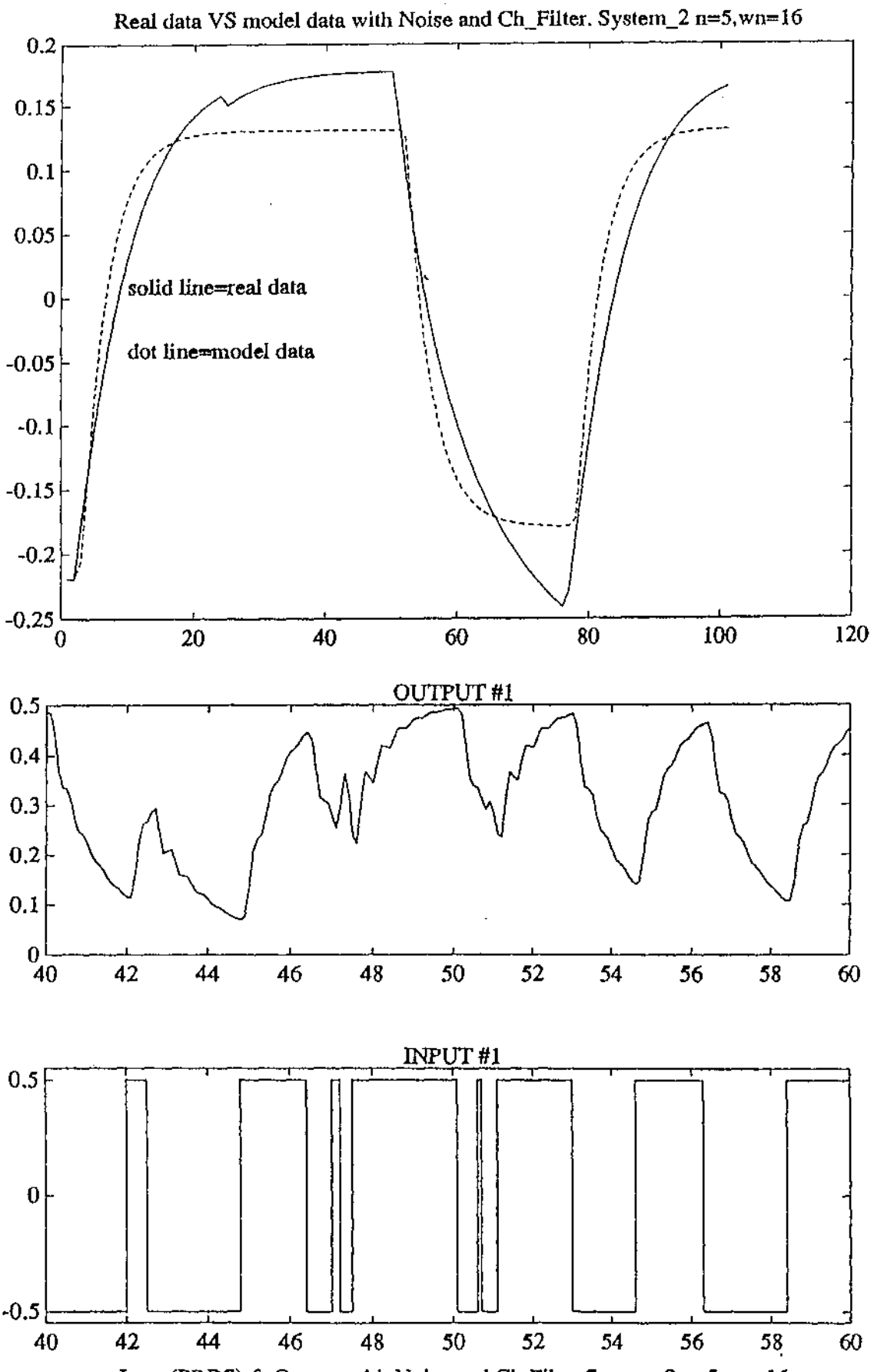
Input(PRBS) & Output with Noise and Ch-Filter.System_2 n=3,wn=4



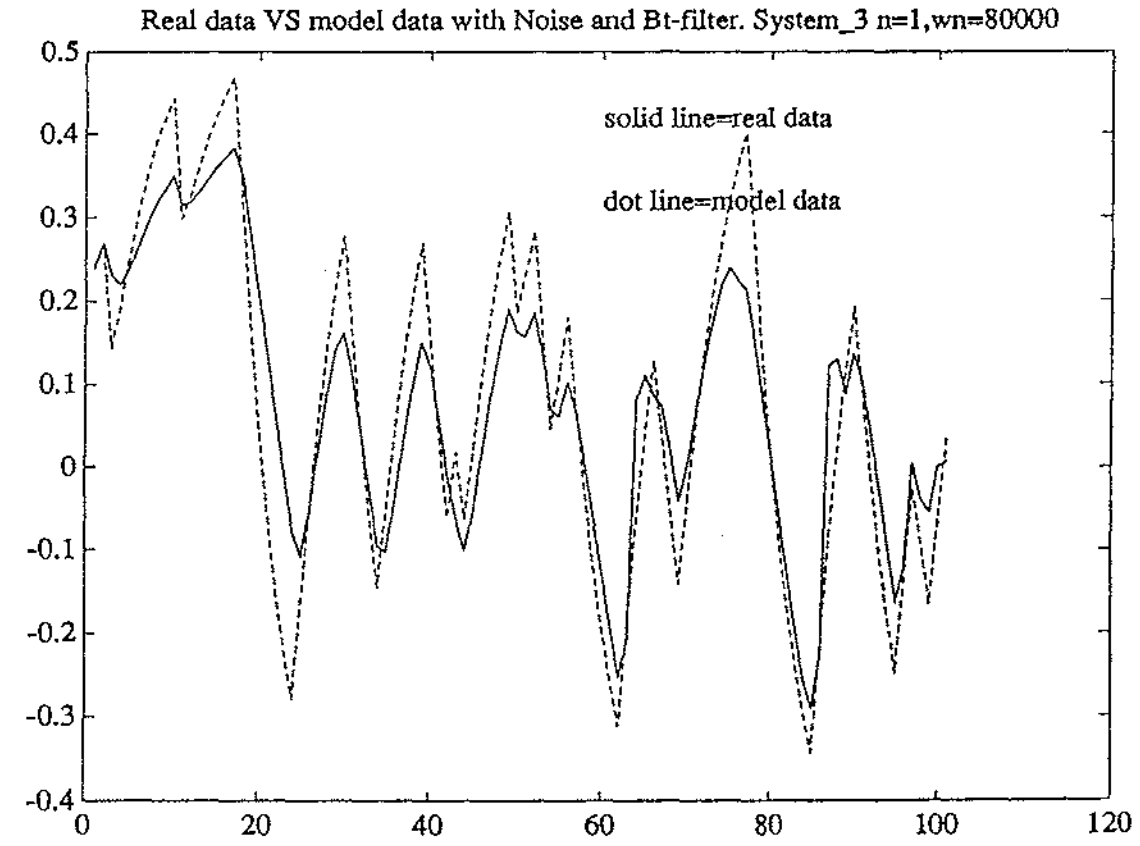


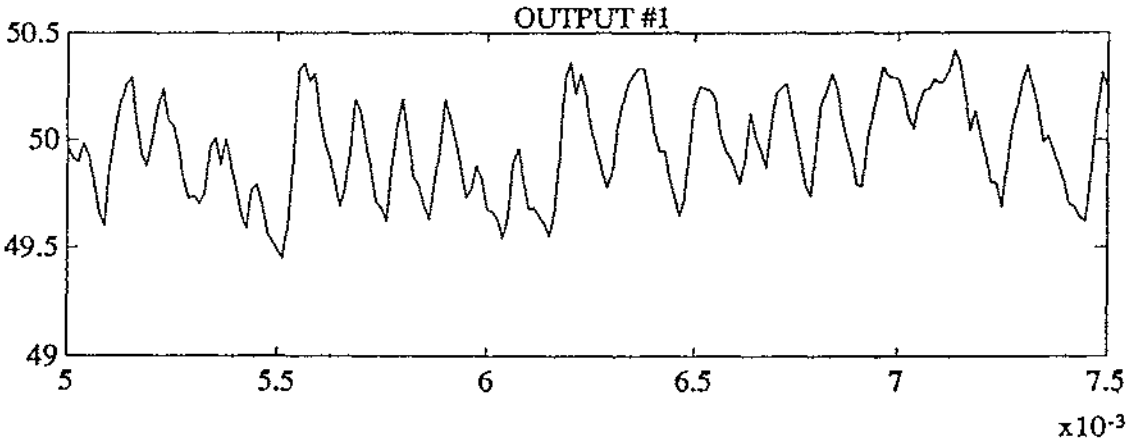


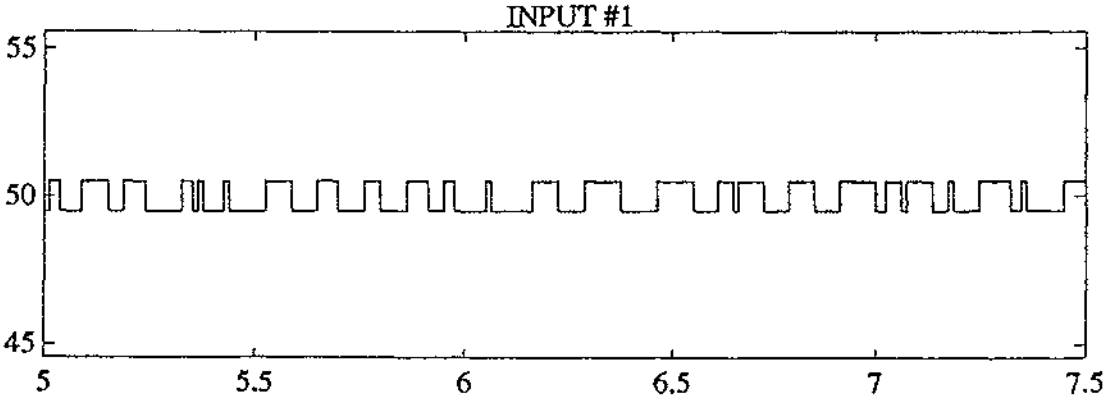
Input(PRBS) & Output with Noise and Ch-Filter.System_2 n=4,wn=40



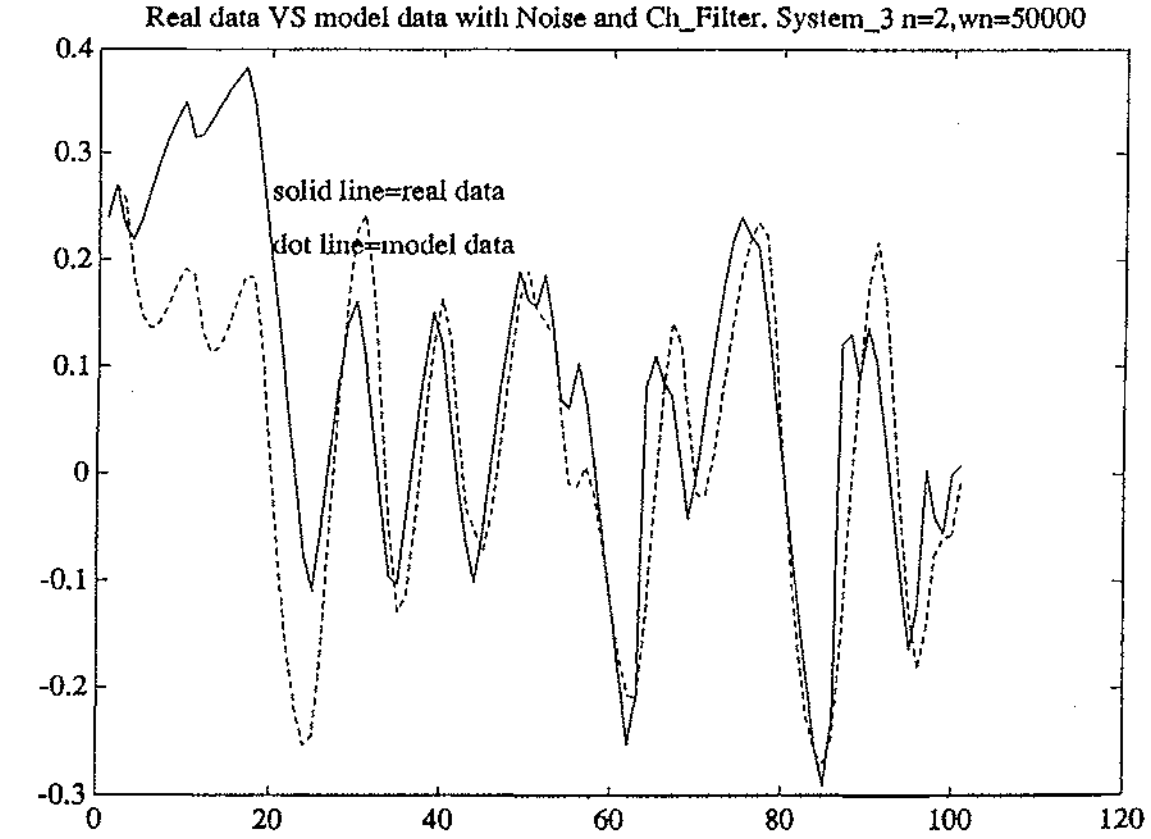
Input(PRBS) & Output with Noise and Ch-Filter.System_2 n=5,wn=16

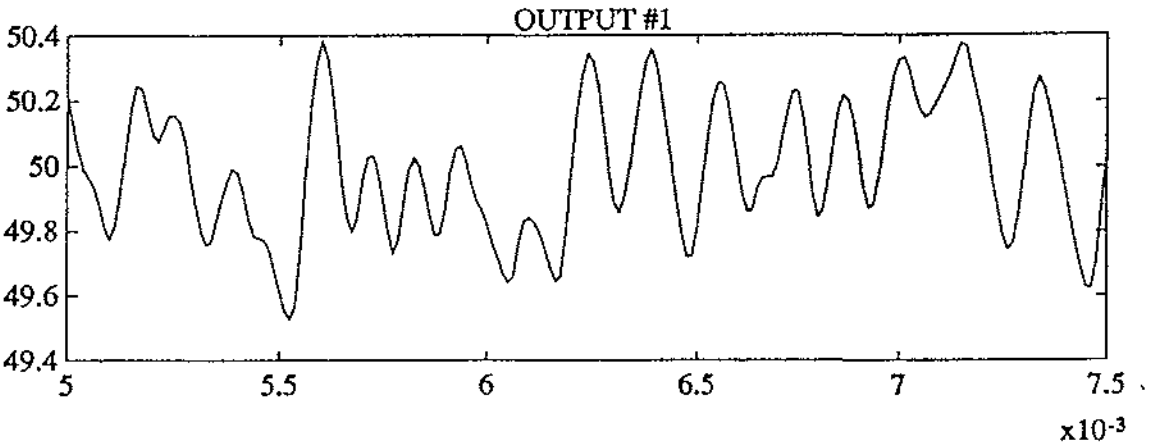


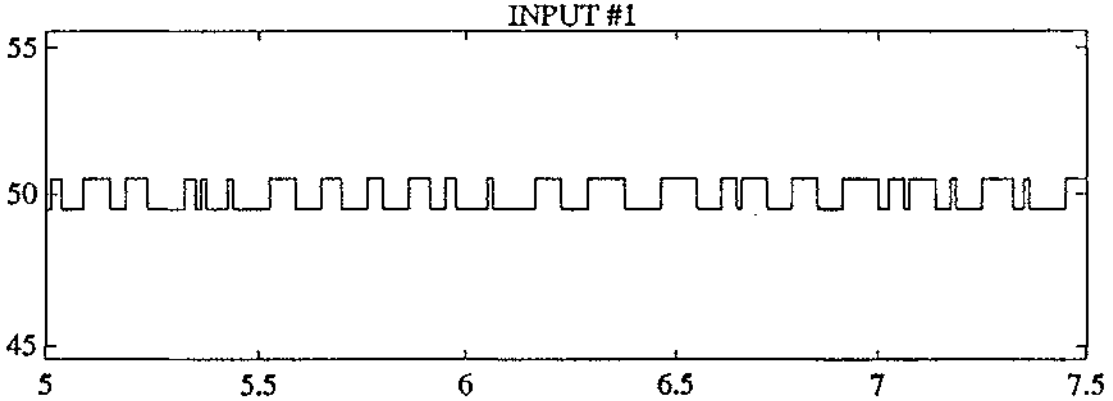




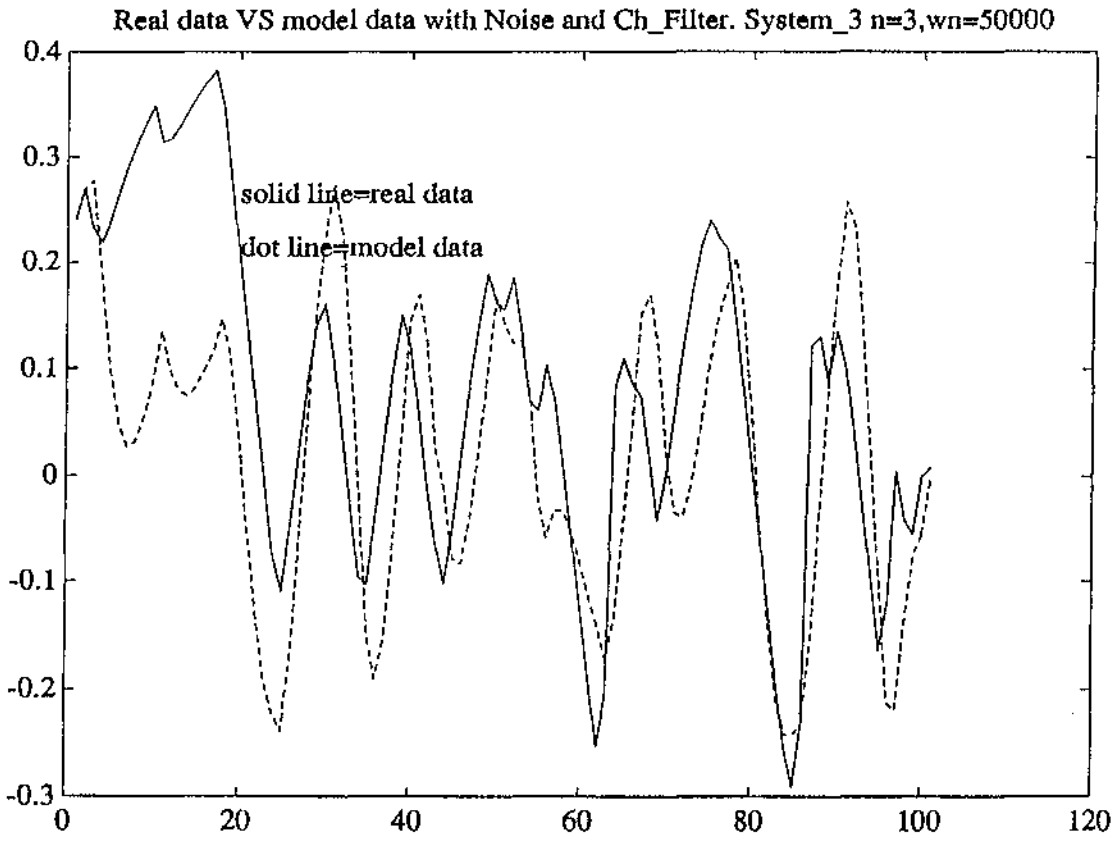
Input(PRBS) & Output with Noise and Bt-filter. System_3 n=1,wn=80000 x10^{-3}

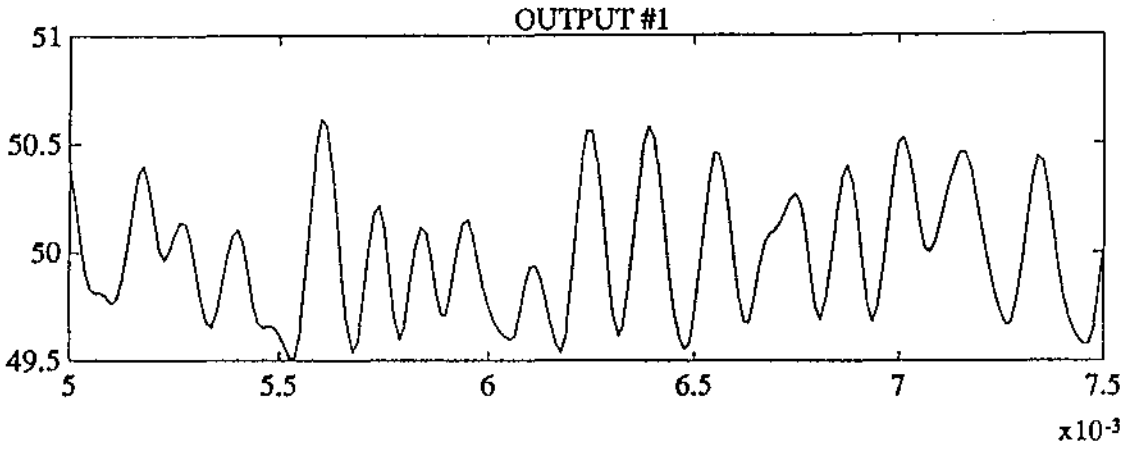


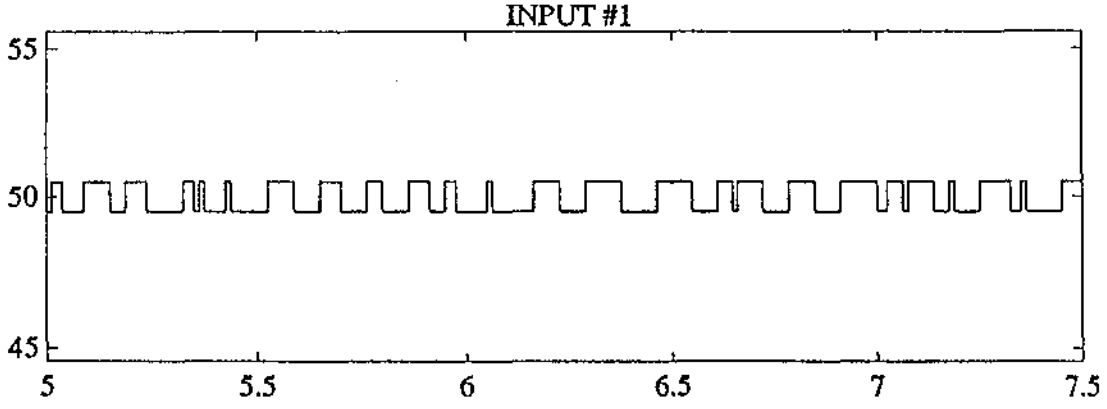




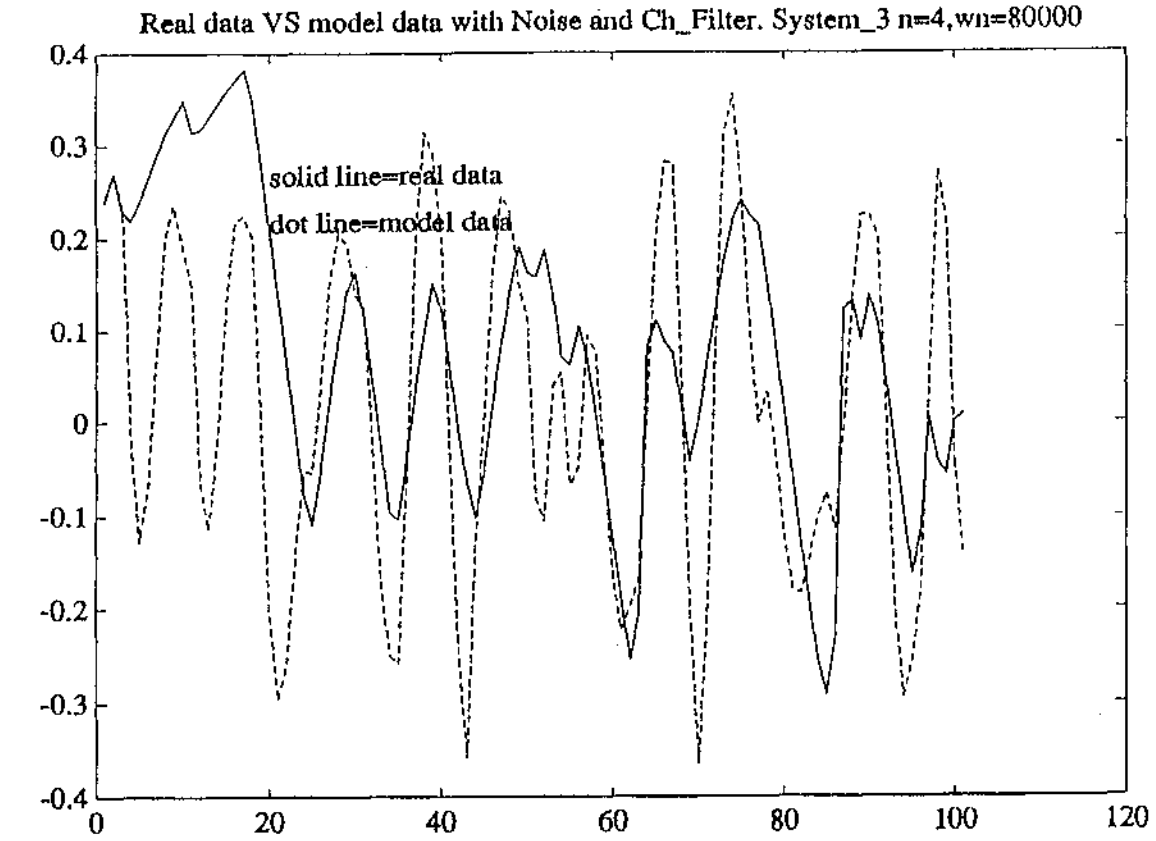
Input(PRBS) & Output with Noise and Ch-Filter.System_3 n=2,wn=50000 x10^{-3}

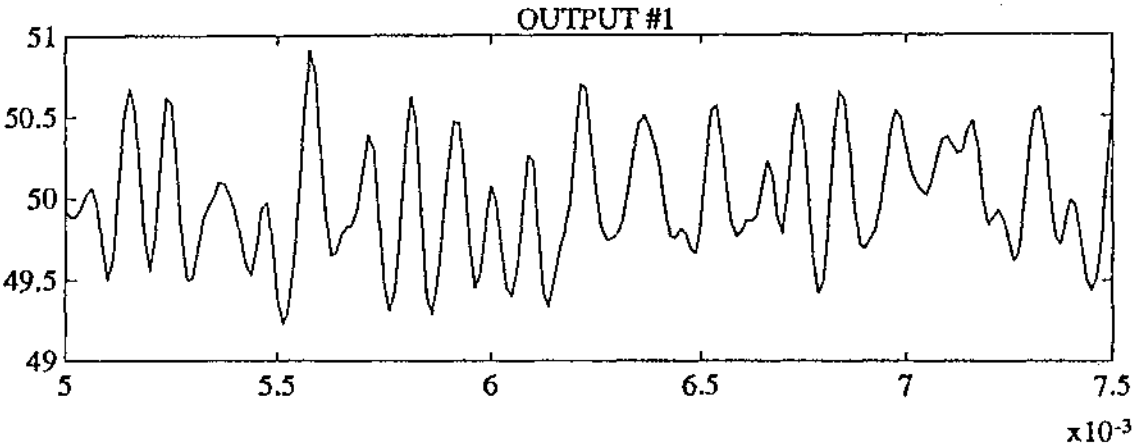


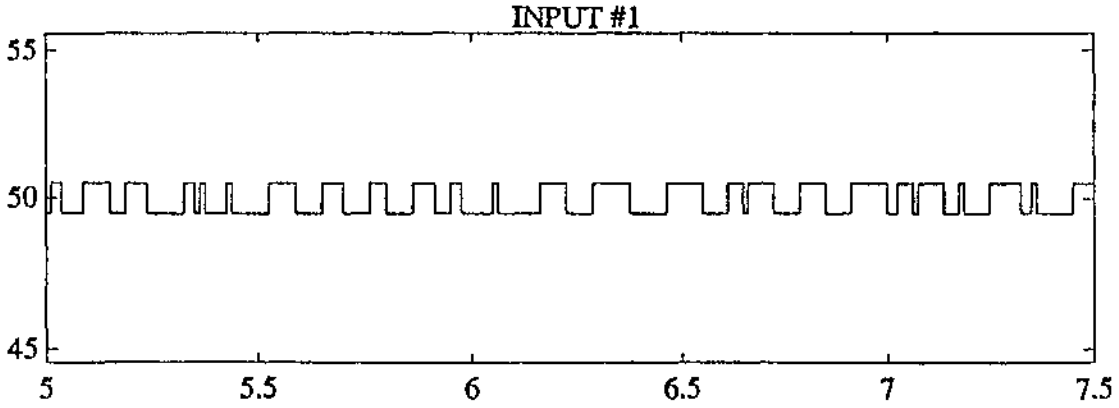




Input(PRBS) & Output with Noise and Ch-Filter.System_3 n=3,wn=50000 x10-3







Input(PRBS) & Output with Noise and Ch-Filter.System_3 n=4,wn=80000 x10-3