Identification of the Harmonic Transfer Functions of a Helicopter Rotor

by

Afreen Siddiqi

S.B., Massachusetts Institute of Technology (1999)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

Signature of Author ..... /.....
Department of Aeronautics and Astronautics
February, 2001

Certified by .....
Steven R. Hall
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Chair, Department Graduate Committee

Identification of the Harmonic Transfer Functions of a Helicopter Rotor

by

Afreen Siddiqi

Submitted to the Department of Aeronautics and Astronautics
on February, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Identification of the time periodic dynamics of a helicopter rotor system is necessary for effective controller design and analysis. The transfer properties of a linear time periodic (LTP) system can be described by harmonic transfer functions (HTF) that give the input-output relationship between the Fourier coefficients of the input signal and those of the output signal. A method for identifying harmonic transfer functions of linear time periodic systems is developed in this thesis. The system identification scheme employs a least square estimation technique to obtain HTF estimates. This least square estimation problem is underdetermined; therefore, an additional assumption, that the transfer function is smooth, is made in the ID scheme to achieve a well-posed problem. The estimates are calculated by applying a quadratic penalty to the curvature of the transfer functions. The identification scheme has been implemented in MATLAB, and partly coded in C programming language for maximum computational efficiency. This system ID method has been validated with analytical results for a few well-known LTP systems. The validation results show excellent agreement between the identified and analytical transfer functions.

Thesis Supervisor: Steven R. Hall
Title: Professor of Aeronautics and Astronautics

Acknowledgments

At this final stage of completion of my Master's program, I would like to profoundly thank my advisor, Professor Steven R. Hall, for his academic guidance and support. He deserves a lot of credit for enabling me to complete my studies in a timely manner with his advice and help, and for managing to maintain my research assistantship throughout the course of my S.M. degree. I also want to acknowledge Dr. Mauro J. Atalla for his painstaking efforts in sorting out many C-compiler and MATLAB problems. I really valued his cheerful and helpful guidance.

I owe a great deal of thanks to my wonderful parents, Fazlur Rahman and Tarannum Siddiqi, who gave me the inspiration, dedication and courage to pursue my dreams. Their love, affection, and advice truly allowed me to successfully complete my studies at MIT. My special thanks goes to my grandparents. Their prayers were with me in every step of my academic career. I wish to thank my uncle Rumi Moiz as well, whose passion for engineering and strong encouragement gave me the motivation to pursue a career in engineering for myself. I also want to express my gratitude for all the support that my dear husband Hamid, provided. His patience and optimism were always a great source of strength for me.

This research was performed at MIT as part of the Smart Materials and Structures Demonstrations Consortium, which includes the Boeing Company, MIT, the University of Maryland, and the University of California at Los Angeles, under Articles of Collaboration MDA972-98-3-0001. The Consortium was funded by the Defense Advanced Research Projects Agency (DARPA). Dr. Ephrahim Garcia is the DARPA program manager. Dr. Friedrich Straub of Boeing Mesa is the technical monitor for the MIT effort.

Contents

List of Figures

Notation

A	Dynamics matrix in state space formulation
A_c	Amplitude of chirp signal
$\mathbf{A}$	Coefficient matrix formed from a system of linear equations
$\mathbf{A_s}$	Sheared $\mathbf{A}$ matrix (when $\mathbf{A}$ is band diagonal)
$\mathcal{A}$	Dynamics matrix in harmonic state-space formulation
B	Input matrix in state-space formulation
$\mathcal{B}$	Input matrix in harmonic state-space formulation
C	Output matrix (related to state vector) of state-space formulation
$\mathcal{C}$	Output matrix (related to harmonic state vector) of harmonic state-space formulation
D	Input matrix (related to output) of state-space formulation
$\mathcal{D}$	Input matrix (related to output) of harmonic state-space formulation
$\mathbf{D^2}$	Second difference operator matrix
E_b	Young's modulus
E	Expected value function
$G(s)$	Transfer function in Laplace domain
$\widehat{G}(\omega)$	Empirical transfer function estimate
$\widehat{\mathbf{G}}(\omega)$	Harmonic transfer function matrix
$\mathcal{G}$	Harmonic transfer function
I	Identity matrix
I_b	Bending moment of inertia
J	Cost function
K	Controller transfer function
N	Number of chirps in an input signal
N_b	Number of blades in a helicopter rotor
N_h	Number of significant transfer functions of an LTP system
$\mathcal{N}$	Modulation frequency (block diagonal) matrix
R_{uy}	Cross-correlation function between $u(t)$ and $y(t)$
R_{uu}	Autocorrelation function of $u(t)$
T	System time period
T_c	Chirp period
T_d	Delay time
T_r	Rotation period
T_s	Time span of a truncated continuous signal
$\mathbf{T}$	Control response matrix used in HHC

U	Input matrix with elements in frequency domain
$\mathbf{U}$	Matrix with modulated and Fourier transformed input data vector elements
$\mathbf{U_c}$	Chirp input vector for rotor blade
$\mathbf{V}$	Vector for rotor blade input signals
$\mathcal{X}$	Harmonic states vector
Y	Output matrix with elements in frequency domain
$\mathbf{Y}$	Fourier transformed vector of $\mathbf{y}$
a_{ij}	Element of ith row and jth column of $\mathbf{A}$ matrix
b	Vector of forcing terms in a non-homogenous linear system of equations
d_1	Number of sub-diagonals
d_2	Number of super-diagonals
$e(t)$	Disturbance to a system
$\mathbf{e}$	Error vector
f	Forcing function in the Laplace equation
f_0	Initial frequency of chirp
f_1	Final frequency of chirp
f_{IN}	Frequency of input signal to a Phase Locked Loop
f_{VCO}	Frequency of output signal from a Voltage Controlled Oscillator
$g(t)$	Impulse response function of an LTI system
$\widehat{\mathbf{g}}$	Vectorized $\widehat{\mathbf{G}}$ matrix
$\mathbf{g}_m$	mth transfer function (in vector form)
g_i	ith element of $\widehat{\mathbf{g}}$
g_{m_i}	ith element of $\mathbf{g}_m$
h	Step size in discretizing mesh for the Laplace equation
$p(x)$	Continuous function of x
$q(x)$	Load per unit length of beam
n	Number of data points in input vector $\mathbf{u}$
n_c	Number of chirps generated by chirp generator
n_h	Number of harmonic transfer functions required to be evaluated
n_Φ	Product of n and n_h
t	Time (in seconds)
t_d	Chirp duration
t_p	Pause time between chirps in an input sequence
$\widehat{t}$	Pseudo time obtained by shifting time vector based on system phase

$u(t)$	Input signal in time domain
$\mathbf{u}$	Vector of discretized $u(t)$
x	State vector
$x_{ss}(t)$	Steady-state response
$y(t)$	Output signal in time domain
$y_m(t)$	Modulated output
$\mathbf{y}$	Vector of discretized $y(t)$
$\mathbf{z}$	Vibration amplitudes vector
$\mathbf{z}_0$	Vibration amplitude vector in absence of control inputs
α	Scalar for weighing difference operator matrix
β	Coefficient determining degree of periodic modulation
ζ	Damping ratio
$v(x)$	Deflection of beam in vertical direction as function of beam length
ψ	Rotor azimuth angle
$\boldsymbol{\psi}$	Vector of azimuth measurements
ω	Frequency in rad/sec
ω_n	Natural Frequency
ω_p	Pumping frequency
Δ	Lumped transfer functions
ϕ	State transition matrix
ϕ_c	Chirp phase
Φ	State transition matrix over interval $[0, T]$
Φ_{uu}	Power spectral density of $u(t)$
Φ_{uy}	Cross-spectral density of $u(t)$ and $y(t)$
$\mathbf{\Phi_{UU}}$	Product of $\mathbf{U}$ and conjugate transpose of $\mathbf{U}$
$\mathbf{\Phi}_{UU_i}$	Matrix in ith position of the third dimension of $\mathbf{\Phi_{UU}}$ array
$\mathbf{\Phi_{UUs}}$	Block diagonal matrix obtained from $\mathbf{\Phi_{UU}}$ array
$\mathbf{\Phi_{UY}}$	Product of conjugate transposed $\mathbf{U}$ matrix with vector $\mathbf{Y}$
$\mathbf{\Phi}_{UY_i}$	Matrix in ith position of the third dimension of $\mathbf{\Phi_{UY}}$ array
$\mathbf{\Phi_{uy}}$	Vectorized form of $\mathbf{\Phi_{UY}}$ array
$\mathbf{\Phi}_{uy_i}$	ith element of $\mathbf{\Phi_{uy}}$ vector
Ω	Rotor frequency

Superscripts

$()^T$	Matrix transpose
$()^*$	Complex conjugate
$\overline{()}$	Expected value (mean)
$\widehat{()}$	Estimated value
$\dot{()}$	First derivative with respect to time
$\ddot{()}$	Second derivative with respect to time

Subscripts

$()_T$	Truncated discrete signal (in time domain)

Chapter 1

Introduction

In this thesis, we develop a system identification methodology for determining harmonic transfer functions of linear time periodic (LTP) systems. In this chapter, we first present a brief discussion of the motivation for this effort, followed by a commentary on previous work related to LTP system identification. Finally, a brief outline of the thesis is described in the last section of the chapter.

1.1 Motivation

Helicopter rotors often experience significant vibrations in forward flight, due to unsteady airloading on the helicopter blades. It has been determined that these load variations primarily arise from the interaction of each rotor blade with the vortex of the preceding blade [1]. Atmospheric turbulence, blade/fuselage interaction, and blade instabilities (air resonances) are also contributing factors [2]. If this vibration problem can be addressed, pilot effectiveness and passenger comfort can be increased significantly. Furthermore, effective vibration reduction can considerably lower the maintenance and operational costs of helicopters.

The airloading on each rotor blade can be considered to be almost periodic, and the forces on the blades are harmonics of the rotor frequency, Ω. Due to the rotor symmetry, only those vibrations at frequencies that are integer multiples of the blade passage frequency ($N_b\Omega$, where N_b is the number of blades) get transmitted to the fuselage through the rotor mast. One way to reduce vibration, then, is to alter the periodic forces to cancel or suppress the vibration-inducing harmonics.

In order to devise effective control strategies and ultimately develop useful controllers, it is important to have a good understanding of the rotor dynamics. Because the rotor aerodynamics are periodic, the rotor dynamics cannot be modeled as time-invariant, although they may be approximated as linear. The development of a system identification scheme that can determine the dynamics of a time periodic

system is therefore highly desirable, since such a tool will greatly aid in controller design.

It should also be noted that such a development will not only be applicable in the specific case of helicopter vibration reduction, but will be useful for a variety of other mechanical and aeronautical systems. Wereley [4] mentions a few systems in this regard, such as wind turbines, the rolling motion of ships, and gravitationally stabilized earth pointing satellites. Furthermore, once a scheme for determining the dynamics of an LTP system has been developed, linear time invariant (LTI) systems can also be incorporated in the set of systems suitable for analysis by such a tool. We will discuss this point in Section 3.3.1 in more detail.

1.2 Background

Before we embark on our specific goal of developing an identification (ID) scheme for LTP systems, we will briefly analyze the methods previously employed for analyzing the dynamics of periodic systems.

1.2.1 Analytical Methods

Richards [5] presented an analysis on a number of systems described by second (and in some cases higher) order differential equations with periodic coefficients. He extensively studied various types of periodic coefficients (saw tooth, rectangular, periodic exponential, *etc.*) and derived their solutions. However, his formulation represented the system outputs in terms of the input signals modulated with periodic terms that may contain an infinite number of harmonics of the system's *pumping frequency* (ω_p). The pumping frequency is the frequency of variation of the system coefficients, and is distinct from the commonly used *forcing frequency,* which is the frequency of input excitation.

Richards did not develop an operator that mapped input signals to output signals (the way the transfer function of an LTI system does). For instance, his derivation for the steady state forced response, x_{ss}, of a general periodic system to a sinusoidal input of frequency, ω_i, was expressed as an infinite sum of periodic coefficients, ν_{nk}, so that

$$x_{ss}(t) = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \nu_{nk} \exp\left[j\left(\omega_i + (n+k)\omega_p\right)t\right] \tag{1.1}$$

Such a description, however, is not helpful for developing a convenient system ID method that may be implemented with reasonable accuracy.

Wereley and Hall [6] built on the work of several researchers, including Richards, and represented the response of an LTP system by defining a linear operator that relates input and output complex exponential signals. Since these signals are of fundamental importance to LTP systems, their theoretical

developments have aided in our efforts for defining and validating our identification scheme. They derived a frequency domain representation of the LTP system model that is analogous to the widely used frequency domain formulations of LTI systems. The linear operator they developed was called the *Harmonic Transfer Function* (HTF), which described the relationship between the harmonics of the input signal, $u(t)$, and those of the output signal, $y(t)$, as

$$\mathcal{Y} = \widehat{\mathcal{G}}(s)\mathcal{U} \tag{1.2}$$

where $\widehat{\mathcal{G}}(s)$ is the HTF, and $\mathcal{U}$ and $\mathcal{Y}$ are vectors representing the harmonics of the signals $u(t)$, and $y(t)$, respectively. Further elaboration on these harmonic transfer functions will be presented in the next chapter.

1.2.2 Empirical Methods

In the specific case of control of helicopter vibrations, the problem of describing the time periodic plants has usually been addressed by using the so-called **T** matrix formulation, especially in the so-called Higher Harmonic Control (HHC) approach [1]. In HHC, the rotor blades are controlled at the blade passage frequency. The **T** matrix is a control response matrix that relates the sine and cosine components of the input at the N/rev frequency to the sine and cosine components of the output response, also at N/rev. The matrix **T** depends on the flight conditions, which in principle necessitates the measurement and storage of large libraries of **T** matrices corresponding to various flight conditions. In some HHC approaches, the **T** matrix is assumed to be unknown, and is therefore estimated in real-time, yielding an adaptive controller. In other cases, it is treated as a known quantity, and those applications use several **T** matrix libraries for control purposes [3].

Some investigators have claimed that large variations in the matrix **T** necessitate the use of adaptive control [3]. Shaw [7], however, obtained contrary results in his wind-tunnel tests, and determined that the variation in the matrix **T** was small throughout the wind-tunnel test envelope. He attributed his results to better measurement and precise rotor blade control techniques. Shaw's results also showed that since the variation of the control response with changing flight conditions was small, fixed gain controllers perform as well as adaptive controllers for vibration reduction.

Many researchers studying the HHC problem have used a quasi-steady assumption, thereby eliminating the need for a model of the periodic rotor dynamics. They represented the rotor response as

$$\mathbf{z} = \mathbf{T}\mathbf{u} + \mathbf{z}_0 \tag{1.3}$$

where $\mathbf{z}$ is the vector of vibration amplitudes, $\mathbf{u}$ is the vector of control amplitudes, and $\mathbf{z}_0$ is the vector of vibration amplitudes with no control. The $\mathbf{T}$ matrix thus served to describe the system under control, and was usually determined empirically during wind-tunnel and flight tests.

Since the $\mathbf{T}$ matrix does not provide information about the system dynamics, and therefore closed-loop stability, the control approaches have been somewhat *ad hoc*. The system identification methodology that will be developed in the following chapters aims to provide the means for determining the actual dynamic behavior of helicopter rotors. Once the periodic nature of the system dynamics has been identified, future control implementations will be greatly improved. We will discuss system ID implications on controller design in more detail in the final chapter.

1.3 Thesis Overview

The objective of this thesis is to present the theoretical development of a system identification method for LTP systems, with a special focus on helicopter rotor systems. Also, this thesis presents the description of optimized software that has been developed for the implementation of our LTP system ID scheme.

In Chapter 2, an overview of harmonic transfer functions and important properties of LTP systems are first presented, along with a brief discussion on the development of the harmonic state-space model. Chapter 3 deals with the theoretical development of an identification scheme for determining harmonic transfer functions of LTP systems. The various assumptions, simplifications, and limitations of the scheme are also discussed. In the latter sections of the chapter, the actual implementation of the estimation scheme is described in detail. Chapter 4 presents some specific issues pertaining to input signal generation, data acquisition, and data reduction for helicopter rotor systems. Chapter 5 discusses the validation of the developed scheme by analyzing a few LTP systems, and comparing the transfer functions obtained analytically and empirically through our system ID software routine. Conclusions and recommendations for future work are given in Chapter 6. Various implications of our ID scheme on control applications are also discussed. Some of the limitations encountered in our effort, and possible future solutions are presented. All the MATLAB and C codes used in our analysis and implementation have been included in Appendix A and B, respectively.

Chapter 2

Frequency Response of Linear Time Periodic Systems

As discussed in the previous chapter, we need to minimize the periodic unsteady aerodynamic loads on helicopter blades in order to reduce helicopter vibrations. This requires a good understanding of the periodic dynamics of the rotor, so that effective controllers may be designed. Therefore, we present a brief summary of the characteristics of linear time periodic systems in this chapter. We also provide an overview of the development of a linear operator for describing transfer properties of LTP systems. The discussion in this chapter will set the stage for the formulation of an estimation technique to identify dynamic characteristics of LTP systems.

2.1 Overview of Frequency Response of LTI Systems

When an input $u(t)$ is applied to a linear time invariant system, its output $y(t)$ can be expressed as

$$y(t) = \int_{-\infty}^{\infty} g(\tau)u(t-\tau)d\tau \tag{2.1}$$

where $g(\tau)$ is the *impulse response* of the system. Because the output depends on past inputs, we usually have that

$$y(t) = \int_{0}^{\infty} g(\tau)u(t-\tau)d\tau \tag{2.2}$$

The response to an exponential,

$$u(t) = e^{st} \tag{2.3}$$

is

$$\begin{aligned} y(t) &= \int_{-\infty}^{\infty} g(\tau) e^{s(t-\tau)} d\tau \\ &= \int_{-\infty}^{\infty} g(\tau) e^{-s\tau} e^{st} d\tau \\ &= \int_{-\infty}^{\infty} g(\tau) e^{-s\tau} d\tau \; e^{st} = G(s) e^{st} \end{aligned} \tag{2.4}$$

where $G(s)$ is the *transfer function* of the LTI system, given by

$$G(s) = \int_{-\infty}^{\infty} g(t) e^{-st} dt = \mathcal{L}\{g(t)\} \tag{2.5}$$

Note that $G(s)$ is the Laplace transform of the impulse response. Alternatively, for imaginary s ($s = j\omega$), we may express the transfer function as the Fourier transform of the impulse response, so that

$$G(j\omega) = \mathcal{F}\{g(\tau)\} = \int_{-\infty}^{\infty} g(\tau) e^{-j\omega\tau} d\tau \tag{2.6}$$

It is clear from Equation 2.4 that $G(s)$ provides a linear map between the input and output of the system. It characterizes the dynamic properties of the system by describing its frequency response. If the input is $e^{j\omega t}$, then the output is $G(j\omega)\; e^{j\omega t}$.

The differential equations governing the dynamics of an LTI system have constant coefficients, and are often represented in the state-space form as

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.7}$$

$$y(t) = Cx(t) + Du(t) \tag{2.8}$$

where $x\,(t)$ is the state vector, A is the dynamics matrix, B is the input matrix, C is the output matrix, and D is the feedthrough matrix. The transfer function for a system described as above is

$$G(s) = C(sI - A)^{-1}B + D \tag{2.9}$$

where I is the identity matrix.

2.2 Linear Time Periodic Systems

Linear time periodic systems are more difficult to characterize than LTI systems. In LTP systems, the coefficients of the differential equations that describe the dynamics are time-varying and periodic. These systems may be modeled in state-space form as

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \tag{2.10}$$

$$y(t) = C(t)x(t) + D(t)u(t) \tag{2.11}$$

where the matrices $A(t)$, $B(t)$, $C(t)$, and $D(t)$ are periodic, with period T. In other words,

$$A(t + NT) = A(t) \tag{2.12}$$

for any integer N, and similarly for B, C, and D.

When a complex exponential (or sinusoid) is used to excite an LTP system, the output response consists of a superposition of sinusoids not only at the input frequency ω, but also at several (possibly an infinite number) other frequencies, $\omega + n\omega_p$, each with its own magnitude and phase [6], where n is an integer, and ω_p is the pumping frequency, given by

$$\omega_p = 2\pi/T \tag{2.13}$$

The frequencies $n\omega_p$ are harmonics of the pumping frequency. Thus, the frequencies $\omega + n\omega_p$ are shifted harmonics, and we often refer to these frequencies simply as "harmonics."

The description of the dynamic behavior of an LTP system response is more complex, but nonetheless it is highly desirable to have a means of characterizing such systems. In this regard, Wereley and Hall [4] have discussed the notion of harmonic transfer functions for LTP systems that are analogous to transfer functions of LTI systems. A brief summary of their work is presented below.

2.2.1 Fundamental Signal Spaces in LTP System Analysis

Floquet theory has been widely used in the study of LTP systems to derive several important results. According to this theory of LTP systems, the state vector x, at time t, is related to the state vector a full period away by the system's *discrete transition matrix* Φ [5], so that

$$x(t + T) = \Phi x(t). \tag{2.14}$$

Φ is defined as the *state transition matrix* of the system evaluated over the interval $(0, T)$, so that

$$\Phi = \phi(T, 0) \tag{2.15}$$

where ϕ is the state transition matrix from τ to t. It is easily shown that

$$\phi(nT, 0) = \phi(T, 0)^n \tag{2.16}$$

Therefore, in general

$$x(t + nT) = \Phi^n x(t) \tag{2.17}$$

When $x(0)$ is an eigenvector of Φ, corresponding to eigenvalue z, the solution $x(t)$ will satisfy

$$x(t + nT) = z^n x(t)$$

for all t. This implies that $x(t)$ has the form

$$x(t) = e^{st}\overline{x}(t) \tag{2.18}$$

where $s = (log\ z)/T$, and $x(t)$ is periodic. That is, $x(t)$ is an *exponentially modulated periodic* (EMP) function. This suggests, by analogy to LTI system analysis, that EMP functions are the appropriate signals to use to describe the input-output relationship of an LTP system, just as exponentials are used to describe input-output relationship for LTI systems. This eventually lead Wereley and Hall to develop the concept of using EMP test signals to determine transfer functions for LTP systems. They expressed EMP signals as the complex Fourier series of a periodic signal of frequency ω_p, modulated by a complex exponential signal,

$$u(t) = \sum_{n \in Z} u_n e^{s_n t} \tag{2.19}$$

where $s_n = s + nj\omega_p$ $(s \in C)$, and u_n are Fourier coefficients of $u(t)$.

2.2.2 Harmonic Transfer Functions

After defining EMP signals, Wereley and Hall expressed the elements of the matrices (Equations 2.10 and 2.11) in terms of their Fourier series, and used the harmonic balance approach to arrive at

$$\begin{aligned} s_n x_n &= \sum_{m \in Z} A_{n-m} x_m + \sum_{m \in Z} B_{n-m} u_m \\ y_n &= \sum_{m \in Z} C_{n-m} x_m + \sum_{m \in Z} D_{n-m} u_m \end{aligned} \tag{2.20}$$

After expressing the summations in matrix form, the equations reduce to

$$\begin{aligned} s\mathcal{X} &= (\mathcal{A}-\mathcal{N})\mathcal{X}+\mathcal{B}\mathcal{U} \\ \mathcal{Y} &= \mathcal{C}\mathcal{X}+\mathcal{D}\mathcal{U} \end{aligned} \tag{2.21}$$

The state vector $\mathcal{X}$ represents the states at various harmonics of a given frequency, and is expressed as

$$\mathcal{X} = \begin{bmatrix} \vdots \\ x_{-2} \\ x_{-1} \\ x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} \tag{2.22}$$

The dynamics matrix $\mathcal{A}$ is a doubly-infinite Toeplitz matrix, given by

$$\mathcal{A} = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \\ \cdots & A_0 & A_{-1} & A_{-2} & \cdots \\ \cdots & A_1 & A_0 & A_{-1} & \cdots \\ \cdots & A_2 & A_1 & A_0 & \cdots \\ & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.23}$$

The matrix A_n is the nth Fourier coefficient of $A(t)$. The matrices $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ are similarly defined. The modulation frequency matrix, $\mathcal{N}$, is an infinite block diagonal matrix, given by

$$\mathcal{N} = \begin{bmatrix} \ddots & & & & & & \\ & -2\cdot j\omega_p I & & & 0 & & \\ & & -1\cdot j\omega_p I & & & & \\ & & & 0\cdot j\omega_p I & & & \\ & 0 & & & 1\cdot j\omega_p I & & \\ & & & & & 2\cdot j\omega_p I & \\ & & & & & & \ddots \end{bmatrix}$$

where I is the identity matrix of same dimensions as that of A_n.

The harmonic transfer function (HTF) is defined as an operator that relates harmonics of the input

signal to harmonics of the output signals, and is given by

$$\widehat{\mathcal{G}}(s) = \mathcal{C}[sI - (\mathcal{A} - \mathcal{N})]^{-1}\mathcal{B} + \mathcal{D} \tag{2.24}$$

The LTP transfer function is thus analogous to the widely used LTI transfer function, in that it describes the input-output properties of LTP systems in the frequency domain. Although the matrices in Equation 2.21 are infinite (due to the infinite Fourier coefficients), for practical purposes we can truncate the number of terms in the Fourier series, and simply use the smallest number of harmonics that adequately represent the system dynamics. Since the harmonics generally get smaller with increasing harmonic number, only the consideration of the first few harmonics is usually adequate in describing the system behavior.

We can now employ this notion of harmonic transfer functions in developing a method for system identification of LTP systems.

Chapter 3

Linear Time Periodic System Identification Scheme

In this chapter we present a procedure to identify harmonic transfer functions of LTP systems. We first describe the theoretical development, and then discuss our actual implementation of the identification scheme.

3.1 Overview of LTI System Identification

For a linear time invariant system, the transfer function estimate, $\widehat{G}(j\omega)$, is given by the ratio of the Fourier transforms of the input, $u(t)$, and output, $y(t)$, so that

$$\widehat{G}(j\omega) = \frac{\mathcal{F}\{y(t)\}}{\mathcal{F}\{u(t)\}} \tag{3.1}$$

In the absence of noise, and assuming that $u(t)$ has components at all frequencies, $\widehat{G}(j\omega)$ should be a good estimate of the transfer function, $G(j\omega)$. This estimate will be in error if the data sets (of the input and output measurements) have significant noise. Consider a system that has measurement noise, $e(t)$. Then

$$y(t) = g(t) * u(t) + e(t) \tag{3.2}$$

where $*$ denotes the convolution operator. For stationary processes, the cross-correlation of an input and output, $R_{uy}(\tau)$, is given as the expected value of $u(t)$ and $y(t+\tau)$, so that

$$
\begin{aligned}
R_{uy}(\tau) &= \overline{u(t)y(t+\tau)} \\
&= E\left\{ u(t) \left[\int_0^\infty g(r)u(t+\tau-r)dr + \int_0^\infty e(t+\tau-r)dr \right] \right\} \\
&= \int_0^\infty g(r)\overline{u(t)u(t+\tau-r)}dr + \int_0^\infty \overline{u(t)e(t+\tau-r)}dr
\end{aligned} \tag{3.3}
$$

For measurement noise e that is independent of u, and has zero mean, we have that

$$
\overline{u(t)e(t+\tau-r)} = 0 \tag{3.4}
$$

Therefore,

$$
\begin{aligned}
R_{uy}(\tau) &= \int_0^\infty g(r)\overline{u(t)u(t+\tau-r)}dr \\
&= \int_0^\infty g(r)R_{uu}(\tau-r)dr \ .
\end{aligned} \tag{3.5}
$$

where $R_{uu}(\tau)$ is the autocorrelation of the input signal. Equation 3.5 can be written as

$$
R_{uy}(\tau) = g(\tau) * R_{uu}(\tau) \tag{3.6}
$$

since the last integral in Equation 3.5 is simply the convolution integral. Converting the above equation into the frequency domain (by taking Fourier transform of the cross-correlation function), we get the cross-spectral density, Φ_{uy}, and power spectral density, Φ_{uu},which are related by

$$
\begin{aligned}
\Phi_{uy}(\omega) &= \widehat{G}(j\omega)\Phi_{uu}(\omega) \\
\widehat{G}(j\omega) &= \frac{\Phi_{uy}(\omega)}{\Phi_{uu}(\omega)} \ .
\end{aligned} \tag{3.7}
$$

This relation, with the auto and cross-spectra of the input and output, is widely used for the so-called *empirical transfer function estimate* (ETFE) of linear time invariant systems [9].

We will now extend this development further for linear time periodic systems.

3.2 LTP System Identification

In an LTP system, an input sinusoid at a single frequency generates a superposition of sinusoids at several frequencies of various magnitudes and phase in the output. Thus, common LTI system ID techniques

cannot be used to determine its transfer functions. Some simplifying assumptions, however, can be made which aid in developing a system ID method for an LTP system with specific application for our helicopter rotor system.

To illustrate the approach, we will initially account for only three frequencies in the output of an LTP system, for each input frequency. The LTP system will have period T, and corresponding frequency ω_p. We therefore assume that the output, Y, at each frequency ω, comprises of the linear combination of the responses due to inputs at frequencies ω, $\omega + \omega_p$, and $\omega - \omega_p$. The system output can then be assumed to be a linear combination of three different transfer functions (each corresponding to one of the three frequencies in the output): G_0, G_1, and G_{-1}, respectively. Y can thus be expressed as

$$Y(j\omega) = G_0(j\omega)U(j\omega) + G_1(j\omega)U(j\omega - j\omega_p) + G_{-1}(j\omega)U(j\omega + j\omega_p) \tag{3.8}$$

Equivalently in the time domain, the output may be written as

$$y(t) = g_0(t) * u(t) + g_1(t) * [u(t)e^{j\omega_p t}] + g_{-1}(t) * [u(t)e^{-j\omega_p t}] \tag{3.9}$$

From Equation 3.9, one can see that the output $y(t)$ is defined as the total response due to an input $u(t)$ that has been modulated appropriately, and then convolved with the respective impulse response functions. We can state more specifically that the nth transfer function is defined to be the linear operator that maps the output at frequency ω to an input, at frequency ω, modulated with $e^{jn\omega_p t}$. This linear system can be represented in the block diagram of Figure 3-1.

As is evident from Equations 3.8 and 3.9, we have three transfer functions, G_0, G_1, and G_{-1}, that need to be estimated. For a given input $U(j\omega)$ and resulting output $Y(j\omega)$, we thus have three unknowns, but only one equation. Our identification problem is therefore underdetermined. One approach to solve this problem is to apply three inputs and measure the resulting outputs. We can thus form three independent equations of the form of Equation 3.8, and will subsequently be able to evaluate the desired transfer functions.

In order to generate the three outputs, it is extremely important to take into consideration the time of application of each input relative to the system period. This is due to the time-varying nature of the system dynamics during one period. If the system behavior is to be completely characterized, the system needs to be excited with appropriate input signals at various times in its period T. Since in this case only three transfer functions are being evaluated, three identical input signals should be applied that are evenly spread out over the system period. Figure 3-2 depicts when each of the three inputs should be initiated, relative to the system period.

We define T_d, as the amount of time (in seconds) elapsed between the beginning of a new period of

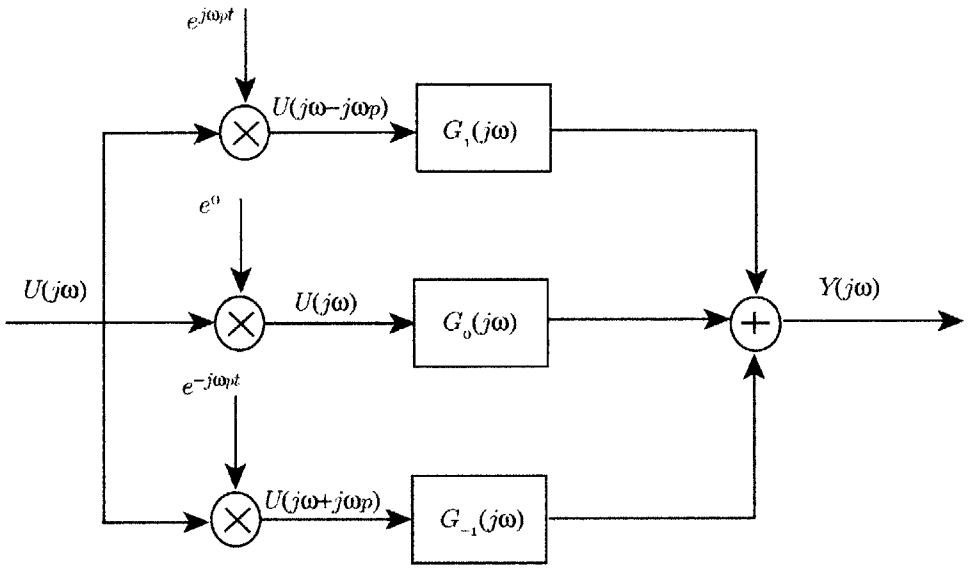

Figure 3-1: LTP system model with three transfer functions.

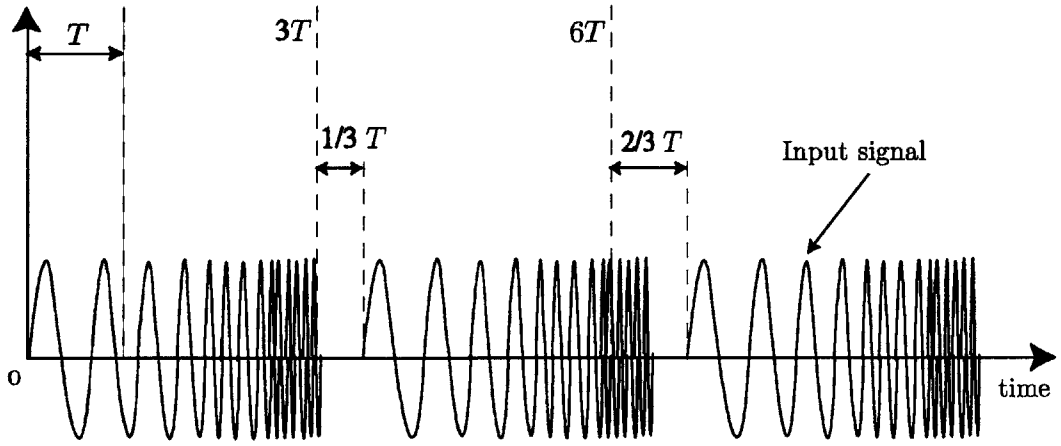

Figure 3-2: Input signals initiated at appropriate time intervals over the system periods.

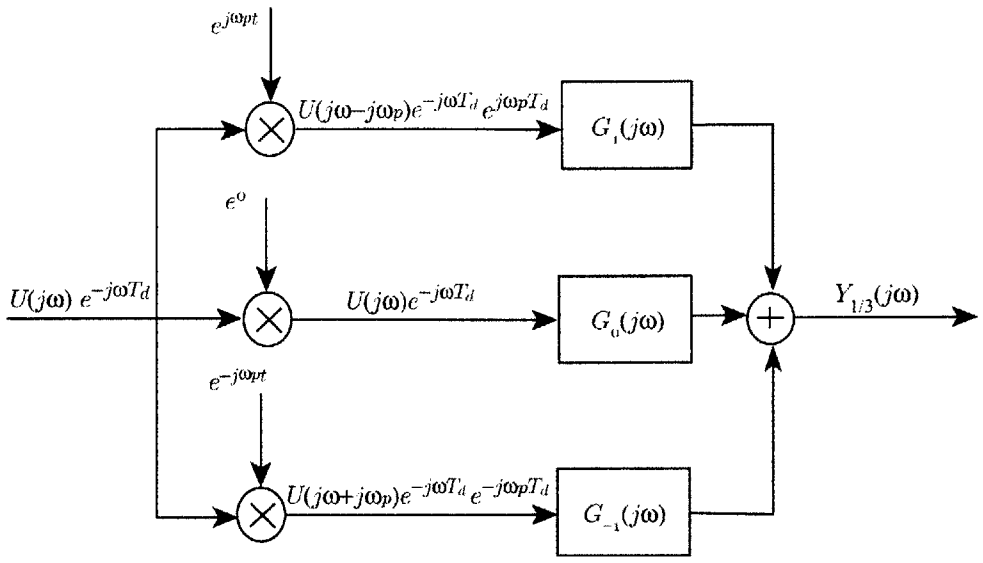

Figure 3-3: Delayed input and corresponding output of LTP system.

the system and the start of the first input signal since the beginning of that particular period. In this specific case, where we have just established that the three inputs should be spaced uniformly apart, T_d is given as

$$T_d = T/3 = 2\pi/3\omega_p \tag{3.10}$$

The first input should have zero delay between the start of a system period and its time of initiation. The input U, and output Y_0, to the system are modeled as depicted in Figure 3-1. For the second chirp, there should be a delay of T_d seconds between the start of a system period and its time of application, therefore the actual input will be $U(j\omega)e^{-j\omega T_d}$. The system output is modeled as shown in Figure 3-3. Similarly, the delay time for the third input will be $2T_d$. The output vector Y then can be expressed as

$$\begin{bmatrix} Y_0 \\ Y_{1/3} \\ Y_{2/3} \end{bmatrix} = \begin{bmatrix} U(j\omega) & U(j\omega - j\omega_p) & U(j\omega + j\omega_p) \\ U(j\omega) & U(j\omega - j\omega_p)e^{j\omega_p T_d} & U(j\omega + j\omega_p)e^{-j\omega_p T_d} \\ U(j\omega) & U(j\omega - j\omega_p)e^{j\omega_p 2T_d} & U(j\omega + j\omega_p)e^{-j\omega_p 2T_d} \end{bmatrix} \begin{bmatrix} G_0 \\ G_1 \\ G_{-1} \end{bmatrix} \tag{3.11}$$

where $Y_{1/3}$ and $Y_{2/3}$ are the outputs due to the second and third inputs respectively. If we define $W = e^{j\omega_p T_d}$, then substituting the value of T_d, we get

$$W = e^{j2\pi/3} \tag{3.12}$$

From this definition, we have $W^{-1} = e^{-j2\pi/3}$, and therefore $W^2 = e^{j4\pi/3} = W^{-1}$, and $W^4 = W^{-2}$. Equation 3.11 then becomes

$$\begin{bmatrix} Y_0 \\ Y_{1/3} \\ Y_{2/3} \end{bmatrix} = \begin{bmatrix} U(j\omega) & U(j\omega - j\omega_p) & U(j\omega + j\omega_p) \\ U(j\omega) & U(j\omega - j\omega_p)W & U(j\omega + j\omega_p)W^{-1} \\ U(j\omega) & U(j\omega - j\omega_p)W^2 & U(j\omega + j\omega_p)W^{-2} \end{bmatrix} \begin{bmatrix} G_0 \\ G_1 \\ G_{-1} \end{bmatrix} \tag{3.13}$$

If we separate out the W terms, we get

$$\underbrace{\begin{bmatrix} Y_0 \\ Y_{1/3} \\ Y_{2/3} \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & W & W^2 \\ 1 & W^2 & W^4 \end{bmatrix} \begin{bmatrix} U(j\omega) & 0 & 0 \\ 0 & U(j\omega - j\omega_p) & 0 \\ 0 & 0 & U(j\omega + j\omega_p) \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} G_0 \\ G_1 \\ G_{-1} \end{bmatrix}}_{G} \tag{3.14}$$

The first matrix on the right hand side of Equation 3.14 is the same as the one that commonly arises in discrete Fourier transform notations. We can also write Equation 3.14 more simply as

$$Y = UG \tag{3.15}$$

so that

$$G = U^{-1}Y \tag{3.16}$$

Each row of the U matrix corresponds to an input signal, and each column corresponds to the harmonics. Note that the inputs should be chosen so that U is non-singular, since U^{-1} is needed to compute G.

These set of equations have been developed for determining three transfer functions. However, it is clear that we can easily extend this method for determining any number of transfer functions of a system. We can apply several input signals, spaced appropriately relative to system period, and account for more transfer functions. The observed output, Y, can be refined and smoothed using autocorrelations of the spectra. We will discuss this issue in more detail in the next section.

It is important to point out, that in this approach, the output measurements due to each input must be obtained by allowing the response to settle down significantly before the next input signal is initiated. In that case, it can be reasonably assumed that Y_0 is due to the first input, $Y_{1/3}$ due to the second input, and so on. However, to speed up the system identification process, we can generate inputs (at appropriate system phase) with very little idle time between each successive signal, and effectively treat the entire sequence as one input signal. The resulting outputs from the successive inputs are also taken as one effective output signal. In this case the problem becomes underdetermined, since we have more transfer functions to identify than we have known quantities. In order to obtain a problem that is

adequately defined, we can make assumptions about certain characteristics of G. We can thus constrain the problem by applying those assumptions, and the transfer functions can be subsequently identified. In our system ID scheme, we essentially use this second approach to develop an efficient identification method.

3.3 ID Scheme Implementation

A separate section describing the implementation of our ID method is necessary, since all the preceding sections have dealt with continuous-time equations, while in the actual case we deal with discrete data from digitally sampled signals of finite time duration. In the following subsections, we present a description of our system identification scheme, which has been implemented as a MATLAB tool. We will also give a detailed discussion of our estimation error minimization methodology, and a brief section on issues related to MATLAB efficiency, processor speed, and memory.

3.3.1 Data Transformations

Our system ID algorithm is primarily based on the results obtained in the previous sections, with some modifications for practical implementation. The ID routine requires three sets of data, specifically the input u, output y, and time measurements (at which u and y occur) ψ. In the case of a rotor system, the values of ψ are the azimuth measurements. The data is assumed to have been digitally acquired, hence all the data points can be assembled in a vector of length n, where n is the total number of data points. Each data point is essentially a digital record of the measurement of an analog signal determined at some fixed sampling frequency. The input data is therefore expressed as

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & \cdots & u_n \end{bmatrix} . \tag{3.17}$$

$\mathbf{y}$ and $\boldsymbol{\psi}$ are similarly defined.

The total number of transfer functions of the system that need to be identified, n_h, have to be specified in advance. Recall that in Equation 3.14 we had built a matrix U, whose elements were modulated inputs in the frequency domain. An $n_h \times n$ matrix $\mathbf{U}$ is therefore constructed, where each row consists of an

appropriately modulated and Fourier transformed data vector $\mathbf{u}$, so that

$$\mathbf{U} = \begin{bmatrix} \mathcal{F}\{[e^{mj\psi_1}u_1 & \cdots & \cdots & e^{mj\psi_n}u_n]\} \\ \vdots & & \vdots & \vdots \\ \mathcal{F}\{[e^{1j\psi_1}u_1 & \cdots & \cdots & e^{1j\psi_n}u_n]\} \\ \mathcal{F}\{[e^{0j\psi_1}u_1 & \cdots & \cdots & e^{0j\psi_n}u_n]\} \\ \mathcal{F}\{[e^{-1j\psi_1}u_1 & \cdots & \cdots & e^{-1j\psi_n}u_n]\} \\ \vdots & & \vdots & \vdots \\ \mathcal{F}\{[e^{-mj\psi_1}u_1 & \cdots & \cdots & e^{-mj\psi_n}u_n]\} \end{bmatrix}, \tag{3.18}$$

where $m = \dfrac{n_h - 1}{2}$. In more compact notation, the $\mathbf{U}$ matrix is really

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}(\omega - m\omega_p) \\ \vdots \\ \mathbf{u}(\omega - \omega_p) \\ \mathbf{u}(\omega) \\ \mathbf{u}(\omega + \omega_p) \\ \vdots \\ \mathbf{u}(\omega + m\omega_p) \end{bmatrix} \tag{3.19}$$

Similarly, we define a $\mathbf{Y}$ matrix analogous to our output vector in equation 3.14, as the discrete Fourier transform of vector $\mathbf{y}$

$$\mathbf{Y} = \mathcal{F}\left\{\begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_n \end{bmatrix}\right\} \tag{3.20}$$

In Section 3.1, it was mentioned that Equation 3.7, involving the cross- and power spectral densities, is used for the ETFE of LTI systems. In our discrete implementation, we will also use an analogous form.

Recall that in our derivation of Equation 3.7, the Fourier integral limits were from zero to infinity, therefore for a random process an absolute integral will not exist. But, consider a stochastic signal $u(t)$, that has been truncated to a time span of T_s. When T_s is large, then it has been shown [9] that

$$\lim_{T_s \to \infty} E\left[\frac{1}{T_s}\left|\mathcal{F}\{u_{T_s}(t)\}\right|^2\right] = \int_{-\infty}^{\infty} R_{uu}(\tau)e^{-j\omega\tau}d\tau \tag{3.21}$$

where E is the expectation operator. The power spectral density and cross-spectral density can thus be

written in the limit as

$$\Phi_{uu}(\omega) = \frac{\overline{|U(\omega)|^2}}{2T_s} = \frac{\overline{U^*(\omega)U(\omega)}}{2T_s} \tag{3.22}$$

and

$$\Phi_{uy}(\omega) = \frac{\overline{U^*(\omega)Y(\omega)}}{2T_s} \tag{3.23}$$

where $U^*(\omega)$ is complex conjugate of $U(\omega)$. Using Equations 3.22 and 3.23, we define

$$\mathbf{\Phi_{UU}} = \mathbf{U}^{*T}\mathbf{U} \tag{3.24}$$

and

$$\mathbf{\Phi_{UY}} = \mathbf{U}^{*T}\mathbf{Y} \tag{3.25}$$

where $\mathbf{U}^{*T}$ is complex conjugate transpose of $\mathbf{U}$. We can now write an expression, analogous to Equation 3.7, that represents harmonic transfer functions of LTP systems as

$$\widehat{\mathbf{G}}(\omega) = (\mathbf{\Phi_{UU}})^{-1}\mathbf{\Phi_{UY}} \tag{3.26}$$

$\widehat{\mathbf{G}}(\omega)$ is a matrix whose row vectors are the transfer functions $\mathbf{g}_i$, that is,

$$\widehat{\mathbf{G}}(\omega) = \begin{bmatrix} \mathbf{g}_m \\ \vdots \\ \mathbf{g}_1 \\ \mathbf{g}_0 \\ \mathbf{g}_{-1} \\ \vdots \\ \mathbf{g}_{-m} \end{bmatrix} \tag{3.27}$$

It is important to note that ordinary matrix multiplication is not performed in the computation of $\mathbf{\Phi_{UU}}$ and $\mathbf{\Phi_{UY}}$. Rather, each column vector of $\mathbf{U}$ is transposed to form a row vector, and multiplied with its corresponding column vector to yield an $n_h \times n_h$ matrix. Each of these matrices is assembled in a three dimensional $n_h \times n_h \times n$ array to form $\mathbf{\Phi_{UU}}$. $\mathbf{\Phi_{UY}}$ is similarly constructed as an $n_h \times 1 \times n$ array.

Consider the 2-D matrix, *i.e.*, all the rows and columns, in the ith position of the third dimension of

$\mathbf{\Phi_{UU}}$ and denote it as $\mathbf{\Phi}_{UU_i}$, where

$$\mathbf{\Phi}_{UU_i} = \begin{bmatrix} |u_i(\omega - m\omega_p)|^2 & \cdots & u_i(\omega - m\omega_p)u_i^*(\omega) & \cdots & u_i(\omega - m\omega_p)u_i^*(\omega + m\omega_p) \\ \vdots & \ddots & \vdots & & \vdots \\ u_i(\omega)u_i^*(\omega - m\omega_p) & \cdots & |u_i(\omega)|^2 & \cdots & u_i(\omega)u_i^*(\omega + m\omega_p) \\ \vdots & & \vdots & \ddots & \vdots \\ u_i(\omega + m\Omega)u_i^*(\omega - m\omega_p) & \cdots & u_i(\omega + m\omega_p)u_i^*(\omega) & \cdots & |u_i(\omega + m\omega_p)|^2 \end{bmatrix} \tag{3.28}$$

Similarly for the $\mathbf{\Phi_{UY}}$ array, the ith vector in its third dimension is

$$\mathbf{\Phi}_{UY_i} = \begin{bmatrix} u_i^*(\omega - m\omega_p)y_i(\omega) \\ \vdots \\ u_i^*(\omega - \omega_p)y_i(\omega) \\ u_i^*(\omega)y_i(\omega) \\ u_i^*(\omega + \omega_p)y_i(\omega) \\ \vdots \\ u_i^*(\omega + m\omega_p)y_i(\omega) \end{bmatrix} \tag{3.29}$$

Note that the $\mathbf{\Phi}_{UU_i}$ matrix is Hermitian, and positive definite. Hence, its inverse will exist, and Equation 3.26 can be utilized for evaluating $\widehat{\mathbf{G}}(\omega)$. Also, note that the specific structure of $\mathbf{\Phi_{UU}}$ has some useful implications for efficient Gaussian elimination. A detailed discussion in this regard will be presented in Section 3.3.3.

Also, note that for $n_h = 1$, (*i.e.,* when only the fundamental transfer function, $\mathbf{g}_0$, needs to be identified) $\mathbf{\Phi_{UU}}$ and $\mathbf{\Phi_{UY}}$ reduce to $1 \times 1 \times n$ arrays, where $\mathbf{\Phi}_{UU_i}$ is $|u_i(\omega)|^2$, and $\mathbf{\Phi}_{UY_i}$ is $u_i^*(\omega)y_i(\omega)$. Each element of $\mathbf{g}_0$ is given as

$$g_{0_i}(\omega) = \frac{u_i^*(\omega)y_i(\omega)}{|u_i(\omega)|^2} \tag{3.30}$$

which is the same relation as that used for ETFE for LTI systems. Thus, as mentioned in Section 1.1, this scheme reduces to LTI system ID if only one transfer function is considered in the calculations.

Once the $\mathbf{\Phi_{UU}}$ and $\mathbf{\Phi_{UY}}$ matrices have been assembled, it is tempting to simply use Equation 3.26 to find our transfer functions. However, such a computation will not yield an accurate result. This is because we have made a simplification in our analysis, and have considered only a few harmonics (instead of an infinite number). The cumulative effect of the neglected harmonics may be significant, therefore, our results will incorrectly identify the few transfer functions that have been evaluated. For instance, suppose a given system has N_h transfer functions of relatively significant magnitudes, but only n_h are

evaluated through this method. In that case, the system has been modeled with n_h number of transfer functions, and its output response due to an input may be expressed as

$$\mathbf{Y} = \underbrace{\sum_{k=-m}^{m} \mathbf{u}(\omega - k\omega_p)\mathbf{g}_k}_{modeled\ part} + \underbrace{\sum_{m<|l|\leq M} \mathbf{u}(\omega - l\omega_p)\mathbf{g}_l}_{unmodeled\ part} , \tag{3.31}$$

where $m = \dfrac{n_h - 1}{2}$ and $M = \dfrac{N_h - 1}{2}$. The unmodeled part essentially appears as an error, $\mathbf{e}$, in our output equation for the ID scheme, since it takes into account only the modeled part. Therefore,

$$\begin{aligned} \mathbf{Y} &= \sum_{k=-m}^{m} \mathbf{u}(\omega - k\omega_p)\mathbf{g}_k + \mathbf{e} \\ &= \mathbf{U}^T\widehat{\mathbf{G}} + \mathbf{e} \, . \end{aligned} \tag{3.32}$$

3.3.2 Estimation Error Reduction

In the previous section, we briefly discussed errors that are introduced in our estimation due to modeling simplifications. In addition to these modeling inadequacies, we also do not have enough constraints on our problem so that we may obtain an accurate estimate. We have only one input and output data set, while we want to identify n_h transfer functions. Therefore, we need to make additional assumptions so that the identification problem is well-posed. In this regard, we assume that the transfer functions are smooth, *i.e.,* there are no rapid variations (with frequency) in the transfer functions. This is a reasonable assumption, since rapid variations with frequency in a transfer function usually are not physical.

In order to reduce $\mathbf{e}$, and to apply the assumption we have just made, we formulate our problem as the minimization of a cost function, J, that penalizes the quadratic error and the curvature of the transfer functions, so that

$$J = \min \left[\left(\mathbf{Y} - \mathbf{U}^T\widehat{\mathbf{G}} \right)^2 + \alpha \left(\mathbf{D}^2\widehat{\mathbf{G}} \right)^2 \right] \tag{3.33}$$

where $\mathbf{D}^2$ is the second difference operator, and α is a constant. Before proceeding onwards, we will briefly digress to present an explanation of the second difference operator.

Consider the classical 1-D Laplace equation

$$\frac{\partial^2 p}{\partial x^2} = f \tag{3.34}$$

where p, is a continuous function of independent variable x, f is a forcing function, and the boundary conditions of p are zero [10]. This function p, can be discretized into k points on a mesh of step size h.

The derivative of p with respect to x, may be approximated in discrete form as

$$\frac{\partial p}{\partial x} \approx \frac{p_{i+1/2} - p_{i-1/2}}{x_{i+1/2} - x_{i-1/2}} \approx \frac{p_{i+1/2} - p_{i-1/2}}{h} \tag{3.35}$$

The second derivative is then approximately

$$\frac{\partial^2 p}{\partial x^2} \approx \frac{(p_{i+1} - p_i) - (p_i - p_{i-1})}{h^2} = \frac{p_{i+1} - 2p_i + p_{i-1}}{h^2} \tag{3.36}$$

The Laplace equation can thus be expressed as a system of linear equations as

$$\begin{aligned} p_2 - 2p_1 + p_0 &= h^2 f_1 \\ p_3 - 2p_2 + p_1 &= h^2 f_2 \\ &\vdots \\ p_k - 2p_{k-1} + p_{k-2} &= h^2 f_{k-1} \\ p_{k+1} - 2p_k + p_{k-1} &= h^2 f_k \end{aligned} \tag{3.37}$$

The boundary values p_0, and p_{k+1}, are zero. In matrix notation, we therefore have

$$\begin{bmatrix} -2 & 1 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & & & \\ & & & 1 & \\ & & & 1 & -2 \end{bmatrix} \begin{bmatrix} p_1 \\ \\ \vdots \\ \\ p_k \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ \\ \vdots \\ \\ f_k \end{bmatrix} \tag{3.38}$$

The tri-diagonal matrix, which we denote as $\mathbf{D}^2$, is the second difference operator, and is widely employed in numerical analysis for evaluating second order differential equations.

Returning back to our discussion, it is evident from Equation 3.33 that by selecting various values for α, we can weight the second derivatives of $\widehat{\mathbf{G}}$ more or less in the cost function, and thus penalize the curvature so that rapid variations with frequency in $\widehat{\mathbf{G}}$ are reduced. In order to get a closed form expression for $\widehat{\mathbf{G}}$, Equation 3.33 is evaluated by taking the derivative of J with respect to $\widehat{\mathbf{G}}$, setting it equal to zero, and then solving for $\widehat{\mathbf{G}}$, which gives

$$\widehat{\mathbf{G}} = \left[\mathbf{U}^T\mathbf{U} + \alpha\mathbf{D}^4\right]^{-1}\mathbf{U}^T\mathbf{Y} \tag{3.39}$$

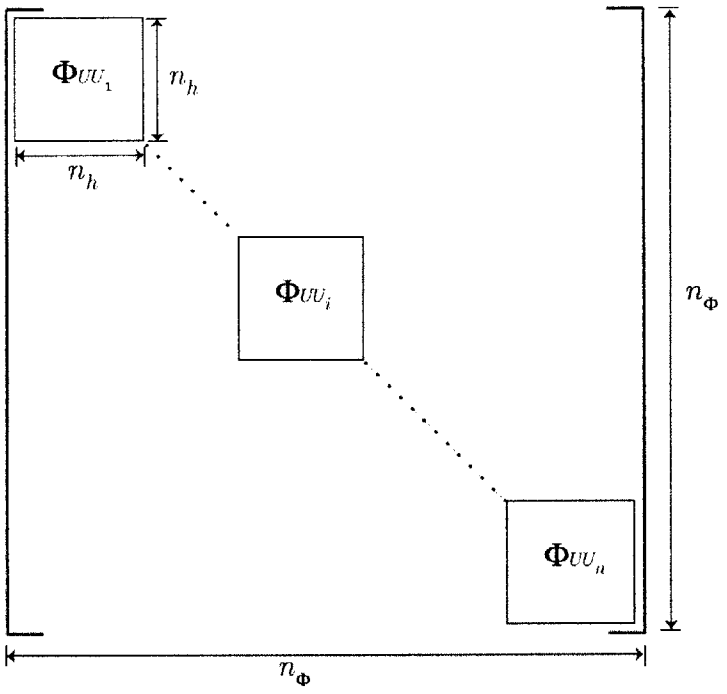

Figure 3-4: Structure of block diagonal $\mathbf{\Phi_{UUs}}$ matrix.

where

$$\mathbf{D}^4 = \mathbf{D}^2 \cdot \mathbf{D}^2 \tag{3.40}$$

Recall that we had assembled $\mathbf{U}^T\mathbf{U}$ and $\mathbf{U}^T\mathbf{Y}$ in three dimensional arrays, $\mathbf{\Phi_{UU}}$ and $\mathbf{\Phi_{UY}}$, respectively. However, in order to numerically evaluate Equation 3.39, we need to reduce these arrays into two-dimensional matrices. The $\mathbf{\Phi_{UU}}$ array is therefore transformed into a square, block diagonal, $n_\Phi \times n_\Phi$ matrix $\mathbf{\Phi_{UUs}}$, in which the ith block is $\mathbf{\Phi}_{UU_i}$, and

$$n_\Phi = n_h \cdot n \tag{3.41}$$

Thus, $\mathbf{\Phi_{UUs}}$ has the form as illustrated in Figure 3-4. $\mathbf{\Phi_{UY}}$ is transformed into a column vector $\mathbf{\Phi_{uy}}$, of length n_Φ, with the form as shown in Figure 3-5. The $\widehat{\mathbf{G}}$ matrix is also transformed into a vector $\widehat{\mathbf{g}}$, where $\widehat{\mathbf{g}}$ has n sub-vectors of length n_h, each of which is a column vector of $\widehat{\mathbf{G}}(\omega)$. Equation 3.39 then becomes

$$\widehat{\mathbf{g}} = \left[\mathbf{\Phi_{UUs}} + \alpha\widetilde{\mathbf{D}}^4\right]^{-1} \mathbf{\Phi_{uy}} \tag{3.42}$$

where $\widetilde{\mathbf{D}}^4$ indicates the change of the dimensions of $\mathbf{D}^4$, similar to the change in dimensions going from

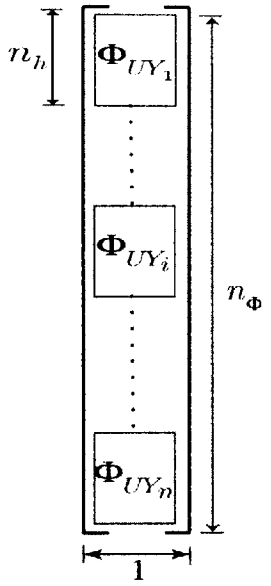

Figure 3-5: Structure of vector $\mathbf{\Phi_{uy}}$.

$\mathbf{U}^T\mathbf{U}$ to $\mathbf{\Phi_{UUs}}$.

After this conversion, it is evident from Equation 3.42 that the harmonic transfer function identification problem now entails the solution of a linear system of the standard form $\mathbf{A}x = b$, where in this case

$$\mathbf{A} = \mathbf{\Phi_{UUs}} + \alpha\widetilde{\mathbf{D}}^4, \tag{3.43}$$

$$b = \mathbf{\Phi_{uy}}, \tag{3.44}$$

and x is the vector of transfer function estimates, *i.e.*, $\widehat{\mathbf{g}}$.

It is important to note from Equation 3.43 that the three diagonals in $\widetilde{\mathbf{D}}^2$ need to be spaced apart according to the block diagonal structure of $\mathbf{\Phi_{UUs}}$. Since each block element is an $n_h \times n_h$ square matrix, the $\widetilde{\mathbf{D}}^2$ matrix has its super-diagonal (of unity elements) starting from the n_hth column, and its sub-diagonal (also of unity elements) starting from the n_hth row. By forming $\widetilde{\mathbf{D}}^2$ in this manner, $\widetilde{\mathbf{D}}^4$ acquires the form of a band diagonal matrix.

Note that the system in Equation 3.42 is analogous to the mechanical system of a beam under bending due to a distributed load along its length. For a horizontal beam (its axis in the x-direction),

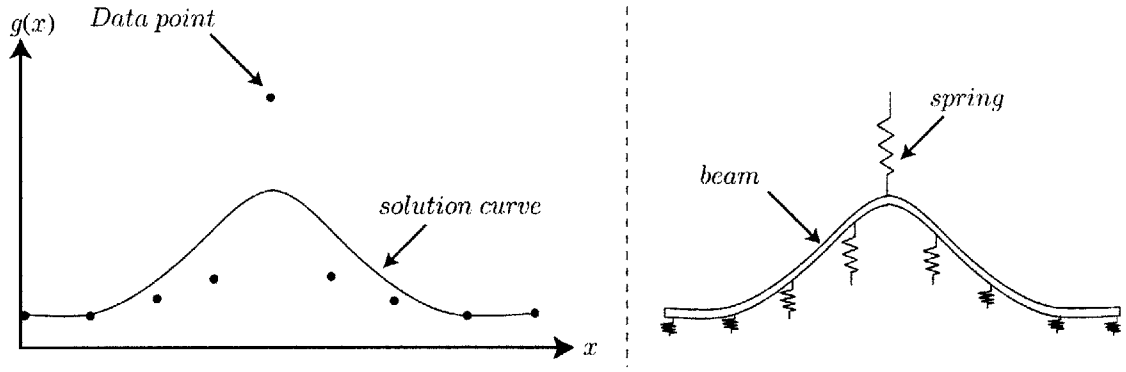

Figure 3-6: Numerical and mechanical analogy for smoothing transfer function estimates.

with stiffness $E_b I_b$, and load per unit length $q(x)$, its deflection $v(x)$ in the vertical direction is given as

$$E_b I_b \frac{\partial^4 v(x)}{\partial x^4} = q\,(x) \tag{3.45}$$

Equation 3.42 is essentially the discretized form of this same equation, where $\mathbf{\Phi_{uy}}$ can be considered to be the applied load $q(x)$, α is equivalent to the beam stiffness $E_b I_b$, and the desired solution $\widehat{\mathbf{g}}$ is equivalent to $v(x)$. We can thus think of our solution as a beam bending process under a given load, in which we restrict the deflection to minimize the energy in the system. Analogously, we can say that we determine a $\widehat{\mathbf{g}}$ that is a loose spline fit to the data. We estimate a $\widehat{\mathbf{g}}$, so that the mean squared error is minimized without sacrificing the smoothness of the transfer function, *i.e.*, $\widehat{\mathbf{g}}$ does not have significant curvature. Figure 3-6 provides an illustration to this effect, where we can think of each data point to have a spring that tries to pull the beam towards itself. The weight α (analogous to the beam stiffness), penalizes the deflection in order to reduce bending energy.

In our ID scheme, the transfer functions are ultimately determined by solving the linear system of equations represented by Equation 3.42. Therefore, our method offers an approach for identifying transfer functions, with rapid variations with frequency filtered out, by formulating the problem as a linear system that can be readily solved using conventional techniques. We do not perform numerical convolution, which is computationally intensive.

As a separate note regarding improvement of our transfer function estimates, it should be realized that by accounting for a larger number of transfer functions in the system model, *i.e.*, increasing the value of n_h, we may reduce the errors that result from modeling deficiencies. However, this may not be very expedient, because each transfer function, that needs to be identified adds n more equations to the linear system in Equation 3.42. A larger number of computations are therefore required. If n_Φ gets

very large, there are severe demands on the processor and memory. If the processor speed is slow, the calculation times become extremely long, or the system memory might run out if the storage space is inadequate. Section 3.3.4 elaborates this issue in more detail.

3.3.3 Linear System Solver

Now that our transfer function evaluation problem has essentially been reduced to a large system of linear equations, the only thing left to address is how to solve these equations efficiently.

The most common and robust method for solving linear equations is Gaussian elimination with full or partial pivoting. For a fully populated $n \times n$ matrix, where n is large, approximately $n^3/3$ operations are required to reduce it to an upper triangular matrix. Back substitution takes another $n^2/2$ operations to arrive at the final solution [11]. The $\mathbf{A}$ matrix that we form is $n_{\mathbf{\Phi}} \times n_{\mathbf{\Phi}}$. For the kind of data sets we will deal with (for rotor system identification) it is not uncommon to have a number of data points, n, at least on the order of 10^5. Now if only the fundamental and the first positive and negative harmonic transfer functions are evaluated (so that $n_h = 3$), the number of required operations will be on the order of 10^{16}. If we have larger data sets (as will quite often be the case), and wish to determine more than three transfer functions, we can see that the number of required computations will be unacceptable.

We can obtain significant savings in calculations, however, if we take advantage of the specific structure of the $\mathbf{A}$ matrix. As was discussed, $\mathbf{\Phi_{UU_s}}$ is a block diagonal matrix, whereas the $\widetilde{\mathbf{D}}^2$ matrix is tri-diagonal (with the super- and sub-diagonals spaced apart from the main diagonal). The $\widetilde{\mathbf{D}}^4$ matrix is a band diagonal matrix, because it is the square of $\widetilde{\mathbf{D}}^2$. Since $\mathbf{A}$ is the sum of $\mathbf{\Phi_{UU_s}}$ and weighted $\widetilde{\mathbf{D}}^4$ matrix, $\mathbf{A}$ is also a band diagonal matrix with d_1 sub-diagonals and d_2 super-diagonals. When the fourth difference operator is used, both d_1 and d_2 are equal to $2n_h$.

The number of calculations required to arrive at the solution of this linear system can be significantly reduced if the operations are carried out on only the diagonal band, since the rest of the matrix elements are zero. We therefore apply Gaussian elimination algorithm on the band elements only, and thus perform nd_1d_2 calculations (where $d_1 << n$ and $d_2 << n$) for row reductions and approximately nd_2 operations for back substitution. The number of required operations is thus reduced by a factor of n^2. If we analyze the same case that we did before, and that assume $n = 10^5$ and $n_h = 3$, then $nd_1d_2 = 1.2 \times 10^6$. There is a reduction in number of operations by a factor of about 10^{10}.

In Gaussian elimination, often the rows or both rows and columns of the matrix are appropriately interchanged during the evaluation steps to avoid any division by zero. This process is called *pivoting*, and is necessary in most cases for solution stability [12]. It has been shown, however, that Gaussian elimination can be carried out on a matrix without pivoting, if and only if the leading sub-matrices are all nonsingular [11]. Our matrix $\mathbf{A}$ is positive definite (due to the nature of $\mathbf{\Phi_{UU_s}}$ and $\widetilde{\mathbf{D}}^4$), and each of

its sub-matrices are also positive definite. We can therefore avoid pivoting altogether in our algorithm, thereby making significant savings in computation time.

The linear system solver gives the solution vector $\widehat{\mathbf{g}}$. After obtaining this $\widehat{\mathbf{g}}$, it is reshaped into an $n_h \times n$ matrix, $\widehat{\mathbf{G}}$. Each of the row vectors of $\widehat{\mathbf{G}}$ represent a transfer function. A detailed description of the specifics of our algorithm is given in Appendix B.

3.3.4 Computational Issues

It is desirable for the software to be optimally coded, since we require fast execution time for quick system identification that may aid in developing effective control strategies in the future. Therefore, we have taken extra care in ensuring high calculation speed. Since MATLAB is really an interpreter, it has significant overhead, so that some operations are quite time-consuming. To speed the computation, the solver routine has been coded as a *mex file* (a C-file readable by MATLAB) [14]. The band diagonal matrix $\mathbf{A}$ is sheared to form a reduced order, but nearly fully populated matrix, $\mathbf{A_s}$, with $n_{\mathbf{\Phi}}$ rows and $d_1 + d_2 + 1$ columns [12]. This shearing makes the storage more efficient, and is better suited for manipulation in the C environment. Furthermore, it was observed (after experimenting with a few data sets) that MATLAB was also slow in transforming the 3-D $\mathbf{\Phi_{UU}}$ matrix to the sparse 2-D $\mathbf{\Phi_{UUs}}$ matrix. We have therefore coded this portion in C as well.

Our routine operates on the sheared matrix $\mathbf{A_s}$, and the column vector $\mathbf{\Phi_{uy}}$, along with information about the specified number of total harmonics, n_h. As the row reductions are carried out, elements of $\mathbf{A_s}$ and $\mathbf{\Phi_{uy}}$ are replaced with their respective new elements, to conserve memory. Consequently, these matrices cannot be re-used (with different weights on the $\widetilde{\mathbf{D}}^4$ matrix, for instance) and must be recalculated.

It has also been observed that there is a considerable decrease in calculation time at any given step when the stored data/variables are reduced. When there are large data sets (on the order of several megabytes), and their manipulations through our ID scheme create even larger matrices, the system memory rapidly gets filled up. As a result, even simple operations such as addition of two matrices (to form our matrix $\mathbf{A}$, for instance) require a significant amount of time. It is therefore extremely important to free up as much memory as possible by clearing away variables that are no longer required for further operations in the scheme.

MATLAB operates best when the calculation variables fit within the computer's cache. A detailed analysis by the MathWorks Inc. has shown that the execution time of an operation may be sped up by at least a factor of 3 if the matrix sizes do not exceed the cache size. In our case however, it will be almost impossible to take advantage of this, since the typical data sets that are used for system ID purposes are usually several tens of megabytes, whereas the maximum cache size one might find these

days will not be more than a couple of megabytes at the most.

3.3.5 Summary of ID method implementation

A brief overview of the actual steps carried out in our system ID routine (implemented in MATLAB and C code) is as follows:

1. The input, output, and time data are assembled in row vectors $\mathbf{u}$, $\mathbf{y}$, and ψ respectively. Each vector has the same length n.

2. A matrix $\mathbf{U}$ of dimension $n_h \times n$ is formed, in which each row is the FFT (Fast Fourier Transform) of $\mathbf{u}$ modulated with ψ, as expressed in Equation 3.19.

3. A vector $\mathbf{Y}$, of length n, is formed by taking the FFT of $\mathbf{y}$.

4. $\mathbf{u}$, $\mathbf{y}$, and ψ are cleared to free up memory.

5. A 3-D, $n_h \times n_h \times n$ matrix, $\mathbf{\Phi_{UU}}$, is created by multiplication of $\mathbf{U}$ with its complex conjugate transpose, $\mathbf{U}^*$.

6. The array $\mathbf{\Phi_{UY}}$, of dimension $n_h \times 1 \times n$, is created by multiplying $\mathbf{U}^*$ with $\mathbf{Y}$.

7. $\mathbf{\Phi_{UY}}$ is transformed into a column vector, $\mathbf{\Phi_{uy}}$, of length $n_{\mathbf{\Phi}}$.

8. $\mathbf{\Phi_{UU}}$ is transformed into a 2-D, block diagonal, $n_{\mathbf{\Phi}} \times n_{\mathbf{\Phi}}$ matrix, $\mathbf{\Phi_{UUs}}$.

9. A matrix $\mathbf{D}$, (of dimension $n_{\mathbf{\Phi}} \times n_{\mathbf{\Phi}}$) is formed with its main diagonal set to unity and, appropriately placed, super-diagonal set to -1.

10. $\widetilde{\mathbf{D}}^2$, the second difference operator, is obtained by multiplication of matrix $\mathbf{D}$ with its transpose, $\mathbf{D}^T$.

11. $\mathbf{\Phi_{UU}}$, $\mathbf{\Phi_{UY}}$, and $\mathbf{D}$ are cleared from memory, leaving behind only $\mathbf{\Phi_{uy}}$, $\mathbf{\Phi_{UUs}}$ and $\widetilde{\mathbf{D}}^2$.

12. The matrix $\mathbf{A}$ is formed by adding $\mathbf{\Phi_{UUs}}$ and the product of $\widetilde{\mathbf{D}}^2$ with itself, weighted by a scalar α.

13. $\mathbf{\Phi_{UUs}}$ and $\widetilde{\mathbf{D}}^2$ are also cleared.

14. The sparse, band-diagonal matrix $\mathbf{A}$, with d_1 sub-diagonals and d_2 super-diagonals, is sheared to form $\mathbf{A_s}$ of dimension $n_{\mathbf{\Phi}} \times (d_1 + d_2 + 1)$.

15. Gaussian elimination and back substitution are performed on $\mathbf{A_s}$ and $\mathbf{\Phi_{uy}}$ to obtain the solution vector, $\widehat{\mathbf{g}}$, of length $n_{\mathbf{\Phi}}$.

16. $\widehat{\mathbf{g}}$ is reshaped to form the $n_h \times n$ harmonic transfer function matrix, $\widehat{\mathbf{G}}$. The row vectors of $\widehat{\mathbf{G}}$ are the desired transfer functions of the system.

Chapter 4

Signal Generation and Processing for Rotor System ID

In this chapter, we address some specific issues pertaining to suitable input signal generation and data processing for reliable system ID results. A comprehensive and accurate system characterization effort requires input test signals to have appropriate frequency content and, in the case of LTP systems, phase. Section 4.1 discusses this topic, along with a description of our implementation for helicopter rotors in more detail.

A related issue is accurate knowledge of system phase in the transfer function identification process. We require this information in order to produce modulation, of the input signal which is needed for our LTP system model. Section 4.2 presents a detailed discussion of this topic.

4.1 Input Data Requirements

As mentioned in Section 2.2, sinusoids are often used for determining transfer functions. More specifically, chirp signals (swept sine waves) are used to obtain the system response over a specific range of frequencies. The chirps may have frequencies that vary linearly, quadratically, or even logarithmically with time. The frequency content and time interval of the chirp is dependent on the system characteristics. In the case of an LTP system, it is important to take the chirp phases into consideration as well. Unlike an LTI system, the dynamic response of an LTP system is dependent on the system's phase at time of input application, in addition to the input frequency content. It is therefore imperative to ensure that during data collection, the test signals are applied with appropriate phases relative to the system period.

Note that in many instances, a system ID routine is implemented on data that has not been acquired

in accordance with the ideal guidelines for an LTP system. In the absence of such a luxury, one can still determine the time of initiation of the input signals, and if even a few test signals turn out to be appropriately placed in time (relative to the system period), our system ID method can be used. For instance, if ten chirps have been applied in an experiment without regard to their phases, and subsequent analysis shows that at least three chirps are far enough apart that $\mathbf{\Phi_{UU}}$ is well conditioned, then up to three transfer functions may be reasonably evaluated.

4.1.1 Chirp Inputs for Helicopter Rotor Systems

Since we are developing our LTP system ID methodology for immediate application to rotor systems, we will discuss some specific issues related to helicopter rotors. In Section 3.2, it was pointed out that great care needs to be taken in applying the input signals at appropriate time intervals relative to the system period. For a rotor system, it should also be remembered that the system period is not the period of the rotor, it is the blade passage period. Therefore, for a helicopter rotor with N_b blades, and rotation period T_r, the period T of the system is actually T_r/N_b. Thus, the output frequencies of an input signal at frequency ω, are shifted by positive and negative multiples of the blade passage frequency, ω_p, where

$$\omega_p = 2\pi N_b/T_r \tag{4.1}$$

Consider a three bladed rotor in collective mode, *i.e.,* all the blades have the same pitch. In order to identify three transfer functions, the first chirp should be initiated at 0 degrees azimuth, the second at 40 degrees, and the third at 80 degrees (see Figure 4-1). When Blade 1 has rotated by 120 degrees and occupies the initial position of Blade 2, the system has returned to its initial orientation, and has completed one period.

4.1.2 Chirp Generation for Helicopter Blade Excitation

We have developed Simulink models that generate linear chirps to enable system ID of rotor systems. The chirp generator that has been built (Figure 4-2) uses user-defined parameters in constructing the input signal to the blades. To specify the chirp characteristics, a user can set the initial signal frequency, f_0, final frequency, f_1, time duration of a single chirp, t_d, pause time between two successive chirps, t_p, and total number of different chirp phases, N.

The user-defined parameters are used in controlling the rotor system input signal generation. A counter keeps track of time, and produces an impulse after every chirp time period, T_c (the sum of t_d and t_p). A model block called the Azimuth Unwrapper reads in the rotor azimuth data that is in the form of a saw-tooth wave function with limits of 0 deg and 360 deg.

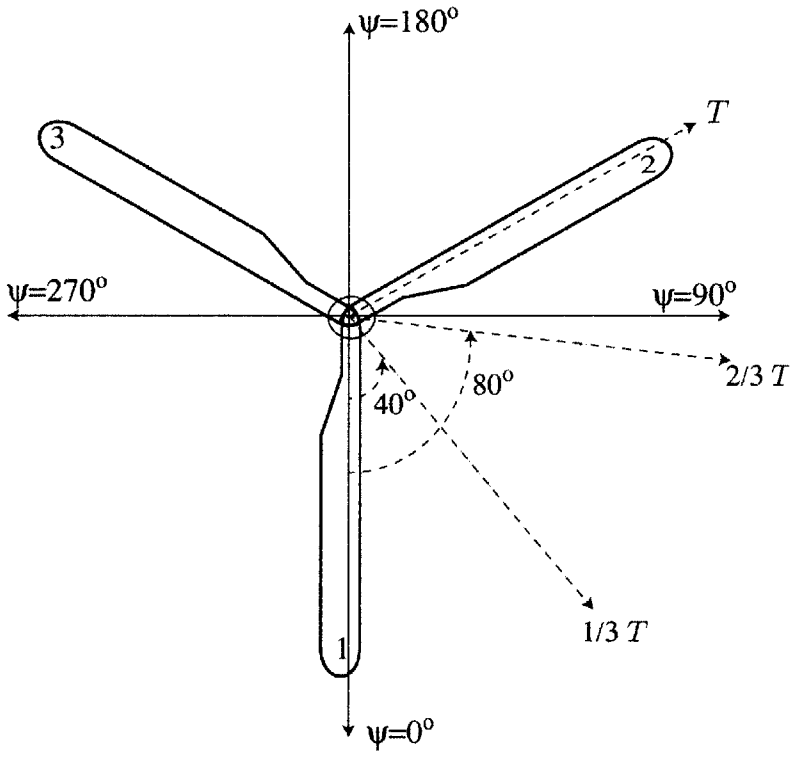

Figure 4-1: Chirp initiation at 0 deg, 40 deg, and 80 deg azimuth for a three bladed rotor in collective mode for identification of three transfer functions.

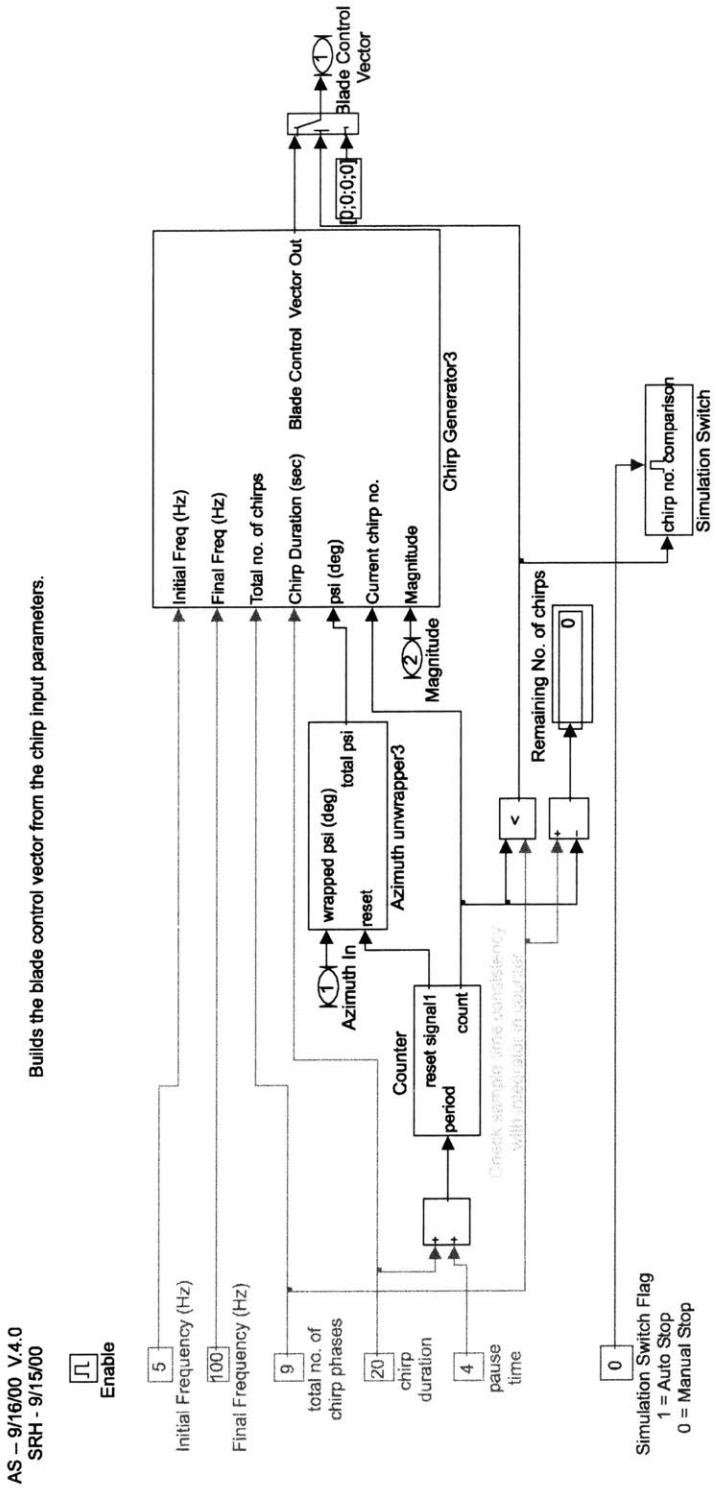

Figure 4-2: Simulink model of a chirp generator.

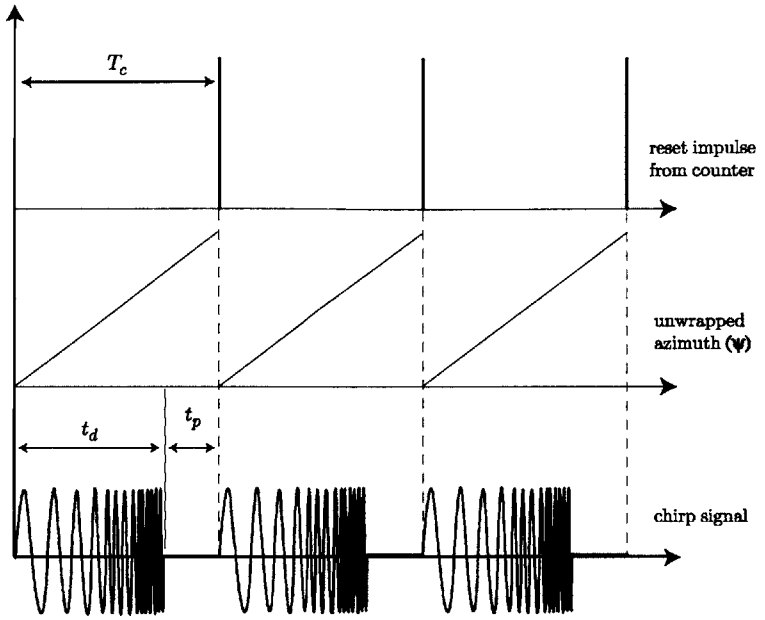

Figure 4-3: Output signals from various sub-systems of the chirp generator.

The data is unwrapped into a continuously increasing linear azimuth vector, and is reset every time an impulse from the counter is received. A chirp is initiated at each impulse and lasts for t_d seconds. This process is illustrated in Figure 4-3.

Since the chirps are linear, the frequencies are a linear function of time, given as

$$f = f_0 + \frac{f_1 - f_0}{T_c} t \tag{4.2}$$

where the frequencies are expressed in Hz. Therefore, the chirp phase ϕ_c is the integral of this frequency time function, so that

$$\phi_c(t) = (f_0 t + \frac{f_1 - f_0}{2T_c} t^2)\ 2\pi \tag{4.3}$$

where the phase is in radians. The chirp generator operates by using the unwrapped azimuth angle measurements. The relation between rotor speed Ω (rpm) and azimuth angle ψ (deg) is

$$\psi = \frac{360}{60} \Omega t \tag{4.4}$$

where time t is in seconds.

A pseudo-time, $\widehat{t}$, (in seconds) is calculated based on this relation by assuming a nominal value for the rotor speed, and using the unwrapped measurements of ψ (in degrees here). Note that in reality,

the rotor speed is not a fixed constant; however, its variations are small. The required uniform phase distribution of the N chirps over 360 deg is produced by considering the number of chirps that have already been generated n_c, and offsetting the time vector accordingly, so that

$$\widehat{t} = \left[\psi - \left\{360 + \mathrm{mod}\left(360\frac{n_c}{N}, 360\right)\right\}\right] \frac{60}{360\Omega} \tag{4.5}$$

The function "mod" is the modulo function that returns the remainder obtained from the division of its two arguments. The total chirp signal, $\mathbf{U_c}$, with magnitude A_c, for the rotor blades is formed from the relation

$$\mathbf{U_c} = A_c \sin[2\pi\phi_c(\widehat{t})]\mathbf{V} \tag{4.6}$$

where $\mathbf{V}$ is a vector of length N_b. Provisions for generating collective amplitude, differential amplitude, and sine or cosine cyclic amplitude inputs are made in forming the vector elements of $\mathbf{V}$.

4.2 Data Processing for Phase Extraction

Traditionally, the data required for identifying a transfer function is the input and corresponding output of interest from the system under analysis. As mentioned previously, the auto- and cross-spectra of the input and output are sufficient for determining the ETFE of an LTI system.

In case of a helicopter rotor, which is an LTP system, it is clear from our discussion in Section 3.2 that in addition to input and output data, we require azimuth information for our system identification method. In most cases, however, the azimuth is not determined with the same frequency as the input and output response. For instance, an encoder may be employed to measure azimuth angle at specific intervals that are greater than the sampling time intervals of input and output data. An encoder that outputs, say, 60 pulses per revolution, will give phase information after every 6 deg of angular motion. A striker (a sensor which uses magnetic induction to give out one pulse per completed rotor revolution) is also often used. Due to these less frequent phase measurements, it is generally necessary to extrapolate the required azimuth information, so that it may be used in conjunction with the input and output data for system ID purposes.

4.2.1 Phase Locked Loops

A Phase Locked Loop (PLL) is commonly used to accurately determine the phase of a signal. Since our ID scheme requires system phase measurements, a PLL can be used for obtaining phase information from the sensor data. A typical PLL consists of a phase detector, amplifier, and a voltage controlled oscillator (VCO). Figure 4-4 shows the schematic of a traditional PLL.

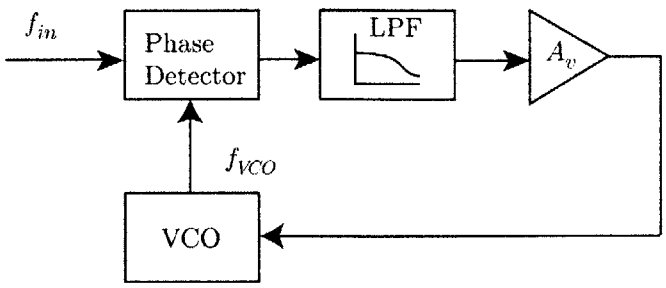

Figure 4-4: Schematic representation of a typical phase locked loop [13].

The function of the phase detector is to compare two input frequencies (f_{IN}, and f_{VCO}), and to generate an output signal proportional to their phase difference. If f_{IN} is not equal to f_{VCO}, the control voltage (which is really the filtered and amplified phase-error signal) causes the VCO frequency to deviate in the direction of f_{IN}. If the PLL is designed properly, the VCO locks onto f_{IN} very quickly and maintains a fixed phase relationship with the input signal. It is important to note that the VCO output is a locally generated frequency equal to f_{IN}, so that it actually provides a clean replica of f_{IN}, while the original signal itself may have been noisy. The VCO output signal can be of any form, so that one can generate sine waves or saw-tooth waves locked on to a train of input pulses [13].

4.2.2 Application of a PLL on Experimental Data

Continuing on with our specific focus on helicopter rotor systems, we will now discuss a PLL that has been designed in Simulink for determining azimuth angle. Several hover tests of a 1/6 scale model of a CH-47 helicopter blade were carried out at MIT in 1999 [2]. The data from those experiments was used to develop our PLL algorithm. The algorithm was then validated using a simulated system.

In the MIT tests, a striker was used to obtain rotor revolution data in the form of a train of pulses. Figure 4-5 shows a typical plot of the striker data. A pulse occurred about every 45 milliseconds, since the rotor was spun at approximately 1336 RPM. Each pulse lasted for about 3 milliseconds, giving a duty cycle of about 7%. The data was sampled at a frequency of 1000 Hz, and since the rotor went through 8 degrees of angular motion every millisecond, there was ± 4 degrees of uncertainty in time of the leading edge of the pulses.

A PLL circuit (Figure 4-6) was implemented in Simulink for determining azimuth angle of the rotor from pulse data that had been streamed out from a striker in one of the MIT tests. The PLL operates

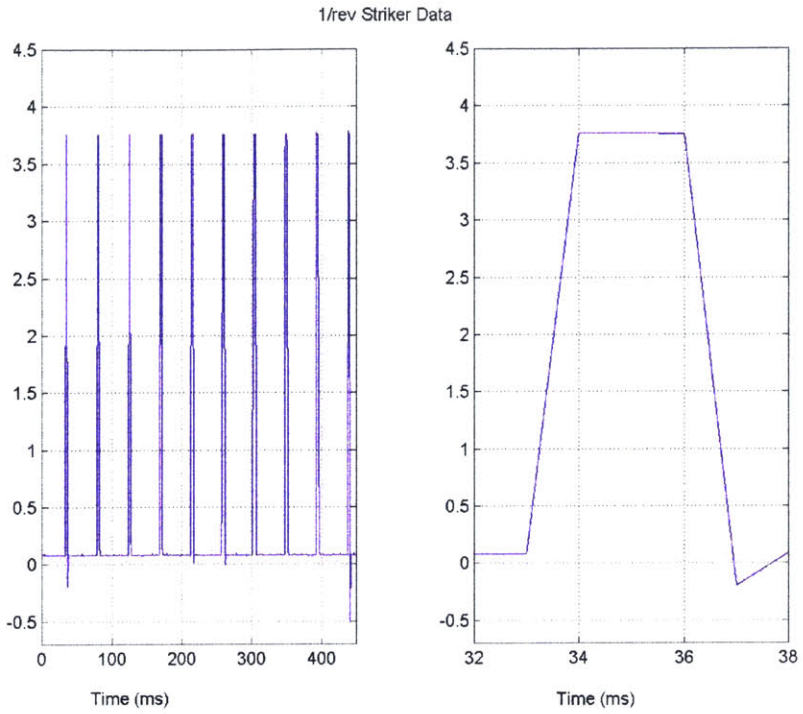

Figure 4-5: Striker data (one pulse per revolution of the rotor)

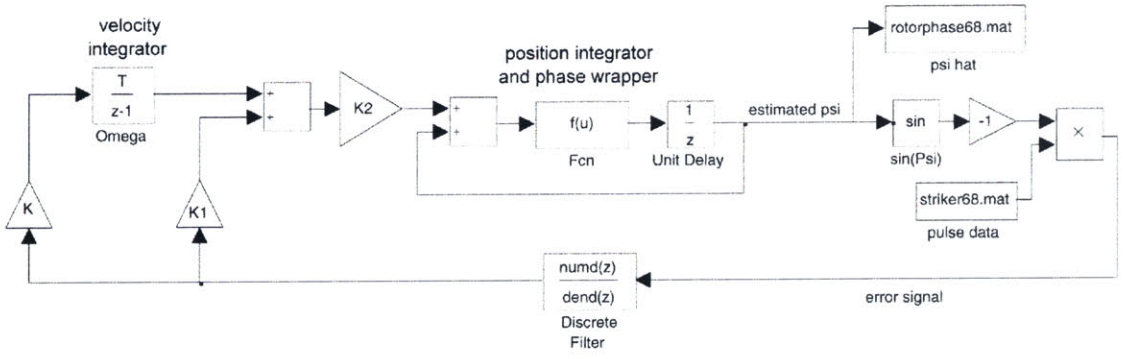

Figure 4-6: Phase Locked Loop circuit for determining rotor phase from striker data.

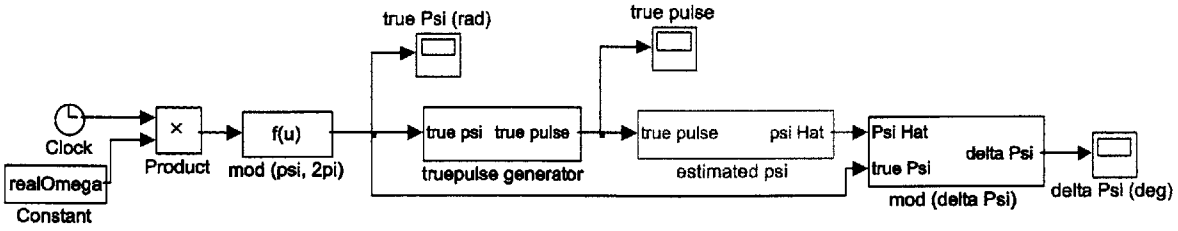

Figure 4-7: Truth model for determining difference between ψ and $\widehat{\psi}$.

by assuming an initial value of Ω, which is then integrated in discrete time to get an estimate, $\widehat{\psi}$, of the phase. The estimate is then compared with the actual striker data. The comparison is made by taking the sine of $\widehat{\psi}$, and multiplying it with the train of pulses obtained form the striker. For zero phase difference, each 1/rev pulse should be aligned with the rising zero crossing of the sine wave. However, when a phase difference does exist, the pulse will be shifted either backwards or forwards (along the time axis) relative to the sine wave. The phase estimate $\widehat{\psi}$ should then be shifted by an appropriate amount to reduce this phase difference.

To adjust the phase estimate, the error signal resulting from the phase comparison is first passed through a low pass filter for smoothing, and the resulting output is multiplied by a gain K and fed back to the velocity integrator to adjust the estimated rotor speed. Additionally, the output is multiplied by another gain K_1, and fed back to the position integrator. The gain K_1 is required to stabilize the estimation loop. Without this gain, the estimate would oscillate. The value of these gains, along with the initial assumed value of Ω, determine the transient response of the PLL.

In order to evaluate the accuracy of this Simulink PLL circuit, we developed a truth model (Figure 4-8). In this truth model, a *true value* for rotor speed was chosen, and a continuously increasing vector of ψ, using a digital clock, was generated. The ψ values were wrapped around a period of 2π, and then used to form a train of *true pulses* of appropriate duty cycles. These true pulses were fed into the Simulink PLL circuit, which in turn gave an estimate, $\widehat{\psi}$, of the azimuth. The difference between $\widehat{\psi}$ and ψ quantified the accuracy of our PLL.

In our simulations, we assumed a rotor speed of approximately 392 rpm (a time period of 153 ms), along with a sampling frequency of 2000 Hz. This low speed and high sampling frequency combined to give a good azimuth resolution of approximately 1.18 deg. The plot in Figure 4-8 shows the phase error in our PLL circuit estimate of ψ as calculated by our truth model. It can be seen that there is an overshoot of about 25 deg due to the initial conditions that were chosen in the simulation. The phase error settles down after about 4 seconds, and continues to oscillate between ± 0.4 deg. Note that the

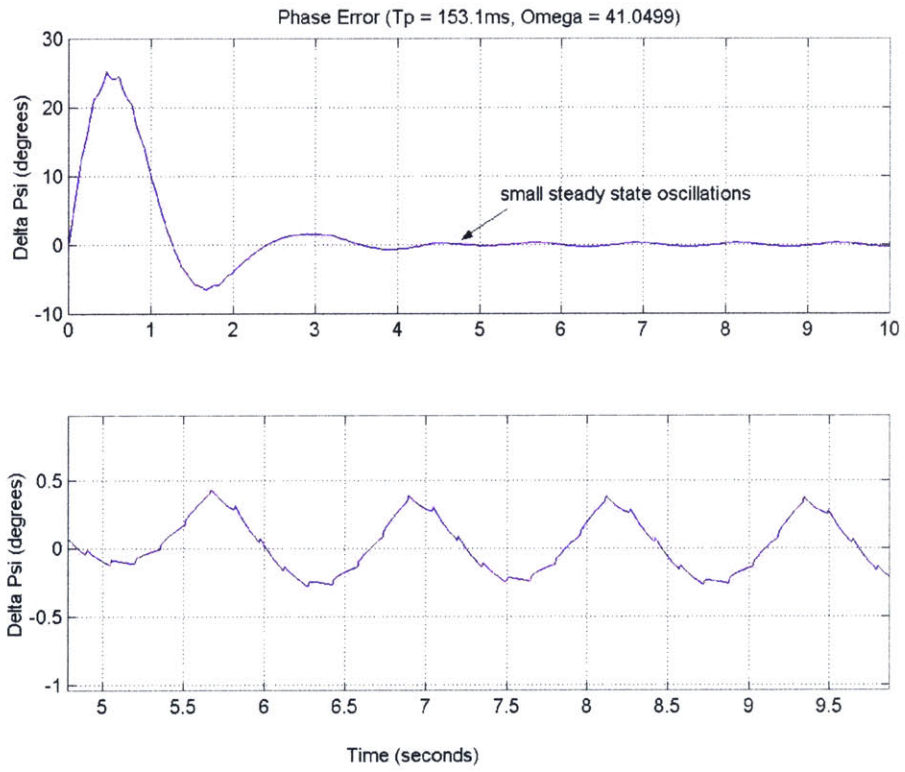

Figure 4-8: Phase error in PLL estimate, along with zoomed view of steady-state behavior.

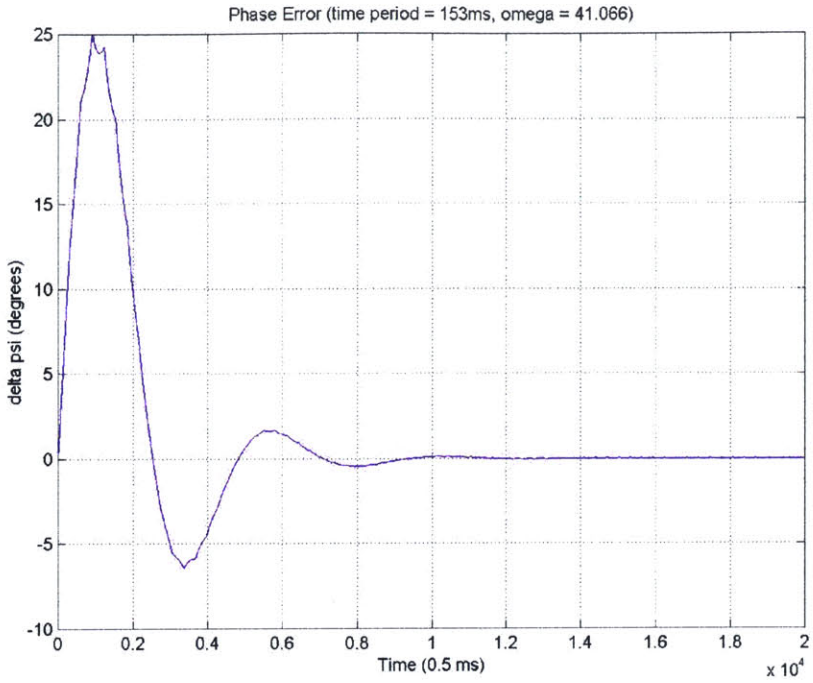

Figure 4-9: Phase error with rotor time period of 153 ms.

overshoot in the initial transient response can be altered by changing the gains, and the initial value of Ω in the PLL model. A rotor time period of 153.1 ms was used in generating this plot. Since our sampling frequency is 2 kHz, an integral number of samples do not fit in the rotor period. There are 306 samples in each rotor period, with 0.1 ms left over. Therefore, an error accumulates over several periods, and gets corrected after every few cycles when it becomes large enough to be accurately sampled. Hence the oscillatory behavior of the estimation error.

There is significant reduction in the steady-state oscillations when we make the rotor speed such that the time period is exactly covered by an integral number of samples. For a time period of 153 ms, we obtain the plot shown in Figure 4-9. In this case, the oscillations are essentially negligible, and the steady-state error is zero. Figure 4-10 shows a case where the time period is again slightly modified to be 152.9 ms. We see the same oscillatory behavior as was present in the 153.1 ms period case. Note, however, that depending on the initial conditions, the constant error can be any value within the resolution of our estimation algorithm.

In order to model extreme case scenarios, we analyzed rotor speeds with up to 20% variation, which is much larger than can occur in practice. We found that for a speed range of 313 rpm to 470 rpm, the

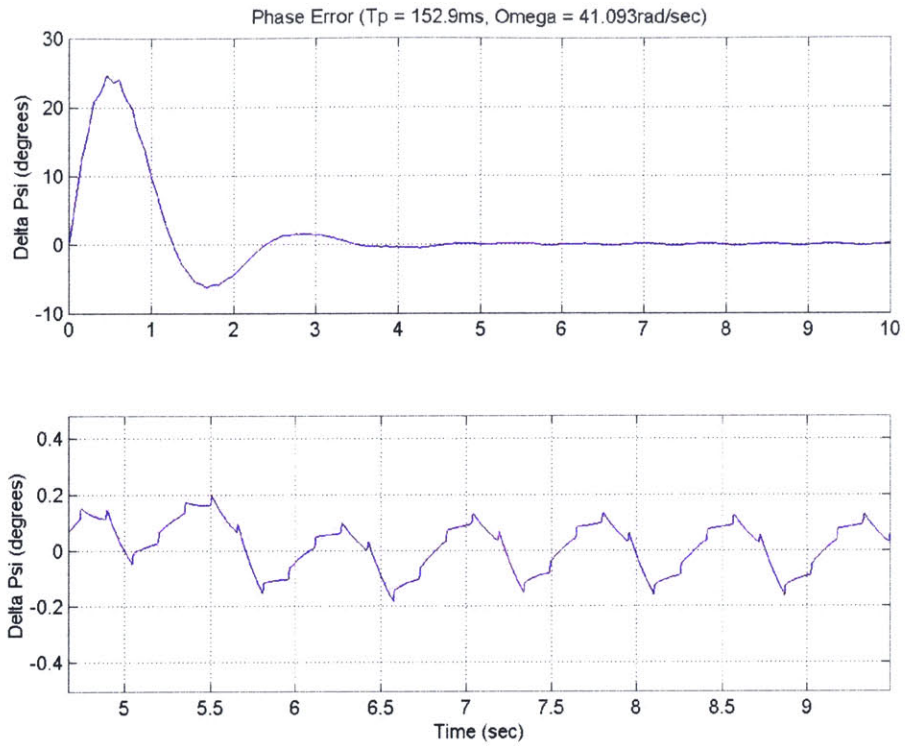

Figure 4-10: Phase error for rotor time period of 152.9 ms.

worst steady-state error behavior occurs for the 152.9 ms period, as depicted in Figure 4-10. When the time periods have integral number of samples, we have no oscillations. For other cases, the amplitude of the estimation error is usually between ± 0.2 deg and ± 0.4 deg.

Chapter 5

System ID Method Validation

At this point, we have all the necessary methodology in place to identify the harmonic transfer functions of a linear time periodic system from its input, output, and phase data. It is important to verify the accuracy of our scheme, and to establish its validity for identifying harmonic transfer functions. To achieve this goal, we will analyze a few LTP systems described by second-order differential equations. Their harmonic transfer functions will be analytically determined using Equation 2.24. These systems will then be modeled in Simulink, and their output response due to various chirp input signals will be simulated. This input and output data will then be processed by our identification routine, and the resulting harmonic transfer functions will be compared with those obtained theoretically.

5.1 LTI Oscillator

5.1.1 Theoretical Analysis

A very simple example of a system exhibiting periodic behavior is a second order LTI system, whose output is modulated with a time periodic signal. A similar case was analyzed by Wereley [4], who considered the case where the input is modulated instead of the output. We model the LTP system in block diagram form, as shown in Figure 5-1, with an input $u(t)$, and output $y_m(t)$. $G(s)$, the transfer function of the LTI portion of the system, is

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2} \tag{5.1}$$

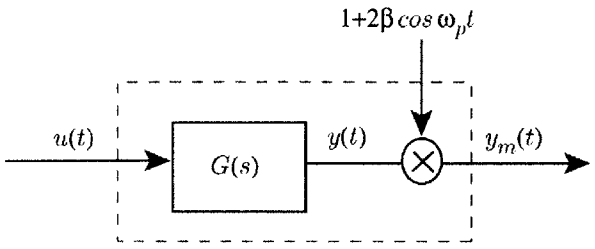

Figure 5-1: LTI oscillator with modulated output.

Note that the parameter β in the modulating signal determines the degree of modulation of the output. The state space representation of the system is

$$A = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} = A_0 \tag{5.2}$$

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = B_0 \tag{5.3}$$

All the higher coefficients of A and B are zero. The output matrix is periodic, so that

$$C(t) = (1 + 2\beta \cos \omega_p t) \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{5.4}$$

Therefore, we have that

$$C_0 = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{5.5}$$

$$C_1 = \begin{bmatrix} \beta & 0 \end{bmatrix} \tag{5.6}$$

$$C_{-1} = C_1 \tag{5.7}$$

All coefficients for the second and higher harmonics of C are also zero.

Using the above information, Equation 2.24 is employed in a MATLAB script to calculate the frequency response for this system. We used ± 10 harmonics (which is more than adequate in this case) for our state-space model formulation.

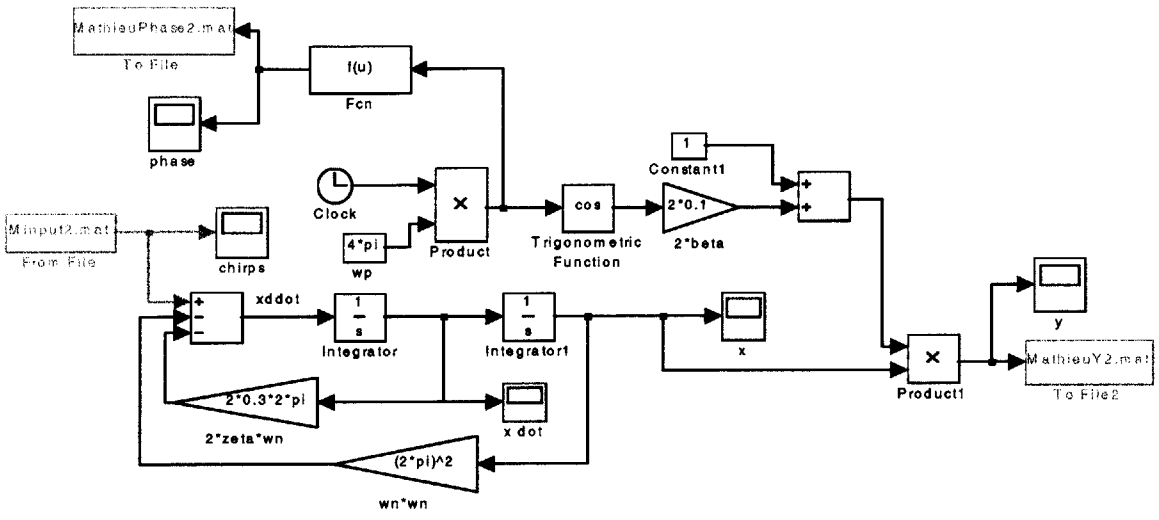

Figure 5-2: Simulink model of LTI oscillator

5.1.2 Empirical Analysis

The system was then modeled in Simulink with the block diagram shown in Figure 5-2. In addition to storing the input and output data, the phase of the system is also determined and recorded at each time step of the simulation. We used carefully designed swept sine waves, or chirps, as our test input signals. Each chirp (Figure 5-3) lasted 10 seconds, over which time the frequency varied from 0 to 10 Hz. The frequency variation was slightly nonlinear, with the instantaneous frequency given by

$$f(t) = 10\left(\frac{t}{10}\right)^{1.2} \tag{5.8}$$

The chirp signal, $u_c(t)$, is then

$$u_c(t) = \sin\left(\frac{2\pi \cdot 10 \cdot t^{2.2}}{2.2 \cdot 10^{1.2}}\right) \tag{5.9}$$

The FFT plot of the chirp, in Figure 5-3, shows that there is roughly the same amount of energy at each frequency in the band of interest.

A complete input sequence was designed to have the same number of chirps as the desired number of transfer functions to be identified. Furthermore, each chirp in the input sequence differed in phase, by equal amounts over a period. In our case, we typically used 3 chirps, with phases 0, $2\pi/3$, and $4\pi/3$ (relative to the modulation period) to identify G_0, G_1, and G_{-1} transfer functions.

Once the test signals had been appropriately designed, they were used to simulate the output of our LTI oscillator model. The system parameters used in the simulation were $\beta = 0.1$, $\zeta = 0.3$,

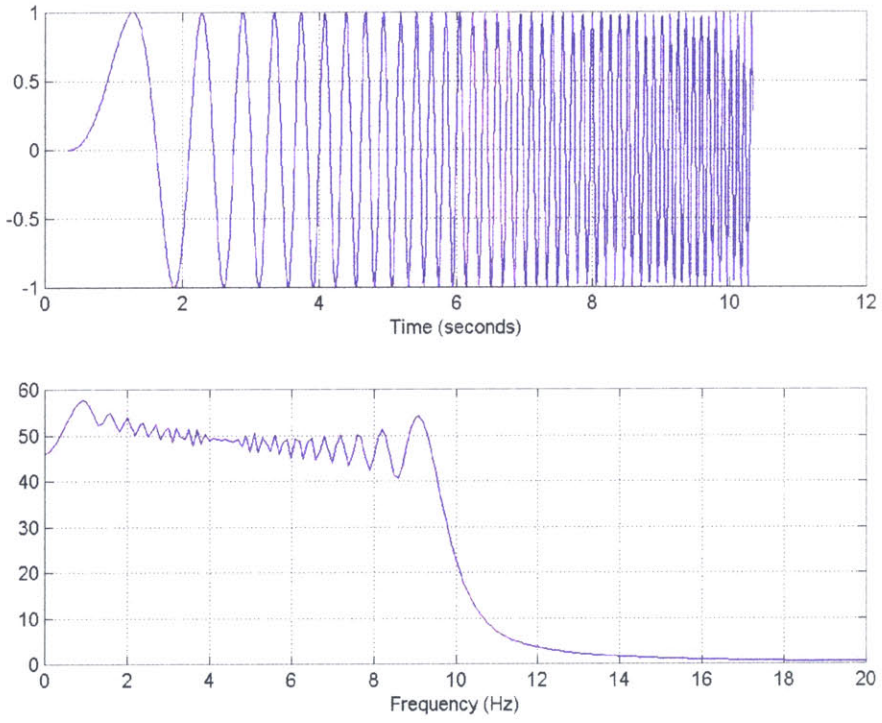

Figure 5-3: 10 second chirp with nearly uniform energy distribution over 0–10 Hz.

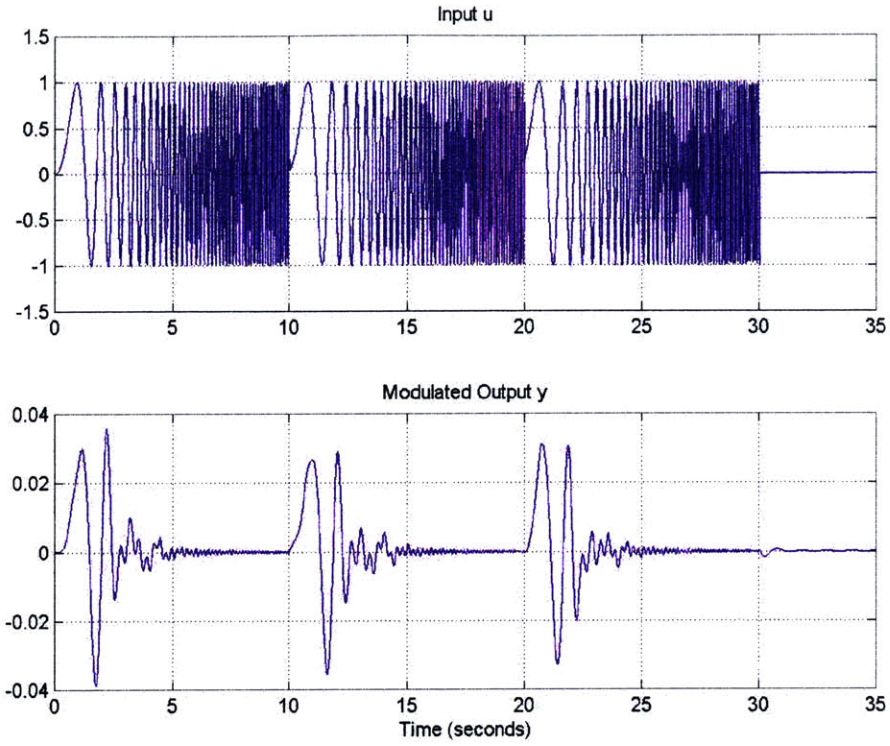

Figure 5-4: Simulated input signals and corresponding response of LTI oscillator system.

$\omega_n = 2\pi$ rad/sec, and $\omega_p = 4\pi$ rad/sec. As expected, the output corresponding to each of the chirps, differing only in phase, was slightly different in magnitude. Figure 5-4 shows the input signals and the corresponding output plot during one of our simulation runs.

The system ID software was then used on this input and output data to estimate the transfer functions of the system under study. The value of α was 10, and the fourth difference operator was used for smoothing purposes. Although there is no noise in our simulation, recall that we have modeled only three harmonics, instead of a total of 21 as were assumed in the analytical analysis. Furthermore, the set of three chirps are treated as one long input signal, and their outputs as one output signal. We therefore need to apply the least squares technique, described in Section 3.3.2, in order to have a well-defined identification problem. Figure 5-5 shows the transfer functions identified by our software, along with the analytical transfer function. As expected (in the absence of random noise), there is no discrepancy between the two, and they match each other with good precision.

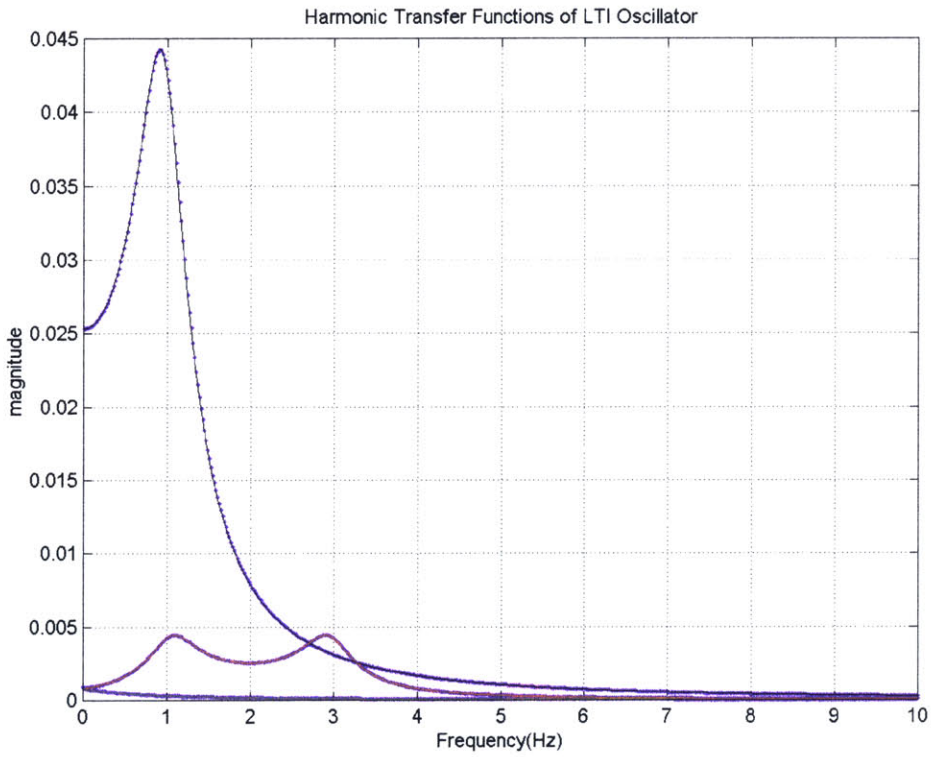

Figure 5-5: Superimposed analytical (solid line) and empirical (dotted line) transfer functions plots of LTI oscillator system.

5.2 Lossy Mathieu Equation

5.2.1 Theoretical Analysis

The Mathieu Equation is a well-known equation representing an LTP system. The canonical form of the lossy Mathieu equation that we employed for our analysis was

$$\ddot{x}(t) + 2\zeta\omega_n\dot{x}(t) + (1 + 2\beta\cos\omega_p t)\omega_n^2 x(t) = 0 \tag{5.10}$$

Again, note that the value of β determines the importance of the periodic effects, *i.e.*, for small β, the system is essentially LTI, whereas for large β, the periodic effects are significant.

In state-space, the system matrices are

$$A(t) = \begin{bmatrix} 0 & 1 \\ -(1 + 2\beta\cos\omega_p t)\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \tag{5.11}$$

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5.12}$$

$$C = \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{5.13}$$

The harmonics of the dynamics matrix, $A(t)$, are

$$A_0 = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \tag{5.14}$$

$$A_1 = \begin{bmatrix} 0 & 0 \\ -\beta\omega_n^2 & 0 \end{bmatrix} \tag{5.15}$$

$$A_{-1} = A_1 \tag{5.16}$$

Since there are no periodic elements in B and C matrices, B_0 and C_0 are the same as the matrices B and C given in Equations 5.11 and 5.12, while all the higher harmonics are zero. Using Equation 2.24, we can calculate the HTF of the Mathieu equation.

5.2.2 Empirical Analysis

The system was then modeled in Simulink, with $\beta = 0.1$, $\zeta = 0.3$, $\omega_n = 2\pi$ rad/sec, and $\omega_p = 4\pi$ rad/sec. Figure 5-6 shows our Simulink block diagram. Input signals (chirps) with appropriate phase and frequency content were again generated, and used in the simulation to obtain the output response

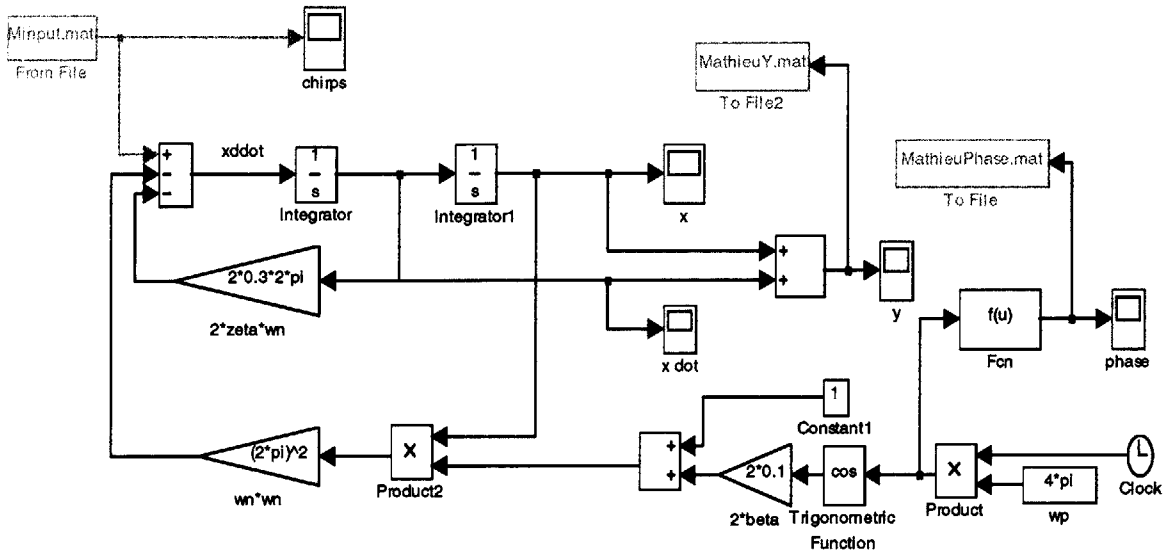

Figure 5-6: Simulink model of the lossy Mathieu equation.

of the system. Using the ID software, we obtained transfer functions of our simulated system. Figure 5-8 shows a plot of G_0, G_1, and G_{-1}, determined both analytically and empirically. The dotted lines represent the result obtained from our ID routine, while the solid lines are the analytical results. From the plots in Figure 5-9, it is evident that although G_0 and G_{-1} agree very nicely with the predictions, G_1 is far from satisfactory. However, if we increase the number of chirps in our input (to identify more transfer functions), we find that there is a significant improvement in the identification of the first few transfer functions. Figure 5-10 shows the plots obtained by using an input of 5 chirps, and estimating 5 transfer functions, *i.e.*, G_0, G_1, G_2, G_{-1}, and G_{-2} with the same weights on the difference operator as before. The phase of each of the 5 chirps was evenly spread over a 2π period. It is found that the error of G_1 has been almost eliminated. This example highlights the point made in Section 3.2.2 that we can improve our transfer function estimates by modeling for larger n_h.

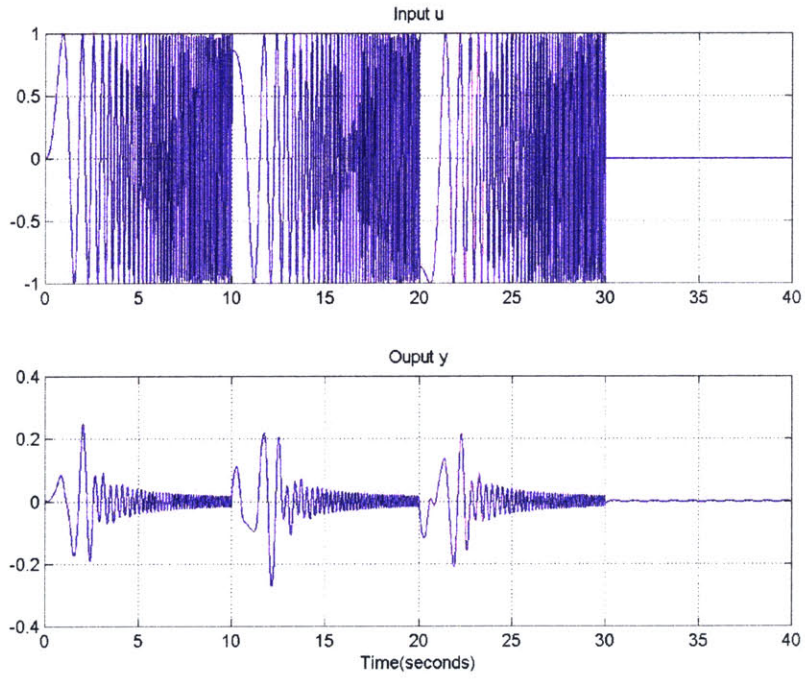

Figure 5-7: Input signals and corresponding response of the lossy Mathieu equation.

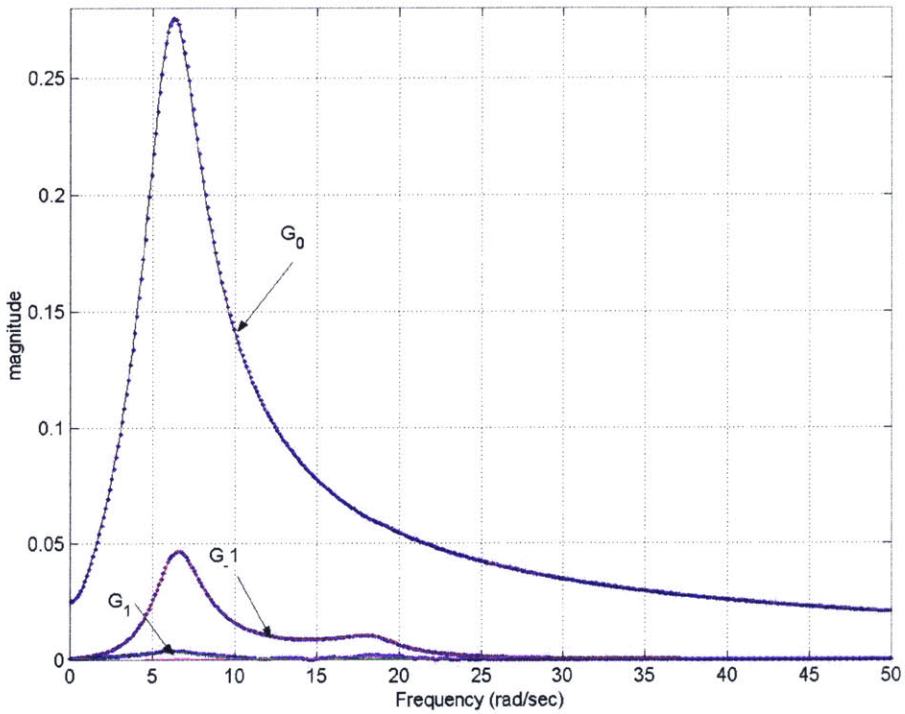

Figure 5-8: Three transfer functions of Mathieu equation obtained analytically and empirically.

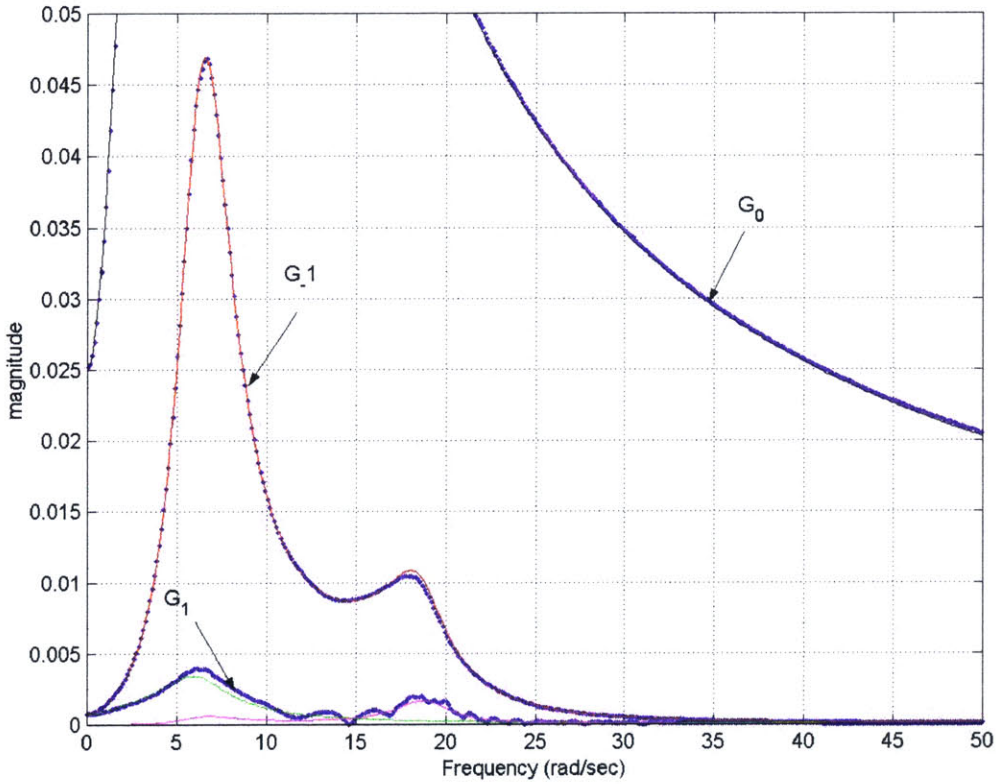

Figure 5-9: Zoomed plot of the three transfer functions of Mathieu equation.

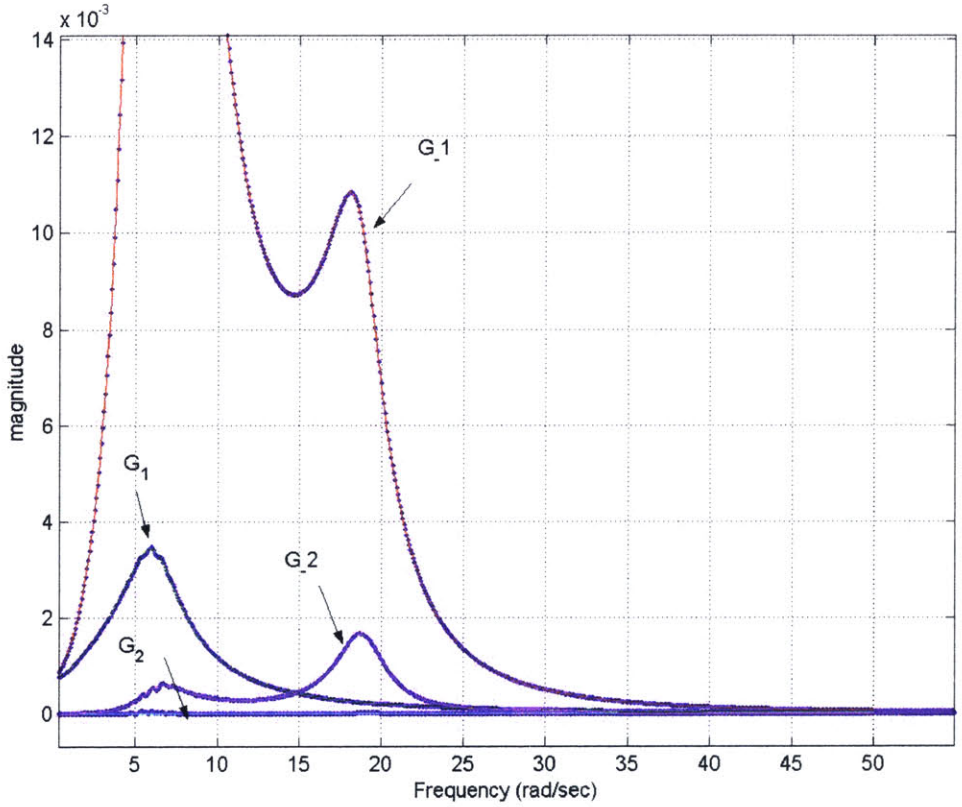

Figure 5-10: Five transfer functions of the Mathieu equation.

Chapter 6

Conclusions

In this chapter, we present a final discussion of our system ID method, and the implications for rotor control. Recommendations for further improvements are also made, along with some concluding remarks.

6.1 Summary

We have developed a transfer function identification scheme, based on the theoretical work of Wereley and Hall [6], to identify the harmonic transfer functions of helicopter rotor systems. A least squares estimation technique is used to estimate the transfer functions. The underdetermined problem is made feasible by assuming the transfer functions to be smooth. This assumption is incorporated into the ID scheme by penalizing high curvature in the estimated transfer functions with a quadratic penalty. We have implemented our ID method in MATLAB, and use a tailored Gaussian solver, written in C, to minimize computation time. Our analysis of a few LTP systems has shown good agreement between the transfer functions estimated by our identification scheme and those obtained analytically.

6.2 Implications for Controller Design

In general, a system's transfer function is determined so that appropriate control may be applied to get desirable system behavior. Our case is no different, and in fact the main objective of this thesis is the development of a sound system identification technique for LTP systems, so that effective controller designs for helicopter rotors may be developed.

For LTP systems, once the system's transfer function has been reasonably evaluated, a controller, K, can be designed based on an index m_h, where m_h is the number of harmonics in the transfer functions that are considered to be significant. Because good tools for the design of LTP controllers do not exist,

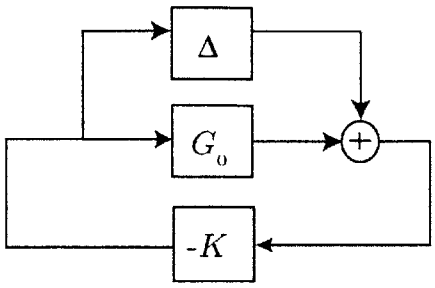

Figure 6-1: System with controller designed for only the transfer function, G_0.

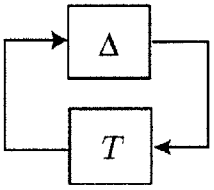

Figure 6-2: System with controller designed for only the non-trivial transfer functions.

it would be desirable to design an LTI controller, based on an LTI model of the system. For this $m_h = 1$ design, only the fundamental frequency transfer function will be treated in the analysis, while the transfer functions of the higher and lower harmonics can be lumped together in a collective uncertainty transfer function, Δ. Such a situation is illustrated in Figure 6-1.

It is worth noting that although the transfer functions at other harmonics usually will be lower in magnitude than the fundamental, there might be specific frequencies where they exceed the fundamental magnitude. It is therefore necessary to identify the transfer functions due to the harmonics and account for such situations in the controller. If we define the transfer function T as

$$T = \frac{-K}{1 + G_0 K} \tag{6.1}$$

then we can express the system as in Figure 6-2. In this form, the effect of uncertainties on the closed-loop behavior is highlighted. Using the small-gain theorem, one can conclude that the system will remain stable as long as

$$\|\Delta\| < \frac{1}{\|T\|}. \tag{6.2}$$

By designing the controller so that this condition is satisfied, an LTI controller can be designed that will be satisfactory for the LTP system.

Note that the above description of the controller design process is somewhat sketchy. To determine whether the approach would be successful, we would need a realistic LTP model of helicopter rotor dynamics, or flight test data, which is beyond the scope of this thesis.

6.3 Recommendations

In any system identification process, the most important performance metric is accuracy. When large data sets are involved, another important metric is computation speed. Since we not only deal with large data sets, but also expand them in size even further in our data arrays, it is important to consider issues related to increasing memory and computation speed. Therefore, we make a few related recommendations in this section. Additionally, we also make a few suggestions related to further research and analysis of rotor system ID and control.

6.3.1 Increase Cache Size

It has been shown through various tests by the MathWorks Inc. that the execution speed of MATLAB increases by at least a factor of 3 when the size of the variables under manipulation can fit within the computer cache. It is therefore recommended that a system with the largest possible cache allocation be used, since it can make a significant difference in computation time. Unfortunately, for our case this might not be possible, since presently the cache sizes commonly available are no greater than a couple of megabytes, whereas some of the variables generated by our scheme from typical rotor data sets are on the order of tens of megabytes.

6.3.2 Make Data Variables Re-usable

We mentioned in Section 3.3.4 that the large intermediate variables produced during the calculation process of our estimation routine are cleared from memory as often as possible to speed operation times. While this does speed up the computation process, it becomes quite tedious to perform several estimation runs using different smoothing weights, since the data sets have to be reloaded into memory each time. This situation can be remedied if the $\mathbf{A}$ and $\mathbf{D}^2$ matrices and the $\mathbf{\Phi_{UY}}$ vector are not cleared. In order to achieve this, there should be large enough RAM available for computation. We recommend the computer system should have 256 MB RAM for processing rotor data sets, and at the very least should have 128 MB RAM to get results in a reasonable amount of time (on the order of a few minutes). Note that these recommendations are only for cases when multiple harmonic transfer functions are being estimated, *i.e.*, $n_h \geq 3$. The memory requirements for LTI system ID, and for LTP system ID where only the fundamental transfer function is evaluated, are significantly less.

6.3.3 Apply ID Scheme on Rotor Data

In Chapter 5, we presented the results of our ID method that were obtained from the analysis of an LTI oscillator system and the Mathieu equation. Since the ID scheme has been primarily developed for the identification of the HTF of rotor systems, it is important to process actual rotor data through our software, and analyze its performance. Our ID scheme should first be applied on a few rotor data sets in order to test the validity of its transfer function estimates, before any final controllers are designed based on its results.

6.3.4 Perform Detailed Control Design Analysis

In Section 6.2, we have presented a brief overview of how controllers may be designed once the HTF of a rotor system has been identified. A detailed design analysis, accounting for stability and robustness should be performed, however, before any type of controller is actually implemented. Nyquist stability criterion for LTP systems [8], or H_∞ techniques can be used to determine if Δ can cause instability. Sensitivity of the controller to speed, flight path angle, and other flight parameters should also be determined in order to pick a suitable control law.

Bibliography

[1] S. R. Hall and N. M. Wereley, "Performance of Higher Harmonic Control Algorithms for Helicopter Vibration Reduction," Technical Note 4, American Institute of Aeronautics and Astronautics, July-August 1993.

[2] E. Prechtl, "Design and Implementation of a Piezoelectric Servo-Flap Actuation System for Helicopter Rotor Individual Blade Control," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, January 2000.

[3] J. A. Molusis, C. E. Hammond, and J. H. Cline, "A Unified Approach to the Optimal Design of Adaptive and Gain Scheduled Controllers to Achieve Minimum Helicopter Rotor Vibration," In 37th Annual Forum of the American Helicopter Society, New Orleans, Louisiana, May 1981.

[4] N. Wereley, "Analysis and Control of Linear Periodically Time Varying Systems," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, February 1991.

[5] J. Richards, *Analysis of Periodically Time Varying Systems.* Communications and Control Engineering Series, Springer-Verlag, 1983.

[6] N. Wereley and S. R. Hall, "Frequency Response of Linear Time Periodic Systems," In 29th IEEE Conference on Decision and Control, Honolulu, Hawaii, December 1990.

[7] J. Shaw "HHC: Wind Tunnel Demonstration of Fully Effective Vibratory Hub Force Suppression," In 41st Annual Forum of the American Helicopter Society, Fort Worth, Texas, 1985.

[8] S. R. Hall and N. M. Wereley, "Generalized Nyquist Stability Criterion for Linear Time Periodic Systems," In American Control Conference, San Diego, California, May 1990.

[9] R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, 1997.

[10] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, 1986.

[11] G. Strang, *Linear Algebra and Its Applications*, Academic Press, 1976.

[12] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992.

[13] P. Horowitz and W. Hill, *The Art of Electronics*, Cambridge University Press, 1989.

[14] *MATLAB Application Program Interface Guide*, The MathWorks Inc., 1998.

Appendix A

MATLAB Scripts

We have carried out all the data processing for system identification in MATLAB routines (also known as *m-files*). This appendix includes the m-file code we have developed for identifying harmonic transfer functions, and the m-file codes that were used for obtaining theoretical HTFs for the LTI oscillator system and the Mathieu equation.

A.1 M-file for System ID

```
%mathieuID_Csolvers.m
%This MATLAB script estimates harmonic transfer functions using
%data sets supplied by the user.  It prompts the user to choose
%the number of transfer functions to be estimated.  It uses two
%C-routines, sparseUUc.c and solver2f.c for computation.
%This particular m-file uses data generated in Simulink from
%the Mathieu equation model.
%01-31-01
clear
load Minput                 %loading input data file
load MathieuY              %loading output data file
load MathieuPhase     %loading phase data file

nharm = input('Enter no.  of harmonics:');

%time vector is stored in the first row of var psi
```

```
t= psi(1,:);
dt=t(2)-t(1);
y = y(2,:);
n=length(y);
u = u(2,1:n);
psi = psi(2,1:n);

UU=zeros(nharm,nharm,n);
UY=zeros(nharm,1,n);
Y=fft(y);

c = nharm - ceil(nharm/2);

U = zeros(nharm,n);
for i = 1:nharm
    U(i,:)=fft(exp(c*j*psi).*u);
    c = c-1;
end
clear u y t psi

for i=1:nharm
    for k=1:nharm
        UU(i,k,:)  = U(k,:).*conj(U(i,:));
    end
    UY(i,1,:)  = conj(U(i,:)).*(Y);
end

uy = UY(:);
uu = sparseUUc(UU);
clear UU UY

m = size(uu,1);
D =(sparse(1:m,1:m,ones(1,m),m,m))+...
 (sparse(1:m-nharm,(nharm+1):m,-1*ones(1,m-nharm),m,m));
DD =D'*D;
clear D
```

```
A= uu+DD*DD*10^13;
clear DD uu

%only for the case of DD*DD
m1= 2*nharm;
m2 = 2*nharm;

[a]=shear(A,m1,m2);
[x]=solver2f(a,uy,m1,m2);
figure
N = length(x);
f=((0:(N-1))/N/dt);
plot(f,abs(x),'.')
grid on
xlabel('Hz')
```

A.2 M-file for Shearing a Band Diagonal Matrix

```
%shear.m
%This code is the definition of the shear function used in
%mathieuID_Csolvers.m.  Its arguments are a band diagonal, square
%matrix (A), number of subdiagonals of the matrix (m1), and super
%diagonals (m2).  It straightens out the n x n matrix A, into a
%a fully populated m1+m2+1 x n matrix (a).

function [a]=shear(A,m1,m2)
[m,n]=size(A);

if m ~= n
 error ('Error:  Matrix A is not square')
end

a = zeros(n,m1+1+m2);
a(:,m1+1)=diag(A);

for k=1:m1
```

```
    j = length(diag(A,-k));
    a((n-j)+1:n,m1+1-k)=diag(A,-k);
end

for k=1:m2
    j=length(diag(A,k));
    a(1:j,m1+1+k)=diag(A,k);
end
```

A.3 M-file for Plotting Transfer Functions of LTI Oscillator

```
%theory_LTIosc.m
%This script calculates and plots the theoretical harmonic
%transfer functions of the LTI system with periodically
%modulated output.
NN=10; %no of harmonics
M=2*NN+1; %include plus minus all the harmonics and the zeroth.
n=2; %no.  of states in system
wn = 2*pi; %rad/sec
wp = 4*pi;
nM = n*M;
zeta = 0.3;
beta= 0.1;

%LTI Oscillator with modulated output
A0 = [0 1; -wn^2 -2*zeta*wn];
A1 = [0 0;0 0];
A_1 = A1;

B0 = [0 ; 1];
B1 = [0; 0];
B_1 = B1;

C0 = [1 0];
C1 = [beta 0];
C_1 = C1;
```

```
A = zeros(nM,nM);
N = zeros(nM,nM);
B = zeros(nM,M);
C = zeros(M,nM);
q=-NN:NN;

for i=1:M;
    A(((i-1)*n+(1:n)),((i-1)*n+(1:n)))=A0-1j*q(i)*wp*eye(n);
    B(((i-1)*n+(1:n)),i) = B0;
    C(i,((i-1)*n+(1:n))) = C0;
end
for i=1:M-1;
    C(i+1,((i-1)*n+(1:n))) = C_1;
    C(i,((i)*n+(1:n))) = C1;
end
D=zeros(M,M);
w=[0:0.1:2*pi*10];
GG=[];

for i=1:length(w)
    G=C*inv(w(i)*1j*eye(nM)-A)*B+D;
    GG=[GG; G(1+NN,:)];
end

%plotting G0,G1 and G_1
figure
f=w/(2*pi);
plot(f,abs(GG(:,11)),'k',f,abs(GG(:,10)),'r',f,abs(GG(:,12)),'g')
grid on
xlabel('Frequency(Hz)')
ylabel('magnitude')
title('G0:blue, G-1:Red, G1:Green');
```

A.4 M-file for Plotting Transfer Functions of the Mathieu Equation

```
%theory_mathieu.m
%This script calculates and plots the theoretical harmonic transfer
%functions of the Mathieu Equation.
NN=10; %no of harmonics
M=2*NN+1; %include plus minus all the harmonics and the zeroth.
n=2; %no.  of states in system
wp =4*pi;
wn = 2*pi;
nM = n*M;
zeta = 0.3;
beta= 0.1;

%Lossy Mathieu Equation...
A0 = [0 1; -1*wn^2 -2*zeta*wn];
A1 = [0 0; -beta*wn^2 0];
A_1 = A1;

B0 = [0 ; 1];
C0 = [1 1];

A = zeros(nM,nM);
N = zeros(nM,nM);
B = zeros(nM,M);
C = zeros(M,nM);
q=-NN:NN;

for i=1:M;
    A(((i-1)*n+(1:n)),((i-1)*n+(1:n)))=A0-1j*q(i)*wp*eye(n);
    B(((i-1)*n+(1:n)),i) = B0;
    C(i,((i-1)*n+(1:n))) = C0;
end

for i=1:M-1;
```

```
    A(((i-1)*n+(1:n)),((i)*n+(1:n)))=A_1;
    A(((i)*n+(1:n)),((i-1)*n+(1:n)))=A1;
end

w=[0:0.1:50];
GG=[];
D=zeros(M,M);
Gs =zeros(M,M,length(w));

for i=1:length(w)
    G=C*inv(w(i)*1j*eye(nM)-A)*B+D;
    Gs(:,:,i)=G;
    GG=[GG; G(1+NN,:)];
end

%plotting G0,G1 and G_1
figure
plot(w,abs(GG(:,11)),'b',w,abs(GG(:,10)),'r',w,abs(GG(:,12)),'g',...
 w,abs(GG(:,13)),'c',w,abs(GG(:,9)),'m',...
 w,abs(GG(:,14)),'y',w,abs(GG(:,8)),'k')
grid on
xlabel('rad/sec')
ylabel('magnitude')
title('Harmonic Transfer Functions of the Mathieu Equation')
```

A.5 M-file for Producing Chirps

```
%chirps.m
%This m-file produces chirps to use in Simulink models
%for generating data to obtain transfer function estimates.
%01-31-01

t=0:.01:10;
T = 1; %time period of system (mathieu) in seconds
nchirps = input('Enter number of chirps to be produced \n')
u1 = [];
```

```
for f = 1:nchirps
    u2=sin(10*t.^(2.2)/2.2*2*pi/10^1.2);
     t=t+T/nchirps;
    a =cat(2,u1,u2);
    u1 = a;
end
u(2,:)=[a 0*t 0*t 0*t];
n = length(u);
dt = t(2)-t(1);
u(1,:)=(0:(n-1))*dt;
save Minput u
```

Appendix B

C code

This Appendix presents a detailed description of the algorithm, and the C code used for our tailored Gaussian solver. Our solver performs Gaussian elimination on a band diagonal matrix, and subsequently does back-substitution to determine the final solution of a linear system of equations. The solver is implemented in C programming language in the form of a *mex file* (C code interfaced with MATLAB) for maximum efficiency.

The conversion of a large three dimensional array into a block diagonal matrix was found to be time consuming in MATLAB, therefore this operation was also implemented in C, and its mex file has also been included in this appendix.

B.1 Algorithm for Gaussian Solver

As mentioned in section 3.3.3, the matrix $\mathbf{A}$, is a band-diagonal matrix. We denote its element in the ith row and jth column as a_{ij}. $\mathbf{A}$ has d_1 sub-diagonals and d_2 super-diagonals. Therefore, we have that

$$a_{ij} = 0, \quad j < i - d_1 \text{ or } j > i + d_2 \tag{B.1}$$

The specifics of our Gaussian elimination algorithm are as follows. The first row is divided by a_{11} (the first main diagonal element) so that it becomes unity. This division is performed only on elements in columns 1 through d_b (where $d_b = d_2 + 1$), since the rest of the row elements are zero. This modified first row is then multiplied by a_{21}, and the product is subtracted from the second row so that the first element of the resulting modified second row, a'_{21}, becomes zero. The prime denotes the modified value of the element a_{21}. These operations, of multiplication and subtraction, are repeated with each of the a_{i1}'s and the ith rows, up to $a_{d_a\,1}$ (where $d_a = d_1 + 1$). At the end of this procedure, all the non-zero

elements of the first column get reduced to zero, except for the first element a'_{11}, which is unity. Note that the subtraction of each row needs to be carried out only from elements a_{i1} to $a_{i\ d_b}$, since these are the only non-zero elements of the row. Equivalent operations are also carried out on corresponding elements of the $\mathbf{\Phi_{uy}}$ vector (*i.e.*, from $\mathbf{\Phi}_{uy_1}$ to $\mathbf{\Phi}_{uy_{d_1+1}}$) to maintain equality. This process of row reduction is repeated for each row, with the operations being carried out on progressively changing column numbers as we traverse through the band diagonal matrix, until the $n_{\mathbf{\Phi}} - 1$ row is reached. In general, for the ith row, the first two operations are therefore

$$\begin{aligned} a'_{ij} &= \frac{a_{ij}}{a_{ii}} \\ \mathbf{\Phi}_{uy'_i} &= \frac{\mathbf{\Phi}_{uy_i}}{a_{ii}} \ , \end{aligned} \tag{B.2}$$

where $j = 1, \ldots r$. The primes denote modified values of the elements (due to previous row reduction operations) from their original values. The value of r depends on the row number i and is $i + d_1$, as long as $i \leq n_{\mathbf{\Phi}} - d_2$. When i is greater than $n_{\mathbf{\Phi}} - d_2$, then r is equal to $n_{\mathbf{\Phi}}$.

The ith row elements are then multiplied and subtracted from rows $i + 1$ to $i + d_1$, so that

$$\begin{aligned} a'_{lk} &= a_{lk} - (a_{ik} a_{li}) \\ \mathbf{\Phi}_{uy'_l} &= \mathbf{\Phi}_{uy_l} - (\mathbf{\Phi}_{uy_i} a_{li}) \ , \end{aligned} \tag{B.3}$$

where $l = i + 1, \ldots i + d_1$.

When the computations are carried out to the $n_{\mathbf{\Phi}} - 1$ row, $\mathbf{A}$ gets reduced to a banded upper triangular matrix, where all of its main diagonal elements (except for the last one) are unity.

Once the matrix $\mathbf{A}$ has been reduced, back substitution is carried out to obtain the final solution vector. This is achieved by first obtaining the last element that has already been isolated at the tip of the upper triangular matrix,

$$g_{n_{\mathbf{\Phi}}} = \frac{\mathbf{\Phi}_{uy_{n_{\mathbf{\Phi}}}}}{a_{n_{\mathbf{\Phi}}\ n_{\mathbf{\Phi}}}} \tag{B.4}$$

After determining the last element, we proceed to find the second to last, and so on, working our way backwards up to the first element. A typical backward substitution step from row $n_{\mathbf{\Phi}} - 1$ to $n_{\mathbf{\Phi}} - d_2$ is

$$g_i = \mathbf{\Phi}_{uy_i} - \sum_{j=i+1}^{n_{\mathbf{\Phi}}} a_{ij} g_j \ , \tag{B.5}$$

where $i = n_{\mathbf{\Phi}} - 1, n_{\mathbf{\Phi}} - 2, \ldots n_{\mathbf{\Phi}} - d_2$. From row $n_{\mathbf{\Phi}} - d_2 + 1$ to 1, we sum up to only the last non-zero

element in each row. Therefore,

$$g_i = \mathbf{\Phi}_{uy_i} - \sum_{j=i+1}^{i+d_2} a_{ij} g_j \ , \tag{B.6}$$

where $i = n_{\mathbf{\Phi}} - d_2 + 1, n_{\mathbf{\Phi}} - d_2, \ldots 1$.

As mentioned in Section 3.3.2, the sub-vectors of $\widehat{\mathbf{g}}$ are column vectors of $\widehat{\mathbf{G}}$ matrix. The solution vector is therefore reshaped into $n_h \times n$ matrix, $\widehat{\mathbf{G}}$, with each of its row vectors representing a transfer function.

B.2 Mex file for Gaussian Solver

```
/*solver2f.c*/
#include ''mex.h''
#include ''matrix.h''
#include <stdio.h>
/*call this function in MATLAB as :  [x]=solver2f(a,b,m1,m2) */
/*computational subroutine*/
void solver(int m1, int m2, int n, int c, double *pra, double *pia,
double *prb, double *pib, double *prx, double *pix, int isComplexa, int isComplexb)
{
    int i, j, k, l, d, rows, last, isComplexX;
    double tempr, tempi, OLDprb, OLDpra, piaValue, pibValue, pixValue, sumr, sumi;
    d = m1;
    rows = n;
    n = n-1; /*array index in C, unlike MATLAB, starts at 0 instead of 1 */
    c = c-1;
    piaValue = 0;
    pibValue = 0;
    pixValue = 0;
    tempi = 0;
    isComplexX = 0;

    /*some of the C code lines have their corresponding m-file code in comments
    above them*/

    /*row reductions*/
    for (i=0; i < n ; ++i)
        {
```

```
/*temp = a[i][d];*/
tempr = pra[i+rows*d];
if (isComplexa) tempi = pia[i+rows*d];

/*b[i]=b[i]/temp;*/
if (isComplexb) pibValue = pib[i];
OLDprb = prb[i];
prb[i] = (1/(tempr*tempr+tempi*tempi))*(prb[i]*tempr+pibValue*tempi);
if (isComplexb) pib[i] =
(1/(tempr*tempr+tempi*tempi))*(pib[i]*tempr-OLDprb*tempi);

for (l=d; l<=c; ++l)
    {
        /*a[i][l]= a[i][l]/temp;*/
        if (isComplexa) piaValue = pia[i+rows*l];
        OLDpra = pra[i+rows*l];
        pra[i+rows*l] =
        1/(tempr*tempr+tempi*tempi)*(pra[i+rows*l]*tempr+piaValue*tempi);
        if(isComplexa) pia[i+rows*l] = 1/(tempr*tempr+tempi*tempi)*
        (pia[i+rows*l]*tempr-OLDpra*tempi);
    }
k = 1;

if (i< n-(m1-1)) last = m1+i;
else last = n;
    {
        for (j=i+1; j<=last; ++j)
            {
                /*b[j] -= b[i]*a[j][d-k];*/
                if (isComplexa) piaValue = pia[j+rows*(d-k)];
                if (isComplexb) pibValue = pib[i];
                prb[j] -= prb[i]*pra[j+rows*(d-k)]-pibValue*piaValue;
                if (isComplexb) pib[j] -=
                prb[i]*piaValue+pib[i]*pra[j+rows*(d-k)];
```

```
                    /*temp = a[j][d-k];*/
                              tempr =pra[j+rows*(d-k)];
                              if (isComplexa) tempi = pia[j+rows*(d-k)];

                    for (l=d-k; l<=c-k; ++l)
                        {
                            /*a[j][l] -= a[i][l+k]*temp;*/
                            if (isComplexa) piaValue = pia[i+rows*(l+k)];
                            pra[j+rows*l] -=
                            pra[i+rows*(l+k)]*tempr-piaValue*tempi;
                            if(isComplexa) pia[j+rows*l] -=
                            pra[i+rows*(l+k)]*tempi+pia[i+rows*(l+k)]*tempr;
                        }
                    k++;
                }
            }
    }
/*back substitution*/
/*initializing vector x */
if(isComplexa || isComplexb) isComplexX =1;
/*for (j=0; j<= n; ++j) x[j]=0;*/
for (j=0; j <=n; ++j)
    {
        prx[j] = 0;
        if(isComplexX) pix[j] = 0;
    }
/* x[n] = b[n]/a[n][d];*/
if (isComplexa) piaValue = pia[n+rows*d];
if (isComplexb) pibValue = pib[n];
prx[n] = 1/(pra[n+rows*d]*pra[n+rows*d]+piaValue*piaValue)*
(prb[n]*pra[n+rows*d]+pibValue*piaValue);
if (isComplexX) pix[n] = 1/(pra[n+rows*d]*pra[n+rows*d]+piaValue*piaValue)*
(pibValue*pra[n+rows*d]-prb[n]*piaValue);
k =1;
```

```
for (i = n-1; i>= n-m2; --i)
    {
        sumr = 0;
        sumi = 0;

        for (j=1; j<= k; ++j)
            {
                if(isComplexa) piaValue = pia[i+rows*(d+j)];
                if(isComplexX) pixValue = pix[i+j];
                sumr +=pra[i+rows*(d+j)]*prx[i+j]-piaValue*pixValue;
                if(isComplexX) sumi +=
                pra[i+rows*(d+j)]*pixValue+piaValue*prx[i+j];
            }
        k++;

        /*x[i] = b[i] - sum;*/
        prx[i] = prb[i] - sumr;
        if (isComplexb) pibValue = pib[i];
        if (isComplexX) pix[i] = pibValue - sumi;
     }
for (i =n-(m2+1); i>=0; --i)
    {
        sumr = 0;
        sumi = 0;

        for (j=1; j<=m2; ++j)
            {
                /*sum += a[i][d+j]*x[i+j];*/
                if (isComplexa) piaValue = pia[i+rows*(d+j)];
                if (isComplexX) pixValue = pix[i+j];
                sumr +=pra[i+rows*(d+j)]*prx[i+j]-piaValue*pixValue;
                if (isComplexX) sumi+=
                pra[i+rows*(d+j)]*pix[i+j]+piaValue*prx[i+j];
            }
```

```
            /*x[i] = b[i] - sum;*/
            prx[i] = prb[i] - sumr;
            if (isComplexb) pibValue = pib[i];
            if (isComplexX) pix[i] = pibValue - sumi;
        }
}

/*gateway subroutine*/
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    int i,j, n, c, m1, m2, isComplexa, isComplexb;
    double *pra, *pia, *prb, *pib, *prx, *pix;

    /*check input arguments*/
    if (nrhs !=4)
    mexErrMsgTxt(''Four inputs required!
    (a matrix, a vector and two scalars)\n'');

    else if (nlhs > 1)
    mexErrMsgTxt(''Too many output arguments!'');
    else if (!mxIsNumeric(prhs[0]) || !mxIsNumeric(prhs[1]))
    mexErrMsgTxt(''Input matrix or vector is either not numeric or complex!'');

    /*Get input arguments*/
    n = mxGetM(prhs[0]);
    c = mxGetN(prhs[0]);
    isComplexa = mxIsComplex(prhs[0]);
    isComplexb = mxIsComplex(prhs[1]);
    mexPrintf(''isComplexa = %d isComplexb = %d \n'',isComplexa,isComplexb);
    m1 = mxGetScalar(prhs[2]);
    m2 = mxGetScalar(prhs[3]);

    /*create output vector x*/
    plhs[0] = mxCreateDoubleMatrix(n,1,(isComplexa ||
    isComplexb)?mxCOMPLEX:mxREAL);
```

```
    /*get pointers of matrix a and vector b*/
    pra = mxGetPr(prhs[0]);
    prb = mxGetPr(prhs[1]);
    prx = mxGetPr(plhs[0]);
    pia = mxGetPi(prhs[0]);
    pib = mxGetPi(prhs[1]);
    pix = mxGetPi(plhs[0]);

    /*call the computation C subroutine*/
    solver(m1, m2, n, c, pra, pia, prb, pib, prx, pix,
    isComplexa,isComplexb);
    return;
}
```

B.3 Mex file for Transforming a 3D Array to Matrix Form

```
/*sparseUUc.c*/
/*mex file for converting input 3-D array data into a 2D sparse block diagonal
matrix.  Input data must be either a column vector or a 3 dimensional numeric
matrix.  October 12, 2000 */
#include ''matrix.h''
#include ''mex.h''
#include <stdio.h>

/*gateway routine*/
void mexFunction (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    double *prUU, *piUU, *sr, *si;
    int i,j,k, m, index, rows, cols, length, cmplx, nzmax, nharm, *irs, *jcs,
    Ndims; const int *dims;
    Ndims = mxGetNumberOfDimensions(prhs[0]);
    dims = mxGetDimensions (prhs[0]);
    rows = dims[0];
    cols = dims[1];
```

```
/*check proper input and output*/
if (nrhs !=1)
mexErrMsgTxt(''One input required!'');
else if (nlhs > 1)
mexErrMsgTxt(''Too many output arguments!'');
else if (!mxIsNumeric(prhs[0]))
mexErrMsgTxt(''Input must be numeric!'');
else if (Ndims > 3)
mexErrMsgTxt(''Input cannot have more than three dimensions!'');
else if (Ndims ==2 && cols!=1)
mexErrMsgTxt(''Input should be a column vector!'');

/*get pointers to real and imaginary data in UU matrix*/
if (Ndims ==3 ) length = dims[2];
nzmax = rows*cols;
if (Ndims == 3) nzmax = nzmax*length;

/*number of harmonics considered will be the no.  of rows in UU */
nharm = cols;
prUU = mxGetPr(prhs[0]);
piUU = mxGetPi(prhs[0]);
cmplx = ((piUU==NULL)? 0:1);
if (Ndims <3) m = rows*cols;
else m = rows*length;
plhs[0]=mxCreateSparse(m,m,nzmax,(cmplx ?  mxCOMPLEX:mxREAL));
/*getting pointers for the output array*/
sr = mxGetPr(plhs[0]);
if(cmplx) si = mxGetPi(plhs[0]);
irs = mxGetIr(plhs[0]);
jcs = mxGetJc(plhs[0]);
/*populating the row-index, column-index and data arrays */
index = 0;
if(Ndims >2)
    {
```

```
            for (k=0; k<length;k++)
                {
                    for (j=0; j<cols;j++)
                        {
                            for(i=0; i<rows;i++)
                                {
                                    irs[index]= k*nharm+i;
                                    sr[index] = prUU[i+rows*j+rows*cols*k];
                                    if (cmplx) si[index]=piUU[i+rows*j+rows*cols*k];
                                    index++;
                                }
                        }
                }
        }
    else
        {
            for (i=0; i<rows;i++)
                {
                    irs[index]= i;
                    sr[index] = prUU[i];
                    if (cmplx) si[index]=piUU[i];
                    index++;
                }
        }
    for (j=0; j<m;j++) jcs[j]=j*nharm;
    jcs[m]=nzmax;
    mxSetJc(plhs[0], jcs);
    return;
}
```