REDUCED ORDER MODELING OF INFINITE DIMENSIONAL SYSTEMS FROM FREQUENCY RESPONSE DATA

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Okan Demir

September, 2014

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Hitay Özbay(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Ömer Morgül

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Mehmet Önder Efe

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

REDUCED ORDER MODELING OF INFINITE DIMENSIONAL SYSTEMS FROM FREQUENCY RESPONSE DATA

Okan Demir
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Hitay Özbay
September, 2014

In this thesis, a system identification method using frequency response data is studied. Identification method is applied to various types of distributed parameter systems, in particular flexible structures. One of the challenging tasks in the control of flexible structures is the estimation of the dominant modes (location of resonant frequencies and associated damping coefficients). In the literature, there are several studies where transfer functions of flexible structures are derived from PDEs (Partial Differential Equations); these are infinite dimensional models. In this study, a numerical method is proposed to identify the dominant flexible modes of a flexible structure with an input/output delay. The method uses a frequency domain approach (frequency response data) to estimate the resonating frequencies and damping coefficients of the flexible modes, as well as the amount of the time delay. A sequential NLLS (Non-Linear Least Squares) curve fitting procedure is adopted. Instead of optimizing over all available data collected on a frequency interval, a data selection scheme that increases the amount of data at each step is followed. Selecting relevant parts of data and optimizing sequentially increasing number of coefficients in every step is the essential part idea behind this approach. The optimization problem solved reduces to a curve fitting problem. It is illustrated that such a Newtonian optimization method has the capability of finding the parameters of a reduced order transfer function by minimizing a cost function involving nonlinearities such as exponential and rational terms. Further model reduction techniques can be applied by analyzing Hankel singular values of the resulting transfer function. Comparisons with other methods solving similar problems are illustrated with examples. Simulation results demonstrate efficiency of the proposed algorithm.

Keywords: Frequency response, system identification, distributed parameter systems, model reduction, time delay.

ÖZET

SONSUZ BOYUTLU SİSTEMLERİN FREKANS TEPKİSİ VERİSİNDEN İNDİRGENMİŞ DERECELİ MODELLENMESİ

Okan Demir
Elektrik ve Elektronik Mühendisği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Hitay Özbay
Eylül, 2014

Bu tezde, frekans tepkisi verisini kullanan bir sistem tanılama yöntemi üzerine çalışılmıştır. Sistem tanılama yöntemi bir grup dağıtık parametreli sistem (özel olarak esnek yapılar) üzerinde uygunlanmıştır. Esnek yapıların kontrolünü zor kılan bir nokta baskın kiplerin (rezonant frekanslarının konumu ve ilgili sönümlenme katsayıları) kestirilmesidir. Literatürde, Kısmi Diferansiyel Denklemler'den esnek sistemlerin transfer fonkisyonlarını elde eden ve esnek sistemleri sonsuz boyutlu modellerle tanımlayan pek çok çalışma bulunmaktadır. Bu tezde, girdi/çıktı gecikmesi içeren bir esnek yapının baskın esnek kiplerini tanımlayan bir sayısal yöntem önerilmiştir. Yöntem, esnek kiplerin rezonant frekansları ve sönümlenme katsayılarını, aynı zamanda zaman gecikmesinin miktarını kestirmek için bir frekans alanı yaklaşımı (frekans tepkisi verisi) kullanmaktadır. Ardışık bir Doğrusal Olmayan En Küçük Kareler (DOEKK) eğri eşleştirme işlemi benimsenmiştir. Bir frekans aralığında toplanmış tüm mevcut veri üzerinden optimizasyon yapmak yerine, her bir adımda veri sayısını artıran bir veri seçimi şeması takip edilmiştir. Verinin ilgili parçalarının seçimi ve her adımda ardışık olarak artan sayıdaki katsayıların optimize edilmesi, basit ifadeyle bir eğri eşleştirme problemi olan optimizasyon probleminin çözülmesi için esas teşkil eder. Bu tür bir Newton optimizasyon yönteminin doğrusal olmayan üstel ve oransal öğeler içeren bir maliyet fonksiyonunu küçülterek indirgenmiş dereceli bir transfer fonksiyonun katsayılarını bulabildiği gösterilmiştir. Elde edilen çok yüksek dereceli modellerin Hankel tekil değerleri incelenerek daha düşük dereceli sistemlere indirgenme özellikleri de araştırılmıştır. Ayrıca, benzer problemleri çözen diğer yöntemler uygulanıp, karşılaştırmaları sonuç olarak verilmiştir. Benzetim sonuçları önerilen algoritmanın etkinliğini ortaya koymaktadır.

Anahtar sözcükler: Frekans tepkisi, sistem tanılama, dağıtık parametreli sistem, model indirgeme, zaman gecikmesi.

Acknowledgement

I am grateful to Prof. Dr. Hitay Özbay for his supervision, encouragement and rewarding guidance throughout my graduate studies.

I am indebted to Prof. Dr. Arif Bülent Özgüler for his guidance in my research and would also like to thank Prof. Dr. Ömer Morgül and Prof. Dr. Mehmet Önder Efe for reading and commenting on the thesis.

I would like to express my gratitude to my family for their continuous support.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Aim and Scope

Control of distributed parameter systems is a important research field in control systems and has many applications in industrial area, e.g. aerospace technology, robotics, where flexible structures are modeled by Partial Differential Equations (PDEs). One of the challenging tasks in the control of flexible structures is the estimation of the dominant modes. Resonance and anti-resonance terms parameterized as resonant frequencies and associated damping coefficients need to be obtained for this purpose.

Studies on control systems and algorithms for controller design mostly cover models derived from ordinary differential equations. Adequate models can be obtained for many systems, e.g. RLC circuits, rigid robot arms. For the systems which input/output relation taken into consideration depends on more than one independent variable are expressed appropriately by PDEs. For example, a rotating beam which a torque applied to, generates angular velocity output on its rotation axis that depends on the distance to the point where the torque applied, [1], i.e. it is a distributed parameter system. Since there are more than one independent variables in such systems, dynamics are modeled by PDEs, [2].

By using the Laplace transform, infinite dimensional transfer functions are obtained from PDEs. An interesting feature of these infinite dimensional models is that they contain a few parameters, depending on the properties of the structure. Although they provide a complete abstraction of their physical properties, small variations on the modal parameters may generate large errors in the frequency response. This fact makes it difficult to develop robust control algorithms.

In order to overcome this difficulty finite order approximations must be used for these systems which are represented by transcendental functions of complex Laplace variable 's'. A finite order model having a large number of parameters that matches the system response precisely leads to a lower relative error level than a distributed parameter model whose coefficients are not estimated very accurately. As the number of states is increased in the finite order approximation, mathematical model might be expected to be close to the original transfer function. In a precise manner, a transcendental transfer function for a flexible beam can be represented as an infinite product of second order terms. Truncating high frequency modes is a method for expressing an infinite product in finite order. An adequate amount of information on the system is preserved by truncating the exact poles and zeros in a frequency band of interest.

For example, consider a free-free beam with end points at $x = 1$ and $x = -1$ and control input is a moment $m(t)$ applied to the middle of the beam. The dynamics of the beam are described by

$$\frac{\partial^2 w}{\partial x^2} + \epsilon \frac{\partial^5 w}{\partial x^4 \partial t} + \frac{\partial^4 w}{\partial x^4} = \delta(0) f(t) - \delta_x(0) m(t) \tag{1.1}$$

where $w(x,t)$ is the deflection along the beam and $f(t)$ is a point normal force. If output is selected as the deflection of the middle of the beam, Laplace transform of the (1.1) results to an in-homogeneous differential equation having independent variable 's'. By solving the resulting differential equation, transfer function is obtained as

$$G(s) = \frac{\beta}{2s^2} \left(\frac{1 + \cosh\beta \cos\beta}{\cosh\beta \sin\beta + \sinh\beta \cos\beta} \right) \tag{1.2}$$

where $\beta^4 = \frac{-s^2}{(1+\epsilon s)}$, [3].

This transfer function can be expanded into a infinite product of second order

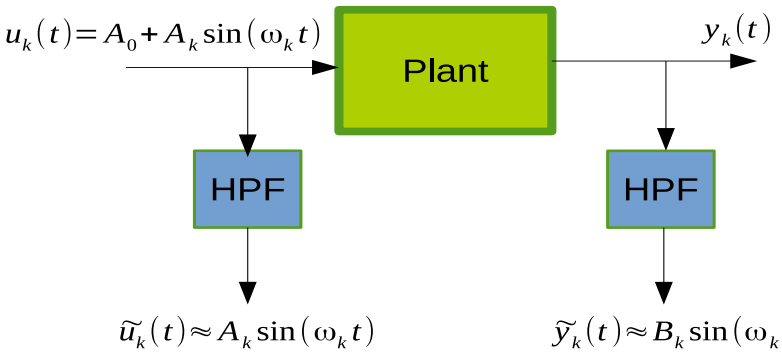

Figure 1.1: *Experimental procedure conducted on rigid robot arm to collect input/output data.*

terms:

$$G(s) = \frac{1}{2s^2} \prod_{n=1}^{\infty} \frac{\left(1 + \epsilon s + \frac{s^2}{c_n^4}\right)}{\left(1 + \epsilon s + \frac{s^2}{\mu_n^4}\right)}$$

which has zeros at $s = \frac{-c_n^4}{2}\left(\epsilon \pm \sqrt{\epsilon^2 - \frac{4}{c_n^4}}\right)$ and poles at $s = \frac{-\mu_n^4}{2}\left(\epsilon \pm \sqrt{\epsilon^2 - \frac{4}{\mu_n^4}}\right)$.

In general, modeling the dynamics of an arbitrary system by differential equations is not always possible. Obtaining an idealized physical equation as done for the free-free beam example above might be difficult to evaluate or an idealized model can not give sufficient information on the real system. Some external effects may not be included in the mathematical model. Robust control of these systems by using frequency response requires input/output relationship at the frequency band of interest. Characteristics of the system can be accessed by conducting experiments. Critical steps are input design and applying a parametric identification method on the collected data.

This approach can be considered as 'black box' modeling. For this purpose, several experiments that simulates systems process in different types of possible operating conditions can be designed and implemented. During experimental process, input and output data are recorded. Comparison between output and corresponding input signal demonstrates system behavior for inputs having different frequencies. Figure 1.1 shows a representation of data collection procedure. This procedure is conducted on a flexible, rigid robot arm for the purpose of extracting an approximating transfer function. Inputs are selected as sinusioids

having different frequencies denoted by ω_k; $y_k(t)$ is the velocity output of the system on which torque $u_k(t)$ is applied. By removing DC term A_0 from $u_k(t)$ using a high pass filter, a sinusoidal signal with additional noise is handled by the non-parametric identification method. The DC term is added in order to overcome possible nonlinear effects, such as friction.

When a sinusoidal input at a constant frequency is applied to a linear system, it produces a sinusoidal output at the same frequency but having a different magnitude and phase. If difference of phases and ratio of magnitudes between input and output signal is calculated, an estimation of systems response at the corresponding frequency can be made.

Comparison of input and output signals can be evaluated in time domain. Peak points of sinusoidal input signal and related output signal at steady state are compared in the sense of magnitude change and time shift. But this method may produce erroneous results that are caused by distorted output signal. Another option is to transform all data to Fourier domain; DFT (Discrete Fourier Transform) is used for this purpose. DFT at specified frequencies is applied separately to data sets containing sinusoids having different frequencies. Ratio of DFT of output and input sinusoids results to frequency response at a frequency point, [4]. Frequency domain data makes it available to select a parametric model that approximates 'black box' system. In [5], optimal inputs for experiment design are characterized by a sum of sinusoids. A method for selecting optimal parameters of sinusoids is proposed to excite system. Also, [6] investigates selecting optimal inputs with a constraint on the energy of input.

1.2 Literature Survey

System identification methods deal with two groups of data:

- time domain data,
- frequency domain data.

In [7] both time domain and frequency domain data sets are handled; this book contains detailed study of various system identification techniques. A set of models (e.g., AR, ARMA, ARMAX) expressed by linear difference equations are defined. Furthermore, state space models are investigated. Non-parametric and parametric system identification methods are handled. Least squares and maximum likelihood are used as solutions for identifying parameters from time and frequency response data.

Time domain data based approaches have great priority in real time modeling, especially for adaptive control. Time domain data can be collected real time and easily used to update parameters by low cost linear least squares calculations. Adaptive control algorithms use least squares calculations as a key element for on-line determination of parameters. Parameter count is the decisive factor on the order of the resulting transfer function. Model structure, experimental procedure and parameter estimation method must be selected logically.

Particularly, [8] focuses on on-line estimation of parameters for adjusting control oriented parameters dynamically. Parameters are distributed linearly in the model to simplify calculations for identification. Input which are used in experiments is based on some knowledge of the process and selected carefully. Recursive estimation of parameters is also investigated. Transfer function models are selected as FIR or ARMA models, see for example [9]. Robust control oriented identification is handled in [10]. 'Unfalsification' arises as a new paradigm in system identification area by directing identification algorithms to make a modification in order to be compatible with robust control design. Main point is such that "Given some data, a model is said to be validated if and only if it could have produced the data.", [11]. Model unfalsification is defined as a feasibility problem. Study uses FIR and ARX models as transfer function structure and linear least squares solution methods are also proposed to determine parameters. Furthermore, uncertainty 'w' is embedded into model and a bound on uncertainty is introduced to prove a system is unfalsified. In [12], definition of systems validation is given and uncertainty model unfalsification is adopted to closed loop systems in the presence of noise, see also [13].

In all the studies discussed above, constraints are not defined on the parameter set in the least squares solutions. This may lead to a transfer function with properties which are inconsistent with the real system. For example when a flexible system has collocated actuators and sensors its transfer function is minimum phase, i.e. it has no right-half plane poles and zeros.

Frequency response based methods deal with experimental data which are difficult to evaluate in real time, since obtaining frequency response points may need to excite system for a large period of time. On the other hand, these methods are formulated to represent an infinite dimensional system with finite number of states. Thus, obtaining frequency response from analytic formula of the transfer function of distributed parameter system is also a viable approach.

For instance, [14] investigates approximating a given infinite dimensional transfer function by a finite order transfer function by minimizing infinity norm of the error. Infinite dimensional transfer function is assumed to be analytic in the right-half plane. A Fourier transform based approach is used in order to determine Fourier series, Fast Fourier Transform is preferred for efficiency of computations. Resulting large number of coefficients lead to a very high order transfer function after bilinear transform applied to FIR transfer function parameterized by time domain data. Thus, a model reduction method is applied. Furthermore a bound on error is investigated. A subspace based approach is proposed in [15]. State space matrices are obtained from observability matrix by following a method based on Ho and Kalman realization algorithm. Their method deals with uniformly spaced frequency response data and Hankel matrix coefficients are obtained by Inverse Discrete Fourier Transform. Noisy data is also handled and number of data is a key point to suppress noise to converge to correct transfer function. Non-uniformly spaced data case is also investigated. Additionally, in [16] subspace based algorithm is applied to spectral data $S_k = G^*(\omega_k)G(\omega_k)$ in order to obtain a minimum phase transfer function. Technique has similarities with the method used in the second algorithm of [15]. Non-positive definiteness problem rises at the point where B and D matrices are extracted by solving a least squares problem followed by a Riccati equation. However first two methods cannot be used for simultaneous estimation of delay term and minimum phase part;

and it may result in a non-minimum phase transfer function. In [16], a convex optimization procedure is required in order to make Riccati equation solvable.

In [17], three identification algorithms, Sanathanan and Koerner algorithm, Levenberg-Marquardt method and the two-stage nonlinear algorithm, are compared. Advantages and disadvantages are shown for three examples in discrete time domain. Connections between frequency and time domain techniques are discussed in [18]. Parseval's theorem shows that minimization in time domain is analogous to minimization over frequency response data. Advantages of frequency domain approach for certain cases are noticed. In [19], authors deal with Hammersytein model identification based on frequency response data. Non-linear effects are also handled. An experimental procedure, which uses sinusoidal inputs and collects data by using the Hammerstein model, is adopted. Linear and nonlinear parts are identified separately from filtered output data. Robust control oriented system identification is also the subject of [20]. Identified model is structured by linear combination of bases from Laguerre functions. Likewise, experimental frequency response data is used for non-parametric identification and developing a model by using Chebyshev polynomials are investigated in [21]. In [22], an iterative scheme is proposed to adjust plant and controller parameters sequentially for robust control. Minimization over parameters are made for parameters of the closed loop system. Consistency of an open-loop model are investigated in [23]. Model validation problem is determined by the relation of uncertainty in new experiments and a predetermined uncertainty bound. This method checks if a model satisfies an uncertainty bound when different inputs are applied, which may lead to selecting a larger uncertainty bound. Similar to time domain methods, [24] uses rational functions of polynomials as models for curve fitting. Same method can be adopted to frequency response data by separating real and imaginary parts.

1.3 Overview of the Proposed Method

In all mentioned methods, constraints on resulting model are not considered to be a priority. Aim of this study is to determine a parametric model from obtained frequency response data. Furthermore, resulting parametric model must have a frequency response that leads to a small relative error but also must satisfy known properties of the system. Thus, attribute 'black box' may not fully define the system to be identified; 'grey box' might be a more legitimate term.

By using well-known properties of the system, a 'true' transfer function is assumed to be in the form $G(s)$ or $G_0(s)$ given below:

$$G(s) = \frac{K_0}{s} G_0(s) \quad \text{and} \quad G_0(s) = P(s)e^{-hs} \tag{1.3}$$

where h is the effective time delay, $P(s)$ is the minimum-phase part and if the integral action is present K_0 is the associated gain. The goal is to find estimated values of the parameters h and K_0 and a reduced order minimum-phase transfer function $P_N(s)$ approximating $P(s)$.

$$P(s) = \prod_{k=0}^{\infty} \frac{\left(\frac{s^2}{\omega_{k,n}^2} + \frac{2\zeta_{k,n}s}{\omega_{k,n}} + 1\right)}{\left(\frac{s^2}{\omega_{k,d}^2} + \frac{2\zeta_{k,d}s}{\omega_{k,d}} + 1\right)} \tag{1.4}$$

For many flexible systems when actuator and sensors are collocated, $P(s)$ turns out to be minimum-phase. On the other hand, non-collocated actuator and sensors lead to a transfer function having zeros on the right half plane. Nevertheless, it is possible to separate this zeros and approximate them by a single lumped delay e^{-hs}, see for example [25] where high frequency dynamics caused by elasticity, non-collocated actuator and sensors, effects of computer and zero hold are also approximated by a time delay, [26].

Since minimum phase is a requirement for the solution, constraints need to be involved in the mathematical representation of the minimization problem. When the estimate of $P(s)$ is defined as

$$P_N(s) = \prod_{k=1}^{N} \frac{a_k s^2 + 2\zeta_{k,n}\omega_{k,n}s + \omega_{k,n}^2}{b_k s^2 + 2\zeta_{k,d}\omega_{k,d}s + \omega_{k,d}^2}, \tag{1.5}$$

in order to have all poles and zeros on the left half plane, this requirement can be embedded into the optimization problem as a positivity constraint on the coefficients a_k, b_k, $\zeta_{k,n}$, $\zeta_{k,d}$, $\omega_{k,n}$, $\omega_{k,d}$ for $k = 1, \ldots, N$, and h. If integral action is present, a positive K_0 must be involved in calculations. If these coefficients are collected in one vector β for $k = 1, \ldots, N$, constraint can be expressed as

$$\underline{\beta} \succeq 0$$

where '$\succeq$' means element-wise inequality of a vector and

$$\underline{\beta} = [h, K_0, \underline{\theta}_1^T, \underline{\theta}_2^T, \ldots, \underline{\theta}_N^T]^T \tag{1.6}$$

where $\underline{\theta}_k$ is

$$\underline{\theta}_k = [a_k, \zeta_{k,n}, \omega_{k,n}, b_k, \zeta_{k,d}, \omega_{k,d}]^T.$$

In this study proposed algorithm uses log-barrier method to satisfy positivity constraint. Log-barrier method has similarities with the solution of a dual problem in non-linear optimization, [27] and gives efficient results for most cases despite its simpler structure.

In addition, an alternative basis function is investigated. Proposed algorithm can be modified to a product of first order terms instead of second order ones. Alternative model, simulation results and comparisons are also taken into scope of this study. Obtained transfer functions are further reducible, and model reduction properties are also handled.

This thesis is based on our earlier publications, [28, 29, 30]. The thesis is organized as follows. In Chapter 2, structure of the transfer functions considered are discussed in detail, and related optimization problem is expressed. Preliminaries of Newton's methods are given and methods applied to solve the optimization problem are also investigated in Chapter 2 and furthermore, PDE based models and non-parametric identification method are handled. In Chapter 3, details of the NLLS method are investigated and proposed algorithm is represented step by step. Chapter 4 is the part where simulation result are given for several examples. Concluding remarks are made in Chapter 5.

Chapter 2

PROBLEM DEFINITION

2.1 System Structure

In this thesis a numerical method for extracting a finite order transfer function followed by a delay term from frequency response data of an infinite dimensional system's transfer function is investigated. Frequency response data can be collected by conducting experiments on the system or obtained from infinite dimensional transfer functions. Infinite dimensional transfer functions which are taken into scope of this study are derived from PDEs and in a form of transcendental functions of Laplace variable 's'. They can be expanded to a infinite product of second order terms. Models in the scope of this study are assumed to be in a general form

$$G(s) = e^{-hs}\frac{K_0}{s}P(s) \quad \text{or} \quad G_0(s) = e^{-hs}P(s) \tag{2.1}$$

where the minimum phase part of the transfer function $P(s)$ is expanded to

$$P(s) = \prod_{k=1}^{\infty} \frac{(s/\omega_{k,n})^2 + 2\zeta_{k,n}(s/\omega_{k,n}) + 1}{(s/\omega_{k,d})^2 + 2\zeta_{k,d}(s/\omega_{k,d}) + 1}.$$

and $h \geq 0$ is the effective time delay. If there is an integral action is present, K_0 is the associated gain. $\omega_{k,n}$ and $\omega_{k,d}$ are natural frequencies of resonance and anti-resonance terms and $\zeta_{k,n}$, $\zeta_{k,d}$ are corresponding damping coefficients.

Frequency response at distinct frequencies are given by $\Phi_i = G(j\omega_i)$ or $\Psi_i = G_0(j\omega_i)$ at frequency points ω_i, for $i = 1, \ldots, M$. After collecting frequency response data in a frequency region of interest, following procedure can be considered as solving a complex curve fitting problem in basic terms. This is a minimization problem and literature has many techniques to solve this optimization objective. In this study, an algorithm using Non-Linear Least Squares (NLLS) solution is preferred. Although NLLS iterations have a big cost of time and computation expense compared to linear least squares solutions, better results are obtained with resulting transfer functions having lower orders.

The goal is to estimate an approximating transfer function $G_N \cong G$, or $G_N \cong G_o$ which is defined as

$$G_N(s) = e^{-hs}\frac{K_0}{s}P_N(s) \quad \text{or} \quad G_N(s) = e^{-hs}P_N(s) \tag{2.2}$$

$$P_N(s) = \prod_{k=1}^{N} \frac{(a_k s^2 + 2\zeta_{k,n}\omega_{k,n}s + \omega_{k,n}^2)}{(b_k s^2 + 2\zeta_{k,d}\omega_{k,d}s + \omega_{k,d}^2)}. \tag{2.3}$$

The terms a_k, b_k are added in order to adjust the low frequency gain of P_N. As it can be seen $P_N(s)$ is a product of second order terms. Second order transfer functions in the product is selected as a basis to approximate resonance and anti-resonance terms. Non-linear terms like exponential term and ratio of polynomials make NLLS solution a preferred method instead of linearizing the problem in parameters. Non-linear parameter search techniques give the opportunity of adding constraints on the parameters and give results that minimizes the error by a lower order transfer function. This fact lessen importance of the cost of large number of iterations compared to its results.

The objective is to minimize the relative error between a set of given frequency response data points Φ_i and $G_N(j\omega_i)$. Precisely, the constrained optimization problem is defined as

$$\begin{aligned} \underset{\underline{\beta}}{\text{minimize}} \quad & \epsilon(\underline{\beta}) = \sum_{i=1}^{M} \left| \frac{\Phi_i - G_N(j\omega_i, \underline{\beta})}{G_N(j\omega_i, \underline{\beta})} \right|^2 \\ \text{subject to} \quad & \underline{\beta} \succeq 0, \end{aligned} \tag{2.4}$$

where '$\succeq$' means element-wise inequality of a vector, and the parameter vector $\underline{\beta}$

is defined as

$$\underline{\theta}_k = [a_k \ \zeta_{k,n} \ \omega_{k,n} \ b_k \ \zeta_{k,d} \ \omega_{k,d}]^T \tag{2.5}$$

$$\underline{\beta} = \left[h, K_0, \underline{\theta}_1^T, \underline{\theta}_2^T, \ldots, \underline{\theta}_N^T\right]^T \tag{2.6}$$

where the count of items in vector $\underline{\beta}$ is denoted by K which will be used in sequent sections.

Solution of the constrained optimization problem given in (2.4) must obey the non-negativity constraint beside finding minimizing coefficients distributed in the objective function non-linearly.

2.2 Preliminaries

Newtonian optimization methods give a base for solving one equation or multiple equations of multiple unknown parameters. If objective function is linear in parameters, Newton's Method solves the problem in one iteration. Nevertheless iterations last more than one step due to non-linearly distributed parameters in objective function. If initialization point of parameters is not selected reasonably, iterations may never converge to a minimum. When objective function is denoted by 'f', objective is,

$$\text{minimize} \quad f(\underline{\beta}), \tag{2.7}$$

whose minimizing Newton step is

$$\Delta\underline{\beta} = -\nabla^2 f(\underline{\beta})^{-1} \nabla f(\underline{\beta}). \tag{2.8}$$

where

$$\nabla f(\underline{\beta}) = \begin{bmatrix} \frac{\partial f(\underline{\beta})}{\partial \beta_1} \\ \frac{\partial f(\underline{\beta})}{\partial \beta_2} \\ \vdots \\ \frac{\partial f(\underline{\beta})}{\partial \beta_K} \end{bmatrix}$$

$$\nabla^2 f(\underline{\beta}) = \text{and} \quad \begin{bmatrix} \frac{\partial^2 f(\underline{\beta})}{\partial \beta_1^2} & \frac{\partial^2 f(\underline{\beta})}{\partial \beta_1 \partial \beta_2} & \cdots & \frac{\partial^2 f(\underline{\beta})}{\partial \beta_1 \partial \beta_K} \\ \frac{\partial^2 f(\underline{\beta})}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 f(\underline{\beta})}{\partial \beta_2^2} & \cdots & \frac{\partial^2 f(\underline{\beta})}{\partial \beta_2 \partial \beta_K} \\ \vdots & \ddots & & \vdots \\ \frac{\partial^2 f(\underline{\beta})}{\partial \beta_K \partial \beta_1} & \frac{\partial^2 f(\underline{\beta})}{\partial \beta_K \partial \beta_2} & \cdots & \frac{\partial^2 f(\underline{\beta})}{\partial \beta_K^2} \end{bmatrix}$$

Equation (2.8) is derived from second-order approximation of the variated version $\hat{f}$ of f at $\underline{\beta}$; $\hat{f}$ is

$$\hat{f}(\underline{\beta} + \underline{\beta}_+) = f(\underline{\beta}) + \nabla f(\underline{\beta})^T \underline{\beta}_+ + \frac{1}{2} \underline{\beta}_+^T \nabla^2 f(\underline{\beta}) \underline{\beta}_+. \tag{2.9}$$

$\hat{f}$ is minimized when $\underline{\beta}_+$ is selected as $\underline{\beta}_+ = \Delta\underline{\beta}$ of (2.8). In order to minimize objective function $\underline{\beta}$ should be variated towards $\underline{\beta} + \Delta\underline{\beta}$, [27]. Objective to solve may be expressed in equality form

$$f(\underline{\beta}) = 0,$$

instead of minimizing $f(\underline{\beta})$. Target objective function of this study which is given in (2.4) is of this kind and general solution is handled in the next section.

2.2.1 Newton-Raphson Method for Solving One Equation

Problem of solving one non-linear equation can be defined in the form

$$\arg f(\underline{\beta}) = 0, \tag{2.10}$$

which has a solution $\underline{\beta} = \underline{\beta}^*$.

For simplicity, consider a function $f(\beta) : \mathbb{R} \to \mathbb{R}$ continuous in its parameter β and by linearizing the function at a point β_c and draw a tangent at point $(\beta_c, f(\beta_c))$. This line is modeled as

$$M_c(\beta) = f(\beta_c) + f'(\beta_c)(\beta - \beta_c) \tag{2.11}$$

The point that $M_c(\beta)$ crosses the β axis can be easily shown

$$0 = f(\beta_c) + f'(\beta_c)(\beta^* - \beta_c)$$

$$\beta^* = \beta_c - \frac{f(\beta_c)}{f'(\beta_c)}$$

where $-f(\beta_c)/f'(\beta_c)$ is known as Newton-Raphson method's step. Newton-Raphson steps iterates through the point where $f(\beta)$ crosses the β axis if problems is well defined, means $f(\beta) = 0$ at some point.

2.2.2 Multivariate Case

When the aim is to solve M number of equations in K number of parameters, Gauss-Newton method gives a solution. Abstract model 2.11 can be generalized to multidimensional case as given in (2.12).

$$M_c(\underline{\beta}) = F(\underline{\beta}) + J(\underline{\beta})(\underline{\beta} - \underline{\beta}_c) \tag{2.12}$$

where $F(\underline{\beta})$ and $J(\underline{\beta})$ are defined as

$$\underline{F} = \begin{bmatrix} f_1(\underline{\beta}) \\ f_2(\underline{\beta}) \\ \vdots \\ f_M(\underline{\beta}) \end{bmatrix} \quad \text{and} \quad J_{ij} = \left[\frac{\partial f_i(\underline{\beta})}{\partial \beta_j}\right] \tag{2.13}$$

Since $M_c(\underline{\beta})$ is a vector, in general it is not expected that there is a $\underline{\beta}^*$ that makes $M_c(\underline{\beta}) = 0$. A least-squares solution can be found.

$$\text{minimize} \quad \frac{1}{2}\|M_c(\underline{\beta})\|_2^2 \tag{2.14}$$

If $J(\underline{\beta})$ has full column rank, then the (2.14) is expanded to

$$\text{minimize} \quad \frac{1}{2}\left(\underline{F}(\underline{\beta}) + J(\underline{\beta})(\underline{\beta} - \underline{\beta}_c)\right)^T \left(\underline{F}(\underline{\beta}) + J(\underline{\beta})(\underline{\beta} - \underline{\beta}_c)\right)$$

which has a minimizing solution [31]

$$\Delta\underline{\beta} = -\left(J(\underline{\beta}_c)^T J(\underline{\beta}_c)\right)^{-1} J(\underline{\beta}_c)^T \underline{F}(\underline{\beta}_c). \tag{2.15}$$

2.3 Partial Differential Equation Based Models

Dynamics of a physical system are modeled by Partial Differential Equations (PDEs) when relation between chosen input and output depends on more than one variable. As an example, dynamics of a large space structure from a torque input to a deflection or angular velocity at some point on the structure depends on time and the distance from where the input is applied. A second order PDE is represented as a function of a function depending n variables and its first and second partial derivatives,

$$F\left(x_1, x_2, \ldots, x_n, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_n}, \ldots, \frac{\partial^2 u}{\partial x_n \partial x_n}\right) = 0$$

having a solution

$$u = u(x_1, \ldots, x_n).$$

A second order PDE in two independent variables x_1 and x_2 and has a linear form can be represented as

$$A\frac{\partial^2 u}{\partial x_1^2} + B\frac{\partial^2 u}{\partial x_1 \partial x_2} + C\frac{\partial^2 u}{\partial x_2^2} + D\frac{\partial u}{\partial x_1} + E\frac{\partial u}{\partial x_2} + Fu = G$$

where A, B, C, D, E and F are constants or functions of x_1 and x_2, [32].

An application to flexible systems is from a study of Raskin and Halevi [1]. In this study a transfer function model governed by wave equation is derived and a model based controller is developed to increase performance characteristics. Structure is a uniform rod with free ends having length L, an input $M(t)$ which is torque moment at one end of the rod and output $\theta(x,t)$ which is the torsion angle at distance x from where the torque applied shown in Figure 2.1.

Dynamics are represented by the wave equation $\theta(x,t)$

$$\frac{\partial^2 \theta(x,t)}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 \theta(x,t)}{\partial t^2}$$

and boundary conditions are

$$GI_p \frac{\partial \theta(x,t)}{\partial x_{|x=0}} = -M(t), \quad \frac{\partial \theta(x,t)}{\partial x_{|x=L}} = 0$$

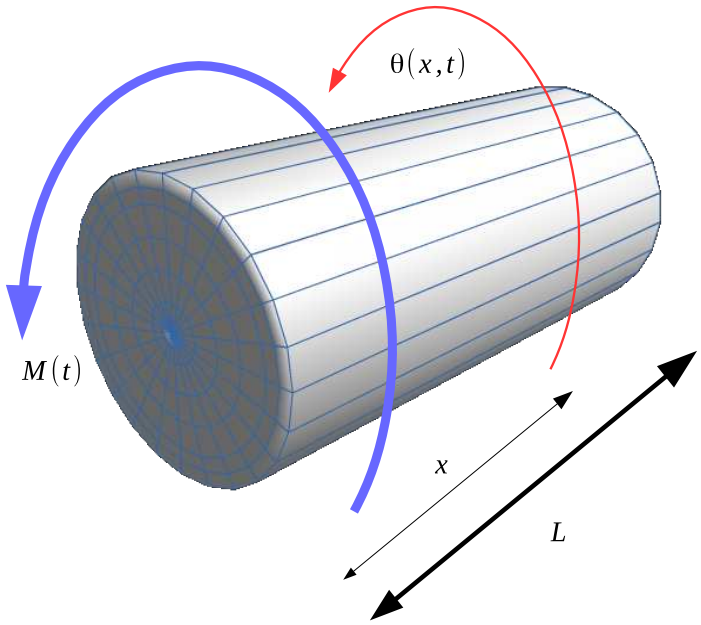

Figure 2.1: *Free-free uniform rod with input $M(t)$ and output $\theta(x,t)$.*

where I_p is the polar moment of inertia, G is the shear elasticity modulus, ρ is the material density and $c = (G/\rho)^{1/2}$ is the wave propagation velocity. After taking Laplace transform, PDE turns into be an ordinary differential equation in x

$$\frac{\partial^2\theta(x,t)}{\partial x^2} - \frac{s^2}{c^2}\theta(x,s) = 0$$

Then the solution is

$$\theta(x,s) = C_1(s)e^{\frac{sx}{c}} + C_2(s)e^{-\frac{sx}{c}}$$

from the boundary conditions we can solve for C_1 and C_2, and obtain

$$\frac{\theta(x,s)}{M(s)} = \frac{c}{GI_p}\frac{1}{s}\frac{e^{-2\tau(1-\lambda)s}+1}{1-e^{-2\tau s}}e^{-\tau\lambda s}$$

where $\lambda = x/L$ is normalized distance and $\tau = L/c$ is a time constant. Resulting transfer function has infinite number of poles and zeros on imaginary axis, at points given as

$$p_k = \frac{k\pi}{\tau}j \quad \text{and} \quad z_k(\lambda) = \frac{(k+1/2)\pi}{\tau(1-\lambda)}j.$$

Thus it can not be represented by a finite number of states but can be approximated. Small perturbations in parameter L may result large changes in pole and zero locations of the plant model.

2.4 Experiment Based Non-Parametric Identification

For the cases which a consistent system model that includes all dynamics can not be defined for the real system or an abstract structure does not include all external effects, experiments can be conducted on the real system. This gives an understanding of input/output characteristics of the system. Procedure starts by defining a reasonable signal as an input to the plant and collecting samples of output data. Comparing magnitude and phase properties of input and output at distinct frequencies results into accurate frequency response characteristics. In order to obtain these characteristics a non-parametric identification method based on Discrete Fourier Transform (DFT) is used. Frequency response characteristics at predefined, distinct frequencies are obtained in terms of phase and magnitude values by applying sinusoids having constant frequencies and magnitudes. DFT is derived from Fourier Transform in continuous time domain

$$X_c(j\omega) = \int_{-\infty}^{\infty} x_c(t)e^{-jwt}dt$$

by discretizing time axis,

$$\hat{X}(j\omega) = \sum_{-\infty}^{\infty} x_c(kT_s)e^{-jwkT_s}T_s, \quad t = kT_s$$

is obtained, by discretizing frequency axis,

$$\hat{X}(j\omega_n) = \sum_{-\infty}^{\infty} x_c(kT_s)e^{-jw_nkT_s}T_s$$

is obtained.

Input signals are selected as

$$x(t) = Asin(\omega t)$$

and in discretized form

$$x[kT_s] = Asin(\omega kT_s)$$

Frequency points ω_k, $k = \{1, \ldots, M\}$ are in the interval $0 < \omega_k < \frac{2\pi}{T_s}$ by Nyquist sampling theorem. If this interval is separated to N uniform steps, discrete frequency steps ω_n are defined for N point DFT

$$\omega_n = \frac{2\pi n}{NT_s}, \quad n = 0, 1, ..., N-1 \tag{2.16}$$

$$\frac{1}{T_s}\hat{X}(j\frac{2\pi n}{NT_s}) = \sum_{0}^{N-1} x[k]e^{-j2\pi kn/N}$$

If $X[n]$ is defined as

$$X[n] = \frac{1}{T_s}\hat{X}(j\frac{2\pi n}{NT_s})$$

DFT of $x[n]$ is calculated by the sum below

$$X[n] = \sum_{0}^{N-1} x[k]e^{-j2\pi kn/N}$$

In order to obtain magnitude and frequency response of the system at given frequency point ratio of the DFT sum of output and input is calculated.

$$Y[n] = \sum_{k=0}^{N-1} y[k]e^{-j2\pi kn/N}$$

$$U[n] = \sum_{k=0}^{N-1} u[k]e^{-j2\pi kn/N}$$

$$G[n] = \frac{Y[n]}{U[n]}$$

$$G[n_0] = |G[n_0]|\angle G[n_0] = \frac{|Y[n_0]|}{|U[n_0]|}e^{j(\angle Y[n_0] - \angle U[n_0])}$$

where $n_0 = \frac{\omega_0 NT_s}{2\pi}$ from (2.16).

Non-parametric identification by using DFT gives more reliable results than comparing input and output signals in time domain.

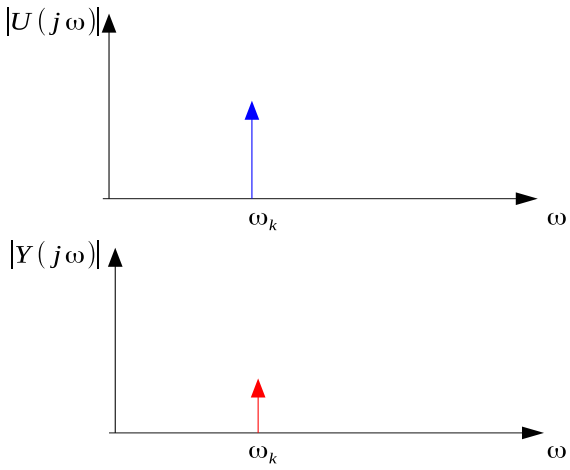

Figure 2.2: $y(t) = A_y cos(\omega_k t + \psi_y)$ *and* $u(t) = A_u cos(\omega_k t + \psi_u)$

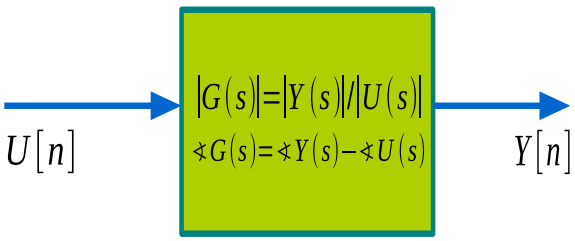

Figure 2.3: *Block model.*

2.5 Model Constraints

In this study proposed method uses frequency response of the system taken into consideration to match a transfer function. Frequency response can be obtained by solving dynamic equations of the system or by conducting experiments on the system. These two methods were explained in previous two sections. After phase and magnitude responses at distinct frequencies which are not needed to be uniformly spaced are collected, next step is to solve an optimization problem. Aim is to find parameters of a finite order transfer function multiplied by a delay term. Parameter search problem is solved by Non-Linear Least Squares (NLLS). Resulting parameters must not only minimize the relative error between previously obtained frequency response and identified parametric model but also satisfy constraints.

Target transfer function after separating integral action and delay term, $P_N(s)$ in (1.5) is restricted to be minimum phase. When negative gain situation is neglected, hence it can be separated from collected frequency response data, restrictions can be embedded into the optimization problem as a non-negativity constraint. Therefore, all parameters in the vector $\underline{\beta}$ defined in (1.6) must be non-negative.

After adding constraints, optimization problem (2.7) turns into the form

$$\text{minimize} \quad f(\underline{\beta}) \tag{2.17}$$

$$\text{subject to} \quad f_i(\underline{\beta}) \leq 0, \quad \text{for} \quad i = 1 \ldots, L. \tag{2.18}$$

One solution is rewriting the constrained optimization problem by making constraints implicit in the objective function. [27]

$$\text{minimize} \quad f(\underline{\beta}) + \sum_{i=1}^{K} I(f_i(\underline{\beta})) \tag{2.19}$$

where $I(f_i) : \mathbb{R} \to \mathbb{R}$ for $i = 1, \ldots, L$ is the indicator function:

$$I(f_i) = \begin{cases} 0 & f_i \leq 0 \\ \infty & f_i > 0 \end{cases}. \tag{2.20}$$

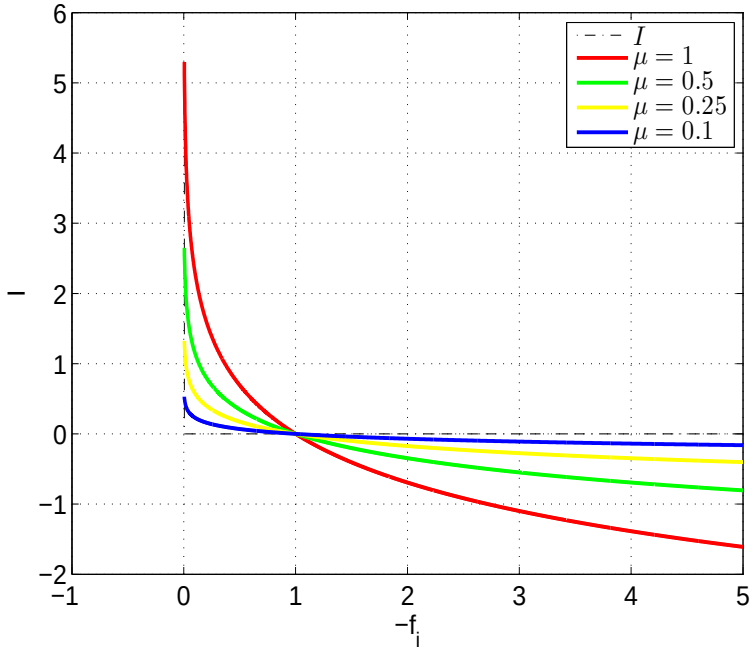

Figure 2.4: *Log-barrier function for several μ values. ($I(f_i)$ vs. $-f_i$)*

Indicator function is not differentiable and can not be used in Newton method. Logarithmic barrier function is an approximation to the indicator function, differentiable and it is given as

$$\hat{I}(f_i) = -\mu \log(-f_i(\underline{\beta})), \tag{2.21}$$

where μ is a positive constant. Figure 2.4 shows log-barrier function for decreasing μ values. As μ decreases approximation becomes more accurate, [27].

Log-barrier method can be implemented in the proposed method of this study by selecting constraint functions f_i as negative of each parameter of vector $\underline{\beta}$. When $\underline{\beta}$ is redefined as

$$\underline{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{bmatrix},$$

then constraint functions are

$$f_i = -\beta_i \quad \text{for} \quad i = 1, 2, \ldots, K.$$

By inserting constraints to the objective function and selecting indicator function as logarithmic function, overall optimization objective becomes

$$\underset{\underline{\beta}}{\text{minimize}} \quad \epsilon(\underline{\beta}) = \sum_{i=1}^{M} \left| \frac{\Phi_i - G_N(j\omega_i, \underline{\beta})}{G_N(j\omega_i, \underline{\beta})} \right|^2 - \mu Q(\underline{\beta}) \tag{2.22}$$

where Φ_i are frequency response data points $G(j\omega_i)$ as obtained from the procedure of Section 2.4 and

$$Q(\underline{\beta}) = \sum_{i=1}^{K} \log(\beta_i). \tag{2.23}$$

Substitute (2.23) in (2.22), obtain:

$$\underset{\underline{\beta}}{\text{minimize}} \quad \epsilon(\underline{\beta}) = \frac{1}{\mu} \sum_{i=1}^{M} \left| \frac{\Phi_i - G_N(j\omega_i, \underline{\beta})}{G_N(j\omega_i, \underline{\beta})} \right|^2 - \sum_{i=1}^{K} \log(\beta_i), \tag{2.24}$$

$G_N(s)$ has the structure of (2.2) with parameters $\underline{\beta}$ (2.6) to be determined. In terms of a simplified notation the optimization problem defined above can be considered as minimizing

$$\epsilon(\underline{\beta}) = \underline{F}^H \underline{F} \tag{2.25}$$

where

$$\underline{F} = \begin{bmatrix} \frac{1}{\mu} \frac{\Phi_1 - G_N(j\omega_1, \underline{\beta})}{G_N(j\omega_1, \underline{\beta})} - \sum_{i=1}^{K} \log \beta_i \\ \frac{1}{\mu} \frac{\Phi_2 - G_N(j\omega_2, \underline{\beta})}{G_N(j\omega_2, \underline{\beta})} - \sum_{i=1}^{K} \log \beta_i \\ \vdots \\ \frac{1}{\mu} \frac{\Phi_M - G_N(j\omega_M, \underline{\beta})}{G_N(j\omega_M, \underline{\beta})} - \sum_{i=1}^{K} \log \beta_i \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_M \end{bmatrix} \tag{2.26}$$

Furthermore the Jacobian of vector $\underline{F}$ is defined as

$$J_{ij}(\underline{\beta}) = \left[\frac{1}{\mu} \frac{\partial F_i(\underline{\beta})}{\partial \beta_j} - \frac{1}{\beta_j} \right] \tag{2.27}$$

Log-barrier method has similarities with solution methods for dual optimization problems and efficient for most cases. The above discussion summarizes the main idea behind the steps of the algorithm.

2.6 Alternative Basis Function

Whereas the aim is extracting resonant modes from frequency response data, basis function in (2.2) is expressed by second order terms. For different cases

basis can be selected as a first order term, and minimum phase part $P_N(s)$ of $G_N(s)$ turns into a product of first order transfer functions,

$$P_N(s) = \prod_{k=1}^{N} \frac{(a_{k,1}s + a_{k,0})}{(b_{k,1}s + b_{k,0})}. \tag{2.28}$$

In an other study proposed method is implemented to find a finite order approximation of a fractional order transfer function that models non-laminated magnetic suspension system, [30]. Basis function is selected as above. Suitable results that are very close to the fractional expansion method used in Matsuda's study [33] are obtained. Simulation results and comparisons are also given in Section 4.2.

Chapter 3

NLLS APPROACH and PROPOSED ALGORITHM

3.1 Gauss-Newton Method

Problem is mainly finding a suitable complex function of '$s = j\omega$' in the form of (2.2) that matches M number of complex data points as close as possible and satisfies non-negativity constraints. Objective can be expressed in vectorial form as

$$\text{minimize} \quad \epsilon(\underline{\beta}) = \underline{F}^H \underline{F} \tag{3.1}$$

where $\underline{F}$ is defined in (2.26).

For solving a set of non-linear equations, Gauss-Newton method is used. This non-linear solver has advantages over linearizing problem such as adding constraints, on the other hand comes up with some disadvantages. Advantages are:

1. Locally quadratically convergent on problems whose optimal solutions are zero or very small.

2. Quickly locally linearly convergent on problems which are not highly non-linear and have solutions very close to zero.

3. It takes one iteration to solve linear problems.

and disadvantages are:

1. Slowly locally linearly convergent on problems that are highly non-linear or have large errors at optimal points.
2. Not locally convergent on problems that are very non-linear or have very large errors at optimal points.
3. Not well defined if Jacobian does not have full column rank.
4. Does not guarantee to converge to global minimum.

Although Gauss-Newton method can not guarantee convergence to the global minimum, it is certain that it iterates through a local minimum. [31] Newton's method may also fail to converge and μ parameter of the log-barrier method may not be chosen optimally, picking a static μ value may make log function a bad approximation to the indicator function in (2.20). In order to overcome these two difficulties, an extension to log-barrier method given in Boyd's [27] and Levenberg-Marquardt algorithm in Lourakis' [34] is used as inner iterations of the algorithm proposed in this study.

Full structure of the implementation can be expressed as a tree model as given below:

- **Input:** Collected frequency response data at distinct frequency points, desired order of the resulting transfer function.
- **Main loop:** Data set selection and parameter initialization.
  - **Outer loop:** Extended log-barrier iterations of decreasing μ.
    - **Inner loop:** Levenberg-Marquardt algorithm.
- **Output:** Minimizing parameter set, $\underline{\beta}^*$.

3.1.1 Inner Loop

All non-linear optimization problems are not solvable by most basic form of Gauss-Newton method. There are ill-conditioned cases, for example objective function $\epsilon(\underline{\beta})$ may not cross $\underline{\beta}$ plane. Ill-conditioned circumstances impose to extend basic form of iterations with modified Newton steps that guarantee convergence. Trust region methods are a class of solvers and Levenberg-Marquardt method which can be considered as a predecessor of trust region methods constitutes inner loop of the algorithm.

Trust region methods increases the reliability of iterative optimization methods and can be applied to ill-conditioned problems. Trust region is a neighborhood of the result of the current iteration which is centered at current result, [35]. Trust region is adjusted at every iteration in a reasonable way in order to find a better minimizer. Precisely, trust region is expanded when the result of current iteration improves solution of the problem. On the other hand it shrinks if it reduces the optimality of the solution.

The critical step is to compute the radius of the trust region. Method starts with an initial trust region and a point $\underline{\beta}$ at the center of the region. Newton step is modified by recalculation of the trust region. Contribution of the current iteration is obtained and a merit function updates trust region for the next iteration. Levenberg-Marquardt method adjusts trust region by increasing or decreasing a damping coefficient. Step is modified by σ as follows

$$M_{LM}(\underline{\beta}) = J^H(\underline{\beta}_c)J(\underline{\beta}_c) + \sigma I \tag{3.2}$$

$$\underline{s}_N = -M_{LM}^{-1}(\underline{\beta}_c)J^H(\underline{\beta}_c)\underline{F}(\underline{\beta}_c) \tag{3.3}$$

$$\underline{\beta}_n = \underline{\beta}_c + \underline{s}_N \tag{3.4}$$

where I is the identity matrix, multiplied by damping coefficient σ [34], $\underline{\beta} = \underline{\beta_c}$ denote the current selection of the parameters, it is updated by $\underline{s}_N$.

Step $\underline{s}_N$ is a minimizer of

$$\min \quad \|\underline{F}(\underline{\beta}_k) + J^H(\underline{\beta}_k)\underline{s}\|_2^2 \tag{3.5}$$

$$\text{s. t.} \quad \|\underline{s}\|_2 \leq \Delta_k, \tag{3.6}$$

which is a similar expression as (2.14), but a bounding constraint is added. Region Δ_k is called the trust region radius, [36].

Constraint 3.6 provides trust region method properties to Levenberg-Marquardt algorithm and 'trust region' Δ is revised at every iteration by the update of σ.

Algorithm that is used in Inner Loop is as follows

Step 1 Initialize parameter set $\underline{\beta} = \underline{\beta}_0$ and ϵ_1, ϵ_2, ϵ_3, $k = 0$, $\tau > 0$, k_{max} is a very large integer.

Step 2 $k = 0$; $v = 2$; $\underline{\beta} = \underline{\beta}_0$

Step 3 $A = J^H J$; $\epsilon_{\underline{\beta}} = \underline{F}^H(\underline{\beta})\underline{F}(\underline{\beta})$ of (2.25); $\underline{g} = J^H \epsilon_{\underline{\beta}}$;

Step 4 Stop if $\|\underline{g}\|_\infty \leq \epsilon_1$.

Step 5 $\sigma = \tau \max_{i=1,\ldots,m}(A_{ii})$

Step 6 Repeat:

Step i k = k+1;

Step ii Solve $(A + \sigma I)\delta_{\underline{\beta}} = \underline{g}$;

Step iii Stop if $\|\delta_{\underline{\beta}}\| \leq \epsilon_2 \|\underline{\beta}\|$;

Step iv $\underline{\beta}_{new} = \underline{\beta} + \delta_{\underline{\beta}}$;

Step v $\rho = \left(\|\epsilon_{\underline{\beta}}\|^2 - \underline{F}^H(\underline{\beta}_{new})\underline{F}(\underline{\beta}_{new})\right) / \left(\delta_{\underline{\beta}}^T \left(\sigma\delta_{\underline{\beta}} + \underline{g}\right)\right)$

Step vi If $\rho \leq 0$, $\sigma = \sigma v$; $v = 2v$; Goto Step i.

Step vii $\underline{\beta} = \underline{\beta}_{new}$;

Step viii $A = J^H J$; $\epsilon_{\underline{\beta}} = \underline{F}^H(\underline{\beta})\underline{F}(\underline{\beta})$ of (2.25); $\underline{g} = J^H \epsilon_{\underline{\beta}}$;

Step ix $\sigma = \sigma \max(\frac{1}{3}, 1-(2\rho-1)^3)$; $v = 2$; Goto Step i.

Step x Stop if $\left(\|\underline{g}\|_\infty \leq \epsilon_1\right)$ or $\left(\|\epsilon_{\underline{\beta}}\|^2 \leq \epsilon_3\right)$.

Step xi Stop if $k = k_{max}$.

Step xii Goto Step i.

Step 7 $\underline{\beta}^* = \underline{\beta}$

At *Step ii* of the Levenberg-Marquardt algorithm, a complex least squares solution with constrained phase is used, [37]. Least squares problem is defined as follows

$$A\underline{x} = \underline{b} \tag{3.7}$$

where $A, \underline{x}$ and $\underline{b} \in \mathbb{C}$, by writing x in polar form

$$A\underline{x}_{mag}e^{j\psi} = \underline{b}. \tag{3.8}$$

Solution to $\underline{x}_{mag}$ is given as

$$\underline{x}_{mag} = M^\dagger \Re(A^H \underline{b} e^{-j\psi}) \tag{3.9}$$

where '†' denotes pseudo-inverse and

$$M = \Re(A^H A). \tag{3.10}$$

Purpose is to obtain real coefficients, so ψ is selected $\psi = 0$.

'Trust region' Δ is initialized at *Step 5* of the Levenberg-Marquardt algorithm. There are four kinds of stopping criterion: procedure stops if

- a large number of iterations is reached, at *Step xi* of the procedure k is checked,
- error which is denoted by $\epsilon_{\underline{\beta}} = \underline{F}^H(\underline{\beta})\underline{F}(\underline{\beta})$ is below a predefined error level ϵ_3,
- magnitude of the gradient $\underline{g} = J^H \epsilon_{\underline{\beta}}$ drops below threshold ϵ_1,
- or the change $\delta_{\underline{\beta}}$ in parameters is less than a threshold $\epsilon_2 \|\underline{\beta}\|$.

Damping factor σ is adjusted at *Step vi* or *Step ix*. $\rho \leq 0$ means new error after the update of parameters is higher than previous error level. If this occurs damping factor σ is increased by multiplier v. Increase in damping factor shrinks 'trust region' Δ. Rationale behind this is that a minimizer is in an area of a smaller radius or the current point itself. Otherwise, $\rho > 0$ means that a better minimizer is reached, and new parameter set $\underline{\beta}$ can be selected as $\underline{\beta} = \underline{\beta}_{new}$. This is followed by *Step ix* where the damping coefficient σ is reset. When current point is close to the solution, σ is selected as a small number, and the step becomes a Gauss-Newton step.

3.1.2 Outer Loop

So far unconstrained optimization is discussed. Further, non-negativity constraints should also be handled, since all poles and zeros of $G_N(s)$ are restricted to be in the left half plane when integrator and delay terms are removed. A simple solution is to use barrier functions, [27].

Log-barrier function is an approximation to the indicator function in (2.20). Purpose of the indicator function is to raise magnitude of the cost function to a very large value if constraints are not satisfied. Precisely, this method inserts a barrier on the parameter space between the region where the constraints are satisfied and the region where they are not satisfied.

Logarithmic barrier function is used in our case and added to objective function implicitly as shown in (2.24). In order to increase optimality of the solution, a simple extension is applied to the log-barrier method. Instead of solving the problem in Inner Loop by using one constant value of μ, a sequence of problems which are turned into be unconstrained by adding constraints to the objective function are solved with decreasing μ values. Parameter set in each problem are initialized by last found values in previously solved problem. This method was first called *sequential unconstrained minimization technique* and proposed by Fianco and McCormick, [27]. A version of the method proposed in Boyd [2009] is as follows

Step 1 Given strictly feasible $\underline{\beta},\ \mu = \mu_0 > 0,\ \alpha > 1$.

Step 2 Set $tol \in \mathbb{R}^+$ a very small number.

Step 3 Compute $\underline{\beta}^*$ by minimizing (2.25), starting at $\underline{\beta}$.

Step 4 Update $\underline{\beta} = \underline{\beta}^*$.

Step 5 Quit if $m\mu < tol$.

Step 6 Decrease μ, $\mu = \mu/\alpha$, Goto Step 3.

Inner Loop where the solution is calculated by Levenberg-Marquardt algorithm is at *Step 3*. Iterations are stopped by the condition in *Step 5* when μm decreases to a number below threshold tol, where $m > 0$ is a constant. Result returned from Outer Loop is a minimizer parameter set for the approximating function $G_N(s)$ that fits data selected in the current iteration of the Main Loop and returned as an initialization point of the next iteration of Main Loop.

However, for good convergence properties, initial values assigned to $\underline{\beta}$ in the parameter space must be selected carefully. There is no perfectly defined initialization point, it can be selected in a logical way for our specific case.

3.1.3 Main Loop

Complex curve fitting operation is made in arbitrary intervals on the frequency domain. Since the aim is to obtain resonance and anti-resonance terms, peaks on the magnitude curve increase in importance. Resonance and anti-resonance terms show themselves as peaks on magnitude curve at some frequencies. Procedure starts at lowest frequency point to obtain integral gain when integral action is present. It is followed by an iterative parameter search sub-step which is performed by focusing on the frequency point, where the highest error between Φ and $G_N(s)$ occurs. Parameter count is dynamic, and increases at every iteration of Main Loop. Adding new parameters to parameter set $\underline{\beta}$ improves the feasibility of the optimization problem in Inner Loop. Outer Loop is continued until reaching a reasonable error level or a given transfer function order.

Applying NLLS methods on all data points at one outer step makes it difficult to define model order and initial values of parameters and will not give desired result. Therefore, selection of frequency intervals where resonant/anti-resonant peaks are visible gives a good starting point. An admissible approach for data point selection is starting from the lowest frequency region and working our way to high frequencies we try to estimate the values of resonance frequencies and damping coefficients by successively adding the frequency intervals where the highest relative error occurs to the previously considered intervals.

These intervals are defined as:

$$\begin{aligned} A &= \{A_{L_0}, A_{L_1}, \dots A_{L_N} | 1 \leq L_i \leq M\}, \\ A_{L_i} &= \{\Phi_l \ | l = 1, 2, \dots L_i\}, \\ L_{i+1} &= \max \left\{ \arg_i \left\| \frac{\Phi_i - G_N(j\omega_i)}{G_N(j\omega_i)} \right\|_\infty, L_i \right\}, \\ A_{L_0} &\subseteq A_{L_1} \subseteq \cdots \subseteq A_{L_N}, \end{aligned} \tag{3.11}$$

$$\begin{aligned} B &= \{B_{L_0}, B_{L_1}, \dots B_{L_N} | 1 \leq L_i \leq M\}, \\ B_{L_i} &= \{j\omega_l \ | l = 1, 2, \dots L_i\}, \\ B_{L_0} &\subseteq B_{L_1} \subseteq \cdots \subseteq B_{L_N}. \end{aligned} \tag{3.12}$$

Every new data set includes previous set and data points from low frequency to high frequency where maximum error occurs. B_{L_i} is the set of data on frequency axis and A_{L_i} is the set of frequency response data. These data is passed to the Inner Loop to conduct optimization algorithm. Main Loop procedure is given as follows

Step 1 Set $tol \in \mathbb{R}^+$ a very small number and $\mathcal{N}$, where $2\mathcal{N}+1$ defines the degree of the resulting transfer function $G_N(s)$,

Step 2 Set $N \leftarrow 0$

Step 3 Calculate integral gain K_0;

Step i Define $L_0 \leftarrow L_{init}$, where $L_{init} \in \mathcal{Z}^+$ and $L_{init} \ll M$,

Step ii Define A_{L_0} and B_{L_0}, according to (3.11), (3.12),

Step iii Define $G_N(s)$, $G_0(s) = e^{-hs}\frac{K}{s}$,

Step iv Initialize: $K \leftarrow \frac{|\Phi_1|}{w_1}$ and $h \leftarrow 0$,

Step v Pass initial parameters K, h and data sets A_{L_0}, B_{L_0} to Outer Loop and solve optimization problem

Step 4 Set $\omega_c \leftarrow \underset{\omega_i}{\arg} \left\| \frac{\Phi_i - G_N(j\omega_i)}{G_N(j\omega_i)} \right\|_\infty$,

Step 5 Calculate the coefficients of increasing number of resonance/anti-resonance terms,

Step i $L_{N+1} = \max \left\{ \underset{i}{\arg} \left\| \frac{\Phi_i - G_N(j\omega_i)}{G_N(j\omega_i)} \right\|_\infty, L_N \right\}$,

Step ii Set $N \leftarrow N + 1$,

Step iii Define data sets A_{L_N} (3.11) and B_{L_N} (3.12),

Step iv Define parameter vector $\underline{\beta} = [K_0, h, \underline{\theta}_1^T, \ldots, \underline{\theta}_N^T]^T$ as given in (2.6).

Step v Initialize: $\underline{\theta}_N$, select $a_N = 1$, $\zeta_{N,n} = 0.5$, $\omega_{N,n} = \omega_c$, $b_N = 1$, $\zeta_{N,d} = 0.5$, $\omega_{N,d} = \omega_c$, values of the remaining items in vector $\underline{\beta}$ come from the previous iteration.

Step vi Pass initial parameters $\underline{\beta}$ and data sets A_{L_N}, B_{L_N} to Outer Loop and solve optimization problem

Step 6 Goto *Step 4* if ($N < \mathcal{N}$ and $max \left| \frac{\Phi_i - G(j\omega_i, \underline{\theta})}{G_N(j\omega_i, \underline{\theta})} \right| > tol$).

Step 7 $\underline{\beta} = \left[K_0, h, \underline{\theta}_1^T, \underline{\theta}_2^T, \ldots, \underline{\theta}_N^T\right]^T$ and exit

Step 1 is the initialization step where error tolerance *tol* and degree of the resulting transfer function $2\mathcal{N} + 1$ are defined. $2\mathcal{N} + 1$ is correct for the case when an integral action is present. Otherwise, result is a $2\mathcal{N}$ order transfer function. In *Step 3* gain K_0 associated with the integral action is calculated. For most cases this step is skipped, since integral action may not be present or can be separated from obtained frequency response. *Step 4* is a critical step where data points are selected to apply optimization procedure on. It is illustrated in

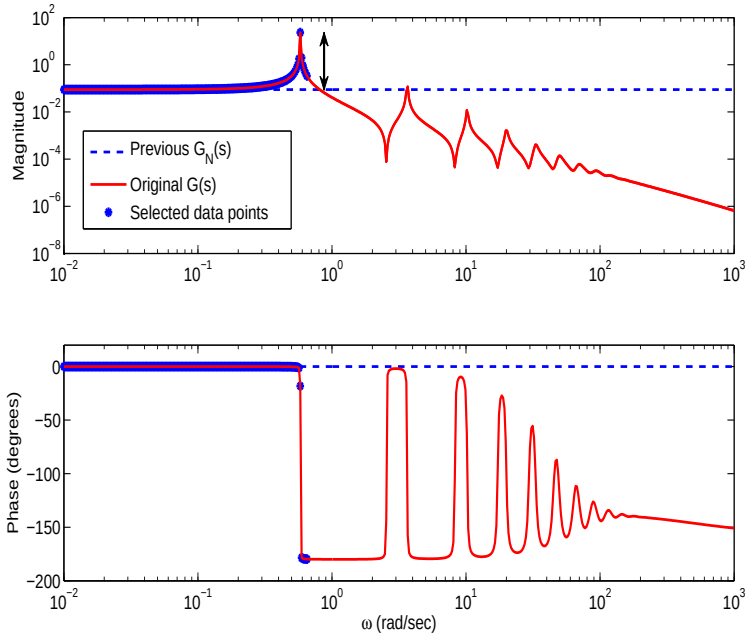

Figure 3.1: *Graphical representation of Step 4 of the Main Loop.*

Figure 3.1. Black arrow shows the magnitude of the maximum error that occurs at a frequency point.

Step 5 is applied in each iteration of the main loop with dynamically increasing number of data points and coefficients. Procedure is continued until desired order of the transfer function is reached. *Step 6* tests the termination conditions.

In order to increase the efficiency of barrier method, initial values of $\underline{\theta}_k$ can be modified, they can be selected as $\underline{\theta}_k = [la_k, \sqrt{l}\zeta_{k,n}, \sqrt{l}\omega_{k,n}, lb_k, \sqrt{l}\zeta_{k,d}, \sqrt{l}\omega_{k,d}]^T$, where $l > 1$.

Finally after obtaining an $(2\mathcal{N}+1)$-th order approximate model for the infinite dimensional system, a balance and truncate method can be used to further reduce the order of the system, [38, 39].

Chapter 4

NUMERICAL EXAMPLES

4.1 Results On Flexible System Models

Three different transfer functions are used for simulation purposes in order to show efficiency of the proposed algorithm. In all the examples $M = 500$ and frequency response is available at logarithmically spaced frequency points (ω_i) between 10^{-2}rad/sec and 10^{3}rad/sec. Furthermore, result are compared to two other methods. Subspace based method from [15] and Fourier Transform based method from [14] are implemented and applied to the same infinite dimensional transfer functions. Approximating transfer functions for each methods are adjusted to have the same order in three examples.

4.1.1 Free-Free Beam System

First example is the transfer function of free-free flexible beam which has an integrator term and a delay term of $h = 0.01$ sec. Its transfer function from force input to velocity measurement is, [3],

$$G_A(s) = \frac{-s \quad e^{-hs}}{(\epsilon_1 s + 1)(\epsilon_2 s + 1)} \frac{(\sinh(m(s))\cos(m(s)) - \cosh(m(s))\sin(m(s)))}{m(s)^3 \quad (\cos(m(s))\cosh(m(s)) - 1)} \tag{4.1}$$

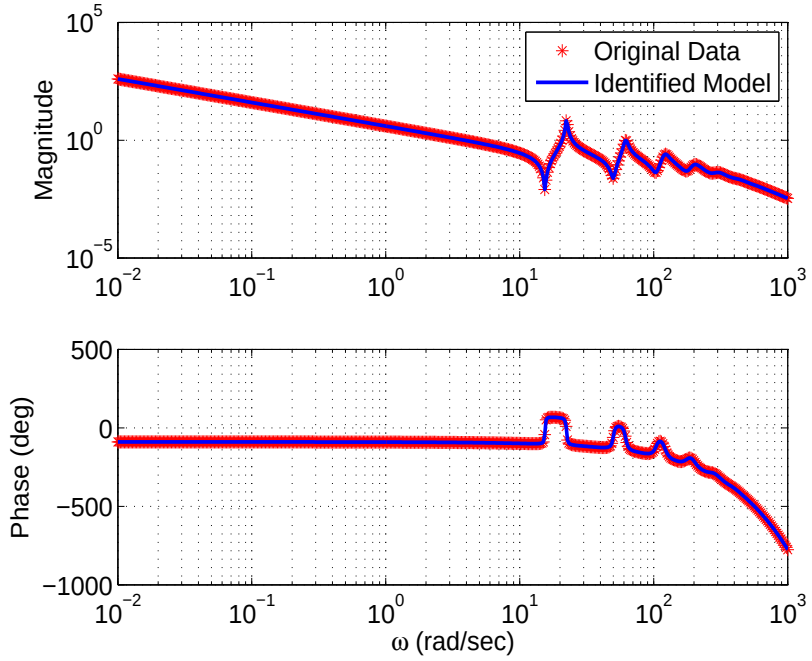

Figure 4.1: *Bode plots of $G_A(s)$ and the identified model $G_N(s)$.*

where $m^4(s) = \frac{-s^2}{(1+\epsilon_1 s)}$ and the damping parameters are selected as $\epsilon_1 = 0.001$ and $\epsilon_2 = 0.0033$.

By applying the algorithm given in above, a 34th order $P_N(s)$ is determined; the estimated delay value is $h = 0.0106$ sec. The Bode plots of the original transfer function and that of $G_N(s)$ are given in Figure 4.1; the resulting relative error is as shown in Figure 4.2. The pole-zero map of G_N is shown in Figure 4.3, see Figure 4.4 for detailed view. The Hankel singular values of P_N are as shown in 4.5.

These result are compared to results of other two methods and relative error plots are given in Figure 4.6. Figure shows plots of NLLS approach (Method I), subspace based method (Method II) and Fourier transform based method (Method III) respectively. Degrees of the transfer functions are 34 for three cases, when integral part is separated.

Subspace based method gives a good approximation of the logarithmically spaced frequency response data. Method works in discrete time domain with

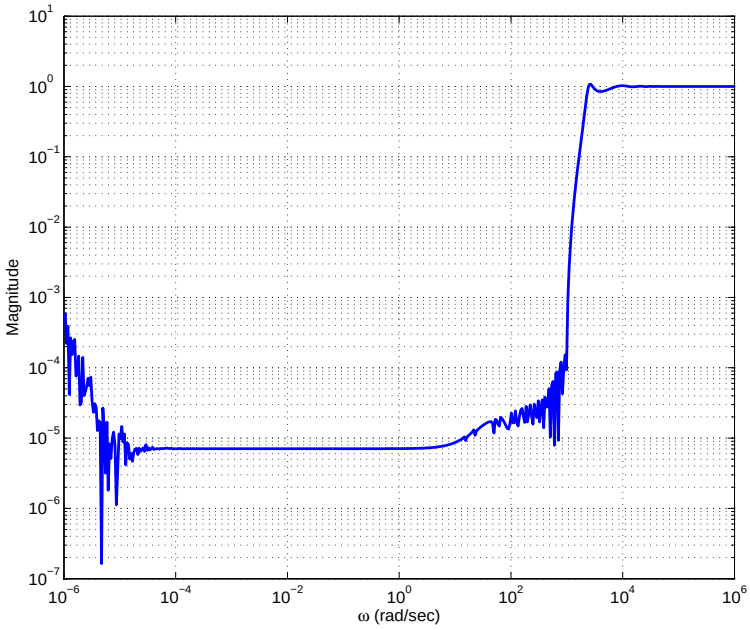

Figure 4.2: *Relative error* $|G_A(j\omega) - G_N(j\omega)|/|G_N(j\omega)|$.

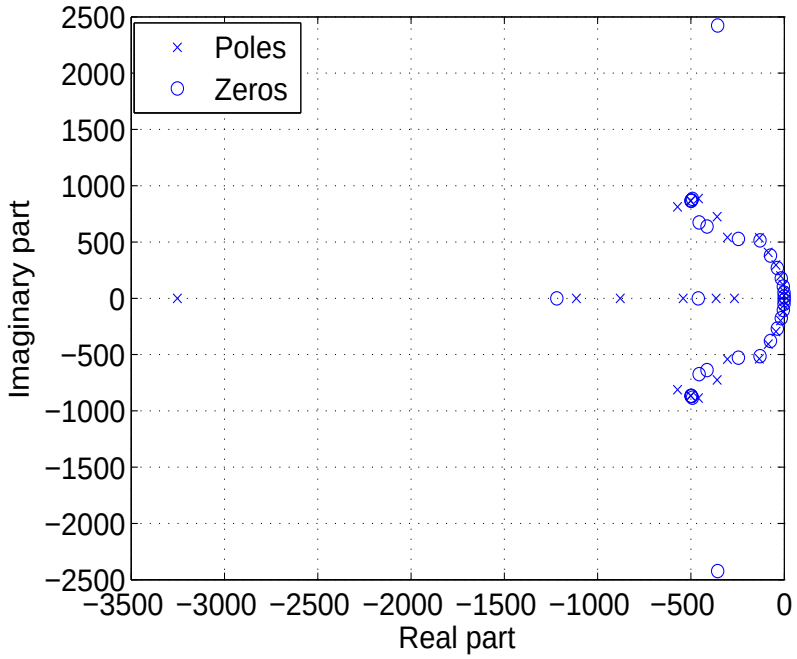

Figure 4.3: *Pole zero map of* G_N.

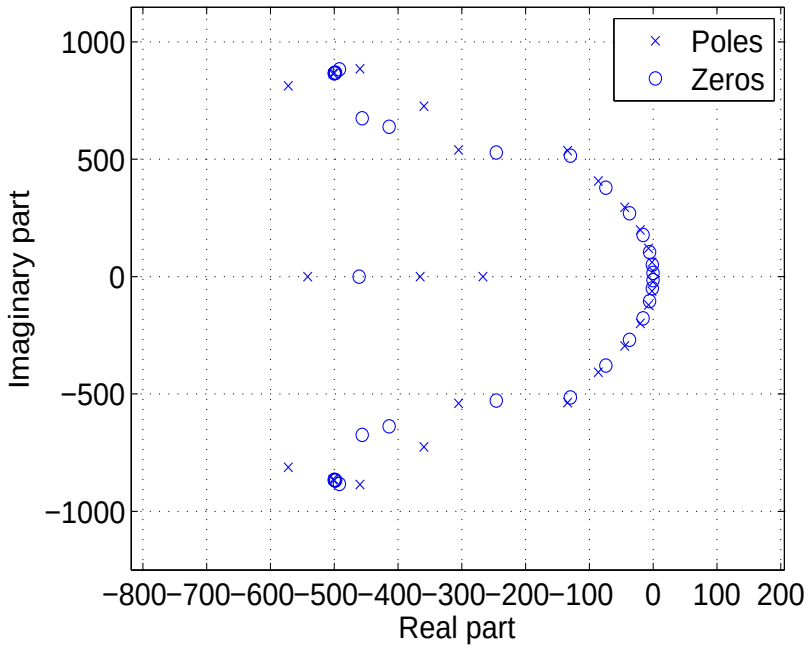

Figure 4.4: *Zoomed pole zero map of* G_N.

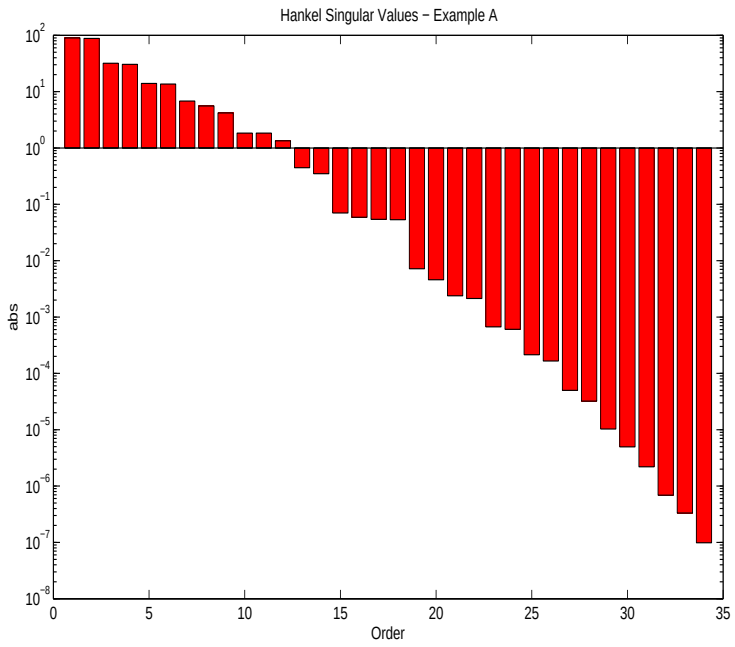

Figure 4.5: *Hankel singular values of* $P_N(s)$ *for example A.*

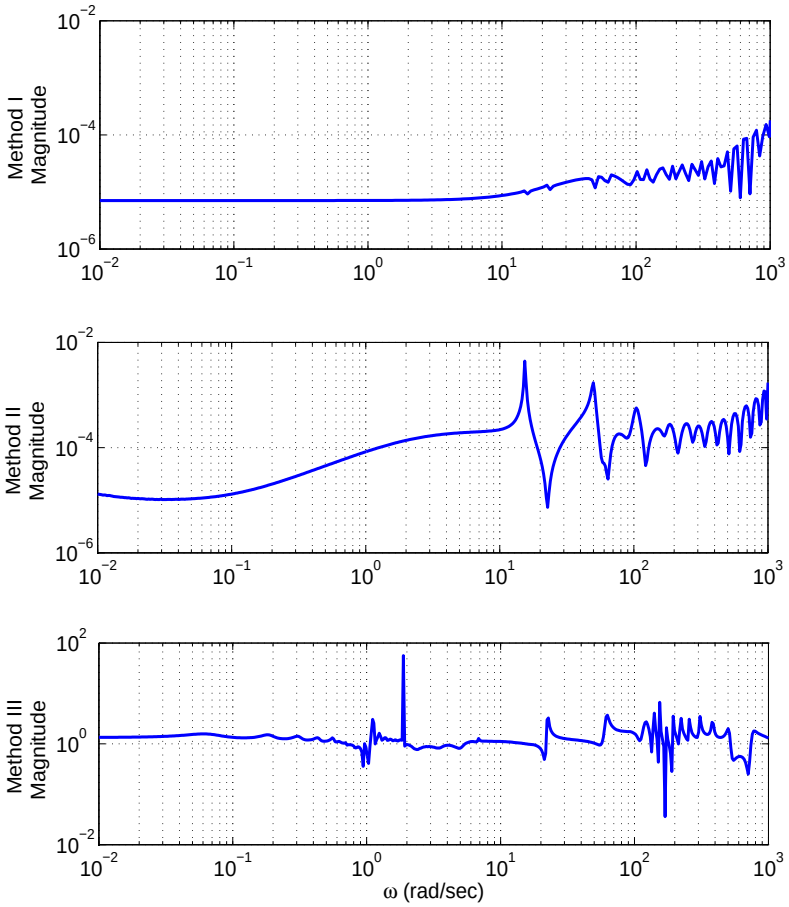

Figure 4.6: *Relative error comparison for three methods.*

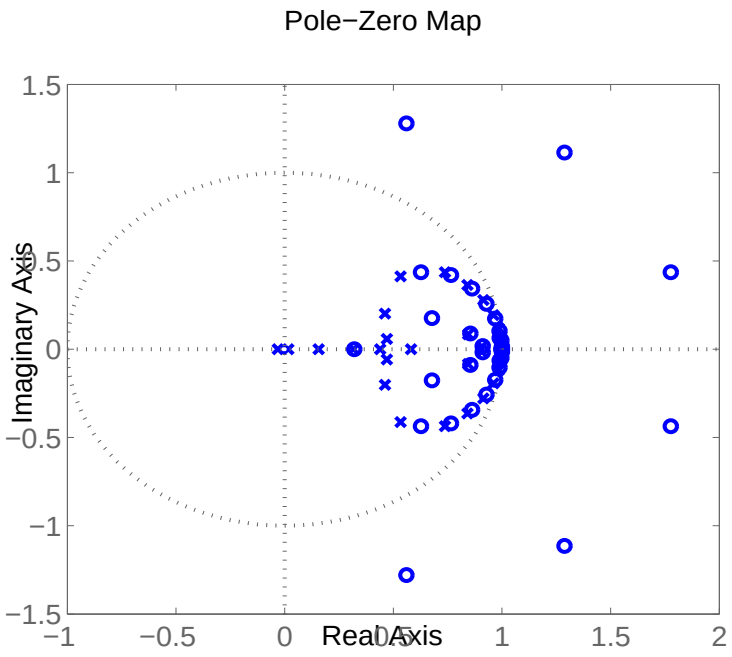

Figure 4.7: *Pole zero map of $G_N(z)$ obtained by subspace based method.*

normalized frequency axis, pole and zero locations are given in Figure 4.7. There are zeros outside of the unit circle and they will be mapped to right-half plane when bilinear transform is applied. Thus, obtained transfer function is not minimum phase.

If $G_A(s)$ of (4.1) is taken as nominal plant, $G_{A,5\%}(s)$ and $G_{A,10\%}(s)$ are perturbed versions of $G_A(s)$. Perturbation is made on parameters ϵ_1 and ϵ_2. Figure 4.8 shows relative errors between nominal transfer function $G_A(s)$ and transfer functions $G_{A,5\%}(s)$ and $G_{A,10\%}(s)$ with $+5\%$ and $+10\%$ modified parameters. Top plot shows relative error between $G_A(s)$ and estimated transfer function $G_N(s)$. In order to clarify the notation relative errors are defined as follows

$$E^{rel}_{G_A,G_N}(j\omega) = \left|\frac{G_A(j\omega) - G_N(j\omega)}{G_N(j\omega)}\right|, \quad E^{rel}_{G_A,G_{A,5\%}}(j\omega) = \left|\frac{G_A(j\omega) - G_{A,5\%}(j\omega)}{G_{A,5\%}(j\omega)}\right|$$

$$\text{and} \quad E^{rel}_{G_A,G_{A,10\%}}(j\omega) = \left|\frac{G_A(j\omega) - G_{A,10\%}(j\omega)}{G_{A,10\%}(j\omega)}\right| \tag{4.2}$$

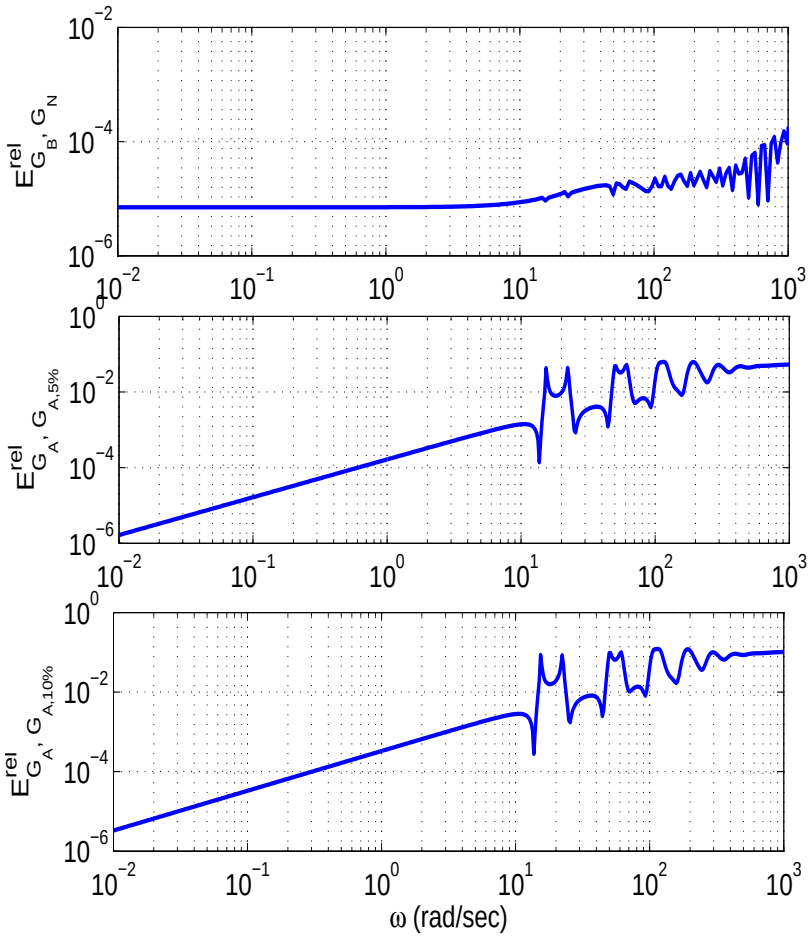

Figure 4.8: *Relative error between* $G_A(s)$ *and* $G_N(s)$, $G_{A,5\%}(s)$, $G_{A,10\%}(s)$ *respectively.*

4.1.2 Clamped-Free Beam

An infinite dimensional transfer function of a vibrating beam determined in [2] is taken into consideration as a second example. The transfer function is defined as

$$G_B(s) = \frac{N(s)}{EIm^3(s)D(s)}$$

where

$$m(s) = \left(\frac{-s^2}{EI + sc_dI}\right)^{1/4}$$

$$N(s) = \cosh(Lm(s))\sin(Lm(s)) - \sinh(Lm(s))\cos(Lm(s)),$$

$$D(s) = 1 + \cosh(Lm(s))\cos(Lm(s)).$$

Here, E and I are material constants and selected as $E = 5$ and $I = 1$; c_d is the damping constant and selected as $c_d = 0.01$. The beam is clamped at $x = 0$ and free at $x = L$ where $L = 5.5$.

At first, proposed algorithm is modified in order to make basis functions proper. G_N is selected as $G_N = \underline{1}|\Phi_1|$ for $N = 0$, where $\underline{1}$ is a vector consisting of ones and *Step 3* is ignored since $G_B(s)$ does not have a pole at the origin. The resulting $G_N(s)$ has a time delay term which is $h = 0.011$ sec. The degree of $P_N(s)$ is 18 and $G_N(s)$ does not contain an integral term. Bode plots of $G_B(s)$ and $G_N(s)$ are given in Figure 4.9, and the relative approximation error is shown in Figure 4.10. The Hankel singular values of G_N are shown in Figure 4.11.

Figure 4.12 gives relative error for three methods. Transfer functions have the same order of 18.

Transfer function $G_B(s)$ is defined as nominal plant. Perturbation is made on parameters E and I. Figure 4.13 shows relative errors between nominal transfer function $G_B(s)$ and transfer functions $G_{B,5\%}(s)$ and $G_{B,10\%}(s)$ with $+5\%$ and $+10\%$ modified parameters. E^{rel} values are defined as similar to those in (4.2).

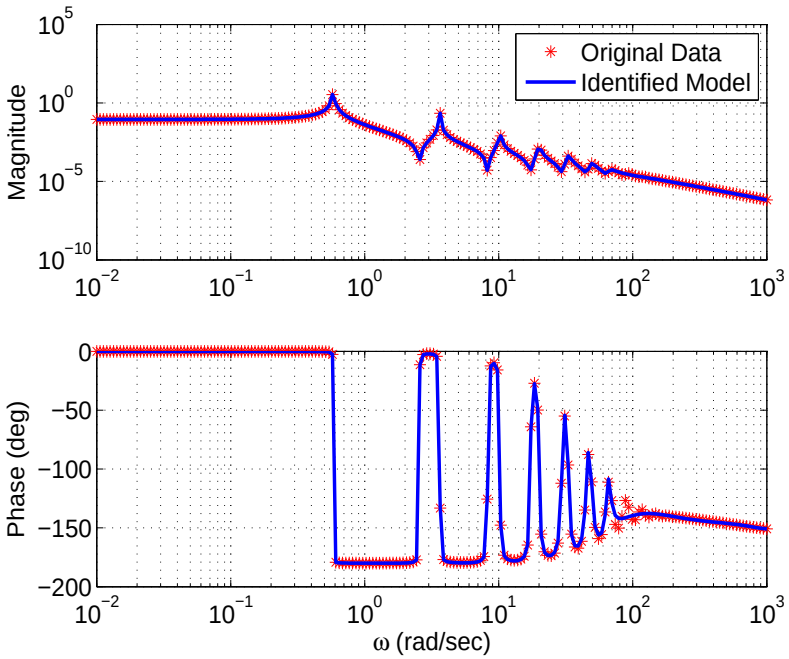

Figure 4.9: *Bode plots of* $G_B(s)$ *and the identified model* $G_N(s)$.

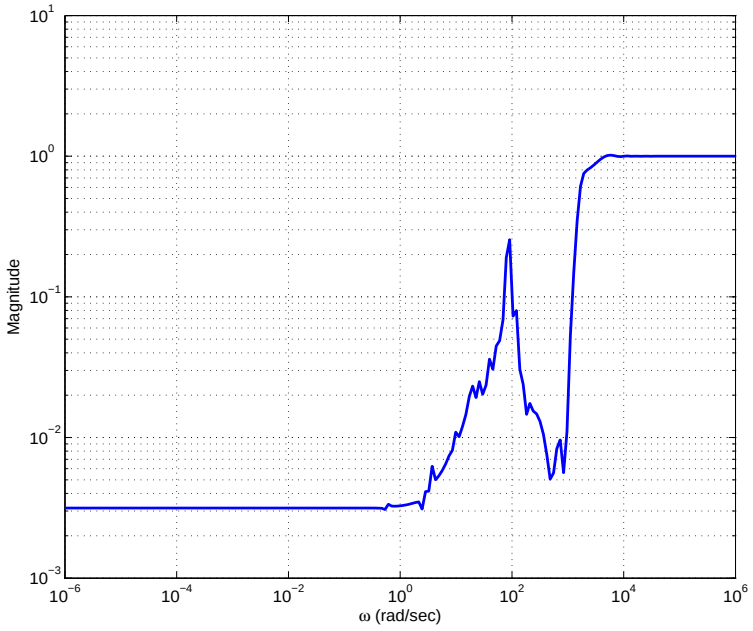

Figure 4.10: *Relative error* $|G_B(j\omega) - G_N(j\omega)|/|G_N(j\omega)|$.

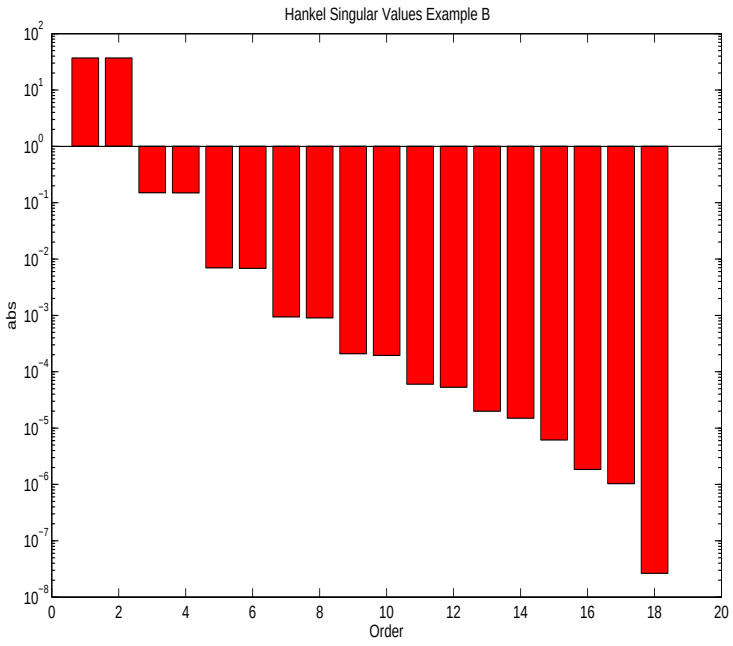

Figure 4.11: *Hankel singular values of $P_N(s)$, for example B.*

4.1.3 Free-Free Uniform Rod

The third example is an infinite dimensional transfer function which is a damped version of the wave system studied in [1]. This transfer function is expressed by delay terms as follows:

$$G_C(s) = \frac{c}{GI_p} \quad \frac{e^{-2\tau(1-\lambda)s} + 1}{(\epsilon s + 1) - e^{-2\tau s}} e^{-\tau\lambda s},$$

where parameters selected as $\lambda = 0.5$, $\tau = 0.0525$, $c = 0.2$, $G = 0.4$, $I_p = 2.5$ and $\epsilon = 0.001$. Resulting transfer function $P_N(s)$ is 54th order. The identified $G_N(s)$ contains an integral term and a time delay whose estimated value is 0.0261, which is very close to the exact value $\tau\lambda$. Bode plots of $G_C(s)$ and $G_N(s)$ are given in Figure 4.14, and the relative approximation error is shown in Figure 4.15. The Hankel singular values of G_N are shown in Figure 4.16.

NLLS approach and subspace based method applied to the third example. Figure 4.17 shows results. Fourier transform based method is omitted. Orders of the resulting transfer functions are 54, when integral part is separated.

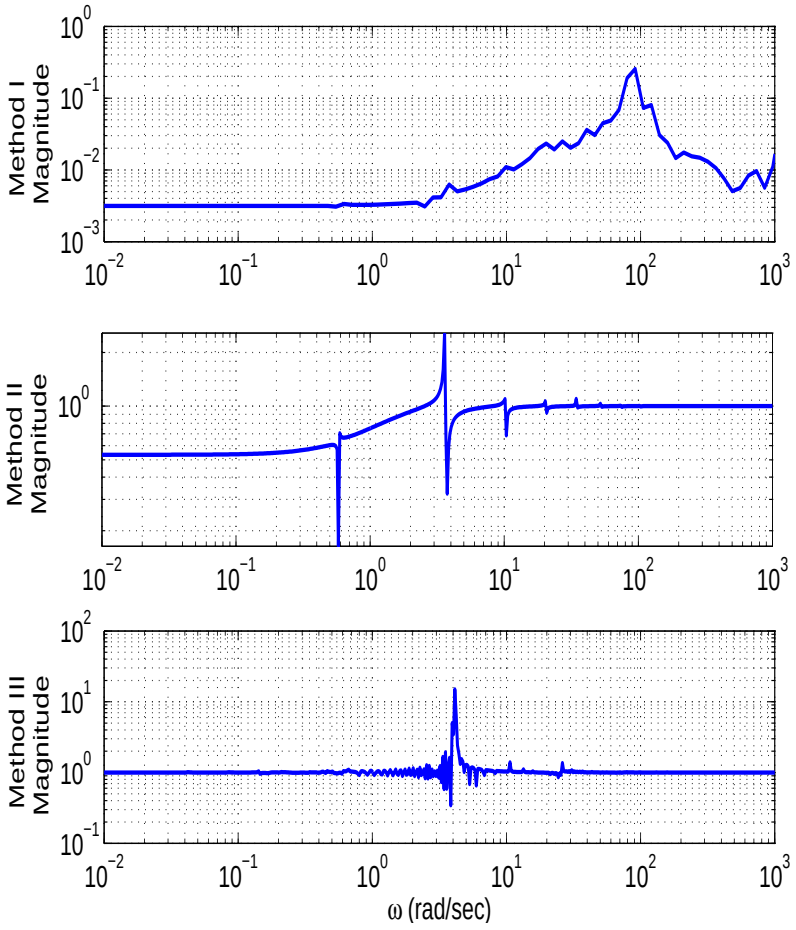

Figure 4.12: *Relative error comparison for three methods.*

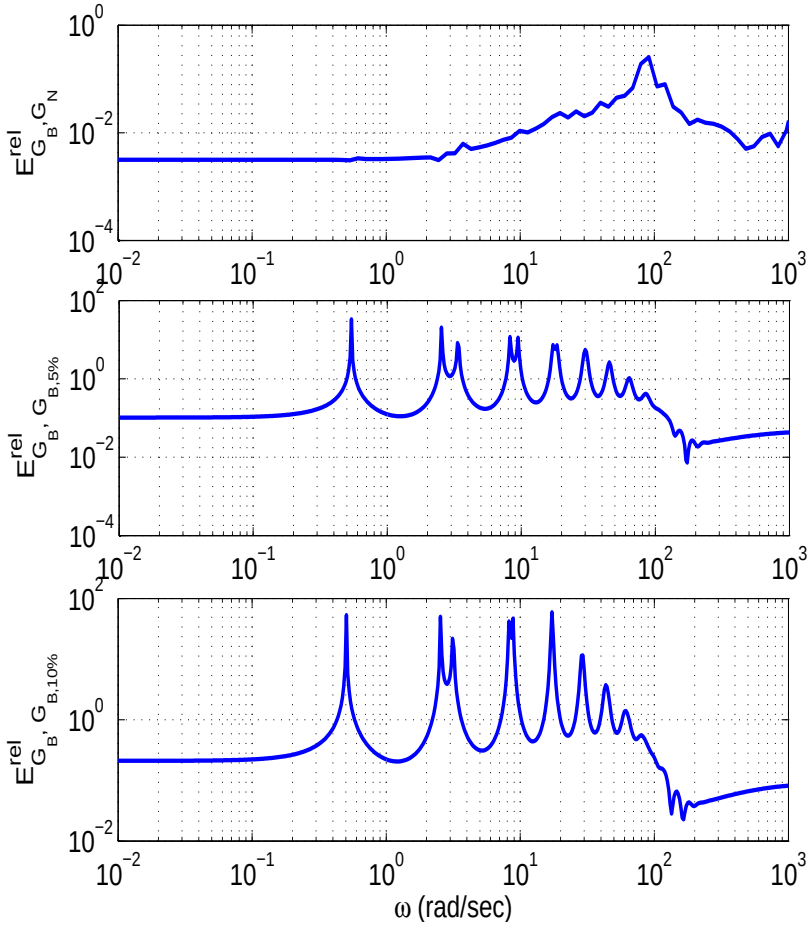

Figure 4.13: *Relative error between* $G_B(s)$ *and* $G_N(s)$, $G_{B,5\%}(s)$, $G_{B,10\%}(s)$ *respectively.*

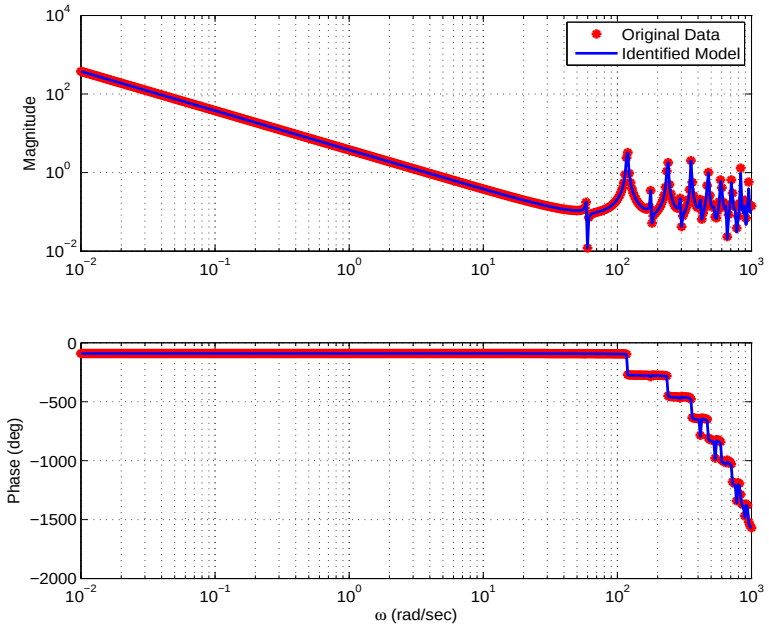

Figure 4.14: *Bode plots of* $G_C(s)$ *and the identified model* $G_N(s)$.

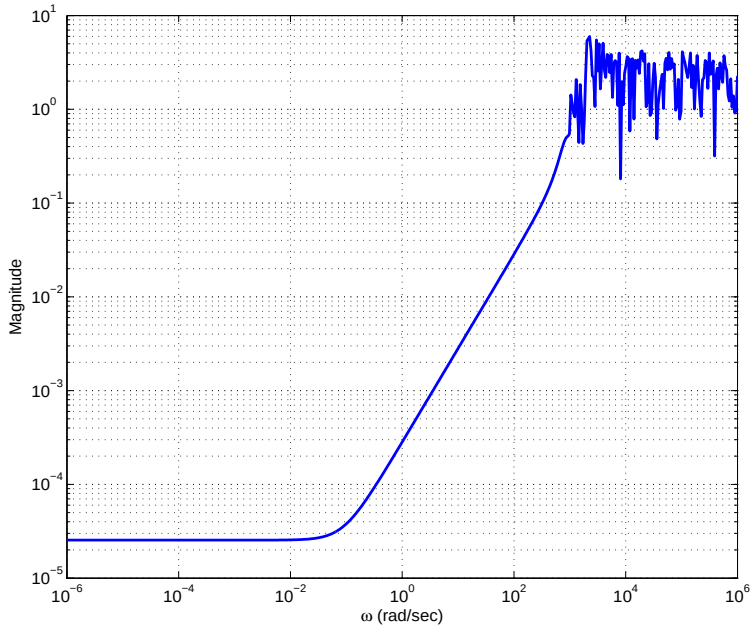

Figure 4.15: *Relative error* $|G_C(j\omega) - G_N(j\omega)|/|G_N(j\omega)|$.

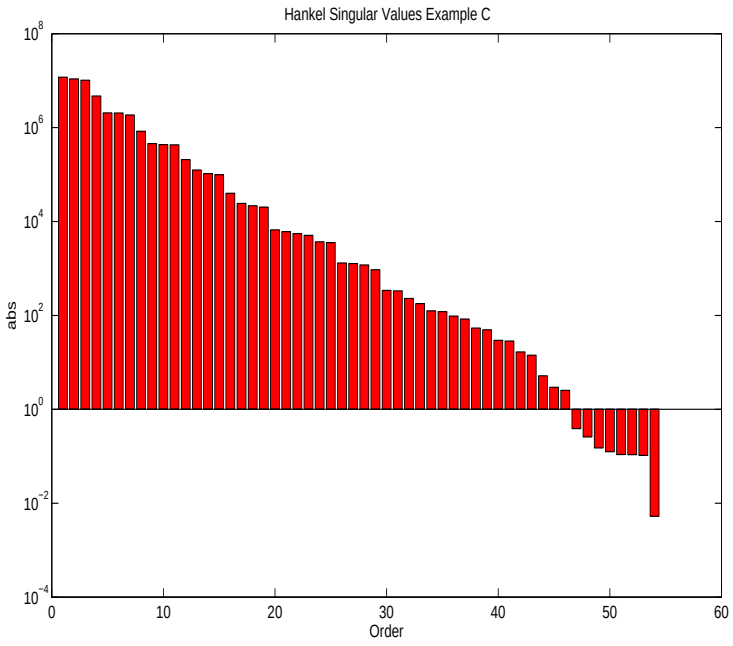

Figure 4.16: *Hankel singular values of $P_N(s)$ for example C.*

	NLLS method	Subspace based method
Example A	0.2549	2.5724
Example B	0.0027	0.0044
Example C	5.9884	7.3382

Table 4.1: *Error norms:* $\left\| \frac{G(\omega)-G_N(\omega)}{G_N(\omega)} \right\|_\infty$

Pole zero map in z-domain is given in Figure 4.18. Subspace method gives right-half plane zeros. Estimated transfer function by Method B is not minimum phase.

Transfer function $G_C(s)$ is defined as nominal plant. Perturbation is made on parameters τ and λ. Figure 4.19 shows relative errors between nominal transfer function $G_C(s)$ and transfer functions $G_{C,5\%}(s)$ and $G_{C,10\%}(s)$ with $+5\%$ and $+10\%$ modified parameters.

Table 4.1.3 displays maximum relative errors for three examples.

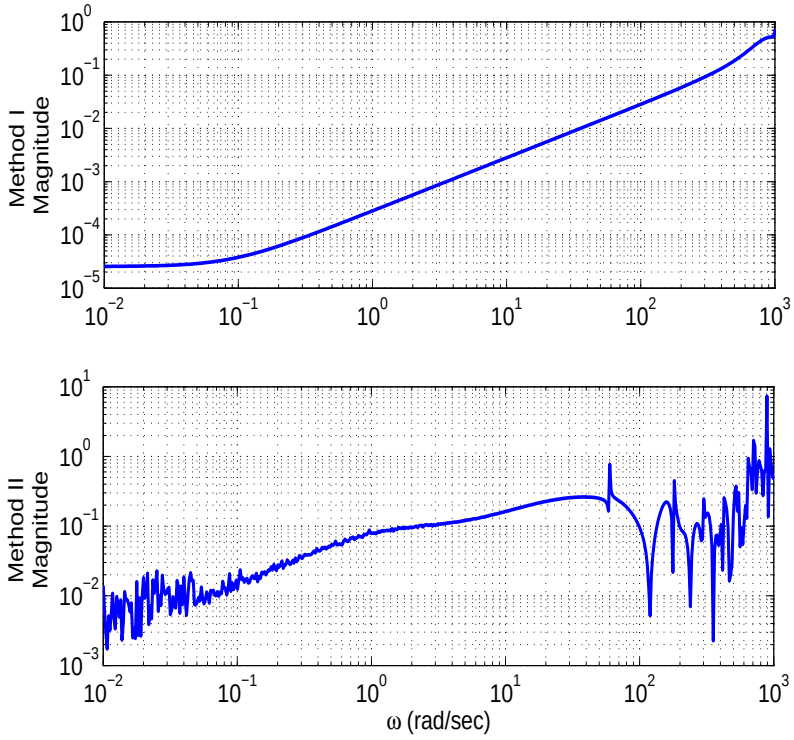

Figure 4.17: *Relative error comparison for two methods.*

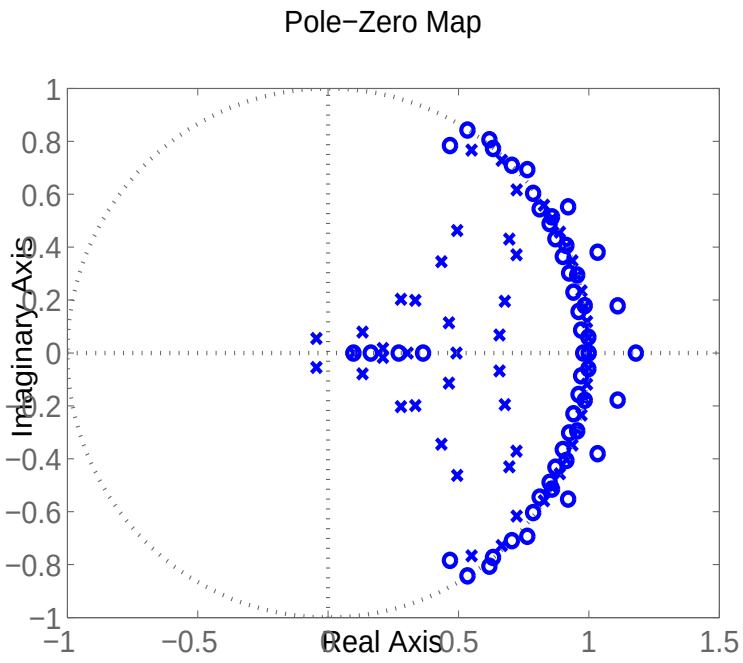

Figure 4.18: *Pole zero map of $G_N(z)$ obtained by subspace based method.*

4.2 Alternative Basis Function Example

Sequential data selection and optimization method is applied to a fractional order transfer function which is also infinite dimensional. Frequency response of $G(s)$

$$G(s) = \frac{1}{\sqrt{s}} \tag{4.3}$$

is calculated for (ω_i) between 10^{-5}rad/sec and 10^{5}rad/sec. Results from NLLS based approach (Method 1) are compared to the ones those from [33] (Method 2) in which continued fraction expansion of square root function is used and [40] (Method 3) using Regular Newton Process. In Figure 4.20, plots show error defined as

$$E(j\omega) = |G_D(j\omega) - G_N(j\omega)|\,.$$

where $G_D(s)$ is

$$G_D(s) = \frac{1}{1+\sqrt{s}}$$

Figure 4.21 shows the error between transfer function of $1/\sqrt{}(s)$ in closed

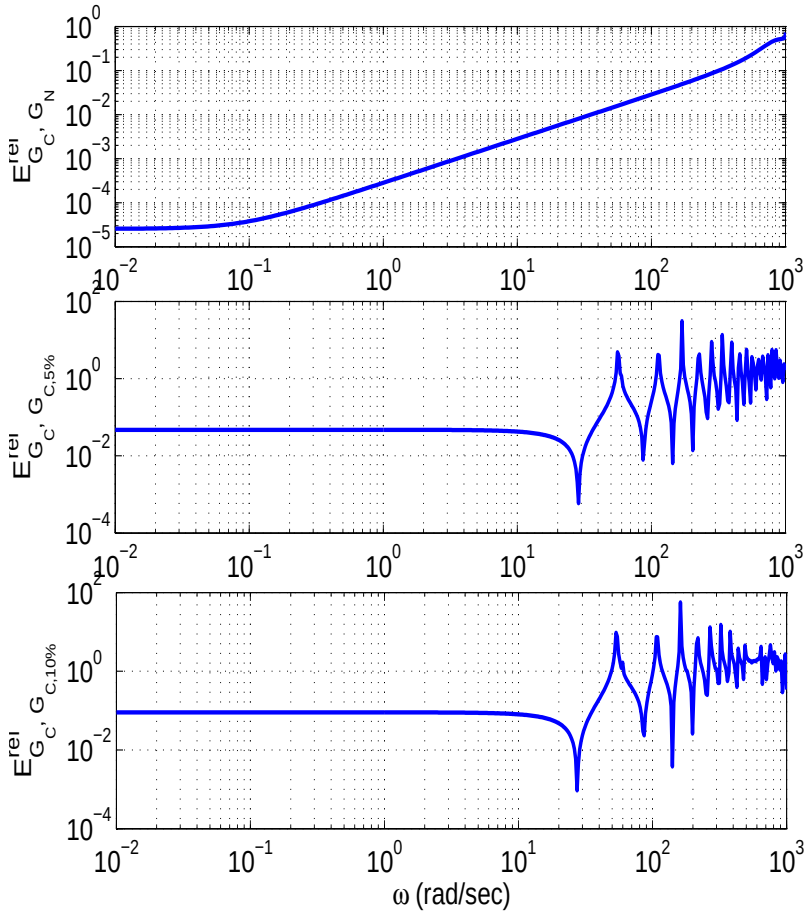

Figure 4.19: *Relative error between* $G_C(s)$ *and* $G_N(s)$, $G_{C,5\%}(s)$, $G_{C,10\%}(s)$ *respectively.*

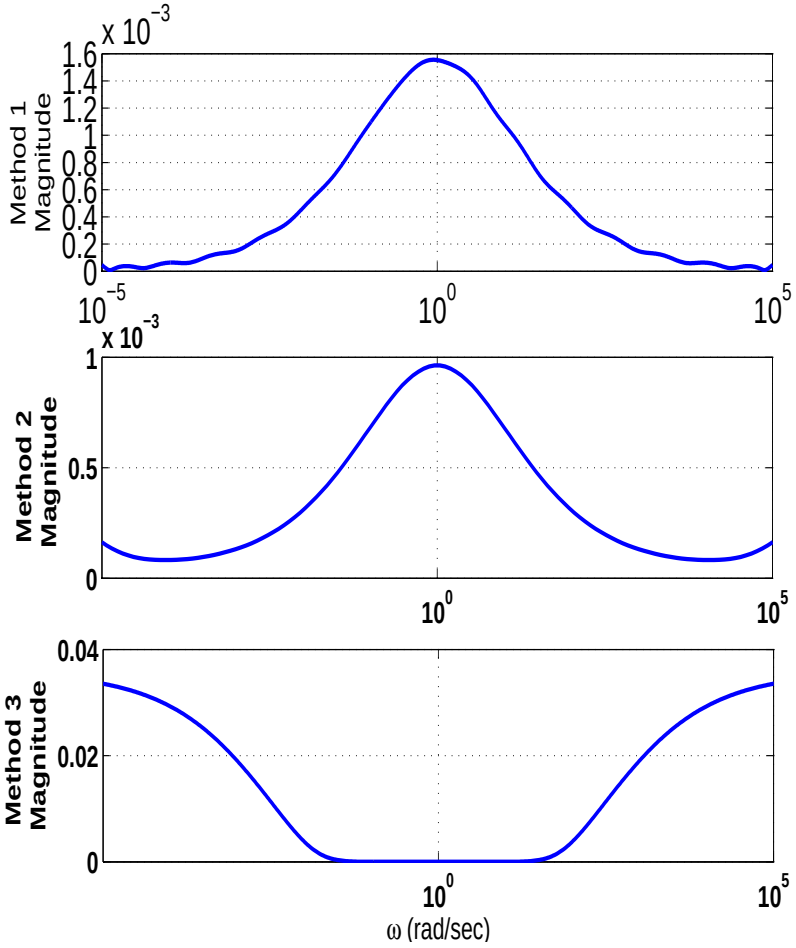

Figure 4.20: *Comparison of $E(j\omega)$ for three methods.*

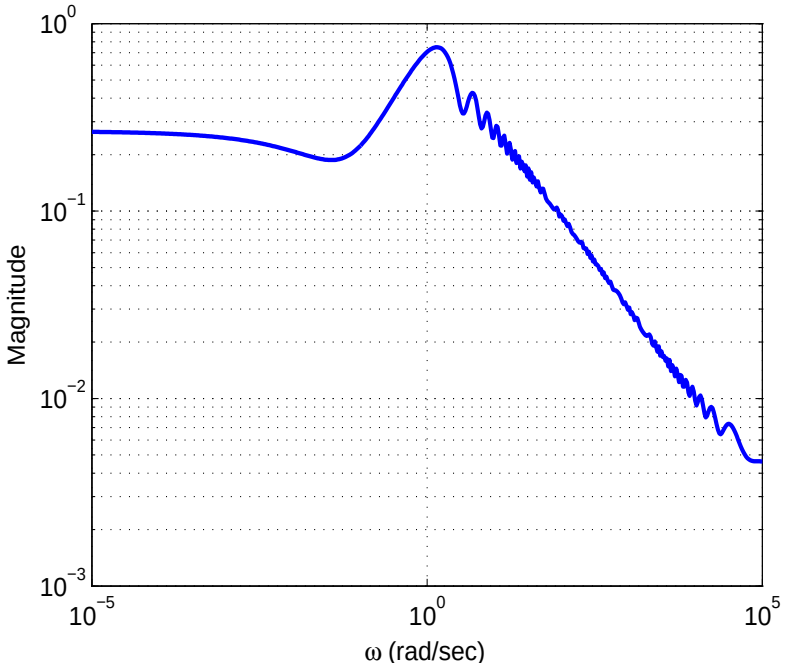

Figure 4.21: *Error between $G_D(s)$ and model identified by Method C.*

loop and estimated transfer function by using Fourier Transform based method, Method C, of previous section.

Chapter 5

CONCLUSIONS

In this thesis, a numerical method is proposed that aims to represent an infinite dimensional transfer function by a finite dimensional model and a delay term. We considered infinite dimensional models defined by PDEs. Non-parametric identification using DFT and experimental procedure for a 'black box' model are illustrated. A data selection scheme is proposed and mature optimization methods are implemented. For this purpose a NLLS approach is proposed to minimize a cost function which is quadratic in relative error in the frequency response. Minimization problem is solved by finding appropriate parameters. A brief description of Newtonian optimization methods is given. The optimization is modified to handle some physical constraints: minimizer parameters are constrained by using the log-barrier method. Intuition behind constraint is such that resulting transfer function must be minimum phase. In order to improve the results a version of the Levenberg-Marquardt algorithm is used. The Levenberg-Marquardt method's main contribution is that convergence is guaranteed. Thus, algorithm inside the Inner Loop prevents infinite loops.

It is shown that the log-barrier method can be efficient since it is a simpler form of solution to a dual problem that aims minimizing a cost function when constraints are involved. Nevertheless, log-barrier can not handle the objective optimization problem's ill-condition for some cases where the damping coefficients are very small.

The results are illustrated with three different examples. From the relative error plots it is observed that the modeling is very good within the frequency range where frequency response data is taken (for all three examples this was between 10^{-2} rad/sec and 10^{3} rad/sec). However, the relative error becomes large outside this frequency band. Result are compared to subspace based method and FFT based method. Also, the Hankel singular value plots show that the order of $P_N(s)$ can be further reduced. Nevertheless, having a large number poles and zeros close to the imaginary axis forces P_N to be of relatively large degree.

A version having an alternative basis function is implemented and applied to a fractional order system. Results are given and compared to two different methods. For this example we have encountered large number of iterations that lasts for a long period of time. On the other hand, when compared to other methods, NLLS approach gives accurate results despite the fact that parameters in the system appear in a non-linear fashion.

Bibliography

[1] N. Raskin and Y. Halevi, "Control of flexible structures governed by the wave equation," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 3, pp. 2486–2491, IEEE, 2001.

[2] R. Curtain and K. Morris, "Transfer functions of distributed parameter systems: A tutorial," *Automatica*, vol. 45, no. 5, pp. 1101–1116, 2009.

[3] K. Lenz and H. Özbay, "Analysis and robust control techniques for an ideal flexible beam," *Control and Dynamic Systems*, vol. 57, pp. 369–369, 1993.

[4] G. F. Franklin, M. L. Workman, and D. Powell, *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., 3rd ed., 1997.

[5] P. Stoica and T. Söderström, "A useful input parameterization for optimal experiment design," *IEEE Transactions on Automatic Control*, vol. 27, no. 4, pp. 986–989, 1982.

[6] R. K. Mehra, "Optimal inputs for linear system identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 192–200, 1974.

[7] L. Ljung, "System identification: theory for the user," *Prentice Hall Series, New Jersey*, vol. 7632, 1987.

[8] K. J. Åström and B. Wittenmark, *Adaptive Control*. Courier Dover Publications, 2013.

[9] R. L. Kosut and B. D. Anderson, "Least-squares parameter set estimation for robust control design," in *American Control Conference, 1994*, vol. 3, pp. 3002–3006, IEEE, 1994.

[10] R. L. Kosut, "Uncertainty model unfalsification: A system identification paradigm compatible with robust control design," in *Proceedings of the 34th IEEE Conference on Decision and Control, 1995.*, vol. 4, pp. 3492–3497, IEEE, 1995.

[11] R. L. Kosut, "System identification for robust control design.," tech. rep., DTIC Document, 1996.

[12] R. L. Kosut, "Iterative adaptive robust control via uncertainty model unfalsification," in *Proc. of 1996 IFAC World Congress*, pp. 91–96, 1996.

[13] W. S. Lee, B. D. Anderson, I. M. Mareels, and R. Kosut, "On some key issues in the windsurfer approach to adaptive robust control," *Automatica*, vol. 31, no. 11, pp. 1619–1636, 1995.

[14] G. Gu, P. P. Khargonekar, and E. B. Lee, "Approximation of infinite-dimensional systems," *IEEE Transactions on Automatic Control*, vol. 34, pp. 610–618, 1989.

[15] T. McKelvey, H. Akçay, and L. Ljung, "Subspace-based multivariable system identification from frequency response data," *IEEE Transactions on Automatic Control*, vol. 41, pp. 960–979, 1996.

[16] H. Akçay, "Subspace-based spectrum estimation in frequency-domain by regularized nuclear norm minimization," *Signal Processing*, vol. 99, pp. 69–85, 2014.

[17] J. H. Friedman and P. P. Khargonekar, "A comparative applications study of frequency domain identification techniques," in *Proceedings of American Control Conference*, 1995.

[18] L. Ljung, "Some results on identifying linear systems using frequency domain data," in *IEEE Conference on Decision and Control*, vol. 4, pp. 3534–3534, 1993.

[19] E.-W. Bai, "Frequency domain identification of hammerstein models," in *Block-oriented Nonlinear System Identification*, pp. 161–180, Springer, 2010.

[20] G. C. Goodwin, M. Gevers, and B. Ninness, "Identification and robust control: Bridging the gap," in *Proc. of the 7th IEEE Mediterranian Conference on Control and Automation*, 1999.

[21] G. J. Balas and J. C. Doyle, "Identification of flexible structures for robust control," *IEEE Control Systems Magazine*, vol. 10, no. 4, pp. 51–58, 1990.

[22] P. M. Van Den Hof and R. J. Schrama, "Identification and controlclosed-loop issues," *Automatica*, vol. 31, no. 12, pp. 1751–1770, 1995.

[23] B. Boulet and B. A. Francis, "Consistency of open-loop experimental frequency-response data with coprime factor plant models," *IEEE Transactions on Automatic Control*, vol. 43, no. 12, pp. 1680–1691, 1998.

[24] L. N. Trefethen, *Approximation Theory and Approximation Practice*. Siam, 2013.

[25] D. Enns, H. Özbay, and A. Tannenbaum, "Abstract model and controller design for an unstable aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 15, pp. 498–508, 1992.

[26] M. Karkoub, G. Balas, K. Tamma, and M. Donath, "Robust control of flexible manipulators via μ-synthesis," *Control Engineering Practice*, vol. 8, pp. 725–734, 2000.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009.

[28] O. Demir, U. Taşdelen, G. Kuralay, H. Özbay, and A. B. Özgüler, "Robust control oriented system identification of a flexible robot arm," in *4th International Conference on Control and Optimization with Industrial Applications, Bulgaria*, 2013.

[29] O. Demir and H. Özbay, "On reduced order modeling of flexible structures from frequency response data," in *13th European Control Conference, Strasbourg, France*, pp. 1133–1138, 2014.

[30] A. E. Karagül, O. Demir, and H. Özbay, "Computation of optimal $\mathcal{H}^\infty$ controllers and approximations of fractional order systems: A tutorial review," *Applied and Computational Mathematics*, pp. 261–288, 2013.

[31] J. E. Dennis Jr and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, vol. 16. SIAM, 1996.

[32] A.-M. Wazwaz, *Partial Differential Equations and Solitary Waves Theory*. Springer, 2010.

[33] K. Matsuda and H. Fujii, "$\mathcal{H}^\infty$ optimized wave-absorbing control-analytical and experimental results," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 6, pp. 1146–1153, 1993.

[34] M. I. Lourakis, "A brief description of the Levenberg-Marquardt algorithm implemented by levmar," *Foundation of Research and Technology*, pp. 1–6, 2005.

[35] Y.-X. Yuan, "A review of trust region algorithms for optimization," in *ICIAM*, vol. 99, pp. 271–282, 2000.

[36] F. V. Berghen, "Levenberg-Marquardt algorithms vs trust region algorithms," *IRIDIA, Université Libre de Bruxelles*, 2004.

[37] M. Bydder, "Solution of a complex least squares problem with constrained phase," *Linear Algebra and its Applications*, vol. 433, pp. 1719–1721, 2010.

[38] S. Gugercin and A. C. Antoulas, "A survey of model reduction by balanced truncation and some new results," *International Journal of Control*, vol. 77, pp. 748–766, 2004.

[39] G. Flagg, C. A. Beattie, and S. Gugercin, "Interpolatory $\mathcal{H}_\infty$ model reduction," *Systems & Control Letters*, vol. 62, no. 7, pp. 567–574, 2013.

[40] G. Carlson and C. Halijak, "Approximation of fractional capacitors by a regular newton process," *IEEE Transactions on Circuit Theory*, vol. 11, pp. 210–213, 1964.

Appendix A

Code

MATLAB® code: main.m

```
1  G = G_A % G_B
2  omega = logspace(-2, 3, 500);
3
4  start_ = 1;
5  step_ = 1;
6
7  [cost, coeffs_int, frep] = cost_and_coeffs_n(G, omega,
       1, 5, [], 1);
8
9  s_ = 1j*omega;
10
11 G_apprx = tf_approx(s_, coeffs_int, []);
12
13 figure;
14 subplot(2, 1, 1);
15 loglog(omega(start_:step_:end), abs(G_apprx(start_:
       step_:end)));
16 hold on;
```

```
17 loglog(omega(start_:step_:end), abs(G(start_:step_:end)
     ), 'r');
18
19 subplot(2, 1, 2);
20 semilogx(omega(start_:step_:end), unwrap(angle(G_apprx(
     start_:step_:end))));
21 hold on;
22 semilogx(omega(start_:step_:end), unwrap(angle(G(start_
     :step_:end))), 'r');
23
24 coeffs2 = coeffs_int;
25 cost_all = Inf;
26 NN = 9;
27 g = 0;
28 w_index_pre = 0;
29 while (g < NN || cost_all > 1e-2)
30
31 H_err = (G-G_apprx)./G_apprx;
32
33 w_maxerr_index = find(abs(H_err) == max(abs(H_err)));
34 w_maxerr = omega(w_maxerr_index);
35
36 if (w_maxerr_index > w_index_pre)
37     w_index_pre = w_maxerr_index;
38 end
39
40 [cost2, coeffs2, frep] = cost_and_coeffs_n_noint(G,
     omega, w_index_pre, w_maxerr, coeffs2);
41 coeffs2
42
43 G_apprx = tf_approx_noint(s_, coeffs2, []);
44
45 hold on;
```

```
46 subplot(2, 1, 1);
47 loglog(omega(start_:step_:end), abs(G(start_:step_:end)
     ), 'r*', 'markers', 7);
48 hold on;
49 loglog(omega(start_:step_:end), abs(G_apprx(start_:
     step_:end)), 'LineWidth', 2);
50
51 subplot(2, 1, 2);
52 semilogx(omega(start_:step_:end), unwrap(angle(G(start_
     :step_:end))), 'r*', 'markers', 7);
53 hold on;
54 semilogx(omega(start_:step_:end), unwrap(angle(G_apprx(
     start_:step_:end))), 'LineWidth', 2);
55
56 g = g+1
57
58 coeffs_all = coeffs2
59
60 end
```

MATLAB® code: cost_and_coeffs.m

```
1 function [ cost, coeffs, frep ] = cost_and_coeffs_n(H,
     omega, start, n, ctnow, sel)
2 %COST_AND_COEFFS_N
3 % calculates cost of optimization problem and
4 % coefficient found until the current step of main loop
5 % H is collected freq response
6 % omega is corresponding frequency axis points
7 % start gives the index of selected part of freq
     response data
8 % sel is given to select optimization of the integral
     part of transfer function or resonance and
9 % anti-resonance terms
```

```
10 % returns current cost
11 % returns current coefficients
12
13     if (nargin > 5 && sel == 1)
14         [cost, coeffs, frep] =
               integrator_cost_and_coeffs(H, omega, start,
               n);
15     else
16         [cost, coeffs, frep] =
               res_antires_cost_and_coeffs(H, omega, start,
                n, ctnow);
17     end
18 end
19
20 function [cost, coeffs, frep] =
      integrator_cost_and_coeffs(H, omega, start, n)
21 tol1 = 1e-15;
22 eps = 0;
23
24 ni = 1;
25 Ni = 1;
26 start_dom = start;
27
28 while (Ni+ni <= n+2)
29
30     if (Ni+ni > n)
31         Ni = n;
32         ni = 1;
33     end
34
35     fin = start_dom+Ni+ni-2;
36     start = start_dom;
37     s_ = 1j*omega(start:fin);
```

```
38      Hi_u = H(start:fin);
39
40      if (Ni == 1)
41          beta_i_1 = [0. abs(Hi_u(1)*1j*s_(1)) 0 0 0 0]';
42      end
43
44      beta_i = beta_i_1;
45
46      t_ = 1200;
47      fi = exp(-beta_i(1)*s_).*(beta_i(2))./s_;
48
49      Jr = create_jacob(s_, [], beta_i, Hi_u, fi, 1200);
50      Jr = Jr(:, 2:end);
51
52      k = 0;
53      v = 2;
54      A = real(Jr'*Jr);
55      eps_p = (Hi_u-fi)./fi-1/t_*log(beta_i(2));
56      g = real(Jr'*eps_p);
57      stop = 1;
58
59      if (max(abs(g)) < tol1)
60          stop = 0;
61      end
62      tau = 1;
63      mu = tau*max(diag(A));
64      rho = 0;
65      while (stop)
66          k = k+1;
67          if (mod(k, 1000) == 0)
68              tol1 = tol1*10;
69          end
70          %     rho = 0;
```

```
71          %     while (rho <= 0 || stop)
72          delta_beta = pinv(A+mu*eye(size(A)))*g;
73
74          if (norm(delta_beta) < tol1*norm(beta_i))
75              stop = 0;
76          else
77              beta_i_1 = beta_i+delta_beta;
78              fi_new = exp(-beta_i_1(1)*s_).*(beta_i_1(2)
                    )./s_;
79              eps_p_new = (Hi_u-fi_new)./fi_new-1/t_*log(
                    beta_i_1(2));
80
81              rho = (norm(eps_p)^2-norm(eps_p_new)^2)/(
                    delta_beta'*(mu*delta_beta+g));
82
83              if (rho > 0)
84                  beta_i = beta_i_1;
85
86                  %             Jri1 = -s_.*exp(-beta_i
                        (1)*s_).*(beta_i(2)*s_+beta_i(3))./
                        s_;
87                  %             Jri2 = exp(-beta_i(1)*s_)
                        .*zeros(length(s_), 1);
88                  %             Jri3 = exp(-beta_i(1)*s_)
                        .*1./s_;
89
90                  Jr = create_jacob(s_, [], beta_i, Hi_u,
                         fi, 1200);
91                  Jr = Jr(:, 2:end);
92
93                  A = real(Jr'*Jr);
94
```

```
95                  fi = exp(-beta_i(1)*s_).*(beta_i(2))./
                       s_;
96                  eps_p = (Hi_u-fi)./fi-1/t_*log(beta_i
                       (2));
97                  g = real(Jr'*eps_p);
98
99                  if (max(abs(g)) < tol1)
100                     stop = 0;
101                 end
102
103                 if (norm(eps_p)^2 < tol1)
104                     stop = 0;
105                 end
106
107                 mu = mu*max(1/3, 1-(2*rho-1)^3);
108                 v = 2;
109             else
110                 mu = mu*v;
111                 v = 2*v;
112             end
113         end
114         %     end
115     end
116
117     Ni = Ni+1;
118     ni = ni+1;
119 end
120
121 coeffs = [beta_i_1; 0; 0; 0; 0];
122 s_ = 1j*omega(1:fin);
123 fi = exp(-coeffs(1)*s_).*(coeffs(2))./s_;
124 cost = cost_func(H(1:fin), fi);
125 s_ = 1j*omega;
```

```
126 fi = exp(-coeffs(1)*s_).*(coeffs(2))./s_;
127 frep = fi;
128 % cost = cost_func(H(1:end), fi);
129 end
130
131
132 function [cost, coeffs, frep] =
      res_antires_cost_and_coeffs(H, omega, start, n,
      ctnow)
133
134 tol1 = 1e-20;
135 eps1 = 1e-12;
136 m_ = 1;
137 eps = 0;
138 mu1 = 20;
139 t_ = 1;
140 lb = 1e-4;
141 alpha = 1e-12;
142
143 ni = n;
144 Ni = 1;
145 start_dom = start;
146
147 while (Ni+ni <= n+1)
148
149     if (Ni+ni > n)
150         Ni = 1;
151         ni = n;
152     end
153
154     NN = length(H);
155
156     fin = start_dom;
```

```
157
158     if (fin > NN)
159         fin = NN;
160     end
161 %     start = start_dom;
162     start = 1;
163     s_ = 1j*omega(start:fin);
164     Hi_u = H(start:fin);
165     % Hi_u = Hwi_;
166
167 %     w_n_n = abs(omega(floor((fin+start_dom)/2)));
168     w_n_n = n;
169     w_n_d = w_n_n;
170
171     zeta_n = 0.5;
172     zeta_d = 0.5;
173
174     tt_ = 30;
175
176     if (Ni == 1)
177 %         beta_i_2 = [0 1 zeta_n w_n_n 1 zeta_d w_n_d]';
178         beta_i_2 = [1*tt_ zeta_n*sqrt(tt_) w_n_n*sqrt(
              tt_) 1*tt_ zeta_d*sqrt(tt_) w_n_d*sqrt(tt_)
              ]';
179     end
180
181 while (m_/t_ > eps1)
182
183     beta_i = beta_i_2;
184
185     fi = tf_approx(s_, ctnow, beta_i);
186
187     Jr = create_jacob(s_, ctnow, beta_i, Hi_u, fi, t_);
```

```
188     Jr = Jr(:, 2:end);
189     k = 0;
190     v = 2;
191     A = real(Jr'*Jr);
192     coeffs_constraint = reshape(ctnow(1:end, 1:end),
           [], 1);
193     coeffs_constraint(1) = [];
194     coeffs_constraint(2:5) = [];
195     eps_p = t_*(Hi_u-fi)./(fi)-sum(log(beta_i(1:end)))-
           sum(log(coeffs_constraint));
196     g = real(Jr'*eps_p);
197     stop = 1;
198
199     if (max(abs(g)) < tol1)
200         4
201         stop = 0;
202     end
203     tau = 0.1;
204     mu = tau*max(diag(A));
205     rho = 0;
206     beta_i_all = [reshape(ctnow, [], 1); beta_i];
207     beta_i_all(3:6) = [];
208     while (stop && k < Inf)
209         k = k+1;
210         if (mod(k, 1000) == 0)
211             tol1 = tol1*10;
212         end
213         %     rho = 0;
214         %     while (rho <= 0 || stop)
215         %     A
216         %     mu
217         mu;
218         if (isinf(mu) || isnan(mu))
```

```
219             55
220         end
221         if (sum(sum(isinf(A))) > 0 || sum(sum(isnan(A))
               ) > 0)
222             A
223         end
224         delta_beta = pinv(A+mu*eye(size(A)))*g;
225
226
227         if (norm(delta_beta) < tol1*norm(beta_i_all))
228             stop = 0;
229         else
230             size(delta_beta);
231             size(beta_i_all);
232             beta_i_2_all = beta_i_all+delta_beta;
233
234             if (beta_i_2_all(1) < 0)
235                 beta_i_2_all(1) = 0;
236             end
237
238
239
240
241             beta_i_2_all_dum = [beta_i_2_all(1:2);
                   zeros(4, 1); beta_i_2_all(3:end)];
242             ctnow = reshape(beta_i_2_all_dum, 6, []);
243             beta_i_2 = ctnow(:, end);
244             fi_new = tf_approx(s_, ctnow(:, 1:end-1),
                   beta_i_2);
245
246             coeffs_constraint = reshape(ctnow(1:end, 1:
                   end), [], 1);
247             coeffs_constraint(1) = [];
```

```
248             coeffs_constraint(2:5) = [];
249             coeffs_constraint;
250             eps_p_new = t_*(Hi_u-fi_new)./(fi_new)-sum(
                   log(beta_i_2(1:end)))-sum(log(
                   coeffs_constraint));
251
252             rho = (norm(eps_p)^2-norm(eps_p_new)^2)/(
                   delta_beta'*(mu*delta_beta+g));
253
254             if (rho > 0)
255                 beta_i_all = beta_i_2_all;
256
257
258                 Jr = create_jacob(s_, ctnow(:, 1:end-1)
                       , beta_i, Hi_u, fi_new, t_);
259                 Jr = Jr(:, 2:end);
260
261                 A = real(Jr'*Jr);
262
263
264                 beta_i_all_dum = [beta_i_2_all(1:2);
                       zeros(4, 1); beta_i_2_all(3:end)];
265                 ctnow = reshape(beta_i_all_dum, 6, []);
266                 beta_i = ctnow(:, end);
267                 fi = tf_approx(s_, ctnow(:, 1:end-1),
                       beta_i);
268
269                 coeffs_constraint = reshape(ctnow(1:end
                       , 1:end), [], 1);
270                 coeffs_constraint(1) = [];
271                 coeffs_constraint(2:5) = [];
272
```

```
273                 eps_p = t_*(Hi_u-fi)./(fi)-sum(log(
                       beta_i(1:end)))-sum(log(
                       coeffs_constraint));
274                 g = real(Jr'*eps_p);
275
276                 if (max(abs(g)) < tol1)
277                     6
278                     stop = 0;
279                 end
280
281                 if (norm(eps_p)^2 < tol1)
282                     7
283                     stop = 0;
284                 end
285
286                 mu = mu*max(1/3, 1-(2*rho-1)^3);
287                 v = 2;
288             else
289                 mu = mu*v;
290                 v = 2*v;
291             end
292         end
293         %     end
294     end
295
296     Ni = Ni+1;
297     ni = ni+1;
298     ctnow = ctnow(:, 1:end-1);
299
300
301 beta_i_2_all_dum = [beta_i_2_all(1:2); zeros(4, 1);
       beta_i_2_all(3:end)];
302 coeffs = reshape(beta_i_2_all_dum, 6, []);
```

```
303 beta_i_2 = coeffs(:, end);
304 ctnow = coeffs(:, 1:end-1);
305 t_ = t_*mu1;
306 end
307
308 end
309 [R, C] = size(coeffs);
310
311 for k = 1:0
312     coeffsi = coeffs(:, k);
313     if (sum(coeffsi(1:end) < -eps) > 0)
314         coeffs = NaN;
315         cost = Inf;
316         frep = NaN;
317         return;
318     end
319
320     if (sum(coeffsi(5:end)) == 0 && k > 1)
321         coeffs = NaN;
322         cost = Inf;
323         frep = NaN;
324
325         return;
326     end
327 end
328
329
330 ctnow = coeffs(:, 1:end-1);
331 beta_i_2 = coeffs(:, end);
332 s_ = 1j*omega(1:fin);
333 fi = tf_approx(s_, ctnow, beta_i_2);
334 cost = cost_func(H(1:fin), fi(1:fin));
335 s_ = 1j*omega;
```

```
336 frep = tf_approx(s_, ctnow, beta_i_2);
337 % cost = cost_func(H(1:end), fi);
338 end
339
340
341 function [cost] = cost_func(H, H_new)
342     cost = sqrt(sum(abs((H-H_new)./H_new).^2)/length(H)
        );
343 %     cost = max(abs((H-H_new)./H_new));
344 end
```

MATLAB® code: create_jacob.m

```
1  function [ J ] = create_jacob( s_, ctnow, cnewinit,
     Hi_u, fi, t_ )
2  % Calculates jacobian of current transfer function
3  % Jacobian is needed for calculating Newton step
4  % Jacobian is calculated by using analytic derivative
     of cost function
5  % s_ is frequency axis data
6  % ctnow is array of coefficients found until the
     current step
7  % Hi_u is the collected frequency response
8  % fi is the frequency response of the approximating
     transfer function
9  % t_ is the coefficient comes from log-barrier method
10 % returns Jacobian matrix
11 [R, C] = size(ctnow);
12 N = length(s_);
13
14 J = zeros(N, 1);
15 alpha = 0;
16 ll_ = 1;
17
```

```
18 for k = 1:C
19     beta_i = ctnow(:, k);
20     if (k == 1)
21         fik = exp(-beta_i(1)*s_).*beta_i(2)./s_;
22         Jri1 = -s_.*exp(-beta_i(1)*s_).*(beta_i(2))./s_
               .*(Hi_u./fi./fik);
23
24         Jri2 = t_*exp(-beta_i(1)*s_).*1./s_.*(Hi_u./fi
               ./fik)-ll_*1/beta_i(2);
25         Jr = [Jri1 Jri2];
26     else
27         fik = (beta_i(1)*s_.^2+2*beta_i(2)*beta_i(3)*s_
               +beta_i(3)^2)./(beta_i(4)*s_.^2+2*beta_i(5)*
               beta_i(6)*s_+beta_i(6)^2+alpha);
28
29
30         Jri2 = t_*s_.^2./(beta_i(4)*s_.^2+2*beta_i(5)*
               beta_i(6)*s_+beta_i(6)^2+alpha).*Hi_u./fi./
               fik-ll_*1/beta_i(1);
31         Jri3 = t_*2.*(beta_i(3)*s_)./(beta_i(4)*s_
               .^2+2*beta_i(5)*beta_i(6)*s_+beta_i(6)^2+
               alpha).*Hi_u./fi./fik-ll_*1/beta_i(2);
32         Jri4 = t_*2.*(beta_i(2)*s_+beta_i(3))./(beta_i
               (4)*s_.^2+2*beta_i(5)*beta_i(6)*s_+beta_i(6)
               ^2+alpha).*Hi_u./fi./fik-ll_*1/beta_i(3);
33
34         Jri5 = t_*(-s_.^2).*(beta_i(1)*s_.^2+2*beta_i
               (2)*beta_i(3)*s_+beta_i(3)^2)./(beta_i(4)*s_
               .^2+2*beta_i(5)*beta_i(6)*s_+beta_i(6)^2+
               alpha).^2.*Hi_u./fi./fik-ll_*1/beta_i(4);
```

```
35          Jri6 = t_*(-s_*2*beta_i(6)).*(beta_i(1)*s_
               .^2+2*beta_i(2)*beta_i(3)*s_+beta_i(3)^2)./(
               beta_i(4)*s_.^2+2*beta_i(5)*beta_i(6)*s_+
               beta_i(6)^2+alpha).^2.*Hi_u./fi./fik-ll_*1/
               beta_i(5);
36          Jri7 = t_*-2.*(s_*beta_i(5)+beta_i(6)).*(beta_i
               (1)*s_.^2+2*beta_i(2)*beta_i(3)*s_+beta_i(3)
               ^2)./(beta_i(4)*s_.^2+2*beta_i(5)*beta_i(6)*
               s_+beta_i(6)^2+alpha).^2.*Hi_u./fi./fik-ll_
               *1/beta_i(6);
37
38          Jr = [Jri2 Jri3 Jri4 Jri5 Jri6 Jri7];
39      end
40      J = [J Jr];
41  end
42
43  beta_i = cnewinit;
44  if (length(beta_i) == 2)
45      fik = exp(-beta_i(1)*s_).*beta_i(2)./s_;
46      Jri1 = -s_.*exp(-beta_i(1)*s_).*(beta_i(2))./s_.*(
           Hi_u./fi./fik);
47  %          Jri2 = exp(-beta_i(1)*s_).*zeros(length(s_),
       1).*(Hi_u./fi.^2);
48      Jri2 = t_*exp(-beta_i(1)*s_).*1./s_.*(Hi_u./fi./fik
           )-ll_*1/beta_i(2);
49      J = [J Jri1 Jri2];
50  else
51      fik = (beta_i(1)*s_.^2+2*beta_i(2)*beta_i(3)*s_+
           beta_i(3)^2)./(beta_i(4)*s_.^2+2*beta_i(5)*
           beta_i(6)*s_+beta_i(6)^2+alpha);
52
53
```

```
54      Jri2 = t_*s_.^2./(beta_i(4)*s_.^2+2*beta_i(5)*
           beta_i(6)*s_+beta_i(6)^2+alpha).*Hi_u./fi./fik-
           ll_*1/beta_i(1);
55      Jri3 = t_*2.*(beta_i(3)*s_)./(beta_i(4)*s_.^2+2*
           beta_i(5)*beta_i(6)*s_+beta_i(6)^2+alpha).*Hi_u
           ./fi./fik-ll_*1/beta_i(2);
56      Jri4 = t_*2.*(beta_i(2)*s_+beta_i(3))./(beta_i(4)*
           s_.^2+2*beta_i(5)*beta_i(6)*s_+beta_i(6)^2+alpha
           ).*Hi_u./fi./fik-ll_*1/beta_i(3);
57
58      Jri5 = t_*(-s_.^2).*(beta_i(1)*s_.^2+2*beta_i(2)*
           beta_i(3)*s_+beta_i(3)^2)./(beta_i(4)*s_.^2+2*
           beta_i(5)*beta_i(6)*s_+beta_i(6)^2+alpha).^2.*
           Hi_u./fi./fik-ll_*1/beta_i(4);
59      Jri6 = t_*(-s_*2*beta_i(6)).*(beta_i(1)*s_.^2+2*
           beta_i(2)*beta_i(3)*s_+beta_i(3)^2)./(beta_i(4)*
           s_.^2+2*beta_i(5)*beta_i(6)*s_+beta_i(6)^2+alpha
           ).^2.*Hi_u./fi./fik-ll_*1/beta_i(5);
60      Jri7 = t_*-2.*(s_*beta_i(5)+beta_i(6)).*(beta_i(1)*
           s_.^2+2*beta_i(2)*beta_i(3)*s_+beta_i(3)^2)./(
           beta_i(4)*s_.^2+2*beta_i(5)*beta_i(6)*s_+beta_i
           (6)^2+alpha).^2.*Hi_u./fi./fik-ll_*1/beta_i(6);
61
62      J = [J Jri2 Jri3 Jri4 Jri5 Jri6 Jri7];
63  end
64
65  end
```

MATLAB® code: tf_approx.m

```
1  function [ fi ] = tf_approx( s_, ctnow, beta )
2  % Calculates frequency response of the approximating
       tranferfunction
3  % found until the current step of
```

```
4  % main loop
5  [R, C] = size(ctnow);
6  N = length(s_);
7
8  fi = ones(N, 1);
9
10 for k = 1:C
11     beta_i = ctnow(:, k);
12     if (k == 1)
13         fi = fi.*exp(-beta_i(1)*s_).*(beta_i(2))./s_;
14     else
15         fi = fi.*(beta_i(1)*s_.^2+2*beta_i(2)*beta_i(3)
               *s_+beta_i(3)^2)./(beta_i(4)*s_.^2+2*beta_i
               (5)*beta_i(6)*s_+beta_i(6)^2);
16     end
17 end
18
19 if (~isempty(beta))
20     beta_i = beta;
21
22     fi = fi.*(beta_i(1)*s_.^2+2*beta_i(2)*beta_i(3)*s_+
          beta_i(3)^2)./(beta_i(4)*s_.^2+2*beta_i(5)*
          beta_i(6)*s_+beta_i(6)^2);
23 end
```