Nonlinear Stochastic System Identification Techniques for Biological Tissues

by

Yi Chen

S.B. in Mechanical Engineering and S.B. in Economics,
Massachusetts Institute of Technology (2008)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author .....
Department of Mechanical Engineering
May 7, 2010

Certified by .....
Ian W. Hunter
Hatsopoulos Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
David E. Hardt
Chairman, Department Committee on Graduate Students

Nonlinear Stochastic System Identification Techniques for Biological Tissues

by

Yi Chen

Submitted to the Department of Mechanical Engineering
on May 7, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This research develops a device capable of measuring the nonlinear dynamic mechanical properties of human tissue *in vivo*. The enabling technology is the use of nonlinear stochastic system identification techniques in conjunction with a high bandwidth actuator to perturb the tissue. The desktop and handheld instruments used for this investigation were custom-built Lorentz force actuators which were able to measure the dynamic compliance between the input force and the output displacement. The actuators have a nominal stroke of 32 mm and were actuated with forces under 15 N. The design includes custom electronics and user software which collects and analyses the information.

This research also explores nonlinear stochastic system identification techniques that would be applicable to biological tissues. Several system identification techniques were used including linear, Wiener static nonlinear, Volterra kernel and partitioning techniques. Real time system identification and real time input generation schemes are also implemented. The mathematical formulation and implementation details of these techniques are also discussed. It was found that a simple linear stochastic system identification technique had a variance accounted for (VAF) of 70 to 75 %. More complicated representations using Volterra kernels or partitioning techniques had a VAF of 90 to 97 %. More complex nonlinear system identification techniques can not only capture more of the nonlinear dynamics but also capture those dynamics in an interpretable way.

Indentation, extension, and surface mechanics experiments were conducted to investigate the nonlinear mechanical compliance of skin *in vivo*. The techniques and devices used in this research can be applied directly to consumer product efficacy analysis, medical diagnosis as well as research in biomechanical tissues.

Thesis Supervisor: Ian W. Hunter
Title: Hatsopoulos Professor of Mechanical Engineering

Acknowledgments

I would like to express my gratitude towards my advisor Professor Ian W. Hunter for his guidance and support throughout this research. Professor Hunter was tireless in promoting the device and an endless resource of great ideas, advice and direction.

I would also like to thank Dr. Cathy Hogan for helping me set up forms for COUHES, patent applications, grant applications and providing me with advice on biology. I would also like to thank Dr. Lynette Jones for providing me advice in human testing and Ms. Kate Melvin for helping me manage materials for the lab.

When working with system identification techniques, the first person I would always turn to was Priam Pillai. I would like to thank him for helping me get started and for being a good springboard for new ideas. I would like to thank Bryan Ruddy and Brian Hemond for their advice on electronics and motor dynamics. I would also like to thank the other members of the lab, Miguel Saez, Adam Wahab, Scott McEuen, Jean Chang, Kerri Keng, and Eli Paster for their support and for volunteering to be subjects in my experiments.

I would like to thank my Father, Mother and Sister for their support throughout the process. Thank you for believing in me.

This research was supported in part through the National Science Foundation Fellowship and the MIT Institute Energy Fellowship sponsored by BP.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Measurement of the mechanical properties of human tissue provides valuable information for deriving mathematical models, creating injury repair techniques, developing tissue vascularization therapies, and diagnosing healthy from damaged tissues *in vivo* [25]. In addition, clinical instrumentation, such as those demonstrated in this thesis, can be used to standardize the qualitative measurements that physicians currently use to diagnose tissue diseases. A device with the ability to diagnose tissue diseases (*e.g.* Scleroderma, Myxoedema, connective tissue diseases [23, 28, 44]) or identify the presence of dehydration [41] can have a large societal impact in healthcare and large market impact in terms of tools that are available to clinicians.

The skin care market represents a $15.1 billion market with facial skin care at $3 billion alone in 2007 [85]. Trends in consumer skin care have shown the use of specific molecules and proteins like tensin [75] that are well known to cause collagen growth or increase skin suppleness in hydration and anti-aging products. Although there are standard testing devices for skin, industry specialists have expressed dissatisfaction with existing commercial products. A standardized measurement technique designed to assess the effectiveness of these products would be invaluable to this field.

In a clinical setting, the mechanical properties of skin and underlying tissue are assessed qualitatively through touch by dermatological specialists and cosmologists [109]. This, however, presents a problem in terms of passing information between different individuals or comparing measurements from different clinical studies for the

diagnosis of skin conditions. A device that is capable of making and standardizing the measurement of mechanical properties of biological materials in an expedient fashion is indispensable for this purpose. In addition, such a device is important for understanding mechanics for manufacturing artificial prosthetic tissue, for determining mechanical properties in locations that are difficult to palpate (such as in the colon during endoscopy), and determining parameters needed for needle-free injection [40].

Currently, there are a few commercial and research devices which try to address this issue with limited success. Several techniques including suction, indentation, torsion, extension, ballistometry, and wave propagation have been used to measure skin mechanics *in vivo* [28]. Commercial devices with varying levels of success include the Cutometer MKA580, DermaFlex, and Diastron dermal torque meter. For the most part, these devices only provide information about limited aspects of skin behavior which may not be enough to properly diagnose disease. Many of these devices also focus on only linear properties such as skin elasticity because linear parameters play the largest role in daily physiological stress levels [4]. A more complete picture can begin to arise when dynamic properties such as damping are taken into account. Although some of these properties only present themselves at nonphysiologic stress levels, they can more fully characterize the properties of the tissue and can be used as additional degrees of freedom in identification and diagnosis. Studies also show that skin is strain hardening [15], which means that the relationship between stress and strain is nonlinear. Although the properties of skin have been studied for many years in material science and biology, no other work has used the innovations in nonlinear stochastic system identification and signal processing to identify the nonlinear properties of biological tissues.

In order to identify these effects in a fast, robust, accurate, and low cost fashion, a high bandwidth perturbation system was designed. The information was analyzed using a series of nonlinear stochastic system identification techniques focused on short test times, fast computational times, and interpretability. Nonlinear systems, as opposed to linear systems, come in many varieties and there are many possible methods that can be used to describe them. A study on the relevant types of nonlinearities

is essential to reducing the number of possibilities. In some fields of study, such as optics, the form of the nonlinearity is restricted by photon combination and therefore the study of nonlinear systems can be formulated in a clear fashion. In the mechanics of tissue, however, the form of the dynamic nonlinearity is not currently well formulated. This work focuses on developing a nonlinear system identification framework that can be generalized for identifying dynamic tissue properties.

This work is organized in order to provide background and motivation, to describe the device and procedures, and to discuss well-known and novel system identification techniques. Background materials are described in Chapter one. Chapter two covers the mechanical, electrical, and software design of the device. Chapter three describes linear system identification theory, static nonlinear system identification methods and results that motivate the use of more advanced nonlinear system identification techniques that are described in Chapters four, five, and six. In addition to well-known nonlinear techniques like Volterra kernels, a few new techniques are also described. To motivate the use of more complex or tailored representations, each of these techniques is accompanied by experimental results. Chapter seven details skin and underlying tissue studies conducted on test subjects *in vivo*. Lastly, research conclusions are detailed.

1.1 Tissue Structure

Biological tissues have several different possible structures that can greatly influence their properties. Biological tissues in humans are difficult to model because of their amorphous bulk nature and their underlying anisotropic structure. There are several different classifications for tissue including connective, epithelial, muscle, and nervous. In addition to these classes, there are also adipose (fat) tissues, vascularization passages, glands, and bone which all can contribute to the mechanical properties of biological tissue.

1.1.1 Cellular Level

In some areas of the body, including on the skin and in the digestive tract, the tissue is composed of thin layers of different types of tissues with different functions. Figure 1.1 shows images of the layers in skin and the layers in the large intestines.

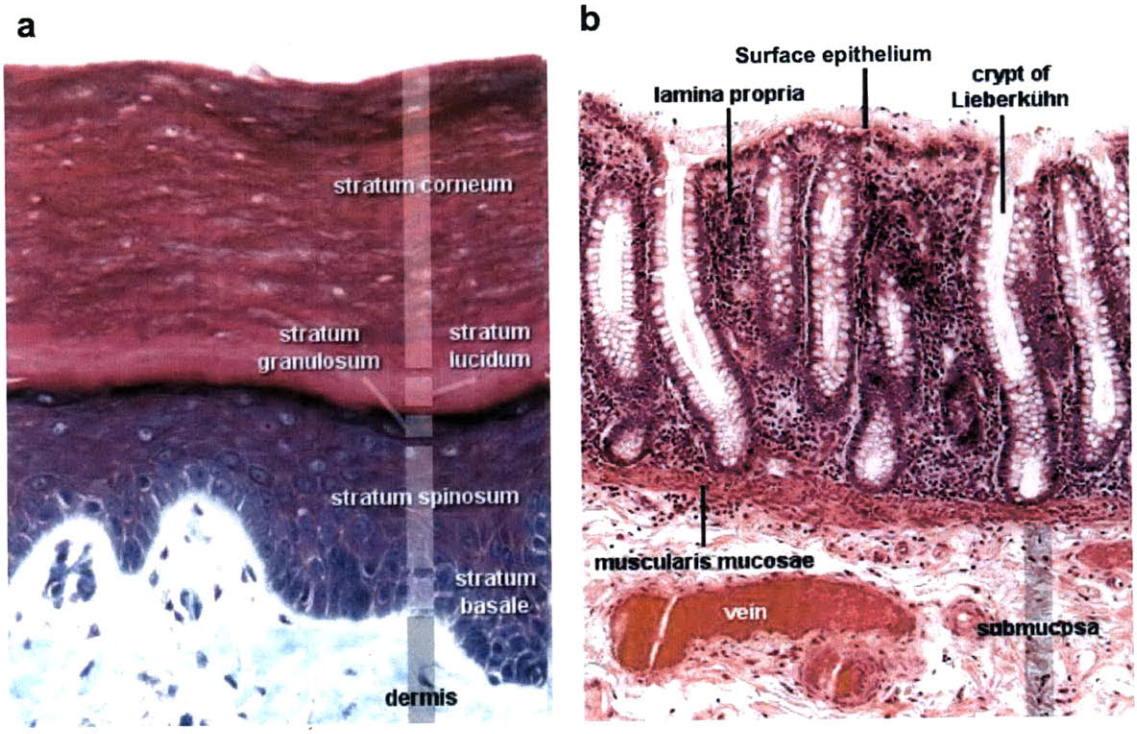


Figure 1.1: (a) Tissue layers of the skin and (b) tissue layers of the colon (compiled from [16, 95]).

In the skin, each of these layers has a different function. The epidermal layers in skin are responsible for the barrier function which keeps harmful bacteria out of the body and maintains hydration in the underlying layers [84]. There are several layers within the epidermis. The top layer is generally known as the stratum corneum followed by the stratum lucidum, stratum granulosum, stratum spinosum and the stratum basale. The bottom layer is the stratum basale which is made up of a single layer of epithelial stem cells that are shaped like columns. As these cells divide, they move upwards and differentiate, flattening out and eventually flaking off. Within this layer, there are several types of specialized cells which produce pigment (melanocytes) and defend against infection (Langerhans' cells). The layers of the epidermis can vary

from 50 μm to about 2 mm [28].

Below the epidermis lies the dermis which includes different types of connective tissue including collagen, elastic fibers, and reticular fibers. There are several specialized cells in this layer including hair follicles, sebaceous glands, eccrine gland, blood vessels and nerve cells. The dermis can be divided into the papillary and reticular components. In the papillary dermis, the collagen fibers are arranged randomly. In the reticular dermis, the collagen fibers are thicker and are arranged in bundles oriented in specific directions along the body. The orientation of collagen bundles are what creates Langer's lines in the skin. These features are important for surgical purposes and important for determining the direction of anisotropy of skin [28]. Below is the subcutaneous layer which includes additional connective tissue, fat, blood vessels and nerves.

Several layers are also present in the colon and these layers are ordered in a slightly different manner. The outermost layer, the mucosa, has several features like the surface epithelium, goblet cells, and several other types of cells which help absorb nutrients [12]. The surface is fairly smooth with glands which extend from the surface through the thickness of the mucosa as shown in Figure 1.1b. This is followed by submucosa (which is composed of connective tissue), muscularis and serosa [32, 95]. Other types of tissues may have fewer distinctive layers. For organs like the liver, the reticular fibers create a supporting, crosslinked mesh which helps hold the other cells together [8, 83].

1.1.2 Connective Tissue and Pathology

Collagen plays a large role in the measured mechanical properties of tissue. Therefore, connective tissue diseases can be well characterized by mechanical testing. Type I collagen comprises approximately 80 % of the dry weight of skin while type II collagen comprises approximately 15 %. The different types of collagen and their functions are:

- Type I collagen - responsible for tendons and is present in scar tissue.

- Type II collagen - a basis for cartilage.
- Type III collagen - common in reticular fibers.
- Type IV collagen - occurs in the basement membrane of skin.
- Type V collagen - surrounds cells.
- Type VI collagen - involved in matrix assembly.
- Type VII collagen - holds fibrils together.

Collagen fibers are formed from three polypeptides which are twisted around each other in a left-handed triple-helix. The constituents of the polypeptide chains can come from several different genes. In type II collagen, the three molecules are encoded by the same gene while in type I collagen, two of the strands come from one gene and the third strand comes from another [28].

Elastin is assembled as a net of material in the skin and it plays a large rule in maintaining the resilience of skin. When elastin begins to break down due to aging or genetic disease, the skin begins to wrinkle. The basement membrane of the skin makes up a thin sheet in the skin between the dermis and epidermis. This layer is responsible for cell migration, cell metabolism and cell differentiation. Diseases in this layer often manifest as blisters.

The diseases most commonly characterized by mechanical testing of the skin are shown in Table 1.1. Scleroderma is one of the most common connective tissue diseases which can be identified by mechanical testing devices [23]. Ehlers-Danlos syndrome is a genetic and progressive disease that can affect the skin, joints, and organ walls [44]. It has been successfully characterized by several types of mechanical testing devices. Additional studies that can quantify the effects of these diseases and others can help improve diagnosis and treatment.

In addition to connective tissue diseases, correlations of skin mechanical properties to age are also common [6, 15, 28, 29, 86]. As the skin ages, wrinkling will occur due to the reduction of fibers in the dermis (decreased amounts of type IV and VII collagen and an increase in type III), photo-damage and medication. Hydration effects can also be studied in the skin using mechanical methods or electrical impedance methods [86].

Table 1.1: Diseases of the skin related to collagen, elastin, and other skin components summarized from [23, 28, 44]. These diseases often cause the mechanical properties of skin to change.

Disease	**Skin Layer**	**Effect**	**Notes**
Scleroderma	Collagen	Thickening and stiffening of skin	Autoimmune disease where the collagen accumulates
Scleredema	Collagen	Thickening and stiffening of skin	Uncommon acquired disease which may be accompanied with redness
Ehlers-Danlos Syndrome	Collagen	Hyperextensibility, thinning, and fragility of skin	There are 9 subtypes of this genetic disease with several possible mutations in the genes that produce collagen
Osteogenesis imperfecta	Collagen	Thinning and fragility of skin	Caused by mutations of type I collagen
Diabetic thick skin	Collagen	Thickening and tautness of skin	Nonenzymatic glycosylation of collagen
Keloids	Collagen	Thickening of skin in pockets	Deposits of collagen
Focal dermal hypoplasia	Collagen	Thinning of dermis	Unknown cause
Steroid-induced atrophy	Collagen	Thinning of skin	Reduction in type I and II collagen
Cutis laxa	Elastin	Loose and sagging skin	Decrease in elastin
Anetoderma	Elastin	Lesions	Degregation of elastin in a localized area
Pseudoxanthoma Elasticum	Elastin	Lax and wrinkled skin	Accumulation of abnormal elastin
Actinic Elastosis	Elastin	Thickening and wrinkling of skin	Accumulation of elastin in the dermis, decrease in collagen and increase of elastin
Marfan Syndrome	Elastin	Hyperextensibility	Mutation in fibrillin gene
Epidermolysis Bullosa	Basement Membrane	Blister formation	Mutations in keratins, mutations in collagen
Erysipelas	Dermis and Subcutaneous Cells	Thickening	Streptococcal infection which causes an inflammatory dermal edema
Psoriasis Vulgaris	Epidermal Cells	Erythematous scaly plaques	Common and genetically inherited, build up of dead cells on the surface

1.1.3 Mechanical Properties

In studying biological tissues, the solution is often for researchers to study a specific layer, such as the epithelium. In order to do this without getting interactions from other layers, researchers often conduct tests *in vitro*. This, however, does not take into account the important interactions between living tissue layers. *In vivo* tests are more difficult to conduct because of these same interactions. Studies conducted on small displacements often state that they are focusing on the properties of a particular layer [28] but in fact, other layers provide a ground plane which can greatly influence results. The mechanical properties of these layers contribute differently to the perceived bulk properties and the ability to look at those nonlinear properties can help us discover information about their condition.

Studying the mechanical properties of the skin requires an understanding of how different layers contribute to the bulk. Connective tissues help generate the heterogeneous, anisotropic material properties. During small displacements, the elastin and reticular fiber matrix provides low resistance to movement thereby generating low stiffness. In addition, creases in the skin help absorb linear forces for small displacements [6, 84]. For larger displacements, the coiled collagen fibers begin to align and contribute more heavily to the overall stiffness. This effect creates strain hardening which can protect the tissue from damage [4]. The orientation of the Langer's lines will determine how displacement in different directions will contribute to the strain hardening effects.

In addition to anisotropic elasticity concerns, there is more information to be gained from dynamic testing. The skin has mass and damping (absorption of energy through losses) effects. There are also hysteretic components relating to fluid inflow and outflow and creep effects related to tissue damage. The range of forces and displacements along with the frequency of mechanical testing can greatly affect the types of information that can be collected. The construction of the testing device, dimensions of the probe and how the probe is attached are also important. Furthermore, living tissue will provide different values than excised tissue. Because tissues have

inherent nonlinear properties, as different regions of the nonlinearities are explored, different property values are recorded. All of these effects combined make it difficult to quote a value for something as simple as skin elasticity.

Part of the problem is that the dynamics of the testing device are often not characterized and are assumed to apply perfect forces to the tissue. The analogy from system dynamics is that actuators are assumed to have perfect output impedance such that the dynamics of the system being tested do not affect the dynamics of the actuator. In many mechanical systems, this is not true and making the assumption of perfect output impedance at the actuator can change the analysis and conclusions drawn from the results. In addition, once a different geometry or testing scheme is used, the measured results are not easily comparable. A single, well-characterized device that can conduct tests in multiple orientations can help alleviate this problem.

1.2 Tissue Characterization

Different types of tests and devices can be used to identify the anisotropic properties of skin. For *in vivo* testing, however, the contribution from directions outside the testing plane will always affect the results. A device that is capable of testing multiple directions at once [88], or is capable of testing in multiple directions separately, can be useful in determining these anisotropic material properties.

1.2.1 Devices

The measurements in this area of study have produced vastly different values for skin parameters which may differ by more than a factor of 3000 even when test parameters are normalized [20]. Recent studies have indicated values for the Young's modulus of the epidermis between 0.1 and 1.1 MPa. This broad range may be due to a simplification of the assumptions made in the arrangement of a specific instrument, nonlinear influences, inherent creep or relaxation of biological materials, and/or the arrangement of the Langer's lines.

A few commercial devices which attempt to make standardized measurements in-

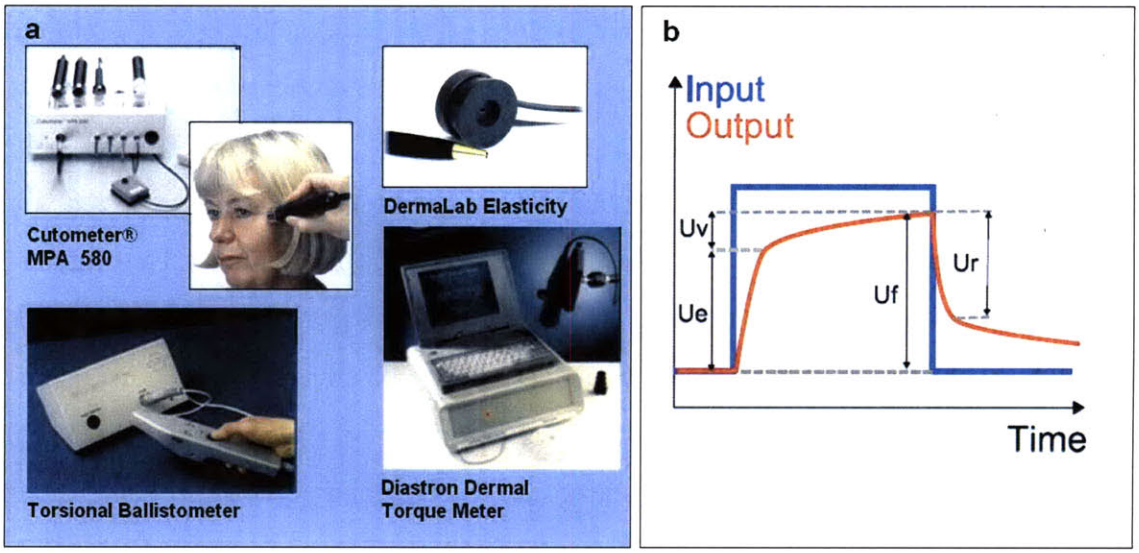


Figure 1.2: (a) Commercial devices and (b) techniques. This figure was compiled from [17, 18, 19, 51].

clude the Cutometer MKA 580, DermaFlex, and Diastron dermal torque meter (see Figure 1.2). Devices like the Cutometer and the Dermalab operate using suction where a pump applies a constant negative pressure at the probe head. The skin in contact with the probe will be pulled up into the probe. Sensors inside the suction probe will determine the maximum displacement of the skin. The focus of this device is to measure the elasticity of the epidermis since the turgor is a well-accepted indicator of dehydration and skin elasticity [37]. This process is inherently nonlinear in that a linear increase in pressure does not display a proportional increase in the displacement of skin or measurement in elasticity. Additionally, the complexities present in the geometry also contribute to the nonlinearities. In the suction systems, the input y-axis of Figure 1.2b represents the input pressure and the output y-axis represents the maximum vertical displacement of skin.

Qualitative tests conducted by health workers involve pinching the skin and then watching how quickly it returns to its original position. The dermal torque meter works similarly by applying a probe which spins and applies a torque to the skin. In torque meters, the input y-axis represents the input torque and the output y-axis represents the displaced angle in Figure 1.2b.

The terminology used by most clinicians and researchers allows them to look at different displacement parameters labeled U_e, U_v, U_r, and U_f which represent the elastic, viscoelastic, relaxation, and total displacement respectively. These values or ratios of these values are often quoted in the literature when studying different positions on the body. As the pressure or torque of the device increases, the skin will first displace elastically. There is almost no resistance in this region so it will appear as almost instantaneous in some studies. The skin will then creep in the viscoelastic region. Once the system becomes stationary, the device is usually timed to release the pressure or torque of the system. The system then relaxes and creeps. The application of the pressure or torque will normally cause the tissue to have some long-term deformation which means that the skin does not go back to its original position [28, 37].

The measured parameters are time-, probe size-, probe position-, preconditioning-, and pump pressure- (or torque application-) dependent making comparisons of devices, even those using the same actuation mechanism, difficult. Each measurement may take as long as 60 seconds to reach the stationary state for each applied pressure or torque which can make the tests conducted by these instruments extremely long [28]. The results have also been shown to vary as the number of cycles increases due to progressive creep [23]. These devices focus specifically on measuring skin elasticity and are not tailored for measuring dynamic or nonlinear behaviors. In addition, the mathematical system identification technique is not optimal. It relies on simple step responses which can theoretically contain a lot of important information which could be modeled using Burger, Maxwell or Kelvin-Voigt methods [28]. The simplification of these important parameters into simple displacement parameters loses sight of clinically important values and dynamic information in favor of expediency. Models, which often do not contain all the necessary dynamics, are then fitted to experimental curves. A more advanced technique which immediately casts the information into more relevant parameters like damping or energy storage/loss is needed.

The ballistometer class of instruments is slightly more advanced. Most ballistometers incorporate a hammer which free-falls (due to gravity) multiple times onto the

skin. The loss of energy (how quickly the ballistometer stops bouncing) is measured and used to determine the viscoelasticity of the skin surface and characterize multiple layers of tissue [28, 103]. Devices like this, however, are subject to patient movement and would require multiple tests to determine parameters at different impact forces. The results would be probe height dependent and one test could only provide bulk tissue (and not depth dependent, tissue layer dependent, or anisotropic) information.

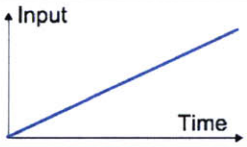
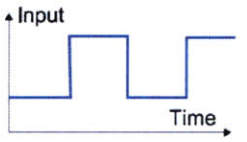
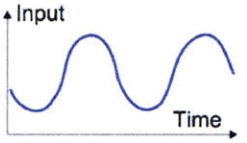
Research devices utilize slightly more advanced techniques for measuring skin parameters, but most of them are based on similar principles. This includes suction [20, 43], torsion [29], extension [15, 62, 76, 79, 97], ballistometry [103], wave propagation [86] and indentometry [6, 8, 83, 100]. There are a class of advanced suction-based devices with additional sensors to measure the changes in the thickness of different layers of the skin. The additional instrumentation includes optical coherence tomography [41, 42, 43] or ultrasound [7, 20, 22]. There are several devices which measure skin friction by sliding probes across the surface [24, 28]. The classic literature in the area uses different forms of extension measurements [11, 15, 62, 89] including measurements with multi-axial extension [88]. High frequency acoustic pulses below ultrasound frequencies can also be used to examine elastic properties [13].

Indentation has been used to identify soft tissues *in vivo*. One of the first instances of the use of indentometry to measure edemas was in 1912 by Schade [109]. There has been extensive work by several researchers to measure the elastic properties, the creep properties, the fat thickness, muscle properties, and even the response of pressure sores using this technique [28, 109]. Indentometer systems can be constructed with leadscrew systems, with voice coils, with pneumatic control, or with piezo actuation. There are also several devices which play on the theme of indentation by using non-contact perturbation (via pressurized air) and measurement (via infrared range finding) [54, 58, 90].

1.2.2 Characterization Techniques

Despite the dizzying array of methods used to measure skin mechanics, the characterization techniques have remained rather simple. Table 1.2 below shows the different types of measurement techniques that are commonly used in research.

Table 1.2: Characterization techniques used in the research literature.

Input Types and Analysis Methods		Most Common Perturbation Techniques	References
Static or Increasing Loads		•Extension •Suction •Indentation •Torsion	Kawahara 2006, Bishchoff 1999, Hendriks 2003 and 2006, Scalari 2000, Dunn 1985, Duchemin 2005, Daly 1979, Kaneko 2004.
Square Waves or Ramps		•Extension •Suction •Indentation •Torsion	Pederson 2006, Zheng 1996, Grove 2006, Reihsner 1998, Diridollou 1998 and 2000, Scilingo 2000, Khatyr 2004, Dobrev 1998, Henry 1996.
Frequency Sweep		•Extension •Indentation	Boyer 2009, Ottensmeyer 2001, Haruhito 2001, Osamu 2003, Potts 1984, Sadao 2001.
Stochastic Linear		•Indentation	Timanin 1999 and 2001, Oka 1997, Bjerring 1991, Kawazoe 1994, Osamu 1998, Takashi 2003.

The most common methods focus on static or increasing loads while slightly more advanced techniques focus on using square waves or ramps. In systems that measure the hardness or elasticity, a simple spring is placed inside the device and the force of the indenter can be applied by changing the position of the internal spring [38, 39, 50, 52]. These methods can be used by a large range of instruments and are easier to implement and analyze. These techniques can be easily used to look at low frequency nonlinearities. Slightly more advanced techniques that look at the frequency response of tissue use frequency sweeps. These methods are limited to methods that are more compatible with high bandwidth actuation. Nonlinear information can also be obtained from frequency sweeps [55]. In the patent literature, there are several

indentometer devices which use more advanced analytical techniques. Linear system identification has been used to identify biological materials using indentation [47, 56, 59, 82], suction [5] and other combined probes [60].

Layered finite element models [4, 31, 43] and anisotropic tensor matricies [62, 102] have been used to model skin tissue. Modeling based on first principles is one way to approach system identification but this must be done carefully. Taking a black (or gray) box model approach is another way to approach the problem. There has been little [102] work done that looks at both the dynamics of tissue and the nonlinear components. There has been no work that focuses on the nonparametric nonlinear system identification of biological materials.

Many of the techniques and instruments presented in the literature are ill-suited for the high bandwidths and the large displacements (20 mm or more) needed for full nonlinear system identification. Indentometry has inherent benefits over methods like suction or ballistometry for dynamic measurements of nonlinear systems. For larger forces, the device in indentometry mode is capable of measuring the mechanical properties of many layers from the epithelium to the underlying connective tissue. The response from different tissue layers [15, 81] creates interesting and clinically relevant nonlinear behavior. The work presented herein focuses on developing devices that are capable of indentometry, where a probe tip is pushed normal to the skin surface. Lessons learned from indentometry can then be applied to extension and surface mechanics testing using different probes.

1.3 Nonlinear System Identification Techniques

Several devices have been used to measure the linear dynamic behavior of skin and other organs [6, 83] but the measurement techniques tend to be slow because they perturb frequencies one at a time. *In vivo* tissue measurements should be conducted in an expedient and efficient manner and the analysis method should be relatively immune to patient motion. Linear stochastic system identification techniques have been developed specifically to satisfy these criteria. They also have many additional

desirable features such as the ability to measure the multiple frequency components at the same time and increase the instrument measurement sensitivity using statistical techniques. These techniques have been used to describe many biological systems successfully [33, 64, 78]. These methods, however, cannot accurately describe nonlinear behavior that is inherent in many biological tissue systems including skin.

Linear techniques have been well documented and have been used to inform some forms of nonlinear modeling. Nonlinear models can be categorized based on levels of prior knowledge (white, grey, or black box), on regressor structure (using the input, output, and/or error as regressors) [30, 74, 78, 94], or on model structure. Some of the different model structures are provided below.

- Prediction/error methods or instrumental variable methods such as NFIR and NARMAX [74].
- Subspace methods which utilize state space methodologies [57, 105, 108].
- Block or cascade structure techniques (*e.g.* Wiener cascades or Hammerstein cascades) [65].
- Global basis functions such as Volterra or Wiener kernels [77].
- Localized basis functions such as wavelets, sigmoid neural networks and fuzzy models [53, 94].

These models and techniques can be discrete or continuous [34, 104], parametric or nonparametric. Parametric models have certain advantages in that they can describe a system with fewer parameters and hence have more statistical power than nonparametric models (requires fewer data samples to obtain a model). Timanin and Eremin (1999) has proposed a few models for the nonlinear parameters of skin derived from first principles [102]. Parametric models, however, require more prior knowledge and can be more easily misused. They can make simplifying assumptions that ignore critical unmodeled data or can cause noise to be fit to parameters. In addition, most parametric models choose parameters that are not orthogonal. This means that if one value is changed, it affects the fit of other parameters.

Nonparametric techniques tend to make fewer assumptions about the form of the

system. They can be constructed to be orthogonal so that the fitting of one parameter does not depend on the fit of any other parameter and are more more immune to uncorrelated noise. In addition nonparametric techniques do not require prior information on the physics of the system. For example, a nonparametric technique that can be used to model the skin can also be moved directly to modeling the liver without having to rederive the fundamental physics. Nonparametric techniques tend to be more robust and can obviously be converted to parametric descriptions as needed.

Nonlinear system identification has evolved significantly since 1887 when Volterra first proposed the Volterra series. Since that time, Norbert Wiener proposed orthogonalization of the Volterra series to produce Wiener series and several efficient solution techniques for both Wiener kernels [66] and Volterra kernels [68] have been produced. These techniques have been used to identify several biological systems such as the stretch reflex dynamics of the human foot [61] or the dynamics of the pupillary system [98]. These kernel-based techniques are often computationally intensive, require specific input properties (such as Gaussian white or Brownian process inputs) and are restricted to time-invariant, finite memory systems. In addition, the results of higher order kernels become difficult to physically interpret.

In many biological systems, block structures such as linear-nonlinear-linear (LNL) cascades have provided simpler methods to modeling the nonlinearity. Black-box, nonlinear stochastic system identification techniques, such as Wiener and Hammerstein static nonlinearity techniques [49, 64, 67], help with this problem by providing broad frameworks of assumptions to model data and have been successfully used to describe biological systems [10, 61]. Since then, other nonlinear techniques such as wavelets, fuzzy-logic models, and neural networks have evolved [94].

1.4 Goals and Methodology

This work focuses on achieving the following goals:

- Create low cost device to identify dynamic compliance of tissue.
- Identify nonlinear dynamics of tissues using stochastic techniques.

- Optimize the identification with input generation techniques.
- Use an identification and computational technique that is fast.
- Use a technique that is good at accounting for variances in the data.
- Use a technique that is readily interpretable.
- The techniques must be capable of producing results that are repeatable and specific.
- The techniques should be able to distinguish the change in skin properties after dehydration or after application of commercial products.

The ability to assess dynamic data is essential to obtaining a more complete picture of tissue properties. In order to assess dynamic properties, a high bandwidth actuator system is necessary. The skin property identification geometry that is most conducive to high bandwidth actuation is indentation. However, a device that is capable of other identification geometries can be more versatile. The design of the device should focus on low cost elements in order to make the technology readily applicable to commercialization.

The key aspects explored in this work are based on optimization of nonlinear system identification techniques. This includes optimizing the input to the system, optimizing the test time, and choosing a technique that is both fast and accurate. In order to do this, simple static nonlinearities are first explored followed by more complicated dynamic kernel based nonlinearities. Partitioned systems can have an additional degree of freedom above static nonlinear systems which give them the ability to fit the form of the dynamic nonlinearity in tissue. By exploring the pros and cons of each method, different methods can then be chosen for different situations, whether it be for commercial or research purposes.

Chapter 2

Device Design

The mechanical, electrical, and software designs of custom devices for system identification are discussed. The device is developed as a platform technology for easy incorporation of multiple application heads (indentation, extension, surface mechanics) in order to measure multiple tissue parameters in several directions. Based on this research, it was determined that the device should have a stroke of 32 mm and the capability of driving at least 15 N of force at a high bandwidth in order to measure the nonlinear properties of skin in different configurations. To achieve these metrics, custom linear Lorentz force actuators are utilized. For clinical applications, the device has a hand-held form factor that is under 30 mm in diameter and 100 mm in length for the applicator body and includes integrated electronics. The focus of the design is on the use of low cost materials and scalable designs that can easily lead to mass production.

The device includes integrated power and sensor electronics with custom software which can be used to calibrate the system and assess the biological properties of skin and other biological tissues. The inclusion of additional sensors capable of measuring non-mechanical properties of skin such as blood reflow (using a light source and sensor), water content (using electrical contacts) and tissue layer thickness (using ultrasound techniques) would serve to increase the capability of the platform.

2.1 System Design

In most models found in the literature, skin and biological tissues in extension or compression are modeled as Maxwell or Kelvin-Voigt systems which include only springs and dampers [28]. Figure 2.1 shows these models.

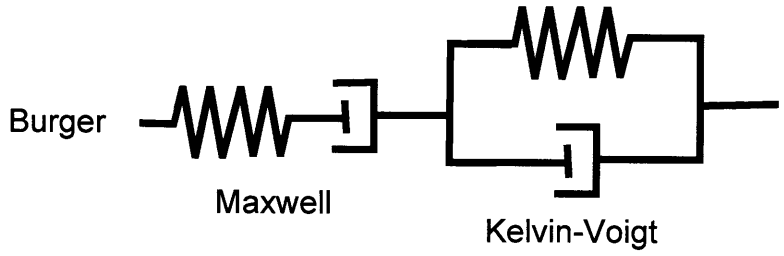


Figure 2.1: Linear models of tissue dynamics include the Burger, Maxwell, and Kelvin-Voigt subcomponents in different configurations.

Tissues have their own inherent mass but this is a second order effect when compared to the spring constant and damping from material properties. With a Kelvin-Voigt first order system in mind, the addition of an actuator with mass will change the system to second order. For systems in friction or surface mechanics mode, there is no spring and nonlinear damping related to friction. A system with this configuration would be second order with a single pole at the origin. These systems would tend to "walk" rather than stay at a constant mean position. Therefore, an additional spring should be added to this type of system to maintain a constant mean position for an input with a non-drifting mean force.

Using these linear components, a more nonlinear model can be conceptualized. Figure 2.2 shows the different layers of the skin and how these layers can be visualized as a system of springs and dampers during indentation. As the skin is compressed in this model, the softest layer, and not necessarily the topmost layer, is compressed first. As the maximum compression depth is reached for this layer, it will become stiffer or stop acting like a linear spring. The mass of this layer can then be added to the total mass of the system. The next-most compliant layer will then contribute more to the overall compliance of the skin and so on as the system recruits more springs, masses, and dampers. From this argument, it is expected that the stiffness of the

system during indentation should increase monotonically with depth into the skin and should be relatively continuous. *In vivo*, however, there are additional contributions from the interconnection of the skin being compressed along with the skin being extended on the sides. This adds additional compliance terms to the model. Each layer of the tissue also has viscoelastic strain hardening contributions from collagen alignment. Several researchers have attempted to model these effects using pressure concentration or geometric arguments [102]. Several authors have also attempted to model the change in impedance of layered tissues [81].

When the tissue is being pulled (when the probe is attached to the tissue with liquid bandage for example), the picture changes. The slow, gradual increase of spring stiffness would be replaced by a much faster increase in stiffness. This is because the tissue in compression can exhibit fluid outflow from the region under compression into other regions. When the tissue is being pulled, fluid must flow into the area underneath the probe from other locations. In addition, the deformation must be matched by the stiffest layer, which is the epithelial layer. Therefore, the stiffness of probe pulling on the skill will be higher for the same displacement.

In extension, different effects occur. The underlying layers of the skin are in shear while the topmost layer of the skin is in extension. With surface mechanics testing, the picture becomes simpler since the probes are not permanently attached to the skin and contributions to stiffness and damping come from interactions with only the bulk tissue as a function of the probe normal force. The differences between these configurations are discussed in Section 2.2.3.

Modeling the skin in this manner leads to the prediction that skin is linear for small displacements and exhibits nonlinear behavior for larger displacements. This nonlinearity, to first order, represents a static nonlinearity that should scale with the depth (position) of the tissue surface with respect to a reference surface. However, the evolution of stiffness, damping, and mass do not have to follow the same nonlinear profile. These will be discussed in subsequent chapters.

With the above predictions, a system can be built to apply the appropriate inputs and acquire the appropriate outputs. Figure 2.3 shows the schematic diagram of

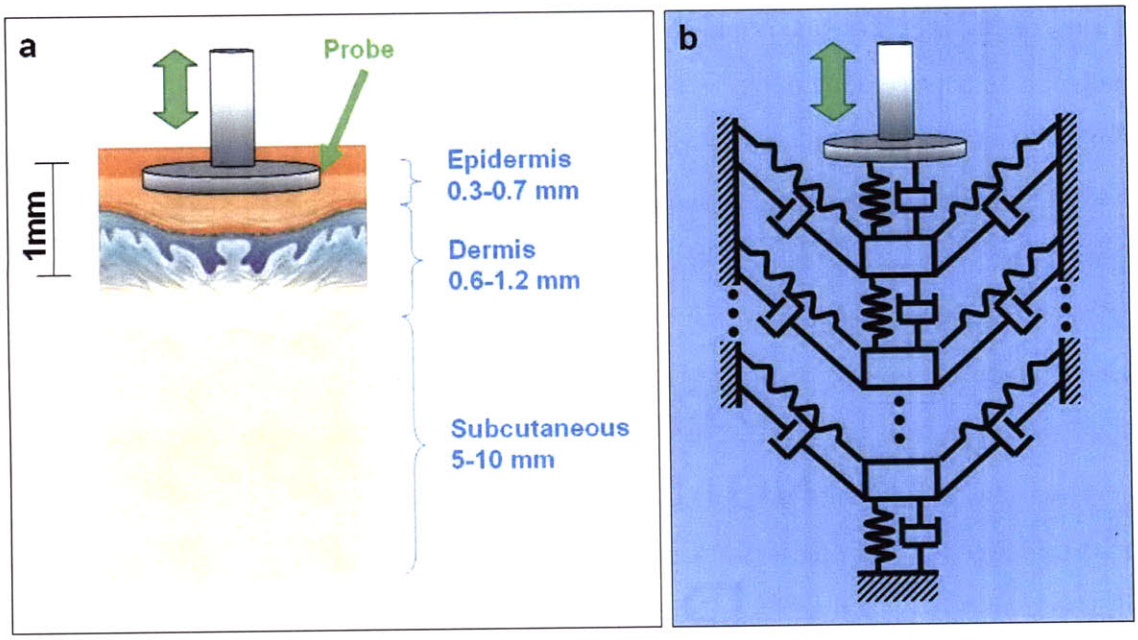


Figure 2.2: Layers of the skin under (a) indentation and (b) dynamic model representing the different layers of the skin as a system of springs and dampers.

the system along with inputs and outputs. It is important to note that the system includes the actuator and tissue dynamics. The actuator has an inherent mass and the bearings and air resistance have inherent damping. There are two possible methods for treating the nonparametric or parametric information derived from the identification of this system. Either the system must be treated as a whole (*e.g.* the derived mass is a system mass) or the system can be treated as a linear addition of actuator parameters and tissue parameters (*e.g.* the derived mass is composed of the actuator mass acquired from calibration plus the tissue mass). The results in this paper are given as values of the system which includes both the actuator and tissue because the actuator cannot be considered to have an isolated input impedance.

In the schematic, the input is a voltage $V^*(t)$ sent through a linear amplifier into the force actuator that perturbs the skin. The applied force is $F^*(t)$ and the position is $P^*(t)$. Because of different sources of sensor error $e_1(t)$ and $e_2(t)$ (see noise floor on Figure 3.2), the measured force is $F(t)$ based on the current $I^*(t)$ and the measured position is $P(t)$. Note that the input generation component is separate from the data acquisition software. Therefore, this system operates open loop. Closed loop input

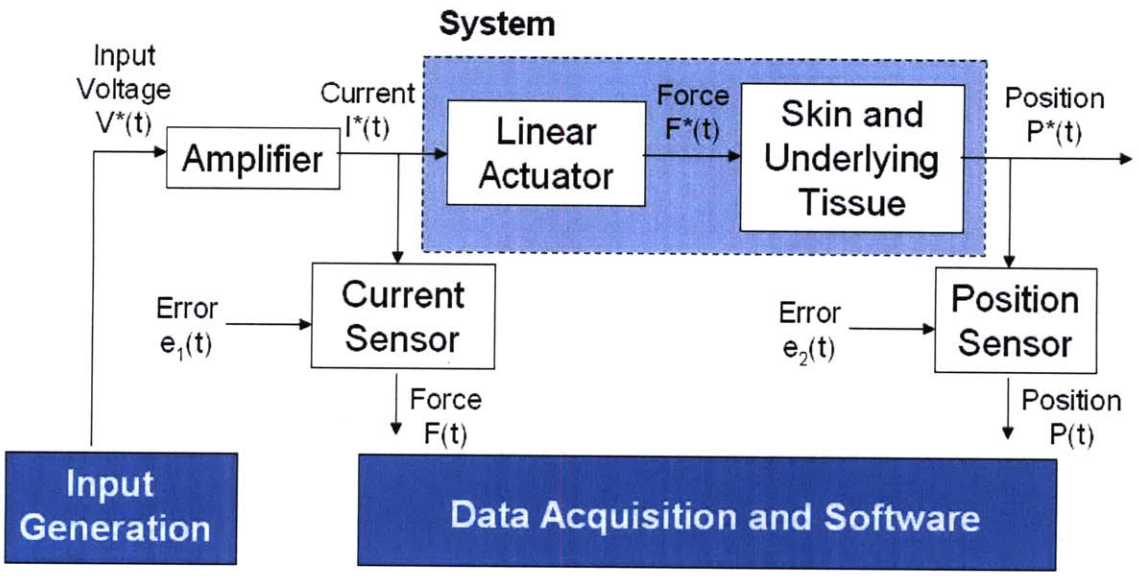


Figure 2.3: Schematic diagram of system signals based on the use of a Lorentz force linear voice coil actuator. The input is a voltage $V^*(t)$, the applied force is $F^*(t)$ and the position is $P^*(t)$. Due to sources of sensor error $e_1(t)$ and $e_2(t)$, the measured force is $F(t)$ and the measured position is $P(t)$.

generation can be implemented when the loop is closed between data collection and input generation (see Section 6.3).

The force measurement is taken after the amplifier for many reasons. First, measuring the current after the amplifier skips the amplifier dynamics and any output timing lags of the software. No matter if the input is a voltage or current command, measuring the dynamics after the amplifier is desirable. Second, a force to displacement measurement would create a causal impulse response of the mechanical compliance, which can be analyzed with simpler system identification techniques. Lastly, because the input to the system is directly related to the force output, as opposed to a position-based actuator system, there is no internal feedback algorithm needed. This creates a mathematically simpler system identification situation with the capability to act at higher frequencies; a system with feedback is required to operate at a significantly lower frequency than its controller/observer poles and zeros. Because of the configuration of the system, real-time controller is not necessary for operation but can still be implemented for real-time input generation schemes.

2.2 Mechanical Design

The system dynamics were carefully considered when designing the mechanical system. An actuator which is designed to apply a force directly (rather than through force feedback) is desirable for high-bandwidth operation. In addition to speed, the system has to have a large stroke in order to test the depth dependent nonlinearities in skin. Several high bandwidth strategies are available, and an indentometer using a Lorentz force voice coil was chosen for this device. The Lorentz force is a force on a point charge caused by an electromagnetic field. The force on the particle is proportional to the field strength B_e and the current I^* that is perpendicular to the field multiplied by the number N_e of conductors in series each with length L_e. When a current is applied to the coil, the charges interact with the magnetic field from the permanent magnet and are accelerated with force,

$$F = I^* L_e N_e \times B_e. \tag{2.1}$$

There are two main types of voice coil configurations known as the overhung and the underhung configurations. In the overhung configuration, the coil windings are taller than the height of the magnetic field gap. This configuration has the advantage of higher sensitivity but suffers from higher coil mass. The underhung configuration is when the coil windings are shorter than the height of the magnetic field gap. This configuration has the advantage of lower mass but creates a bigger control problem if the coil leaves the gap. For the overhung configuration shown in Figure 2.4, the magnetic field only interacts with the current-carrying wires at a single region near the top plate where the field lines are perpendicular to the direction of the current.

If the current is coming out of the page at the top of the drawing and going into the page at the bottom of the drawing, the force generated would cause the coil to move to the left relative to the casing. If the current was reversed, then the direction of the force would also be reversed. For a fixed geometry, the force constant can be defined as $L_e N_e B_e$ for an infinitely long coil. This indicates how much force the coil can output for an increase in the current. Since real coils are of finite length, there

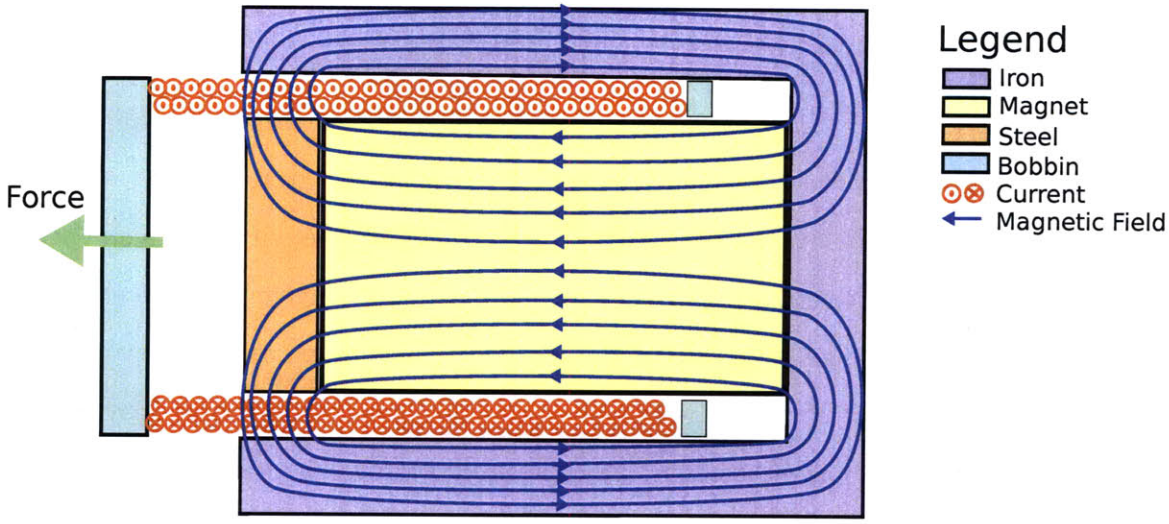


Figure 2.4: Lorentz force linear actuator operating principles. The blue indicates the direction of the magnetic field lines and the red indicates the direction of the current.

are small nonlinearities at either end of the stroke.

The power handling capabilities of a coil are limited by its heat generation (due to ohmic heating) and its heat dissipation capabilities. The housing for many coils provides heat sinking abilities preventing the coil from heating too quickly. In addition, a moving coil can provide convective cooling. The most commonly used forces and test lengths for this device would not require advanced heat handling measures [40] but a temperature sensor is added as a safety measure to monitor the temperature for high force or extended length tests.

In addition to the relatively simple operating principles, the Lorentz force linear actuators where chosen for the following reasons.

- Direct force control: The Lorentz force coil can be driven to produce a force as commanded since current is proportional to force and voltage is proportional to velocity. Forces under 15 N would require that the actuator used in the experiments be driven at voltages lower than 48 V. Directly controlling force open loop gives advantages for proving the identifiability of system parameters when compared to servo-controlled stages.
- Incorporated force sensing: The force can be measured by looking at the current flowing through the actuator. This would be the most low-cost method for

measuring force. For some separate force sensors, there are additional dynamics which are detrimental to the system identification process.

- High force limits: The coil can be driven to high forces which are limited by the amplifier and the heat transfer properties of the actuator.
- Long stroke: The coil can be designed with a long stroke with relatively large regions of linear operation. Other systems like those driven by piezo-electrics [6, 73] do not have as high a stroke and do not generally operate at low voltages.
- High bandwidth: The bandwidth of a Lorentz force coil is limited by input power, the mass of the system, and the stiffness of the tissue being tested. Other strategies including lead screw systems have relatively low bandwidth in comparison.
- Low cost: The actuator consists of a magnet, a steel cap, an iron core to guide the magnetic fields, and a copper coil. The simplicity of the design helps reduce cost.
- Few accessories necessary: In order to operate the coil, the only accessories outside the actuator are an amplifier and power system. Other strategies such as non-contact pressure systems require an additional high pressure pump and valves.

The sensors for this system can also be chosen based on the design simplicity criteria. Several different concepts exist in the literature for sensing position including non-contact LVDTs and laser systems. To minimize the space necessary for the sensor and the cost of the sensor, a linear potentiometer was used. Based on these design decisions, several implementations of the mechanical design were constructed including a prototype, a desktop research device, and a commercializable hand-held device. Figure 2.5 shows an image of each version.

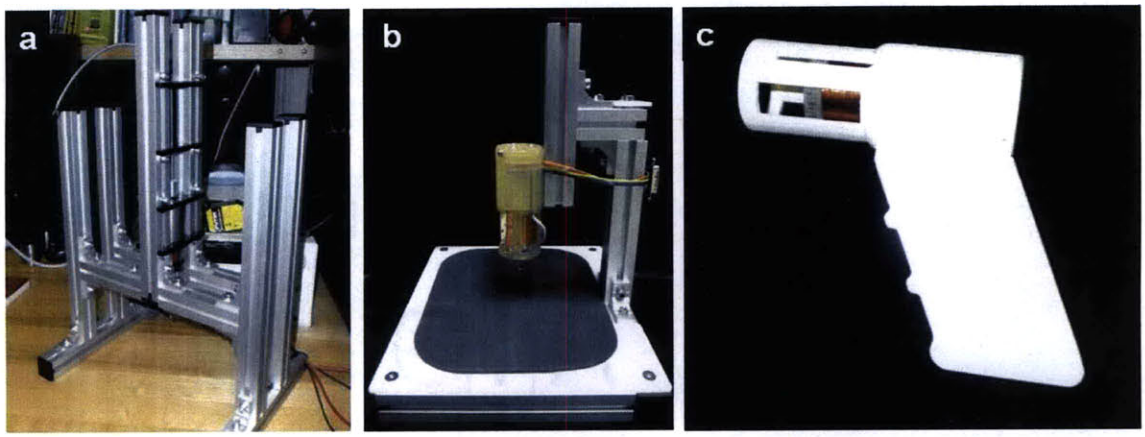


Figure 2.5: (a) The prototype, (b) desktop version, and (c) hand-held version of the device used for nonlinear system identification.

2.2.1 Research Device

The design of the mechanical device consists of an easily controllable actuator and force sensing system, a low-cost position sensor, a temperature sensor, an injection-moldable external bearing system and swappable device probes. The original prototype incorporated a linear position sensor on axis with the actuator with a force sensor in line which made it rather long. This design was used to guide the construction of smaller designs. Figure 2.6 shows an enlargement of the design of the desktop mechanism in Figure 2.5b.

The desktop research instrument is based on a Lorentz force linear actuator with a bobbin mass of 60 g, a total length of 32 mm, and an inner diameter of 25.2 mm. The design of the coil dynamics is important to the identification of the system. The mass of the coil was designed to be low enough to not significantly overshadow the inherent mass of the system to be identified. However, the mass of the coil must be high enough such that the nominal natural frequency of the system $\sqrt{K/M}$ is within a testable range for the actuator and signal processing system.

A BEI Kimco magnet structure was used with a neodymium magnet with a magnetic field strength of 0.53 T. A custom designed overhung bobbin was 3D printed with multiple attachments for a temperature sensor, easily insertable electrical connections, through holes to allow air flow, threaded holes for attachment of custom

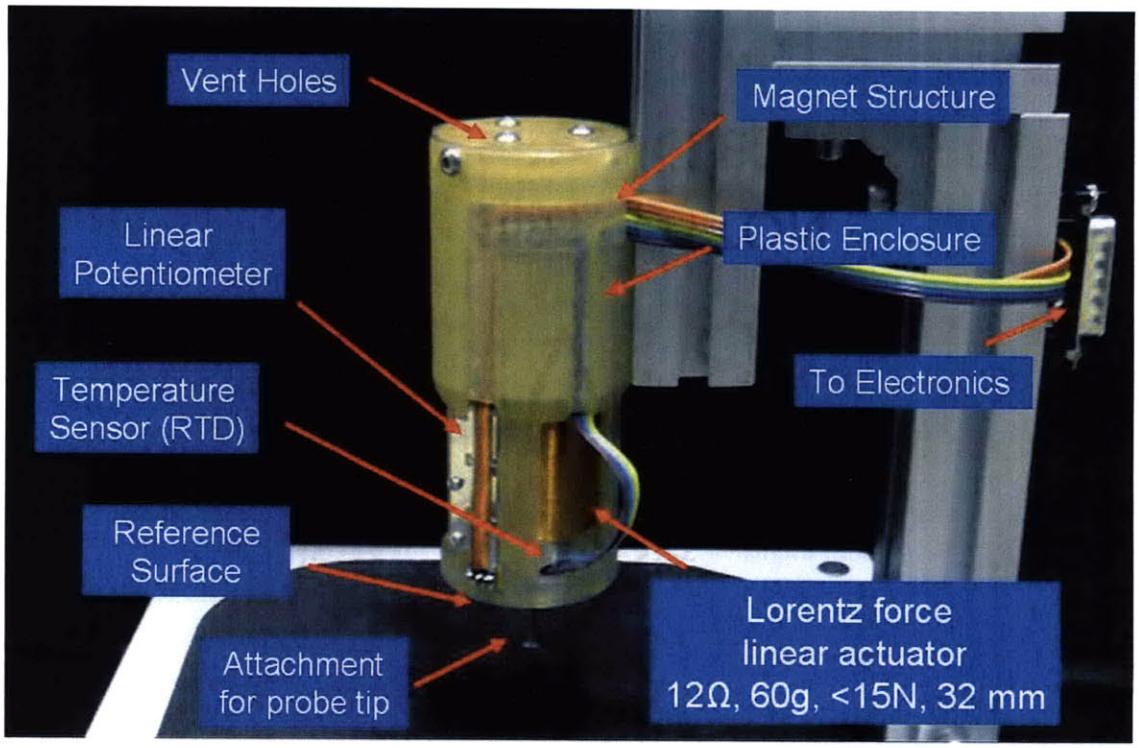


Figure 2.6: Detailed desktop version of the device including a voice coil actuator, bobbin, linear potentiometer, and bearing structure.

probes, and a wire insertion slot. The custom wound coil has a resistance of 12 Ω, inductance of 1.00 mH, a force constant of 3 N/A, and 6 layers of windings using 28 gage wire. This winding structure is chosen to balance the load on the amplifier. A smaller gage wire would require higher input current. A custom circuit for high current is generally more expensive to build than a circuit for high voltage operation. There are significant advantages to working with systems that operate under 48 V in terms of safety regulations ,which guided the design of this configuration. The device includes a force sensing system via a current sense resistor. The coil design also includes the integration of a small OMEGA F2020-100-B Flat Profile Thin Film Platinum RTD into the side of the coil. This RTD monitors the temperature of the coil to prevent actuator burn-out.

A low-cost linear potentiometer ALPS RDC10320RB was used to measure position. When implemented with an amplifier and 16 bit DAQ the position resolution is as low as 0.5 μm. An enclosure which fixtures the actuator, the position sensor, a

position reference surface, internal wiring guides and a handle was also constructed. The body of the device doubles as an encasement for the magnet structure and as a bearing surface for the bobbin. Teflon bearing spray is used to reduce static friction and help create constant dynamic damping. Lastly, the lower end of the body includes an attachment feature which allows it to be slid into MK automation aluminum framing, as in the desktop version of the device, or into a custom handle, as in the handheld version of the device. For indentation, the desktop version of the device can utilize gravity to provide an extra constant preload on the surface.

The MK framing with base allows the system to have a small and stiff structural loop that enables the device to be more precise and helps eliminate system noise. The more important structural loop, however, is the one between the rim of the actuator, known as the reference surface, and the system being tested. This is because forces and positions are being measured with respect to the reference surface thereby allowing the instrument to characterize tissue compliance.

2.2.2 Handheld Version

The handheld version was designed with ease of use in mind. In order for a clinician to test different locations on the body, the testing mechanism must be easily moved and positioned. This, however, creates a few problems with the structural loop of the system. Additional movement of the patient and clinician as the actuator vibrates may cause noise in the readings. Figure 2.7 shows the system dynamics of the handheld system.

As long as the reference surface is placed in firm contact with the surface of the patient's skin, the system will successfully measure the compliance transfer function of the skin and not of other components. To account for force changes when going from measurements in the desktop system with respect to the handheld system, two additional components were added to the system. A bracing band is included to help maintain the lateral position of the actuator on skin. An accelerometer is also added to account for the orientation of the device (compensate for the directional loading differences from gravity). Figure 2.8 shows a solid model rendering and

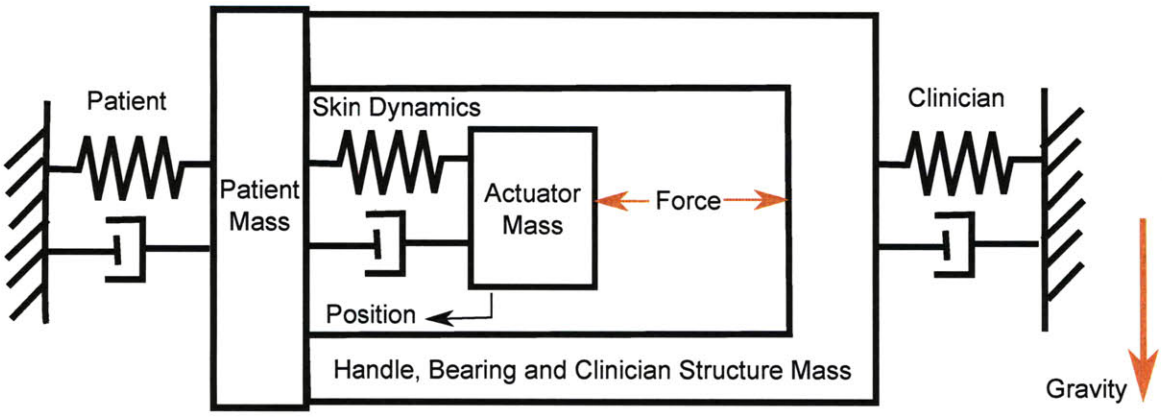


Figure 2.7: The dynamics of the handheld device show that as long as the reference surface is held firmly against the patient, the only additional contribution to the dynamics would be that of the varying direction of gravity. This can be accounted for by using an accelerometer. Note that the skin dynamics is modeled here as a Kelvin-Voigt system but any other skin dynamics can be included here.

implementation of the hand-held version of the device designed for commercialization. The handheld system is smaller than the desktop system and therefore has a lower force output. For smaller designs, more advanced Lorentz force coil designs may be necessary to maintain force outputs. The stroke of the system is nominally 32 mm, the bobbin resistance is 9.5 Ω and the magnetic field strength is 0.35 T. The mass of the actuator is 39.5 g and the total mass of the handheld device is 256 g.

A commercializable device must also be low cost. With some cost optimization in the design of the electronics, the total cost per device (not including labor, tooling, and assembly), would be $239.63 for the mechanical device and the electronics. The outline of prices is listed in Appendix A.1. The cost of the mechanical components is about $45.04 which includes the voice coil, the bearing structure, sensors and probes. The power supply alone costs $84.01. The next most expensive electronics include the linear operational amplifiers and the instrumentation amplifiers. With labor, tooling, and assembly, this price would still be less than the list price of the most popular devices on the market for large quantities. The quoted price of the DermaLab device is $8,100.00 without accessories. The quote for the device is also available in Appendix A.2. Based on the evaluation of price, the device described in this work would be competitive with other devices on the market.

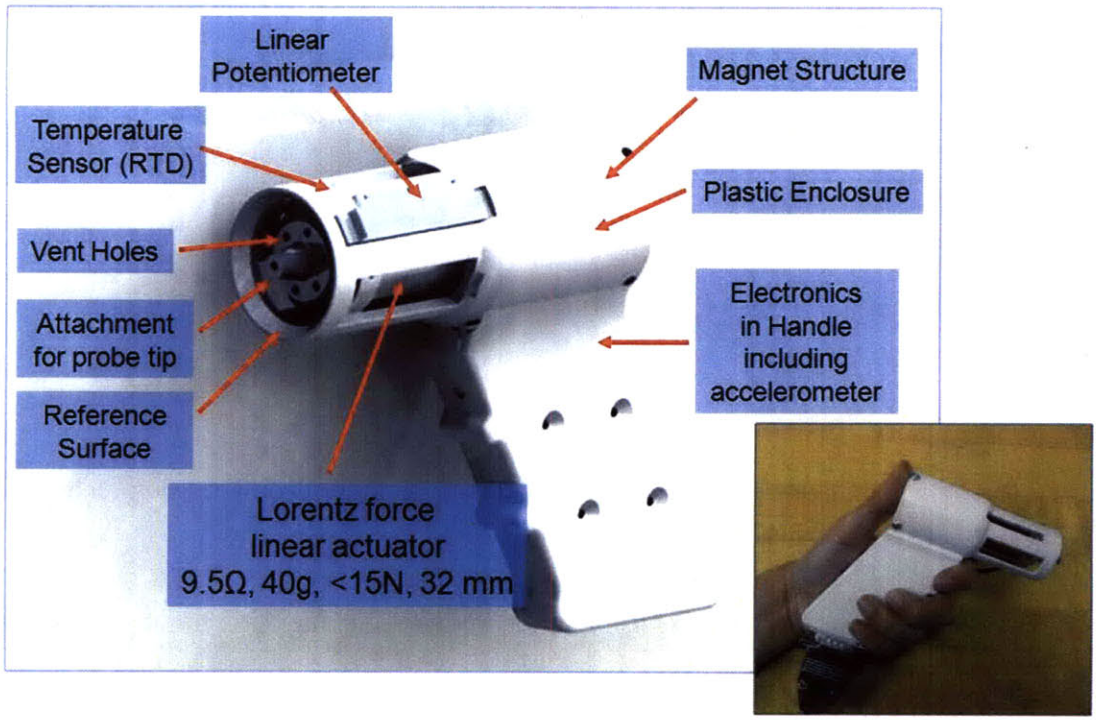


Figure 2.8: Detailed handheld commercializable version of the device with solid model and implementation.

2.2.3 Device Configurations

The design of the instrument allows for several different *in vivo* system identification modes including probe indentation, extension, and surface mechanics testing depending on the custom probe type used. Figure 2.9 shows the different configurations for the device. For indentation, the typical tip used for contacting the skin is a 4.4 mm steel disk with a normal thickness of 1 mm. Tips with different thickness, diameters, and corner radii are also used. The attachment to the actuator is recessed to allow the skin to freely conform without contacting other parts of the probe. All indentation measurements are made with respect to the position reference surface that is significantly larger than the circle of influence from the probe. For indentation into the skin, which is the main focus of this work, no taping or gluing is necessary. However, to do experiments that require lifting the skin, using suction, liquid bandage or some other type of mild glue is required.

For extension experiments, a separate probe and reference surface are used. The

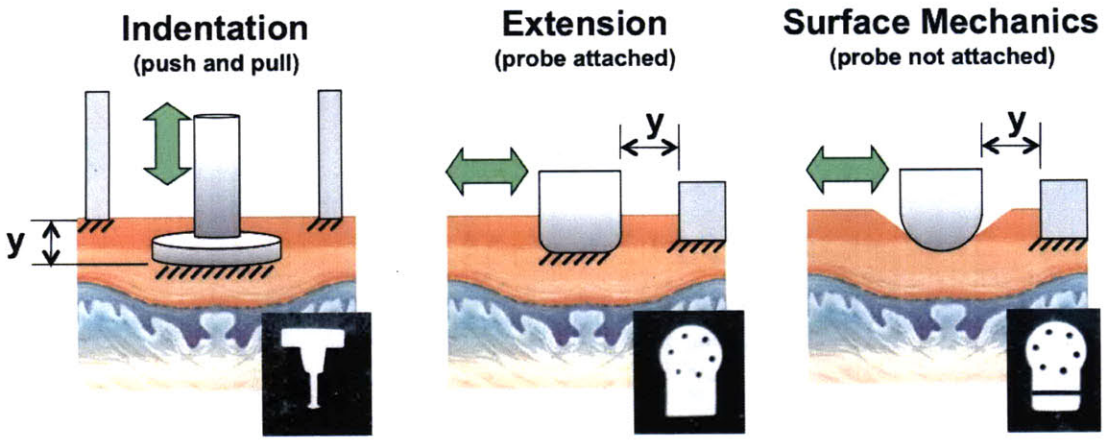


Figure 2.9: Configurations of the device including an indentation configuration, extension configuration, and surface mechanics configuration.

extension probe is 5 mm by 16 mm with rounded edges that have a 2 mm radius. The rounded edges are important for reducing stress concentrations [4]. The reference surface is a flat face on the actuator side. It serves as the second probe surface. Coupling to the skin can be provided by a normal preload and a mild layer of liquid bandage or double-sided tape. The extension system can be oriented along the Langer's lines or in other orientations to characterize anisotropic tissue properties.

Surface mechanics testing, or friction assessment, is configured in a manner similar to extension experiments except that the probe is more rounded and allowed to slide along the surface of the skin [28]. Since the surface mechanics system is second order with a pole at the origin, an external spring is needed in the system to complete linear stochastic system identification. Note that the measured compliance will be a function of the depth of the compression of the probe into the skin depending on the vertical preload.

2.3 Electrical Design

The electrical design consists of input drive circuits and data acquisition circuits. Several designs for the electrical system were also implemented in an effort to reduce the footprint of the device. The first electronics system was driven by a Kepco BOP

50-8D amplifier with a total output limit of 50 V at 8 A. The force, position, and temperature sensors were powered with an Agilent E3631A power supply, amplified with AMP 02 instrumentation amplifiers, and filtered by an anti-aliasing filter with a cutoff frequency of 2 kHz. Subsequent designs used smaller components. The electronic schematics and implementation of electronics are shown in Figure 2.10. The inputs to the electronics are in green, the outputs are shown in blue while sensor and actuator components are in red.

In the schematic, the inputs to the system (outputs of the data acquisition) are in green, the outputs of the system (inputs to the data acquisition) are in blue, and the system actuators and sensors are in red. In order to assure good signal quality and to reduce power supply noise from high current draw from the actuator, the signal acquisition circuits and the coil amplifier are placed on a separate power supplies. There are three separate power supplies in this design. One of the power supplies is at 48 V and is powered by a Phihong P835A1 AC power adapter. It is capable of 2.5 A continuous operation and instantaneous current can be larger with the addition of a 15 mF capacitor. This power supply is significantly smaller than the Kepco BOP 50-8D. Even smaller power supplies can be used if smaller forces are desired. The signal conditioning amplifiers are driven at ± 9 V and the sensors are driven at 5 V.

Linear amplifiers are chosen to drive the coil as opposed to a class-D style binary input because a linear amplifier system gives more flexibility in designing the input values and input distribution. In the design shown in the figure, it is possible to drive the coil forward and backward at up to 48 V. Each of the Burr-Brown OPA 549 linear amplifiers is capable of operating up to 60 V and 10 A. The package has a fast slew rate of 9 $V/\mu s$. Each amplifier circuit can source or sink current. In order to avoid possible deadband near the lower power supply limit, the inputs Vin+ and Vin- are offset.

The Lorentz force coil can be modeled with a resistor, inductor and additional skin mechanics. The voltage across the coil can be monitored using an AMP 02 instrumentation amplifier system measuring across a voltage divider. The instrumentation amplifier buffers the signal and an additional first order low-pass filter reduces

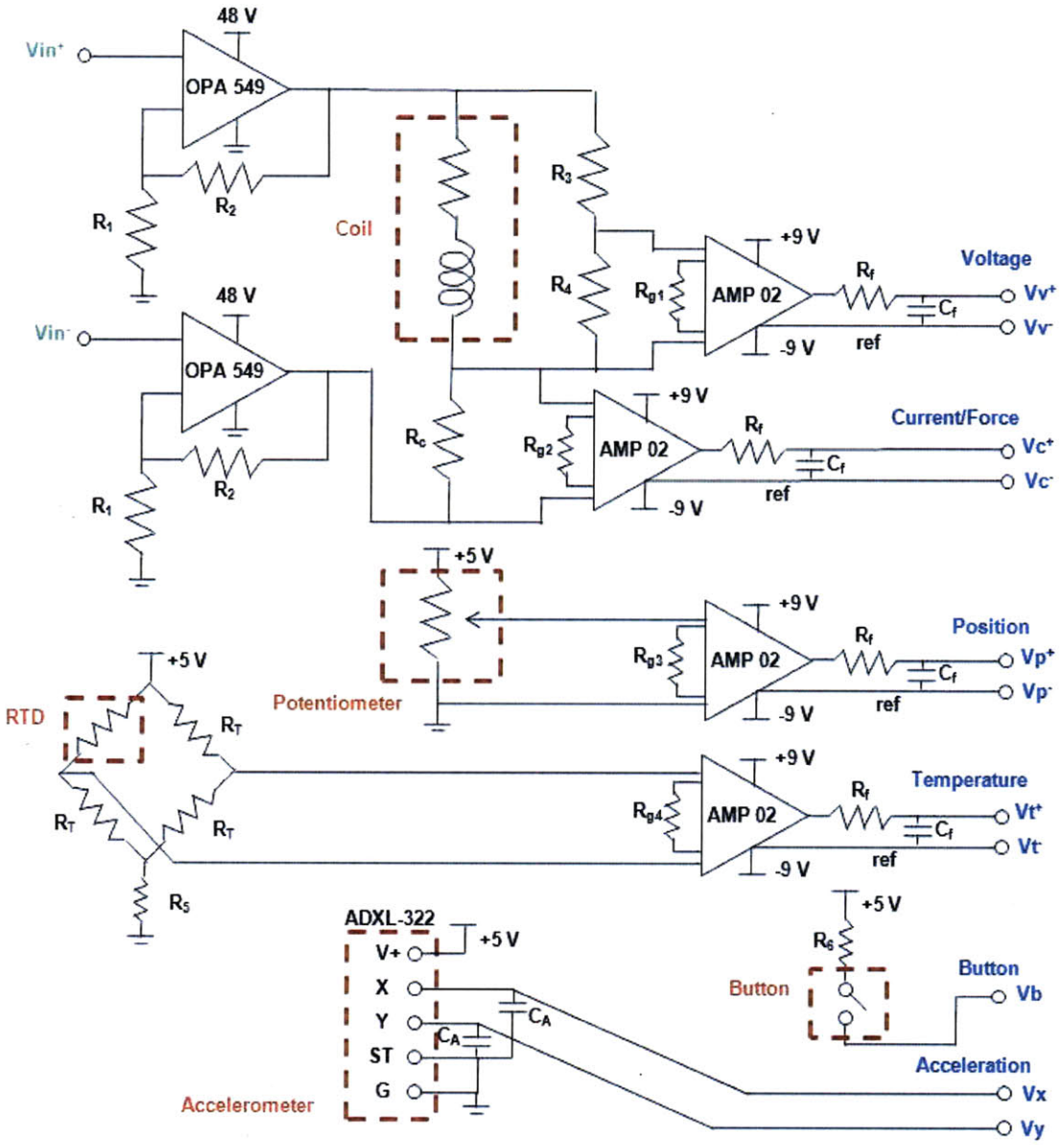


Figure 2.10: Electronics design and schematic for the sensing and power systems. Inputs are in green, outputs are in blue, and sensor and actuator components are in red.

any high frequency noise. The output is differential which takes advantage of the resolution of the ADC.

The current across the coil is measured using a current sense resistor and amplified using another circuit. The potentiometer and the RTD are also buffered using AMP 02 circuits. The RTD offset is reduced with a Wheatstone bridge circuit and the output signal is also amplified. A button on the system is measured with an internal pull

down resistor. The accelerometer, which is used in the handheld system, is an Analog Devices ADXL-322 with two axes.

The data acquisition is completed with a National Instruments USB 6215 with a 16-bit ADC implemented on all input channels and a 16-bit DAC implemented for the output. The system input and output information are stored in a firmware buffer to eliminate communications lag time with the computer. The voltage is measured rather than assumed as the same as the input across V_{in+} and V_{in-} because this skips over the dynamics of the amplifiers. In addition, this bypasses the lag that occurs between the output command and output realization times.

2.4 Software Design

User-friendly software, which interfaces with a conventional laptop computer, was developed to display test procedures, test results and other clinically relevant information. The system identification software is implemented in MATLAB and interfaces with the LabVIEW 8.5 for data acquisition and display. The software has the capability of completing an auto-calibration, completing force constant compensation, driving an input, taking measurements, implementing the system identification, and displaying the information to the user. For the purposes of display, a simpler version of the system identification that runs at a faster speed can be used to initially assess the validity of the data. More complex system identification schemes are later implemented in MATLAB.

2.4.1 Calibration

There are two pieces of calibration software that help determine the performance of the system. The user interfaces for the calibration systems are shown in Figure 2.11 and additional details for the LabVIEW program are shown in Appendix B. The first piece of software is the static calibration system. The static input voltage can be compared to the static force via an initial calibration. Then additional current to force calibrations can be used to calibrate for variations in the power supply for

the sensor system. In order to do this, voltage is first applied to the coil at different constant values and the current is measured. This gives a linear calibration curve. The accelerometer and temperature sensor are calibrated similarly. In the last step, the calibration constants are saved to an external file and read by the system identification software.

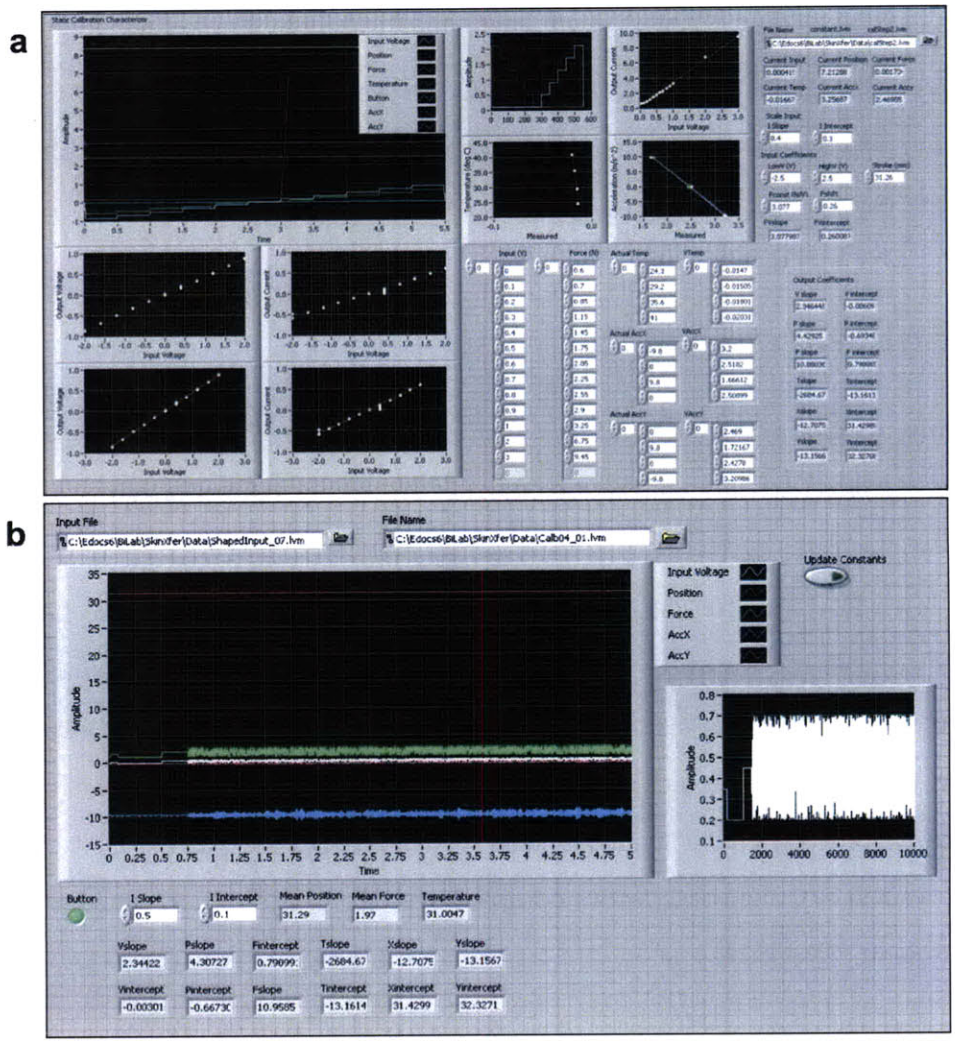


Figure 2.11: Software used to auto-calibrate the system at startup. (a) The static calibration system configures the force, position, and voltage readings while (b) the dynamic calibration system configures the full second order linear dynamics of the coil at different positions.

The dynamic calibration system is used to determine the performance of the coil at different positions along the coil. Although the Lorentz force is theoretically linear for an infinite coil, a finite coil will have slightly different performance parameters

at different locations along the stroke. In order to perform this calibration, a weak spring with a spring constant around 1000 N/m is positioned in the instrument. The position of the spring is then varied incrementally and dynamic data is taken at each of these points. Linear system identification is performed and the fitted values are reported to determine the region of linearity in the coil. For more information on the linear system identification techniques, see Section 3.1. Results from subsequent tests are mapped against the calibration curve to assure that the information is taken in a linear region of the actuator.

The calculated parametric values for different positions on the coil from one such calibration are shown in Figure 2.12. The bode plot is also shown to the right where the damping decreases as the position value increases. The lower position indicates that the coil is further inside the bearing structure. The coil has a 32 mm stroke and a region of about 20 to 25 mm that is relatively linear. The nonlinearity near a position of zero is due mostly to the fact that the magnet has reached one end of the coil and the top portion of the bobbin may be hitting against the magnet structure. For positions near 32 mm, much of the coil has exited the controlled field of the magnet structure. As less and less coil is available, the output force decreases and the system identification software perceives that the resistance to movement has increased. This is interpreted as an increase in the spring constant and effective mass (because the calibration was completed with gravity pointing toward higher positions). In general, calibrations are completed in the same orientation as the subsequent test in order to match calibration curves with curves from measured data.

Table 2.1: Instrument calibration versus skin parameter values

	Instrument Values	**Skin Values**
VAF*	95 to 98%	70 to 95%
M	0.06 to 0.08 kg	0.06 to 0.10 kg
B	3 to 4 Ns/m	5 to 50 Ns/m
K	1000 to 1200 N/m	100 to 7000 N/m

*See definition of Variance Accounted For in Equation 3.13.

The nonlinearities in the coil are much smaller than the nonlinearities seen in the properties of skin. Table 2.1 shows a comparison of the instrument versus the

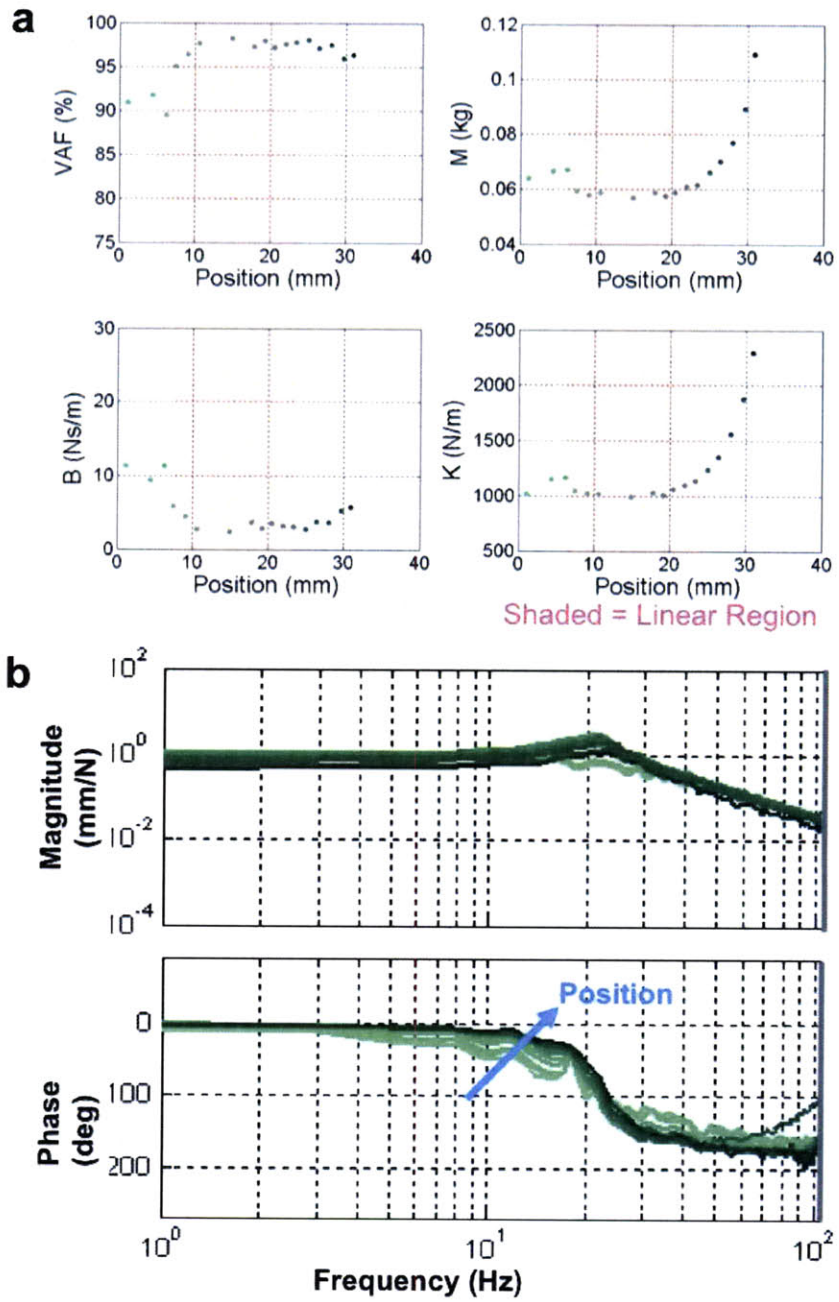


Figure 2.12: (a) Device calibration curves in parameter space showing the Variance Accounted For (VAF), the mass, the damping, and the spring constant associated with different positions along the coil. (b) The associated curves in frequency space are also shown. The position reference increases as the coil is extended outwards (into the skin).

nonlinearities seen in skin samples during indentation testing. Note that the mass and damping measurements reflect the contributions of the coil and skin while the spring constant is represented either the calibration spring or the skin under indentation.

The variation on the calibration spring is about 1 to 1.2 times the actual value in the linear test range.

2.4.2 System Identification Software

The system identification program is designed to give instructions to the user on how to operate the instrument. When the system is started, the program will instruct the user to place the probe on the tissue and to be sure to hold the reference surface firmly in position. The system is then started with a button click. The input applied consists of a initial preload followed by a stochastic signal. When the data is acquired, the program sends the information from all the signals to a MATLAB node which then computes the outputs. The output of the MATLAB node is then displayed on the screen. If the user wishes to complete another test, the start button is pressed and the process begins again.

Figure 2.13 shows the interface used for system identification. Additional details on the LabVIEW program are listed in Appendix B. The input is generated before the test and stored in a file. The input signal can be augmented in magnitude and offset in the program. The first plot on the left shows the input voltage, force and position. Moving to the right, each of these signals is then broken down by chopping off the preload section of the data. The power spectrum of all the signals are calculated to check for high frequency noise. The mean squared coherence of the input force versus the output position are calculated to identify portions of system linearity. In the second row of figures, the impulse response is calculated along with the static nonlinearity. Both of these are accompanied by a parametric fit. The parametric and nonparametric models and the measured output data are then compared. A bode plot can be created for the system and is displayed in the figure on the bottom right. This series of plots is used to check the consistency of the data before being processed by more advanced system identification schemes. Additional details on system identification are discussed in subsequent chapters.

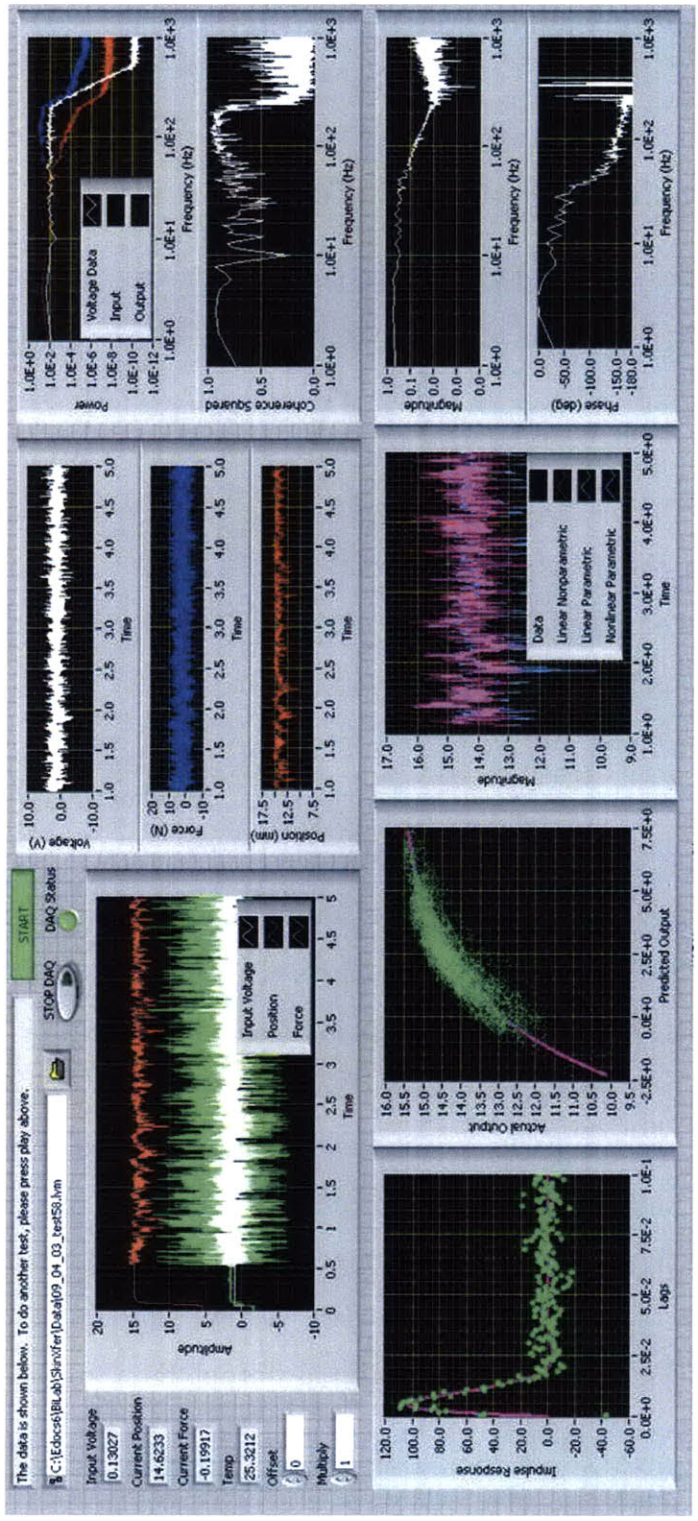


Figure 2.13: LabVIEW user interface which takes measurements, completes system identification and displays information to the user.

Chapter 3

Linear and Static Nonlinear Techniques

To minimize patient discomfort in a clinical setting, skin measurements must be done quickly but accurately. In addition, the results must be both repeatable and unaffected by patient movement. Stochastic system identification techniques, mathematical techniques capable of obtaining a large amount of data from a quick test, are immune to influences uncorrelated to the input. This chapter develops techniques that can be used to estimate a skin model in approximately two to four seconds.

System identification is a powerful tool that has been well established for pure linear systems and some classes of static nonlinearities. Skin and the underlying tissue is dynamically nonlinear but can be approximated, to first order, with a cascade model composed of a linear component followed by a Wiener static nonlinearity. Several linear and static nonlinear system identification techniques exist that can identify global monotonic or non-monotonic nonlinearities. Using the technique developed by Hunter and Korenberg [49, 69, 67], the linear dynamic component and the static nonlinear component of tissue can be identified using white, Gaussian inputs or non-white, non-Gaussian inputs. Results from indentation tests on skin are shown to motivate the use of more advanced tools for system identification. Additional experimental data is displayed in Chapter 7.

3.1 Linear Stochastic System Identification

A linear system is one that can be described by a linear combination (with gains and lags that represent storage and absorption) of the inputs and outputs. Many mechanical systems, such as motors, are well represented by linear time invariant (LTI) assumptions. Many nonlinear systems can be simplified to a linear system when the perturbations are small.

Linear systems have many desirable properties that allow them to be easily manipulated and identified. One of the important factors necessary for identifying a linear system is its impulse response, which is a measure of how the system responds to an instantaneous input pulse. The impulse response also fully characterizes the dynamics of a linear system. It can be understood as how the energy of the input delta function is stored in the system dynamics and released over time. Any input profile can be broken up into a series of delta functions and the corresponding output impulse responses can then be added together to produce the overall output. Mathematically, this is represented by the convolution of the input with the system impulse response. With linear systems theory, the problem of identifying a complex system can be boiled down to looking for the system impulse response.

Real systems are continuous and can be converted into the frequency domain using a Fourier or Laplace transform. On the other hand, when data are collected, the representation of the real system is discrete. Discrete systems can be converted to the frequency domain using discrete Fourier transforms or Z-transforms. There are several different categories of discrete systems, two of which are the finite impulse response systems (FIR) and infinite impulse response (IIR) systems. In FIR systems, the output is only a representation of lagged input variables thereby resulting in an impulse response that will approach zero as the lag approaches infinity. In IIR systems, the impulse response depends on the input and the output and can therefore have a finite value as the lag approaches infinity. IIR systems can drift from their equilibrium value because the absolute position of the system output depends on the entire history of the input. FIR systems are much simpler to model and have a finite

memory length. Memory length can be thought of as the amount of time it takes the system to release the energy it has stored from the input.

Since linear time invariant FIR systems can be represented by a finite length impulse response, the problem of identifying the system can be reduced to identifying the impulse response up to its memory length. How can this be done? The most obvious solution is to apply a delta function input to the system and measure the impulse response directly. This, however, is impossible to do since it is impossible to produce a perfect delta function. Other simple inputs like step inputs can also be used to identify system dynamics. However, there is uncorrelated noise which would require the test to be completed multiple times. Yet another method is to use sine waves of different frequencies to identify the amplitude and phase at different frequencies. This also has additional drawbacks. In conducting a test with a sine wave, the first few samples will always be plagued with transients from frequencies unrelated to the input frequency of the sine wave. This means the first few samples cannot be used when fitting the sine wave. It also takes a long time to complete tests for multiple frequencies.

A more advanced statistical method of determining the impulse response would be to use stochastic system identification to test all the frequencies at the same time. For a linear system, different frequencies can respond independently from other frequencies. A signal that contains information from all frequencies with equal probability is a white signal. There are obvious benefits over step responses and sine waves. A step response has a single input from which a single impulse response can be collected. A single data set gives a poor estimate of the actual system dynamics. In order to be relatively certain of the response of the true system, many samples need to be collected. For the same amount of time it takes to obtain a single estimate from a step response or from one frequency in a sine wave test, a stochastic input signal uses many inputs which produce multiple estimates for the impulse response. This boosts the sensitivity of the measurement.

Stochastic system identification has many other desirable properties. A white input is inherently uncorrelated with itself thereby producing an autocorrelation of

zero for all lags other than at the zero lag point. This means that the cross correlation of the input to the output is a good representation of the impulse response. Therefore, aspects of the output that are uncorrelated with the input will be statistically averaged away by the correlation functions. This produces a robust identification technique that can ignore uncorrelated noise.

Several parametric and nonparametric methods for linear stochastic system identification exist. Assuming that the system has a finite impulse response (FIR) for the discrete transfer function, the impulse response can be calculated many ways including, but not restricted to, the following methods:

1. Frequency domain techniques can be used. This involves dividing an appropriately windowed power spectral density of the input-output relation by the power spectral density of the input. An impulse response can be obtained from an inverse Fourier transform. These methods are exhibited in Section 3.1.2

2. Time domain techniques including least mean squares (LMS), recursive least squares (RLS), and adaptive least squares (ALS) can also be used. The LMS algorithm is used for off-line system identification (described in Section 3.1.3 and the RLS and ALS algorithms can be used for on-line real-time system identification.

3. Orthogonalization of the input with different fitting functions can also be used to look at linear systems. For orthogonalization, a Gram-Schmidt algorithm can be used [66]. For some colored inputs, however, this method can create high frequency noise in the solution. This would require a second filtering step before obtaining the impulse response. These methods are explained in Section 4.1.

3.1.1 Input Generation

Distribution

Stochastic inputs can have a variety of distributions and colors. Gaussian white inputs or Brownian process inputs are the workhorses of classical system identification

methods. In fact, several linear and nonlinear system identification methods are derived expressly for an input with given probability and autocorrelation characteristics [74, 48]. When a Gaussian distribution is put into a linear system, the output is a Gaussian distribution. When other distributions are put into a linear system, the output distributions will change and will often approach a Gaussian. For nonlinear systems, however, an input Gaussian may turn into another distribution as shown for skin under indentation in Figure 3.1.

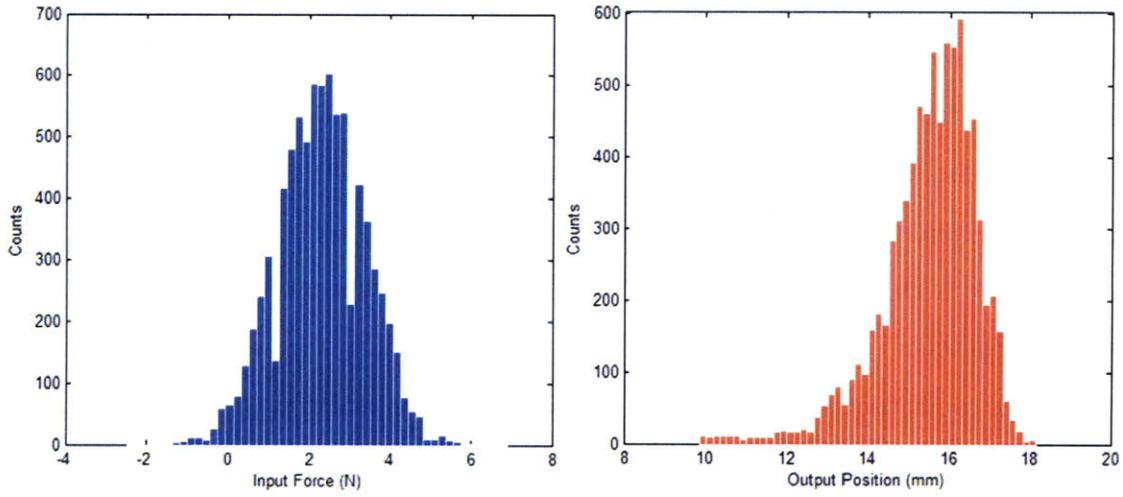


Figure 3.1: Input and output distributions for skin under indentation testing.

Despite the benefits of Gaussian inputs, other inputs also have merits in other situations. In linear systems, the range being tested does not change the system dynamics so it is theoretically possible to test a small range of inputs to understand the entire system. Therefore, it is optimal to use Gaussian inputs with low input ranges which satisfy certain mathematical proofs like the Cramer-Rao lower bound for the maximum variance of estimators [74]. When noise is added into the equation, it then becomes important to increase the input range in order to obtain a better sensitivity.

Real systems have physical limits; actuators may not be able to achieve certain high bandwidth inputs, high bandwidth inputs may damage the system (or in the case of tissue mechanics, be painful for the patient), or the measured input (*e.g.* current) may not be the same as the programmed input (*e.g.* voltage). In order to

measure the full range of the output, it is important that the dynamic range of the sensors be larger than the dynamic range of the system. It is also important that the input spectrum takes full advantage of the dynamic range. For the nonlinear tissue mechanical system, it is especially important to obtain a large output distribution because the manifestation of the static nonlinearity only happens when a large range of positions is explored. Smaller displacements will only show linearized regions of the overall nonlinearity.

One way to characterize an input is to use the crest factor. The crest factor is defined as the peak to root mean square (RMS) average ratio of the input. Since it is important that the RMS value of the input takes advantage of the full range of the real system, the optimal value for the crest factor is one. This means that the peak value is equal to the RMS value. Several signals satisfy this criteria including a DC signal at the maximum input value. A DC signal, however, does not have any frequency content.

Another input that satisfies this criteria and also has a large input bandwidth is a stochastic binary signal. This is composed of an input which varies randomly between the maximum and minimum possible input. The distribution for a stochastic binary is bimodal. When working with biological tissues *in vivo*, additional factors need to be taken into consideration when generating an input. Stochastic binary signals have high frequency content with sharp transitions; which result in subjects experiencing pain. Other types of input signal distributions, such as Gaussians and uniforms, can be generated that test the full dynamics of the input painlessly.

Frequency Content

In addition to input range, it is possible to characterize a signal by its color or frequency spectrum. There is no standard metric for determining the optimal input bandwidth. Nevertheless, there are a balance of factors which can help determine the best range of input bandwidths. In order to derive adequate information to describe the system, a tailored input must be generated with a sufficiently wide bandwidth and output distribution. For the purpose of understanding system dynamics, the input

frequencies must extend well above the system's highest frequency characteristic, such as the natural frequency. The ability for the model to account for the information (such as the variance accounted for or the residual sum of squares) is not a good measure because using a lower bandwidth can increase the overall ability to predict the output but represents a loss of the higher frequency information of the system rather than a better understanding of the overall system dynamics. These mechanisms tend to widen the bandwidth of the input signal.

In order to maintain a reasonable range without excessive input power (the input power limit of the actuator, for example), the bandwidth must be decreased. In addition, the sampling frequency limits of the data acquisition system also limit the maximum input frequency. It is also undesirable to sample at a different frequency than the input since this may cause aliasing. Therefore, the high frequency content of the input should be heavily low pass filtered. These forces tend to decrease the input bandwidth.

A range of inputs were attempted for this system and it was found that a Gaussian white input with a cutoff frequency of 200 Hz (which is well above the natural frequency of the system) implemented with an 8th-order Butterworth filter was appropriate. Other distributions, such as a uniform distribution, were also generated by the method described in Hunter and Kearney [48] with a cutoff frequency of 200 Hz. Methods for generating these inputs are discussed in Chapter 6.

Test Length

In linear systems, the test length is dependent on the memory length (in seconds) of the system. For natural frequencies near 30 Hz with reasonable amounts of damping, the memory length is shorter than 0.1 seconds. This indicates that as little as 0.1 seconds is needed to identify a linear system with dynamic content up to 200 Hz and a damping parameter (or damping ratio) greater than 0.7. As a rule of thumb, the test length (in number of samples) is at least a factor of three greater than the number nonparametric values being fitted [68, 64] when no correction is necessary for the initial conditions. When initial condition corrections must be made, longer test

lengths are required. For linear systems, this means that the test length should be at least three times longer than the memory length or around 0.3 seconds. The longer the test length, the more statistical averaging occurs which means that the impulse response will be closer to the real value.

In nonlinear systems, the test length may be dependent on other factors such as the order of the nonlinearity that is being identified or input range restrictions. An absolute minimum of 2 seconds is needed to fully explore the range of inputs using Gaussian inputs. In order to predict a Wiener static nonlinearity, a final test length of 4 seconds was used.

3.1.2 Frequency Domain Techniques

After generating the input, it is important to revisit the underlying assumptions for high frequency system dynamics and for linearity. There are several frequency domain techniques that can be used as an aid to this assessment. One of the techniques used to look at the frequency content of a signal is the power spectral density (PSD). When the PSD is calculated, Welch's algorithm should be used with properly chosen windowing functions to minimize noise in the frequency domain. The PSD can be used to estimate the signal to noise ratio and to determine the noise floor of a particular signal.

Plots of the signal power spectra from an indentation test into human skin are shown in Figure 3.2a. This plot shows that the input goes up to 200 Hz before an 8th order low pass filter is implemented. The voltage signal level drops 8 orders of magnitude before hitting the noise floor. The corresponding force output of the linear voice coil shares the same cutoff frequency as the input voltage and drops about 4 orders of magnitude before hitting its noise floor. The position output of the tissue and linear voice coil system has its own dynamics which begin to appear near 30 Hz. The DC signal level and the high frequency noise floor differ by about 6 orders of magnitude.

One of the the techniques used to look at the linear regions of a signal is the mean

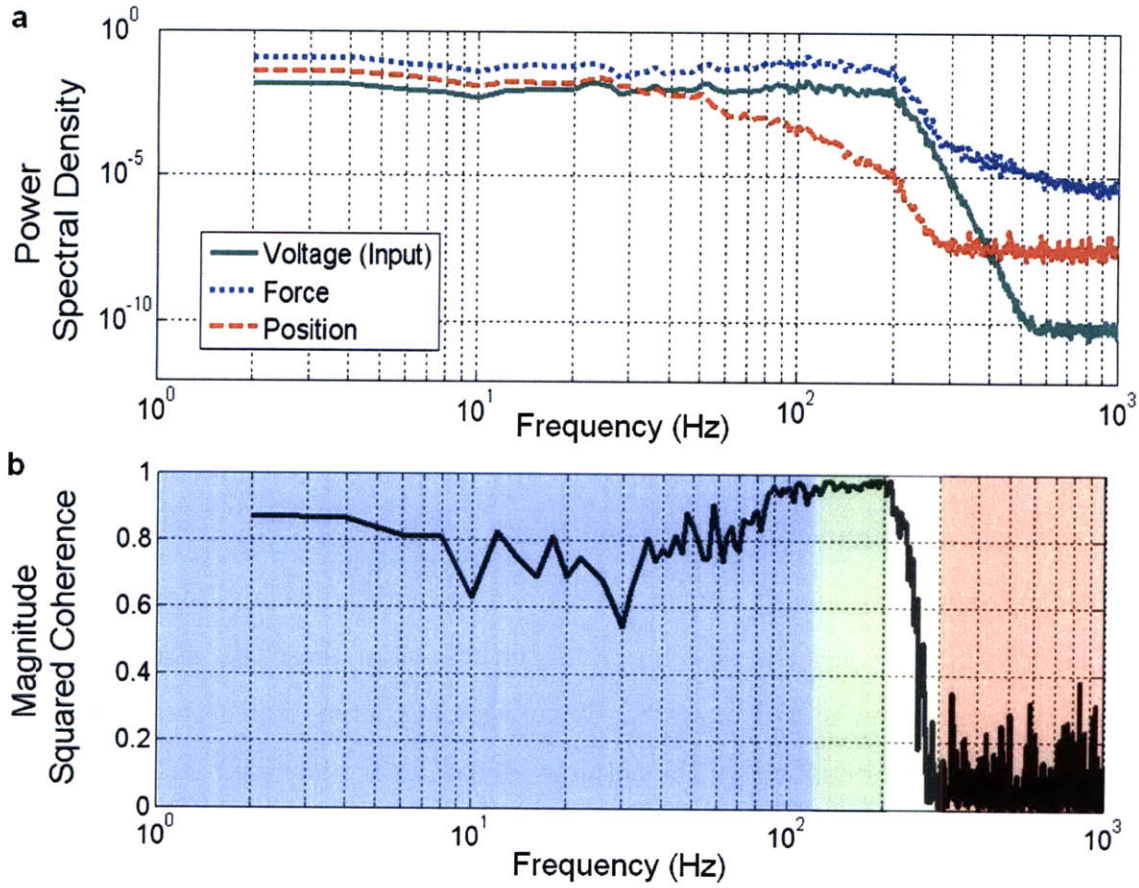


Figure 3.2: (a) Power spectral density of the input voltage, the measured force, and the measured position of the instrument during a nonlinear stochastic measurement of the skin during indentation on the left posterior forearm 40 mm from the wrist. (b) The mean squared coherence of the input force to output position shows the frequency ranges that can be explained by linear elements.

squared coherence (MSC) defined as,

$$C_{xy}(\omega) = \frac{|\Phi_{xy}(\omega)|^2}{\Phi_{xx}(\omega)\Phi_{yy}(\omega)}, \tag{3.1}$$

where Φ_{xx} is the power spectra of the inputs, Φ_{yy} is the power spectra of the outputs and Φ_{xy} is the cross power spectra of the input and the output. Note that this function cannot be implemented as written and should, rather, be implemented with algorithms such as those developed by Welch using windows and overlapping signal components. This is shown in Figure 3.2b.

The MSC of skin can help determine how much of the system can be explained

with linear assumptions. In the region between DC and about 100 Hz, the coherence between the input and output is about 0.8 which means that it is only partially successfully explained using linear assumptions. From 100 to 200 Hz, the MSC is almost unity which means that it is well modeled with linear assumptions. Since there is only input power below 200 Hz, the behavior of the MSC above 200 Hz is not relevant. From 200 Hz to 300 Hz, the MSC drops quickly due to the lack of input power. Above 300 Hz, the low input power makes it difficult to determine the linearity of the system and the MSC is effectively zero. This indicates that skin during indentation may be well modeled to first order using a simple linear system. As expected from arguments in 2.1, nonlinearity is a function of output range.

In the frequency domain, a linear system can be modeled with a Bode plot which breaks down the complex response into a magnitude and a phase lag. The magnitude and phase convention is used in system dynamics and controls and serves as a better representation of systems where the actuator dynamics are included in the measurement. Other conventions for frequency domain representations include breaking the complex response into a real and an imaginary component. This is sometimes used in material science to represent the real and complex storage of a material [101].

Figure 3.3 shows the Bode plot of the compliance (force to position transfer function) of the skin and instrument system. These plots were obtained using the same windowing functions as the PSD and MSC plots. Note that since the input power goes only up to 200 Hz, the information above 200 Hz is removed from the plot.

From the shape of the Bode plot, it is clear that the system is second order up to 200 Hz. The indicators for this include the slope of -2 in the magnitude plot at high frequencies and the corresponding phase approaching -180 degrees. The natural frequency of the system is at 36 Hz (226 rad/s) and the damping parameter is between 0.5 and 1. Obtaining a more specific damping parameter or a damping constant from a noisy Bode plot is generally more difficult. Since the mass of the actuator is known to be 60 g and additional estimates show that the total effective mass is about 91.2 g, a preliminary estimate of the average spring constant of skin can be obtained at around 4.67 kN/m on average for a traversed depth of about 7 mm. Caution is

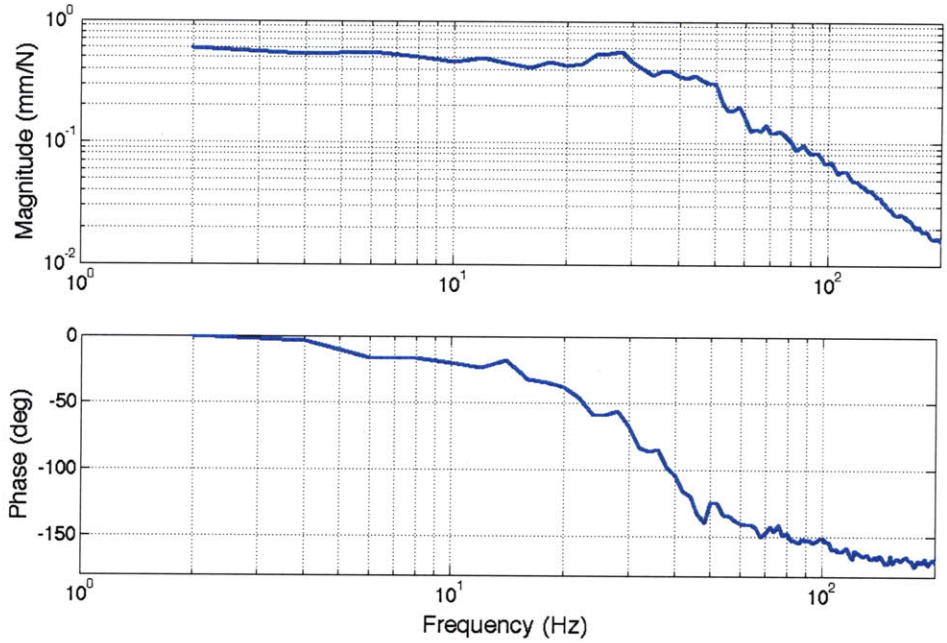


Figure 3.3: Bode plot for force to position relationship for skin. The cutoff frequency is near 36 Hz for this configuration of input mass and skin compliance.

needed to interpret these values. It is important to realize that because the system is essentially nonlinear and depth dependent, the spring constant should vary with depth and will tend to be larger for locations deeper into the skin than values quoted for the surface of the skin. Note that even values for the stiffness of the skin surface will vary by orders of magnitude. Boyer quotes the spring constant of skin at the surface as 25 N/m [6] and this serves as a lower bound to results quoted in this work.

The estimate obtained above is rough and curve fitting in the frequency domain is often impractical. In addition, the choice of windows make fitting subjective to the choice of window length. Time domain techniques for identification, on the other hand, have the benefit of being nonparametric and robust to uncorrelated noise. The time domain techniques produce nonparametric impulse responses which are generally easier to fit to parametric values. The one drawback of time domain techniques is that details like the order of the system are more difficult to determine. Using both system representations is generally helpful for initial identification.

3.1.3 Least Mean Squares

A least mean squares method is essentially the optimal solution to fitting a linear dynamic equation under the assumption that uncorrelated white noise exists on top of the output signal. This method is completed once all the data has been obtained and is therefore an off-line identification method. The derivation for LMS is covered in some system dynamics textbooks [74] so the solution will be given here without proof. The impulse response can be derived from taking a partial derivative of the mean square error with respect to the coefficients of the impulse response.

The conceptual idea behind LMS is that stochastic input can be compared with the output to look for correlated components. Taking the autocorrelation of the inputs and the cross correlation of the outputs removes the uncorrelated noise from the system dynamics via statistical averaging. The two sided auto and cross correlations are shown in Figure 3.4. If the input is perfectly white, then the cross correlation is essentially the impulse response of the system. Since no finite length input can have a perfectly white spectrum, additional manipulation is necessary to obtain the impulse response. Since the impulse response convolved with the system input produces the output, one can obtain the output by an "inverse convolution" of the input with the output. This "inverse convolution" is known as the Toeplitz matrix inversion. The main diagonal of the Toeplitz matrix is the autocorrelation at zero lags and the upper and lower triangular portions of the matrix are symmetric for a symmetric autocorrelation function.

The optimal impulse response satisfies Equation 3.2 where F_s is the sampling frequency. Equation 3.3 shows the biased autocorrelation function of the force input (where $x = F$) while Equation 3.4 shows the biased cross correlation between force input and position output (where $y = P$). Equation 3.5 shows a Toeplitz matrix of the input autocorrelation function. The noise assumptions for this least mean squares equation are that the output noise is white Gaussian and uncorrelated with the input.

$$\widehat{h}(m) = F_s \left[R^{-1} \phi_{xy}(m) \right] \qquad \text{for } m = 1 \ldots M \tag{3.2}$$

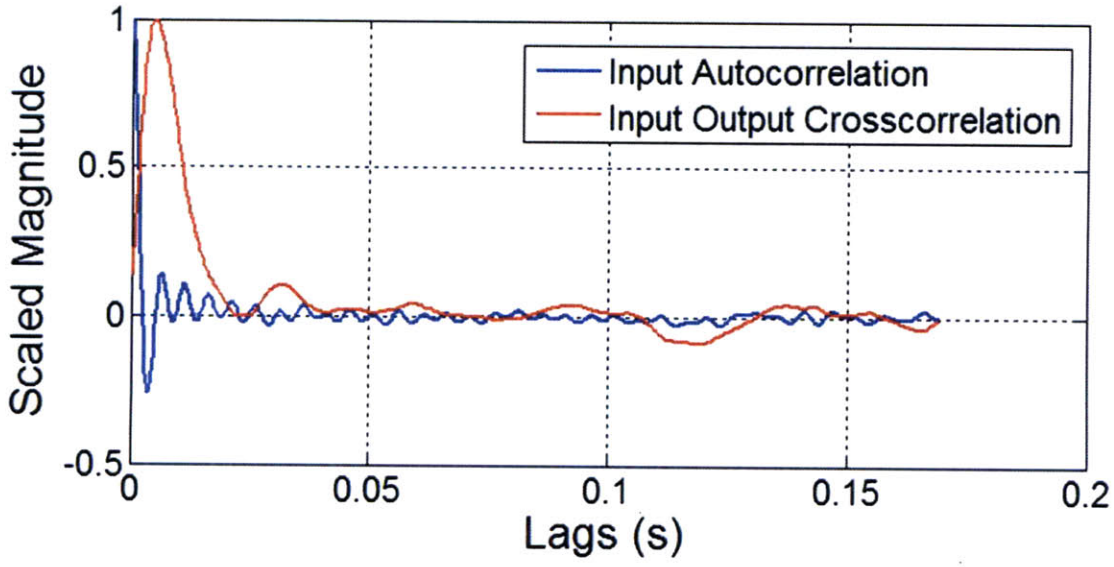


Figure 3.4: The normalized autocorrelation of the inputs and the input output cross correlation are shown. A perfectly white input will have an autocorrelation that is zero for all lags except at zero lag.

where

$$\phi_{xx}(n) = \frac{1}{N}\sum_{i=1}^{N} x(i)x(i+n-1), \tag{3.3}$$

$$\phi_{xy}(n) = \frac{1}{N}\sum_{i=1}^{N} x(i)y(i+n-1), \tag{3.4}$$

$$R = \begin{bmatrix} \phi_{xx}(1) & \phi_{xx}(2) & \phi_{xx}(3) & \dots & \phi_{xx}(M) \\ \phi_{xx}(2) & \phi_{xx}(1) & \phi_{xx}(2) & \dots & \phi_{xx}(M-1) \\ \phi_{xx}(3) & \phi_{xx}(2) & \phi_{xx}(1) & \dots & \phi_{xx}(M-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{xx}(M) & \phi_{xx}(M-1) & \phi_{xx}(M-2) & \dots & \phi_{xx}(1) \end{bmatrix}. \tag{3.5}$$

Efficient inversion techniques, such as singular value decomposition, can be used to produce impulse responses from Equation 3.2. Windowing and other associated algorithms are available in Appendix C.1.1. For a data length of N, the maximum lag M corresponds to the memory length of the system. As the data length N increases, the effect of initial conditions dissipate such that it is beneficial to have longer data lengths even if the memory length M is low. This is especially true for static nonlinear systems because the correlations should include as much of the output

range as possible. The resulting impulse response is shown in Figure 3.5.

One special point to note is that the derivation for the LMS equations requires that the input and output both have zero means (or else the the autocorrelation and cross correlation will look like decreasing ramps rather than centering around a value of zero). In addition, the derivation assumes zero initial conditions (zero for the input and all high order derivatives of the input at a time of zero). The effects of nonzero initial conditions approach zero as the test length increases. In practice, a holding period of a few milliseconds at the average input force before the start of the stochastic signal is good for approximating zero initial conditions.

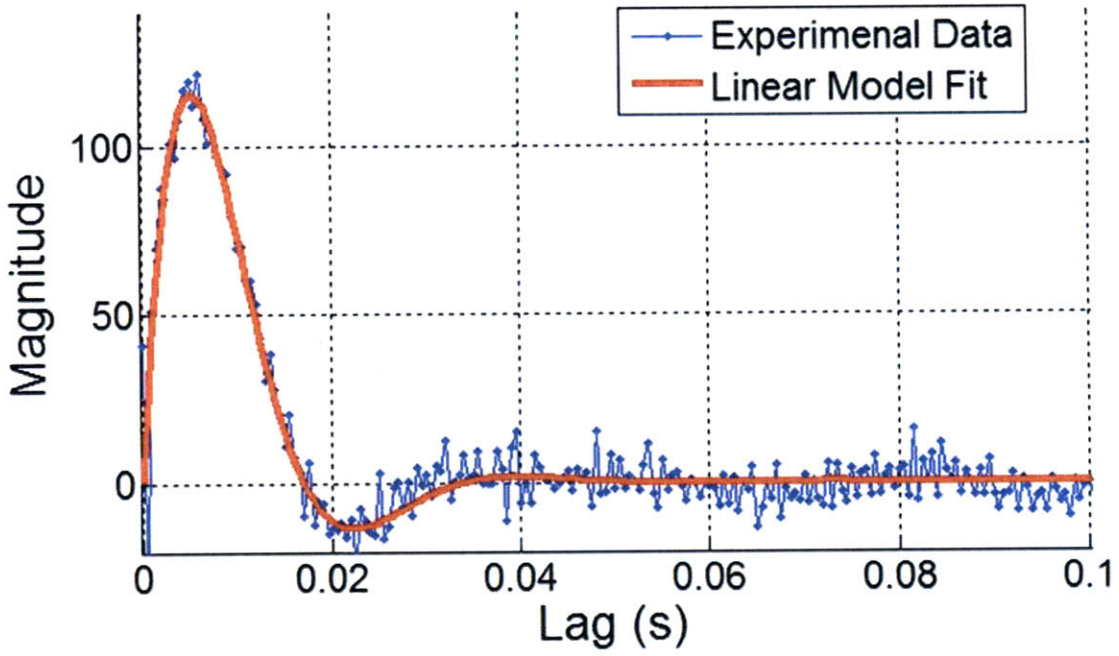


Figure 3.5: Impulse response with parametric fit.

In Figure 3.5, the impulse response dies down quickly indicating that the damping parameter in the system is high but the fact that the impulse response becomes negative also indicates that the damping parameter is not greater than one. The nonparametric estimates, shown as blue dots, are grouped around the average value of the impulse response. A longer test length or a slower sampling rate can reduce the noise in the estimate.

The form of the fitted impulse response is second order based on the frequency

domain analysis up to 200 Hz. From the inverse Laplace transform of the output of the system, the form of a second order impulse response can be obtained,

$$\widehat{h}(t) = A_1 sin(A_2 t) e^{A_3 t}, \tag{3.6}$$

where $\widehat{h}(t)$ is the impulse response, t is the lag in seconds, and A_1, A_2 and A_3 are the fitted parameters. This form was chosen for fitting because each parameter appears only once and convergence occurs more expediently. The physical second order system can be parameterized as a mass M_e (contributions from coil mass and effective inertia of the skin), spring K_e (contributions from average skin stiffness) and damper B_e (contributions from friction, eddy current damping, skin damping) system in Laplace space as

$$\frac{P(s)}{F(s)} = \frac{1}{M_e s^2 + B_e s + K_e}. \tag{3.7}$$

The relations that convert from Laplace space to the impulse response are shown below for a damping parameter ζ, natural frequency ω_n and sampling frequency F_s.

$$\zeta = \left(\frac{A_3^2}{(A_3^2 + A_2^2)} \right)^{1/2}, \tag{3.8}$$

$$\omega_n = \frac{A_2}{\sqrt{1-\zeta^2}}, \tag{3.9}$$

$$K_e = \frac{1}{A_1} \frac{\omega_n}{\sqrt{1-\zeta^2} F_s}, \tag{3.10}$$

$$M_e = K_e / \omega_n^2, \tag{3.11}$$

$$B_e = 2\zeta \omega_n M_e. \tag{3.12}$$

In order to assess the goodness of the fit, we can define the Variance Accounted For (VAF) as,

$$VAF = 1 - \frac{\sigma_{y-y^*}^2}{\sigma_y^2}, \tag{3.13}$$

where σ_{y-y^*} is the standard deviation between the measured and the predicted outputs and σ_y is the standard deviation of the measured signal. The VAF is normally

quoted as a percentage where 100 % VAF shows a perfect match between the predicted model and the measured system. Typical values of the VAF that produce reasonably good models are above 95 %. This gives little resolution between reasonably good models so additional metrics are used for models with good performance. A method of measuring the negative impact of the entropy from additional parameters in estimation is called the Akaike Information Criteria (AIC) [74],

$$AIC = 2k + N\left[ln\left(2\pi\frac{R_{ss}}{N}\right) + 1\right], \tag{3.14}$$

$$R_{ss} = \sum_{N}^{i=1}(y(i) - \widehat{y}(i))^2. \tag{3.15}$$

Where k is the number of parameters in the model, N is the number of observations, R_{ss} is the residual sum of squares, y is the measured output and $\widehat{y}$ is the predicted output. The equation above assumes that the variance of the model error is unknown but equal, thereby setting the maximum likelihood function. When the number of parameters is increased, the estimated fit improves. However, the AIC will also penalize the addition of parameters and discourages over-fitting [74]. Therefore, the best values of AIC are those with the lowest values. There are several corrections to the AIC. One of them is for small sample sizes called the AICc which makes a second order correction,

$$AICc = AIC + \frac{2k(k+1)}{N-k-1}. \tag{3.16}$$

The AIC is not generally robust to different representations of continuous systems. This means that one sampling rate with a certain memory length will obtain one set of values for the AIC for different models while a different sampling rate may give a completely different set of values, possibly with a different order. The AIC and the VAF should not be used to test the validity of a single model (hypothesis testing) but should be used to compare different models for relative efficiency.

With these equations, it is possible to parameterize the nonparametric estimate of linear components of the system and assess the goodness of fit. The effective mass was found to be 0.0912 kg (the measured probe and bobbin mass was 0.060 kg), the

effective damping was found to be 22.77 Ns/m, and the effective spring constant was found to be 4.67 kN/m. This corresponds to a damping parameter of 0.55 and a natural frequency of 226 rad/s. By convolving the impulse response with the input, it is possible to create a prediction of the output which can then be compared with the actual output. This can then be used to obtain a VAF for the nonparametric model of 75.79 % and for the parametric model of 75.64 %. The second order parametric model is therefore a useful simple representation of the system dynamics.

The values obtained from such a fit are valid only for a certain output range and represent the average parameter values within that range. In order to obtain more general estimates of the system parameters that are valid for other input ranges, a nonlinear system identification technique is required.

3.2 Static Nonlinearities

Nonlinearities can come in many different forms and different representations can highlight different aspects of a nonlinear system. A dynamic nonlinearity can be solved using different basis functions, different higher order expansions, or perturbation expansions. A static nonlinearity, on the other hand, can be represented as a system block in the time or frequency domain and can interact with linear dynamic elements as a simple transformation of the dynamic input data in the time domain.

There are many different types of static nonlinearities and each of them requires a slightly different analysis technique. Two of the most fundamental types are Wiener and Hammerstein nonlinearities. Schematics of these are shown in Figure 3.6. Hammerstein nonlinearities occur when inputs to a linear system are transformed with a nonlinear function in the time domain. Wiener nonlinearities occur when the outputs of the linear system are transformed by a nonlinear function in the time domain. Static nonlinearities and linear systems can be cascaded in different combinations to obtain more complex nonlinearities [67]. Static nonlinearities can occur in many different situations in nature. Examples include light intensity output as a function of input current in a filament light bulb and action potential polling in knee-joint

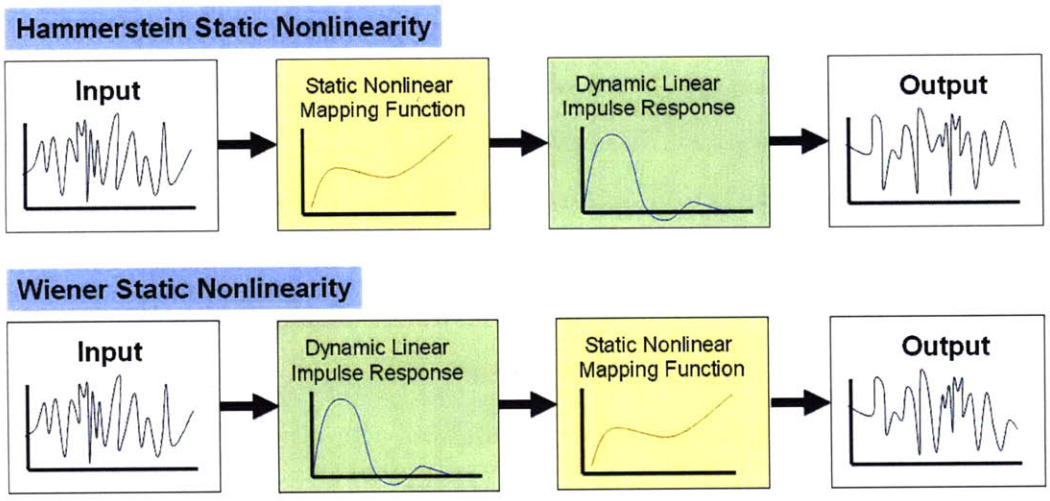


Figure 3.6: Wiener and Hammerstein static nonlinearities show how linear system dynamics can be combined with nonlinear functions

dynamics [61]. If the force from the instrument can be considered to have linear dynamics, then it is predicted that the skin may act, to a reasonable approximation, like a Wiener system where the nonlinearity in the system is a static nonlinearity at the output of a linear system. Methods for identifying Wiener systems will be discussed below while more advanced methods for looking at Wiener and Hammerstein systems are discussed in Chapter 4.

Hunter and Korenberg [49] proposed that Wiener systems could be solved by separating the nonlinear and linear components of the signal. The procedure is shown schematically in Figure 3.7. Other recursive techniques [9, 106], subspace [105, 108] and orthogonalization techniques [68] can also be used. The real Wiener static nonlinearity is a nonlinear function $g(\bullet)$ of a dynamically linear output $z(n)$. The true static nonlinear output $y(n)$ is shown in Equation 3.17 as a nonlinear transformation on the convolution of an impulse response of true memory length I and the input vector $x(n)$ such that,

$$y(n) = g\left[\sum_{i=1}^{I} h(i)x(i+n-1)\right] = g\left[z(n)\right]. \tag{3.17}$$

The algorithm to solve for $z(n)$, h(m) and $g(\bullet)$ can be outlined as follows.

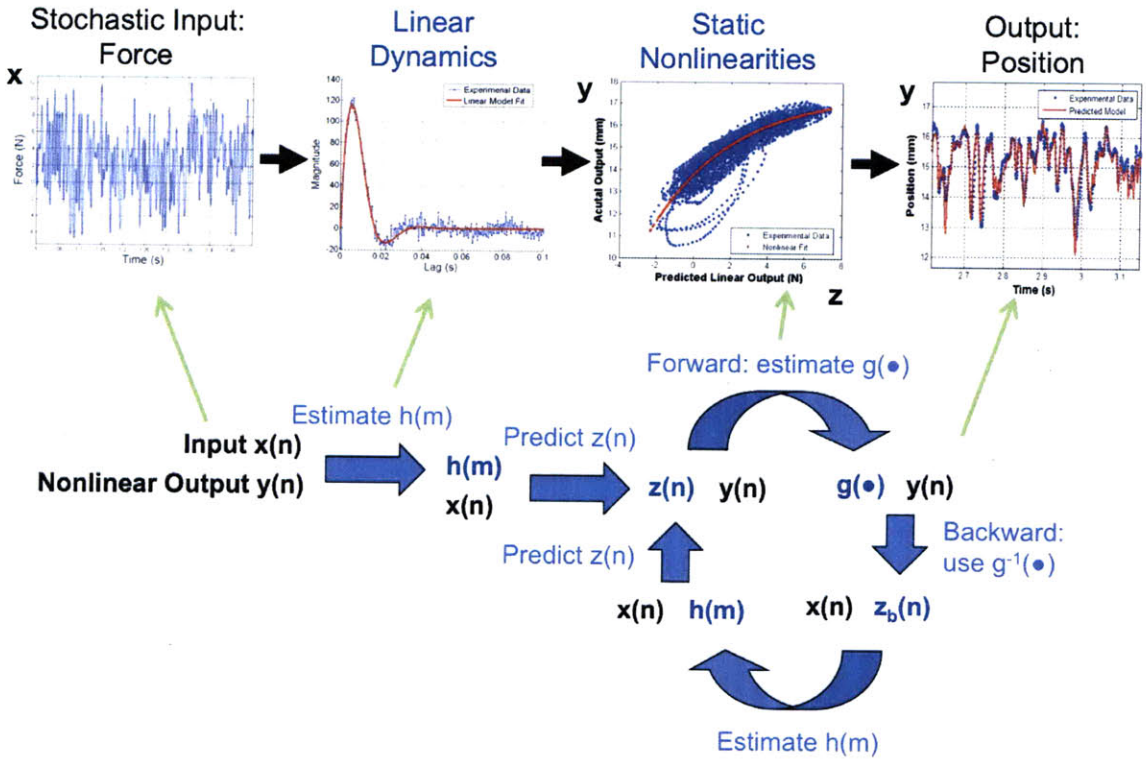


Figure 3.7: Recursive algorithm for solving monotonic Wiener systems which involves estimating the system linear dynamics from input output data, estimating the input to output nonlinearity, taking the inverse of the nonlinear function and making a new linear estimate from the input data and the inverse function.

1. The linear system is identified using LMS techniques to obtain an estimated impulse response $\widehat{h_1}$,

$$\widehat{h}_1(m) = F_s \left[R^{-1} \phi_{xy1}(m) \right], \tag{3.18}$$

where

$$\phi_{xy1}(n) = \frac{1}{Q} \sum_{i=1}^{Q} x(i) y(i+n-1). \tag{3.19}$$

2. The predicted linear output $\widehat{z_1}$ can be calculated by convolving the calculated impulse response with the input,

$$\begin{aligned} \widehat{z_1}(n) &= \textstyle\sum_{j=1}^{M} \widehat{h}_1(j) x(j+n-1) \\ &= \textstyle\sum_{j=1}^{M} F_s R^{-1} \left[\frac{1}{N} \sum_{i=1}^{N} x(i) y(i+n-1) \right] x(j+n-1). \end{aligned} \tag{3.20}$$

3. By plotting the predicted linear output $\widehat{z_1}(n)$ against the measured output $y(n)$,

the residual of the unexplained data can be obtained. If the system has a Wiener static nonlinearity, a high order polynomial or other functional form called $\widehat{g}_1(\bullet)$ can be fit to this unexplained data to obtain the static nonlinearity. A free constant can be moved between the dynamic linear component and the static nonlinear estimate. If the dynamic linear component is divided by its DC compliance, this effectively moves the value of the spring constant into the static nonlinear estimate. This serves as the first iteration estimate of the linear dynamic and static nonlinear system.

4. At this point, a joint estimate is made of the linear and static nonlinear components. To estimate the linear component, the nonlinear output is fed through the inverse function $\widehat{g}_1^{-1}(\bullet)$ of the nonlinear fit to produce the new predicted linear output in the backward direction,

$$\widehat{z_{1b}}(n) = \widehat{g}_1^{-1}\left[y(n)\right]. \tag{3.21}$$

5. A new impulse response can be found between the measured input and the new predicted linear output,

$$\widehat{h}_2(m) = F_s\left[R^{-1}\phi_{xy2}(m)\right], \tag{3.22}$$

where

$$\phi_{xy2}(n) = \frac{1}{Q}\sum_{i=1}^{Q} x(i)\widehat{z_{1b}}(i+n-1). \tag{3.23}$$

6. Convolving the new impulse response with the input, a third predicted linear output is obtained,

$$\widehat{z_2}(n) = \sum_{j=1}^{M} \widehat{h}_2(j)x(j+n-1). \tag{3.24}$$

7. This can then be used to obtain a new static nonlinearity,

$$\widehat{z_{2b}}(n) = \widehat{g}_2^{-1}\left[y(n)\right]. \tag{3.25}$$

This process typically only takes a few iterations to obtain estimates of $g(\bullet)$ and

$h(m)$ for $m = 1 \ldots M$. Note that the nonlinearity predicted in Section 2.1 is monotonic and luckily this method works well for monotonic and invertible static nonlinearities. This method, however, generates estimates of the solution that are not guaranteed to converge on the actual value. Although the estimate bias is not known for an arbitrary form of nonlinearity, it can be calculated for a simulated system with a given $g(\bullet)$ and is shown to be negligible in the case of the well-behaved nonlinearities that are being studied in this work. For noninvertable static nonlinearities, the step requiring that the inverse be taken of the nonlinear fit can be replaced with a feedback equation where the newer predicted linear output is a function of the older predicted linear output plus the error between the measured and predicted outputs [69].

3.3 Representative Results

With the recursive estimate algorithm, it is possible to obtain estimates of the static nonlinearity separate from the linear dynamics. The static nonlinearity along with a parameter fit are shown in Figure 3.8. The measured output, or the depth into the skin is shown mapped against the predicted linear output in newtons. The position shown on the y-axis is the absolute position on the actuator. This is used so that the nonlinearity can be mapped against possible nonlinearities in the actuator to show that the actuator nonlinearities are negligible. The surface of the skin is located at 10 mm on the actuator for this test. Positive values of the predicted linear output indicate forces pressing into the skin and negative values indicate pulling off of the skin. An additional contribution of 0.59 N was added during calibration to account for the contribution of gravity.

There are several features to the Wiener static nonlinearity of interest. First, the backbone of the function increases monotonically as expected. Stiffness can be calculated as the inverse slope of the nonlinear plot. It is shaped like an exponential with stiffness increasing as depth increases. The "noise" in the figure is not instrument noise because other materials that are more linear, such as a spring, show much better

estimates. It is clear from the calibration curves that the system is capable of 95 to 98 % VAF. This indicates that the "noise" in the static nonlinearity comes from other nonlinearities in the tissue rather than from the instrument. One particular point of interest is the looping at low input forces and low positions. This type of deviation can be modeled by more advanced nonlinear techniques as will be described in the following chapters and sections. A model that can reproduce the most salient features of this nonlinearity is described in Section 3.4.

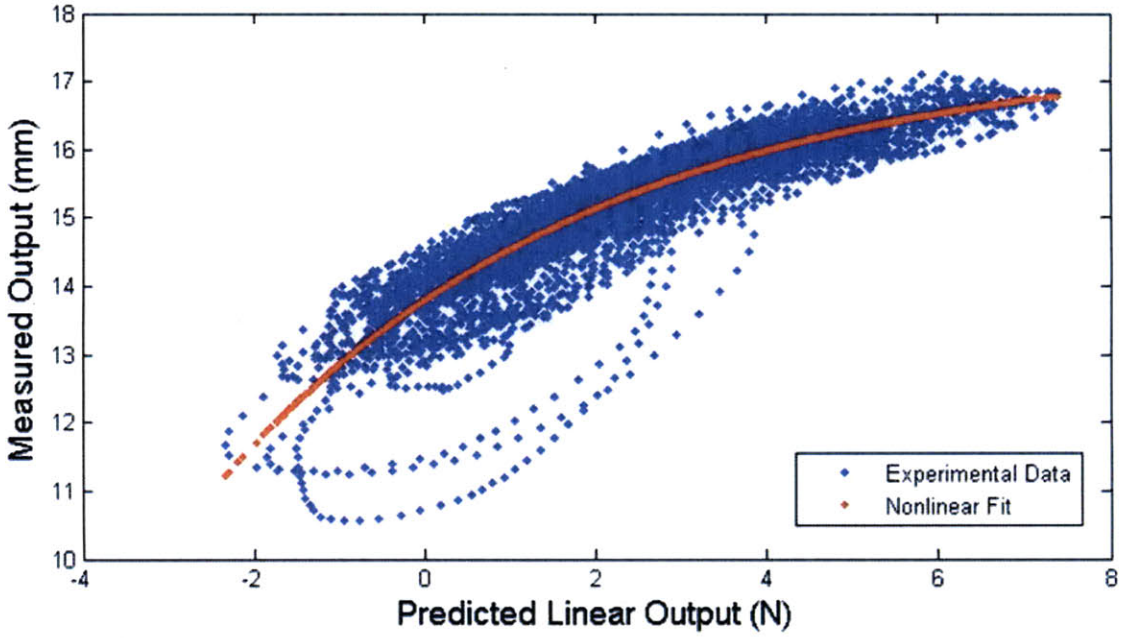


Figure 3.8: Wiener static nonlinearity with parametric fit. The measured output increases as the probe goes deeper into the skin (in mm). The surface of the skin is located at 10 mm on the actuator for this test. An additional 0.59 N was added to account for the contribution of gravity.

The fit to the static nonlinearity is,

$$y = g(z) = C_1(1 - e^{-C_2 z}). \tag{3.26}$$

In this equation, z is the predicted linear output (after several iterations) in newtons and $g(z)$ is the nonlinear function in mm. The reason for the change in units is because the nonlinear function was chosen to carry the significant units when the linear dynamics are scaled by a set constant. In this form, the change in stiffness as

a function of depth can be represented completely by $g(z)$.

The data shown above is the output after six iterations. In order to determine convergence of the nonparametric estimates, parametric values and VAFs are calculated at each iteration point. Figure 3.9 shows these values up to 6 iterations with the scaling constant chosen to be 1000 N/m. The initial estimate for this type of well-behaved nonlinearity is very close to the final values, so the estimates with no iteration can be used as an assessment of tissue if the application has limitations on computation time. The iterations converge quickly and at around 6 iterations, the nonparametric model does not change significantly as exhibited by the parametric values. Figure 3.9a shows how the different VAF's vary as the iterations progress.

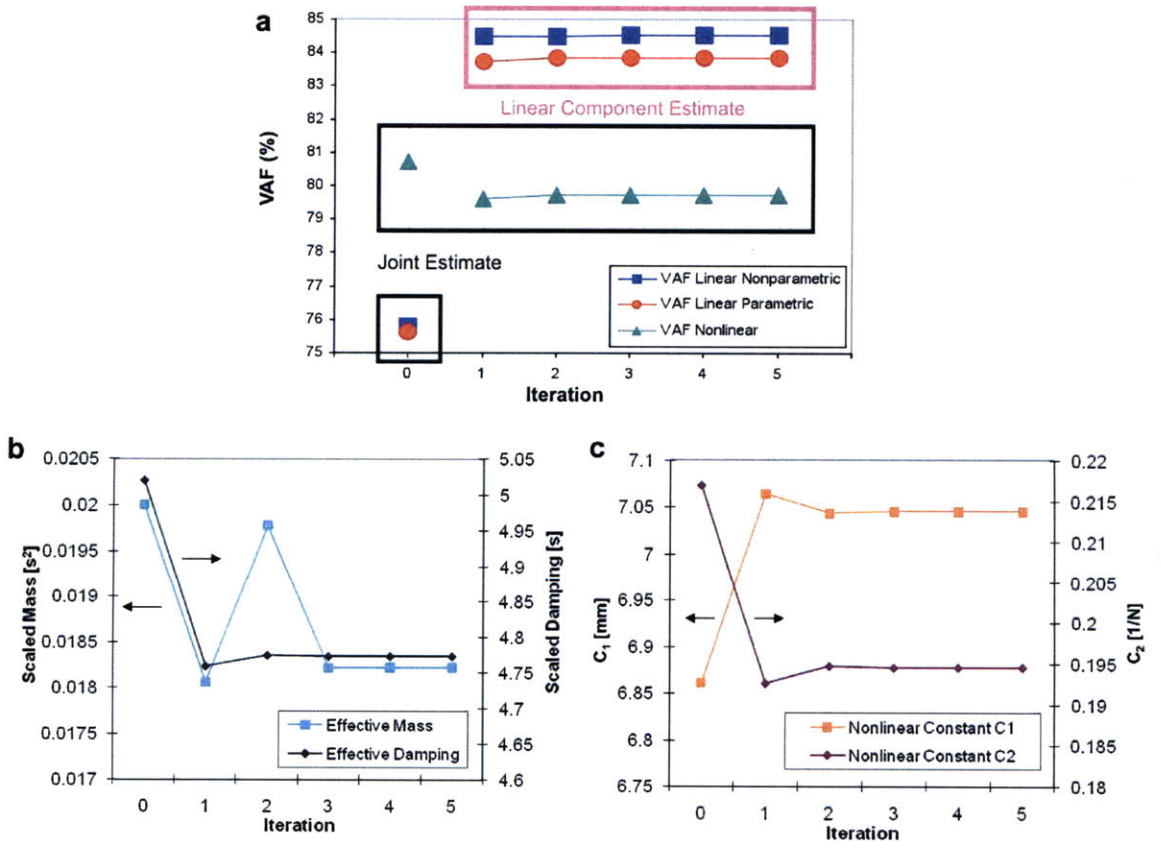


Figure 3.9: (a) Variance accounted for, (b) scaled linear parameters, (c) and nonlinear parameters as a function of the number of iterations.

For the first iteration, the linear estimate is between the input and the measurement. The nonlinear function then serves as a correction to the linear estimate. In

subsequent iterations, the linear system estimate is separated hence the significant increase in the VAF. The nonlinear VAF of the nonlinear estimate decreases slightly afterwards. This is because additional estimates are attempting to optimize two separate estimates at the same time rather than two estimates sequentially. The VAF from the joint estimates are highlighted in black and the single estimates of the linear component are highlighted in pink. The nonlinear VAF is calculated in conjunction with the nonparametric linear model.

Figure 3.9b shows how the linear parameters, the scaled mass in units of $[s^2]$ and the scaled damping in units of $[s]$ vary with the number of iterations. Figure 3.9c shows how the nonlinear parameters vary. The reason for the strange units is because the scaling constant was chosen with units of N/m.

As the number of iterations increases, the initial change caused by the scaling constant drives the majority of the shift. In additional iterations, the parameters converge and the absolute error decreases. The scaled mass oscillates more than the other parameters because it is being modified by changes in ζ and ω_n which both force the scaled mass in the same direction such that the effective deviation increases in the second iteration. In addition, since the Wiener nonlinearity is only an approximation of the real system, the error in the parameter estimates may not initially evolve in a monotonically decreasing fashion.

An illustration of how the nonlinear estimates compare to the output is shown in Figure 3.10. The blue dots represent measured values while the red line represents the model estimates after identification. Since the input to this system is Gaussian and the output is not Gaussian, the system response is obviously nonlinear. The nonlinear model seems to be unable to repeat the peaks and troughs of the real system exactly but does well for estimates near the mean value.

The variance accounted for is 75.8 % for the linear model and 80.7 % with the addition of the static nonlinearity. Note that the increase of about 5 % to 10 % in the VAF is typical for the performance of the static nonlinearity for indentation into the skin. The estimates generated by this procedure are repeatable and able to distinguish between different locations on the body. The mean and standard deviation

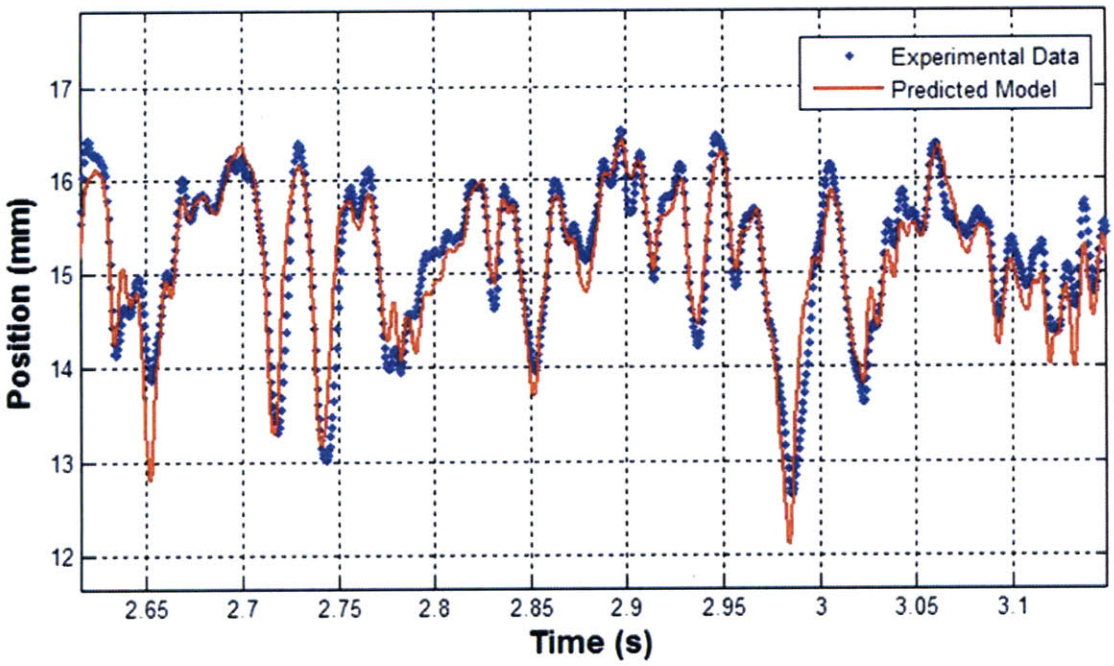


Figure 3.10: The identified system using a Wiener static nonlinearity (red) matches well with the measured results (blue). The surface of the skin is located at 10 mm on the actuator for this test.

of the VAF, linear parameter estimates and nonlinear parameter estimates for 10 tests on each of three different regions of the skin for one test subject are shown in Table 3.1. The anterior forearm position, posterior forearm and distal forearm positions are 40 mm from the wrist of the left arm. From palpation, it is clear that the posterior position is stiffest followed by the anterior position and the distal side. The distal side has the least amount of tissue between the surface and the bone so it is expected that the system will have a sharper transition in the nonlinearity.

The results in Table 3.1 demonstrate that the areas tested are readily distinguishable and demonstrate good repeatability. By looking the the nonparametric VAF versus the nonlinear VAF, it is clear that the VAF increases once a static nonlinear model is used. As expected, the posterior forearm gives the most linear response among the three positions. The stiffest position is the posterior position followed by the anterior position and the distal side. This can be assessed by looking at the relative values of C_1. During palpation, this difference in compressible depth can be sensed qualitatively. The damping, on the other hand, is much harder to sense by

Table 3.1: Device and identification method repeatability

Quantity	Anterior Forearm	Posterior Forearm	Distal Side of Forearm
Linear Nonparametric VAF (%)	78.06±2.33	86.53±0.72	76.05±1.65
Linear Parametric VAF (%)	78.11±2.35	86.04±0.69	76.00±1.59
Nonlinear VAF (%)	82.60±1.96	89.14±0.58	81.33±1.62
Scaled Mass (s^2)	0.0194±0.001	0.0259±0.0004	0.0235±0.0012
Scaled Damping (s)	4.84±0.166	5.22±0.0843	7.00±0.222
Nonlinear Constant C_1 (mm)	7.42±0.302	9.77±0.145	4.71±0.360
Nonlinear Constant C_2 (1/N)	0.191±0.0153	0.154±0.0162	0.442±0.0327

palpation. From the data, the damping and mass can be calculated by multiplying the scaled damping and scaled mass by the value of the stiffness calculated at any given position from the derivative of Equation 3.26,

$$K_e(z) = \frac{1}{C_1 C_2} e^{(C_2 z)}, \tag{3.27}$$

$$M_e(z) = M_s K_e(z), \tag{3.28}$$

$$B_e(z) = B_s K_e(z). \tag{3.29}$$

From this transformation, it is clear that the mass, damping, and spring constant are constrained to vary with the same nonlinear profile when the Wiener static nonlinearity model is used. The estimates generated by these parameters are a mean value for any given position. As is clear from the nonlinearity in Figure 3.8, the error in location can be as much as 1 mm for a total displacement of 7 mm. This is not ideal and a better model should be obtained for tissue dynamics. The static nonlinearity, however does to a good job of predicting position based dynamics to first order.

For skin mechanics, it is imaginable that creep, hysteresis and directional (pushing into the skin is different from pulling on the skin) effects also play a role. In addition, Wiener and Hammerstein static nonlinearities are a particular form of static nonlinearity that transforms physical parameters (such as mass, damping and spring constant) by the same nonlinear factor. It can be imagined that for a system with a large actuator mass that the same nonlinear transformation does not happen to all

the physical parameters in reality. These issues can be addressed with more advanced nonlinear system identification structures.

3.4 Localized Linear Testing

The input range can play a large part in the identification of a nonlinear system. Small input ranges can often be as useful as large input ranges. This concept is best illustrated with a few time-domain mathematical models. For a linear system, the equation of motion is,

$$M_e\ddot{y}(t) + B_e\dot{y}(t) + K_e y(t) = x(t), \tag{3.30}$$

where the input $x(t)$ is a force and the output $y(t)$ is a position. The Wiener static nonlinearity can be written in the time-domain as a nonlinearity which does not affect the dynamic equation,

$$M_e\ddot{z}(t) + B_e\dot{z}(t) + K_e z(t) = x(t), \tag{3.31}$$

$$y(t) = g(z(t)), \tag{3.32}$$

where $z(t)$ is an inaccessible linear dynamic output variable and $y(t)$ is the output position of the system. Only one set of dynamics occurs in this system and is completely encapsulated by Equation 3.31.

Coupling can occur when a dynamic parameter nonlinearity (DPN) is used. This type of model can be written as,

$$M_y(y(t))\ddot{y}(t) + B_y(y(t))\dot{y}(t) + K_y(y(t))y(t) = x(t), \tag{3.33}$$

where M_y, B_y, and K_y are all different functions which vary with position y(t). In this time-domain model, the nonlinearity is clearly a function of depth and cannot be completely decoupled from the dynamics. This type of equation of motion cannot be solved with conventional linear system identification methods since it can have

arbitrarily many sets of dynamics which vary with the output variable $y(t)$. In control theory, the system dynamics of a nonlinear system can often be simplified if the perturbations to the system are small. This is known as linearization about a point. The expansion about this point can be completed using a perturbation theory expansion to first order around the location of interest y_0 for small variations in $y(t)$. With small perturbations about a mean input force, the coupling terms can be linearized,

$$M_y(y_0)\ddot{y}(t) + B_y(y_0)\dot{y}(t) + K_y(y_0)y(t) = x(t). \tag{3.34}$$

When this function is linearized for small inputs, the equation of motion looks like Equation 3.30 and can therefore be solved with linear methods. When larger inputs to this system are used, additional interactions occur between the nonlinearity and the dynamics. This can be interpreted as additional dynamic cross coupling terms added to the left hand side of Equation 3.34. This type of nonlinearity can produce the looping behavior as observed in Figure 3.8. The time-domain models listed in this section can be easily discretized and simulated (See Appendix C.1.4) as shown in Figure 3.11.

For linear system, the dynamics do not change as a function of depth so the red and black input ranges give the same information as shown in Figure 3.11a. For a Wiener static nonlinearity, a smaller test range produces linear results while a larger input range helps resolve the static nonlinearity more clearly. The results from separate small input range tests are consistent with the large range input because they both produce a one-to-one relationship for y and z in Figure 3.11a. Another interesting point to note is that the Wiener static nonlinearity produces the same position dependent effect on the mass, damping, and spring constant as shown in Figure 3.11b.

The DPN model produces completely different results for different input ranges. If a large input range (100 %) is used, simply removing the linear dynamics will produce looping behavior and produce features that can be mistaken for large estimation error. If smaller input ranges (5 %) is used, then a one-to-one relationship appears

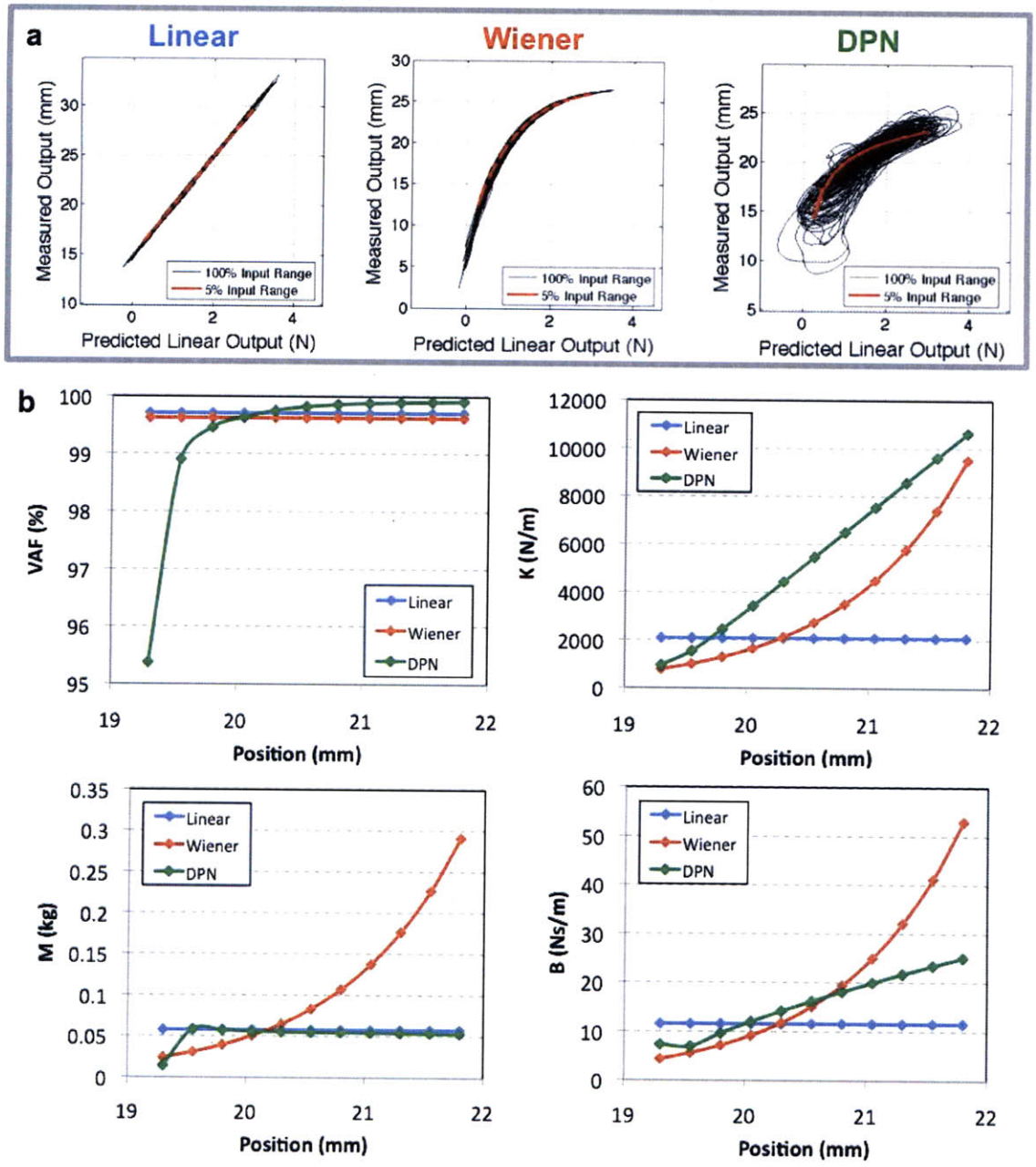


Figure 3.11: A linear dynamic, Wiener static nonlinearity, and a dynamic parameter nonlinearity (DPN) are simulated. (a) Large and small input ranges are tested and the static nonlinearity plot is shown. (b) These graphs show how the parameter values for mass, damping, and spring constant can vary with position when using different types of nonlinearities.

between parameter values and position. With this type of nonlinearity, it is possible to specify completely different nonlinear functions for the mass, damping, and spring constant. With this concept in mind, large range tests as well as small localized linear

tests are conducted on skin during indentation to obtain more information about the underlying dynamic parameter nonlinearities.

Results from these localized linear tests are shown in Figure 3.12. The shaded region in the figure shows the region that was pushed into the skin and the white region corresponds to pulling on the skin. As the probe goes through different positions on the skin, the impulse response changes. The changes in the parameter values for pushing into the skin happen slower than the changes in the parameter values for pulling on the skin. As discussed earlier, pushing into the skin will generate a different response than pulling due to geometric differences.

For a medium range of forces that vary by about 7 N, very little looping behavior is observed in the static nonlinearity. For a larger range of forces around 11 N, the looping behavior begins to appear. On the other end of the spectrum, if the inputs are small with inputs that vary about 1 N each, the looping behavior diminishes even more and each of the smaller inputs begins to align. Each of the 1 N variation groups represents the estimation from a single test. Even with variation of 1 N, it is possible to see nonlinear behavior. As the input force variation decreases to zero, the individual tests will begin to align perfectly to follow a single line.

A separate series of tests was conducted on different locations on the forearm and the impulse responses were fitted with parameter values as shown in Figure 3.13. Note that the measured values for the skin are much more nonlinear than the calibration curve. The two positions on the skin were the posterior forearm 40 mm from the wrist and anterior forearm 40 mm from the elbow.

The spring constant increases quickly as the depth into the skin increases. The increase in the spring constant as the probe is pulled off the skin is even greater. The damping parameter also appears to increase as the probe goes deeper into the skin. As expected the mass of the system does not change appreciably. The frequency plots are also shown. As the probe moves through different positions, the natural frequency and damping parameter change. The DC value also changes because the DC value indicates the average stiffness for a certain position.

The VAF of each individual test is generally better than 90 %. While this may

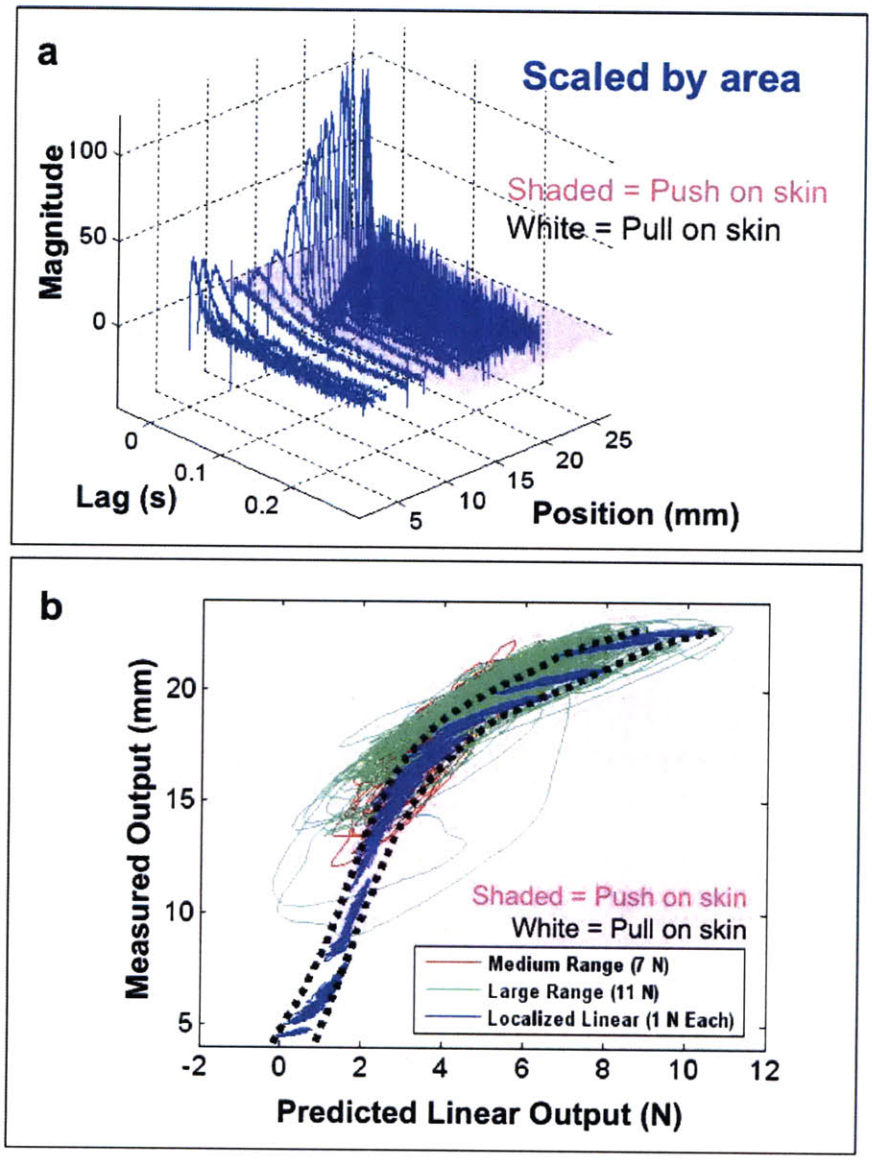


Figure 3.12: The evolution of system dynamics as a function of depth can be explained with the static nonlinearity (a) when plotted as a function of absolute position. Note that the impulse responses are scaled as a function of the underlying area. (b) The static nonlinearity as a function of position for tests with input force variation of 11 N, 7 N and 1 N. Note that the black dotted lines are to serve as a guide to the eye.

appear to be a good method for identifying skin properties, there are few drawbacks. First, testing each of the positions take 4 seconds (which can be reduced to about 0.5 seconds) plus additional setup time. To obtain better linear information at each depth would require even more tests with smaller force variations. That would reduce the input variation which would unfortunately make the input closer to the noise floor.

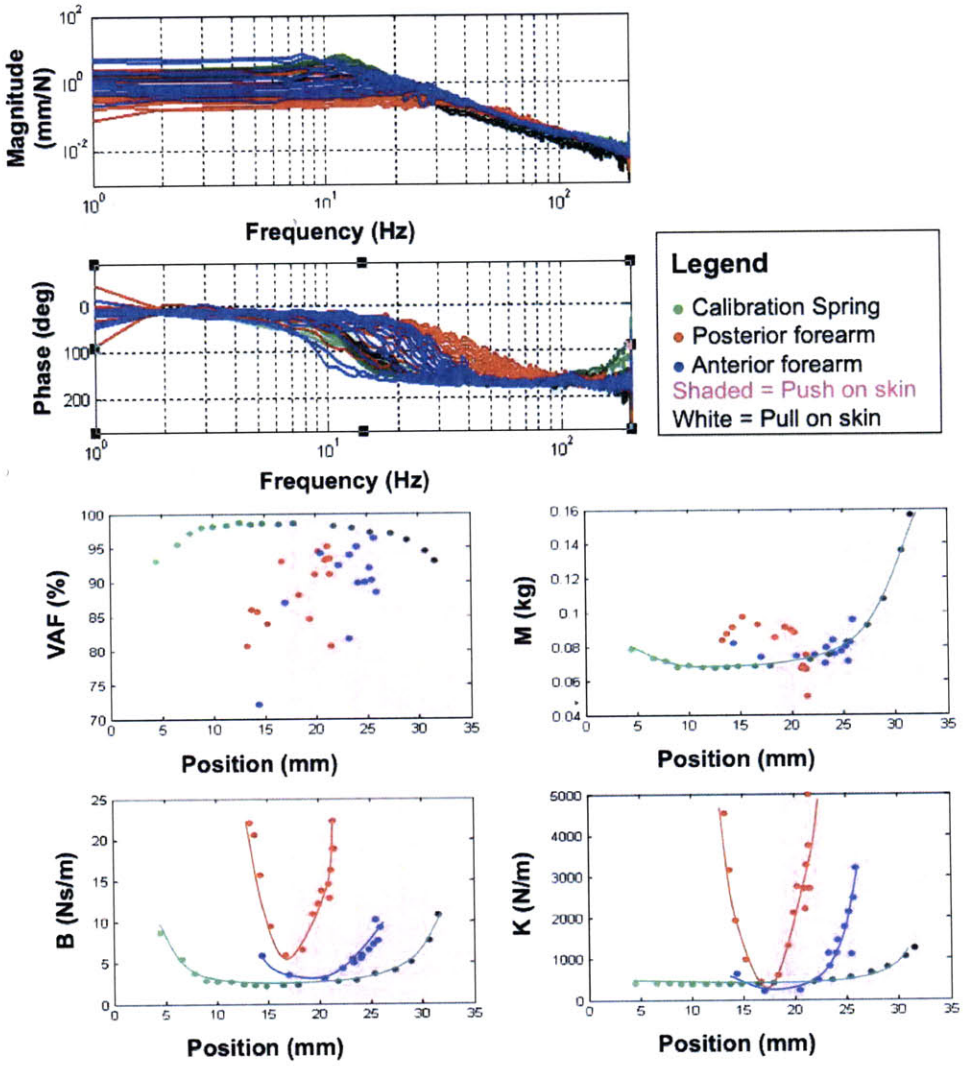


Figure 3.13: Localized linear tests for the posterior forearm 40 mm from the wrist and anterior forearm 40 mm from the elbow mapped against a calibration curve. The Bode plots and parameter values are shown.

Second, having to precisely position each step of the test would make it difficult to reject level changes in the noise associated with patient motion. Lastly, testing each position separately makes it impossible to looking at the nonlinear coupling terms that occur when the probe moves between different positions. These nonlinear coupling terms can be clinically important. For example, other types of nonlinearities such as hysteresis associated with fluid movement and damping as a function of input frequency and change in depth cannot be measured with this method. More advanced methods that can address each of these points are discussed in the following chapters.

Chapter 4

Volterra Kernel Techniques

One of the drawbacks of the Wiener and Hammerstein forms of static nonlinearity is that they impose constraints on the form of the nonlinearity which do not completely match the expected form of the nonlinearity in tissue dynamics. A Wiener static nonlinearity imposes the same nonlinear transform to all the parameters which can be seen in Equations 3.27 to 3.29. A simple cascade representation can be generalized using kernel based system identification techniques. Conceptually, the linear and nonlinear blocks in Figure 3.6 can be grouped into a single block with both linear and nonlinear characteristics. With this type of representation, of which Wiener and Hammerstein nonlinearities are members, it is possible to look at some dynamic nonlinearities. Note that cascade representations can be also grouped in parallel [67]. In this chapter, instead of using iterative techniques to identify these static nonlinearities, a comprehensive technique for determining the Volterra kernels themselves is presented.

The Volterra series is a functional expansion of the general time-invariant nonlinear dynamic system problem. The idea behind the functional expansion is that the zeroth order kernel represents the system average. The first order kernel represents the first order linear perturbation to the system where the output depends linearly on lagged inputs. This kernel is exactly the linear impulse response function. The second order kernel represents the second order perturbation to the system where the "impulse response" function is not a function of one lag but a function of two lags.

This means that the input at some time can interact with the input at another time to produce an effect on the output. This concept can be expanded to higher orders [68]. For a system with a finite memory length I, the discrete functional expansion can be written,

$$y^*(n) = h_0 + \sum_{i=1}^{I+1} h_1(i)x(n-i) + \sum_{i_1=1}^{I+1}\sum_{i_2=1}^{I+1} h_2(i_1,i_2)x(n-i_1)x(n-i_2) + \ldots \qquad (4.1)$$

As it stands, the Volterra expansion is difficult to solve; one of the reasons is that the expansion contains many parameters in h_1, h_2, etc. which grow very quickly with the memory length and kernel order. Secondly, the system is not orthogonal so changing one value will change the optimal fit for other values in the series.

The Volterra kernel is, however, only one functional expansion among many for nonlinear dynamic systems. A modification to the Volterra kernel developed by Norbert Wiener attempts to make solving the system much simpler. The Wiener functional expansion orthogonalizes the Volterra series for an assumed form on the input. By using assumptions for Gaussian white inputs, Wiener was able to create a different expansion such that the first kernel can be solved independent of the second kernel. This means that any noise remaining after solving the first order kernel must either be noise or components of higher order kernels. It is important to note that the Wiener and Volterra kernel solutions are not exactly the same. The zeroth order Wiener kernels are the mean output for one type of Gaussian white input. The first and second kernels, however are the same for the two systems as long as there are no higher order kernels [64].

Several Wiener kernel solution techniques exist including cross-correlation methods [71], repeated Toeplitz matrix inversion techniques, and use of functional expansions. The drawback of the Wiener functional expansion is that only white inputs can be used. Since real inputs can only become white asymptotically, there is inherent uncertainty in the solutions for short test lengths. In addition, as detailed in Section 3.1.1, the input to a real system is rarely optimally Gaussian and white. It is possible to create orthogonal expansions for different types of inputs but not every mathe-

matical function has properties that would allow this to be readily accomplished. In addition, it becomes cumbersome to do system identification if a new expansion needs to be derived for every new input.

Is there, then, a functional expansion that is best to use for biological systems? Based on the functional expansion alone, the Volterra kernel is the most easily interpretable since most other expansions derive from it. The nonlinearity is in terms of high order kernels which are taken with respect to lagged versions of the inputs. Other expansions that orthogonalize the Volterra kernels are less physically interpretable although arguments can be made that they are mathematically more general [64].

There are several different methods that can be used to solve Volterra kernels and some of them are listed below.

- One method of solving for kernels is to use a series of impulses into the system [91]. The first pulse generates the response to the impulse. A second pulse lagged by some time generates another response but this time, additional terms that interact with the first pulse appear (terms of the second Volterra kernel) and can be mathematically separated from the first pulse. For even higher order terms, more pulses can be used. Of the many drawbacks to this method [68], the two most obvious are the restriction on the input type and the difficulty of generating perfect impulses.
- Other techniques that use cross correlation, recursive least-squares or stochastic approximation techniques also exist [36].
- Another method to produce fitting using fewer parameters than the memory length is to use another functional expansion that is fit onto the kernels. Laguerre expansions [77] and Meixner functions [3] can be used to do this. One drawback is that using another expansion to shorten the number of parameters needed imposes smoothing to the kernels which is not always desirable since the real kernel may not be smooth. Other expansions with other expansions and polynomials can be used and several methods of fitting can also be described [68].
- It is also possible to orthogonalize the input itself. This technique developed

by Korenberg and Hunter can be generalized to any input [68]. It imposes no constraint on the input type (input does not need to be Gaussian and white to be solved), length, or smoothing constraints used on the kernels. One possible drawback is the computation time which does place a constraint on the sampling frequency by discouraging oversampling of the data.

Because of the benefits of the Korenberg and Hunter method [68], it is used as the basis for this chapter. The implications of this technique are discussed along with results for skin under indentation. Because this method requires a few modifications for the input types used in this work, additional implications and methods for obtaining interpretable kernel data are discussed.

4.1 Exact Orthonormalization Solver

The method by Korenberg and Hunter requires an exact orthonormalization step for the input, a solution step in the orthonormalized space, and a reconstruction step to take the solution back into the space of the Volterra kernel. A summary of the solutions steps are shown below:

1. Construct: Sort the input data according to a set of rules.
2. Orthonormalization: Use a modified Gram-Schmidt solver to orthonormalize the input data.
3. Solve: Obtain a orthonormalized solution.
4. Resolve: Use the inverse of the Gram-Schmidt process to put the solution back into its original terms.
5. Reconstruct: Use the same partitioning rules to resolve the kernel responses.

Steps 2 through 4 for the orthonormalization technique are well known [99]. In fact, Gram-Schmit orthogonalization is the exact step used by Wiener to orthogonalize the Wiener kernels. Steps 1 and 5 are dependent on different partioning rules which need to satisfy constraints from the modified Gram-Schmidt orthonormalization process. Figure 4.1 shows a schematic of the algorithm used to solve Volterra kernels. A

modified version of the technique used by Korenberg and Hunter was developed. This process is optimized for computational speed in MATLAB and for numerical stability.

$$y\ (n) = h_0 + \sum_{i=1}^{I+1} h_1(i)x(n-i) + \sum_{i_1=1}^{I+1}\sum_{i_2=1}^{I+1} h_2(i_1,i_2)x(n-i_1)x(n-i_2) + \ldots + e(n)$$

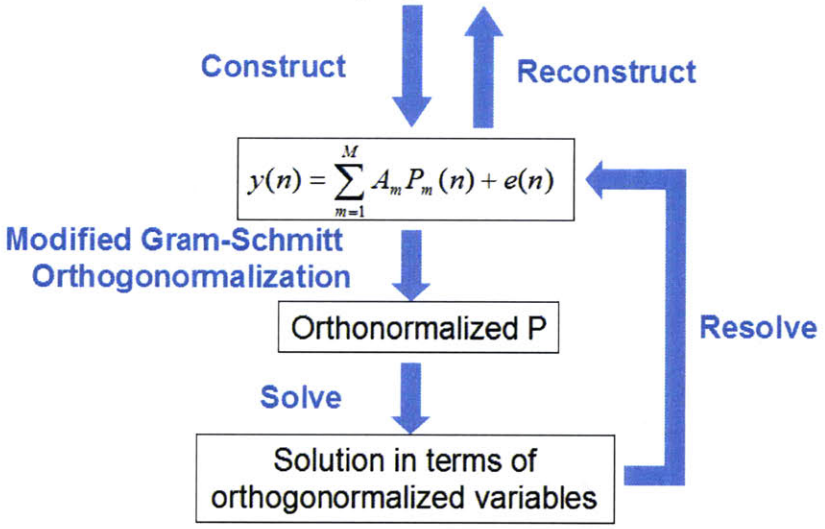


Figure 4.1: The algorithm used to solve for Volterra kernels involves constructing the kernel, orthogonalization, solving, resolving, and reconstructing.

Many physically realizable, finite memory systems can be modeled from an input output relation that is shown,

$$y(n) = \sum_{m=1}^{M} A_m P_m(n) + e(n). \tag{4.2}$$

In the simplest linear case, $y(n)$ is the output of the single input single output system, A_m is the impulse response of the system with memory M and $P_m(n)$ (which is not position in this case) is simply equal to the input $x(n-m-1)$. The measurement contains some error $e(n)$. From this form, one can easily obtain a linear input output relation. In the more general case, m is a value which stores the dynamic memory of the system which stores input lag information while n represents the value at a given time. The $P_m(n)$, however stores information for a particular set of rules that apply at a given m and n. This implies that A_m is a series that is convolved with $P_m(n)$ to

produce the desired output where $P_m(n)$ is constructed based on some partitioning rule. It is typically difficult to directly solve this equation and therefore needs to be orthonormalized into a different form in terms of variables γ_m and β_m,

$$y(n) = \sum_{m=1}^{M} \gamma_m \beta_m(n) + e(n). \tag{4.3}$$

4.1.1 Construction

To solve the Volterra kernel, the input and output data must first be constructed into a usable form using the following relationship for the first two kernels plus the level offset. Third Volterra kernel and higher order kernels can be derived using the same methodology. The function listed below is the construction rule which can be changed for different representations. For example, this function can be replaced with a rule for a different basis function representation.

$$P_m(n) = \begin{cases} 1 & \text{for } m = 1 \text{ and } n = I+1 \ldots N \\ x(n-m+2) & \text{for } m = 2 \ldots I+2 \text{ and } n = I+1 \ldots N \\ x(n-i+1)x(n-j+1) & \text{for } m = I+2 \ldots 1+(I+1)+(I+1)(I+2)/2, \\ & \quad n = I+1 \ldots N, \\ & \quad i = 1 \ldots I+1, \\ & \quad \text{and } j = i \ldots I+1 \end{cases} \tag{4.4}$$

Immediately, one notices the size, or number of parameters needed to obtain a second order Volterra kernel for a given memory length I. The value of M represents the total number of parameters that need to be fit up to an arbitrary size kernel. The following equation shows how M evolves with the fitting of the zeroth kernel (first term), first kernel (second term) and second kernel (third term),

$$M = 1 + (I+1) + (I+1)(I+2)/2 + \ldots \tag{4.5}$$

The amount of time it takes to complete the algorithm is related to the value of M and the test length N. Therefore, decreasing the number of parameters in the fit and the test length will significantly decrease the time for the computation.

One of the important rules for partitioning the function above is that the values stored in $P_m(n)$ must be orthogonal (one row cannot be combined with another row to produce the values of a third row) or basically non-repeating. This can be a problem for the Volterra kernel which is symmetric for the second kernel. The solution to this problem can be achieved by simply grouping the symmetric portions of the kernel into a single value which can then be solved and later separated in the reconstruction step. When separated again, half of the value can be given to one side of the second order kernel and the other half to the other side of the second order kernel. This method can be used to deal with any repeated values in basis functions.

4.1.2 Orthonormalization, Solution, and Resolution

The orthonormalization process produces $\beta_m(n)$ which is orthogonal for a section of the data record from $I+1$ (where the physical memory length is I/F_s seconds long were F_s is the sampling frequency) to the end of the data record N. Values before $I+1$ do not need to be orthonormalized because at point $I+1$, the system "remembers" all the inputs from the beginning to point $I+1$. Because the information before the start of the data record is unknown, at point I, there is unaccounted for information.

The orthonormalization procedure is based on an iterative modified Gram-Schmidt process which tends to be more numerically stable for discrete systems. The process involves selecting one vector to be the first orthogonal vector $P_m^{(1)}(n)$. Part of the vector along the m dimension that is orthogonal to this is removed to form the next orthogonal vector $P_m^{(2)}(n)$ which is now orthogonal to $P_m^{(1)}(n)$. This process is repeated a total of $R = M$ times to obtain R vectors that are all orthogonal to each other. During orthonormalization, the values of $\beta_m(n)$ and $\gamma_m(n)$ are obtained. Initially the first orthogonal vector is set,

$$P_m^{(1)}(m) = P_m(n). \tag{4.6}$$

for $r = 1$ to R, Equations 4.7 through 4.10 are computed during each iteration,

$$\beta_r(n) = P_r^{(r)}(n) \qquad \text{for } n = I+1 \ldots N. \tag{4.7}$$

The value of α_{mr}, which is used to weight the orthonormalization process and map the result back into the original terms after solving, can be obtained from the following,

$$\chi_r(n) = \sum_{m=I+1}^{N} \beta_r^2(n) + \epsilon \tag{4.8}$$

and

$$\alpha_{mr} = \frac{\overline{P_m^{(r)}(n)\beta_r(n)}}{\overline{\beta_r^2(n)}} = \frac{\sum_{n=I+1}^{N} P_m^{(r)}(n)\beta_r(n)}{\chi_r(n)} \qquad \text{for } m = r+1 \ldots M. \tag{4.9}$$

The overbar represents the average of the series but since the length of the series on the bottom and the top are the same, computation time can be saved simply by computing the sum. The term on the bottom $\chi_r(n)$ can be calculated and stored for each iteration for additional computational speed. For numerical stability, a small value ϵ can be added if the value of β_r^2 happens to be smaller than a certain value. This is to prevent the value of α_{mr} from exploding.

In order to orthonormalize the next vector, Equation 4.10 is used following a modified Gram-Schmidt process,

$$P_m^{(r+1)}(n) = P_m^{(r)}(n) - \alpha_{mr} P_r^{(r)}(n) \qquad \text{for } m = r \ldots M. \tag{4.10}$$

Up to this point, the equations have worked exclusively with the input with no consideration given to the output. The most time consuming step is the orthogonalization step and in order to save time for future computations, in cases where the same input is used, the above values can be stored so they do not need to be recalculated. Results obtained with this input can be directly solved using the rest of the procedure thereby increasing the computational speed by 1-2 orders of magnitude (see Figure 4.2). This is an additional benefit of this method.

After the matrix has been orthonormalized, the system identification can be completed, or fit using least squares, simply with Equation 4.11,

$$\gamma_m = \frac{\overline{y(n)\beta_m(n)}}{\overline{\beta_m^2(n)}} = \frac{\sum_{n=I+1}^{N} y(n)\beta_m(n)}{\chi_m(n)} \qquad \text{for } m = 1 \ldots M. \tag{4.11}$$

When these values have been determined, the next step is to convert this back into the original terms with Equations 4.12 and 4.13. This is the resolution step,

$$\text{for } m = 1 \ldots M$$

$$\nu_i = \begin{cases} 1 & \text{for } i = m \\ -\sum_{r=m}^{i-1} \alpha_{ir}\nu_r & \text{for } i = m+1 \ldots M \end{cases} \tag{4.12}$$

and

$$A_m = \sum_{i=m}^{M} \gamma_i \nu_i. \tag{4.13}$$

These equations form a basic orthonormalization procedure which can be used to perform a least squares fit to a single input single output system. Generalizations can also be made for other multi-input multi-output systems using the same concepts.

4.1.3 Reconstruction

In the next step, the system needs to be reconstructed using the following relationship valid for the level offset, the first kernel, and the second kernel. Other kernels can be similarly derived,

$$\begin{cases} h_0 = A(m) & \text{for } m = 1 \\ h_1(i) = A(m) & \text{for } i = 1 \ldots I+1 \text{ and } m = 1+i \\ h_2(j,k) = C_v A(m) & \text{for } j = 1 \ldots I+1 \text{ and } k = j \ldots I+1 \\ & \quad \text{where } m = I+2+(k-j+1)+\sum_{p=2}^{j}(I+1-p). \end{cases} \tag{4.14}$$

The value of C_v depends on its location in the kernel. If $j = k$, then $C_v = 1$ and

if $j \neq k$, then $C_v = \frac{1}{2}$. As mentioned earlier, the second Volterra kernel is defined symmetrically. The orthogonalization process cannot handle redundant parameter values. Therefore, the degenerate or redundant terms that occur when $j \neq k$ hold twice their actual value in orthonormal space.

4.1.4 Performance

This algorithm has several performance features. Unlike what is indicated in Korenberg and Hunter, this implementation scales quadratically with the number of parameters M and not with the memory length of the system. This may be due to matrix memory allocation optimizations. Performance gains are most likely due to the elimination of computations for matrix elements that are not used in future calculations and the storage of repetitive calculations. This leads to performance gains for larger numbers of parameters and larger memory lengths. The lower bound computational cost on a Gram-Schmidt [99] process is asymptotically $O(NM^2)$ which is approximately how this particular algorithm scales as shown in Figure 4.2.

The computational costs are mainly in the orthonormalization process while the solution and reconstruction steps take one to two orders of magnitude less time. It is important to note that for a given input, orthonormalization only needs to occur once and the values for the matrices α, β and χ can be stored. It is also important to note that once A_m is obtained using a certain set of partitioning rules, any new input that is partitioned in a similar fashion can be convolved with A_m to produce a simulated output. Figure 4.2 shows the performance of this algorithm as a function of the number of parameters.

In addition to performance characteristics, there are also mathematical limitations to this type of solver. This algorithm may produce oscillatory estimates for inputs that have low power at high frequencies because low-pass filtered input data tends to have consecutive values that are very similar to one another. This causes the orthonormalization process to appear less stable or look as though it absorbs noise (since there is no constraint on smoothness). This tends to cause minor problems for partitioning schemes that are continuous (such as for identifying Volterra kernels)

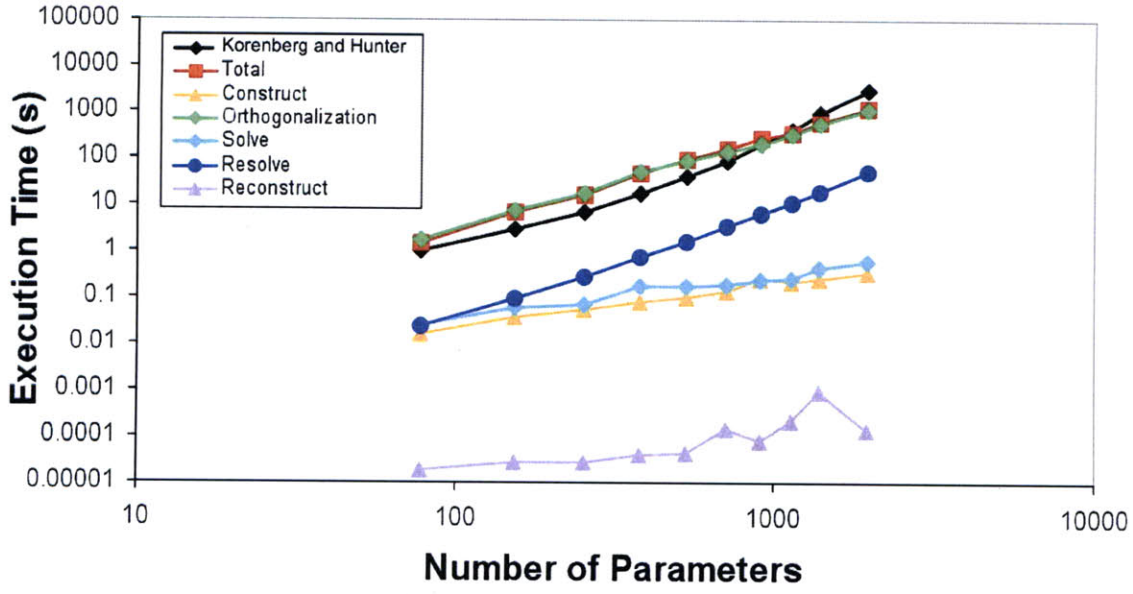


Figure 4.2: Computational times of orthonormalization procedure in MATLAB using a 2.4 GHz Intel Core 2 Duo CPU and 2 GB of RAM. The process scales as the square of the size of the number of parameters M. Orthonormalization takes approximately one to two orders of magnitude more time than the solution or reconstruction steps.

and has a smaller influence on noncontinuous partitions.

Because the total number of parameters in the fit M scales with the memory length I as detailed in Equation 4.5, it is important to choose the proper sampling time so that the memory length is short enough to compute the result in a reasonable time but long enough that the information contained in the system is expressed. This seems to put a constraint on the input but what this indicates in reality is that superfluous sampling is not only unnecessary but detrimental.

4.2 Simulated Results

Before detailing experimental results, it is a good idea to look at the capabilities of the Volterra kernels and attempt to interpret the trends. Volterra kernels can be used to represent both Wiener and Hammerstein systems and can handle both monotonic and non-monotonic nonlinearities. Each of these types of systems produce different characteristic results. By looking at plots of the impulse response, residual nonlinearity, first and second kernels, it is possible to identify these differences.

The definition of the impulse response for a second order Volterra kernel is listed in Equation 4.15. Note that for a second order Volterra kernel, the impulse response does not have the same generality as it does for a linear system. It is simply the response to an impulse input such that,

$$h_{total}(t) = h_0 + h_1(t) + h_2(t,t). \tag{4.15}$$

In the above equation, the first term is the contribution from the zeroth kernel, the second term is the contribution from the first kernel and the last term is the contribution from the second Volterra kernel. Note that for the second kernel, the contribution to the impulse response is the diagonal term. The full response of the system is represented by all the kernels used in the fit.

Figure 4.3 shows the simulated Volterra kernels for a monotonic exponential nonlinearity in the (a) Wiener configuration and (b) Hammerstein configuration. The impulse responses look very much like linear impulse responses. The true impulse response used in the simulation is shown in blue. The contribution of the first kernel is shown to be large and positive for an exponential (monotonic) input. For the contribution of the second order kernel, the diagonal is negative so that the total impulse response is shown in red. This has a pattern similar to a linear impulse response, but is in fact an exponential function of the true impulse response.

This is more clear in the residual nonlinearity plot in Figure 4.3a. For each contribution, the input is convolved with each contributing component. The estimate is then plotted against the measured output. For the contribution of the first kernel the residual looks like the original input nonlinearity. With the second order correction, the residual looks even more like the exponential. This makes it exactly like a Wiener static nonlinearity. As more of the off diagonal terms of the second order Volterra kernel are added, the exponential function begins to disappear and the residual goes to zero in the limit. A perfect fit would result in a residual of zero which is an almost linear curve. This means that the fitting function did a relatively good job of accounting for the nonlinearity. Note that there is an offset of 20 to the right which

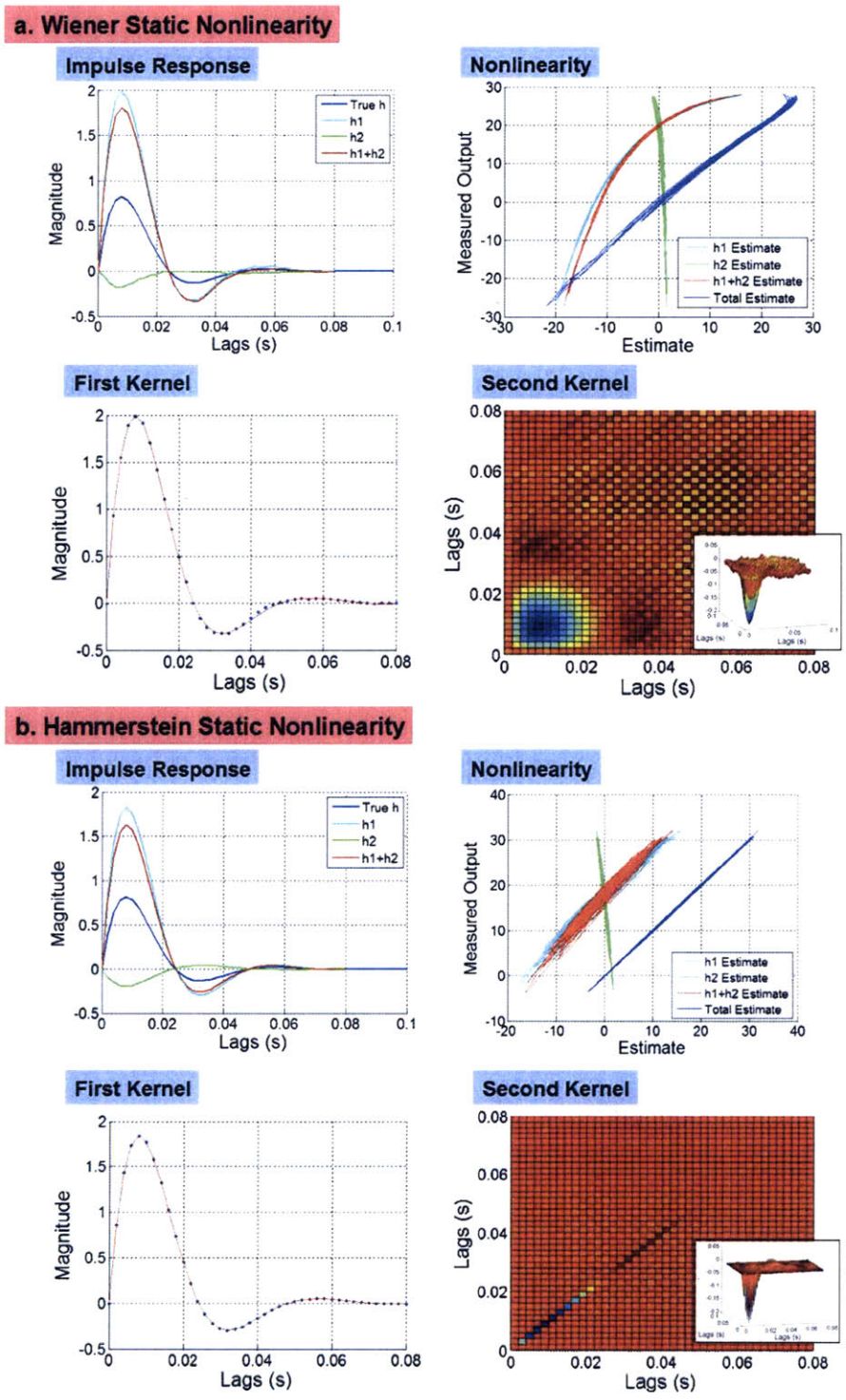


Figure 4.3: Volterra kernels for a simulated system showing the impulse response, residual nonlinearity, first order kernel and the second order kernel. (a) The functional form of the Wiener static nonlinearity used in this function is the same as Equation 3.26. (b) This same nonlinearity is used in the Hammerstein configuration.

comes from the zeroth order kernel.

For high values of the estimate near 30, there is an additional lip. This indicates that a better fit can be obtained with a slightly higher order kernel. The Volterra kernel can be thought of as a dynamic polynomial fit. With a first order kernel, it is possible only to fit linear functions so the only part of the residual that can be removed is a linear component. With a second Volterra kernel, it is possible to fit up to a second order nonlinearity (quadratic). With a third Volterra kernel, it is possible to fit up to a third order nonlinearity (cubic) and so on. Since an exponential can be only partially fit by a second order nonlinearity, some of the residual still remains.

The first kernel is similar to the impulse response in Figure 4.3. In the second kernel, each small block represents a single sample which means that the kernel can either be represented in terms of seconds or in terms of samples. The second Volterra kernel is negative but otherwise has the same shape as the first kernel except swept in two dimensions. The second Volterra kernel is symmetric because the input lag of time t_1 versus t_2 is no different from t_2 versus t_1. The shape itself is important; a slightly different shape would imply a different type of nonlinearity.

The impulse responses in the Hammerstein static nonlinearity in Figure 4.3b look the same as for the Wiener system. However, the residual nonlinearity is different as the Hammerstein nonlinearity does a better job of reducing the unknown nonlinearity so that the residuals look linear. With the addition of extra terms in the second kernel, the fit becomes perfect so that there is no residual left. The Volterra kernels do a better job of fitting the Hammerstein systems in general because Hammerstein systems and Volterra representations both impose nonlinearities based on the input.

The first kernel looks as expected while the second order kernel only exists along the diagonal; consistent with the nonlinearity existing only in the input side of the dynamics so that there is no relationship between t_1 and t_2 other than when $t_1 = t_2$. When it is known that the system is a Hammerstein nonlinearity, the lags other than the ones on the diagonal appear to be unnecessary. When looking at the residual plot however, it is clear that even the off diagonal terms are necessary to reduce the residual nonlinearity. From these arguments, it is clear that a second order Volterra

kernel is able to fit both monotonic Wiener and Hammerstein nonlinearities.

Figure 4.4 shows the simulated Volterra kernels for a non-monotonic quadratic nonlinearity in the Wiener configuration and Hammerstein configuration. By looking first at the Wiener static nonlinearity simulation, it is clear that the impulse response estimates from different contributions do not look like the true impulse response. Rather, the contribution looks like a quadratic function of the true impulse response. This becomes more clear by looking at the residual nonlinearity for the total impulse response. The red line looks like the quadratic input function. With additional terms from the second Volterra kernel, it is possible to reduce the residual nonlinearity to zero. This fit, as opposed to the fit for the Wiener static nonlinearity with an exponential function in Figure 4.3a, is perfect. This is because the second order kernel is the perfect representation for a second order nonlinearity. If a fourth order nonlinearity is used, the fit would not be as perfect.

The first kernel is basically zero and does not contribute much to the estimate. The second order kernel is more organized, has a positive peak and contains all of the dynamics. If the nonlinearity, rather than being centered near zero, was a quadratic function slightly shifted to one side (by adding a constant), then there would be both a linear component and the quadratic component. With this configuration, a first kernel and a second kernel would exist and the residual nonlinearity from the impulse response (red line on residual nonlinearity plot) would be much more asymmetric.

For the Hammerstein system with the same nonlinearity, the results are quite different. The impulse response is a perfect match to the original input impulse response; the residual nonlinearity has no pattern. However, when all the components of the second order Volterra kernel are taken into account, the residual nonlinearity becomes zero. The first order kernel does not contribute significantly while the second order kernel sits mostly on the diagonal. This is consistent with the information obtained for the monotonic nonlinearity.

Based on these trends, it is possible to characterize the results of Volterra kernels on real systems. Note that these simulations are fairly idealized with nonlinearities that are either Wiener or Hammerstein. When a DPN model with exponential

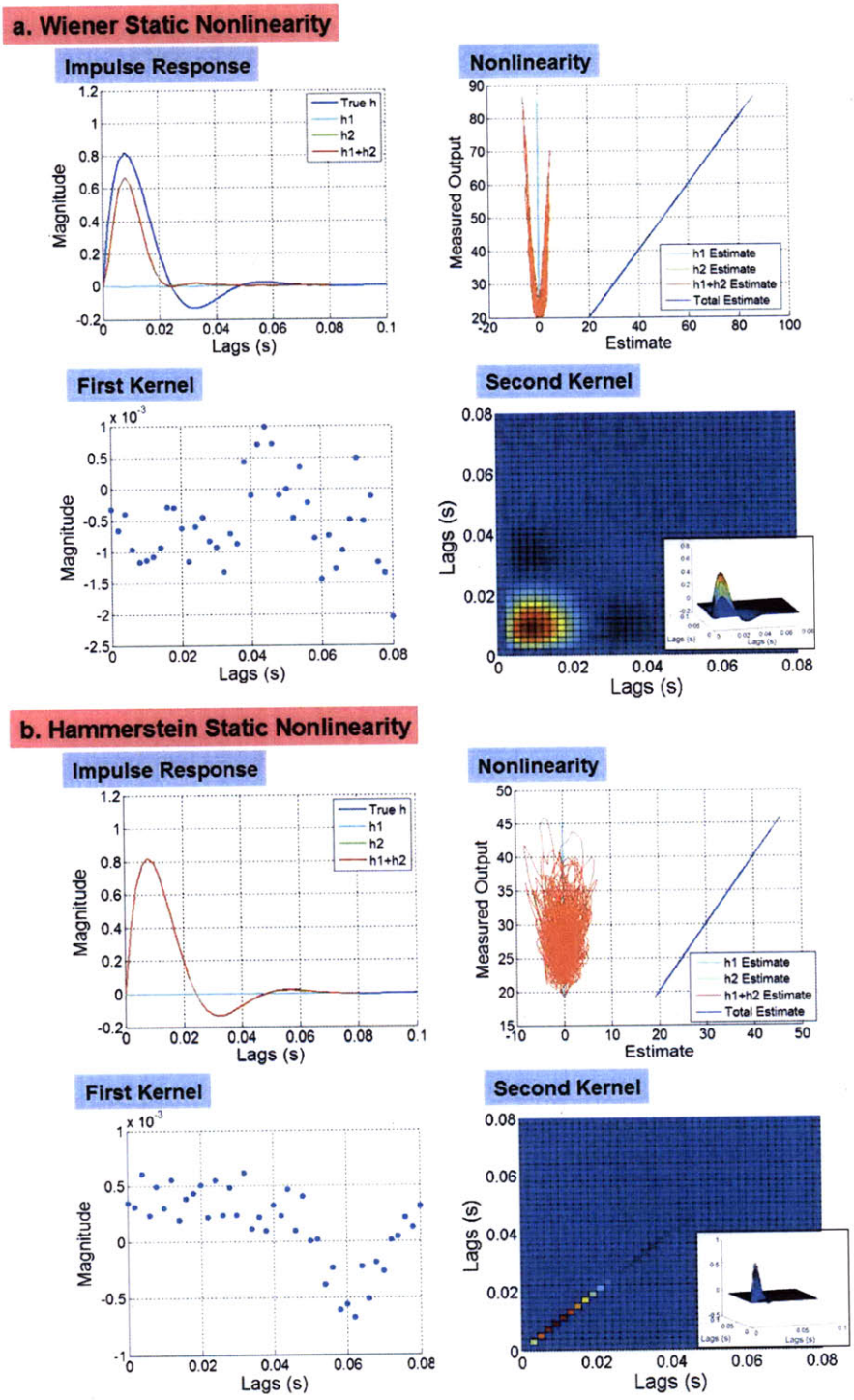


Figure 4.4: Volterra kernels for a simulated system showing the impulse response, residual nonlinearity, first order kernel and the second order kernel. (a) The functional form of the Wiener static nonlinearity used in this function the square of the input. (b) This same nonlinearity is used in the Hammerstein configuration.

dynamic parameter nonlinearities in the spring constant and the damping is used, different features can be produced. Figure 4.5 shows the second Volterra kernel for a simulated DPN system. The center lobe of the second Volterra kernel is more narrow than Wiener static nonlinearity representations and is flanked by two side lobes. The residual nonlinearity is large but the VAF is 95.8 % indicating that a second order Volterra kernel is still a good representation of the DPN model.

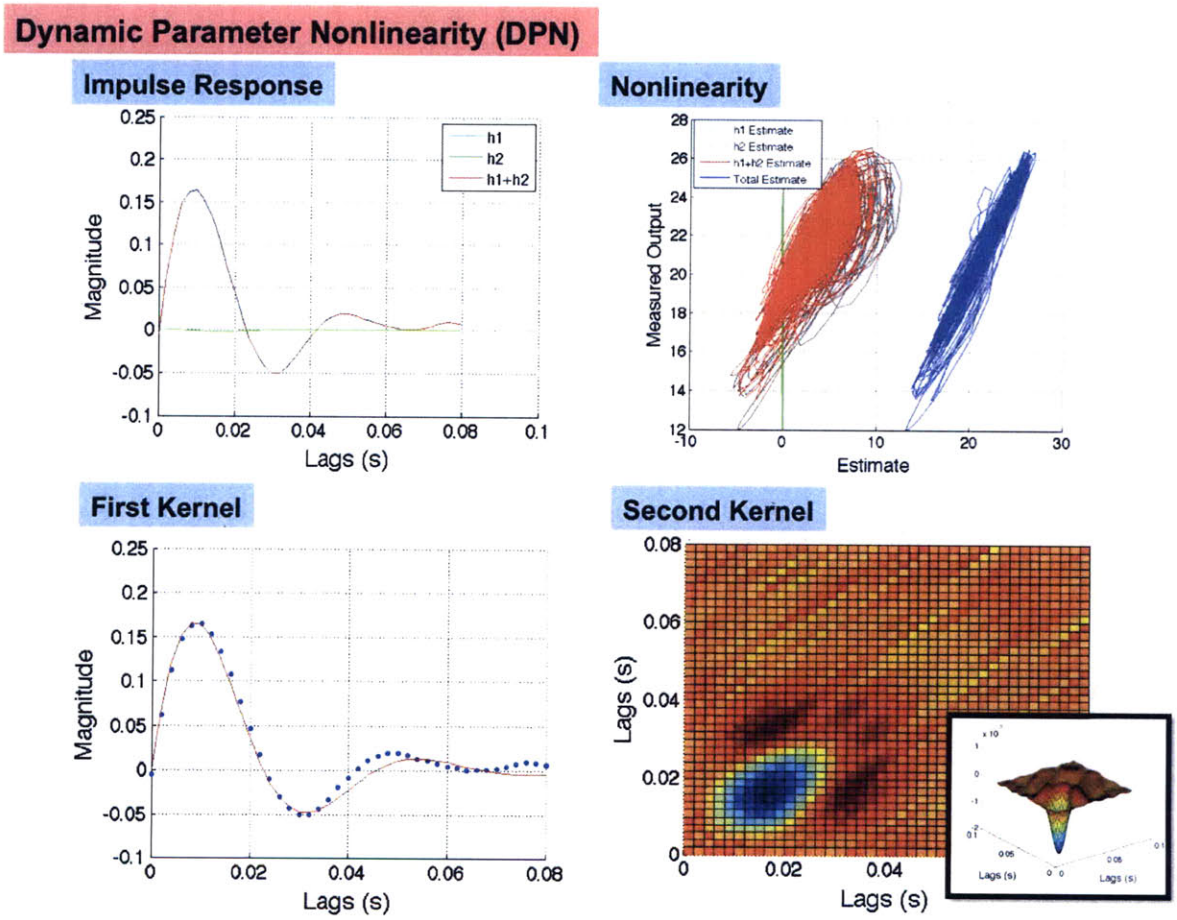


Figure 4.5: A simulated DPN model is used with the algorithm and the impulse response, residual nonlinearity, first order kernel and the second order kernel are shown. This simulation is very similar to experimental Volterra kernels and reproduces many of the salient features. Exponential nonlinearities were used for the spring constant and damping and no nonlinearity was used for the mass.

There are many possibilities for Volterra kernel representations of other types of nonlinearities. One possibility that is not a good fit, however, is a NLN system which is a linear system sandwiched by two static nonlinearities. Similarly, a Volterra

system does not do a good job of fitting a LNL system. These types of nonlinearities can be identified with other cascade methods.

4.3 Experimental Results and Post-Processing

Using the techniques outlined, Volterra kernels were used to analyze experimental data from skin subjected to indentation. The input is sampled at 2 kHz and the cutoff for the input is an 8th order butterworth at 200 Hz. Several distributions and inputs were tested. The distribution used for most of the data shown in this chapter was a uniform and was generated with the strategies listed in Chapter 6. A uniform input was used because it does a better job of exploring the range of the nonlinearity than a Gaussian input.

The input memory length has been shown to be around 250 samples for this sampling rate. Since this would result in an extremely long computational time, the information was initially downsampled by 3 to reduce the number of parameters and the computation time. Figure 4.6 shows the results of the first and second Volterra kernels. The first kernel looks very much like the expected impulse response. The relative size of the first order kernel compared to the second order kernel is also shown in the figure. Most of the data can be explained by the first kernel and the first kernel has a much larger impulse response magnitude than the second kernel contribution.

The second kernel appears to have no discernible pattern. In fact, the oscillations are so violent that there are nodes of noise that go in equal magnitude towards positive values and negative values. This leaves the second kernel completely uninterpretable. It is possible to determine the estimated output by reconstructing the values of A_m and P_m as shown in Figure 4.7. This result shows that the VAF is very high at a value of 96 % or better with a corresponding AIC of 1171.

With the second order kernel uninterpretable, it is possible that the kernel is actually fitting noise rather than any physiological phenomenon. Since the memory length is 40, there are 903 parameters in the fit. The total number of samples in the downsampled system is about 2700. This matches the factor of three rule of thumb

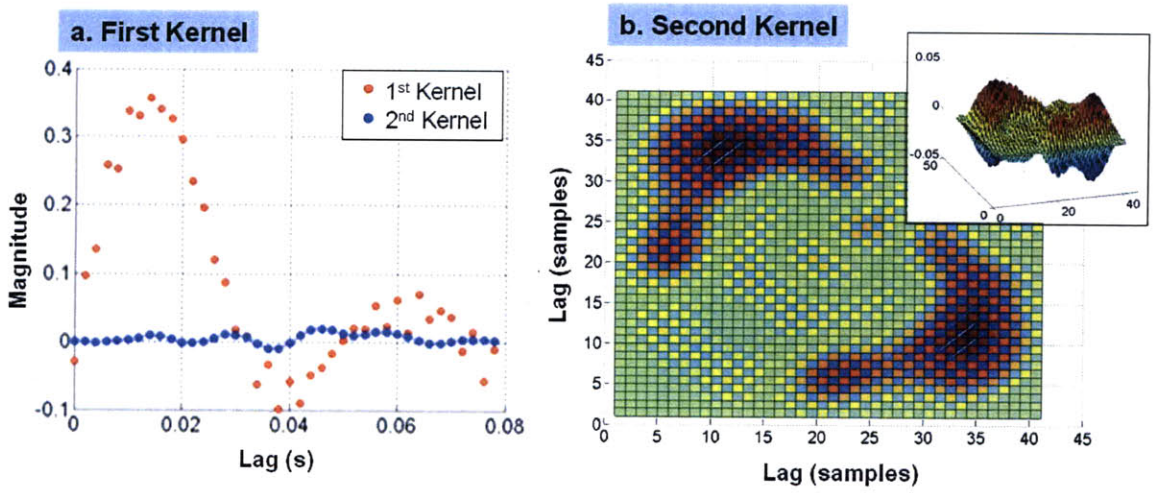


Figure 4.6: Volterra kernels for a skin measurements showing the (a) first order kernel and (b) the second order kernel. Data sampled at 2 kHz with an input cutoff at 200 Hz and downsampled by 3 before processing. Note that the Volterra kernel can be represented in terms of either samples or seconds.

so having too little data is not the likely culprit. In fact, for longer data samples, the same phenomenon occurs.

After exploring the phenomenon of the noisy second Volterra kernel, it was discovered that the source of the noise was the form of the input. In Gram-Schmit orthonormalization, each data point is orthogonalized one at a time. If two data points sitting next to each other are very similar, as with a signal that is heavily low pass filtered, then it becomes difficult to distinguish the information that belongs to one value of the lag versus the information that belongs to the value of the adjacent lag. In reality, this means that the input data itself is not sufficiently orthogonalizable.

Since there are no smoothness guarantees using this method of looking for the Volterra kernels, there should be no expectation that the algorithm will make the second Volterra kernel smooth. Although the algorithm can handle any type of input series, the results from some input types will appear to be noisy. There are several methods which can be used to produce smoothing in the kernel so that it is more easily interpretable.

One method is to use a cross correlation method or a functional expansion in the algorithm to impose smoothing. Since the second order kernel has already been

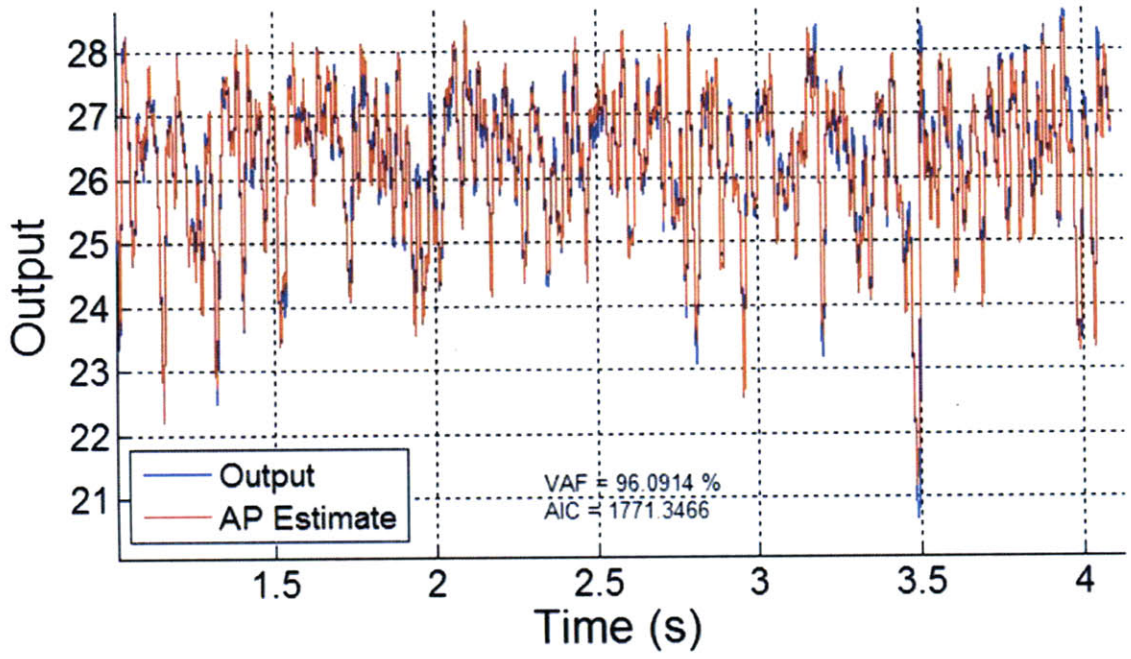


Figure 4.7: Output of the measured system compared to the estimate obtained by using Equation 4.2.

obtained, however, there are other methods which can be used to impose smoothing constraints in post-processing. With these processes, it is possible to look at the kernel before smoothing to determine if smoothing is necessary and then choose the appropriate smoothing technique afterwards. It is also possible to compare the goodness of fit before and after smoothing.

4.3.1 Downsampling

The first method for removing the kernel noise is downsampling. This will remove all the high frequency low-pass filtered information as shown in Figure 4.8. In order to get the data into a usable memory length, it is already necessary to downsample by a factor of 3. The power spectral density for a downsampling of 3 shows that at high frequencies, there still remains a lot of low pass filtered input data. For a downsampling of 5, however, the input no longer appears low pass filtered although the output is still dropping at high frequencies due to system dynamics.

The differences between these two systems is clear in the figures to the right which

show the second Volterra kernels. The kernel for the system that is downsampled by 3 is still uninterpretable and dominated by noise. The system that is downsampled by 5, however begins to show a pattern in the second Volterra kernel. It shows the negative peak that is seen in the monotonic (exponential nonlinearity) Wiener model or DPN model from the simulated systems.

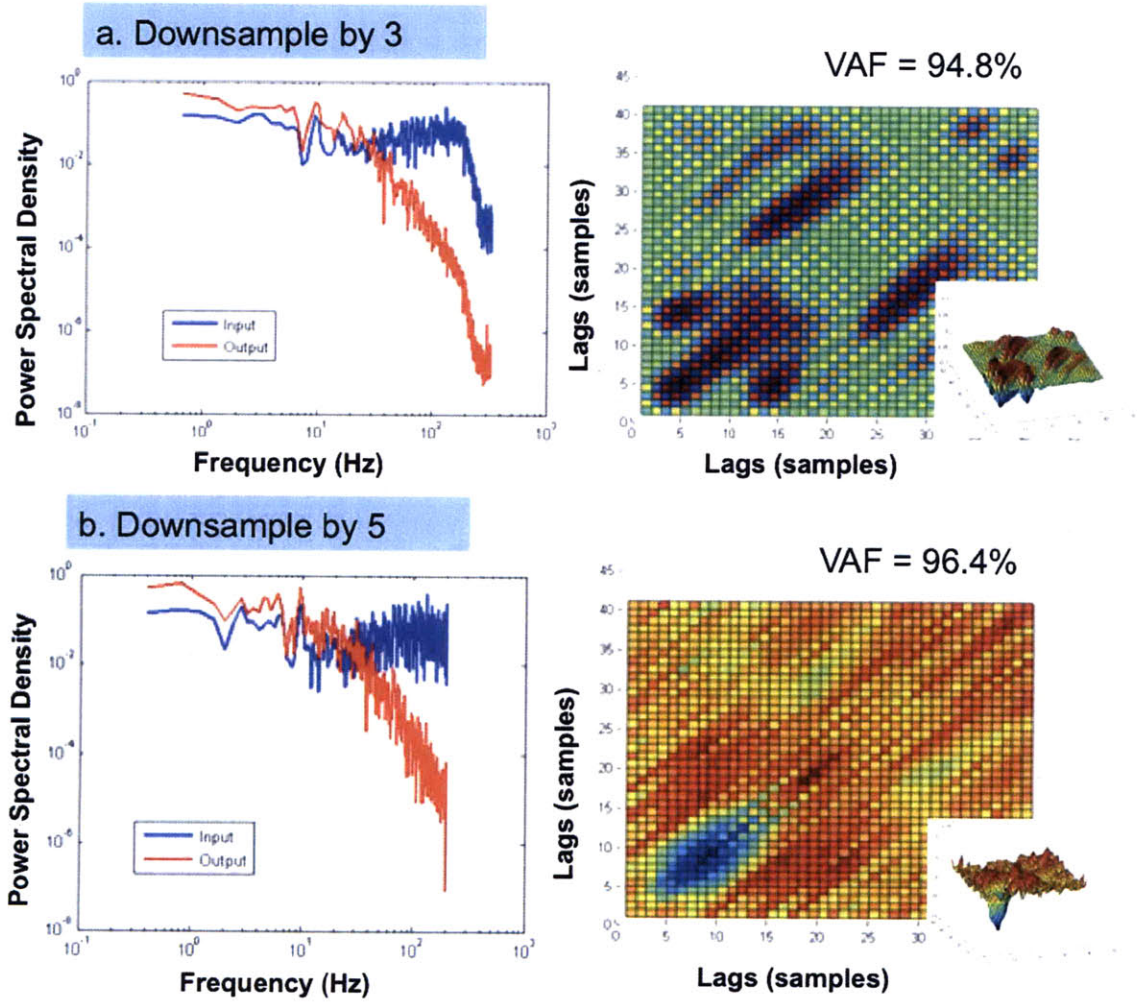


Figure 4.8: Downsampled Volterra kernels can be used to reduce the kernel noise. (a) Downsampling is done at a factor of 3 and (b) a factor of 5. The VAF increases after downsampling due to the fact that the data series is shorter after downsampling and not just because the estimate has become better. Downsampling increases the interpretability of the information. Note that the Volterra kernel can be represented in terms of either samples or seconds.

With the additional downsampling, the VAF increased from 94.8 % to 96.4 %. This increase is not simply due to the fact that the fit is better but is also due to the

fact that the data length had changed between downsampling by 3 and downsampling by 5. The shorter data length translates into a significant drop in memory length in terms of samples and the VAF increases (since there are fewer parameters to fit).

Although downsampling, until the input spectrum is essentially white, fixes the problem of producing interpretable second Volterra kernels, it does not do so in an optimal fashion. First, the data length is reduced so information is essentially lost. Second, it is no longer possible to use any desirable input signal. Lastly, the kernel estimates for the second kernel includes a lot of noise which is not present for non-downsampled white inputs. This is because the downsampling process inherently introduces errors because data points are being skipped.

4.3.2 Filtering

Another method that can be used to obtain the kernel information is to use a two-dimensional filter to directly filter the second Volterra kernel. In Figure 4.9, the unfiltered second Volterra kernel and the frequency domain content of this second Volterra kernel are also displayed. At high frequencies, near the edges of the plot, there are large features in the magnitude of the plot indicating that there is a lot of high spacial frequency content.

In order to reduce these large features in the second Volterra kernel, a two dimensional filter can be used to implement smoothing. This two dimensional filter has a Gaussian profile with a spread of 15 samples in all directions and is designed to remove the high frequency content while leaving the low frequency content intact. Simply setting the high frequency data to zero will not accomplish this task because in order to get a real output after the filtering process, both a real input and a real filter must be used. Simply removing frequency content will introduce imaginary terms which would cause the output, after the filtering operation, to be imaginary as well.

Filtering rather than downsamping results in a more readily interpretable second Volterra kernel. The features, however similar to the Wiener static nonlinearity data, are in actuality very different from the Wiener static nonlinearity. The kernel has

a high peak at the center that is flanked immediately by positive lobes. The peak is also narrower and there appears to be a small series of peaks along the diagonal. In the Wiener static nonlinearity, the peak is rounded and pushed into the corner. The side lobes are clearly at a different radial distance from the corner than the main peak and there are no features along the diagonal. Therefore, it can be concluded that the second Volterra kernel for skin under indentation is a hybrid nonlinearity that is very similar to a Wiener static nonlinearity but also contains other distinct features. If instead, it is compared with the DPN model in Figure 4.5, it is clear that the negative magnitude, the narrow peak, and the side peaks are reproduced. The DPN representation is therefore a better explanation of the experimental data.

While any type of input can now be used with this method, a two-dimensional filter inherently ignores some information and alters the shape of the second Volterra kernel slightly. In order to assess the effects of the smoothing filter, the VAF were compared; for a system that initially has a VAF of 94.7 %, the system after filtering has a VAF of 90.4 %. Although this is a about a 5 % drop in VAF, the data is much more interpretable. A less aggressive filter can also be used to produce more interpretable kernels with fewer changes to the ability of the second Volterra kernel to explain the data.

One way to assess the effect of the filter on the second Volterra kernel is to look at the residual in the nonlinearity as shown in Figure 4.10. Prior to smoothing, there is ample noise in the second order kernel such that the contribution of the second kernel impulse response produces the noise as indicated by the green line. Since the first kernel is essentially the same as the linear impulse response (the cyan line), which looks somewhat like the plot of the Wiener nonlinearity in Figure 3.8, it is clear that the first Volterra kernel was unable to remove the residual in the data by itself. With the addition of a second Volterra kernel, the residual is mostly reduced to a straight line indicated in dark blue.

When smoothing is added to the kernel, the magnitude of the second Volterra kernel is completely reduced so that the contribution from the first kernel is almost no different than the total impulse response contribution. When all of the off diagonal

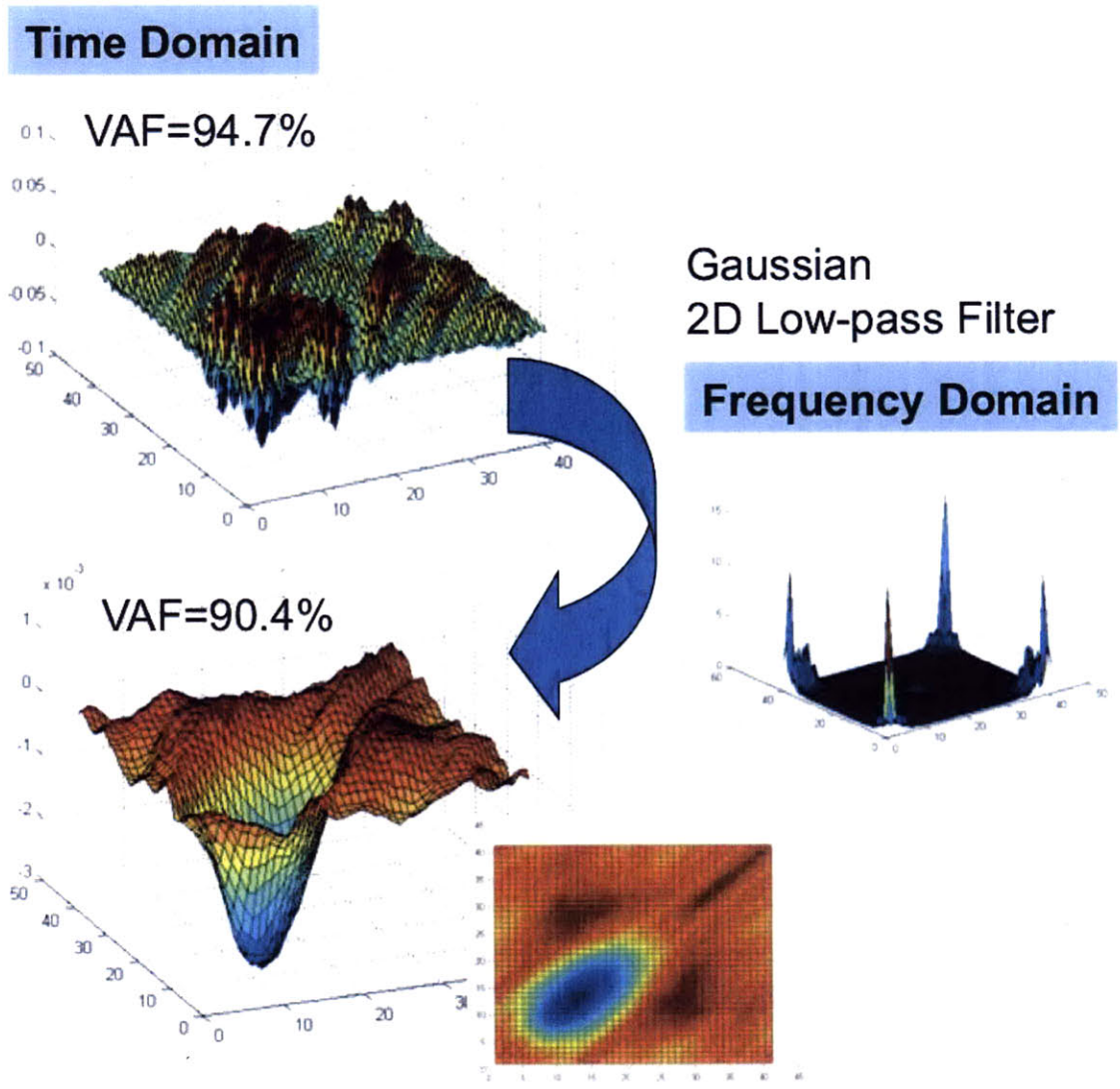


Figure 4.9: Filtered Volterra kernels can be used to reduce the kernel noise. Filtering with a Gaussian or other type of two dimensional filter will reduce the VAF but increase the interpretability of the information. This basically removes the high frequency components in the two-dimensional Fourier transform as shown in the figure on the right. The total data length was 2721 samples and the number of parameters was M = 903.

terms of the second Volterra kernel are added, the residual is reduced and looks very similar to the nonlinear residual before smoothing. There are a few slight differences between the two residual nonlinearities but the main result is that the residuals have been successfully reduced by both versions of the second Volterra kernel. This indicates that the smoothing filter is redistributing the information within the original

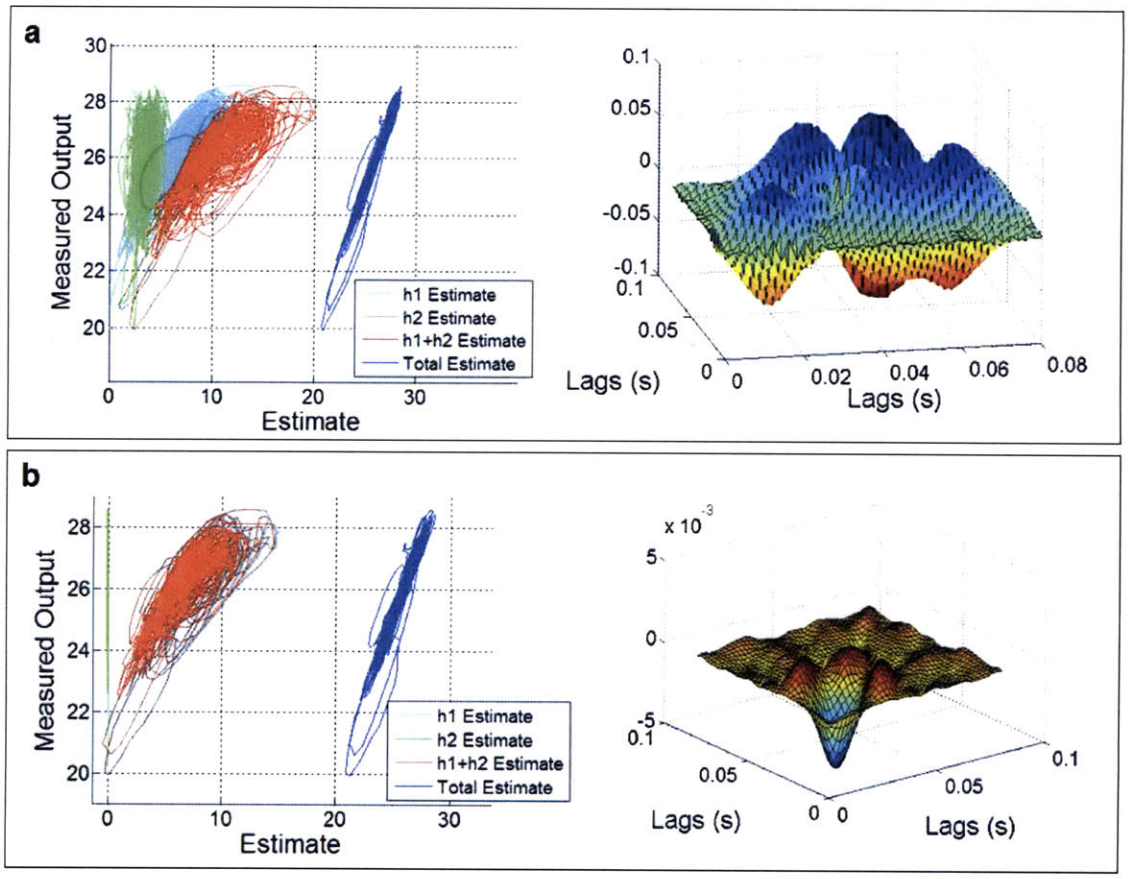


Figure 4.10: (a) The residual nonlinearity (all units in mm) of the initially obtained Volterra kernels and (b) the Volterra kernels after a smoothing filter operation.

second Volterra kernel such that the kernel is more interpretable without heavy losses in fitting ability.

One additional factor to note is that the looping behavior still occurs at the bottom of the residual plots. This could be due to the low number of data points at the lower values causing fits to be inaccurate. It could also be due to the fact that the Volterra kernels are global descriptions of a system. A better description of the system could come from localized representations.

The final output of the system can be compared with the experimental output of the system. Both the filtered and unfiltered estimates are shown in Figure 4.11. The VAF from the unfiltered system is around 96 % and for the filtered system around 93.7 %. Because the unfiltered system has a higher VAF, the corresponding AIC would be lower since both the filtered and unfiltered kernels contain the same

number of parameters. The unfiltered estimate does a good job of fitting the higher values (deeper depression into the skin) which seems to indicate that some of the noise in the second Volterra kernel comes from unsuccessful attempts (due to the constraint of the global representation) to fit data at deeper, stiffer locations.

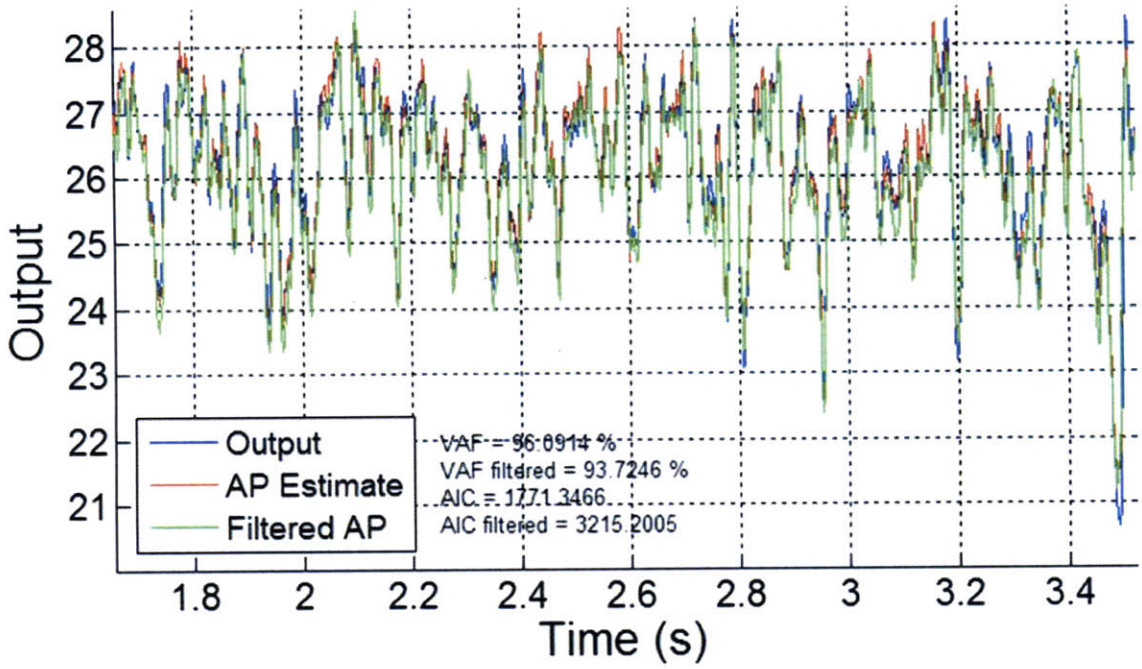


Figure 4.11: Filtered and unfiltered time domain results compared with the measured data.

With this technique, it is possible to begin comparing data between different locations on the skin. Data was obtained at four positions on the left arm: the posterior distal, posterior proximal, anterior distal and anterior proximal locations. The proximal locations were 40 mm from the elbow and the distal locations were 40 mm from the wrist. The data is shown in Figure 4.12. The positions are ordered in decreasing stiffness and damping. The effect becomes clear in the first kernel and in the second kernel. As the stiffness and damping decreased, the peak of the first kernel impulse response begins to rise and oscillations begin to appear. In the second kernel, the peak (blue center) begins to grow and oscillations (red) begin to appear.

This indicates the nature of Volterra kernels for stiffness and damping in a skin system under indentation. The shape of the overall second kernel does not change but changes in stiffness and damping increase the magnitude of peaks and troughs. This

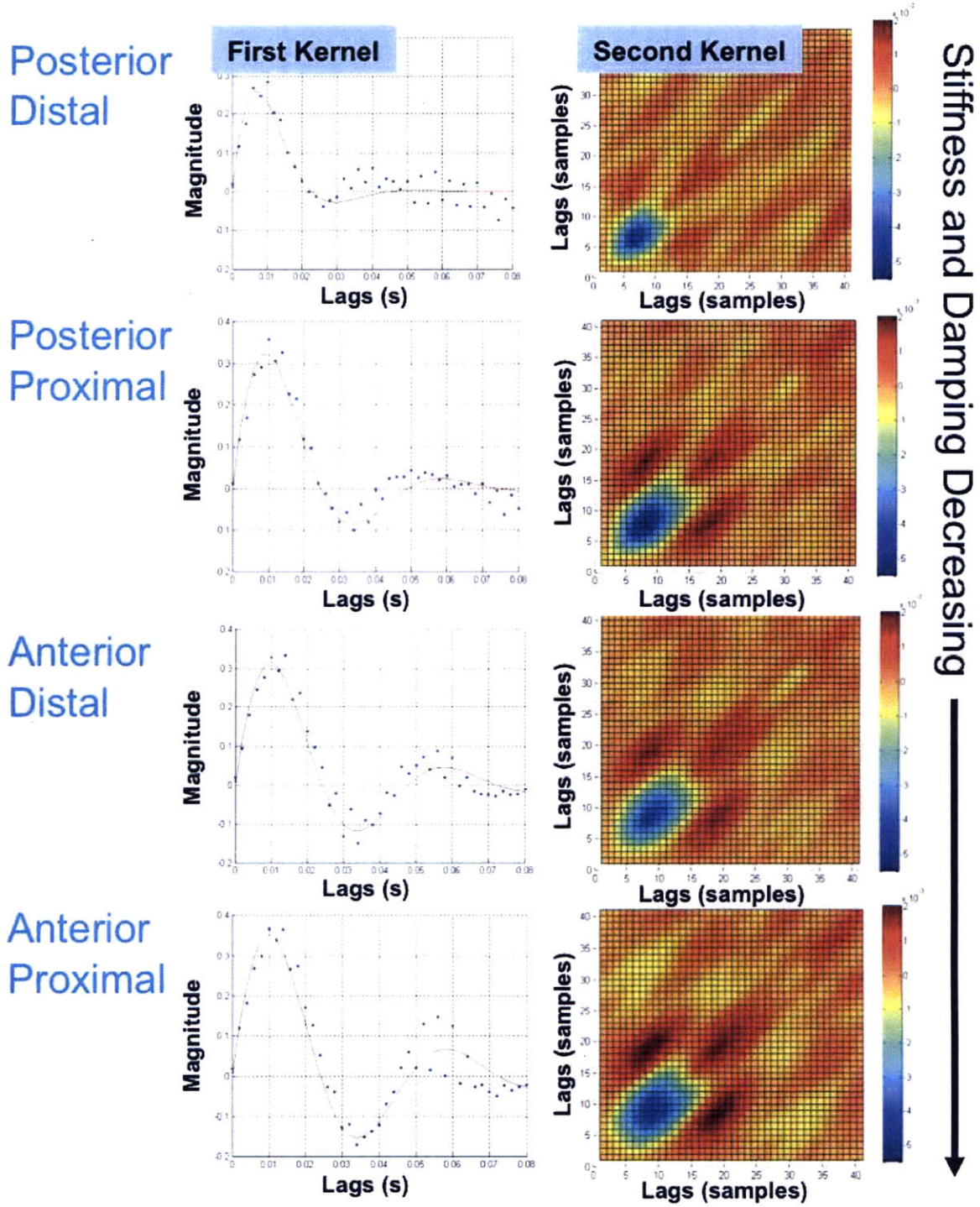


Figure 4.12: Comparison of filtered kernels for four different positions on the left arm including the posterior distal and anterior distal positions 40 mm from the wrist as well as the posterior proximal and anterior proximal positions 40 mm from the elbow.

means that systems with different stiffness and damping can be compared, to first order, by comparing the the relative values of different peaks in the first and second Volterra kernels. It is worth mentioning that this is a very qualitative comparison and that it is difficult for clinicians to compare these values graphically. It is therefore desirable to be able to do a quantitative comparison with fewer parameters or using different representations.

4.3.3 White Input Signals

A white input signal at 500 Hz with no downsampling is used as an input to the system and this is compared to a low pass filtered signal. This is shown in Figure 4.13. When a downsampled kernel is used, the data is uninterpretable. Filtering the kernel sampled at 667 Hz makes the output much clearer. The VAF can be calculated for different configurations with the same input and output data. Using a linear system identification technique, it is possible to obtain a VAF of 74.3 % with about a 5 % gain when a Wiener static nonlinearity is used. When the first order Volterra kernel is used alone, the VAF increases to 85.5 %. When the second kernel is added, the VAF jumps to 94.6 %. If the second kernel is filtered, the VAF drops about 4 to 5 % but the kernel itself is much more interpretable.

With a new white input, the second Volterra kernel is immediately visible without filtering. Additional filtering removes some of the striations in the data. When comparing the VAF of several different techniques on the data, it is clear that the initial VAF for the linear case is higher at 79.6 %. When a Wiener static nonlinearity is used, the VAF increases by about 3 %. The use of just the first Volterra kernel decreased the VAF to 76 % but the addition of a second kernel boosted the VAF to 94.5 %. This second kernel is not only interpretable but has a higher VAF. Filtering the kernel to add more smoothing causes the VAF to drop to 92.6 % which is less than the drop for the low pass filtered signal. This drop, however, indicates that information was lost at the filtering step since the shape of the second Volterra kernel was already interpretable.

It is interesting to note that downsampling and filtering produce a few small dis-

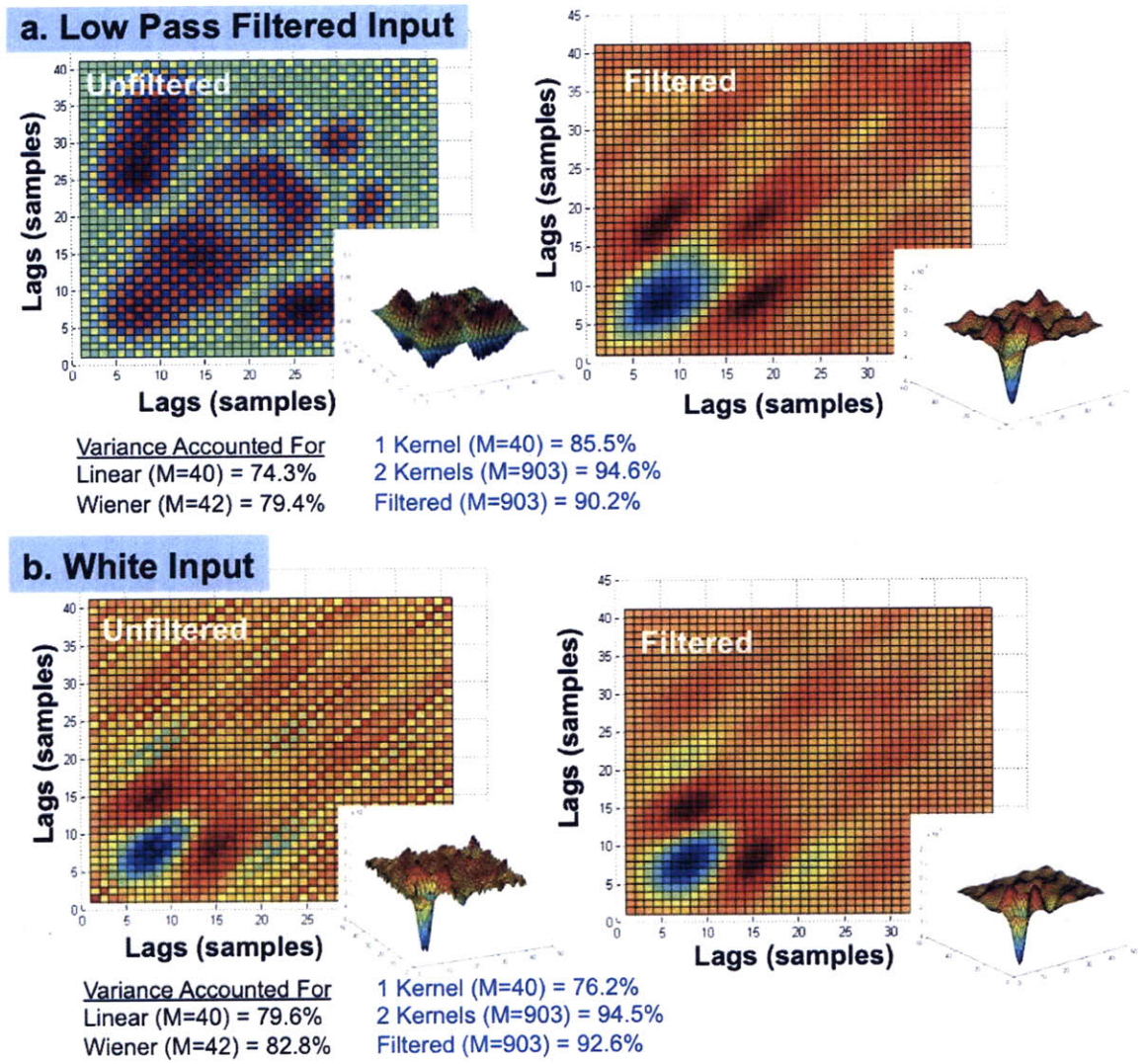


Figure 4.13: Comparison of (a) filtering method with (b) white input signal method. When a white input signal is used as an input to the system, the kernel becomes immediately visible without filtering. The effective sampling rate on (a) is 667 Hz and the effective sampling rate on (b) is 500 Hz.

tortions in the second order kernel. The kernels obtained from white signals exhibit more rounded peaks wheras the kernels obtained by downsampling and filtering exhibit sharp features in the peaks around the edges of the second kernel.

Data from different positions on the skin can also be compared using a white input technique as shown in Figure 4.14. The plots are also placed in order of decreasing stiffness and damping. In these plots, however, the pattern between different values in stiffness becomes less clear. There is an overall pattern as the peak of the first

and second kernels increase. The posterior proximal position and the anterior distal position, however, appear to switch rankings with respect to the measurements taken earlier in Figure 4.12.

Since the cutoff on the earlier figure (Figure 4.12) was at 200 Hz, a white signal up to 500 Hz will explore less of the input range thereby producing results with fewer distinctive nonlinear features and less specificity. This is the opposite of what one would expect from linear system theory. In linear systems, one would expect that as the input frequency increases, more frequency data is obtained which means there should be more specificity. With a nonlinear system, however, there is data which can be obtained from the input range and not just the input frequency. A lower frequency input will have a larger resulting output range than a higher frequency input with the same input distribution.

Taken together, this means that a higher input frequency results in a lower range of outputs and therefore a lower range of measured nonlinearity. This will give a lower overall perceived stiffness. A white input with a cutoff at 200 Hz could produce this output range but would be subject to other signal processing problems. Therefore, for nonlinear system identification, using a low pass filtered input is desirable because the low pass filter can be used to control the input range.

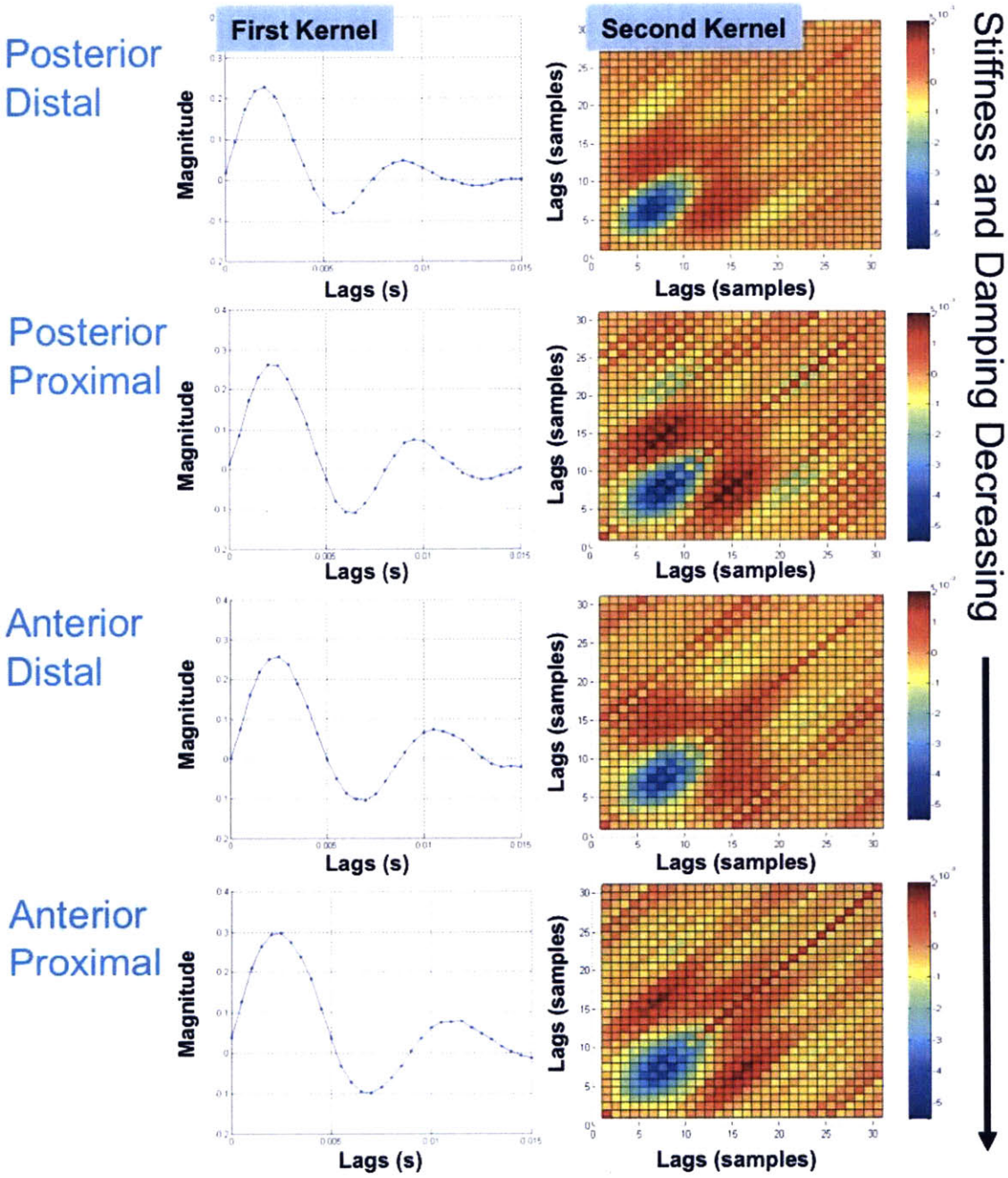


Figure 4.14: Comparison of kernels derived from white inputs for four different positions on the left arm including the posterior distal and anterior distal positions 40 mm from the wrist as well as the posterior proximal and anterior proximal 40 mm from the elbow.

4.4 Summary

Volterra kernels can be used to identify several different types of nonlinearities including Wiener and Hammerstein static nonlinearities. In addition, the order of the kernel used represents the maximum polynomial order of the nonlinearity that the technique is able to fit. This means that a second order Volterra kernel can fit non-monotonic nonlinearities, such as quadratic functions, perfectly. A technique for identifying Volterra kernels based on Korenberg and Hunter method is presented and representative data from several different simulated systems is discussed. Data from indentation studies are shown and several post processing techniques are presented. A summary of these post-processing techniques is shown in Figure 4.15.

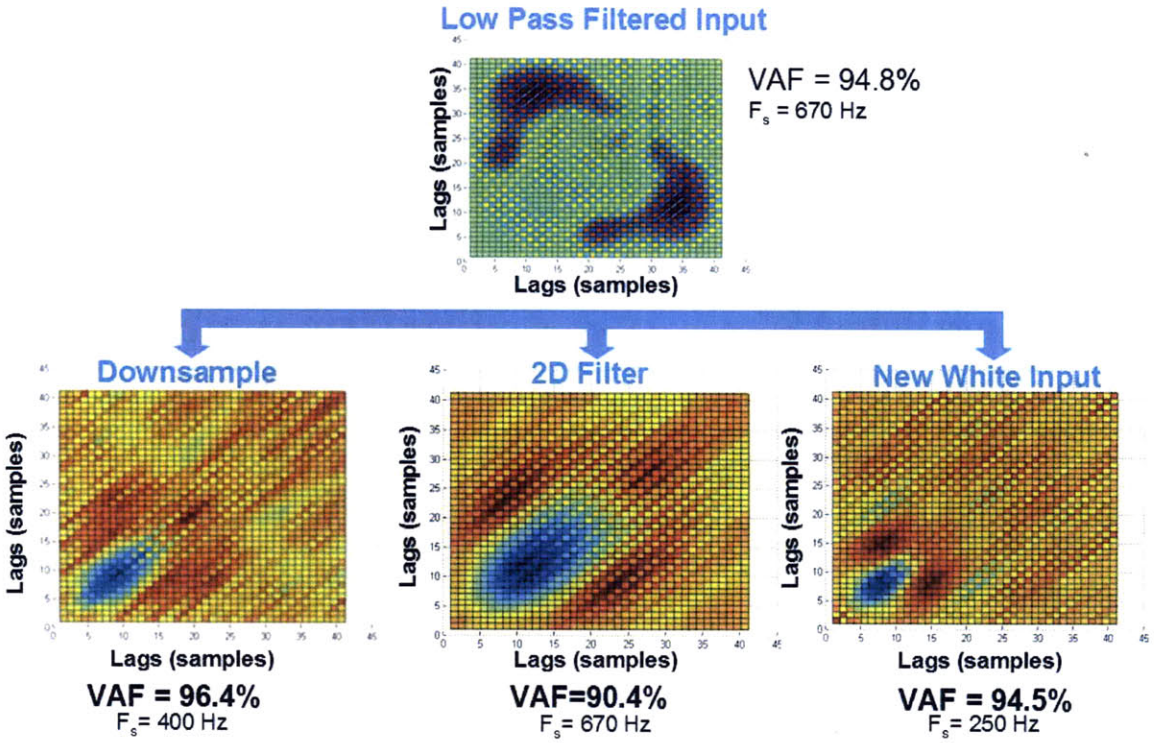


Figure 4.15: The three basic strategies for reducing the noise in Volterra kernels are shown.

Post-processing techniques include downsampling, using a two dimensional smoothing filter, or using a white input signal. All of these techniques help make the second Volterra kernel more interpretable by smoothing the kernel. Downsampling is undesirable because one must downsample until the input is essentially a white input.

With this technique, a lot of high frequency data is lost. A white input signal is also undesirable because it imposes a fixed structure on the input signal. A two-dimensional filtering technique is generalizable to any set of inputs and any choice of downsampling; the major disadvantage being the loss of the fitting ability of the second order kernel. For an original VAF of 94.8 %, an input filter can reduce the VAF to 90.4 %, downsampling can increase the VAF to 96.4 % (but this is due to the fact that downsampling reduces the number of data points), and using a white signal maintains a high VAF comparable to the original.

By choosing post-processing techniques to look at the second Volterra kernels, it is possible to compare the original kernel with the post-processed kernel and make an informed decision about the shape of the nonlinearity. Although there are practical restrictions to the input signal for the algorithm presented, additional post-processing has helped overcome some of them. The heart of the algorithm is based on a Gram-Schmidt orthonormalization technique which can be used to solve not only Volterra kernels but any other type of basis function as long as certain conditions are met. There are several applications of this algorithm including different basis functions and partitioning systems which will be explained in the following chapter.

Chapter 5

Partitioned Techniques

This chapter presents a few nonlinear system identification techniques based on the Gram-Schmidt orthonormalization solver [63, 68] henceforth known as the partitioned techniques. These techniques have interpretability comparable to linear models and yet are able to model higher order static nonlinearities. With modifications to the kernel structure, they are also capable of modeling directional nonlinearities (where the dynamics of going in one direction is different from the dynamics of going in the opposite direction) and non-monotonic static nonlinearities.

The motivation for using a partitioning technique for stochastic system identification follows from a few desirable characteristics: short testing length, physical interpretability (easily quantitatively comparable) and flexible model structures. The concept behind the technique is to break up the input signal into groups based on a set of rules such as the output level, the direction of the signal (whether it is going in a positive or negative direction), or some fuzzy logic-based input and output rules. Then, by using direct orthonormalization as a solver, the "impulse" responses or kernels for each of these rules can be obtained.

The use of stochastic inputs is desirable because the technique queries multiple frequencies at once. When using stochastic inputs with nonlinear systems, there are several ways to approach the initial identification. First, the user can use a small perturbation range since most physical systems can be linearlized for small regions. This localized linear technique, however, will require multiple separate tests or will

require the user to add a ramp or sine function to the stochastic signal during the test. After the test, the user will have to do linear system identification on small chunks of locally linear data. This process can be slow and it is desirable to use a faster technique that can obtain linear and nonlinear information at the same time.

System identification can be used to control the output of a system or can be used to identify unknown characteristics. For the purposes of control, it is usually not important to be able to physically interpret the form of the system (unless it is to analyze failure modes *etc.*). Interpretability is critical for identifying and understanding characteristics. Some parametric and nonparametric mapping structures suffer from lack of interpretability. The physical meaning behind the weights in neural nets or wavelets obviously presents difficulties [53, 94]. Even Volterra kernel structures can be difficult to interpret in physical terms especially when higher order kernels are used. The proposed identification technique, however, produces waveforms similar to impulse responses which can be mapped to transfer functions and therefore mapped to physical parameters like masses, dampers, and spring constants. The nonparametric nature of the technique, however, makes sure that the model is fitted according to very general assumptions. Being interpretable also tends to confirm that the model is not fitting to noise in the system.

Lastly, the solution technique presented shows flexibility in the model structure. Simple monotonic static nonlinearities, non-monotonic static nonlinearities and direction-dependent systems, global and localized basis functions can be modeled depending on the partitioning rules imposed. Volterra kernels can be thought of as a particular partitioning rule where the rules are global across the entire input and output range. Simple linear analysis can also be thought of as a particular partitioning rule that is also global. The partitioned techniques presented have basis functions that are localized over sections of the input and output ranges.

5.1 Depth Dependent Partitions

The formulation for depth dependent partitions assumes a monotonic static nonlinearity and is not applicable for non-monotonic nonlinear functions. The basic idea behind this representation is that the dynamics of a system change as a function of depth into the skin and that the dynamics do not have a particular pattern that must be matched by all the constituents. For example, this means that M_e, B_e and K_e do not have to evolve with the same underlying static nonlinearity. Data from different depths is loosely grouped together and an overall "impulse" response or kernel for that group is given. This is similar to the idea of completing localized linear system identifications with inputs of smaller ranges.

The key difference between this technique and simply completing localized linear system identification is that this technique imposes the separation of the different depths after the data over the entire output range is collected. This means that it can be used to artificially group non-contiguous sections of data that are collected at the same depth in order to make an estimate of the dynamics.

It is expected that the "impulse" responses from these partitioned kernels would not produce results that look exactly like the results obtained from localized linear tests. The main reason is because the localized linear techniques contain little or no data for cross dynamic terms between different depths. The depth dependent partitioning, however, does contain cross dynamic terms which will cause some averaging to occur across the kernels. In the case where there are no cross dynamic terms, however, the localized linear and depth dependent partitioning techniques would produce exactly the same results. Other more advanced types of partitioning schemes can be used to separate the cross dynamic terms.

The method used to solve for the partitioning technique involves using the Gram-Schmidt Orthonormalization technique that was presented in Section 4.1. The construction and reconstruction steps, however, are replaced by the equations listed below. The construction equation for the depth dependent partitions listed below

would replace Equation 4.4 in the solution process,

$$P_m(n) = \begin{cases} 1 & \text{for } m = 1 \text{ and } n = I+1 \ldots N \\ x(n-j+1) & \text{for } m = 2 \ldots K_{max}(I+1), \\ & \quad n = I+1 \ldots N, \\ & \quad \text{and } j = 1 \ldots I+1 \\ & \quad \text{when } L(k) \leq y(n-j+1) \leq L(k+1) \\ & \quad\quad \text{for } k = 1 \ldots K_{max}, \end{cases} \tag{5.1}$$

where K_{max} is the total number of partitions, k is the partition counting variable and $L(k)$ is the partition breakpoint. The most important criteria necessary for generating a partition scheme is that the construction equation must be orthogonal. This means that there cannot be overlapping segments or repetition of any segments. For stability and noise rejection, the output y used for the construction of the kernels can be low-pass filtered while the y used for the solution steps are not altered.

The following equation would then be used to replace Equation 4.14 in the solution process for reconstruction. This equation contains information for different partitioned kernels, each of which is similar to a "impulse" response. The total number of these kernels is equal to the number of partitions K_{max},

$$\begin{aligned} \widehat{h}_p(i,j) = A(m) \qquad & \text{for } i = 1 \ldots K_{max} \text{ and } j = 1 \ldots I+1 \\ & \text{where } m = j + (i-1)(I+1). \end{aligned} \tag{5.2}$$

The number of partitions can be chosen to be any value up to the fitting limit. This means that the total number of parameters (which is equal to the number of partitions multiplied by $I+1$) must be at most one third of the total test length.

The partitioning breakpoints can be set based on several different criteria. The first obvious criteria would be equal partitions where each partition covers the same distance. Realistically, this means that each partition would have a different number of data points which would then cause some partitions to have too few data points

for proper fitting.

Another possible partitioning breakpoint algorithm would involve choosing breakpoints such that each partition has the same number of data points within it. This avoids the problem of having too few data points in any partition but causes some averaging effect to occur. This is the type of breakpoint that is used in most of the following analysis. The algorithm used to generate these breakpoints involves putting the entire output data series in order from the lowest value to the highest value. The break points can then be chosen to split the number of data points evenly. The depth value that would correspond to this even split can then be directly chosen from the ordered series.

In the ideal case, it is best to have partitions that are equally spaced where each partition has an equal number of data points. This would mean that the ideal output distribution would have to be uniform over the test range. This type of constraint can then be used as a criteria for optimizing the selected input. Methods for accomplishing this are discussed in Section 6.3.

It is important to point out that the depth dependent partitioning scheme is directly dependent on the output of the system whereas the Volterra kernels were dependent on only the input. Therefore, any computational savings in the solution method for the Volterra kernel, where the input orthogonalization is pre-calculated, cannot be used. The output partitioned kernels, however, generally have fewer parameters than the Volterra kernels which means that the computation time is lower.

5.2 Performance

Before going on to examine real data, the performance of the depth dependent kernel algorithm is shown. A Wiener static nonlinearity is generated using the nonlinearity in Equation 3.26. This simulated nonlinearity has no cross terms in the dynamics since it is a pure static nonlinearity. Therefore, it is expected that the localized linear solution should be a perfect match with the results from the depth dependent partitioning technique.

Two types of inputs were generated to interrogate the simulated static nonlinearity. The first input was a stepping function with random noise superimposed on top of it shown in red in Figure 5.1. The dotted lines shown in the output indicate the breakpoints between different segments. This is essentially the type of dataset that would be used for localized linear testing where each section is at least as long as the memory length of the system. The second input was a Gaussian stochastic input signal, shown in blue in Figure 5.1. Each of the data segments is not guaranteed to be as long as the memory length of the system. The data was broken up into the corresponding effective mass, damping, and spring constants. It is clear that these three components of the static nonlinearity all evolve with the same nonlinear function as the depth changes.

The identified nonlinearity from the two inputs match the trend of the nonlinearity well with some deviation near the limits. Since the data is grouped in order to maintain a certain number of data points in each partition, the additional averaging brings down the estimate for the higher values of stiffness, damping and mass while increasing the estimate for the lower values of stiffness, damping and mass. This effect gives a slight error to the estimate that is a function of the total number of data points and the continuos data length.

Simulation data for this is shown in Figure 5.2. In these simulations, the system memory length is 60 samples and 5 partitions were chosen. The number of repetitions represent the number of times a chunk of data of a specific input offset value was put into the system. With more repetitions, the total dataset length increases and fewer

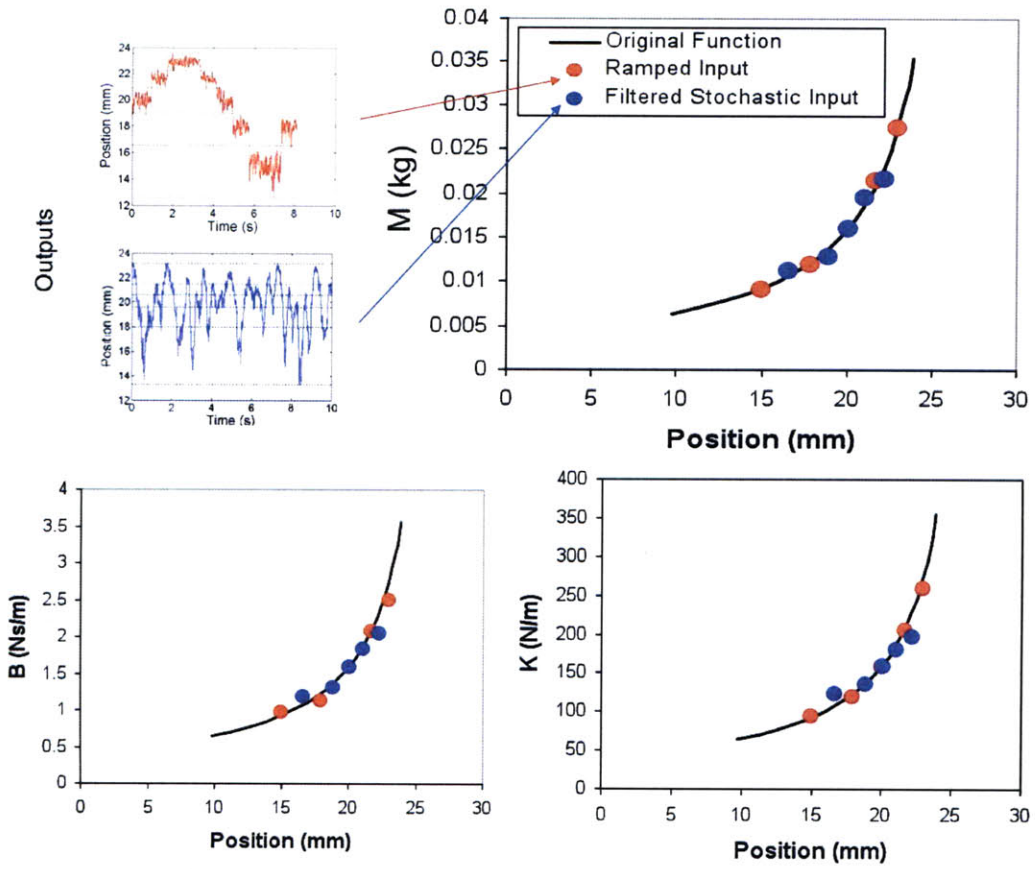


Figure 5.1: Depth dependent partition validation with a Wiener static nonlinearity shows that mixed signals can be partitioned and solved to obtain the original Wiener static nonlinearity. The black line indicates the original theoretical nonlinearity which can be obtained using a localized linear technique and the red and blue points are from depth dependent partitioning technique identification of the same nonlinearity using different inputs.

continuous data sets are necessary to make a low error estimate. The shorter the continuous data length (for the same number of repetitions), the higher the error associated with the estimate. When the data length increases, the percent error in parameter estimates tends to decrease. In order to obtain an estimate error of 5 % or less, at least 500 data points are needed (multiplying number of repetitions by the continuous data length and the number of partitions). Note that the total number of parameters is 300 which indicates that having at least 2 to 3 times more data than the number of parameters will produce lower error estimates. As more and more data points are used, the estimate improves until the value approaches the original model.

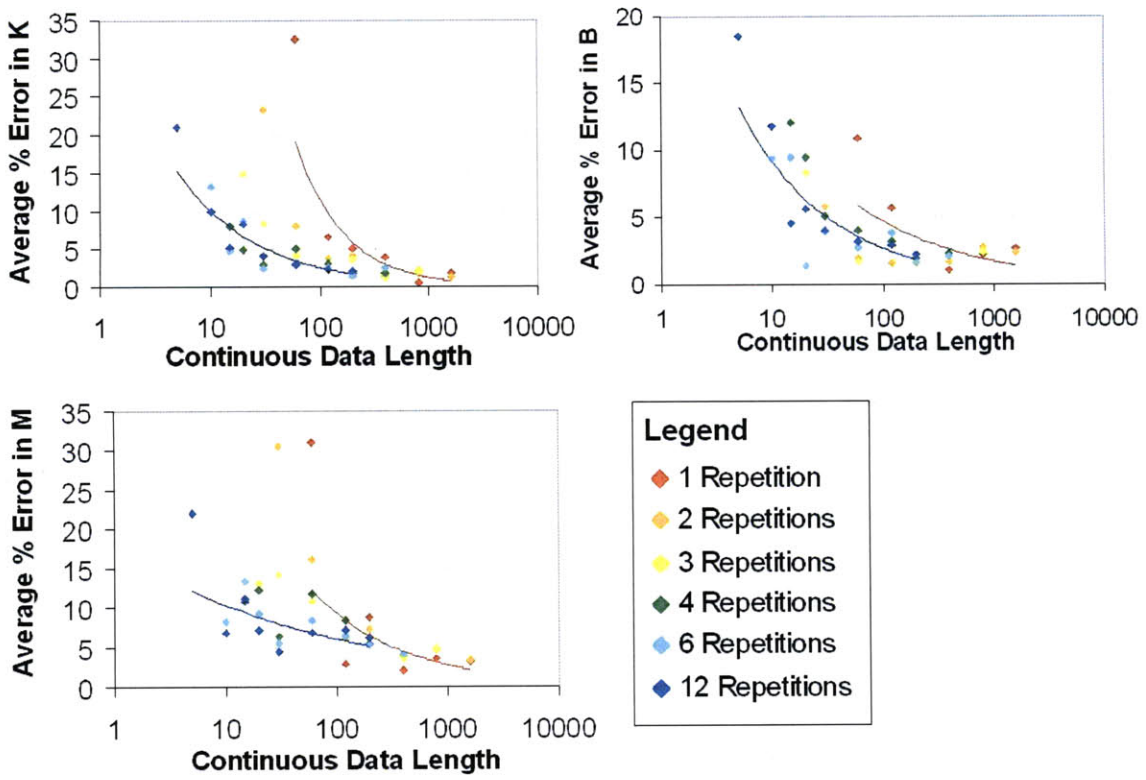


Figure 5.2: Depth dependent partition validation as a function of continuous data length for simulated data. As the data length increases in log space, the percent error decreases. As the number of repetitions increase, the average error also decreases. The lines serve as a guide to the eye for the trends associated with the maximum and minimum repetitions.

With a DPN model, the fit is also close and will approach the real value as more parameters are used. Figure 5.3 shows how the depth dependent partition technique works for a DPN model. The nonlinearity is different for all three parameters with exponential forms for the spring constant and damping and a constant for the mass. This method can be used as a nonparametric technique for identifying many different types of depth dependent parameter nonlinearities.

5.3 Experimental Results

Based on these performance data, results can be examined for experimental data. Since the partitioning technique is less sensitive to the memory length of the system, it is possible to use the full 2 kHz of data without downsampling. Some downsampling however, is useful for faster computation. Figure 5.4 shows the breakpoints for the

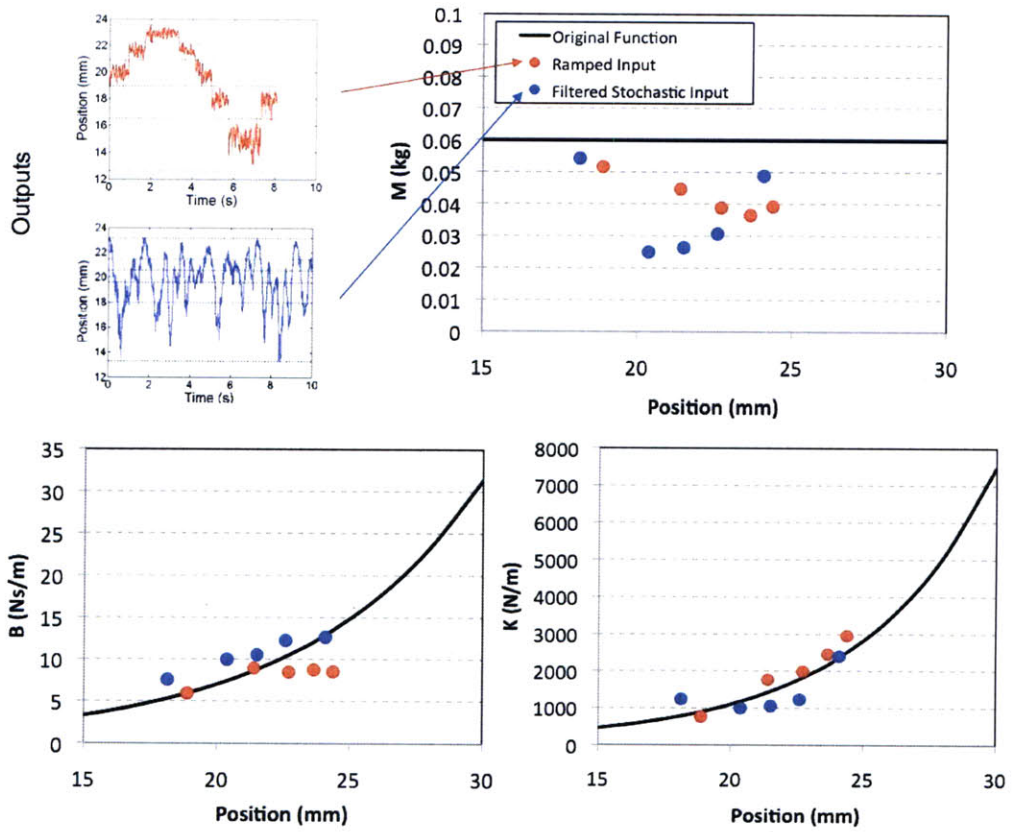


Figure 5.3: Depth dependent partition validation with a DPN model shows that mixed signals can be partitioned and solved to obtain the original DPN model. The black line indicates the original theoretical nonlinearity. The red and blue points are from the depth dependent partitioning technique identification of the same nonlinearity using different inputs.

partitioning along with the kernels obtained from the estimate. The breakpoints were chosen by data density so they are clustered heavily at the deeper values. One large partition exists at the very deepest values and one large partition exists at the shallowest values.

The resulting kernels look like "impulse" responses as a function of depth. The deepest depth at 22.9 mm has a very heavily damped "impulse" response while the most shallow depth at 16.9 mm is not heavily damped and shows more of the negative oscillatory peak. One interesting feature is the large negative value estimate of the zero lag in the most shallow kernel. This phenomenon was investigated and it was found that this point represents the large offset of the kernel from the location of the average value. Since the average value is still being estimated in the partitioned kernel,

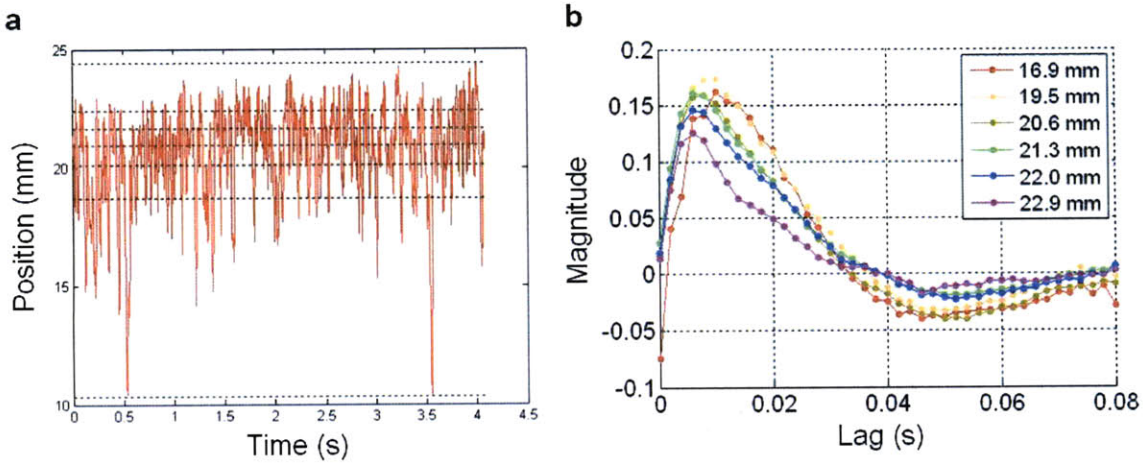


Figure 5.4: (a) Depth partitioned results for skin showing the partitioning of the output record (b) chosen by data density and calculated kernels at different depths.

there is some coupling of all the kernels with the average value. If each individual kernel was given its own estimate of the average value, the estimate obtained is generally less accurate.

5.3.1 Comparison with the Wiener Static Nonlinearity

To assess the goodness of fit, the VAF and AIC are calculated and the residual nonlinearity is plotted as shown in Figure 5.5. In this data set, there are 15 partitions and the data was downsampled by 4. The estimated memory length was 40 samples. The VAF of this system is fairly good at 92.2 %. This is on par with results obtained using Volterra kernels except with fewer parameters (600 parameters total for the partitioned technique versus 903 parameters for a second order Volterra with the same memory length).

The plot to the right shows the residual nonlinearity. This residual has significant noise grouped around each position in depth. The dependency, however, is fairly linear except for the lower values of the nonlinearity. This is probably due to the low number of data points obtained at lower values and the large size of the partition. The noise in the residual nonlinearity is most likely due to other dynamics like hysteresis or direction-based tissue dynamics associated with fluid outflow.

The two panels on the bottom of Figure 5.5 show the evolution of the kernels as a

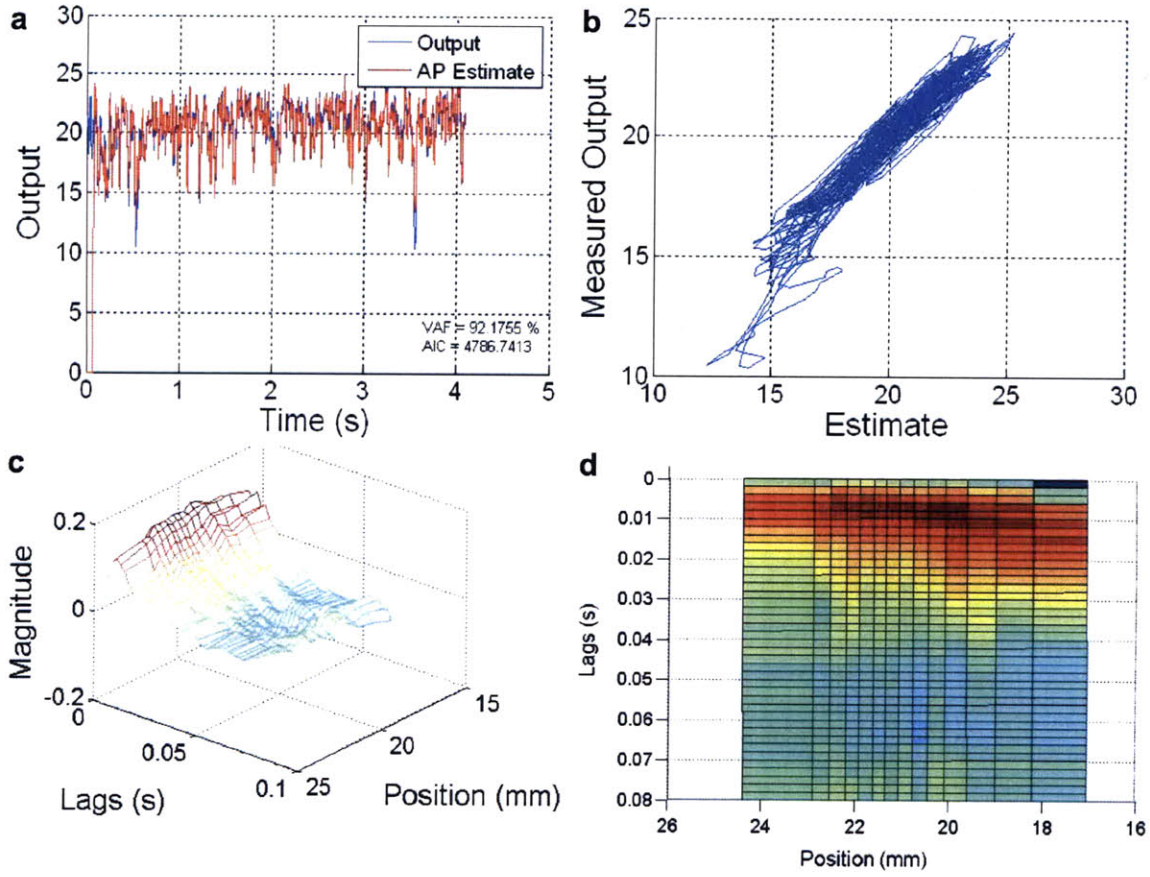


Figure 5.5: The depth dependent results with 15 partitions. The panels are (a) the output estimate, (b) the residual nonlinearity, (c) the impulse response from a three dimensional view and (d) the impulse response from a top view.

function of depth. As the position becomes deeper, the peak (shown in red) decreases in width. At shallow depths, the peaks become broader but stay at about the same height. The maximum in the peak moves slightly as a function of depth.

In order to fully assess how this differs from a Wiener static nonlinearity and a DPN model, simulated systems are directly compared with a experimental system as shown in Figures 5.6, 5.7 and 5.8. All three figures have 6 partitions and are downsampled by 4 from a 2 kHz dataset. In Figure 5.6, which is the Wiener static nonlinearity simulated system, the VAF of the fit is very high (near 96 % with 240 parameters in this fit). The residual nonlinearity is fairly linear with small deviations at the lower depths due to the size of the partition. In the simulated system, the peak value increases as the depth decreases and the lag on the peak value does not

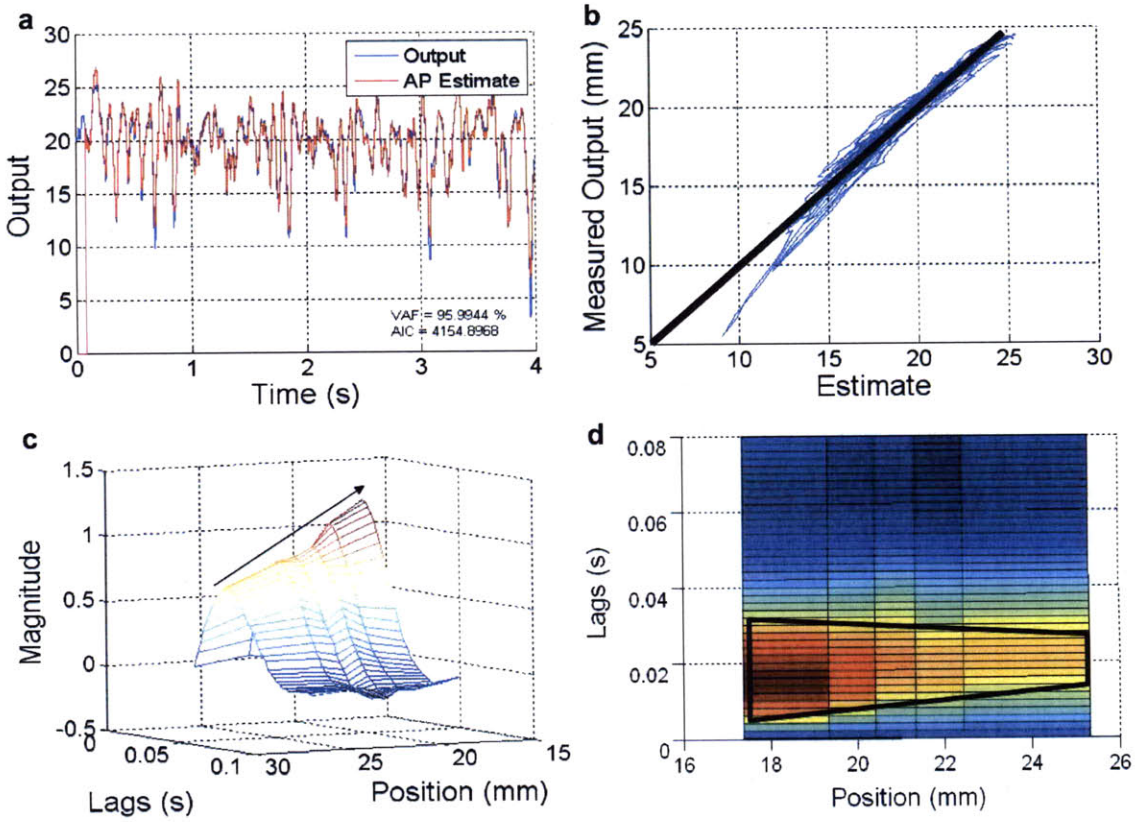


Figure 5.6: The depth dependent simulated results with 6 partitions. The panels are (a) the output estimate, (b) the residual nonlinearity, (c) the impulse response from a three dimensional view and (d) the impulse response from a top view.

change as a function of depth (see black box). The experimental system show in Figure 5.8 has a VAF of 90.3 %. The nonlinear residual shows the same trends as in the simulated Wiener static nonlinearity system. The kernel plots, however, look slightly different. The peak value of the kernels does not increase significantly and the location of the peak shifts backwards towards zero for deeper kernels (see black box).

If a DPN nonlinearity is used, however, the results in Figure 5.7 look very much like the experimental results. Once again, it is apparent that the Wiener static nonlinearity is a good first order approximation to the data but additional dynamics show that skin under indentation is more complex and very similar to a DPN model.

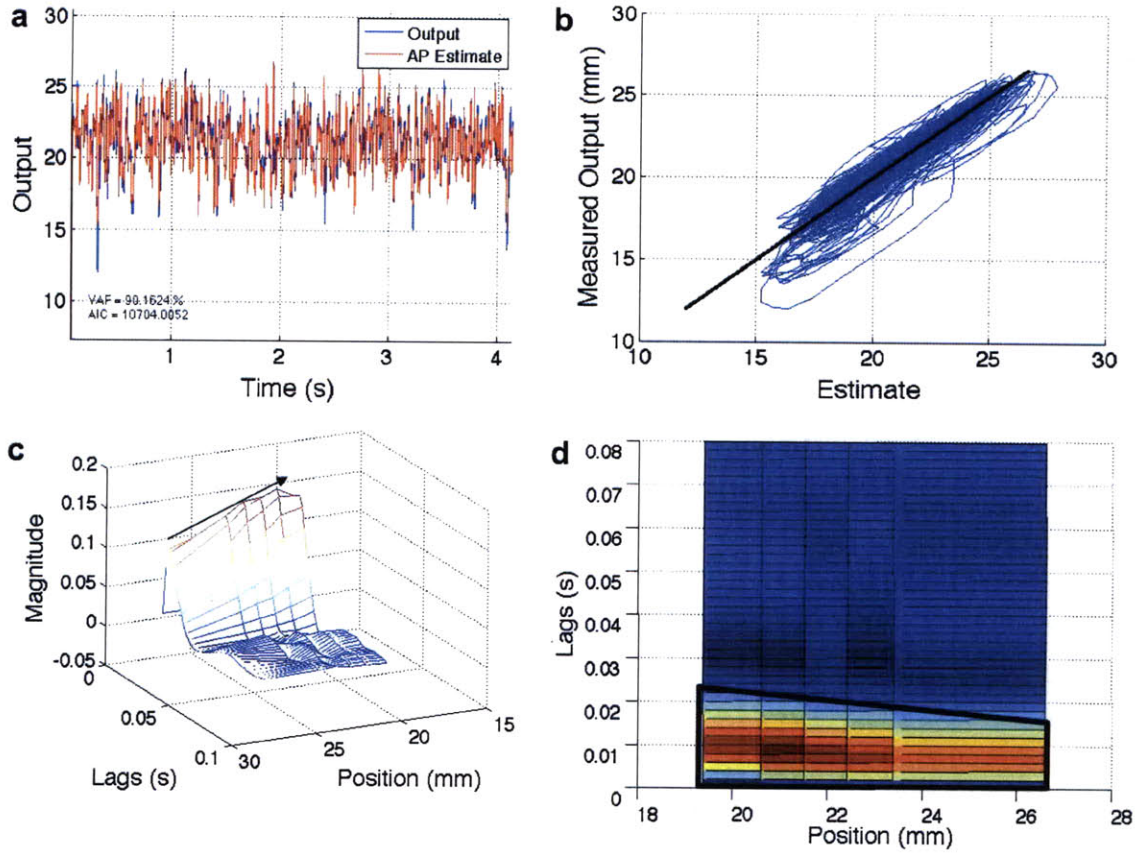


Figure 5.7: The depth dependent simulated DPN model with 6 partitions. The panels are (a) the output estimate, (b) the residual nonlinearity, (c) the impulse response from a three dimensional view and (d) the impulse response from a top view.

5.3.2 Comparison with the Localized Linear Technique

When cross terms become significant, the depth dependent partitions begin to deviate from localized linear solutions. This is shown for two positions on the left arm; the posterior position 40 mm from the wrist and the anterior position 40 mm from the elbow (Figure 5.9). It is important to note that the depth dependent kernels are not exactly the "impulse" responses at different depths although they can begin to approach them for some limit as indicated by Figure 5.1.

In Figure 5.9, the solid points are results from localized linear tests while the hollow points are results from a single four second data set analyzed using depth dependent partitioning. These are plotted on top of a calibration curve that helps indicate the linear regions of the Lorentz force coil. The localized linear tests were conducted both

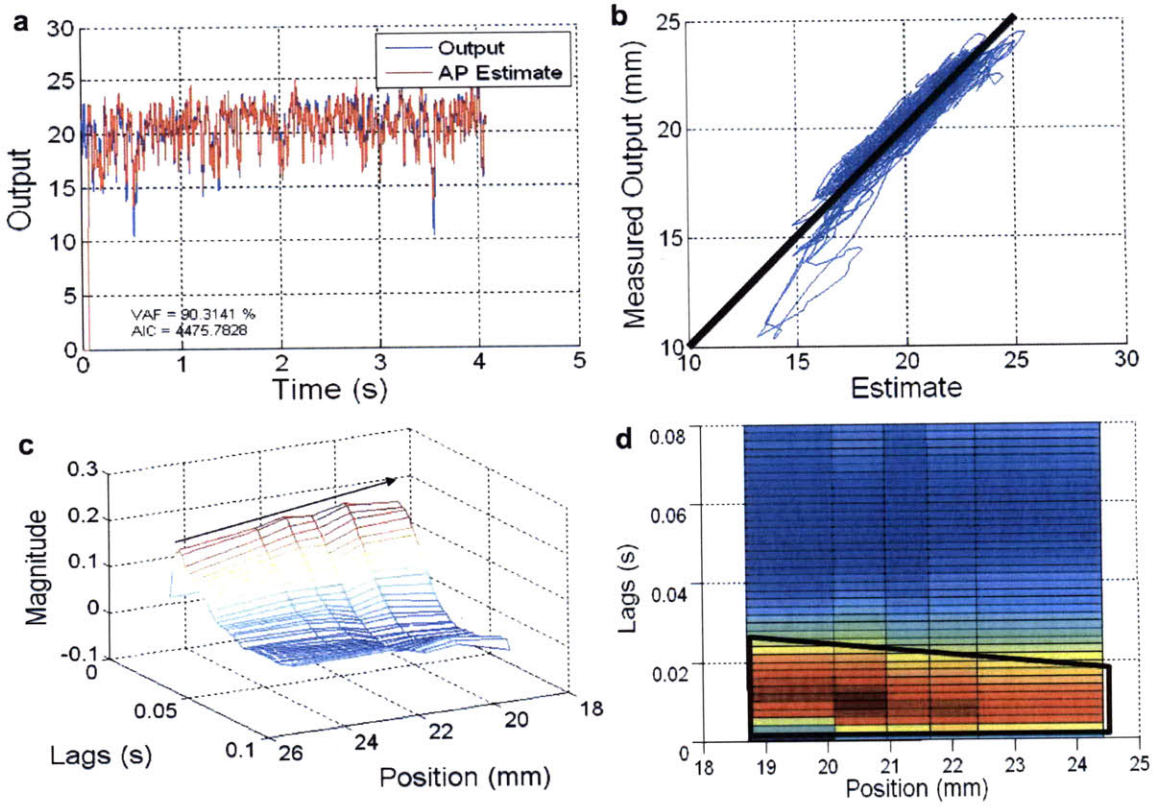


Figure 5.8: The depth dependent experimental results with 6 partitions. The panels are (a) the output estimate, (b) the residual nonlinearity, (c) the impulse response from a three dimensional view and (d) the impulse response from a top view.

pushing into and pulling off the skin while the depth dependent partitioning was only conducted on data from pushing into the skin. The effective mass estimates are very similar while the effective damping and effective spring constant estimates show some more significant differences.

First of all, these two positions on the skin react differently to the forces applied. In general, the posterior position is stiffer for every depth. The results from the depth dependent partition are generally less stiff at the specified depth but show exactly the same trend. The damping, however, is more similar between the localized linear tests and the depth dependent partitions. It is important to note that the trends on the spring, mass, and damping do not follow the same nonlinear pattern as they would for a Wiener static nonlinearity (see Figure 5.1).

The deviation between the localized linear results and the depth dependent par-

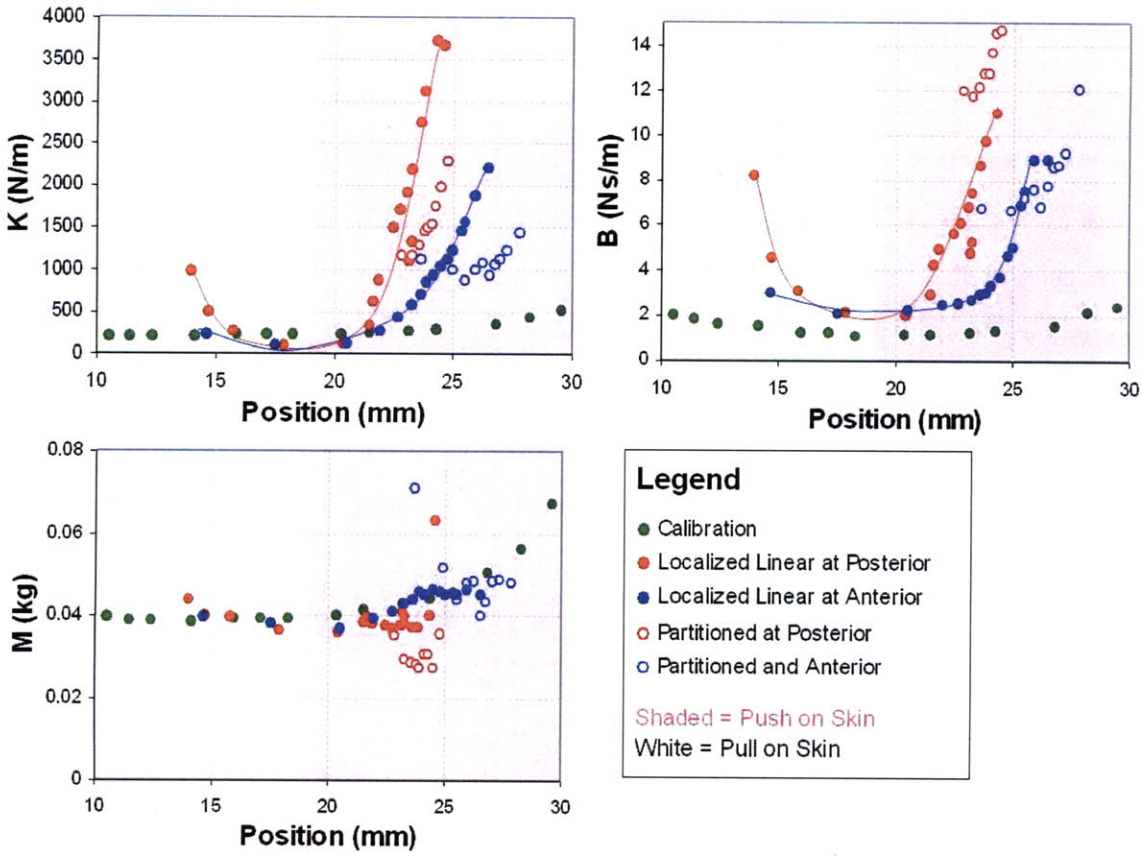


Figure 5.9: The depth dependent results are compared with localized linear results and a calibration reference. The solid lines serve as a guide to the eye.

tition results could have several sources but can be summarized as the difference between a system that has no cross dynamic terms and a system that has significant cross dynamic terms. Note that there is the additional difference that the depth dependent dynamics contains coupling across the zeroth order kernel which tends to produce a small averaging effect across all the kernels.

With this technique, it is possible to create trends that would not only begin to approach the effectiveness of localized linear tests but would do so for a single test with the additional benefit of statistical averaging. There is also an additional benefit in that the depth dependent kernels capture some frequency dependent data from the dynamic cross terms in the nonlinearity.

5.3.3 Frequency Dependence

Since there are indications that dynamic cross terms exist in tests with large ranges, one can expect to find some frequency dependence in the estimates for depth dependent kernels. Figure 5.10 shows data where two different sampling frequencies were used with input cutoff frequencies that are exactly one tenth of the sampling frequency. As indicated before, the mass does not significantly change as one moves between different frequency limit tests. The damping does not seem to change much either since the end of one estimate matches up perfectly with the beginning of another estimate at a different frequency. The solid blue dots, for example, line up with the hollow blue dots and so on.

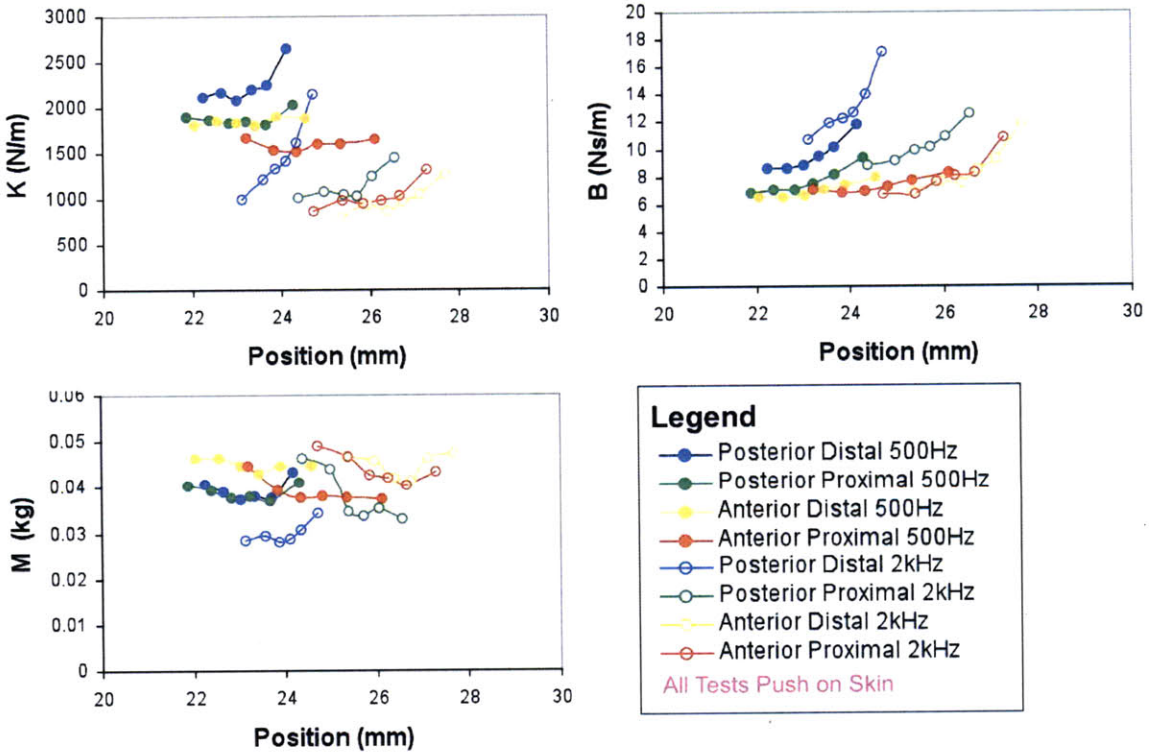


Figure 5.10: As the input frequency is changed, the damping stays the same and the mass stays about invariant. The spring constant, however, decreases as frequency is increased. The sampling frequencies are 500 Hz and 2 kHz and the input cutoff frequencies are 50 Hz and 200 Hz respectively.

The more interesting fact is that the spring constant does not match up in this fashion. In fact, the spring constant appears to shift up as the input frequency is

decreased. This begins to indicate that cross term is a function of input frequency and exhibits itself most heavily in the spring constant estimation. This is in line with results obtained in Figure 5.9. With localized linear testing, there are no cross term dynamics because perturbation about an offset linearizes all the terms. For an input cutoff at 50 Hz, there are fewer cross term dynamics so the spring constant estimate approaches the values of the localized linear testing. For an input cutoff at 200 Hz, the cross terms become more significant and the spring constant decreases.

In general, the depth dependent partitions do a good job of representing the data and do so in an intuitive manner. These depth dependent kernels can be split into representative spring, mass, and damping terms that are a function of depth. The slopes and mean values of these estimates can be compared and be used to distinguish between different locations on the skin. In order to tease out the effect of cross terms and direction dependent terms, other partitioning schemes are necessary.

5.4 Direction Dependent Partitions

A slightly more advanced technique is the direction dependent partitioning method. The idea is that as the probe moves into the skin, it will produce different dynamics than when moving out of the skin. At low displacements near the surface of the skin, one does not expect to see this effect. At deeper positions into the skin where the compression is heavy, one expects to see a difference between the dynamics of pushing and pulling.

Linear system identification and most other nonlinear identification techniques including Volterra kernels assumes no directional dependencies. There are a class of system identification techniques based on characterizing hysteresis [1, 2, 70, 96] that are capable of looking at direction dependent and history dependent dynamics. By partitioning the output, it is also possible to begin looking at the direction dependent dynamics as a function of depth. The construction equation listed below is very

similar to construction equation for the depth dependent case,

$$P_m(n) = \begin{cases} 1 & \text{for } m = 1 \text{ and } n = I+1 \ldots N \\ x(n-j+1) & \text{for } m = 2 \ldots K_{max}(I+1), \\ & \quad n = I+1 \ldots N \text{ and } j = 1 \ldots I+1 \\ & \quad \text{when } L(k) \leq y(n-j+1) \leq L(k+1) \\ & \quad\quad \text{and } sign(y(n-j+1) - y(n-j)) = D \\ & \quad\quad \text{for } k = 1 \ldots K_{max}, \end{cases} \tag{5.3}$$

where D is positive for one direction and negative for the other direction. As with before, the output y that is used to construct the partitions can be low pass filtered to reduce noise. The reconstruction equation is also very similar to the depth dependent case except the number of kernels is twice as many as before since it covers both cases for D,

$$\begin{aligned} \widehat{h}_p(i,j) = A(m) \qquad & \text{for } i = 1 \ldots 2K_{max} \text{ and } j = 1 \ldots I+1 \\ & \text{where } m = j + (i-1)(I+1). \end{aligned} \tag{5.4}$$

With these two equations, it is possible to identify direction dependent dynamics. The results of this compared to depth dependent solutions is shown in Figure 5.11. The depth dependent dynamics are first collected with 10 partitions and the resulting mass, damping and spring constants are shown. The VAF for this estimate is relatively good at 91.9 %.

Next, the direction dependent dynamic equations are used. The dynamics for one direction are shown in a different color from the dynamics in the opposite direction. There are also 10 total partitions but 5 of those partitions assess the dynamics going into the skin and the other 5 partitions assess the dynamics coming out of the skin. The VAF of this system drops slightly to 91.2 % but does not change significantly. The mass as a function of depth does not change at all between pushing into the skin and pulling off the skin at the same depth. The estimate for the damping and the spring constant, however, diverge as the probe goes deeper into the skin. This is in

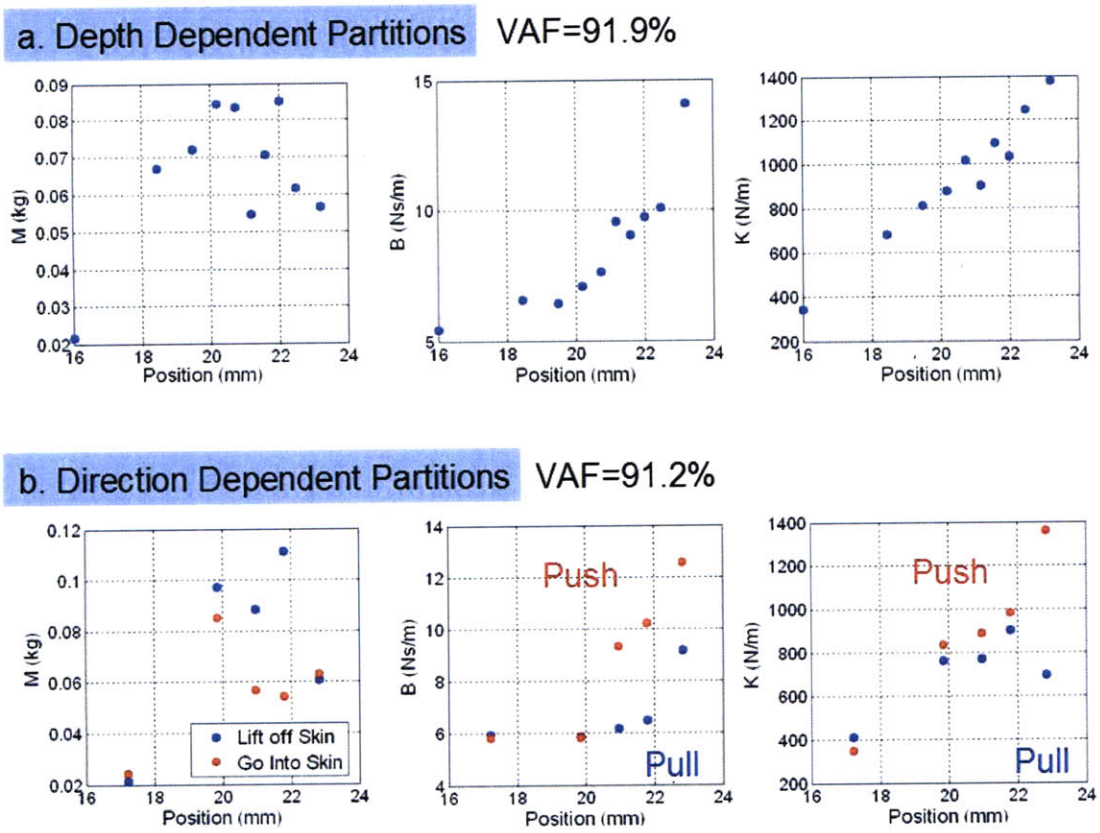


Figure 5.11: Comparison of (a) depth with (b) direction dependent partitions in terms of mass, damping, and spring constant.

line with expectations since it is expected that the skin would have more damping resistance going into the skin than pulling off the skin. It is also expected that pushing into the skin produces higher stiffness estimates than pulling off the skin at the same depth.

With the results from these two different types of partitioned kernels, one is able to tease apart some components of the dynamics using data from a single test. These methods tend to have a fewer number of parameters and high VAF. In addition, they are more interpretable and easier to compare than Volterra kernels. Conceptually, they are capable of measuring more dynamic features than simple static nonlinearities. Therefore, these partitioning techniques lie between the capabilities of the Volterra kernel and the static nonlinearities. Other partitioning schemes can be used to group data in other configurations to obtain more general nonlinear dynamic estimates.

Chapter 6

Input Generation and Real Time System ID

The work presented in earlier chapters focused on off-line system identification techniques where the input generation, the system perturbation, and the system identification are three separate events completed in series. This chapter focuses on the background for input generation and for real-time system identification (where the system perturbation and system identification steps are completed together). This chapter then concludes by combining input generation with system perturbation and system identification with a single real-time algorithm.

6.1 Input Generation

Input generation is the key to stochastic system identification. For some techniques, such as for Wiener kernels, only certain input types will work. An input can be classified by its distribution (Gaussian, uniform, Rayleigh, *etc.*) and by its spectral content (white or colored). Input signals can be generated with several different methods optimized for different situations. The first type of input generation includes techniques which try to create inputs with a pre-specified distribution and spectrum (or autocorrelation).

Another avenue in input generation focuses on producing the desired spectral

content one frequency at a time. These techniques, known as multi-sine techniques [14], combine sine waves at desired frequencies to produce the desired output. With this method, it is possible to completely skip some frequencies that would cause resonances or produce sharp notches and sharp cutoff frequencies without the use of filters.

Another class of techniques use algorithms that attempt to find the perfect input that would satisfy some condition of the output. One of these techniques uses a genetic algorithm to combine different possible inputs using building blocks called "genes". If one input type is successful, its "genes" live on in the next iteration of inputs. This process continues until a perfect input is found [26]. A genetic algorithm has benefits when there are many possible local minima for the selection of the perfect input. When the input to output relationship is more straight-forward, a simple cost function and minimization technique can also be used to find the optimal input.

When dealing with linear systems, the selection of the optimal input is simple and either Gaussian white inputs (or colored Gaussians) or stochastic binary signals are used. The reasons for choosing these types of inputs are discussed in Section 3.1.1. For nonlinear functions, however, it is much more difficult to make a general assessment of the optimal type of input. For the types of nonlinearities found in biological tissues, where the nonlinearity is a function of the depth into the tissue, a good guess for an optimal input is an input that can generate a uniform output. A uniform output would explore the entire range of positions in the nonlinearity and provide an equal number of values for all depths. In order to generate this optimal input, however, the system nonlinearity must be known. It is possible to generate one input, assess the output, and then iteratively generate new inputs until the desired output is found. In order to do this, one must first be able to specify the input distribution and the input spectrum (or autocorrelation).

Producing an input of a certain distribution or probability density function (PDF) is a well-known problem that can be solved several ways. One method is to create the desired distribution with closely spaced bins of data and then randomly scrambling the data to produce the desired distribution with a white spectrum. This method

cannot be done in real time. A second method is to generate a uniform input and construct a cumulative distribution function (CDF) of the desired input distribution. By mapping each value of the uniform input to the equivalent value in the desired distribution using the CDF, it is possible to continuously generate values of a certain distribution in real time.

Generating the desired spectrum is also fairly simple. The most common methods for generating inputs are created for Gaussian white functions since these are the most commonly used inputs. These Gaussian white inputs can be manipulated to produce the desired spectral content simply by using low pass filters or band pass filters. If the filters are causal, low pass and linear, then the output of the filter will remain Gaussian and the filter itself will not influence the distribution. If the input to the filter is a uniform distribution, however, the output of the filter will be some other distribution. This is because fast changes of the input, say from one limit of the uniform to the other limit of the uniform, will be filtered out so that the resulting value will be some average output value. It is therefore difficult to jointly specify the PDF and the spectral content. Hunter and Kearney 1983 [48] attempted to address this limitation by developing a stochastic minimization technique to generate inputs of jointly specified PDF and ACF (autocorrelation function).

The idea is to generate the desired input distribution and then assess the autocorrelation after the values have been scrambled. Then two values in this series can be switched and the autocorrelation can be assessed again. If the autocorrelation is closer to the desired autocorrelation, then the change is kept. If not, then another two input data points are selected and the process is repeated.

This algorithm was implemented and the results are shown in Figure 6.1. The first panel on the top left shows two distributions, a uniform distribution (which is the desired distribution) and a Gaussian distribution generated with a common input generation technique used in MATLAB known as "idinput" which simply applies an 8th order butterworth filter to a Gaussian input. The panel to the right shows the power spectral density produced by idinput (black), and the power spectral density achieved by the algorithm (red). The desired cutoff is at 200 Hz.

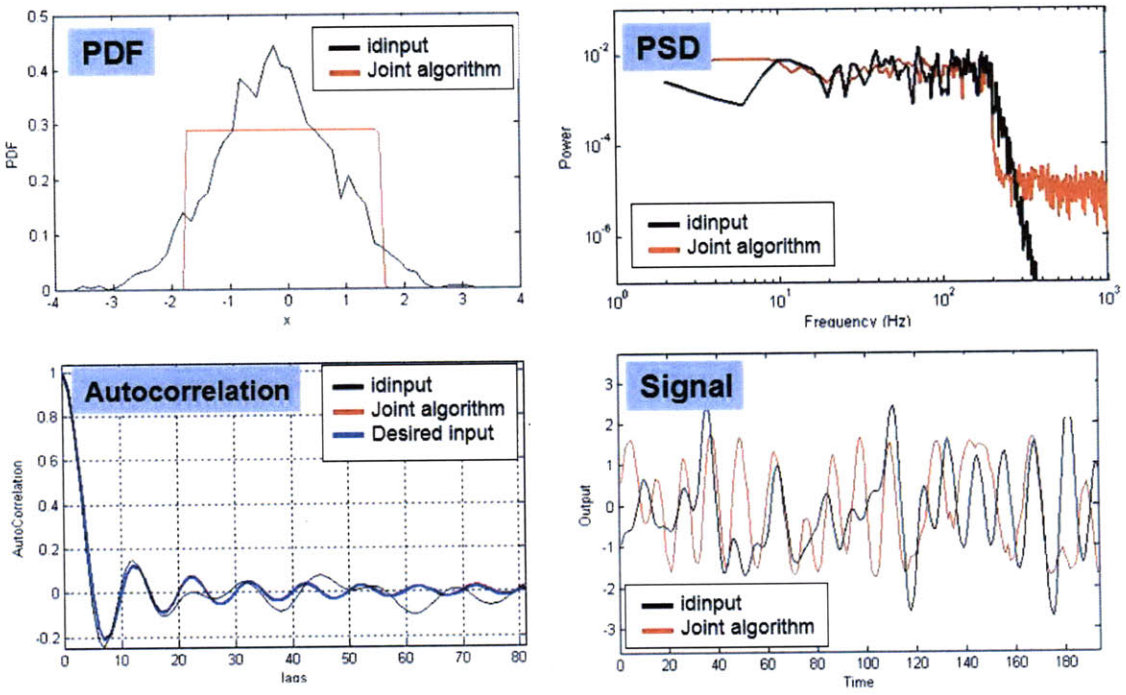


Figure 6.1: Generation of an input with an arbitrary PDF and autocorrelation. The PSD and time domain representation are also shown.

The autocorrelation, which shows the same information as the PSD is shown in the bottom left panel. The desired autocorrelation with the 200 Hz cutoff is shown in blue. The output time series of the signal is shown in the final panel. It is clear that although the idinput algorithm has the desired cutoff frequency, it does not have the desired distribution. The result produced by the algorithm for a uniform distribution has a very unique time domain signal which is completely contained between the limits of the desired distribution.

The generation of this algorithm, however comes at high costs. The swapping technique is fairly inefficient and the time required grows with the length of the input desired. For an input that is 10,000 samples long, over one million swaps are necessary to generate the desired autocorrelation. The most time consuming step is calculating the autocorrelation itself. In order to reduce the computational time, Hunter and Kearney 1983 produced an autocorrelation assessment algorithm that only assesses the autocorrelation for a single swap. This algorithm significantly increases the speed

of the algorithm and is available in Appendix C.4.

The speed of different autocorrelation algorithms is shown in Figure 6.2. A blind autocorrelation which uses the well-known autocorrelation formula takes the longest time to complete as the vector size increases. In fact, this algorithm calculation time goes at about the vector size squared with order $O(n^2)$. The MATLAB "xcorr" operation is more efficient and has an order $O(n^{0.7})$. The swap autocorrelation has a lower overhead than the MATLAB function but has an order $O(n^1)$ which means that it is at a disadvantage at very large vector sizes. However, with this particular problem, it is not necessary to compute the entire autocorrelation. In fact, the autocorrelation can be limited to only the first few values. By fixing the length, the time required for each calculation does not increase with the size of the vector and remains constant. This has clear benefits over other algorithms in terms of overhead and in terms of growth as a function of vector size. With this improvement, it takes 1800 seconds to generate a single input with the desired qualities using a 2.4 GHz Intel Core 2 Duo processor with 2 GB Ram.

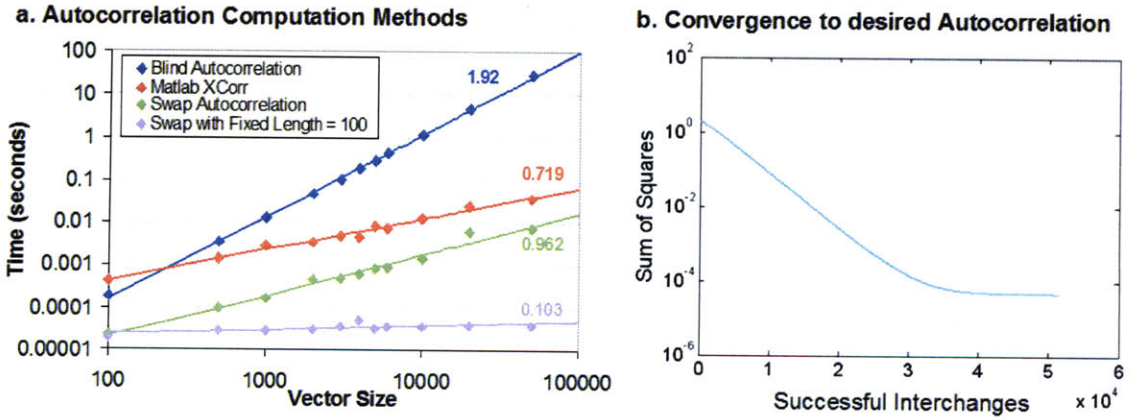


Figure 6.2: (a) Computational speeds of different autocorrelation methods and (b) the leveling of the sum of squares (convergence to a minimum) of the output.

Although this swapping behavior can be used to achieve any autocorrelation, there is a limit to how good the swapping procedure can be. If the bins of the PDF are infinitely small, it should be able to achieve an autocorrelation to finer detail than if the bins of the PDF are very large. Therefore, for any finite size bin on the PDF, the benefits of additional swaps will eventually disappear. This is shown in

Figure 6.2b. As the number of successful interchanges increases, the error measured by the sum of squares decreases until it reaches some limit. At this point, it is no longer beneficial to make additional swaps since the incremental benefit of each swap decreases significantly.

Using this algorithm, a series of uniform inputs were generated for nonlinear system identification in the open loop configuration. If closed loop input generation is desired, a different paradigm is necessary since this method is time consuming and cannot be done in real time.

6.2 Real Time System Identification

Adaptive and recursive algorithms can identify a system in real time. Each additional data point obtained can be be used to modify the system impulse response immediately. These techniques are valuable in situations where the system changes or degrades over time [72]. With recursive least squares, the additional data point is used to optimally improve the overall estimate and information from all the data points is kept in the estimate. With adaptive least squares, some of the prior information is forgotten and the impulse response better reflects the most recent information. Adaptive and recursive least squares algorithms are commonly used and multiple derivations exist and are optimized for different situations [74]

The equations used to calculate the RLS algorithm are as follows, for all the data points collected from n = 1 to N. First, a matrix T is defined with a vector length equal to the memory length M. The input to the system at any time is $x(n)$,

$$T = [x(n) \quad 0 \quad 0 \quad \ldots \quad 0]. \tag{6.1}$$

Next, the error and Kalman gain are calculated with the variable V_n that is initialized as an M by M identity matrix with a diagonal value that is at least two orders of magnitude larger than the variance of the input. This helps with convergence. The discrete impulse response $\widehat{h}(m)$ is initialized as a vector of zeros of length M. If

initialized with values that more closely correspond to the expected impulse response, the algorithm converges faster. Since the output of the system is $y(n)$, the error is calculated as the difference between the output and the expected output. Then, the Kalman gain can be calculated,

$$e_n = y(n) - T^T h_n(m) \qquad \text{for } m = 1 \ldots M, \tag{6.2}$$

$$K_n = \frac{V_n T}{\lambda + T^T V_n T}. \tag{6.3}$$

The variable λ_r is the forgetting factor that allows the system to "forget" older inputs. If λ_r is set to 1, then the calculation never forgets any of the input information. It was found that a value for λ_r of 0.95 to 1 was necessary to achieve good estimations. The next step is to update the V_n using the following relation,

$$V_{n+1} = \frac{V_n - K_n T^T V_n}{\lambda_r}. \tag{6.4}$$

In this algorithm, the impulse response is constantly being updated using the Kalman gain and the estimate error information gained from each additional data point such that,

$$\widehat{h}_{n+1}(m) = \widehat{h}_n(m) + K_n e_n \qquad \text{for } m = 1 \ldots M. \tag{6.5}$$

It is obvious from the algorithm that each data point contributes new information to the construction of the impulse response in real time.

The equations that are used to calculate the ALS algorithms, while very similar, exhibit a few differences. The transformation matrix T stays the same,

$$T = [x(n) \quad 0 \quad 0 \quad \ldots \quad 0]. \tag{6.6}$$

However, the error is calculated using a single error parameter representing the error in the entire impulse response,

$$e_n = y(n) - \sum_{m=1}^{M} \widehat{h}_n(m) T(m). \tag{6.7}$$

The update step occurs for each different position in the impulse response. Note that the parameter λ_a represents the change factor (how quickly the impulse response is allowed to adjust) and not the forgetting factor. The larger the factor, the faster the impulse response will alter in the presence of a changing system. The value of λ_a should be tuned with respect to the sampling rate and the input spectrum in order to cover the appropriate dynamics without adjusting to noise instead. After optimization, a value of 10^{-4} or smaller was used for λ_a,

$$\widehat{h}_{n+1}(m) = \widehat{h}_n(m) + \lambda_a e_n T(m) \qquad \text{for } m = 1 \ldots M. \tag{6.8}$$

The RLS algorithm is fairly similar to the LMS algorithm except that it asymptotically approaches the same solution as the LMS algorithm for LTI systems as more data is obtained. The RLS algorithm can also be compared with LMS systems since it has a similar representation structure where the number of parameters in the memory length is the number of parameters necessary to represent the system. As the test length increases, the number of parameters representing the system does not increase but the ability to reject noise gets better.

The ALS algorithm, however, does not have the same mathematical guarantees and uses many more parameters to represent the system. Each new data point that is obtained can be used to update the entire impulse response such that the number of parameters representing the system grows linearly with test length. The advantages include the fact that the system is adaptable and can be used to identify nonlinear systems. The disadvantages are that the system is more susceptible to noise, the algorithm can go unstable if λ_a is too large and the algorithm has no assurance for optimality.

These two real-time identification methods can be compared with LMS for making nonparametric estimates of skin dynamics under indentation. Figure 6.3 shows the linear impulse response and static nonlinearities using LMS, RLS, and ALS algorithms. In this figure, the estimates for the ALS dynamic impulse response are shown as an average of all the estimates over time. Because the dynamics of the ALS

model are changing with time, it is able to account for the changes in the dynamics of the skin as the probe moves in and out of the skin and the looping in the residual nonlinearity plot begins to disappear. Since the looping occurs over longer periods, the dynamic fitting of the ALS algorithm was able to adapt. For smaller changes in the system dynamics, the ALS algorithm was unable to adapt fast enough so there is still looping or noise in the center of the residual nonlinearity plot.

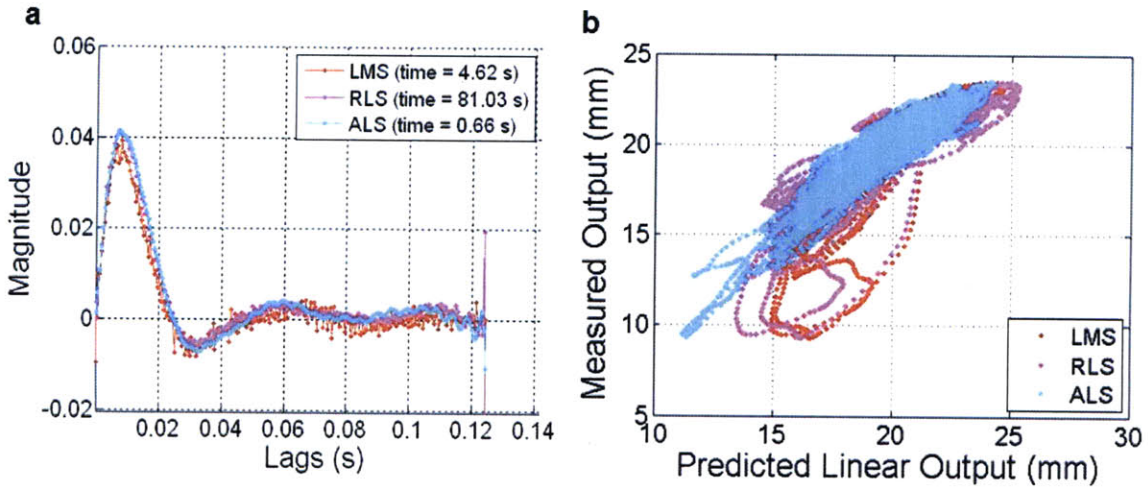


Figure 6.3: Recursive Least Squares and Adaptive Least Squares algorithms applied to obtain (a) the linear impulse response and (b) the residual nonlinearity. Note that the impulse response is mapped in mm vs. mm such that the nonlinearity carries no units and the transfer function carries the units of mm/N. The surface of the skin is located at 10 mm on the actuator for this test.

The calculation time on a 2.4 GHz Intel Core 2 Duo CPU is also displayed on the top corner. The experimental test length was approximately 4 seconds. As the calculation time indicates, the LMS algorithm for this configuration with a memory length of 250 samples has a calculation time that is comparable to the experimental time and is a reasonable time for the clinician to wait for results. The RLS algorithm, on the other hand, takes too long to calculate to be used for real-time system identification [93]. The VAF of the LMS system is 73.8 % while the RLS system has a VAF of 72.0 %. The ALS algorithm is much faster and the calculation time is viable for real-time system identification. The VAF also increases to 86.7 % for the ALS algorithm. The caveat to the ALS algorithm is that it uses many more parameters so the AIC value is two orders of magnitude higher. The AIC for Figure 6.3 are 6.78×10^4

for the LMS algorithm, 6.79×10^4 for the RLS algorithm, and 4.15×10^6 for the ALS algorithm. Additional comparisons for AIC are shown in Figure 8.1. Despite the higher entropy, the ALS algorithm has an additional benefit in that it can describe more of the hidden information within the nonlinearity. Figure 6.4 shows how the impulse response changes as the input and output vary.

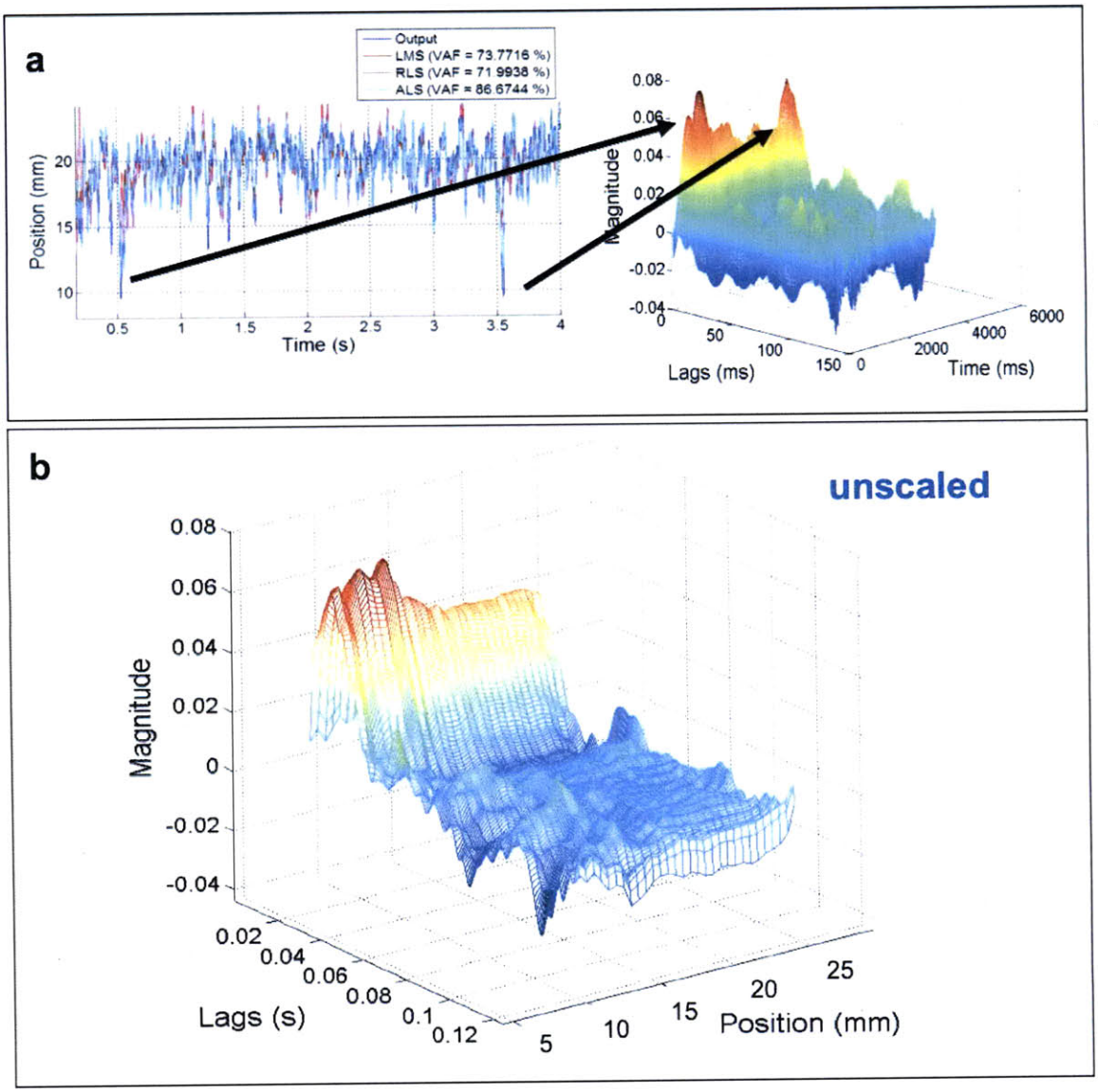


Figure 6.4: ALS system impulse response following changes in the system as a function of depth. (a) The output of the system is shown along with the impulse response as calculated with the ALS algorithm as a function of time. The surface of the skin is located at the 10 mm mark for this test. (b) The impulse response can be reordered to show how it changes as a function of depth. The impulse responses here are unscaled.

As the input varies stochastically, the output also appears to be varying stochastically. At certain points of the output, however, there is a drastic change such that negative peaks are observed. These peaks correspond to peaks in the impulse response from the ALS algorithm. An increase of the peak in the impulse response generally indicates a change in the damping and spring constant. This indicates the the impulse response tends to change dynamically with position.

In order to see these changes more clearly, the impulse responses calculated using the ALS algorithm can be reordered to reflect the change as a function of output position. The reordered plot indicates a pattern in the impulse response as a function of depth. The deeper the probe, the stiffer the spring constant and the higher the damping. Since the system dynamics are changing, the impulse response averaging done in the plot is not entirely accurate. The averaging tends to group all the impulses at the same depth together.

6.3 Input Generation with System Feedback

To properly identify the features of a nonlinear system where the nonlinearity is dependent on the output parameter, it is important that sufficient information is gathered across the output range. There are several ways to achieve the optimal output that can be used to identify the system including (but not limited to):

- Multistep process: Use a reasonable guess for the input, obtain the measured output, and identify the system. Create a new, more optimal input based on the model of the identified system and use this input to obtain a more complete picture. - *This procedure may take several iterations. Generating an input that would produce the optimal output for system identification can be time consuming and would need to be supervised by someone trained in the task.*
- Real time input generation (RTIG) with output PDF feedback: Measure the output of the system in real time and adjust the offset of the input to meet the desired output PDF. - *The optimal system output would be one in which the output PDF is uniformly distributed across a desired range for an input*

frequency of a sufficiently high cutoff. In order to achieve this, it is possible to add a properly filtered input range to an offset which can be controlled to achieve the proper output range. In this strategy, it is not important to identify features of the system other than the output PDF. This procedure is described in Section 6.3.1.

- Real time input generation (RTIG) with output PDF feedback and system identification: Measure and identify the system in real time and adjust the offset of the input to meet the desired output PDF. - *In this procedure, the system impulse response is identified and the control system for determining the input offset is modulated using a self tuning regulator with gain scheduling. This is described in Section 6.3.2*

6.3.1 Output PDF Feedback

One of the most basic configurations for obtaining the desired output PDF in real time is shown in Figure 6.5. First, a Gaussian or uniform white input can be generated and filtered with a controllable high order discrete Butterworth filter to the desired input cutoff. This can then be added to a lower frequency control input that is commanded via a feedback loop. Information about the output distribution can be used to increase or decrease the mean offset of the input in order to explore different locations of the nonlinearity in the output. For the PDF shifting algorithm, it is necessary to know three things about the system: the desire input range, the desired output range, and the minimum system memory length. The desired input and output ranges are used to bound the input and output to ensure that the system does not go beyond physical limits.

The minimum system memory length is used as the feedback rate. Since the input of the system is a stochastic signal with an offset, the output of the system will have a mean value that is directly related to the input offset if the rate over which the the mean is being taken is longer than the minimum system memory length. If the feedback rate is faster than the minimum system memory length, the relationship between the input offset and the output mean begins to break down and the RTIG

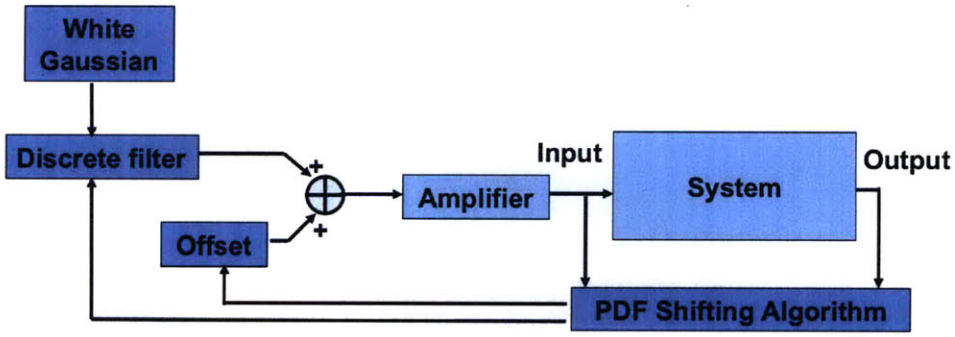


Figure 6.5: Real time input generation (RTIG) scheme with output PDF feedback.

system can become unstable. In addition, to reduce the computational load for real-time input generation, a slower feedback rate is better. On the other hand, it is desirable to keep the test length short so a reasonably high feedback rate is required. The 5 % impulse response settling time as a function of the spring constant and damping for a second order system with a mass of 0.06 kg is shown in Figure 6.6. The settling time is short for intermediate values of damping. As the damping increases for low stiffnesses, the impulse response tends to tail off slowly thereby increasing the settling time. For very low damping, the system tends to oscillate thereby increasing the settling time. The 5 % settling time memory length in seconds for regions in dark blue is about 0.06 seconds (or 120 samples for a sampling rate of 2 kHz). In practice, a minimum memory length of 0.025 seconds (or 50 samples for a sampling rate of 2 kHz) can be used for feedback control in the dark blue regions.

The control system for the PDF shifting algorithm requires a few steps during each feedback loop (once every 50 samples for a sampling rate of 2 kHz):

1. First, split the output range into a number of bins. Output positions that have already been explored can be placed in these bins. The desired output distribution function is a uniform so in order to explore all the bins evenly, the desired output for the next cycle would be the one which coincides with the bin with the least number of entries. This desired location can be called y^*. This idea can also be extended to other probability distributions by simply subtracting the entries from bins already filled with the desired distribution and

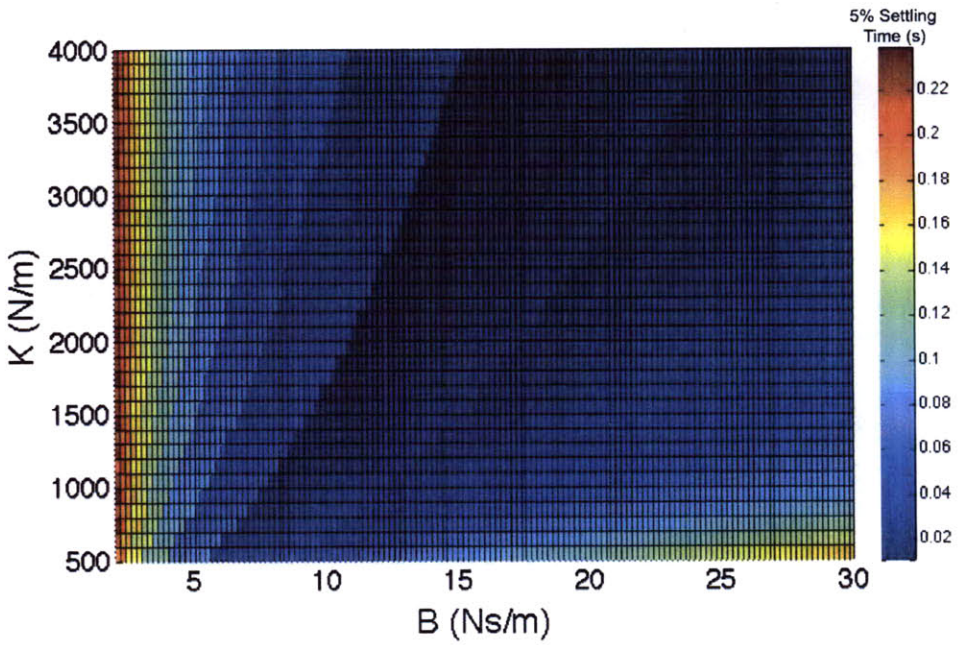


Figure 6.6: The 5 % impulse response settling time as a function of stiffness and damping.

finding the bin with the largest number of entries.

2. Next, determine the current output mean in terms of the number of bins y.
3. Modulate the other input parameters:

   (a) Determine the overall gain to the system: This parameter G_o can determined from a general idea of the system stiffness or memory length.

   (b) Determine the edge modulation: Uniform distributions have sharp edges which requires that the gain be decreased as the output approaches these edges. Depending on the location of desired output location y^* and the current output location y, it is possible to increase or decrease the overall feedback gain using the edge modulation parameter E_m

   (c) Determine the stochastic input $S(t)$: For a desired PDF with sharp edge contrast like a uniform distribution, it is generally a better idea to use a uniform stochastic input. Otherwise, a Gaussian distribution will also work.

   (d) Determine filter $F(\bullet)$ and gain G_s for the stochastic input: Larger gains will obtain more information about the system but can also cause instabil-

ity. The gain G_s can be scaled up to about 20 to 30 % of the total input range and remain stable. The input filter is not necessary for algorithm functionality but can be implemented to clean up the signals.

4. Combine the input with the following formula where $x(t)$ is the new input.

$$x(t) = x_o + G_o E_m(y^* - y) + G_s F(S(t)) \tag{6.9}$$

The first term x_o is the old input offset while the second term is the offset change. The last term is the stochastic input. This configuration with the old input offset included in the output works well when the system input and output are not zero mean. Additional derivative or integral terms can be added to speed up the response of the system although additional speed could cause instability if the output mean can no longer be mapped to the input offset.

By looking at this algorithm, it is clear that it can be improved with an identification step where the the gain G_o is changed depending on the system stiffness. However, to obtain an output with the desired PDF, it is not necessary to identify the system as long as the nonlinearity in the stiffness does not vary heavily. In addition, by not identifying the system, it is possible to obtain computational savings. Figure 6.7 shows the input and output time series simulation results. The output range was constrained to vary between 0 and 6 mm and the input range was allowed to vary from -5 to 25 N.

In the first 0.2 to 0.5 seconds, the algorithm goes outside the described bounds, but soon it centers on a predictable pattern. It sweeps through the output range by going up and down in force. When the input nears the top or bottom edge of the designated output range, the rate of input change plateaus due to the edge modulation algorithm. The input to the system tends to oscillate between the maximum and minimum value in a predictable fashion. The linear system, which has a stiffness of 1000 N/m, has the fastest cycling of about 0.5 seconds per cycle. The Wiener static nonlinearity, which has an average stiffness of about 1200 N/m has a lower cycling rate of about 0.7 seconds. The DPN system with an average stiffness around 2500 N/m has an

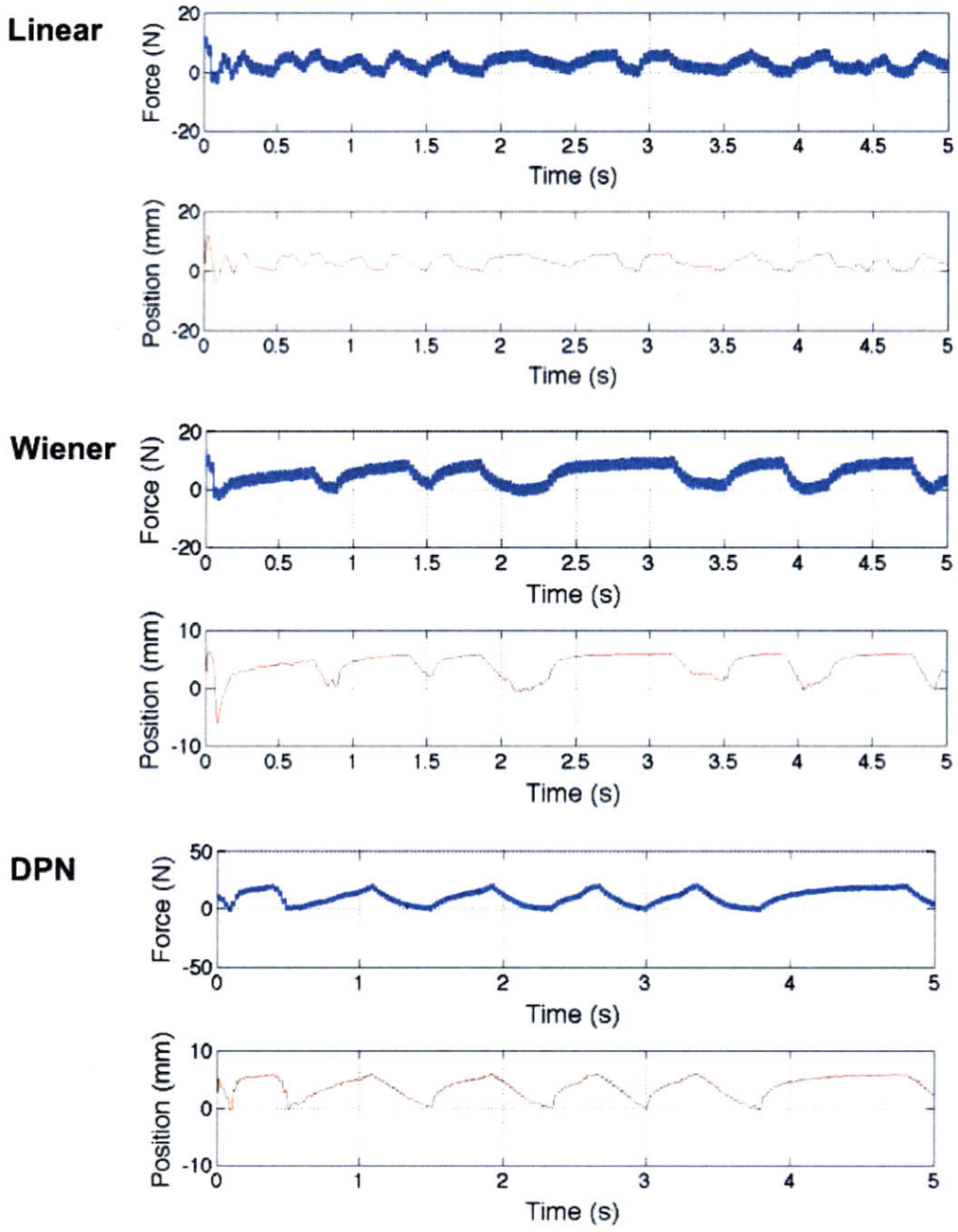


Figure 6.7: Real time input generation with PDF feedback time series results for the input (blue)and output (red). Three different systems, a linear system, a Wiener static nonlinear system, and a DPN system are shown.

even slower cycling rate up to about a second.

The algorithm is controlled by the output PDF which is shown in Figure 6.8 along with the input PDF, the identified average impulse response, and the identified static

nonlinearity. The linear system, shows the ideal output PDF that is very close to a uniform shape between 0 and 6 mm with sharp edge features. The Wiener and DPN systems, however, have shifted probability distribution functions. The main reason for this is that the underlying nonlinearities in these two models have a variation in stiffness for deeper positions. If the bin with the least number of entries is near the stiffer regions, more and more input force is necessary to reach that depth. Since the stiffness is unknown, the algorithm does not adapt to increase gain for these stiffer locations and therefore takes a longer time to reach those regions. Therefore, more entries occur in stiffer regions. Although the resulting distribution is not ideal, it is still possible to control the input and output ranges with this algorithm.

The real time input generation scheme used here has one additional interesting feature. Since the cyclic behavior is slow, the resulting predictions from the static nonlinearity are not as accurate. The static nonlinearity for the Wiener system is scattered and noisy. The static nonlinearity plot for the DPN system, on the other hand, is very narrow and does not exhibit the effects of the cross dynamic terms since the output did not travel across large ranges fast enough.

Despite these drawbacks, the real time input generation without system identification works well for linear systems and systems with small variations in stiffness. The algorithms are also relatively fast and can be completed in 1.3 seconds for a 5 second test making it a viable candidate for real time input generation. Additional details and code can be found in Appendix C.4.2.

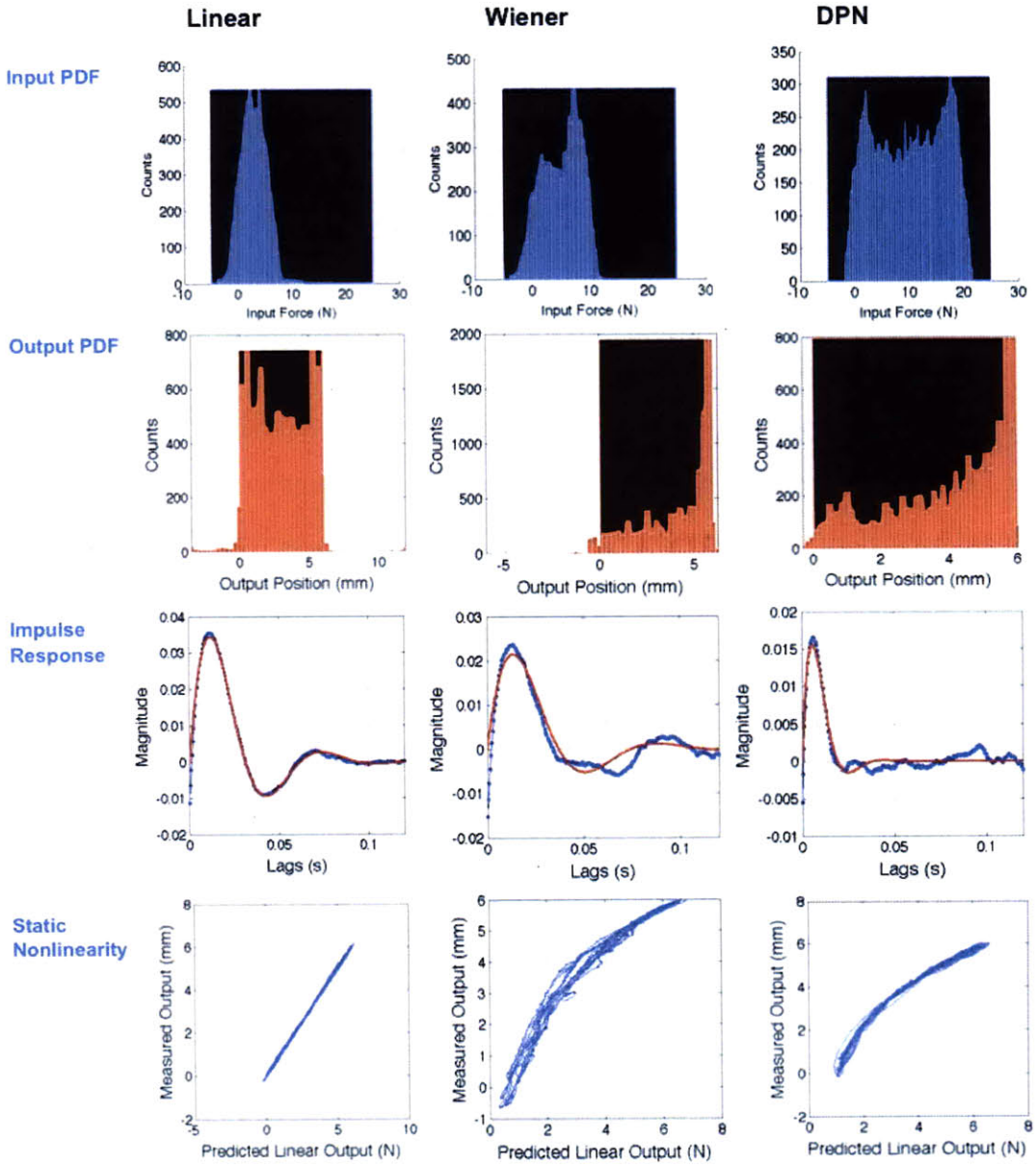


Figure 6.8: Real time input generation with PDF feedback distribution results for the input and output, the identified impulse response, and the identified static nonlinearity. Three different systems, a linear system, a Wiener static nonlinear system, and a DPN system are shown.

6.3.2 Output PDF Feedback with System Identification

When the identification is done in real time, as with the RLS and ALS techniques, the identified system can only approach the optimal solution [93]. If the goal is to identify the system, then an ALS algorithm can be used with some modification in conjunction with a real-time input generation scheme. This idea is shown in Figure 6.9 where the data acquisition and identification steps are connected directly to the input generation step. This methodology is distinct from the idea of using an observer since the observer poles must be faster than the system poles. With this method, the feedback poles must be much slower than the system poles.

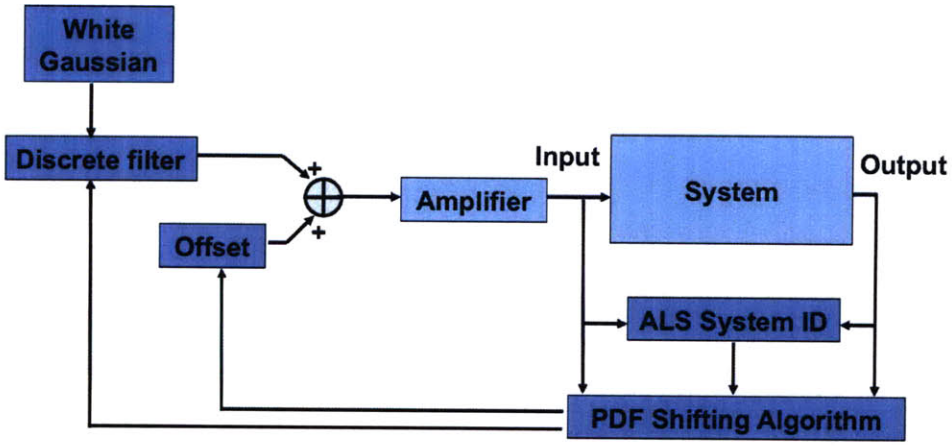


Figure 6.9: Real time input generation (RTIG) scheme with output PDF feedback and system ID.

This scheme is similar to Section 6.3.1 except for the additional system identification step using an ALS algorithm. This algorithm utilizes information about the identified system to determine the optimal input gains. The ALS system ID block functions as follows once every feedback cycle:

1. Add new input and output sequences to the ALS algorithm as described in Section 6.2 using a change factor λ_a of 10^{-5}. Note that including the ALS algorithm may cause the RTIG system to go unstable if the ALS algorithm itself is unstable due to the choice of the identification memory length (which does not have to be the same size as the feedback memory length), impulse

response fitting functions or the change factor. The ALS system will adapt as the system goes to different regions of the output nonlinearity.

2. Fit the most current impulse response to determine the stiffness, damping, and mass and associate these values with the average local bin position. This step can be the most computationally intensive and time consuming and will increases the total computation time to 7.5 seconds for a 5 second test. This, however, can be easily shortened by decreasing the number of parameters over which the fit is conducted or trading of fitting accuracy for speed.
3. Scale the gain G_o and other input parameters with the identified system parameters at the average local bin position. The simplest version of this can be done by scaling the input gain directly with the identified local stiffness. More complex methods like pole placement can also be used. This database of input parameters that depend on the local bin position can be passed on to the PDF shifting algorithm.

The results based on this algorithm are shown in Figure 6.10. These data show an immediate improvement in the cycling time which can be as low os 0.2 seconds for all the systems marking an improvement in the time necessary to identify the nonlinear system. The plateau periods are also shorter and less pronounced. Improvements also appear in the output PDF as shown in Figure 6.11.

When system identification is not used, it was only possible to achieve a uniform PDF for the linear system. With the addition of depth dependent system identification via the ALS algorithm, it was possible to achieve near uniform PDFs for the same Wiener static nonlinearity and DPN systems. In addition, since the cycling speed increased, cross dynamic terms start to become visible in the static nonlinearity plot of the DPN system. The static nonlinearity plot of the Wiener system also becomes better defined.

The values from the ALS can also be used to identify the depth dependent parameters of the system. Figure 6.12 shows how the parameters of the three simulated systems changes with position. The linear system shows no variation in the three parameters as a function of depth. The Wiener system shows how the three parameters

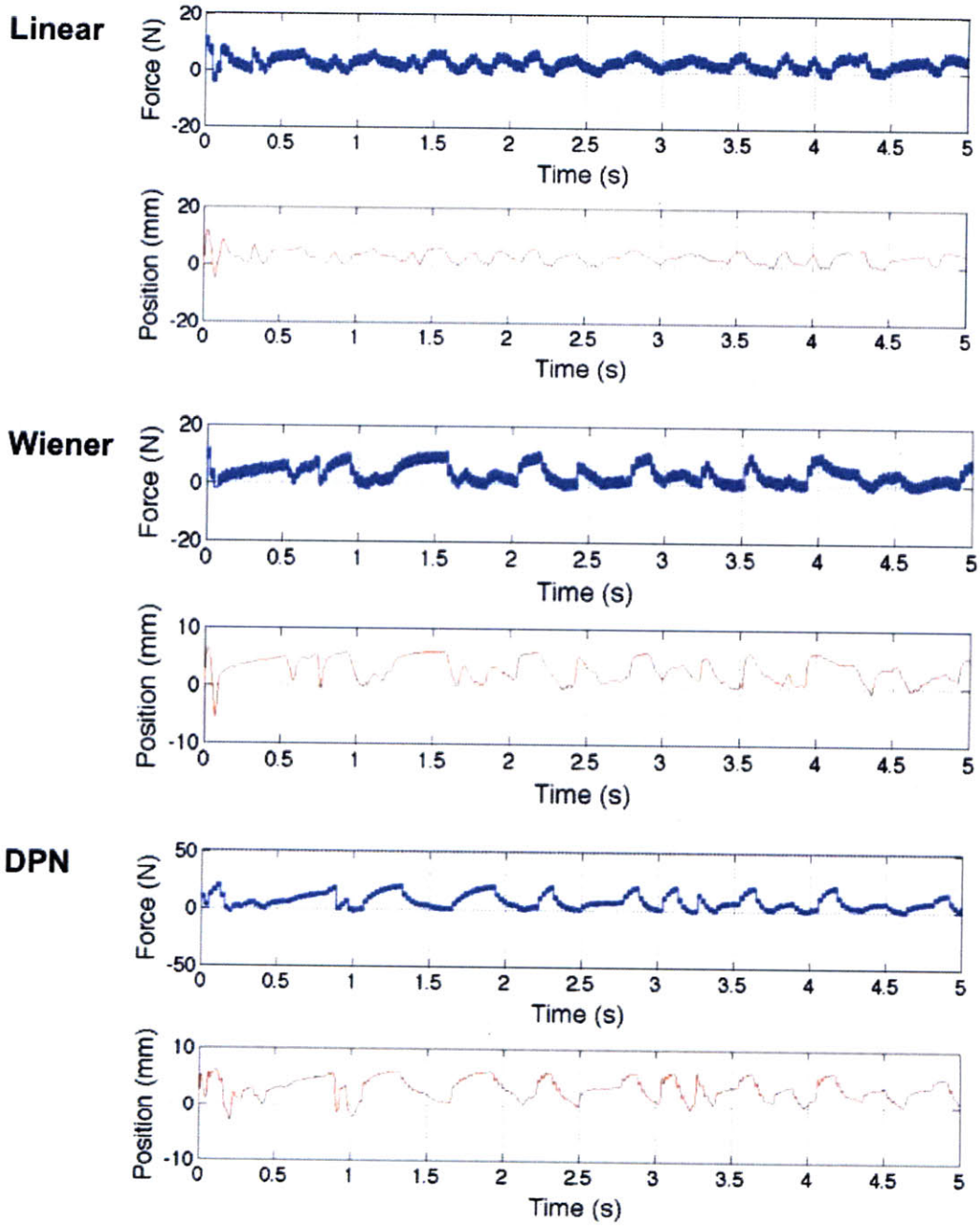


Figure 6.10: Real time input generation with PDF feedback and system ID time series results for the input (blue)and output (red). Three different systems, a linear system, a Wiener static nonlinear system, and a DPN system are shown.

of mass, damping, and spring constant all change with depth in the same fashion. The DPN model shows how the three different parameters can vary differently with depth.

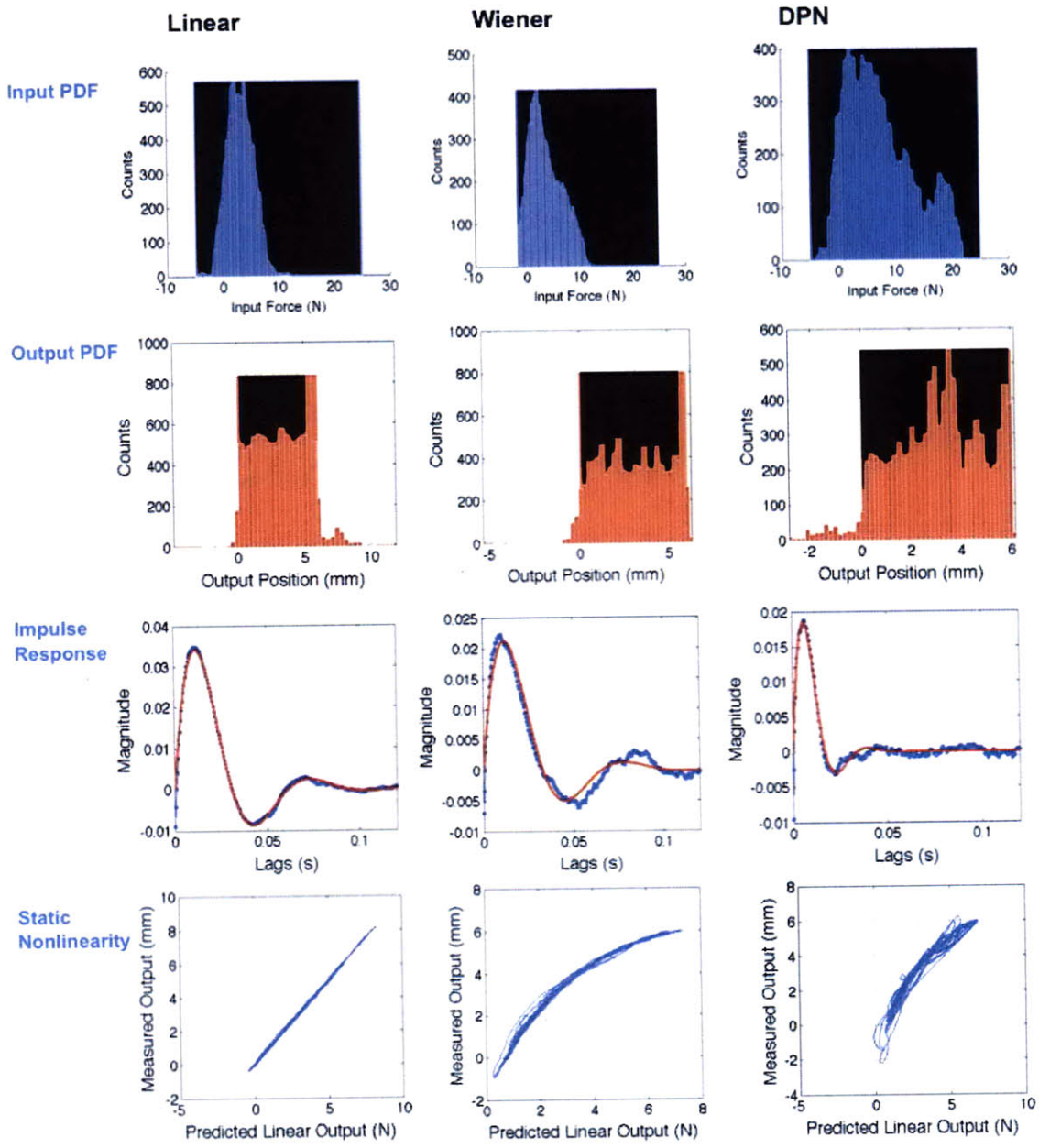


Figure 6.11: Real time input generation with PDF feedback and system ID distribution results for the input and output, the identified impulse response, and the identified static nonlinearity. Three different systems, a linear system, a Wiener static nonlinear system, and a DPN system are shown.

There are two general caveats to using the ALS algorithm for system identification in this case. First, since the stochastic input range is large, a lot of averaging across depths can occur which skews the parameter estimates by averaging across

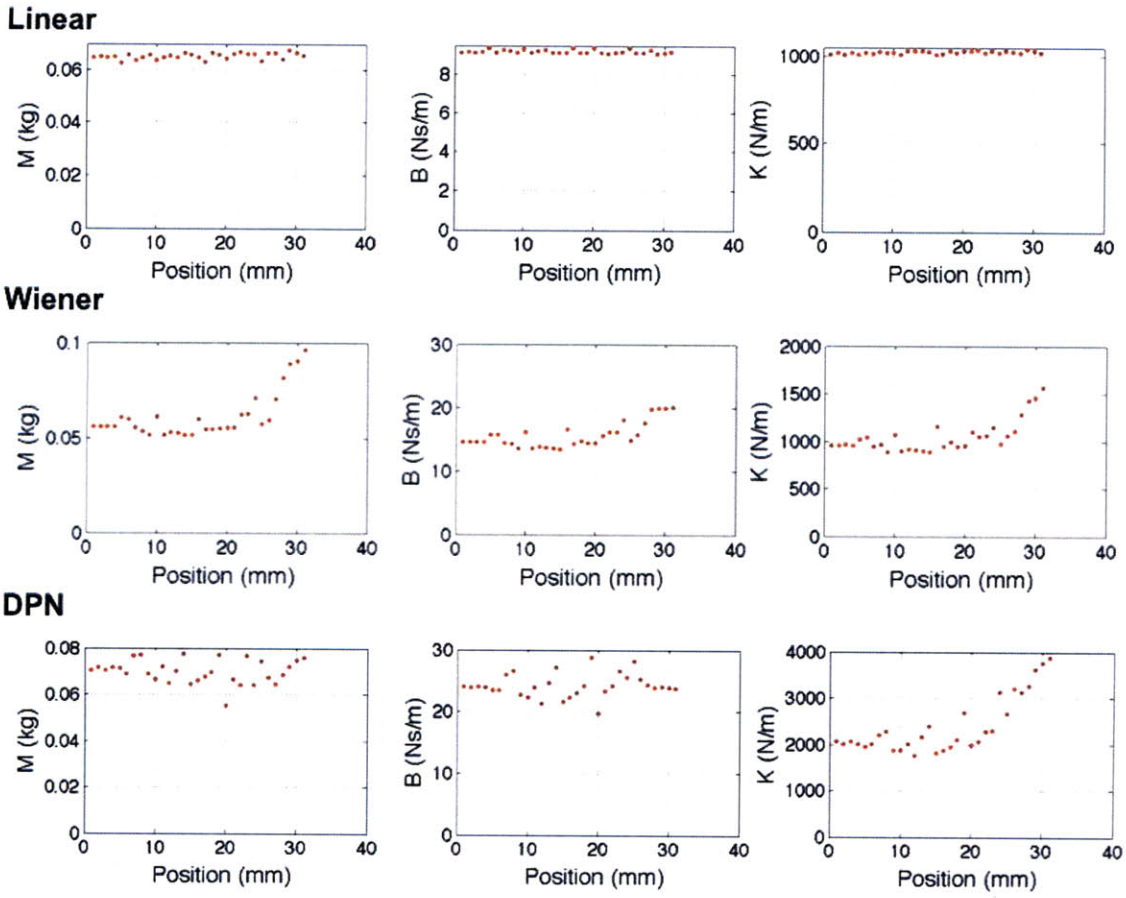


Figure 6.12: Identified parameters from real time input generation with PDF feedback and ALS algorithm. Three different systems, a linear system, a Wiener static nonlinear system, and a DPN system are shown.

depths. The ALS algorithm itself will additionally skew estimates across time. Better estimates can be obtained from the input and output data if an offline system identification technique is used after the data has been gathered. Additional details and code can be found in Appendix C.4.2.

Chapter 7

Skin Studies

In previous chapters, it was shown that the Wiener static nonlinear system identification technique was able to differentiate between different positions on the skin with better specificity than possible with palpation. Since this device can be designed to be more sensitive than human assessments of tissue mechanics, it is therefore possible to use this device to identify key differences in a population. In addition, results from different configurations of the device can be tested and the effectiveness of commercial products can be assessed.

7.1 Indentation Population Studies

The nonlinear system identification techniques covered in previous chapters can be used to characterize skin properties in a population. The procedures and goals of the test are first outlined. Indentation studies on several participants were conducted and repeatability assessed. Then, multivariate statistical techniques were used to look at trends in the data. Analysis of variance (ANOVA) techniques were used to look for statistically significant differences within the data. Subgroups, based on demographic similarities, of the data were also analyzed. These procedures helped identify key parameters that provide indicators for the body mass index (BMI) and help identify similarities across the population.

7.1.1 Test Procedures

The general experimental procedure for any given test involves first checking the area of the skin or tissue for any markings or signs of wear. The device can then be lowered onto the surface of the skin and held in place. When the program is started, it gives a predefined preload (of approximately the mean load of the following test) to the skin for 0.5 seconds to acclimate the tissue. This typically reduces the creep or tissue relaxation during the test. The input signal is then implemented. This signal is constructed such that the tip of the device never leaves the surface of the skin and that the maximum force is less than 10 to 12 N and corresponding displacements on the order of 10 mm or less.

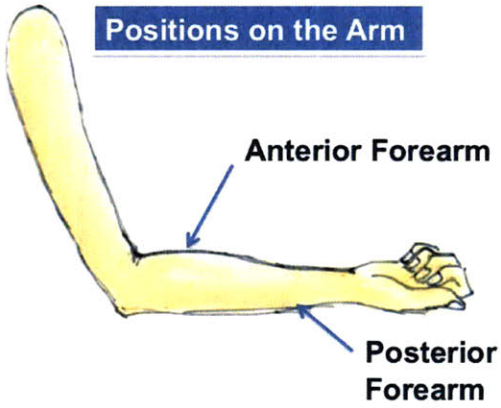


Figure 7.1: Anterior and posterior arm locations for population studies.

In this study, the procedure involved testing four different positions on the forearms of the subjects. Two spots were 40 mm from the wrist on the posterior left and right forearms. The other two spots were on the anterior forearm at a distance of 40 mm from the elbow as shown in Figure 7.1. These spots were chosen because they are easily accessible, flat regions. Demographic information is first obtained and then each spot is marked and a picture is taken of the surrounding area to control for follicle density and skin surface structure. The posterior and anterior positions on the right arm are tested first followed by the left arm. Once the probe on the desktop version of the device is lowered onto the skin, five consecutive tests are conducted using the custom software.

7.1.2 Population Statistics

Sixteen individuals of both sexes and multiple races participated in this study. In order to look at variations in the estimates of skin parameters, the analyzed data is plotted in Figure 7.2. Note the scaled parameter estimates differ from the effective parameter estimates because the scaled parameters are achieved by dividing the impulse response by the DC compliance. This fixes the normalized spring constant at 1000 N/m. The parameters listed are from the Wiener static nonlinearity model. The damping parameter and natural frequency are one possible parameterized representation of the system dynamics while the scaled mass and scaled damping are another possible representation. These measurements are redundant and represent the same information so they should not be considered separate parameters. The depth dependent stiffness in the skin where C_1 is the total compressible stiffness and C_2 is the rate of change in stiffness with depth are also shown.

These data show the mean values and the standard deviations for all estimates for every subject. From this figure, it is clear that the results are fairly repeatable. For most of the subjects, the means of the values estimated from different positions on the skin are statistically very different.

There are a few clear patterns in the individual data. First, the anterior values on both the left and right arms produce different values for the scaled mass, for the natural frequency, and for both of the nonlinear constants. The damping parameter, on the other hand, shows no statistical difference between different positions across the population. The anterior forearm has more scaled damping, a lower natural frequency, as well as a larger compressible depth which are all characteristic of thicker and more compliant tissue.

A better way to look at the data is to look at the averages and standard deviations across all of the subjects. This is shown in Figure 7.3. In this figure, it becomes clear that the anterior positions are statistically different from the posterior positions across five of the six metrics. When comparing the anterior position to the posterior positions, it is expected that the stiffness will be lower which would drive both

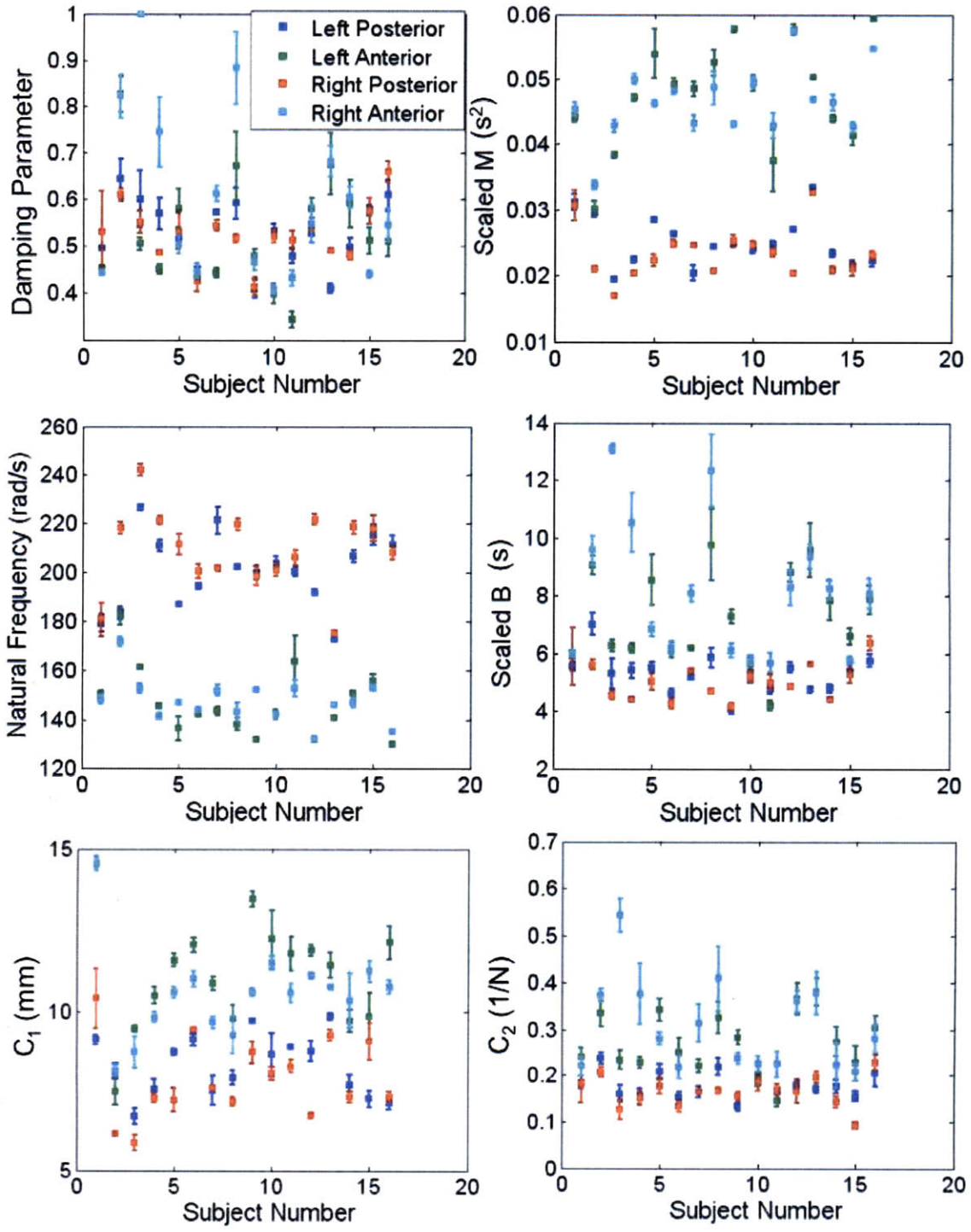


Figure 7.2: Individual subject data showing the damping parameter, the natural frequency, the scaled mass, the scaled damping, and the two nonlinear constants for the four different positions tested.

Table 7.1: Individual coefficients of variation

Position	ζ	w_n	Scaled M	Scaled B	C_1	C_2
Left Posterior	0.040	0.012	0.024	0.039	0.031	0.083
Left Anterior	0.052	0.014	0.028	0.052	0.034	0.092
Right Posterior	0.039	0.014	0.028	0.038	0.030	0.092
Right Anterior	0.043	0.010	0.020	0.048	0.027	0.105
Average	0.044	0.013	0.025	0.044	0.031	0.093

Table 7.2: Population coefficients of variation

Position	ζ	w_n	Scaled M	Scaled B	C_1	C_2
Left Posterior	0.132	0.075	0.156	0.123	0.114	0.154
Left Anterior	0.230	0.095	0.170	0.221	0.153	0.242
Right Posterior	0.117	0.079	0.169	0.124	0.159	0.197
Right Anterior	0.301	0.061	0.117	0.290	0.135	0.313
Average	0.195	0.077	0.153	0.189	0.140	0.226

the scaled mass and scaled damping upwards. It is also expected that the natural frequency will be lower. Since the anterior position has thicker tissue, it is expected that the total compressible thickness will be larger which creates a larger value in the nonlinear constants C_1 and C_2. Not only can this device distinguish between different positions on the skin for a single patient but it can be used to distinguish differences between positions on the skin within a population.

There is no significant difference between the left and right positions. The difference between the right and left hands is expected to be minor although some differences in the musculature is expected for the dominant hand. Since this identification procedure looks at shallower tissue, however, the effect of handedness is expected to be small.

Table 7.1 shows the average coefficient of variation (CV), which is defined as the standard deviation divided by the mean, for individual subjects for each position while Table 7.2 shows the population CV. The CV is a measure of the dispersion of a probability distribution. A lower coefficient of variation indicates a tighter distribution.

From these data, it is clear that the individual CVs are much smaller than the population CVs which indicates that the system makes repeatable measurements

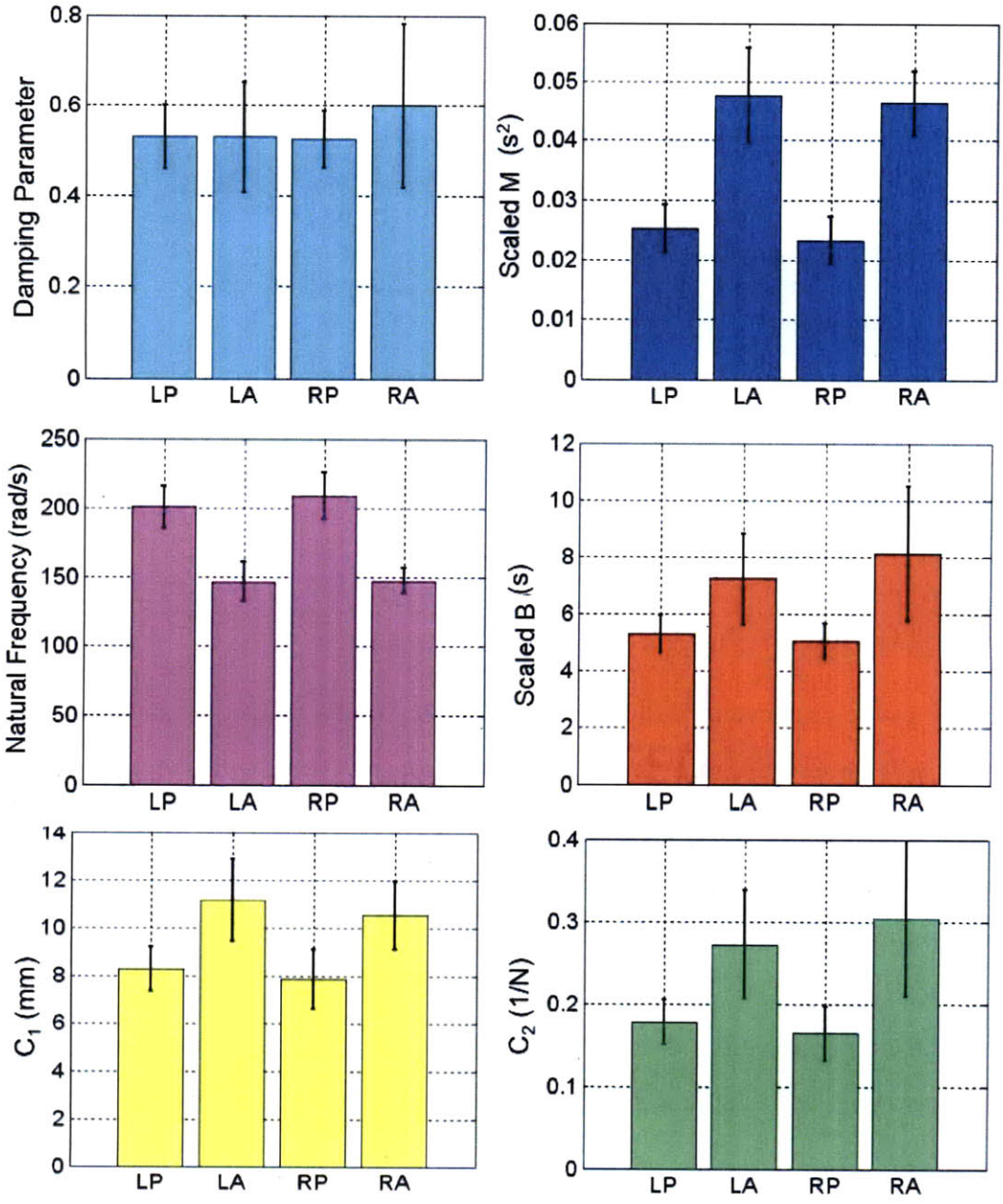


Figure 7.3: Parameter means of the population for Wiener static nonlinearity model showing the damping parameter, the natural frequency, the scaled mass, the scaled damping, and the two nonlinear constants.

which can distinguish individuals from within a population. The CVs obtained for measurements using this device are comparable to those obtained using a Cutometer conducted on a larger population (3 tests each for 450 subjects) [45]. The CVs for the Cutometer study varied from 0.042 to 0.114 for the arm. For similar locations on the

body, the instrument used in this research has comparable (or better) performance when compared with commercial devices.

Despite taking data averages and standard deviations, it is clear that the data collected does not present itself in a random fashion but has some sequential dependency. Since five data sets were obtained for each position, it is also possible to look at how the initial data set differs from the other four data sets. The values of the initial data set were subtracted from the averages of the other four data sets and the results are shown in Figure 7.4. For each of these initial difference plots, the mean offset and the standard deviation for the offset are shown.

The damping parameter, the scaled damping and the nonlinear constant C_1 have statistically significant differences from zero. This means that the first test affects the results of other tests. The change in the nonlinear constant C_1 is especially important because it indicates a change in the compressible depth of the skin. After the skin is tested once, it retains a slight compression (< 1 mm) thereby reducing the compressible depth for future tests. Other tests done with different devices also show this trend which speaks to the malleable nature of skin.

The next factor is to determine if the skin parameters are different across different demographics. To evaluate the effect of demographics on skin parameters, data was collected on age, weight, height, gender, handedness, ethnicity and known skin conditions. Across all of these demographics, there is very little correlation between the measurement of any property and differences in demographics. There is, however a small difference between genders as shown in Figure 7.5. In this figure, the data is plotted as population means for each demographic with standard deviations and data point distributions.

The males in the population tend to have a slightly higher scaled mass for the anterior positions than females as indicated by the circled plot. This may be due to the fact that females in the study are generally smaller than males in the study. In order to calculate the effect of a person's size on their skin properties, it is possible to calculate the body mass index (BMI) which can be calculated by the weight divided by the height squared. A map of the relevant parameters versus the BMI of the

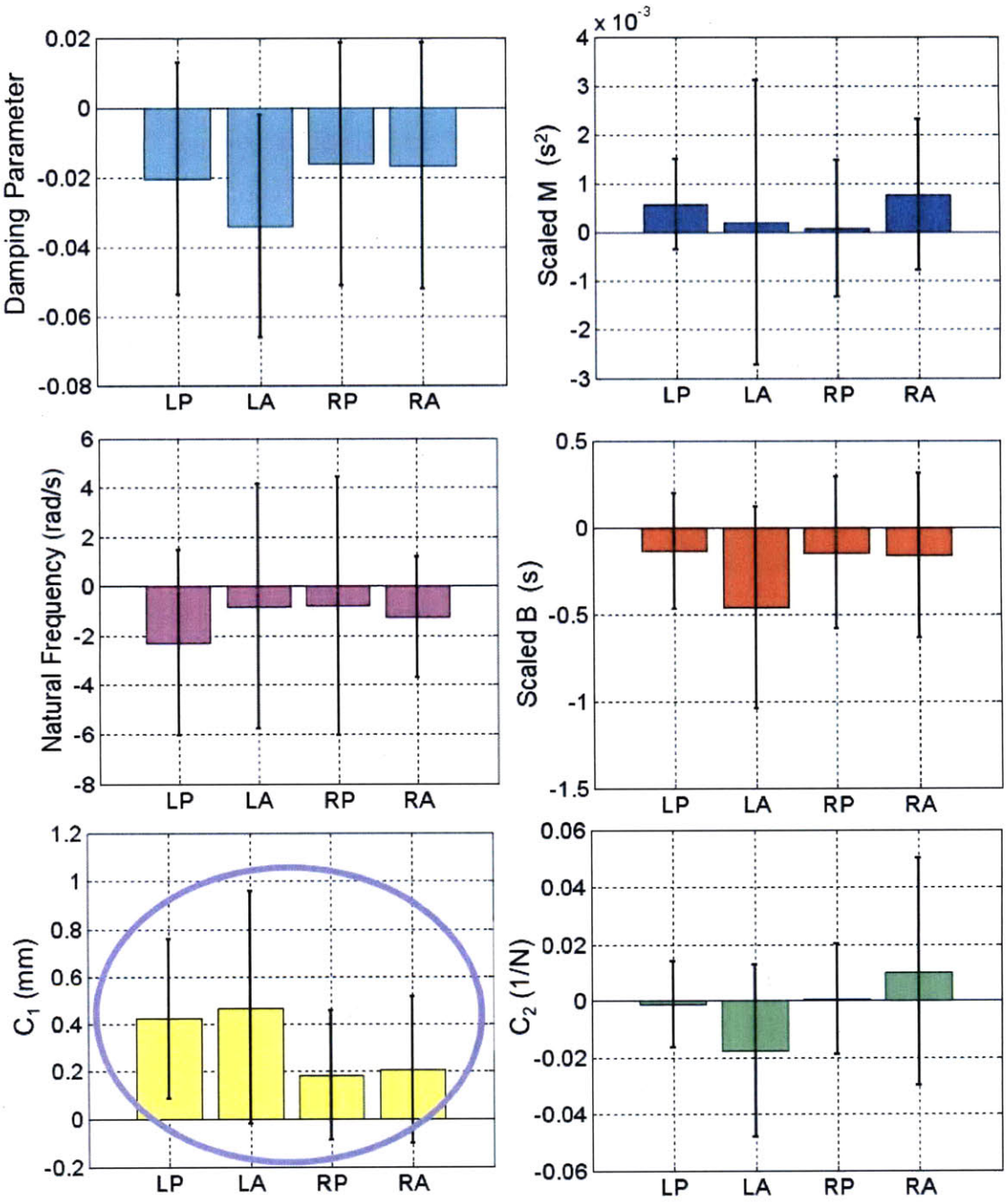


Figure 7.4: Initial differences for Wiener static nonlinearity model showing the damping parameter, the natural frequency, the scaled mass, the scaled damping, and the two nonlinear constants.

subjects is shown in Figure 7.6. This plot includes points for each study participant and lines which are linear fits to the data of each position on the skin.

In these plots, there is generally no trend in the slope of the linear fits indicating

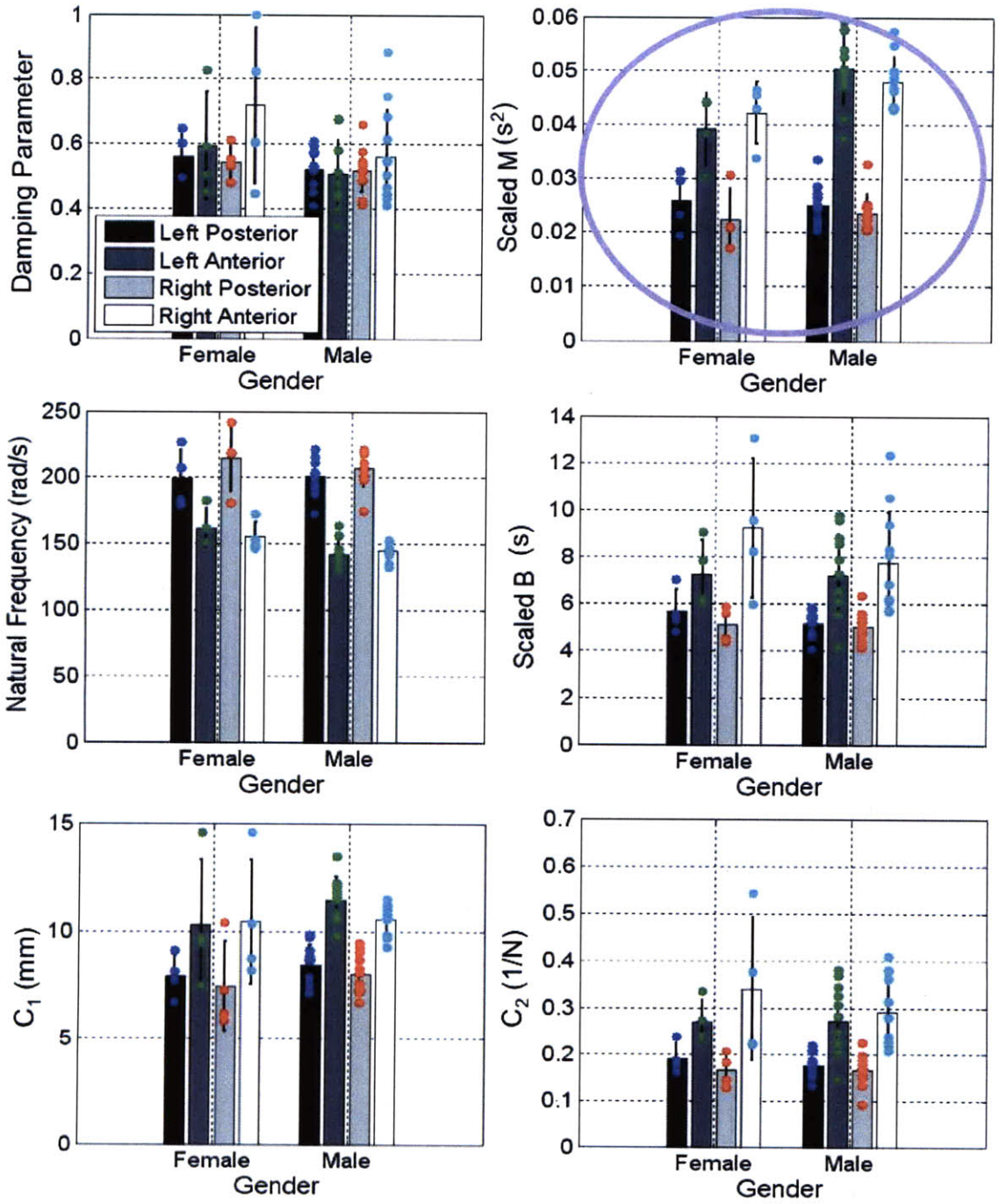


Figure 7.5: Gender differences for Wiener static nonlinearity model showing the damping parameter, the natural frequency, the scaled mass, the scaled damping, and the two nonlinear constants.

no trend as a function of BMI. However, there is an upward trend in the scaled mass and a downward trend in the natural frequency as a function of the BMI for anterior

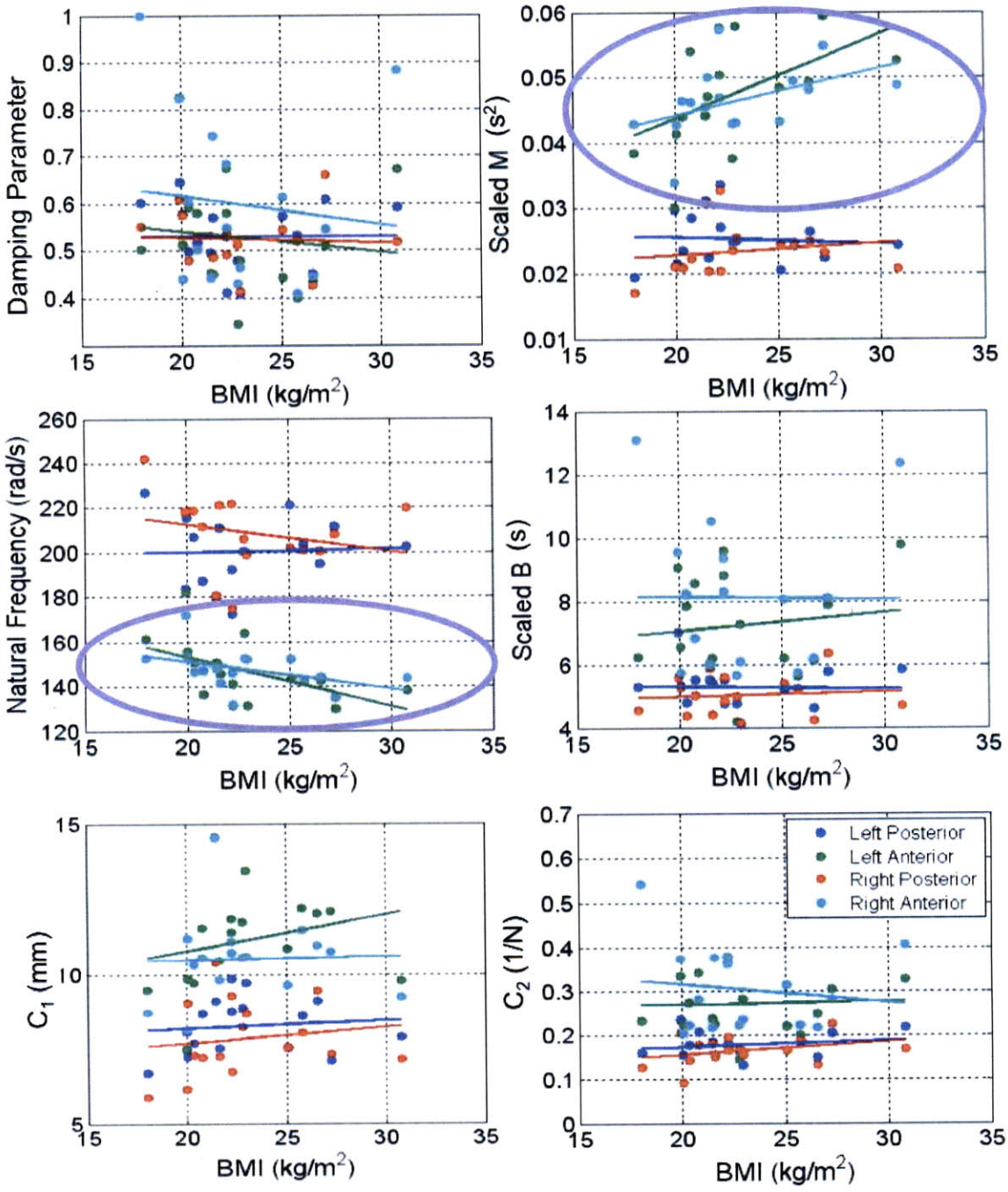


Figure 7.6: Effect of BMI for Wiener static nonlinearity model showing the damping parameter, the natural frequency, the scaled mass, the scaled damping, and the two nonlinear constants.

positions. The anterior positions carry more fat and tissue which would normally decrease the springiness of tissue. Therefore, the scaled mass would increase as a function of BMI for this position on the skin. Correspondingly, a decrease in the

spring constant would decrease the natural frequency of the system. Therefore, this test is actually sensitive enough to sense a difference in the BMI from an indentation test into the skin.

Although plots can be used to assess the instrument and method, it is better to use analysis of variance (ANOVA) methods to determine if any of the differences are statistically significant. Minitab, a software package for analyzing statistics, was used. One-way ANOVA and Generalized Linear Analysis of Variance were conducted on several parameters and the p-values are reported. The lower the p-value, the better the certainty that there is a statistically significant difference between different categories. This is shown in Table 7.3a. In these tables, p-values that are less than 10 % are highlighted. The parameters tested include gender, ethnicity (split into four groups with Caucasians, Asians, African Americans, and Hispanics) and age grouped by people under 25 and people over 25 (the oldest participant in the study was 36 and the youngest was 18).

From the one-way ANOVAs, it is clear that the left anterior position and the right anterior position show statistically significant differences for the gender category that are better than 1 % (which indicates a 99 % level of certainty). There also seems to be a statistically significant difference between ethnicities for the left anterior position but not the right anterior position. For age, there appears to be a statistically significant difference between older and younger individuals for only the right anterior position. The General Linear Analysis of Variance shows a similar trend where the difference in gender is statistically significant for the left anterior position.

The p-values from single independent variable linear regressions are shown in Table 7.3b. In this table, each regressor is used with a constant and p-values under 10 % are highlighted. A statistically significant constant indicates that there is an offset. From the data, there is no statistically measurable difference with age. For the values of height, weight, weight divided by height and BMI, however, there is a statistically measurable difference in the left and right anterior positions in the scaled mass and the natural frequency. This shows the same information as Figure 7.6 but includes a quantitative assessment of the significance of the data. Weight alone seems

(a)

P-Value from One-Way ANOVA

	Left Posterior						Left Anterior						Right Posterior						Right Anterior					
	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2
Gender	0.357	0.828	0.717	0.199	0.365	0.344	0.229	0.009	0.01	0.948	0.259	0.936	0.499	0.421	0.61	0.81	0.449	0.979	0.133	0.053	0.067	0.295	0.88	0.413
Ethnicity	0.115	0.895	0.962	0.137	0.296	0.924	0.81	0.04	0.033	0.686	0.485	0.634	0.249	0.474	0.611	0.307	0.943	0.412	0.397	0.139	0.139	0.553	0.995	0.399
Age	0.545	0.853	0.873	0.512	0.798	0.259	0.446	0.565	0.571	0.684	0.695	0.984	0.884	0.731	0.752	0.737	0.37	0.712	0.342	0.112	0.093	0.548	0.613	0.674

P-Value from General Linear Analysis of Variance

	Left Posterior						Left Anterior						Right Posterior						Right Anterior					
	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2
Gender	0.736	0.626	0.564	0.421	0.813	0.215	0.335	0.075	0.079	0.733	0.559	0.812	0.602	0.947	0.902	0.591	0.513	0.332	0.279	0.368	0.432	0.406	0.915	0.561
Ethnicity	0.186	0.814	0.86	0.238	0.455	0.621	0.963	0.193	0.164	0.666	0.755	0.639	0.295	0.668	0.707	0.297	0.978	0.264	0.631	0.556	0.519	0.658	1	0.484

(b)

P-Value from Linear Regression (Single Independent Variable with Constant)

	Left Posterior						Left Anterior						Right Posterior						Right Anterior					
	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2
Constant	0.001	0	0.003	0.03	0	0.001	0.054	0	0.001	0	0	0.035	0	0	0.001	0.026	0.007	0	0.055	0	0.019	0	0.001	0.049
Age	0.682	0.966	0.965	0.711	0.551	0.86	0.584	0.582	0.547	0.662	0.321	0.861	0.489	0.629	0.649	0.747	0.471	0.126	0.932	0.994	0.996	0.987	0.972	0.834
Constant	0.266	0.044	0	0	0.051	0.255	0.358	0	0	0	0.133	0.741	0.174	0.018	0	0	0.024	0.812	0.556	0	0	0	0.013	0.916
Height	0.605	0.447	0.415	0.979	0.853	0.805	0.994	0.077	0.059	0.411	0.886	0.565	0.644	0.887	0.748	0.787	0.3	0.376	0.899	0.262	0.25	0.671	0.279	0.552
Constant	0	0	0.005	0.059	0	0	0.002	0	0.417	0.01	0	0.011	0	0	0.019	0.107	0	0.01	0.016	0	0.711	0.001	0	0.025
Weight	0.682	0.654	0.489	0.983	0.974	0.47	0.822	0.02	0.016	0.439	0.667	0.685	0.845	0.654	0.853	0.695	0.784	0.192	0.947	0.087	0.086	0.733	0.604	0.949
Constant	0	0	0.002	0.02	0.002	0.003	0.007	0	0.107	0.001	0.002	0.033	0	0	0.006	0.04	0.001	0.048	0.033	0	0.398	0	0	0.041
W/H	0.786	0.682	0.57	0.968	0.543	0.454	0.755	0.02	0.017	0.502	0.543	0.744	0.969	0.528	0.726	0.713	0.982	0.192	0.839	0.076	0.076	0.835	0.768	0.869
Constant	0.004	0.091	0.001	0.005	0.007	0.005	0	0	0.013	0	0.007	0	0.006	0.009	0.002	0.011	0.003	0.001	0	0.003	0.143	0	0.005	0
BMI	0.986	0.89	0.754	0.888	0.383	0.471	0.655	0.037	0.034	0.661	0.383	0.873	0.835	0.368	0.539	0.767	0.596	0.237	0.679	0.085	0.088	0.978	0.907	0.594

(c)

Number of Participants Per Level for Power of 80% and 90% for α = 0.05 using ANOVA

	Left Posterior						Left Anterior						Right Posterior						Right Anterior					
	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2	ζ	ω_n	Scaled M	Scaled B	C_1	C_2
Gender, 2 levels																								
Difference (min)	0.039	1.997	0.0009	0.492	0.51	0.016	0.087	19.46	0.011	0.064	1.142	0.003	0.025	7.938	0.001	0.091	0.569	5E-04	0.158	9.896	0.006	1.469	0.13	0.047
Stdev (max)	0.075	22	0.0055	0.948	0.989	0.033	0.165	14.84	0.007	1.712	3.019	0.073	0.065	25.33	0.006	0.745	2.073	0.036	0.243	11.55	0.006	2.967	2.899	0.153
Power (80%)	66	1907	588	60	61	72	59	11	8	11234	111	7987	106	161	567	1054	210	77136	39	23	17	66	7816	169
Power (90%)	88	2252	786	79	81	96	78	14	10	15039	148	10692	142	215	758	1410	280	1E+05	51	30	23	87	10462	226
Age (under 25, over 25), 2 levels																								
Difference (min)	0.022	1.486	0.0003	0.224	0.127	0.016	0.049	4.217	0.002	0.344	0.353	7E-04	0.005	2.992	7E-04	0.111	0.585	0.006	0.09	7.255	0.005	0.746	0.379	0.021
Stdev (max)	0.071	17.85	0.0048	0.787	1.01	0.032	0.134	15.5	0.009	1.49	2.195	0.061	0.064	17.28	0.004	0.678	1.359	0.029	0.214	8.814	0.006	2.725	1.883	0.11
Power (80%)	157	2267	4020	195	988	64	121	214	319	296	608	1E+05	2870	525	514	587	86	323	91	25	24	211	388	427
Power (90%)	210	3034	5381	261	1323	85	161	285	427	396	814	2E+05	3842	702	688	786	115	431	121	33	32	282	519	571
Ethnicity, 4 levels																								
Difference (min)	0.047	1.372	5E-05	0.402	0.229	0.004	0.023	1.932	0.001	0.293	0.069	0.017	0.041	5.19	4E-04	0.162	0.102	0.007	0.046	5.318	0.004	0.624	0.064	0.034
Stdev (max)	0.07	20.74	0.0052	0.728	1.011	0.038	0.158	14.17	0.007	2.035	2.608	0.084	0.066	22.11	0.005	0.648	1.937	0.045	0.244	11.3	0.006	3.118	2.559	0.137
Power (80%)	49	4985	235846	73	425	1593	1012	1174	1070	1053	30827	563	58	397	3409	350	7901	919	626	100	57	546	34989	354
Power (90%)	64	6475	306559	94	552	2070	1315	1526	1390	1369	40070	732	75	516	4430	455	10269	1194	813	129	65	709	45480	460

Table 7.3: (a) ANOVA, (b) regressions, and (c) statistical power for Wiener static nonlinearity model.

to be a strongly significant estimator of the natural frequency and damping while the height is not as strong an indicator. More test subjects would needed in order to conduct higher order regressions with multiple independent variables. Since a lot of information can already be obtained from single independent variable regressions, there is not much to gain from increasing the number of participants for this particular type of study.

In order to determine the number of test subjects necessary to obtain statistically significant results, a test of the power level at 80 % and 90 % was conducted using an α of 0.05 (corresponding to a p-value of 5 %). In Table 7.3c, the difference between the means of the categories of gender, age, and ethnicity are shown. The minimum difference between the means and the maximum standard deviations are used to calculate the power and the number of participants required to create the desired power level are listed. In this table, participant numbers under 50 per level are highlighted. In order to discern the gender difference for the left anterior position, between 8 and 16 participants are needed per level. For the right anterior position, 17 to 40 participants per level are needed for the gender and age metric. In order to distinguish between ethnicities, a much larger population is needed. In order to get statistically significant results for some metrics, hundreds or thousands of participants are needed. Based on this table, it is clear that additional studies with larger populations would yield very little additional information.

In addition to the least squares assessments, a partial least squares was also conducted for the categories of age, weight, height, BMI and gender. In partial least squares, combinations of the damping parameter, natural frequency, scaled mass, scaled damping, and both nonlinear constants are used. Since it is possible that these choices of output parameters are not the best ones, a partial least squares can be used to indicate if other combinations of these parameters yield statistically significant differences. This is shown in Table 7.4. In this table, there is a slight dependence that appears in the left posterior and right posterior positions. The strongest levels with p-values under 10 % appear in measures of the left anterior position for weight, BMI and gender. This is consistent with earlier data and serves to confirm other estimates.

P values for Partial Least Squares of Skin Parameters on Demographics

	Left Posterior	Left Anterior	Right Posterior	Right Anterior
Age	0.612	0.919	0.168	0.998
Height	0.366	0.198	0.633	0.188
Weight	0.086	0.032	0.789	0.26
BMI	0.197	0.09	0.602	0.296
Gender	0.424	0.09	0.94	0.023

Table 7.4: Partial least squares for Wiener static nonlinearity model.

7.1.3 Subgroup Statistics

The population is diverse and many of the trends could be masked by demographic differences. A subgroup of the test participants, right-handed males between the ages of 18 and 28, were evaluated separately. A plot of the means and standard deviations for two different positions on the left arm are shown in Figure 7.7. This figure shows much smaller standard deviation bars than earlier and statistically significant differences across all four metrics.

Table 7.5: Linear and nonlinear parameters and p-values of the left anterior and posterior forearms of male, right-handed subjects

Quantity	Anterior Forearm	Posterior Forearm	P-value
Scaled Mass (s^2)	0.0256±0.0042	0.0532±0.0048	<0.0005
Scaled Damping (s)	5.02±0.57	7.35±1.49	0.002
Nonlinear Constant C_1 (mm)	8.66±1.03	11.99±0.79	<0.0005
Nonlinear Constant C_2 (1/N)	0.170±0.023	0.287±0.069	0.001

Five measurements were taken at each location using the same stochastic input. Variation between individuals is more than ten times the variation between measurements for the same individual. In Figure 7.7, the anterior position generally has a lower spring constant which boosts the scaled mass and the scaled damping. In addition, the anterior position has a larger compressible depth which boosts estimates of C_1 and C_2. The means, standard deviations and the p-value from an ANOVA study are shown in Table 7.5.

The p-values show that the linear and nonlinear constants are significantly dif-

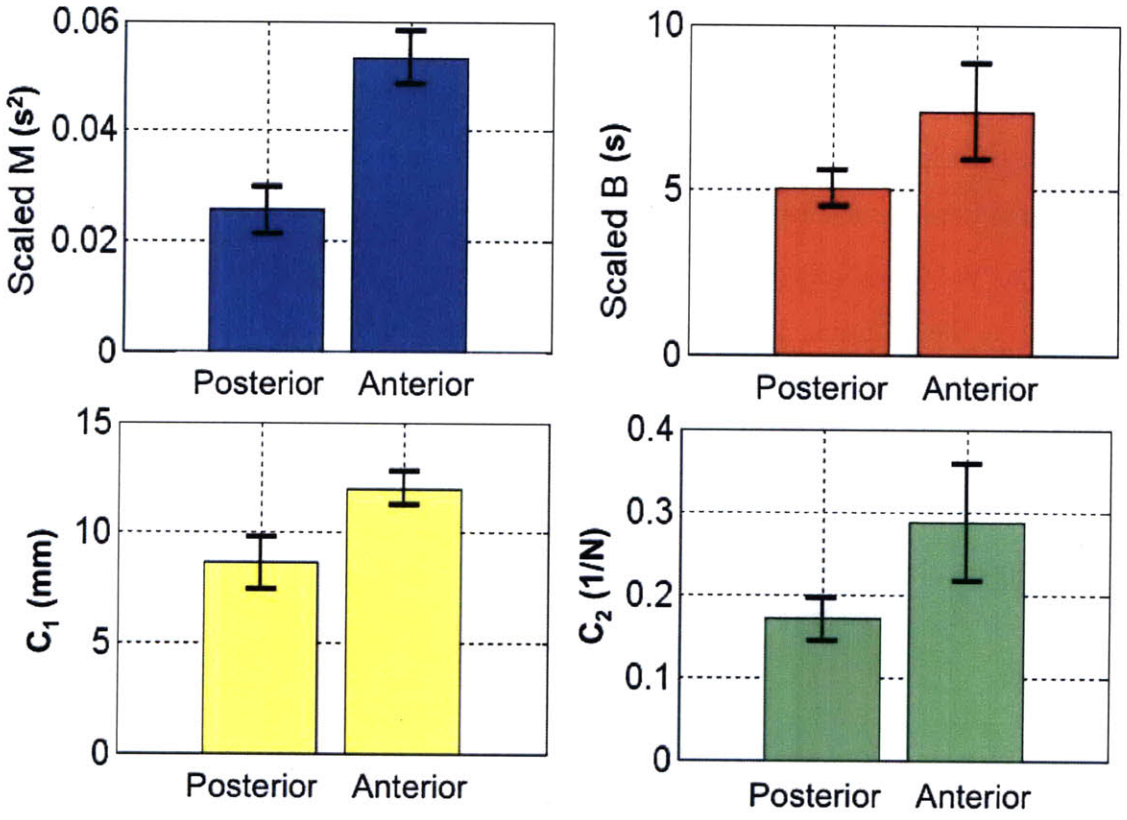


Figure 7.7: Parameter means of the left anterior and posterior forearms of male, right-handed subjects for Wiener static nonlinearity model.

ferent for the two positions demonstrating that the device can easily differentiate between the tissue properties at one site from those at another. In this table, all the metrics are statistically significant with p-values that are much less than 5 %. Subgroup statistics can therefore be used to provide clearer differences between different parameters and can be used to distinguish between changes in tissue properties.

7.1.4 Partitioned Kernel Statistics

A different analysis method can also be used to look at subgroup statistics. In the earlier studies, the Wiener static nonlinearity method was used to assess the data. In this section, the depth dependent partitioning technique is used. The parameters from the depth dependent partitioning were split into a mean value and a slope for the trend which gives exactly 4 different parameters. Higher order fits can be used

to obtain more data metrics. The spring constant and damping were used while the mass was not used due to its inherently low variation.

Figure 7.8 shows how the parameters change for the posterior and anterior positions on the right arm for the same male patients as in the earlier study. In this figure, the difference is statistically significant for the average spring constant and the nonlinearity in the spring constant and the nonlinearity in the damping. The average difference in the damping is not quite as significant but there is a difference in the means.

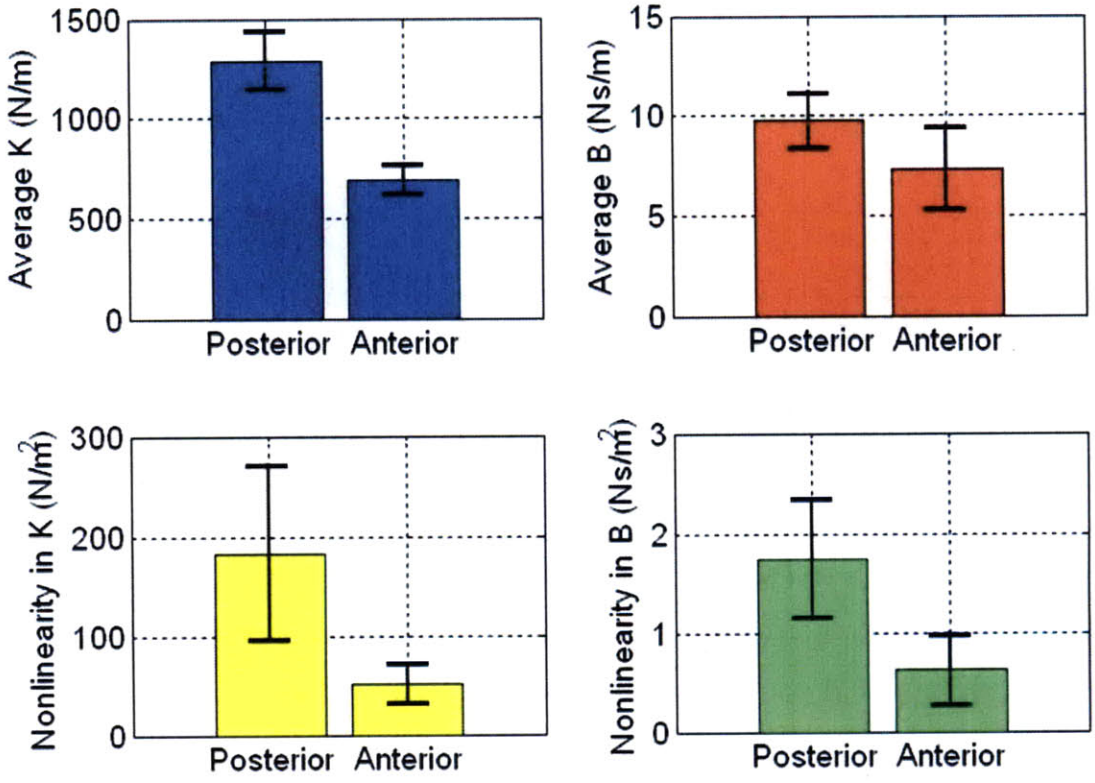


Figure 7.8: Parameter means for male, right-handed subjects for partitioned kernel model.

The spring constant on the anterior position is generally lower and the slope of this spring constant is also lower. This is consistent with conclusions using the other nonlinear system identification method. The conclusion that is new from this method is that the damping parameter is lower and the change in slope of the damping parameter is lower for the anterior position. It was not possible to see this trend

from the earlier analysis because the effect of the nonlinear spring constant bled into the measurement of both the scaled mass and the scaled damping making those parameters not entirely independent.

Based on these studies, it can be concluded that the instrument is capable of measuring differences in demographics and in body mass index through a simple indentation test on different positions of the skin. Not only are the results repeatable but they can be used to distinguish between individuals and have the potential to distinguish differences between skin conditions or to assess the efficacy of commercial products.

7.2 Indentation Comparison

Many synthetic materials are used to simulate human skin; one such material that is used in needle free injection is Acrylamide [92]. In order to assess the similarity of synthetic materials with human skin, indentation tests were conducted on several locations of the skin as indicated in Figure 7.9. For injections into the skin, two of the most common positions are the upper arm 40 mm from the elbow and the anterior proximal lower arm about 40 mm from the elbow. Since these two locations have similar mechanical properties, a stiffer location on the arm, the posterior distal location 40 mm from the wrist, was also chosen to illustrate the range of skin properties.

The acrylamide used in this study consists of 10 %, 20 %, and 30 % concentration by volume in distilled water. Ammonium persulfate (0.075 % by volume) and Tetramethylethylenediamine (2.7 % by volume) were added to catalyze the polymerization of the acrylamide. The acrylamide gels were tested in 6.8 $\times$ 8.5 $\times$ 3.5 cm containers. In addition, Buna (also known as Buna-N or Nitrile) sheet with a thickness of 2.7 mm and latex sheet with a thickness of 87 μm were stretched across a fixture with a 13 mm diameter hole to be tested with indentation techniques. The results of stochastic linear tests below 1 N are show in Figure 7.10 using the 4.3 mm diameter, 0.4 mm corner radius indentation probe. Skin tests are highlighted in blue

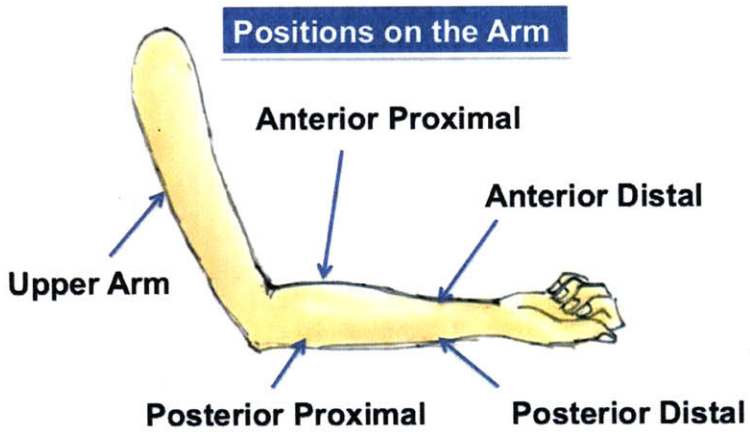


Figure 7.9: Arm locations for indentation comparison studies.

and acrylamide tests are highlighted in red. Each material was tested 10 times with error bars representing standard deviations.

Due to the small force range, the results were relatively linear with high VAFs. As the percentage of acrylamide increased, the stiffness of the material increased with very little change in the mass or damping. When comparing acrylamide with skin, 10 % acrylamide is most similar in stiffness and in damping. Since there is high variation in the stiffness of skin, acrylamide concentrations as high as 30 % can be used to simulate skin stiffness (see stiffness plot in Figure 7.12). If the damping parameter and natural frequency are used to compare the materials, 10 % acrylamide is also the most similar in terms of natural frequency and energy absorption. Nevertheless, acrylamide tends to have very low energy absorption when compared to skin and tends to fracture under larger loads. This quality becomes especially important for needle free injection since the level of local energy absorption determines if the acrylamide will fracture during injection. Therefore 10 % acrylamide is most similar to tissue. The other two materials used in this comparison are more similar to skin in terms terms of energy absorption. The Buna material has a high stiffness and a high level of internal damping. The latex material is most similar to skin at the surface due to its relatively high absorption qualities and low stiffness.

Differences in probe geometry can greatly affect the comparison of materials. As

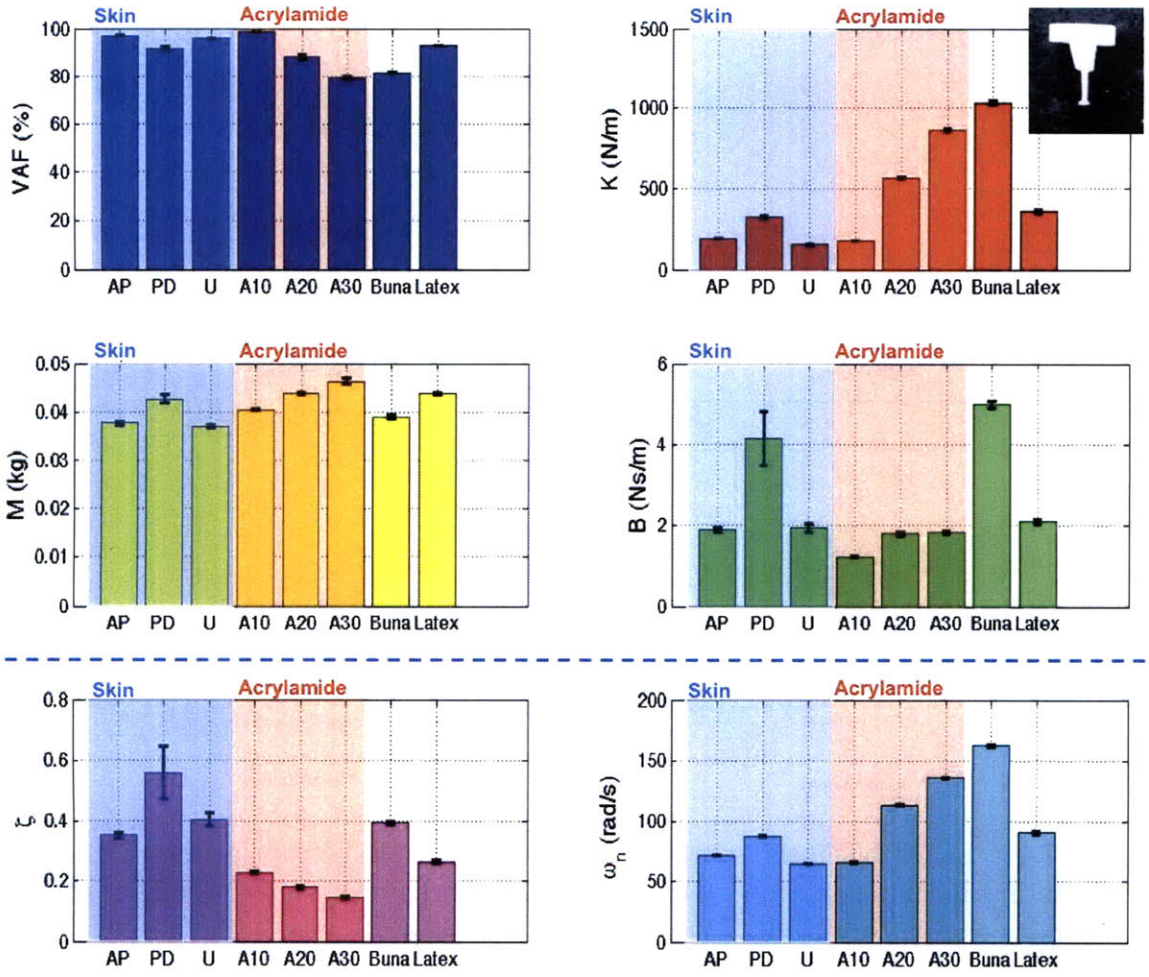


Figure 7.10: Linear parameter comparison of skin and synthetic materials during indentation. Three spots on the arm were tested at the anterior proximal (AP), posterior distal (PD), and upper arm (U) locations. Acrylamide at 10 %, 20 %, and 30 % by volume, 2.7 mm thick buna sheet, and 87 μm thick latex was also tested.

the probe diameter increases, the relative amount of material that undergoes compression increases and the relative amount of material that undergoes extension around the edges decreases. As the corner radius of the probe increases, stress concentrations decrease. In order to illustrate these effects, a more complete localized linear comparison of the materials was also conducted for the smaller 4.3 mm diameter, 0.4 mm corner radius probe (Figure 7.11) and the larger 5 mm diameter, 1.3 mm corner radius probe (Figure 7.12).

In Figures 7.11 and 7.12, the curves obtained for skin is shaped exponentially

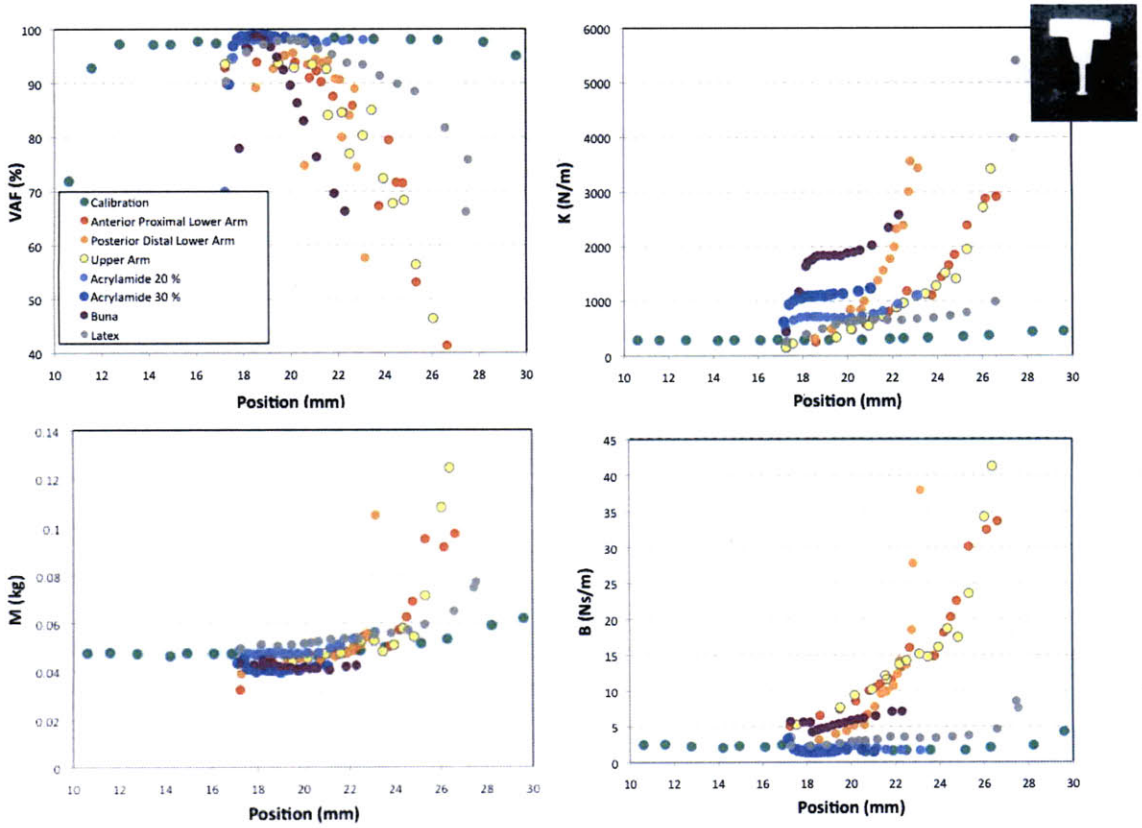


Figure 7.11: Localized linear comparison using the small 4.3 mm diameter 0.4 mm corner radius indentation probe. Due to the small corner radius of the smaller probe, it was not possible to test 10 % acrylamide with localized linear techniques without fracturing the material.

while the curves obtained for the synthetic materials has a more sigmoidal shape. The sigmoidal shape comes mostly from the influence of the material under compression. Materials such as acrylamide and latex have a large region of relative linearity in stiffness and damping. Although synthetic materials can be made to match biological materials in terms of linear properties, it is more difficult to match other nonlinear properties. The synthetic materials in this study tend to be more linear than the biological materials as indicated by the small ranges in the observed stiffness and damping. This is especially clear in Figure 7.11 where the stiffness of the synthetic materials does not change significantly over several millimeters of displacement.

The other techniques described in this work were also used to assess the nonlinear characteristics of biological materials with respect to synthetic materials. Figure

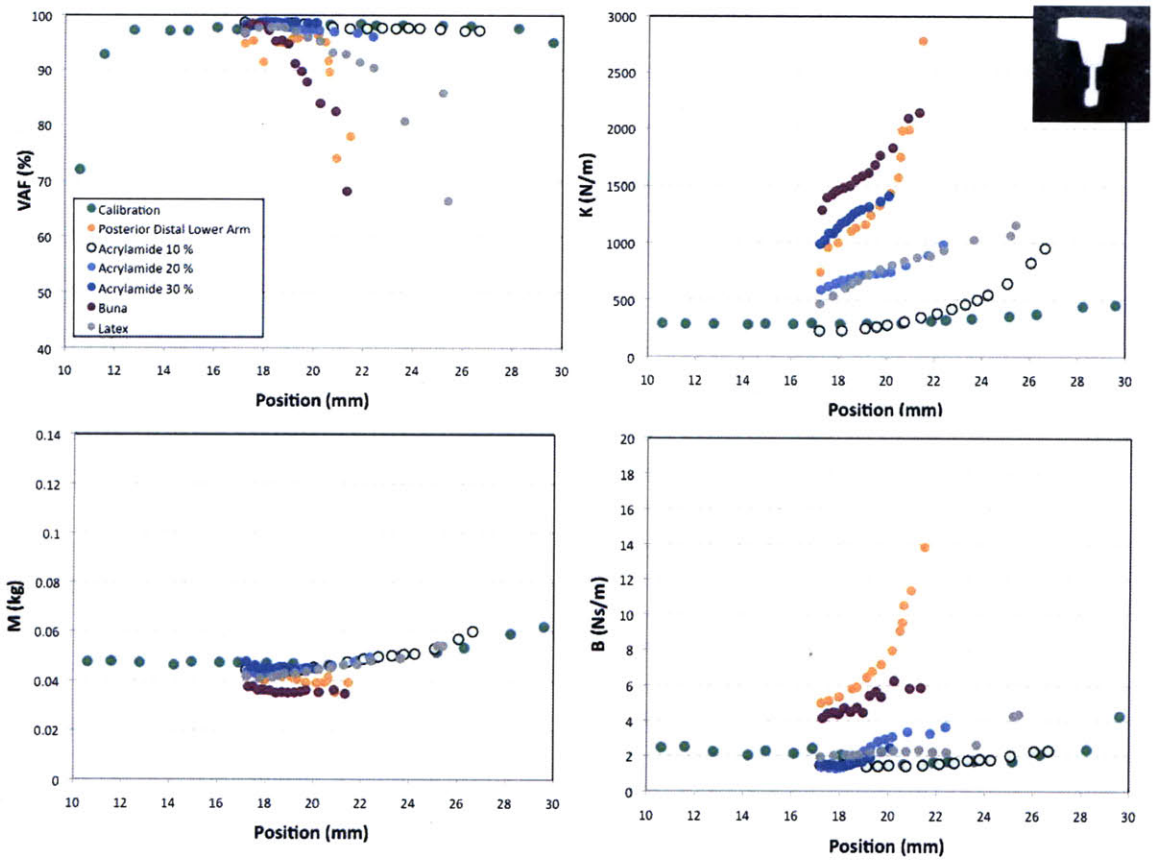


Figure 7.12: Localized linear comparison using the large 5.0 mm diameter 1.3 mm corner radius indentation probe.

7.13 shows the Wiener static nonlinear analysis for the posterior distal forearm, 20 % acrylamide, 30 % acrylamide, and Buna. Skin is significantly nonlinear and can be well-modeled by the static nonlinearity, the Buna is fairly linear and the acrylamides are show evidence of nonlinearity that cannot be modeled by a Wiener static nonlinearity. Since the acrylamide has very little energy absorption, heavy oscillations are observed which decreases the VAF when the memory length is restricted to 250 samples at 2 kHz.

Identification via Volterra kernels is shown in Figure 7.14. The nonlinearity in skin and Buna is well modeled by Volterra kernels while the heavy oscillations in the acrylamide are more difficult to capture for reasonably sized Volterra kernels. This also leads to differences in the shape of the second kernel for acrylamide. The use of Volterra kernels does significantly increases the VAF to 90 % or better for all the

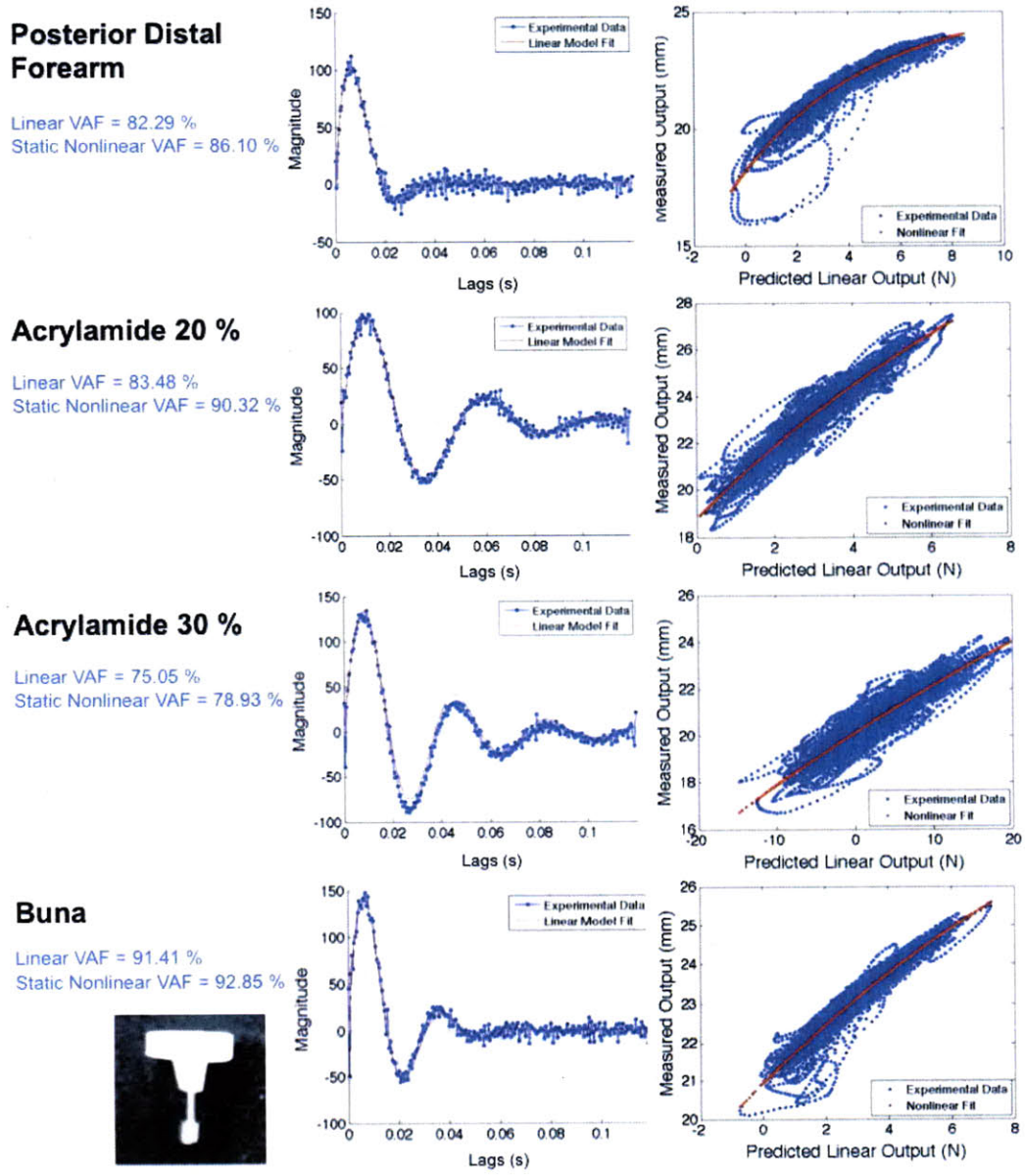


Figure 7.13: Wiener static nonlinearity comparison of skin and synthetic materials using large indentation probe.

materials.

When depth-dependent partitions are used as in Figure 7.15, the VAF increases even though the number of parameters in the fit decreases. This shows that the depth-dependent partitioning is an effective representation for this type of nonlinearity. The skin shows more variation in the peaks of the depth-dependent kernels while the depth dependent kernels for the other materials are relatively invariant. This indicates that

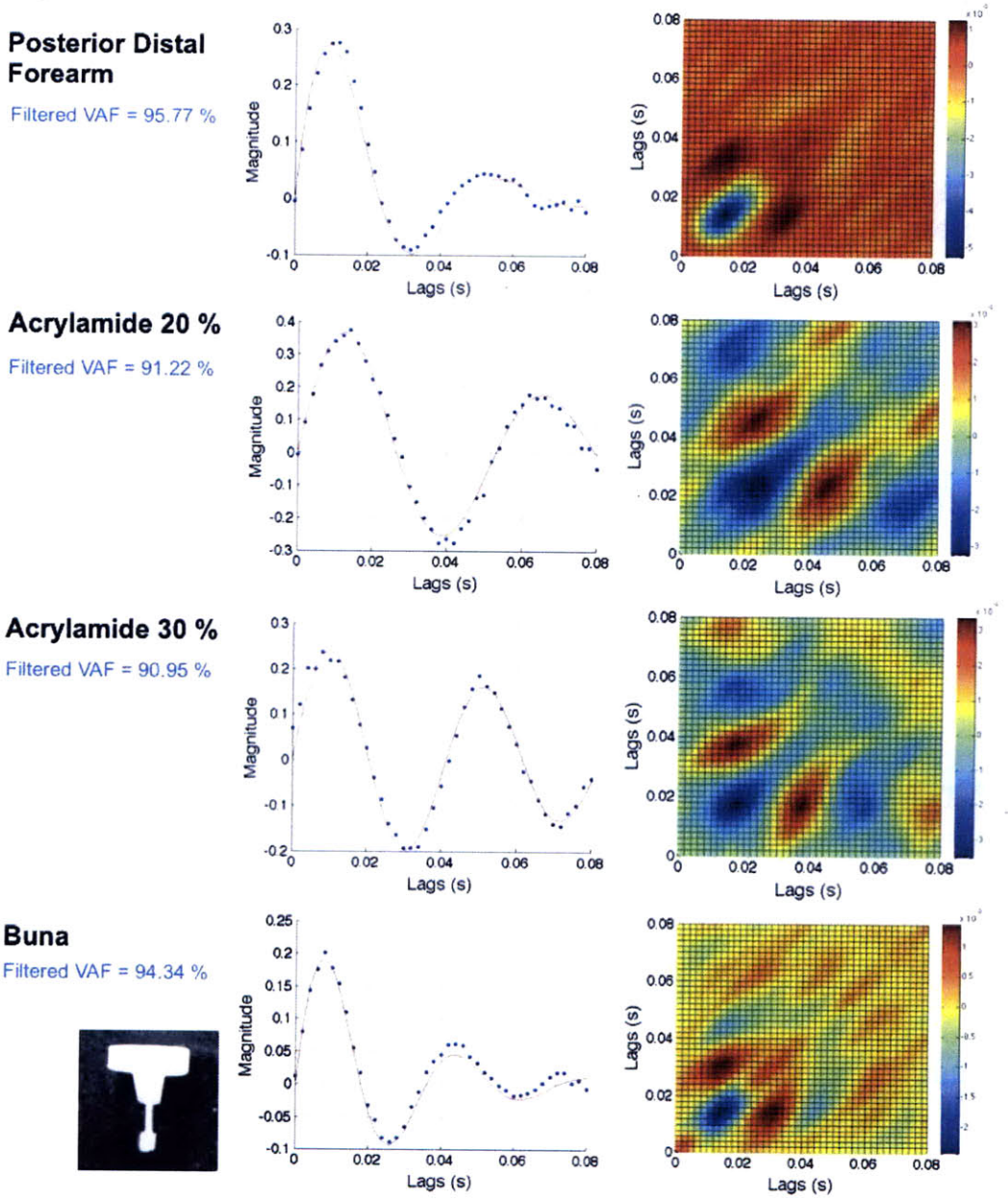


Figure 7.14: Volterra kernel comparison of skin and synthetic materials using large indentation probe.

the skin exhibits more nonlinear characteristics than the other synthetic materials in this study. Indentation is an effective method for interrogating and comparing layered biological materials and synthetic materials.

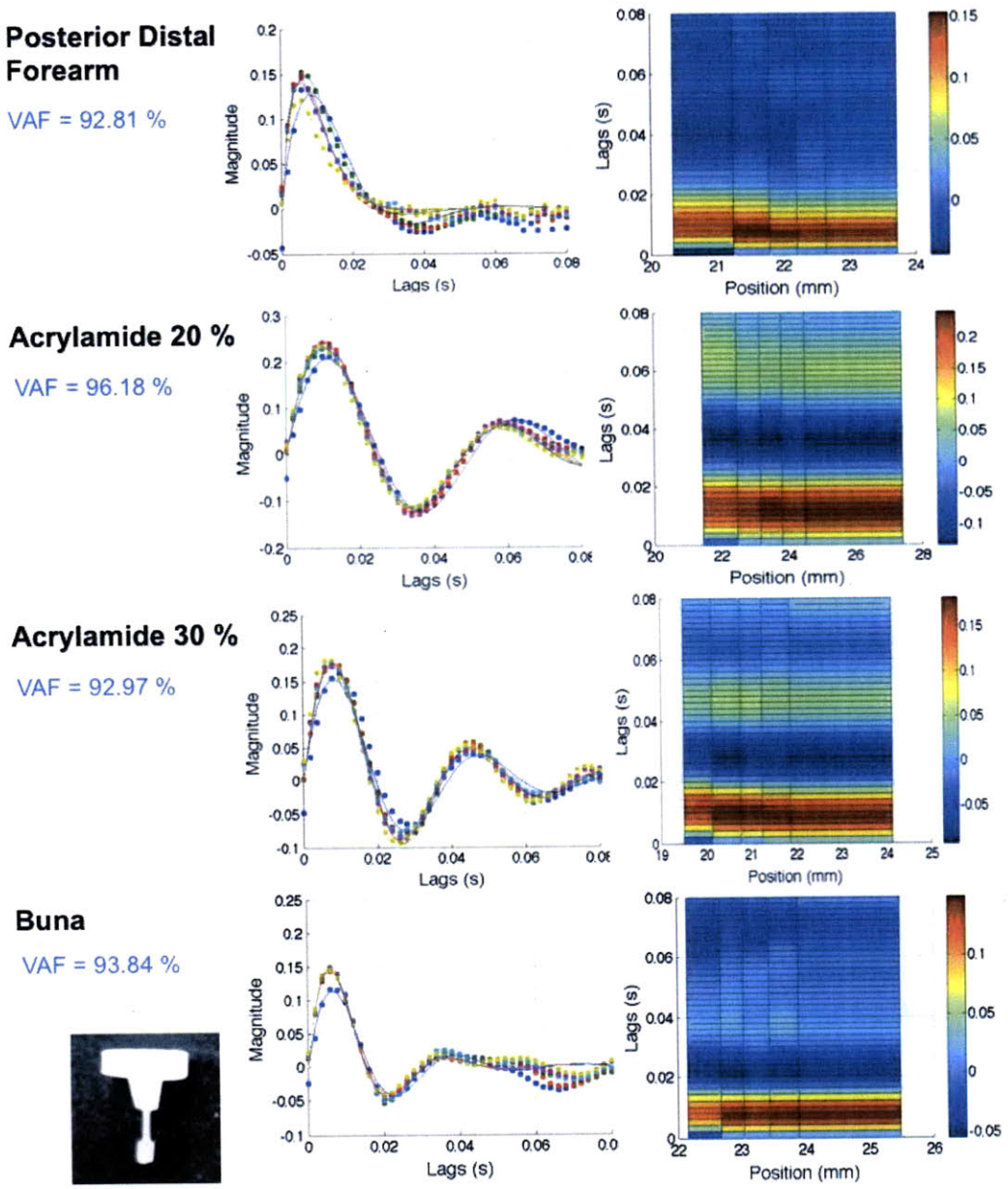


Figure 7.15: Depth-dependent partitioning technique comparison of skin and synthetic materials using large indentation probe.

7.3 Extension

Conventional extension is done *in vitro* but can also be conducted *in vivo*. Two positions on the skin were tested including the anterior proximal position and posterior proximal position 40 mm from the elbow as shown in Figure 7.16. The vertical and horizontal directions were tested and the localized linear results are shown. For lower forces, there does not appear to be a preferential alignment of the stiffness of skin in a particular orientation. As the forces increase however, preferential alignment or increased stiffness begins to appear in the orientation of the Langer's lines in those regions. For the anterior proximal location, the vertical orientation is stiffer than the horizontal orientation at larger forces. For the posterior proximal location, the horizontal orientation is stiffer. These results are in line with expectations with higher stiffness or tension in the direction parallel to the Langer's lines [21, 87].

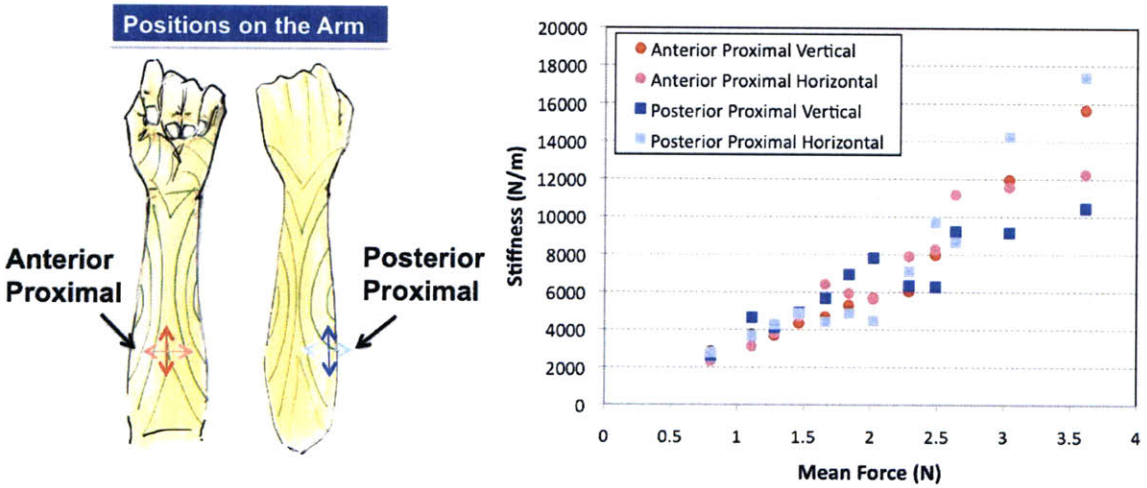


Figure 7.16: Vertical and horizontal extension testing on skin to determine the orientation of Langer's lines (green). The initial length of the area being tested was consistent for all four configurations.

Skin in extension can also be compared with other materials like 87 μm thick latex sheet and 416 μm thick Lycra Powernet as shown in Figure 7.17. The skin in extension is generally stiffer than during indentation. The latex is significantly less stiff than the skin or the Lycra material even though it had similar characteristics to skin under indentation.

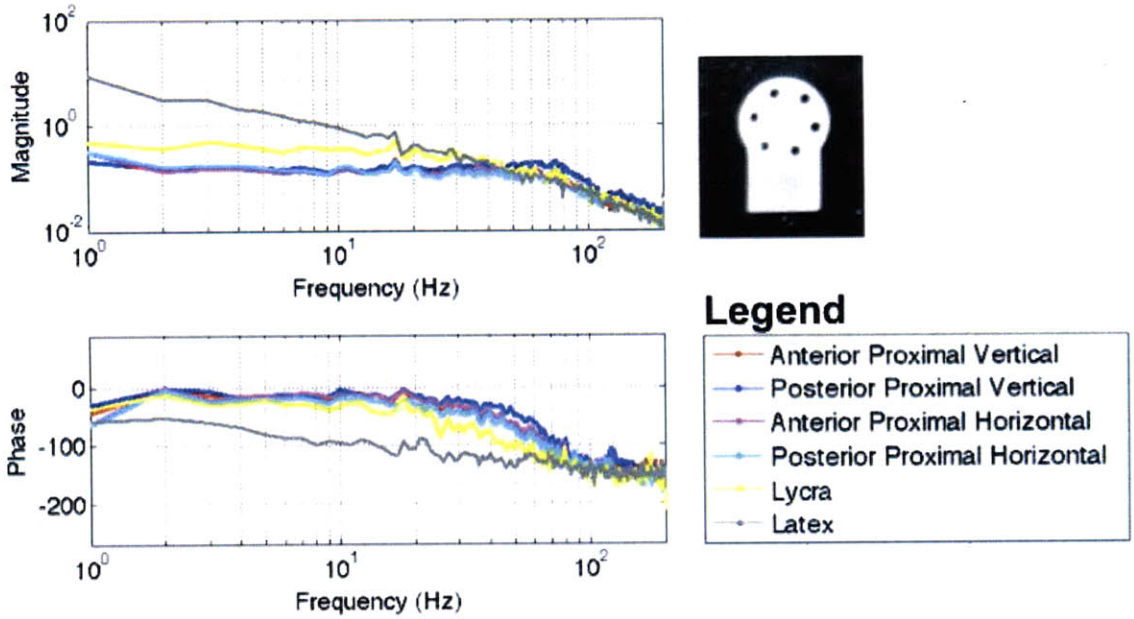


Figure 7.17: Frequency domain comparison of skin and synthetic materials during extension. Four positions on the skin are compared including the anterior proximal vertical, posterior proximal vertical, anterior proximal horizontal and posterior proximal horizontal positions. Lycra Powernet with a thickness of 416 μm and latex with a thickness of 87 μm were also tested.

When these materials are analyzed using the Wiener static nonlinearity technique, as shown in Figure 7.18, it is clear that skin, Lycra and latex all exhibit some nonlinear behavior. In the case of skin, there is looping at the bottom of the plot of the static nonlinearity that is not exhibited by the latex or Lycra. This is an indication that the behavior of the parameters (spring constant, damping, and mass) as a function of depth for skin in extension is similar to skin in indentation but different from the synthetic materials in this study. Effectively, this is the same as stating that the DPN parameter variations is similar for skin in indentation and extension but different from synthetic materials.

The extension data can also be analyzed using Volterra kernels as shown in Figure 7.19. The second Volterra kernels for skin in extension are very similar to those seen from indentation studies but the shape of the second Volterra kernels for synthetic materials is very different. The main valley in the second Volterra kernel in synthetic materials has a different shape that for skin again indicating that the DPN parameter

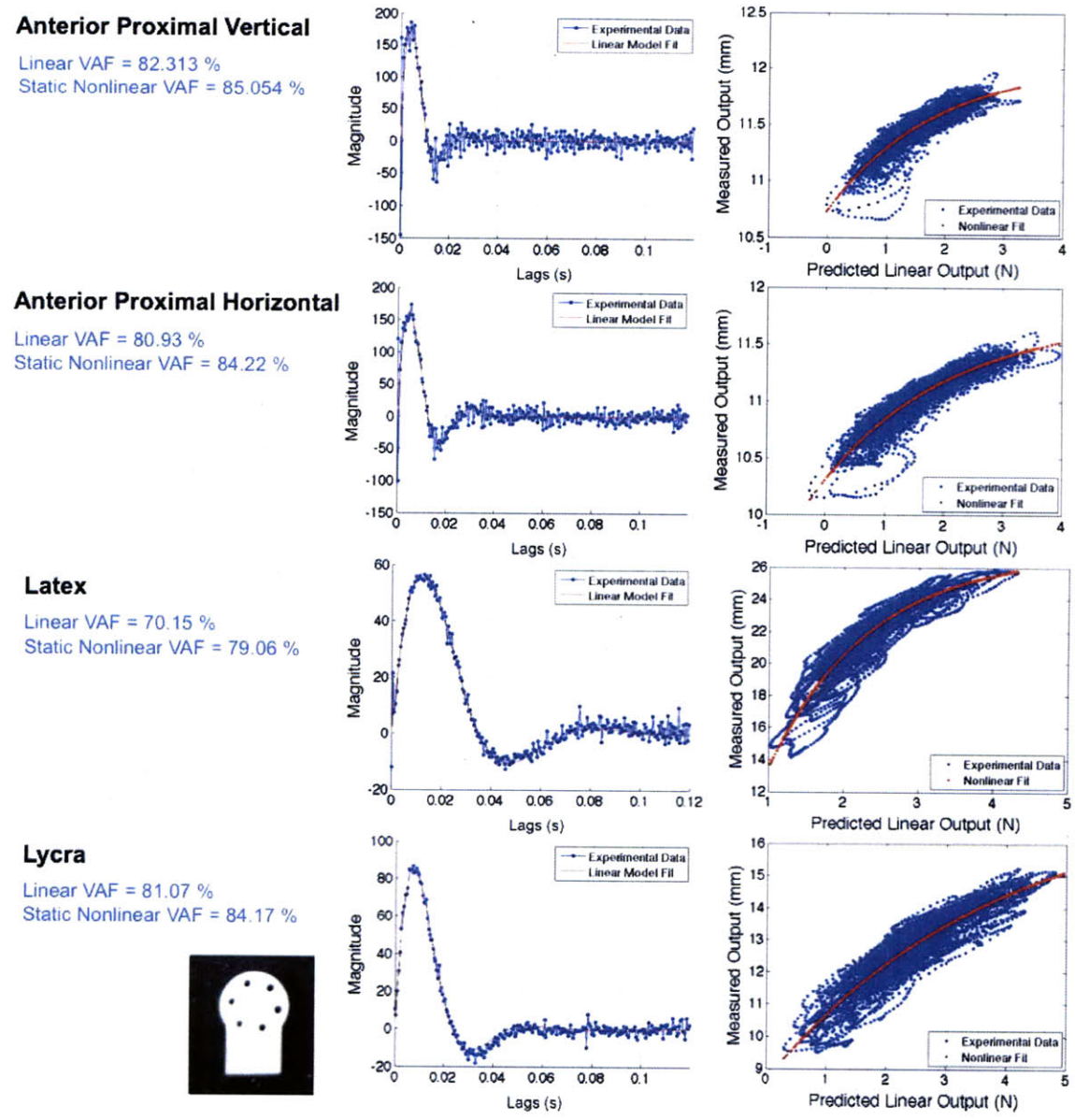


Figure 7.18: Wiener static nonlinearity comparison of skin and synthetic materials using extension probe.

variation is different.

Depth-dependent partitions are used to analyze the data in Figure 7.20. There is some variation in all off the depth-dependent kernels across the materials which means that the materials are significantly nonlinear across the tested ranges. The VAF increases slightly for the Volterra kernel representations of the skin but increases significantly for the synthetic materials. This indicates that the depth dependent

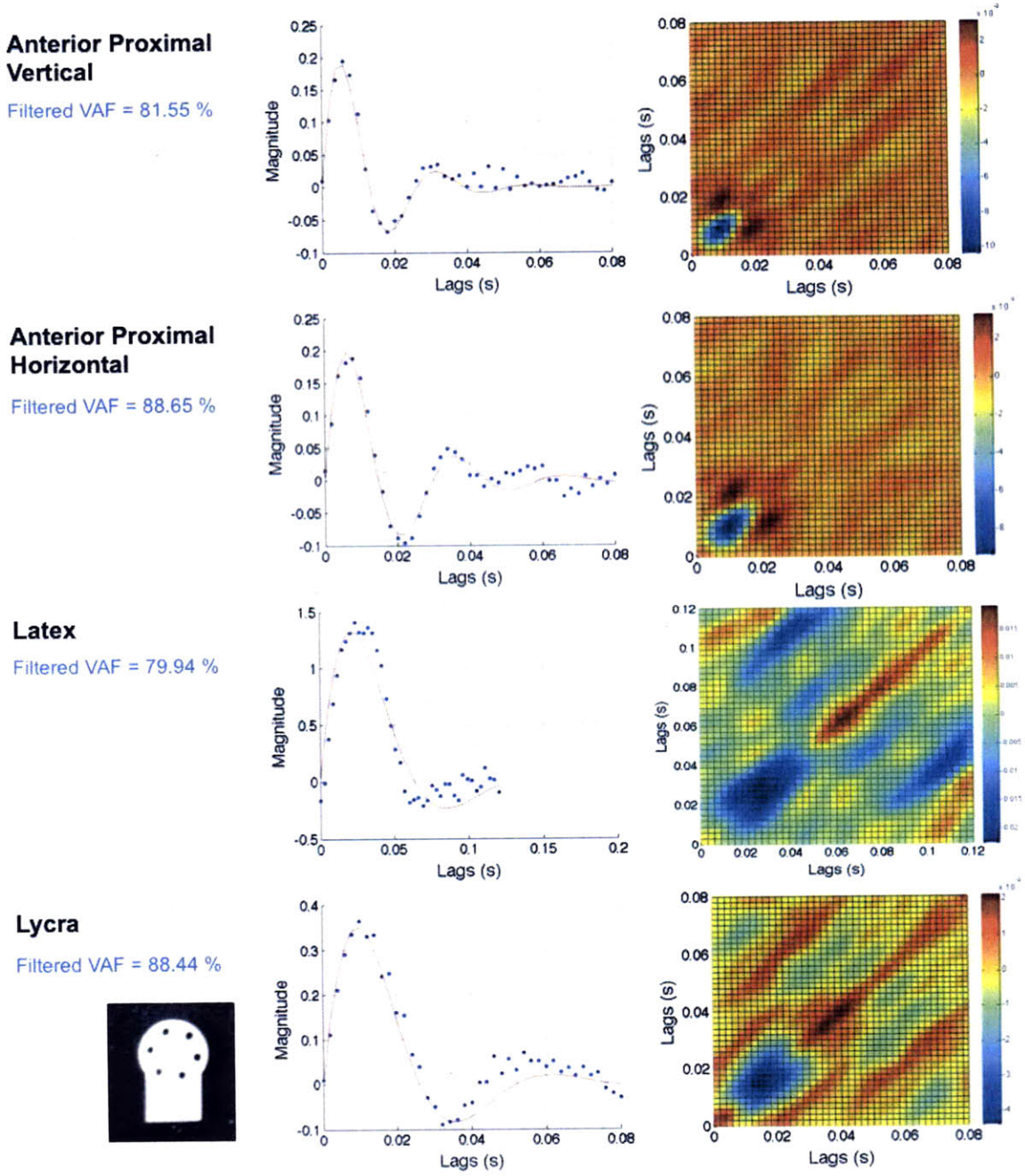


Figure 7.19: Volterra kernel comparison of skin and synthetic materials using extension probe.

partitions are a good representation of the synthetic material system and an efficient representation for biological materials. Depth dependent partitioning is an effective representation for the identification of biological materials under indentation and extension.

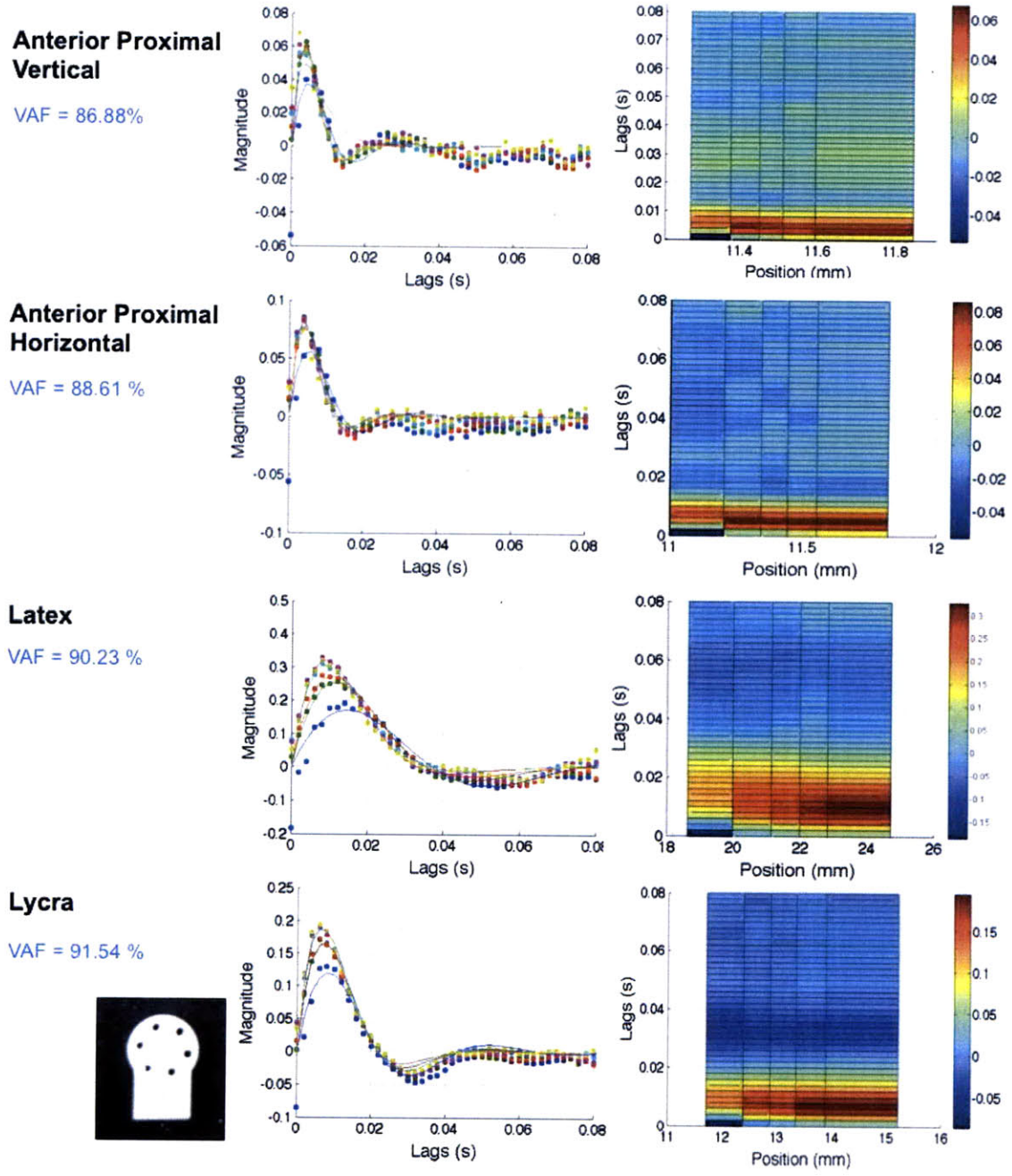


Figure 7.20: Depth-dependent partitioning technique comparison of skin and synthetic materials using extension probe.

7.4 Surface Mechanics

The frictional properties of skin have been studied by several research groups [27, 35, 46, 80, 107] using custom and commercial instruments (such as the Measurement Technologies Skin Friction Meter by Aca-Derm Inc.). On the other hand, the surface mechanics of skin includes several properties including skin texture, suppleness, and friction [107]. When a probe is placed on the skin, the skin can deform contributing to changes in the measured damping (from skin energy absorption, skin friction and other factors) as well as the spring constant (from skin suppleness, skin stiffness and other factors). In order to test this hypothesis, it is possible to measure the change in the spring constant of the skin as well as the damping using the surface mechanics probe.

Commercial products such as hydrating lotions claim that they help hold moisture in the skin. Two different products with two different hydrating strategies were tested. The Chanel Hydramax + Active is a non-greasy hydrating lotion which creates a protective layer on the skin. The Vaseline Total Moisture cream is a more viscous mixture. First, indentation studies were conducted with the localized linear method at different depths into the skin. A baseline was first collected and an hour later one of the lotions was applied and the skin was tested again. Once a lotion was applied, the skin was allowed to rest for more than one day before any further testing. The results of the indentation tests are shown in Figure 7.21.

There is a very small difference between the use of different lotions and the control baseline. There is no significant difference in the mass or the damping but there is a spread in the spring constant. With the Chanel product, the spring constant of the skin actually increased slightly so that the skin became firmer. For the Vaseline product, the skin became softer and the spring constant was lower for every depth tested.

In terms of assessing the Chanel product, however, an indentation test is not the most ideal. The major effect of the product is to generate a hydrated layer in the skin which is best characterized by a change in smoothness or in the surface mechanics.

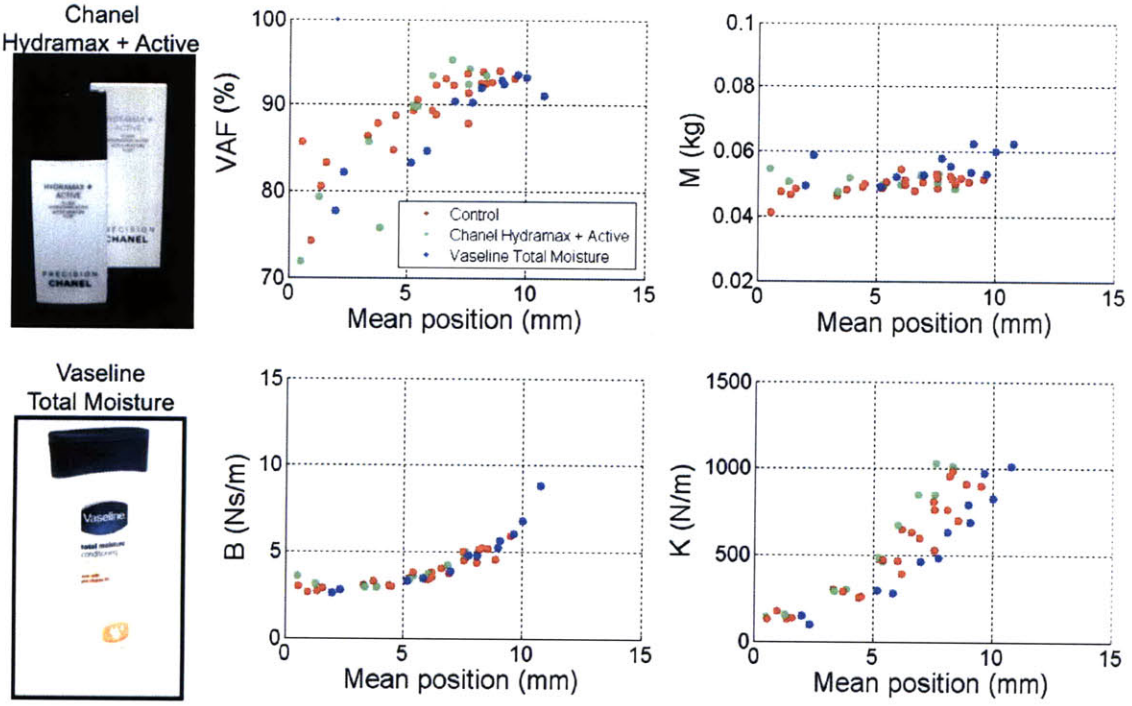


Figure 7.21: Skin surfaces were tested using indentation with and without lotions applied to the surface. The indentation tests show that the lotions have a minor effect on skin properties under an indentation test.

Therefore, a surface mechanics test was conducted after applying the product on the skin. The results of this test are shown in Figure 7.22. Each test was conducted ten times and the means and standard deviations are shown.

The spring constant of the mechanical spring used in the system was subtracted from the measurement of the compliance. Therefore, the remaining compliance is only the contribution of the skin in a surface mechanics geometry. From these results, it is clear that there is a difference between the compliance of the skin before and after the product was applied. The Chanel product made the skin more compliant in the sliding direction which means that the skin surface would appear to be smoother and more supple. In addition, the Chanel product reduced the damping of the skin which indicates that it significantly reduced the roughness or friction of the skin at the surface.

Additional long term testing with other products is necessary to fully assess the effect of different lotions and creams immediately after application and a few minutes

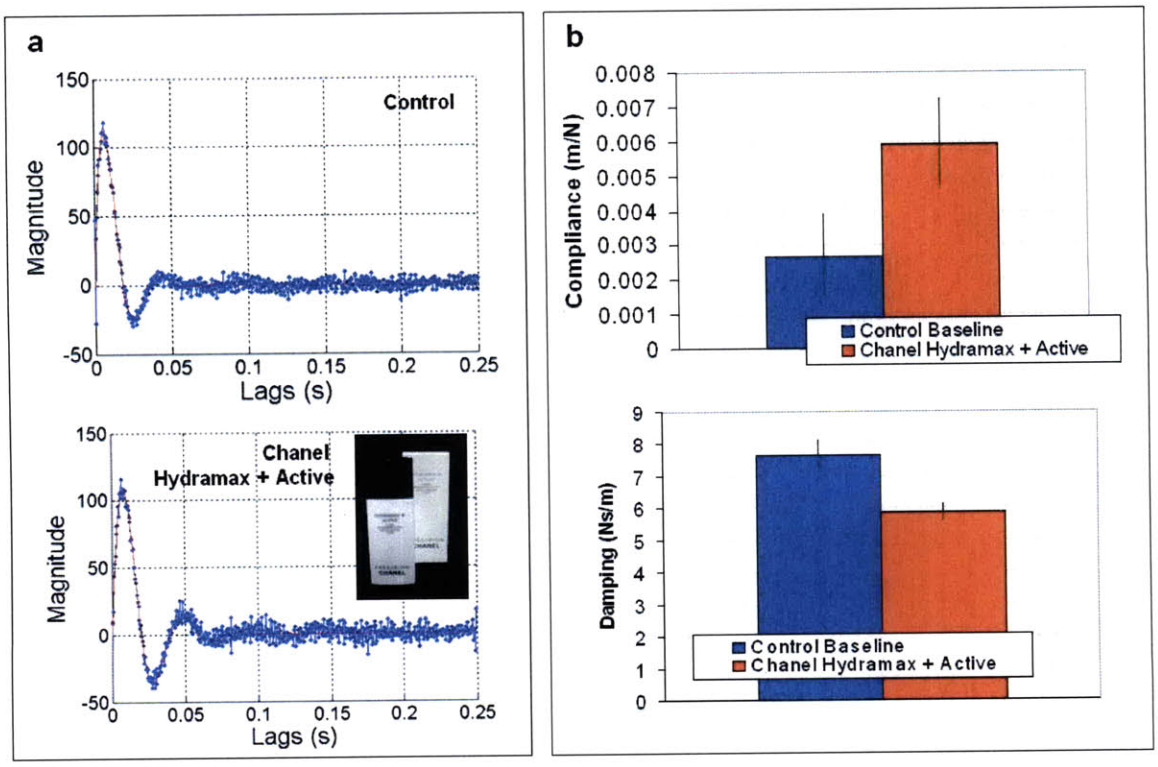


Figure 7.22: Skin surfaces were tested in a surface mechanics configuration with and without lotion. These tests show that the Chanel Hydromax + Active lotion has a larger effect in on the surface mechanics of skin after application. (a) The impulse responses and (b) fitted parameter data are shown. The normal preload is 1 N. The probe dimensions are 5 mm by 16 mm with an edge radius of 1.5 mm.

or hours after application. Nevertheless, these results indicate that the instrument designed in this research is consistent and sensitive enough to measure differences that can be felt by touch or palpation. In addition, it makes a clear quantitative assessment that can be compared directly.

Chapter 8

Conclusions and Recommendations

The dynamic properties of biological tissues can be characterized using nonlinear stochastic system identification techniques. Devices, electronics, and data acquisition software were created. Several different linear and static nonlinear system identification techniques have been discussed. In addition Volterra kernels and partitioned techniques were outlined.

As a final overview, a comparison of the different system identification techniques discussed in this work is shown in Figure 8.1. For a real system (skin under indentation) with a memory length on the order of 40 to 60 samples, the VAF generally increases for the identification techniques discussed in this work. The lowest values of VAF are for the LMS method followed by the Wiener static nonlinearity method, which generally increases the VAF by about 5 %. The increase in the VAF is rapid until about 30 samples after which the VAF levels off. Since the true memory length is on the order of 40 to 60 samples, this result is reasonable. For memory lengths shorter than 30 samples, the LMS technique has trouble generating an impulse response that can adequately fit the data since dynamics from longer lags are lost due to the structure of the algorithm.

The first Volterra kernel is essentially the impulse response obtained through an orthonormalization process. Unlike the LMS algorithm, the algorithm used to obtain the first Volterra kernel prevents the system from losing the information at longer lags. Rather, the Gram-Schmidt algorithm attempts to do its best at fitting the

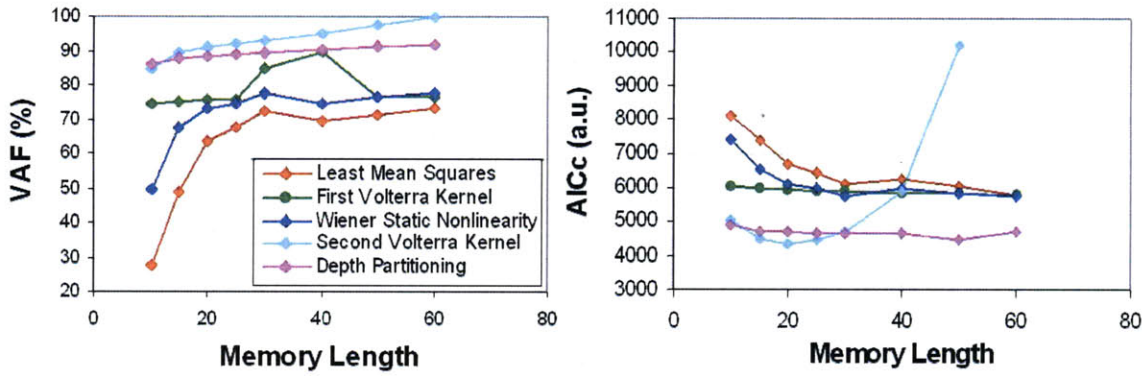


Figure 8.1: A comparison of nonlinear system identification techniques in terms of VAF and AICc as a function of memory length.

data within the given constraint. This causes the first Volterra kernel to have a VAF that is higher at shorter memory lengths than for the LMS algorithm. When the second Volterra kernel is added, the VAF increases significantly such that at memory length of 60, the fit is almost perfect. This effect, however, is misleading. The second Volterra kernel has many more parameters than any of the other models and therefore could simply be fitting noise.

In order to assess this effect, the AICc is used. As the memory length increases, the AICc decreases for the LMS, Wiener static nonlinearity, and first Volterra kernel. This is in line with expectations from the VAF. The second order kernel, however, has an AICc that increases after 20 to 30 samples. This fast increase is due to the fast growth in the number of parameters as the memory length increases. At a memory length of 40, the second Volterra kernel becomes less optimal than the other techniques.

The depth dependent partitions with seven partitions, however, follow a different trend. For depth dependent partitions, the value of the VAF remains high while the value of the AICc remains low and becomes more optimal than all other techniques after 30 samples. This indicates that the depth dependent partitions produce one of the best estimates in terms of goodness of fit and low entropy.

Table 8.1 shows a comparison of the different techniques in terms of VAF, interpretability (ability to compare results from different tests quantitatively) and com-

putation time. The stochastic linear or LMS technique has a low VAF but good interpretability and very short computational times. With the addition of Wiener static nonlinearity, the VAF increases and the computation time increases as well. The Volterra kernel produces a high VAF but has poor interpretability and extremely long computation times. The depth partitioning has good performance in all three criteria. It has a high VAF, good interpretability, and a reasonable computation time. The Wiener static nonlinearity and depth partitioning both have good characteristics

Table 8.1: Performance of nonlinear system ID methods

	VAF	Interpretability	Computation Time
Stochastic Linear (LMS)	70 to 75 %	good	0.2 to 1.0 s
Wiener Static Nonlinearity	75 to 80 %	good	2.0 to 5.0 s
Volterra Kernel	90 to 97 %	poor	80 to 100 s
Depth Partitioning	90 to 95 %	good	5.0 to 7.0 s

that can be used to assess data for clinicians. The Wiener static nonlinearity is easily interpretable and can be reduced to four parameters (or three parameters and a nonlinear function). The reduction of the depth partitioning values to fewer parameters causes the information to be slightly less interpretable (mean and slope of the spring constant, damping, and mass) and loses some of the important information. For a clinical instrument, a Wiener static nonlinearity could be a valid technique to use. For a research instrument, the depth dependent partition or the Volterra kernels capture the most information about biological tissues.

The original goals of this work along with the methods used to achieve them are listed below.

- Create low cost device to identify dynamic compliance of tissue - The device created in this work has a high bandwidth and is relatively simple. It incorporates a Lorentz force linear actuator for applying forces and a potentiometer for measuring position. The low cost of the device is outlined in Appendix A.1.
- Identify nonlinear dynamics of tissues using stochastic techniques - The nonlinear properties of tissue were assessed with a Wiener static nonlinearity, localized

linear testing, Volterra kernels, and partitioning techniques.

- Optimize the identification with input generation techniques - Inputs with different distributions and frequency spectra were generated. Real time input generation schemes optimize for the nonlinearities present in biological tissues were implemented and discussed.
- Use a fast identification and computational technique - Several fast techniques were used including the Wiener static nonlinearity and the depth dependent partitions. The computation times for these methods are on the order of 2 to 7 seconds.
- Use a technique that is good at accounting for variances in the data - The methods that have the highest VAF are the Volterra kernel techniques and the depth dependent partitioning techniques. VAF's in excess of 95 % were achieved.
- Use a technique that is readily interpretable - The Wiener static nonlinearity and the depth dependent partitioning are both readily interpretable and were used to assess the skin properties in a population study.
- The techniques must be capable of producing results that are repeatable and specific - The data obtained with these techniques is specific and very repeatable across multiple tests of the same tissue sample for a single individual.
- The techniques should be able to distinguish the change in skin properties after dehydration or after application of commercial products - Studies conducted on skin after the application of hydrating lotions showed a significant difference between the compliance properties before and after application. Other products can be assessed using these instruments and methods.

Future work for this research includes looking at other nonlinear system identification techniques including wavelets, subspace methods, and fuzzy logic models. These other models could produce better interpretability and higher VAF for a few number of parameters. Additional studies on different configurations of the partitioned techniques is also desirable in order to further assess their capabilities. The effectiveness of these techniques will also be assessed for looking at the different diseases that

are listed in Table 1.1. More population studies with clinical applications would be the last step necessary for taking this concept from research to implementation as a medical instrument.

Bibliography

[1] A. M. Andronikou and G. A. Bekey. Identifiability of hysteretic systems. In *Decision and Control Including the Symposium on Adaptive Processes*, pages 1072–1073, 1979.

[2] A. M. Andronikou, G. A. Bekey, and F. Y. Hadegh. Identifiability of nonlinear systems with hysteretic elements. *Journal of Dynamic Systems, Measurement, and Control*, 105(4):1247–1252, 1983.

[3] M. H. Asyali and M. Juusola. Use of Meixner functions in estimation of Volterra kernels of nonlinear systems with delay. *IEEE Transactions on Biomedical Engineering*, 52(2):229–237, 2005.

[4] J. E. Bischoff, E. M. Arruda, and K. Grosh. Finite element modeling of human skin using an isotropic, nonlinear elastic constitutive model. *Journal of Biomechanics*, 33:645–652, 2000.

[5] P. Bjerring and L. Arendt-Nielsen. An apparatus for penetration-free measurement of at least one mechanical property of soft biological tissue, 1991. International Patent WO 91/16003.

[6] G. Boyer, L. Laquièze, A. Le Bot, S. Laquièze, and H. Zahouani. Dynamic indentation on human skin in vivo: aging effects. *Skin Research and Technology*, 15:55–67, 2009.

[7] E. Brusseau, J. Fromageau, N. G. Rognin, P. Delachartre, and D. Vray. Investigating elastic properties of soft biological tissues. *IEEE Engineering in Medicine and Biology*, 21:86–94, 2002.

[8] F. J. Carter, T. G. Frank, P. J. Davies, D. McLean, and A. Cuschieri. Measurements and modeling of human and porcine organs. *Medical Image Analysis*, 5:231–236, 2001.

[9] H. Chen. Recursive system identification by stochastic approximation. *Communications in Information and Systems*, 6:253–272, 2006.

[10] Y. Chen and I. W. Hunter. In vivo characterization of skin using a Wiener nonlinear stochastic system identification method. In *31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6010–6013, 2009.

[11] M. S. Christensen, C. W. Hargens III, S. Nacht, and E. H. Gans. Viscoelastic properties of the intact human skin: instrumentation, hydration effects, and the contribution of the stratum corneum. *The Journal of Investigative Dermatology*, 69:282–286, 1977.

[12] W. M. Copenhaver and D. E. Kelly an R. L. Wood. *Bailey's Textbook of Histology*. The Williams and Wilkins Company, seventeenth edition, 1978.

[13] W. Courage. Measuring device for measuring the elastic properties of a surface structure, 2002. U.S. Patent 2002/0029924 A1.

[14] P. Crama and J. Schoukens. First estimates of Wiener and Hammerstein systems using multisine excitiation. In *IEEE Instrument and Measurement Technology Conference*, pages 1365–1369, 2001.

[15] C. H. Daly and G. F. Odland. Age-related changes in the mechanical properties of human skin. *The Journal of Investigative Dermatology*, 73:84–87, 1979.

[16] D. S. Dempsey. Skin, 2010. `http://www.nku.edu/~dempseyd/SKIN.htm`.

[17] Dermaviduals. Cutometer® MPA 580. Kosmetik Konzept KOKO GmbH & Co.KG, 2008.
`http://www.dermaviduals.de/deutsch/hautmessgeraete/sonde-cutometer.html`.

[18] Deutch Scientific. Courage + khazaka electronic gmbh - scientific skin test, 2010. `http://www.dutechscientific.com/products_scientific_ck_scientific.php`.

[19] Dia-stron. Dia-stron instrumentation: Skin instruments, 2010.
`http://www.diastron.com/diastron_instrumentation_skin_instruments.html`.

[20] S. Diridollou, M. Berson, V. Vabre, D. Black, B. Karlsson, F. Auriol, J. M. Gregoire, C. Yvon, L. Vaillant, Y. Gall, and F. Patat. An in vivo method for measuring the mechanical properties of the skin using ultrasound. *Ultrasound in Medicine and Biology*, 24:215–224, 1998.

[21] S. Diridollou, D. Black, J. M. Lagarde, and Y. Gall. Sex- and site-dependent variations in the thickness and mechanical properties of human skin in vivo. *International Journal of Cosmetic Science*, 22:421–435, 2000.

[22] S. Diridollou, F. Patat, F. Gens, L. Vaillant, D. Black, J. M. Lagarde, Y. Gall, and M. Berson. In vivo model of the mechanical properties of the human skin under suction. *Skin Research and Technology*, 6:214–221, 2000.

[23] H. Dobrev. In vivo study of skin mechanical properties in Scleredema of Buske. *Acta Dermato-Venereologica*, 78:103–106, 1998.

[24] G. Duchemin, P. Maillet, P. Poignet, E. Dombre, and F. Pierrot. A hybrid position/force control approach for indentification of deformation models of skin and underlying tissues. *IEEE Transactions of Biomedical Engineering*, 52(2):160–170, 2005.

[25] M. G. Dunn, F. H. Silver, and D. A. Swann. Mechanical analysis of hypertrophic scar tissue: Structural basis for apparent increased rigidity. *The Journal of Investigative Dermatology*, 84:9–13, 1985.

[26] V. Duong and A. R. Stubberud. System identification by genetic algorithm. In *IEEE Aerospace Conference*, pages 5–2331–5–2337, 2002.

[27] A. F. El-Shimi. In vivo skin friction measurements. *Journal of the Society of Cosmetic Chemists*, 28:37–51, 1977.

[28] P. Elsner, E. Berardesca, K. Wilhelm, and H. I. Maibach. *Bioengineering of the Skin: Skin Biomechanics.* CRC Press, 2002.

[29] C. Escoffier, J. Rigal, A. Rochefort, R. Vasselet, J. Lévêoque, and P. G. Agache. Age-related mechanical properties of human skin: an in vivo study. *Journal of Investigative Dermatology*, 93:353–357, 1989.

[30] P. Eykhoff. *Nonlinear Dynamic Modeling of Physiological Systems.* John Wiley & Sons, Inc., 1974.

[31] D. M. Flynn, G. D. Peura, P. Grigg, and A. H. Hoffman. A finite element based method to determine the properties of planar soft tissue. *Journal of Biomedical Engineering*, 120(2):202–210, 1998.

[32] D. J. Friedman, B. M. Künzli, Y. I A-Rahim, J. Sevigny, P. O. Berberat, K. Enjyoji, E. Csizmadia, H. Friess, and S. C. Robson. Cd39 deletion and exacerbates experimental murine colitis and human polymorphisms increase susceptibility to inflammatory bowel disease. *Proceedings of the National Academy of Sciences*, 106:16788–16793, 2009.

[33] M. Garcia-Webb, A. Taberner, N. C. Hogan, and I. W. Hunter. A modular instrument for exploring the mechanics of cardiac myocytes. *American Journal of Physiology: Heart and Circulatory Physiology*, 293:H866–H874, 2007.

[34] H. Garnier, M. Mensler, and A. Richard. Continuous-time model identification from sampled data: Implementation issues and performance evaluation. *International Journal of Control*, 76(13):1337–1357, 2003.

[35] W. A. Gerrard. Friction and other measurements of the skin surface. *Bioengineering and the skin*, 3:123–139, 1987.

[36] Y. Goussard, W. C. Krenz, L. Stark, and G. Demoment. Practical identification of functional expansions of nonlinear systems submitted to non-gaussian inputs. *Annals of Biomedical Engineering*, 19:401–427, 1991.

[37] G. L. Grove, J. Damia, M. J. Grove, and C. Zerweck. *Handbook of Non-Invasive Methods and the Skin, Second Edition*, chapter 68: Suction chamber method for measurement of skin mechanics: The DermaLab, pages 593–599. Informa Healthcare, 2006.

[38] T. Haku and M. Ri. Instrument for measuring mechanical characteristics on surface of viscoelastic body, 2004. Japanese Patent 2004-085548.

[39] P. Hansma, B. Drake, D. Rehn, J. Adams, and J. Lulejian. Methods and instruments for materials testing, 2009. U. S. Patent 2009/0056427 A1.

[40] B. D. Hemond, D. M. Wendell, N. C. Hogan, A. J. Taberner, and I. W. Hunter. A Lorentz-force actuated autoloading needle-free injector. In *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 679–682, 2006.

[41] F. M. Hendriks, D. Brokken, C. W. J. Oomens, and F. P. T. Baaijens. Influence of hydration and experimental length scale on the mechanical response of human skin in vivo, using optical coherence tomography. *Skin Research and Technology*, 10:231–241, 2004.

[42] F. M. Hendriks, D. Brokken, C. W. J. Oomens, D. L. Bader, and F. P. T. Baaijens. The relative contributions of different skin layers to the mechanical behavior of human skin in vivo using suction experiments. *Medical Engineering & Physics*, 28:259–266, 2006.

[43] F. M. Hendriks, D. Brokken, J. T. W. M. van Eemeren, C. W. J. Oomens, F. P. Baaijens, and J. B. A. M. Horsten. A numerical-experimental method to characterize the non-linear mechanical behavior of human skin. *Skin Research and Technology*, 9:274–283, 2003.

[44] F. Henry, V. Goffin, C. Piérard-Franchimont, and G. E. Piérard. Mechanical properties of skin in Ehlers-Danlos Syndrome, types I, II, and III. *Pediatric Dermatology*, 13:464–467, 1996.

[45] G. Hillebrand. Research proposal [e-mail]. Available e-mail: hillebrand.gg@pg.com, June 2010.

[46] R. J. Hills, A. Unsworth, and F. A. Ive. A comparative study of the frictional properties of emollient bath additives using procine skin. *British Journal of Dermatology*, 130:37–41, 1994.

[47] I. W. Hunter. Measuring properties of an anatomical body, 2009. U. S. Patent 7,530,975 B2.

[48] I. W. Hunter and R. E. Kearney. Generation of random sequences with jointly specified probability density and autocorrelation functions. *Biological Cybernetics*, 47:141–146, 1983.

[49] I. W. Hunter and M. J. Korenberg. The identification of nonlinear biological systems: Wiener and Hammerstein cascade models. *Biological Cybernetics*, 55:135–144, 1986.

[50] T. Imoto. Hardness tester for living body, 1999. U. S. Patent 5,879,312.

[51] IndiaSmart. Derma lab elasticity module, 2010. `http://www.indiamart.com/gv-biomedicals/biomedical-products.html`.

[52] B. Joseph, K. Mulpuri, S. Paravatty, S. U. Kamath, and R. A. Varghese. Device for measurement of tissue hardness, 2004. International Patent WO 2004/058066 A1.

[53] A. Juditsky, H. Hjalmarsson, A. Benveniste, B. Deylon, L. Ljung, J. Sjöberg, and Q. Zhang. Nonlinear black-box models in system identification: Mathematical foundations. *Automatica*, 31:1725–1750, 1995.

[54] M. Kaneko and T. Kawahara. Co-axis type non-contact impedance sensor. In *IEEE International Conference on Robotics and Automation*, pages 709–714, 2004.

[55] O. Kaneko and T. Fujimura. Apparatus for measuring hardness, 2004. Japanese Patent 2004-239686.

[56] O. Kaneko, K. Inagaki, H. Inaba, and M. Yamaki. Sensor incorporating vibration exciter for measuring dynamic characteristic of biological surface part, 1998. Japanese Patent 10-314122.

[57] Tohru Katayama. *Subspace Methods for System Identification (Communications and control engineering)*. Springer-Verlag London Limited, 2005.

[58] T. Kawahara, K. Tokuda, N. Tanaka, and M. Kaneko. Noncontact impedance sensing. *International Symposium on Artificial Life and Robotics*, 10:35–40, 2006.

[59] T. Kawazoe, K. Saratani, Y. Ishii, N. Kaneko, and K. Kimura. Probe having a giant magnetorestrictive material for organism diagnosis, 1994. U.S. Patent 5,277,200.

[60] H. Kazama, T. Osanai, S. Akasaki, N. Inoue, and M. Kawai. Skin character measuring probe, 2001. Japanese Patent 2001-046344.

[61] R. E. Kearney and I. W. Hunter. Nonlinear identification of stretch reflex dynamics. *Annals of Biomedical Engineering*, 16:79–94, 1988.

[62] F. Khatyr, C. Imberdis, P. Vescovo, D. Varchon, and J. Lagarde. Model of the viscoelastic behavior of skin in vivo and study of anisotropy. *Skin Research and Technology*, 10:96–103, 2004.

[63] M. J. Korenberg. Identifying nonlinear difference equation and functional expansion representations: the fast orthogonal algorithm. *Annals of Biomedical Engineering*, 16:123–142, 1988.

[64] M. J. Korenberg. The identification of nonlinear biological systems: Wiener kernel approaches. *Annals of Biomedical Engineering*, 18:629–654, 1990.

[65] M. J. Korenberg. Parallel cascade identification and kernel estimation for nonlinear systems. *Annals of Biomedical Engineering*, 19:429–455, 1991.

[66] M. J. Korenberg, S. B. Bruder, and P. J. McIlroy. Exact orthogonal kernel estimation from finite data records: extending Wiener's identification of nonlinear systems. *Annals of Biomedical Engineering*, 16:201–214, 1988.

[67] M. J. Korenberg and I. W. Hunter. The identification of nonlinear biological systems: LNL cascade models. *Biological Cybernetics*, 55:125–134, 1986.

[68] M. J. Korenberg and I. W. Hunter. The identification of nonlinear biological systems: Volterra kernel approaches. *Annals of Biomedical Engineering*, 24:250–268, 1996.

[69] M. J. Korenberg and I. W. Hunter. Two methods for identifying Wiener cascades having non-invertible static nonlinearities. *Annals of Biomedical Engineering*, 27:793–804, 1999.

[70] A. Kyprianou and K. Worden. Identification of hysteretic systems using the differential evolution algorithm. *Journal of Sound and Vibration*, 248(2):289–314, 2001.

[71] Y. W. Lee and M. Schetzen. Measurement of Wiener kernels of a non-linear system by cross-correlation. *International Journal of Control*, 2(3):237–254, 1965.

[72] C. C. Lin, T. T. Soong, and H. G. Natke. Real-time system identification of degrating structures. *Journal of Engineering Mechanics*, 116(10):2258–2274, 1990.

[73] O. A. Lindahl, S. Omata, and K. A. Ängquist. A tactile sensor for detection of physical properties of human skin in vivo. *Journal of Medical Engineering and Technology*, 22(4):147–153, 1998.

[74] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1987.

[75] S. H. Lo. Molecules in focus: Tensin. *The International Journal of Biochemistry and Cell Biology*, 36:31–34, 2004.

[76] J. F. M. Manschot and A. J. M. Brakkee. The measurement and modelling of the mechanical properties of human skin in vivo - I. the measurement. *Journal of Biomechanics*, 19(7):511–515, 1986.

[77] V. Z. Marmarelis. Identification of nonlinear biological systems using Laguerre expansions of kernels. *Annals of Biomedical Engineering*, 21:573–589, 1993.

[78] V. Z. Marmarelis. *Nonlinear Dynamic Modeling of Physiological Systems*. John Wiley & Sons, Inc., 2004.

[79] A. Menciassi, G. Scalari, A. Eisinberg, C. Anticoli, P. Francabandiera, M. C. Carrozza, and P. Dario. An instrumented probe for mechanical characterization of soft tissues. *Biomedical Microdevices*, 3(2):149–156, 2001.

[80] S. Nacht, J. A. Close, D. Yeung, and E. H. Gans. Skin friction coefficient: changes induced by skin hydration and emollient application and correlation with percieved skin feel. *Journal of the Society of Cosmetic Chemists*, 32:55–65, 1981.

[81] H. Oka and T. Irie. Mechanical impedance of layered tissue. *Medical Progress through Technology*, 21:1–4, 1997.

[82] S. Omata. Equipment for calculating age of skin and its method, 2001. Japanese Patent 2001-212087.

[83] M. P. Ottensmeyer and J. K. Salsbury. In vivo data acquisition instrument for solid organ mechanical property measurement. *Lecture Notes in Computer Science*, 2208:975–982, 2001.

[84] L. Pedersen and G. B. E. Jemec. Mechanical properties and barrier function of healthy human skin. *Acta Dermato-Venereologica*, 86:308–311, 2006.

[85] S. Pitmann. US skin care market still has potential despite economic woes, February 2008. `http://www.cosmeticsdesign.com`.

[86] R. O. Potts, E. M. Buras, and D. A. Chrisman. Changes with age in the moisture content of human skin. *The Journal of Investigative Dermatology*, 82:97–100, 1984.

[87] R. Reihsner, B. Balogh, and E. J. Menzel. Two-dimensional elastic properties of human skin in terms of an incremental model at the in vivo configuration. *Medical Engineering and Physics*, 17(4):304–313, 1995.

[88] R. Reihsner and E. J. Menzel. Two-dimensional stress-relaxation behavior of human skin as influenced by non-enzymatic glycation and the inhibitory agent aminoguanidine. *Journal of Biomechanics*, 31:985–993, 1998.

[89] M. D. Ridge and V. Wright. Mechanical properties of skin: a bioengineering study of skin structure. *British Journal of Dermatology*, 77:1602–1606, 1965.

[90] G. Scalari, A. Eisinberg, A. Menciassi, M. C. Carrozza, and P. Dario. Micro instrumentation for non-invasive measurement of mechanical properties of tissues. In *1st Annual International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine & Biology*, pages 199–202, 2000.

[91] M. Schetzen. Measurement of the kernels of a nonlinear system of finite order. *International Journal of Control*, 1:251–263, 1965.

[92] J. Schramm-Baxter and S. Mitragotri. Needle-free jet injections: dependence of jet penetration and dispersion on the skin on jet power. *Journal of Controlled Release*, 97:527–535, 2004.

[93] N. K. Sinha and B. Kuszta. *Modeling and Identification of Dynamic Systems.* Van Nostrand Reinhold Company Inc., 1983.

[94] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P. Glorennec, H. Hjalmarsson, and A. Judisky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31:1691–1724, 1995.

[95] L. Slomianka. Blue histology - gastrointestinal tract, May 2009. `http://www.lab.anhb.uwa.edu.au/mb140/CorePages/GIT/git.htm`.

[96] A. W. Smyth, S. F. Masri, E. B. Kosmatopoulos, A. G. Chassiakos, and T. K. Caughey. Development of adaptive modeling techniques for non-linear hysteretic systems. *International Journal of Non-Linear Mechanics*, 37:1435–1451, 2002.

[97] T. T. Soong and W. N. Huang. A stochastic model for biological tissue elasticity in simple elongation. *Journal of Biomechanics*, 6:451–458, 1973.

[98] L. W. Stark. The pupillary control system: its nonlinear adaptive and stochastic engineering design characteristics. *Automatica*, 5:655–676, 1969.

[99] G. Strang. *Computational Science and Engineering.* Wellesley-Cambridge Press, 2007.

[100] E. M. Timanin. Models of vibrations generation within biological tissue by a surface source. In *XI Session of the Russian Acoustical Society*, pages 712–715, 2001.

[101] E. M. Timanin. Interpretation of impedance characteristics of biological soft tissues in the models with a pressure source of vibrations with friction. In *XIII Session of the Russian Acoustical Society*, pages 581–584, 2003.

[102] E. M. Timanin and E. V. Eremin. Mechanical impedance of biological soft tissues: possible models. *Russian Journal of Biomechanics*, 3:78–86, 1999.

[103] A. Tosti, G. Compagno, M. L. Fazzini, and S. Villardita. A ballistometer for the study of the plasto-elastic properties of skin. *The Journal of Investigative Dermatology*, 69:315–317, 1977.

[104] H. Unbehauen and G. P. Rao. Continuous-time approaches to system identification - a survey. *Automatica*, 26(1):23–35, 1990.

[105] D. Westwick and M. Verhaegen. Identifying MIMO Wiener systems using subspace model identification methods. *Signal Processing*, 52:235–258, 1996.

[106] T. Wigren. Recursive prediction error identification using the nonlinear Wiener model. *Automatica*, 29(4):1011–1025, 1993.

[107] M. Zhang and A. F. T. Mak. In vivo friction properties of human skin. *Prosthetics and Orthotics International*, 23:135–141, 1999.

[108] Y. Zhao and R. E. Kearney. Identification of Hammerstein system using subspace methods with applications to ankle joint stiffness. In *31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6010–6013, 2009.

[109] Y. P. Zheng and A. F. T. Mak. An ultrasound indentation system for biomechanical properties assessment of soft tissues in-vivo. *IEEE Transactions of Biomedical Engineering*, 43:912–918, 1996.

Appendix A

Device Cost

A.1 Commercializable Device

In order to be commercializable, the device must be fundamentally low cost. Therefore, the cost of the components for the device was assessed and compared with a quote for another device in the market. Component prices are listed below in Table A.1. It was found that the total cost per unit is about $239.68. Note that labor, tooling and assembly is not included and prices are for volumes up to 2000 units from suppliers. Prices can be lower for higher volumes.

The pricing is broken up into costs for the mechanical structure and costs for the electronics. Without electronics, the materials cost is $45.04. This includes the voice coil, the bearing structure, sensors and probes. The electronics, on the other hand, cost much more mostly due to the requirements of the power supply. The power supply alone costs $84.01 while other electronic components cost $110.63. The most expensive electronics include the linear operational amplifiers and the instrumentation amplifiers.

Table A.1: Cost for commercializable device.

Item	Supplier	Part Number	Cost per unit	Qty	Cost
Mechanical Structure					
Iron Core	McMaster	8909K783	$1.27	1	$1.27
Magnet - NdFeB	K J Magnetics	DX0X0-N52	$11.40	1	$11.40
Steel	McMaster	8920K23	$1.45	1	$1.45
Bobbin - Acetal, Turcite for Bearing Applications	McMaster	7521T16	$2.60	1	$2.60
Copper Coil - 28 Gage	McMaster	7588K53	$1.39	1	$1.39
Bearing Structure - Injection Molded Acetal	Protomold	Custom*	$3.36	1	$3.36
Handle - Injection Molded Acetal	Protomold	Custom*	$3.36	2	$6.72
Hardware (Screws - M3)	McMaster	91420A116	$0.02	12	$0.29
Hardware (Screws - M1.6)	McMaster	92005A001	$0.16	11	$1.71
Button	Digikey	PN12SHNA03QE	$0.85	1	$0.85
Accelerometer	Digikey	ADXL345BCCZ-RL7	$3.85	1	$3.85
Potentiometer	Mouser	688-RDC10320RB	$1.95	1	$1.95
Temperature Sensor	Omega	F2020-100-B	$1.00	1	$1.00
Indentation Probe	Protomold	Custom*	$2.40	1	$2.40
Extension Probe	Protomold	Custom*	$2.40	1	$2.40
Surface Mechanics Probe	Protomold	Custom*	$2.40	1	$2.40
Electronics					
Microcontroller - Arm Cortex M3, 32 bit, 72 MHz, 512 Kb Flash, 12 bit A/D (21), 12 bit D/A (2), USB	Digikey	STM32F103ZET6	$16.92	1	$16.92
Voltage Regulator - 3.3 V	Digikey	NDLT1963AEST-3.3	$2.50	1	$2.50
PCB - 2 Layer with Soldermask, 58 cm^2	PCB Express	Express E2**	$8.88	1	$8.88
Linear Operational Amplifier	Mouser	OPA549T	$15.30	2	$30.60
Instrumentation Amplifier	Digikey	AMP02FS	$7.19	4	$28.76
Current Sense Resistor - 4 Wire, 0.04 Ω	Mouser	13FPR040E	$3.15	1	$3.15
Resistors and Capacitors	Digikey	Multiple	$0.10	24	$2.40
Power Supply - 48 V, 2 A	Digikey	DT100PW480C	$84.01	1	$84.01
USB Mini-B Connector	Mouser	897-43-005-00-100001	$0.56	1	$0.56
USB Cable	Mouser	UX40-MB-5PP-500-1002	$16.86	1	$16.86
Total Cost Per Unit					**$239.68**

* The tooling cost is approximately $3,000 per plastic part.

** The PCB cost is for a quantity of 100 and may include tooling costs from the supplier.

A.2 Quote for Cortex Technology DermaLab

The following quote is for the Cortex Technology DermaLab elasticity testing device.

cyberDERM

275 New Darlington Road
Media, PA 19063

Certified WBENC Women's Business Enterprise

19 January 2009

Attn: Ellen Yi Chen
Department of Mechanical Engineering
Massachusetts Institute of Technology

Price Quotation

Part #	Quantity	Description	Price
103	1	Cortex Technology DermaLab Elasticity Module includes Probe- Includes 1 roll of 500 suction cup adhesives	$7,775
214	1	Power Supply for Application Module	$275
220	1	Weee Fee (electrical waste Disposal) - $50.00 Each	$50
		Total	**$8,100**
		Accessories	
203	1	Additional Probe	$3,850
206		500 Suction Cup Adhesives	$300

Terms of delivery: These prices include all customs and insurance and UPS ground delivery to a USA address. It does not include local and/or state taxes if applicable. Delivery would be in 4-6 weeks of receiving a Purchase Order.

Terms of sale: We warrant all products to be free from defects in material and workmanship. Our obligation is to repair or replace any of the products that prove to be defective within one year of shipment. No other warranty is expressed or implied. This warranty is void for products that are handled carelessly or modified by the user.

Terms of payment: Net 30 days.

Validity: These prices will remain in effect for 90 days and the order can be placed at above address. General Terms and Conditions of Sale apply.

Fax: 610-325-0881 **e-mail:** cyberderm@comcast.net **web page**: www.cyberderm-inc.com

Table A.2: Quote for DermaLab elasticity device.

Appendix B

LabView

The detailed LabVIEW software for this work is described in this chapter. In order to maintain a clean code structure, abstraction through extensive usage of variable names and external processing with MATLAB was utilized. In addition, inputs were pre-generated and saved in separate files. Lastly, buffer functions within the data acquisition boards was used to optimize data output and collection.

B.1 Calibration Software

The LabVIEW software used for calibration is shown below in Figure B.1. There are two pieces of software for static calibration and dynamic calibration. For static calibration, the program applies different forces to the coil to move it back and forth to determine the total range as well as the input voltage to output force conversion. The LabVIEW code shows how the specialized input is split into two signals which go to the linear actuator. It also shows how data is collected, scaled, and plotted. Additional pieces of software create calibration constants and this information is saved into a readable file. This is shown in Figure B.2. Figure B.3 shows the dynamic calibration program which grabs the data that is saved from the static calibration and supplies it to the scaling constants used in the system. This program also uses a pre-specified input that is sent to the coil for calibration. The data is also collected, plotted and saved in a file.

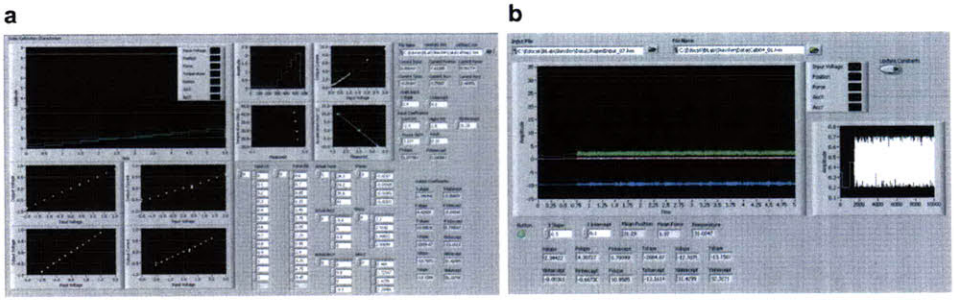


Figure B.1: Software used to calibrate the system at startup. (a) The static calibration system configures the force, position, and voltage readings while (b) the dynamic calibration system configures the full second order linear dynamics of the coil at different positions.

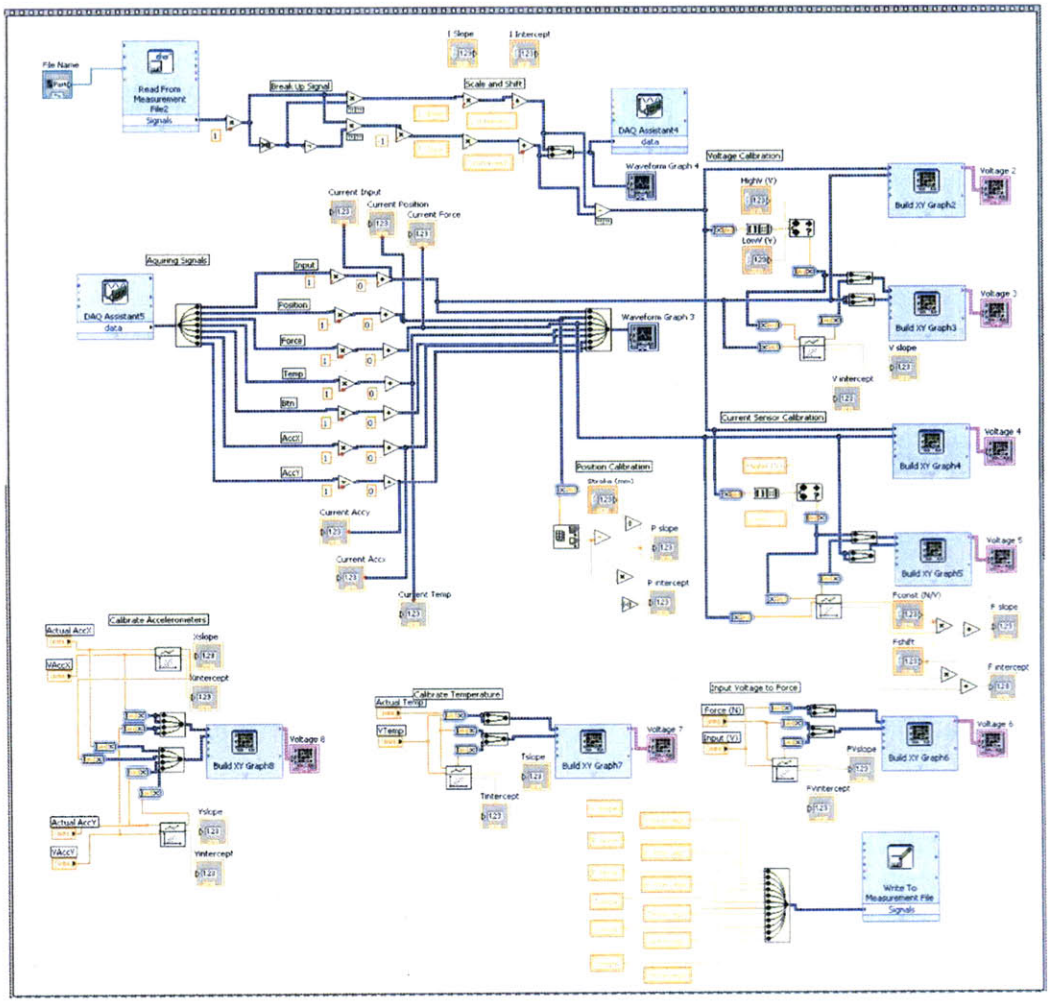


Figure B.2: Static calibration LabVIEW program.

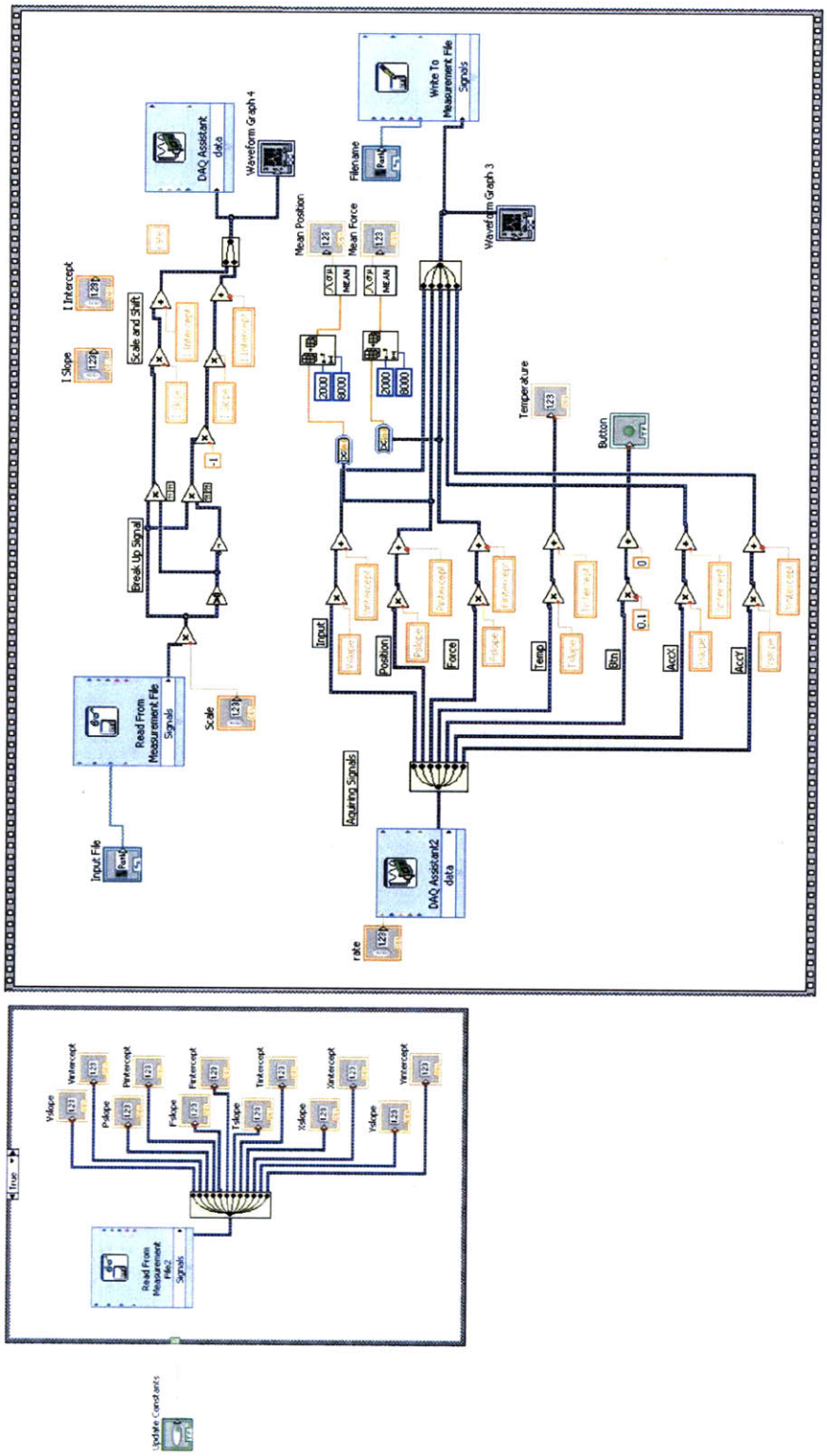


Figure B.3: Dynamic calibration LabVIEW program.

B.2 System Identification Software

The system identification software is responsible for completing a Wiener static nonlinear system identification process on the incoming data. It is made to be user friendly and gives a quick overview of all relevant parameters. This is shown in Figure 2.13. The first step in the process is a simple set of instructions and a start button as shown in Figure B.5. The next step is to load a pre-generated stochastic input, translate the input for the two linear operational amplifiers, and then read in the data from the potentiometer. The data is scaled with calibration constants after the information has been collected. This shown in Figure B.6. The last step is to send the information to a MATLAB node which processes it and sends it back to the LabVIEW program to be plotted. This is shown in Figure B.7.

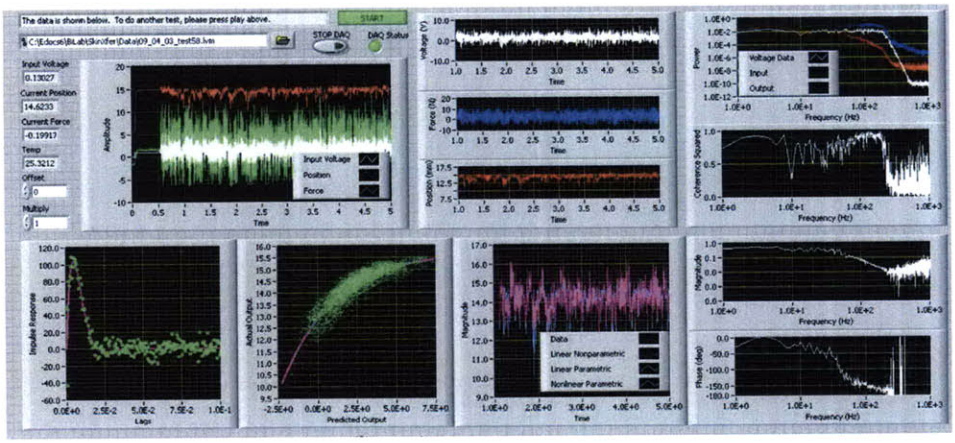


Figure B.4: LabVIEW user interface which takes measurements, completes system identification and displays information to the user.

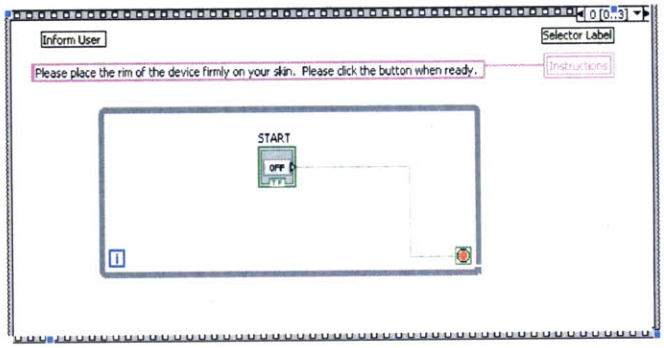


Figure B.5: System identification LabVIEW program start panel.

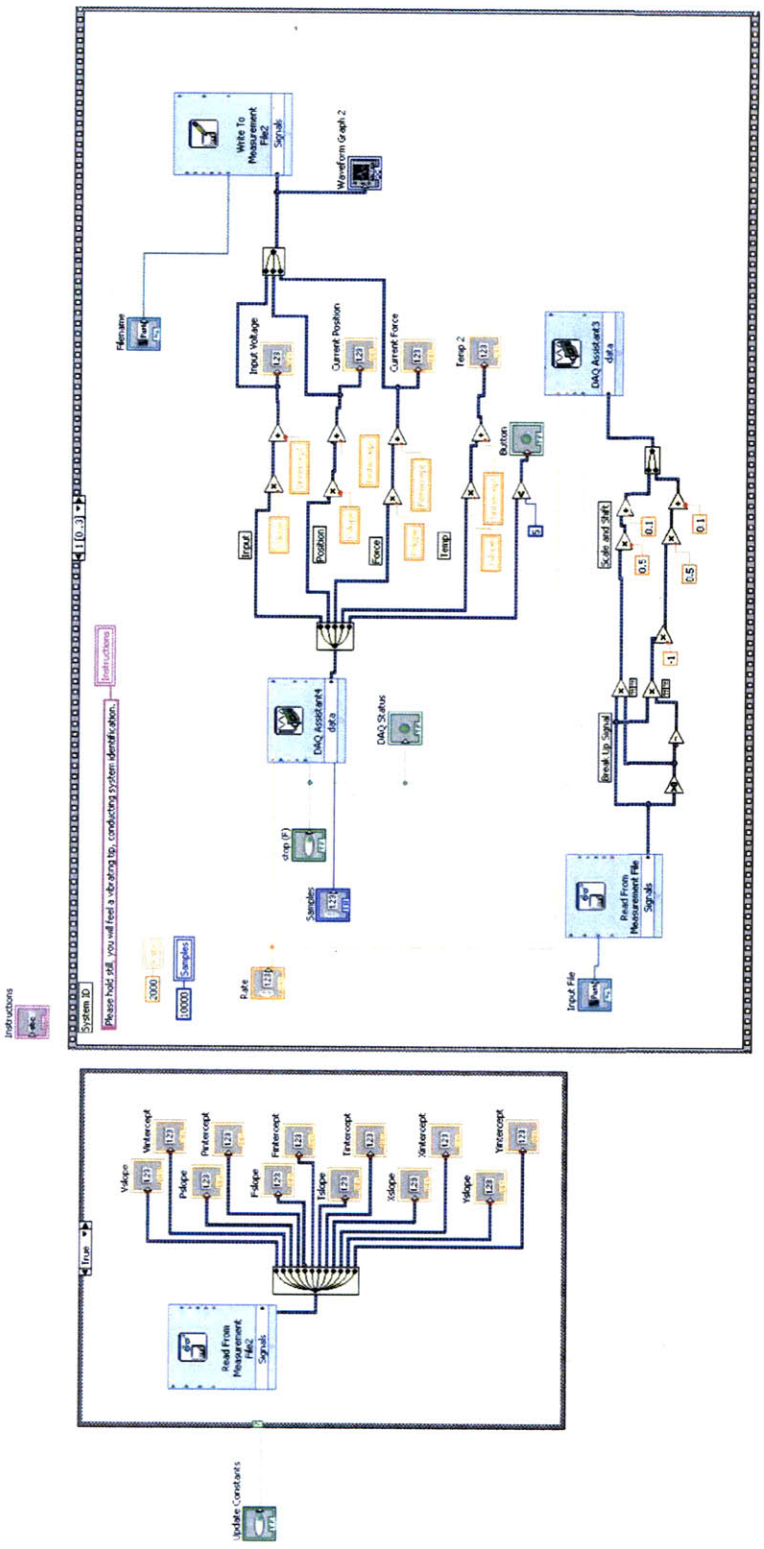


Figure B.6: System identification LabVIEW program data acquisition panel.

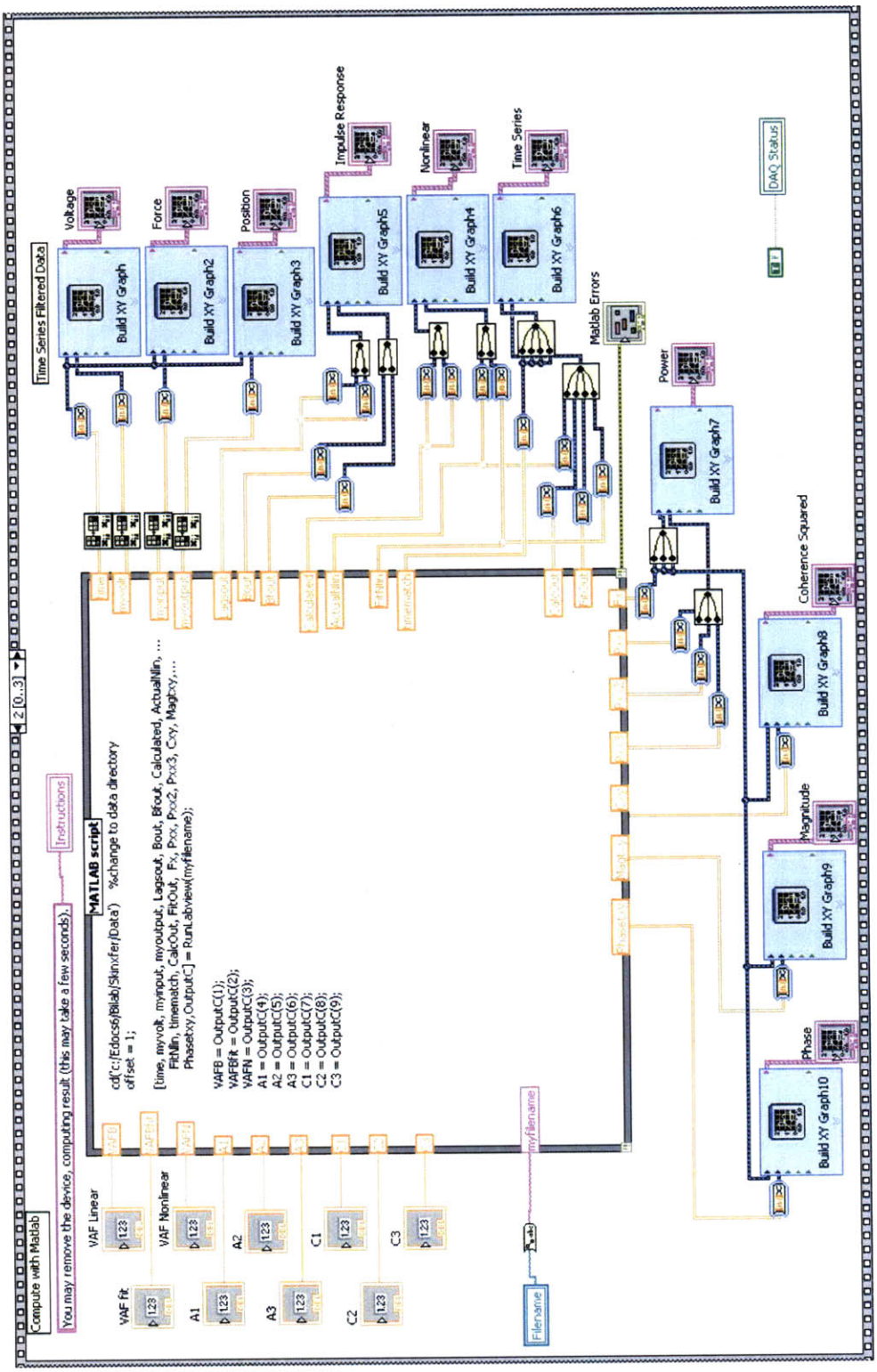


Figure B.7: System identification LabVIEW program analysis panel. The code for the MATLAB program is available in Appendix C.2.3.

Appendix C

MATLAB

Selected MATLAB code is shown in this appendix. The code that is displayed includes code for linear system identification, Wiener static nonlinearity identification, Volterra kernel identification, partitioning identification, and input generation.

C.1 Linear System Identification Routines

C.1.1 Myimp.m and MyLMS.m

This piece of code implements linear system identification algorithms.

```
function [B, SeriesB, stdevs, Lags, condition] = myimp(input, output, window, noverlap, nfft, type)

param  = 1e7;
if nargin<5
    type = 'normal';        %Normal Algorithm
end

if strcmpi(type,'one')    %Initial impulse response
    counta = 1;
    shift = [0];
elseif strcmpi(type,'cond')  %Nonoverlapping sections
    counta = floor((size(input,1)-noverlap)/nfft);
    shift = [0];
elseif strcmpi(type,'condmulti')  %Overlapping sections
    counta = floor((size(input,1)-noverlap)/nfft);
    shift = [0 nfft/2];
end

if strcmpi(type,'normal')        %Normal Algorithm
    [B, Lags, condition] = myLMS(input-mean(input),output-mean(output),nfft,'SVD');
    B = B';
    SeriesB = 0;
    stdevs = 0;
else
    ii = 0;
    for k = 1:length(shift)
        analyseinput = input(1+shift(k):end);
        analyseoutput = output(1+shift(k):end);
        for i = 0:counta-1
            ii = ii+1;
            startpoint = i*nfft-noverlap;
            if startpoint<1
                startpoint = 1;
            end
            myinput = analyseinput(startpoint:startpoint+nfft-1).*window;
            myoutput = analyseoutput(startpoint:startpoint+nfft-1).*window;
            [OutB(:,ii) , Lags, condition(:, ii)] = myLMS(myinput-mean(myinput),myoutput-mean(myoutput),nfft,'SVD');
```

```
        end
    end
    SeriesB = OutB;
    if strcmpi(type,'one')
        B = OutB';
        stdevs = zeros(1, size(OutB,1));
    elseif strcmpi(type,'cond') || strcmpi(type,'condmulti')
        j = 0;
        for i = 1:size(OutB,2)
            if condition(i)<param
                j = j+1;
                B2(:,j) = OutB(:,i);
            end
        end
        B = sum(B2')/j;
        stdevs = std(B2');
    elseif strcmpi(type,'normal')
        B = sum(OutB')/size(OutB,2);
        stdevs = std(OutB');
    end
end


function [B, Lags, condition] = myLMS(f,d,M,solver)
N = length(f);
[phiff, lagsf] = xcorr(f, 'biased');
phifd = xcorr(d,f, 'biased');

rff = phiff(N:N+M-1);
R = toeplitz(rff);
P = phifd(N:N+M-1);

if strcmpi(solver,'fast')
    B = R\P;
elseif strcmpi(solver,'SVD')
    B = pinv(R)*P;
end

Lags = lagsf(N:N+M-1);
condition = cond(R);
```

C.1.2 MyALS.m

This piece of code implements the Adaptive Least Squares (ALS) algorithm.

```
function [b,lags,yhat] = myALS(input,output, parameters, type)
close all

%Adaptive least squares/Moving least squares
lambda=parameters.L;
M=parameters.M;
Fs=parameters.Fs;
mylength = length(input);

f_history = zeros(1,M);
b = zeros(1,M);
b(1) = 0;
y = 0;
yhat = zeros(mylength,1);
e = zeros(mylength,1);
b = zeros(mylength, M);
f_delay = zeros(1,1);

for J = 1:mylength
    f_delay(1:-1:2) = f_delay([1:-1:2]-1);
    f_delay(1) = input(J);
    f_history(M:-1:2) = f_history([M:-1:2]-1);
    f_history(1) = f_delay(1);
    y = 0;
    y=sum(b(1:M)*f_history(1:M));
    e = d-y;
    b(1:M) = b(1:M) + lambda*e*f_history(1:M);
    b(J,:) = b1;
end
lags = 0:length(M)/Fs;

end
```

C.1.3 MyRLS.m

This piece of code implements the Recursive Least Squares (RLS) algorithm.

```
function [b,lags,yhat] = myRLS(input,output, parameters, type)
lambda =parameters.lambda;
Fs = parameters.Fs;
M = parameters.m;

mylength = length(input);
delta = 2*100*var(input); %greater than 100*var(input)
mydelay = 0;

e = zeros( mylength,1);
f_delay = zeros(mydelay+1,1);
yhat = zeros(mylength,1);

%intialize
F = zeros(M,1);
Rinv = delta*eye(M);
B = zeros(M,1);

for J = 1:mylength
    f_delay(2:end) = f_delay(1:end-1);      %shift f_delay backwards
    f_delay(1)=input(J);                    %add f(J) to first spot
    F(2:end) = F(1:end-1);                  %Shift F backwards
    F(1) = f_delay(mydelay+1);              %add f to first spot

    yhat(J) = F'*B;
    error = output(J)-yhat(J);

    K = Rinv*F/(lambda+F'*Rinv*F);          %calculate Kalman gain
    Rinvn = (Rinv-K*F'*Rinv)/lambda;

    B = B+K*error;
    b = B;
    Rinv = Rinvn;
    e(J) = output(J)-yhat(J);
end

lags = 0:length(M)/Fs;

end
```

C.1.4 BioModel.m

This piece of code implements discrete models of biological tissues including linear models, Wiener static nonlinearities, Hammerstein static nonlinearities, and dynamic parameter nonlinearities (DPN).

```
function output = BioModel(input)
        if nargin == 0
            close all
            clear all
        end

        %Define Parameters
        Struct.M = 0.06;
        Struct.B = 20;
        Struct.K = 1000;
        Struct.C1 = 7;
        Struct.C2 = 0.2;
        Struct.C3 = 0;
        Struct.A1 = 0;
        Struct.A2 = 0.3;
        Struct.A3 = 0.3;
        Struct.D1 = 0;
        Struct.D2 = 0;
        Struct.D3 = 0.5;
        Struct.Fscale = 1000;
        Struct.x = 0;
        Struct.xdot = 0;
        Struct.xout =0;
        Struct.type = 'DPN'; %'Linear', 'Wiener', 'Hammer', 'DPN'
        Struct.inputtype = 'FS'; % 'FS', 'test01', 'uniform'
        Struct.offset = -0.1;
        Struct.gain = 1;

        if nargin == 0
```

```
            %Define Input
            [Struct, input, timestep, Fs]=defineinput(Struct);
            %Compute Result with Model
             [input, output, Fs] = DynamicSim(Struct, input, timestep, Fs);
             startcut = Struct.startcut;
             input = input(startcut:end);
             output = output(startcut:end);
        else
            startcut = 1;
            Fs = 500; timestep = 1/Fs;
            %Compute Result with Model
            [input, output, Fs] = DynamicSim(Struct, input, timestep, Fs);
            [Models.LMS, CalcOut, Pout.LMS] = myLMS(Struct, input, output, Fs, 'plot');
            output = output';
        end
end

function [Struct, input, timestep, Fs]=defineinput(Struct)
        %Input Parameters

        if strcmp(Struct.inputtype, 'FS') %Full Spectrum
            Fs = 500; timestep = 1/Fs;
            maxtime = 16;
            timespan = [ 0 : timestep : maxtime ];
            randn('state', 1);
            input = Struct.gain*randn(1, length(timespan))+Struct.offset; %Random
            Struct.startcut = 100;
        elseif strcmp(Struct.inputtype, 'test01') %Shaped Inputs
            ShapedInput_test01; input = Struct.gain*shapedinput+Struct.offset; timestep = sampling;
            Struct.startcut = 919*2; %Cut off beginning of signal
        elseif strcmp(Struct.inputtype, 'uniform')
            ShapedInput_uniform; input = Struct.gain*shapedinput+Struct.offset; timestep = sampling;
            Struct.startcut = 919*2; %Cut off beginning of signal
        end
end


function [myforce, mypos, Fs]=DynamicSim(Struct, input, timestep, Fs)
        timespan = [1:1:length(input)]*timestep;
        split = 1;
        timestepc = timestep/split;

        k = 1;
        for j = 1:length(timespan)       %System ID loop
            for jj = 1:split %Continuous Domain Loop
                realforce(k) = input(j);
                Struct = SetVoltage(Struct, timestepc, realforce(k));
                realpos(k) = Struct.xout;
                timespanc(k) = k*timestepc;
                k = k + 1;
            end
            myforce(j) = realforce(k-split);
            mypos(j) = realpos(k-split);
        end
end

function Struct = SetVoltage( Struct, timestep, input)
        x = Struct.x;
        xdot = Struct.xdot;

        if strcmp(Struct.type, 'Linear') %Linear Model
            Struct.x = x + (xdot)*timestep ;
            Struct.xdot = xdot + (-Struct.B*xdot-Struct.K*x+input*Struct.Fscale)/Struct.M*timestep;
            Struct.xout = Struct.x;
        elseif strcmp(Struct.type, 'Wiener') %Wiener Static Model
            Struct.x = x + (xdot)*timestep ;
            Struct.xdot = xdot + (-Struct.B*xdot-Struct.K*x+input*Struct.Fscale)/Struct.M*timestep;
               Struct.xout = Struct.C1*(1-exp(-Struct.C2*(x + (xdot)*timestep)+Struct.C3));
        elseif strcmp(Struct.type, 'Hammer') %Hammerstein Static Model
            y = Struct.C1*(1-exp(-Struct.C2*input+Struct.C3));
            Struct.x = x + (xdot)*timestep ;
            Struct.xdot = xdot + (-Struct.B*xdot-Struct.K*x+y*Struct.Fscale)/Struct.M*timestep;
            Struct.xout = Struct.x;
        elseif strcmp(Struct.type, 'DPN') %DPN Model
            Struct.x = x + (xdot)*timestep ;
            Struct.xdot = xdot + (-((exp(Struct.A2*x)+Struct.D2)*Struct.B)*xdot...
                -((exp(Struct.A3*x)+Struct.D3)*Struct.K)*x+input*Struct.Fscale)...
                /((exp(Struct.A1*x)+Struct.D1)*Struct.M)*timestep;
            Struct.xout = Struct.x;
        end
end
```

C.2 Wiener Static Nonlinearity Routines

C.2.1 Loaddataiterate.m

This piece of code loads the data and iterates a solution through the Sequence.m program.

```
close all
clear all
graphoverwrite = 1; %1 turns off all graphs
fileoverwrite = 1;

modelactual = 0; %0=acutal, 1=model

iterategraph = 0; %LoadData graphs
iterategraph2 = 0; %Sequence graphs

if modelactual == 0
    iteratefile = 'Development\09_11_05\LPf_uniform1.lvm';
    LoadData;
else
    ShapedInput_test01
    LoadModel
end
iteration = 0;

loops = 6;
for i = 1:loops
    Sequence;
    Output(i,:) = [iteration VAFB VAFBfit VAFN zeta wn M C K C1 C2 C3];

    if (i == 1) %Initial Plots
        figure('Color','w','Name', 'Impulse Response');
        plot(Lags/Fs,B*Fs/abs(sum(Bfit)), '.-'); grid on; hold on
        plot(Lags/Fs,Bfit*Fs/abs(sum(Bfit)), 'r');
        xlabel('Time'); ylabel('Magnitude'); title('Impulse Response')
        legend('Non Parametric', 'Second Order TF')
        prettyfigure;

        figure('Color','w','Name', 'Nonlinearity');
        plot(CalcOut2, myoriginaloutput(end-length(CalcOut2)+1:end), '.'); hold on
        plot(CalcOut2, outest, '.r', 'Markersize', 6)
        xlabel('Predicted Linear Output'); ylabel('Acutal Output')
        title('Static Nonlinearity'); legend('Predicted Linear', 'Nonlinear Fit')
        prettyfigure;
    end
end

%turns off the overwrite
graphoverwrite = 0;
fileoverwrite = 0;
```

C.2.2 Sequence.m

This piece of code calculates the Wiener static nonlinearity and makes other linear estimates.

```
showpower = 1;
staticnonlin = 1;
iteron = 1;
plotpriority = 2;  %0=no plots, 1=nonlinear/dynamic plots only, 2=all plots

if graphoverwrite ==1
    plotpriority = iterategraph2; %no graphs
end

if plotpriority>=1
    close all
end

if iteron ==1 && iteration ==0
    myoriginaloutput = myoutput;    %uses original output first time
    myoriginalinput = myinput;
elseif iteron ~= 1
    myoriginaloutput = myoutput;    %always use provided output
    myoriginalinput = myinput;
end
```

```
if plotpriority>0
figure('Color','w', 'Name', 'Input/Output')
subplot(2,1,1); plot(time,myinput,'r')
ylabel('Input'); grid on
subplot(2,1,2); plot(time,myoutput, 'b','MarkerSize',1)
ylabel('Output'); grid on
end

noverlap0 = 120;
nfft = 2*noverlap0;
mywindow = ones(nfft,1);
[B, SeriesB, StdB, Lags, condition] = myimp(myinput-mean(myinput),myoutput-mean(myoutput), mywindow, noverlap0, nfft, 'normal');
atry = [0.8; 100; -40];
try
    [Bfit, ahat]=myfit(Lags/Fs, B, atry);
end

%Plot impulse
if plotpriority>0
figure('Color','w','Name', 'Impulse Response'); hold on
plot(Lags/Fs,B*Fs/abs(sum(Bfit)), '.-'); grid on; hold on
plot(Lags/Fs,Bfit*Fs/abs(sum(Bfit)), 'r');
xlabel('Time'); ylabel('Magnitude');
legend('Non Parametric', 'Second Order TF')
prettyfigure;
end

%Convolve input with filter B
nhat =100;
Bhat = Bhat2(1,1:nhat);
CalcOut = convn(myinput-mean(myinput), B', 'valid')+mean(myinput);
CalcOutHat = convn(myinput-mean(myinput), Bhat', 'valid')+mean(myinput);
FitOut = convn(myinput-mean(myinput),Bfit', 'valid')+mean(myinput);

if plotpriority>1
figure('Color','w', 'Name', 'Time Series Matching')
plot(time(1:size(myoutput,1)), myoutput-mean(myoutput), 'b');
hold on
plot(time(size(B,2):end), CalcOut-mean(CalcOut), 'r');
plot(time(size(Bhat,2):end), CalcOutHat-mean(CalcOutHat), 'm');
plot(time(size(Bfit,2):end), FitOut-mean(FitOut), 'g');
grid on
xlabel('Time (s)'); ylabel('Output');
legend('Actual Measurement', 'Predicted Measurement (Non Parametric)','Predicted Measurement 2 (Non Parametric)',...
        'Predicted Measurement (Parametric)')
prettyfigure
end

%Measure Error
VAF1 = VAF(myoutput(size(B,2):end)-mean(myoutput(size(B,2):end)), CalcOut-mean(CalcOut));
VAF2 = VAF(myoutput(size(Bhat,2):end)-mean(myoutput(size(Bhat,2):end)), CalcOutHat-mean(CalcOutHat));
VAF3 = VAF(myoutput(size(Bfit,2):end)-mean(myoutput(size(Bfit,2):end)), FitOut-mean(FitOut));

AIC1= AIC(myoutput(size(B,2):end)-mean(myoutput(size(B,2):end)), CalcOut-mean(CalcOut), B);
if iteron ==1 && iteration ~=0
    CalcOut0 = convn(myoriginalinput-mean(myoriginalinput), B', 'valid')+mean(myoriginalinput);
    FitOut0 = convn(myoriginalinput-mean(myoriginalinput),Bfit', 'valid')+mean(myoriginalinput);

    VAF10 = VAF(myoriginaloutput(end-length(CalcOut0)+1:end)-mean(myoriginaloutput(end-length(CalcOut0)+1:end)), CalcOut0-mean(CalcOut0));
    VAF30 = VAF(myoriginaloutput(end-length(FitOut0)+1:end)-mean(myoriginaloutput(end-length(FitOut0)+1:end)), FitOut0-mean(FitOut0));
end

if staticnonlin == 1
    Bused = B;
    %Estimation of static nonlinearity
    uinput = mean(myinput);
    CalcOut2 = uinput+ 1/Fs*convn(myinput-uinput, Bused'*Fs/abs(sum(Bused)), 'valid');
    if sum(Bused)<0
        Flag = 'Area under impulse response is negative'
    end

    %Static Nonlinearity
        offsetzero = min(offsetzero, min(myoriginaloutput));

        predicted = @(c,xdat) c(1).*(1-exp(-c(2)*(xdat+c(3))))+offsetzero;
        c0 =[3,0.3,1];

        options = optimset('MaxFunEvals', 1000, 'TolFun', 1*10^(-7), 'LargeScale', 'on');
        [chat, resnorm, residual, exitflag, output, lambda, jacobian]=...
            lsqcurvefit(predicted, c0, CalcOut2, myoriginaloutput(end-length(CalcOut2)+1:end), [], [], options);
        outest = chat(1)*(1-exp(-chat(2)*(CalcOut2+chat(3))))+offsetzero;

        if plotpriority>0
        figure('Color','w','Name', 'Nonlinearity');
        plot(CalcOut2, myoriginaloutput(end-length(CalcOut2)+1:end), '.'); hold on
        plot(CalcOut2, outest, '.r', 'Markersize', 6)
        xlabel('Predicted Linear Output (N)'); ylabel('Measured Output (mm)')
        legend('Predicted Linear', 'Nonlinear Fit', 'Location', 'SouthEast')
        prettyfigure
        end
```

```
        VAF4 = VAF(myoriginaloutput(end-length(CalcOut2)+1:end)-mean(myoriginaloutput(end-length(CalcOut2)+1:end)), outest-mean(outest));
        AIC4 = AIC(myoriginaloutput(end-length(CalcOut2)+1:end)-mean(myoriginaloutput(end-length(CalcOut2)+1:end)), outest-mean(outest), [B 0 0]);
        if iteron ==1 && iteration ~=0
            CalcOutOriginal = mean(myoriginalinput)+ 1/Fs*convn(myoriginalinput-mean(myoriginalinput), Bused'*Fs/abs(sum(Bused)), 'valid');
            outestoriginal = chat(1)*(1-exp(-chat(2)*(CalcOutOriginal+chat(3))))+offsetzero;
            VAF4O =  VAF(myoriginaloutput(end-length(outestoriginal)+1:end)...
                    -mean(myoriginaloutput(end-length(outestoriginal)+1:end)), outestoriginal-mean(outestoriginal));
        end
end


if showpower == 1
    noverlap = 1000;
    nfft = 2*noverlap;
    mywindow =hanning(nfft);

    %Coherence Plot
    [Cxy,Fc] = mscohere(myinput-mean(myinput), myoutput-mean(myoutput),mywindow,noverlap,nfft,Fs);
    %Power Plot
    [Pxx,Fpx] = pwelch(myvolt-mean(myvolt), mywindow,noverlap,nfft,Fs);
    [Pxx2,Fpx2] = pwelch(myinput-mean(myinput), mywindow,noverlap,nfft,Fs);
    [Pxx3,Fpx3] = pwelch(myoutput-mean(myoutput), mywindow,noverlap,nfft,Fs);

    if plotpriority>1
    figure('Color','w', 'Name', 'Coherence and Input Power'); subplot(2,1,1);
    semilogx(Fc, Cxy, 'LineWidth', 2); grid on
    xlabel('frequency (Hz)'); ylabel('Mean Squared Coherence');

    subplot(2,1,2);
    loglog(Fpx, Pxx, 'LineWidth', 2); hold on;
    loglog(Fpx2, Pxx2,  'r', 'LineWidth', 2);
    loglog(Fpx3, Pxx3, 'k', 'LineWidth', 2); grid on
    legend('Voltage Measured', 'Input', 'Output')
    xlabel('Frequency (Hz) '); ylabel('Power Spectral Density');
    end

    [txy, Ft] = tfestimate(myinput-mean(myinput), myoutput-mean(myoutput),mywindow,noverlap,nfft,Fs);
    if plotpriority>1
    figure('Color','w', 'Name', 'TF Estimate')
    subplot(2,1,1);loglog(Ft, abs(txy), 'LineWidth', 2); grid on
    xlabel('Frequency (Hz)'); ylabel('Magnitude');
    subplot(2,1,2);semilogx(Ft, unwrap(angle(txy))*180/pi, 'LineWidth', 2); grid on
    xlabel('Frequency (Hz)'); ylabel('Phase'); ylim([-270 90])
    end
end

zeta = sqrt(ahat(3)^2/(ahat(3)^2+ahat(2)^2));
wn = ahat(2)/sqrt(1-zeta^2);
Zeta_wn = [zeta wn]

VAFB = VAF1;
VAFBfit = VAF3;
VAFN = VAF4;
AICN = AIC4;
Current_VAF = [VAFB VAFBfit VAFN]

if iteron ==1 && iteration ~=0
    VAFBO = VAF1O;
    VAFBfitO = VAF3O;
    VAFNO = VAF4O;
    Original_VAF = [VAFBO VAFBfitO VAFNO]
end

Fit_Parameters = ahat'
K = 1000/ahat(1)*wn/sqrt(1-zeta^2)/(Fs/abs(sum(Bfit)));
M = K/wn^2;
C = 2*zeta*wn*M;
Linear_Parameters_Scaled = [K M C]

K2 = 1000/ahat(1)*wn/sqrt(1-zeta^2)/Fs;
M2 = K2/wn^2;
C2 = 2*zeta*wn*M2;
Linear_Parameters = [K2 M2 C2]

C1 = chat(1);
C2 = chat(2);
Nonlin_Parameters = [C1 C2]

C2actual = chat(2);

if iteron == 1;
    iteration = iteration+1;
    myoutputest = -1/C2actual*log(1-(myoriginaloutput-offsetzero)/chat(1))-C3actual;
    myoutput = myoutputest;
    time = time(end-length(myoutput)+1:end);
    myinput = myinput(end-length(myoutput)+1:end);
end

LocalizedLinearOutput = [noverlapO atry(3) min(pos(startcut:end)) max(pos(startcut:end)) range(pos(startcut:end)) param Current_VAF ...
    Fit_Parameters Zeta_wn Linear_Parameters Linear_Parameters_Scaled]
```

C.2.3 RunLabview.m

This piece of code is used by the LabVIEW program to complete fast Wiener static system identification.

```
function [time, myvolt, myinput, myoutput, Lagsout, Bout, Bfout, Calculated, ActualNlin, ...
       FitNlin, timematch, CalcOutput, FitOutput, Fx, Pxx, Pxx2, Pxx3, Cxy, Magtxy, Phasetxy, OutputC] = RunLabview(getname)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Read %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
file = getname;
u = importdata(file, '\t', 24);
[y,indexer]=max(isnan(u.data(:, 6)));
if y ==0, indexer = size(u.data,1); end
timein = u.data(1:indexer-1, 1);
pos = u.data(1:indexer-1, 4);
force = u.data(1:indexer-1, 6);
input = u.data(1:indexer-1, 2);

sampling = timein(2)-timein(1); %Seconds
Fs = 1/sampling;
offsetzero = pos(1000);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Implement filters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
startcut = 2000; %Cut off beginning of signal
graph = 0; %no
drifttype = 'none'; %linear,

%Impement drift filter
[posout, param] = driftfilter(pos,input,drifttype, startcut, graph, Fs);

%Implement input frequency filter
band = [10 20];
freqtype = 'none';
mymean = 1; %subtract off the mean
postprocess = 1; %truncate series to valid section
[myinput] = myfilter(force(startcut:end), Fs, band, mymean, freqtype, graph, postprocess); %-force
[myoutput] = myfilter(posout(startcut:end), Fs, band, mymean, freqtype, graph, postprocess); %posout

time = timein(end-size(myinput)+1:end);
myvolt = input(end-size(myinput)+1:end);
offsetzero = pos(500);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Linear System Identification%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
noverlap = 120;
nfft = 2*noverlap;
mywindow = ones(nfft,1);
[B, SeriesB, StdB, Lags, condition] = myimp(myinput-mean(myinput),myoutput-mean(myoutput), mywindow, noverlap, nfft, 'normal'); %
[Bfit, ahat]=myfit(Lags/Fs, B, [0.8; 100; -200]);

CalcOut = convn(myinput-mean(myinput), B', 'valid')+mean(myoutput);
FitOut = convn(myinput-mean(myinput),Bfit', 'valid')+mean(myoutput);

Lagsout = Lags/Fs;
Bout = B*Fs/abs(sum(Bfit));
Bfout = Bfit*Fs/abs(sum(Bfit));
CalcOutput = CalcOut';
FitOutput = FitOut';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Nonlinearity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Bused = Bfit; %B or Bfit
CalcOut2 = mean(myinput)+ 1/Fs*convn(myinput-mean(myinput), Bused'*Fs/abs(sum(Bused)), 'valid');
if sum(Bused)<0
    Flag = 'Area under impulse response is negative'
end
offset2 = min(myoutput(size(Bused, 2):end));
predicted = @(c,xdat) c(1).*(1-exp(-c(2)*(xdat+c(3))))+offsetzero;
c0 =[1,1,1];
options = optimset('MaxFunEvals', 1000, 'TolFun', 1*10^(-7), 'LargeScale', 'on');
[chat, resnorm, residual, exitflag, output, lambda, jacobian]=...
    lsqcurvefit(predicted, c0, CalcOut2, myoutput(size(Bused,2):end), [], [], options);
outest = chat(1)*(1-exp(-chat(2)*(CalcOut2+chat(3))))+offsetzero;

Calculated = CalcOut2';
ActualNlin = myoutput(size(Bused,2):end)';
FitNlin = outest';
timematch = time(size(Bused,2):end)';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Frequency Domain%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
noverlap = 1000;
nfft = 2*noverlap;
mywindow =hanning(nfft);
```

```
%Coherence Plot
[Cxy,Fx] = mscohere(myinput-mean(myinput), myoutput-mean(myoutput),mywindow,noverlap,nfft,1/sampling);

%Power Plot
[Pxx,Fx] = pwelch(myvolt-mean(myvolt), mywindow,noverlap,nfft,1/sampling);
[Pxx2,Fx] = pwelch(myinput-mean(myinput), mywindow,noverlap,nfft,1/sampling);
[Pxx3,Fx] = pwelch(myoutput-mean(myoutput), mywindow,noverlap,nfft,1/sampling);

%Frequency Domain
[txy, Fx] = tfestimate(myinput-mean(myinput), myoutput-mean(myoutput),mywindow,noverlap,nfft,1/sampling);
Magtxy = abs(txy)';
Phasetxy = unwrap(angle(txy))'*180/pi;

Fx = Fx';
Pxx = Pxx';
Pxx2 = Pxx2';
Pxx3 = Pxx3';
Cxy = Cxy';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Outputs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

VAFB = VAF(myoutput(size(B,2):end)-mean(myoutput(size(B,2):end)), CalcOut-mean(CalcOut));
VAFBfit = VAF(myoutput(size(Bfit,2):end)-mean(myoutput(size(Bfit,2):end)), FitOut-mean(FitOut));
VAFN = VAF(myoutput(size(Bused,2):end)-mean(myoutput(size(Bused,2):end)), outest-mean(outest));

amp = 5;
Gain = 1000;
R = 3.288;
Bl = 1.4083;
zeta = sqrt(ahat(3)^2/(ahat(3)^2+ahat(2)^2));
wn = ahat(2)/sqrt(1-zeta^2);

k0 = wn/sqrt(1-zeta^2)*Bl*amp*Gain/(ahat(1)*Fs/sum(Bfit)*R);
m0 = k0/(wn^2);
b0 = (2*zeta*wn*m0*R-Bl^2)/R;
A1 = ahat(1);
A2 = ahat(2);
A3 = ahat(3);
C1 = chat(1);
C2 = chat(2)/(Bl*amp*Gain/(k0*R));
C3 = chat(3);
m1 = C3/9.8;
k1 = wn^2*m1;

OutputC = [VAFB, VAFBfit, VAFN, A1, A2, A3, C1, C2, C3];
```

C.3 Volterra Kernel and Partitioning Routines

C.3.1 MyVolterra.m

This piece of code implements Volterra and partitioning system identification with measured and modeled data.

```
function myVoltera
close all

I = 40;  %memory length
type = 'h2'; %h1, h2, hp, or hpd
parameters.num = 7;     %number of sections
parameters.break = 'sort';

[x,y,Fs, B]=LoadInfo('actual');
[P, OutStructure, time1]=CreateBase(type,x,y,I,Fs, parameters);
[Alpha, Beta, Gamma, A, timeout]=OrthogSolve(P, y, I, type);
time2 = timeout(1); time3 = timeout(2); time4 = timeout(3);
[h_struct, time5]=ResolveKernel(type, A, I);
time = [0 time1 time2 time3 time4 time5];
PlotInfo(type, h_struct, OutStructure, A, P, x, y, Fs, I, B, time)
end

function [x,y,Fs,B]=LoadInfo(type)
%create model%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if strcmp(type,'model')
        Fs = 500;
        mylength = 3000;
        impLength = 60;

        s = tf('s');
        wn = 150
```

```
        zeta = 0.5;
        a =  5;

        H = a*wn^2/(s^2+2*zeta*wn*s+wn^2);
        B = impulse(H, (0:impLength-1)/Fs)/Fs;

        %Declare Functions
        c=1;
        none = @(c,x) c*x;
        square = @(c,x) c*(x).^2;
        cube = @(c,x) c.*(0.1*x.^3- x.^2 + x);
        quad = @(c,x) c.*x.^4;
        exponent = @(c,x) c*(10.257*(1-exp(-0.2112.*x)));
        exponentb = @(c,x) c*(7*(exp(1.5.*x)));

        myfunctionH = none;
        myfunctionW = square;

        %Implement Nonlinearities
        randn('state', 1);          %Seed randomization

        x = 1.2*myidinput(mylength,'rgs',[0 0.95],[-1 1], [],8)+0*randn(mylength,1);

        nonlinearinput = myfunctionH(c, x);
        linearOutput = convn(nonlinearinput, B, 'full');
        linearOutput = linearOutput(1:mylength);
        nonlinearOutput = myfunctionW(c, linearOutput);
        y = nonlinearOutput+20;

    elseif strcmp(type,'actual')
        B = 0;

        %actual linear data
        file = 'Development\09_11_05\LAf2_uniform1.lvm';'Test01\s05_LA_04.lvm';
        Fs = 2000;
        u = importdata(file, '\t', 24);
        [y,indexer]=max(isnan(u.data(:, 6)));
        if y ==0, indexer = size(u.data,1); end
        myoutput = u.data(1:indexer-1, 4);      %output position
        myinput = u.data(1:indexer-1, 6);      %input force
        myvolt = u.data(1:indexer-1, 2);    %input voltage

        %Impement drift filter
        startcut = 919*2;%Cut off beginning of signal
        myinput = myinput(startcut:end);
        myoutput = myoutput(startcut:end);
        myvolt = myvolt(startcut:end);

        downrate = 3;
        Fs = Fs/downrate;
        x = downsample(myinput,downrate);
        y = downsample(myoutput,downrate);
        length(y)

    end
        figure('Color', 'w');
        noverlap = 500;
        nfft = 2*noverlap;
        mywindow =hanning(nfft);
        [Pxx,Fpx] = pwelch(x-mean(x), mywindow,noverlap,nfft,Fs);
        [Pxx2,Fpx2] = pwelch(y-mean(y), mywindow,noverlap,nfft,Fs);
        loglog(Fpx, Pxx, 'LineWidth', 2); hold on;
        loglog(Fpx2, Pxx2,  'r', 'LineWidth', 2);
        legend('Input', 'Output')
        xlabel('Frequency (Hz) '); ylabel('Power Spectral Density');
end

function [P, OutStructure, time1]=CreateBase(type,x,y,I,Fs,parameters)
    %use input and output info%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    tic
    N = length(x);% %%record length

    if strcmp(type, 'h1')
        M = 1+(I+1);
        P = zeros(M,N);

        P(1,I+1:N) = ones(1,N-I);  %zeroth order P, size 1

        for n = I+1:N
            for m = 2:I+2            %first order P, size I+1
                P(m,n) = x(n-m+2);
            end
        end
        OutStructure.myrange =0;
    elseif strcmp(type, 'h2')
        M = 1+(I+1)+(I+1)*(I+2)/2;
        P = zeros(M,N);

        P(1,I+1:N) = ones(1,N-I);  %zeroth order P, size 1

        for n = I+1:N
            for m = 2:I+2            %first order P, size I+1
```

```
            P(m,n) = x(n-m+2);
        end
    end

    m = I+2;                    %second order P, size (I+1)*(I+2)/2
    for i1 = 1:I+1;
        for i2= i1:I+1;
            m=m+1;
            for n = I+1:N
                P(m,n)=x(n-i1+1)*x(n-i2+1);
            end
        end
    end
    OutStructure.myrange =0;
elseif strcmp(type, 'hp')

    myy = zeros(size(y));
    average = 0;
    for ii = 1:length(y)-average
        myy(ii+average)= mean(y(ii:ii+average));
    end
    for ii = 1:average
        myy(ii) = mean(y(1:ii));
    end

    sections = parameters.num;
    breaktype = parameters.break;
    if strcmp(breaktype, 'sort')
        sdelta = floor(length(myy(I+1:end))/sections);
        ssorted = sort(myy(I+1:end));
        if sections >1
            myrange = [min(myy(I+1:end)), ssorted(sdelta.*(1:(sections-1)))', max(myy(I+1:end))];
        elseif sections == 1
            myrange = [min(myy(I+1:end)),  max(myy(I+1:end))];
        end
        for i = 0:sections-1
            mymedian(i+1) = mean(ssorted(i*sdelta+1:(i+1)*sdelta));
        end
        OutStructure.mymedian = mymedian;
    elseif strcmp(breaktype, 'even')
        myrange = min(myy(I+1:end)):range(myy(I+1:end))/sections:max(myy(I+1:end));
        OutStructure.mymedian = (myrange(2:end)+myrange(1:end-1))/2;
    end
    OutStructure.myrange = myrange;

    M = 1+sections*(I+1);
    P = zeros(M,N);

    P(1,I+1:N) = ones(1,N-I);  %zeroth order P, size 1

    m = 1;                      %second order P, size (I+1)*(I+2)/2
    for i2= 1:sections;
        for i1 = 1:I+1;
            m=m+1;
            for n = I+1:N
                if (myy(n-i1+1)>=myrange(i2)) && (myy(n-i1+1)<=myrange(i2+1))
                    P(m,n)=x(n-i1+1);
                end
            end
        end
    end

    figure('Color','w');
    runtime = (1:length(y))/Fs;
    plot(runtime, y); hold on
    plot(runtime, myy, 'r');
    plot([zeros(1,sections+1); length(y)/Fs*ones(1,sections+1)],[myrange; myrange], ':k');
    ylabel('Position (mm)'); xlabel('Time (s)')

elseif strcmp(type, 'hpd')

    myy = zeros(size(y));
    average = 0;
    for ii = 1:length(y)-average
        myy(ii+average)= mean(y(ii:ii+average));
    end
    for ii = 1:average
        myy(ii) = mean(y(1:ii));
    end
    diffmyy = [0; diff(myy)];

    sections = parameters.num;
    breaktype = parameters.break;
    if strcmp(breaktype, 'sort')
        sdelta = floor(length(myy(I+1:end))/sections);
        ssorted = sort(myy(I+1:end));
        if sections >1
            myrange = [min(myy(I+1:end)), ssorted(sdelta.*(1:(sections-1)))', max(myy(I+1:end))];
        elseif sections == 1
            myrange = [min(myy(I+1:end)),  max(myy(I+1:end))];
        end
        for i = 0:sections-1
```

```
                mymedian(i+1) = mean(ssorted(i*sdelta+1:(i+1)*sdelta));
            end
            OutStructure.mymedian = [mymedian mymedian];
        elseif strcmp(breaktype, 'even')
            myrange = min(myy(I+1:end)):range(myy(I+1:end))/sections:max(myy(I+1:end));
            mymedian= (myrange(2:end)+myrange(1:end-1))/2;
            OutStructure.mymedian  = [mymedian mymedian];
        end
        OutStructure.myrange = myrange;

        M = 1+sections*(I+1);
        P = zeros(M,N);

        P(1,I+1:N) = ones(1,N-I);  %zeroth order P, size 1

        m = 1;                     %second order P, size (I+1)*(I+2)/2
        for d = [-1 1]
            for i2= 1:sections;
                for i1 = 1:I+1;
                    m=m+1;
                    for n = I+1:N
                        if (myy(n-i1+1)>=myrange(i2)) && (myy(n-i1+1)<=myrange(i2+1) && sign(diffmyy(n-i1+1))==d)
                            P(m,n)=x(n-i1+1);
                        end
                    end
                end
            end
        end

        figure('Color','w');
        runtime = (1:length(y))/Fs;
        plot(runtime, y); hold on
        plot(runtime, myy, 'r');
        plot([zeros(1,sections+1); length(y)/Fs*ones(1,sections+1)],[myrange; myrange], ':k');
        ylabel('Position (mm)'); xlabel('Time (s)')

        figure('Color','w');
        mesh(P);
    end
    time1 = toc;
end


function [Alpha, Beta, Gamma, A, timeout]=OrthogSolve(P, y, I, type)
    %Orthogonalize (Find P, Beta,Alpha)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    M = size(P,1);
    N = size(P,2);
    R = M;
    Alpha = zeros(M,R);
    Beta = zeros(R,N);
    BetaSquare = zeros(R);
    Gamma = zeros(M,1);
    V = zeros(M,1);
    A = zeros(M,1);
    P_orthogonal= P;  %PO(m,n) = P(m,n) for m = 1:M
    Epsilon = 1e-10;  %use for stability

    for r = 1:R
        Beta(r,(I+1):N) = P_orthogonal(r,(I+1):N);
        BetaSquare(r) = sum(Beta(r,(I+1):N).^2);
        if BetaSquare(r)<Epsilon
            BetaSquare(r) = BetaSquare(r)+Epsilon;
        end
        for m = r+1:M
            Alpha(m,r) = sum(P_orthogonal(m,(I+1):N).*Beta(r,(I+1):N))/BetaSquare(r);
        end
        P_orthogonal(r:M,(I+1):N)=P_orthogonal(r:M,(I+1):N)-Alpha(r:M,r)*P_orthogonal(r,(I+1):N);
    end
    time2 = toc;
    %Determine Gamma%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for m = 1:M
        Gamma(m) = sum(y((I+1):N).*Beta(m,(I+1):N)')/BetaSquare(m);
    end
    time3 = toc;
    %Determine A and V%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for m = 1:M
        %Create V
        V(m) = 1;
        for i = m+1:M
            V(i) = -sum(Alpha(i,m:i).*V(m:i)');
        end
        %Create A
          A(m) = sum(Gamma(m:M).*V(m:M));

    end
    time4 = toc;

    timeout = [time2 time3 time4];
end

function [h_struct, time5]=ResolveKernel(type, A, I)
```

```
%Determine h's%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if strcmp(type, 'h1')
        h0 = A(1);
        h1= A(2:I+2);
        h2 = zeros(I+1,I+1);

        h_struct.h0 = h0;
        h_struct.h1 = h1;
        h_struct.h2 = h2;
    elseif strcmp(type, 'h2')
        h2 = zeros(I+1,I+1);

        h0 = A(1);

        h1= A(2:I+2);

        m = I+2;
        for i1 = 1:I+1
            for i2 = i1:I+1;
                m = m+1;
                h2(i1,i2)=A(m);
                if i1 ~= i2
                    h2(i1,i2)=0.5*h2(i1,i2);
                end
            end
        end

        h_struct.h0 = h0;
        h_struct.h1 = h1;
        h_struct.h2 = h2;

    elseif strcmp(type, 'hp')
        h0 = A(1);

        sections = (length(A)-1)/(I+1);
        hp = zeros(sections, I+1);
        m = 1;
        for i1 = 1:sections
            for i2 = 1:I+1;
                m = m+1;
                hp(i1,i2)=A(m);
            end
        end

        h_struct.h0 = h0;
        h_struct.hp = hp;

    elseif strcmp(type, 'hpd')
        h0 = A(1);

        doublesections = (length(A)-1)/(I+1);
        hp = zeros(doublesections, I+1);
        m = 1;
        for i1 = 1:doublesections
            for i2 = 1:I+1;
                m = m+1;
                hp(i1,i2)=A(m);
            end
        end

        h_struct.h0 = h0;
        h_struct.hp = hp;
    elseif strcmp(type, 'hpm')

        sections = (length(A))/(I+1+1);
        h0 = A(1:sections);
        hp = zeros(sections, I+1);
        m = sections;
        for i1 = 1:sections
            for i2 = 1:I+1;
                m = m+1;
                hp(i1,i2)=A(m);
            end
        end

        h_struct.h0 = h0;
        h_struct.hp = hp;
    end

    time5 = toc;
end

function PlotInfo(type, h_struct, OutStructure, A, P, x, y, Fs, I, B, time)
    M = size(P,1);
    N = size(P,2);

    %Algorithm Time
    figure('Color', 'w');
    mydiff= diff(time);grid on;
    bar(mydiff); colormap summer
    ylabel('Time (s)')
    text(4,0.8*time(6), ['Total Time ' num2str(time(6)) ' s']);
```

```
    set(gca,'XTickLabel',{'Create Basis', 'Orthogonalize', 'Identify Gamma', 'Reconstruct', 'Derive Kernel'})
    prettyfigure;

    %Calculating VAF
    yest = zeros(N,1);
    for n = 1:N
        yest(n) = sum(A(1:M).*P(1:M,n));
    end
    VAF_est = VAF(y(I+1:N), yest(I+1:N));
    AIC_est = AIC(y(I+1:N), yest(I+1:N), A);

% Type hp or hpd %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if (strcmp(type, 'hp')) || (strcmp(type, 'hpd')) )
        h0 = h_struct.h0;
        hp = h_struct.hp;

        myrange = OutStructure.myrange;
        xmesh = myrange(2:end)'*ones(1,size(hp,2));
        ymesh = ones(size(hp,1),1)*(0:I)/Fs;
        Lags = (0:I)/Fs;

        %Mesh Plot
        if size(hp,1) >1 && ( strcmp(type, 'hp'))
            figure('Color', 'w'); grid on;
            mesh(xmesh, ymesh, hp);
            xlabel('Position (mm)'); ylabel('Lags (s)'); zlabel('Magnitude')
            prettyfigure;
        end

        %Fit MBK to the impulse response
        atry = [100; 0.8; -200];
        for i = 1:size(hp,1)
            hpplot(i,:) = hp(i,:);
            [Bfit(i,:), ahat(:,i)]=myfit(Lags, hpplot(i,:), atry);
        end
        zeta = sqrt(ahat(3,:).^2./(ahat(3,:).^2+ahat(2,:).^2));
        wn = ahat(2,:)./sqrt(1-zeta.^2);
        K = 1000./ahat(1,:).*wn./sqrt(1-zeta.^2)/Fs;
        Mass = K./wn.^2;
        C = 2*zeta.*wn.*Mass;
        Gain = 1000./Mass./wn.^2;
        averange = (myrange(2:end)+myrange(1:end-1))/2;
        mymedian = OutStructure.mymedian;

        if strcmp(type, 'hp')
            figure('Color', 'w');
            subplot(1,3,1); plot(mymedian, Mass, '.', 'MarkerSize', 15);xlabel('Position (mm)'); ylabel('M (kg)'); grid on;
            subplot(1,3,2); plot(mymedian, C, '.', 'MarkerSize', 15);xlabel('Position (mm)'); ylabel('B (Ns/m)'); grid on;
            subplot(1,3,3); plot(mymedian, K, '.', 'MarkerSize', 15);xlabel('Position (mm)'); ylabel('K (N/m)'); grid on;
        elseif strcmp(type, 'hpd')
            sections = (length(A)-1)/(I+1)/2;
            figure('Color', 'w');
            subplot(1,3,1);
            plot([mymedian(1:sections); mymedian(sections+1:end)]', [Mass(1:sections); Mass(sections+1:end)]', '.', 'MarkerSize', 15);
            xlabel('Position (mm)'); ylabel('M (kg)'); grid on;
            legend('Lift off Skin', 'Go Into Skin');
            subplot(1,3,2);
            plot([mymedian(1:sections); mymedian(sections+1:end)]', [C(1:sections); C(sections+1:end)]', '.', 'MarkerSize', 15);
            xlabel('Position (mm)'); ylabel('B (Ns/m)'); grid on;
            subplot(1,3,3);
            plot([mymedian(1:sections); mymedian(sections+1:end)]', [K(1:sections); K(sections+1:end)]', '.', 'MarkerSize', 15);
            xlabel('Position (mm)'); ylabel('K (N/m)'); grid on;
        end

        %Plot unexplained nonlinearity
        figure('Color', 'w'); grid on; hold on;
        plot(yest(I+1:N),y(I+1:N), 'b');
        plot([min(y) max(y)],[min(y) max(y)], 'k', 'LineWidth', 2);
        xlabel('Estimate'); ylabel('Measured Output (mm)');
        prettyfigure;

        %Plot Output Relation
        figure('Color', 'w'); grid on; hold on;
        time = (1:N)/Fs;
        plot(time, y, 'b');
        plot(time, yest, 'r');
        xlabel('Time (s)'); ylabel('Output');
        legend('Output', 'AP Estimate');
        text( 0.5, 0.1*max(y), {['VAF = ' num2str(VAF_est) '
        prettyfigure;
%Type h1 and h2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    elseif strcmp(type,'h1') || strcmp(type,'h2')
        h0 = h_struct.h0;
        h1 = h_struct.h1;
        h2 = h_struct.h2;

        %Get original impulse response
        B_Kernel=zeros(I+1,1);
        B_h2=zeros(I+1,1);
        for t = 1:I+1
            B_Kernel(t) = h0+h1(t)+h2(t,t);
            B_h2(t) = h2(t,t);
```

```
end

%Plot Wiener nonlinearity
figure('Color', 'w'); grid on; hold on;
yh1 = convn(x, h1,'full');
yh1 = yh1(1:N);
yh2 = convn(x, B_h2,'full');
yh2 = yh2(1:N);
yh2b = convn(x, h2(2, 1:end-1)', 'full');
yh2b = yh2b(1:N);

plot(yh1(I+1:N),y(I+1:N), 'c');
plot(yh2(I+1:N),y(I+1:N), 'g');
plot(yh1(I+1:N)+yh2(I+1:N),y(I+1:N), 'r');
plot(yest(I+1:N),y(I+1:N), 'b');
xlabel('Estimate'); ylabel('Measured Output');
legend('h1 Estimate', 'h2 Estimate', 'h1+h2 Estimate', 'Total Estimate')

%Plot impulse responses
figure('Color', 'w'); grid on;hold on;
Lags = (0:I)/Fs;
LagsB = (0:length(B)-1)/Fs;
plot(LagsB,B,'b');
plot(Lags,h1,'c');
plot(Lags,B_h2,'g');
plot(Lags,h1+B_h2,'r');
xlabel('Lags (s)'); ylabel('Magnitude');
xlim([0 0.1])
legend('True h', 'h1','h2','h1+h2');

%Plot Kernels
Lags = (0:I)/Fs;
figure('Color', 'w'); grid on; hold on;
plot(Lags, h1, '.');
xlabel('Lags (s)'); ylabel('Magnitude')

atry = [0.8; 100; -100];
[Bfit, ahat]=myfit(Lags, h1', atry);
zeta = sqrt(ahat(3).^2./(ahat(3).^2+ahat(2).^2));
wn = ahat(2)./sqrt(1-zeta.^2);
K = 1000./ahat(1).*wn./sqrt(1-zeta.^2)/Fs;
Mass = K./wn.^2;
C = 2*zeta.*wn.*Mass;

h2full = h2+h2';
for j = 1:length(h2full);
    h2full(j,j) = 0.5*h2full(j,j);
end

figure('Color', 'w'); grid on; hold on;
surf(Lags, Lags, h2full); xlabel('Lags (s)'); ylabel('Lags (s)')

%Low pass filter
ffth2 = fftshift(fft2(h2full));
figure('Color', 'w'); surf(Lags, Lags,abs(ffth2)); xlabel('Lags (s)'); ylabel('Lags (s)')
linearfilter = fspecial('gaussian', 15, 1);
h2out=filter2(linearfilter,h2full);
figure('Color', 'w'); surf(Lags, Lags,h2out); xlabel('Lags (s)'); ylabel('Lags (s)')
fh2 = h2out;

myA = zeros(size(A));
myA(1) = h0;
myA(2:I+2)=h1;
if strcmp(type,'h2')
    m = I+2;
    for i1 = 1:I+1
        for i2 = i1:I+1;
            m = m+1;
            myA(m)=fh2(i1,i2);
            if i1 ~= i2
                myA(m)=2*myA(m);
            end
        end
    end
end

h2filtered = zeros(I+1,I+1);
h0filtered = myA(1);
h1filtered= myA(2:I+2);
m = I+2;
for i1 = 1:I+1
    for i2 = i1:I+1;
        m = m+1;
        h2filtered(i1,i2)=myA(m);
        if i1 ~= i2
            h2filtered(i1,i2)=0.5*h2filtered(i1,i2);
        end
    end
end
yestfiltered = zeros(N,1);
for n = 1:N
    yestfiltered(n) = sum(myA(1:M).*P(1:M,n));
```

```
        end

        VAF_estfiltered = VAF(y(I+1:N), yestfiltered(I+1:N));
        AIC_estfiltered = AIC(y(I+1:N), yestfiltered(I+1:N), myA);

        %Plot Output Relation
        figure('Color', 'w'); grid on; hold on;
        time = (1:N)/Fs;
        plot(time, y, 'b');
        plot(time, yest, 'r');
        plot(time, yestfiltered, 'g');
        xlabel('Time (s)'); ylabel('Output');
        legend('Output', 'AP Estimate', 'Filtered AP');
        text( 0.5, 0.1*max(y), {['VAF = ' num2str(VAF_est) '
            ['AIC = ' num2str(AIC_est) ]; ['AIC filtered = ' num2str(AIC_estfiltered) ]})
        prettyfigure;

        %Plot Unexplained Nonlinearity
        figure('Color', 'w'); grid on; hold on;
        plot(yest(I+1:N),y(I+1:N), 'b');
        plot([min(y) max(y)],[min(y) max(y)], 'k', 'LineWidth', 2);
        xlabel('Estimate'); ylabel('Measured Output (mm)');
        prettyfigure;

    end
end
```

C.4 Input Generation Routines

C.4.1 MyCustomInput.m

This piece of code creates the custom input with a jointly specified PDF and auto-correlation.

```
function myCustomInput
close all
tic

PDFtype = 'uniform';
ACFtype = 'power';
dLength = 10000;
dLengthfinal = dLength;
ssLength = 500;
Fs = 500;
cutoff = 200;
plottype = 2; %1=summary, 2=separate, 3 = none,
savefile = 1; %save output

gui_active(1);
h = progressbar( [],0,'Program Progress' );

%Initialize Data
x = 1e-5:1/(dLength):1-1e-5;

%desired PDF and spits out CDF
if strcmp(PDFtype,'gaussian')
    myOut = sqrt(2)*erfinv(x*2-1);
    myOut = myOut-mean(myOut);
elseif strcmp(PDFtype, 'uniform')
    myOut = x*3.4641;
    myOut = myOut-mean(myOut);
elseif strcmp(PDFtype, 'binary')
    myOut = [zeros(1, dLength/2) 2*ones(1, dLength/2)];
    myOut = myOut-mean(myOut);
elseif strcmp(PDFtype, 'exponential')
    beta = 1.01;
    myOut = -beta*log(1-x);
    myOut = myOut-mean(myOut);
end

%desired ACF and spits it out
if strcmp(ACFtype, 'white')
    desAcorr = [1 zeros(1, dLength-1)];
elseif strcmp(ACFtype, 'exponential')
    alpha = 100;
    desAcorr = exp(-alpha*x);
elseif strcmp(ACFtype, 'order2')
    wn = 100;
    zeta = 0.3;
    desAcorr = exp(-wn*zeta*x).*cos(wn*sqrt(1-zeta^2)*x);
elseif strcmp(ACFtype, 'SI')
```

```
    SI=idinput(dLength,'rgs',[0 2*cuttoff/Fs],[], [])';
    desAcorr = myAutoCorr(SI-mean(SI));
elseif strcmp(ACFtype, 'power')
    fp = cutoff*2*dLengthfinal/Fs;
    mag = 1.24e-3*Fs/dLengthfinal;
    for sig = 1:dLength
        if sig<=fp
            myy(sig)= mag;
        else
            myy(sig) = mag/(4*(sig-fp+1/4));
        end
    end
    myy2 = real(fft([myy(1,1:1:end-1) myy(1,end:-1:1)]));
    desAcorr = myy2(1:ceil(length(myy2)/2));
end
myOutPlot = myOut;

%scramble (double stochastic interchange)
myOut = swapsweep(myOut);
countss = 1;
myAcorr = myAutoCorr(myOut);
myAcorr = myAcorr(1:ssLength);
myAcorr3 = myAcorr;
SS(countss) = sum((myAcorr(1:ssLength)-desAcorr(1:ssLength)).^2);
iterations = 1000000;
numSwaps = ones(1,iterations);
countss = 2;

for kk = 1:iterations
    [myOut2, jout] = swap(myOut);
    D = zeros(1, ssLength);
    for k = 2:ssLength
        if jout(1)+(k-1)<=dLength && jout(1)+(k-1) ~=jout(2)
            D(k) = D(k)-myOut(jout(1))*myOut(jout(1)+(k-1))+myOut(jout(2))*myOut(jout(1)+(k-1));
        end
        if jout(1)-(k-1)>=1 && jout(1)-(k-1) ~=jout(2)
            D(k) = D(k)-myOut(jout(1))*myOut(jout(1)-(k-1))+myOut(jout(2))*myOut(jout(1)-(k-1));
        end
        if jout(2)+(k-1)<=dLength && jout(2)+(k-1) ~=jout(1)
            D(k) = D(k)-myOut(jout(2))*myOut(jout(2)+(k-1))+myOut(jout(1))*myOut(jout(2)+(k-1));
        end
        if jout(2)-(k-1)>=1 && jout(2)-(k-1) ~=jout(1)
            D(k) = D(k)-myOut(jout(2))*myOut(jout(2)-(k-1))+myOut(jout(1))*myOut(jout(2)-(k-1));
        end
    end

    myAcorr = myAcorr3 + D/dLength;

    SS(countss) = sum((myAcorr(1:ssLength)-desAcorr(1:ssLength)).^2);
    if SS(countss)<=SS(countss-1) %if error decreases
        myOut = myOut2;
        myAcorr3 = myAcorr;
        countss = countss+1;
    else %if error does not decrease
        numSwaps(countss) = numSwaps(countss)+1;
    end
    if rem(kk,100) == 0;
        h = progressbar( h,100/(iterations*1.2) );
    end

end

%Fit to SS
predicted = @(c,xdat) c(1).*(exp(-c(2)*(xdat)))+c(3);
c0 =[max(SS)-min(SS),0.001,min(SS)];
options = optimset('MaxFunEvals', 1000, 'TolFun', 1*10^(-7), 'LargeScale', 'on');
[chat, resnorm, residual, exitflag, output, lambda, jacobian]=...
    lsqcurvefit(predicted, c0, 1:countss-1, SS(1:countss-1), [], [], options);
SSfit = @(c,xdat) chat(1).*(exp(-chat(2)*(1:countss-1)))+chat(3);

%Compare to shaped input
SI=idinput(dLength,'rgs',[0 0.2],[], [])';
siAcorr=myAutoCorr(SI-mean(SI));

%Power
noverlap = 500;
nfft = 2*noverlap;
mywindow =hanning(nfft);
[Pxx,Fpx] = pwelch(myOut-mean(myOut), mywindow,noverlap,nfft,Fs);
[siPxx,siFpx] = pwelch(SI-mean(SI), mywindow,noverlap,nfft,Fs);
if strcmp(ACFtype, 'power')
    desPxx = myy(4:end);
    desFpx = (4:length(myy))/2*Fs/dLengthfinal;
else
    desPxx0 = real(ifft(desAcorr));
    desPxx = desPxx0(1:dLength/2);
    desFpx = 1:length(desPxx);
end

%Histogram
[histSI,sSI] = histogram(SI, 50);
[histOutPlot,sOutPlot] = histogram(myOutPlot, 50);
```

```
%plot PDF, CDF, ACF
if plottype ~=3
    if plottype == 1; figure('Color', 'w','Name', 'Summary Plot'); end
    if plottype ==1; subplot(4,2,1); elseif plottype==2; figure('Color', 'w','Name', 'PDF'); end
        plot(sOutPlot, histOutPlot, 'r'); hold on
        plot(sSI, histSI, 'k');
        xlabel('x'), ylabel('PDF'); legend('Actual', 'idInput')
    if plottype ==1; subplot(4,2,3); elseif plottype==2; figure('Color', 'w','Name', 'inverse CDF'); end
        plot(x, myOutPlot, 'b');
        xlabel('F(x)'); ylabel('inverse CDF');
    if plottype ==1; subplot(2,2,2); elseif plottype==2; figure('Color', 'w','Name', 'AutoCorrelation'); end
        plot(myAcorr, 'r','LineWidth', 2); hold on
        plot(desAcorr, 'b', 'LineWidth', 2); grid on
        plot(siAcorr, 'k');
        plot([ssLength ssLength], [-0.4 1.2], ':m','LineWidth', 2);
        xlabel('lags'); ylabel('AutoCorrelation');
        legend('Actual Output', 'Desired', 'idInput')
    if plottype ==1; subplot(4,2,5);  elseif plottype==2; figure('Color', 'w','Name', 'Sum of Squares'); end
        [AX,H1,H2] = plotyy(1:countss-1,SS(1:countss-1)/max(SS)*100,1:countss-1, numSwaps(1,1:countss-1));
        hold on
        set(H1,'LineWidth',2); set(H1,'Color', 'c')
        set(H2,'Color', 'g')
        plot(1:countss-1,SSfit(1:countss-1)/max(SS)*100, 'k', 'LineWidth', 2);
        set(get(AX(1),'Ylabel'),'String','
        set(get(AX(2),'Ylabel'),'String','Iterations before Swap')
        xlabel('Sucessful Interchanges');
        legend('Iterations before Swap', '
        text(floor(length(SS)/3),100-0.50*(100-min(SS/max(SS)*100)), {['SS=C_1\ite^{C_2x}+C_3'], ['C_1=', num2str(chat(1))],...
            ['C_2=', num2str(chat(2))],[ 'C_3=', num2str(chat(3))]})
    if plottype ==1; subplot(4,2,7);  elseif plottype==2; figure('Color', 'w','Name', 'Log-Linear Sum of Squares'); end
        semilogy(1:countss-1,SS(1:countss-1), 'c','LineWidth', 2); hold on
        semilogy(1:countss-1,SSfit(1:countss-1), 'k','LineWidth', 2);
        xlabel('Sucessful Interchanges'); ylabel('Sum of Squares');
        legend('Sum of Squares', 'Fit to SS')
    if plottype ==1; subplot(4,2,6); elseif plottype==2; figure('Color', 'w','Name', 'Series'); end
        plot(myOut, 'r'); hold on;
        plot(SI, 'k');
        xlabel('Time'); ylabel('Output'); legend('Actual Output', 'idInput')
    if plottype ==1; subplot(4,2,8); elseif plottype==2; figure('Color', 'w','Name', 'Power'); end
        loglog(Fpx, Pxx, 'r', 'LineWidth', 2); hold on;
        loglog(desFpx, desPxx, 'b','LineWidth', 2);
        loglog(siFpx, siPxx, 'k', 'LineWidth', 2);  ylim([1e-7 1e-1])
        xlabel('Frequency (Hz) '); ylabel('Power');     legend('Actual Output', 'Desired', 'idInput', 'Location', 'SouthWest')
end

if savefile ==1        %Else we don't overwrite the file
    s = xlswrite('myCustomInput.xls', [(1:length(myOut))' myOut']);
    save 'myCustomInput.mat' myOut SS chat;
end
progressbar( h,-1 );
toc
end

function [myOut jout] = swap(myIn)
    N = length(myIn);
    i = ceil(rand(1)*N);
    j = ceil(rand(1)*N);
    swap = myIn(i);
    myIn(i) = myIn(j);
    myIn(j) = swap;
    jout = [i j];
    myOut = myIn;
end

function myOut = swapsweep(myIn)
    N = length(myIn);
    for count = 1:N
        i = ceil(rand(1)*N);
        swap = myIn(count);
        myIn(count) = myIn(i);
        myIn(i) = swap;
    end
    myOut = myIn;
end

function [out,series] = histogram(in,groups)
    mymin = min(in);
    mymax = max(in);
    delta = (mymax-mymin)/groups;
    out = zeros(groups+2,1);
    series = mymin-delta:delta:mymax;
    for i = 2:groups+1
        for j = 1:length(in)
            if (in(j)>=min(in)+(i-1)*delta) && (in(j)<=min(in)+i*delta)
                out(i) = out(i)+1;
            end
        end
    end
    out = out/(delta*length(in));
end
```

```
function [Output] = myAutoCorr(Input)
%Single sided autocorrelation
N = length(Input);
AACF = zeros(1,N);
for k = 1:N
    for j = 1:N-k+1
        AACF(k) = AACF(k)+Input(j)*Input(j+k-1);
    end
end
Output = 1/N*AACF;
end
```

C.4.2 RealTimeInput.m

This piece of code simulates real time input generation (RTIG) and incorporates a real time ALS algorithm.

```
function RealTimeInput
close all; clear all;
%Parameters %%%%%%%%%
system.Fs = 2000;       %Hz
system.maxtime = 5;    %seconds
system.ipdfmin = -2;    %System input limits
system.ipdfmax = 25;     %System input limits
system.opdfmin = 0;   %System output limits
system.opdfmax = 6;     %System output limits
Struct = StructureConstructor; %Define Model

%Input Generation Algorithms
system.plot = 'yes';        %Use dynamic plotting
system.id = 'yes';        %Use or not use ALS system ID
system.wait = 50;        %Feedback cycle time
system.delta = (system.opdfmax-system.opdfmin)/30;
system.mout = 0;
system.bin = zeros((system.opdfmax-system.opdfmin)/system.delta+1,1); %manages output range
system.bincount = zeros((system.opdfmax-system.opdfmin)/system.delta+1,1);
system.binK = 1000*ones((system.opdfmax-system.opdfmin)/system.delta+1,1);%keeps parameter values
system.binB = 10*ones((system.opdfmax-system.opdfmin)/system.delta+1,1);%keeps parameter values
system.binM = 0.05*ones((system.opdfmax-system.opdfmin)/system.delta+1,1);%keeps parameter values
system.num = 0;
system.numold = 0;
system.kk = 0;
system.ii = 0;
system.change = 0;

%Loop %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:1:system.maxtime*system.Fs
    %Get input from algorithm
    if k==1
        [input(k), system] = GenInput(0, 0, system, Struct, 'initial'); tic
    else
        [input(k), system] = GenInput(input,output, system, Struct, 'normal');
    end
    %Send to system
    Struct = SetVoltage( Struct, 1/system.Fs, input(k));
    %Process output
    output(k) = Struct.xout;
    %Execute real-time system ID
end
toc

%Implement Offline System ID %%%%%%%%%%%%
[y_LMS, CalcOut, Pout] = LMSWiener(Struct, system, input, output, system.Fs, 'plot');
end


function Struct = StructureConstructor
%Define Model Parameters %%%%%%
Struct.M = 0.08;
Struct.B = 8;
Struct.K = 1000;
Struct.C1 = 7;
Struct.C2 = 0.2;
Struct.C3 = 0;
Struct.A1 = 0;
Struct.A2 = 0.15;
Struct.A3 = 0.2;
Struct.D1 = 0;
Struct.D2 = 0;
Struct.D3 = 0.1;
Struct.Fscale = 1000;
Struct.x = 0;
```

```
Struct.xdot = 0;
Struct.xout =0;
Struct.type = 'Wiener'; %'Linear', 'Wiener', 'Hammer', 'DPN'
Struct.inputtype = 'FS';
Struct.offset = 3;
Struct.gain = 7;
end


function [input, system] = GenInput(input, output, system, Struct, condition)
    if strcmp(condition, 'initial')
        rand('state', 1);
        randn('state', 1);
        system.oldinput= 10;
        system.numold = 1;
        system.num = 1;
        ALScalc(zeros(system.wait, 1), zeros(system.wait, 1), system, 'initial');
        if strcmp(system.plot, 'yes')
            figure('Color', 'w')
        end
    end
    %Check the output every feedback cycle
    if system.kk == system.wait+1
        binin=zeros(system.wait,1);
        for j = 1:system.wait
            binin(j) = round((output(end-j))/system.delta+(-system.opdfmin)/system.delta)+1;
            if binin(j)<1, binin(j)=1; system.penaltyb = system.penaltyb+1; end
            if binin(j)>length(system.bin), binin(j) = length(system.bin); system.penaltyt = system.penaltyt+1; end
            system.bin(binin(j)) = system.bin(binin(j))+1;
        end
        system.oldinput = mean(input(end-system.wait:end));
        system.numold = system.num;     %dump old desired location
        [system.num]=min(system.bin);
        system.bincount(system.num) = system.bincount(system.num)+1;
        system.mout = mean(binin);
        system.kk = 0;
        if strcmp(system.id, 'yes')
            [M,B,K] =  ALScalc(input(end-system.wait:end), output(end-system.wait:end), system, 'normal');
            system.binM(round(system.mout)) = M;
            system.binB(round(system.mout)) = B;
            system.binK(round(system.mout)) = K;
            system.change = K/1000;
        else
            system.change =1-min(abs(system.mout-system.num)/(system.opdfmax-system.opdfmin), 0.1); %Scale change with speed
        end

         if strcmp(system.plot,'yes') && strcmp(system.id, 'yes')
            figure(1)
            subplot(3,1,1);plot(output, 'r')
            subplot(3,1,2);plot(system.bin, 'r')
            subplot(3,3,7);plot(system.binM, '.r'); xlabel('Position (mm)'); ylabel('M (kg)'); grid on; prettyfigure; YL = ylim; ylim([0, YL(2)])
            subplot(3,3,8);plot(system.binB, '.r'); xlabel('Position (mm)'); ylabel('B (Ns/m)'); grid on; prettyfigure; YL = ylim; ylim([0, YL(2)])
            subplot(3,3,9);plot(system.binK, '.r'); xlabel('Position (mm)'); ylabel('K (N/m)'); grid on; prettyfigure; YL = ylim; ylim([0, YL(2)])
         elseif strcmp(system.plot,'yes')
            figure(1)
            subplot(2,1,1);plot(output, 'r')
            subplot(2,1,2);plot(system.bin, 'r')
         end
    end

    %Combine for input
    if (system.binK(round(system.num))<mean(system.binK)+200 || system.binK(round(system.num))<1200) || strcmp(system.id, 'no')
            modulator = min(max(system.bincount(system.num)-system.bin(system.num),0.5),3);
        else modulator = 1;
    end
    if strcmp(system.id, 'yes')
         addrand = (rand(1)-0.5);
        input = system.oldinput+1*(-0.2*system.change*modulator*(system.mout-system.num)+4*addrand);
    else
        addrand = (rand(1)-0.5);
        input = system.oldinput+1*(-0.2*system.change*modulator*(system.mout-system.num)+4*system.rands*addrand);
    end

    system.kk =system.kk+1;

    %manages input range
    if input<system.ipdfmin, input = system.ipdfmin; end
    if input>system.ipdfmax, input = system.ipdfmax; end
end


function [Me, Be, Ke] = ALScalc(input, output, system, condition)
persistent iLength M L Fs mylength yhat e b f_delay lags

if strcmp(condition, 'initial')
    iLength = 60;
    M = iLength;
    L = 0.00001;
    Fs = system.Fs;
    mylength = length(input);
    lags = 0:length(M)/Fs;
    x = LSadapt('initial', L, M);
```

```
    yhat = zeros(mylength,1);
    e = zeros(mylength,1);
    b = zeros(mylength, M);
    f_delay = zeros(1,1);
else
    for J = 1:mylength
        f_delay(1:-1:2) = f_delay([1:-1:2]-1);
        f_delay(1) = input(J);
        [yhat(J),B_ALS]=LSadapt(f_delay(1),output(J));
    end
    lags = (0:iLength-1)/Fs;
    atry = [0.8; 100; -100];
    [B_fit, ahat]=myfit(lags, B_ALS, atry);
    zeta = sqrt(ahat(3,:).^2./(ahat(3,:).^2+ahat(2,:).^2));
    wn = ahat(2,:)./sqrt(1-zeta.^2);
    Ke = 1000./ahat(1,:).*wn./sqrt(1-zeta.^2)/Fs;
    Me = Ke./wn.^2;
    Be = 2*zeta.*wn.*Me;
end
end


function [y, bout]=LSadapt(f,d,FIR_M)
persistent f_history b lambda M
if strcmp(f, 'initial')
    lambda = d;
    M = FIR_M;
    f_history = zeros(1,M);
    b = zeros(1,M);
    b(1) = 0;
    y = 0;
else
    f_history(M:-1:2) = f_history([M:-1:2]-1);
    f_history(1) = f_delay(1);
    y = 0;
    y=sum(b(1:M)*f_history(1:M));
    e = d-y;
    b(1:M) = b(1:M) + lambda*e*f_history(1:M);
    bout = b;
end
end
```