Model Parameter Estimation of Atherosclerotic Plaque Mechanical Properties: Calculus-Based and Heuristic Algorithms

by

Ahmad S. Khalil

B.S. Mechanical Engineering
Minor Chemistry
Stanford University, 2002

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 2004

Signature of Author .....
Department of Mechanical Engineering
May 7, 2004

Certified by .....
Dr. Mohammad R. Kaazempur-Mofrad, Research Scientist
Department of Mechanical Engineering and Biological Engineering Division
Thesis Supervisor

Certified by ....
Dr. Roger Kamm, Professor
Department of Mechanical Engineering and Biological Engineering Division
Thesis Supervisor

Accepted by .....
Dr. Ain Sonin, Professor
Department of Mechanical Engineering
Chairman, Department Committee on Graduate Students

Model Parameter Estimation of Atherosclerotic Plaque Mechanical Properties: Calculus-Based and Heuristic Algorithms

by

Ahmad S. Khalil

Submitted to the Department of Mechanical Engineering
On May 7, 2004 in partial fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineeering

ABSTRACT

A sufficient understanding of the pathology that leads to cardiovascular disease is currently deficient. Atherosclerosis is a complex disease that is believed to be initiated and promoted by linked biochemical and biomechanical pathways. This thesis focuses on studying plaque biomechanics because (i) there is a dearth of data on the mechanical behavior of soft arterial tissue yet (ii) it is the biomechanics that is able to provide invaluable insight into patient-specific disease evolution and plaque vulnerability.

Arterial elasticity reconstruction is a venture that combines imaging, elastography, and computational modeling in an effort to build maps of an artery's material properties, ultimately to identify plaques exhibiting stress concentrations and to pinpoint rupture-prone locales.

The inverse elasticity problem was explored extensively and two solution methods are demonstrated. The first is a version of the traditional linear perturbation Gauss-Newton method, which contingent on an appropriate regularization scheme, was able to reconstruct both homogeneous and inhomogeneous distributions including hard and spatially continuous inclusions. The second was an attempt to tackle the inherent and problem-specific limitations associated with such gradient-based searches. With a model reduction of the discrete elasticity parameters into lumped values, such as the plaque components, more robust and adaptive strategies become feasible. A novel combined finite element modeling–genetic algorithm system was implemented that is easily implemented, manages multiple regions of far-reaching modulus, is globally convergent, shows immunity to ill-conditioning, and is expandable to more complex material models and geometries. The implementation of both provides flexibility in the endeavor of arterial elasticity reconstruction as well as potential complementary and joint efforts.

Thesis Supervisor: Mohammad R. Kaazempur-Mofrad

Title: Research Scientist in Mechanical Engineering and Biological Engineering Division

Acknowledgements

I would first like to recognize and thank my advisors, Dr. Mohammad R. Kaazempur-Mofrad and Professor Roger D. Kamm, for sharing with me their expertise and, most importantly, their inspiration. Research, they taught me, requires patience, personal motivation, and perseverance, often more so than intellect and academics. An unofficial advisor deserving of much gratitude is Jung-Hoon Lee, with who I shared many fruitful discussions, without which this thesis would certainly not have been possible.

To my friends, Pete, Gina, Jen, Jorge, Eli, Kathie, and Adam, who struggled and celebrated alongside me, thank you for the camaraderie and for encouraging the blissful and fundamental philosophy that life is more than merely a linear progression from one academic degree to another.

To the roommates and greater communities of the Dutch Oven, Citgoset, and Mother Country, thanks for *always* making me feel at home.

I am eternally indebted to the Compton 6 drawgroup, who was with me in spirit, and of course to my family, who was with me in love.

I consent that a Master's thesis may not be a substantial enough work to dedicate. But the many frustrating obstacles and gloomy late nights I encountered seemed to disprove this, especially since one person was always there to shed light in the darkness. So, if allowable, a dedication to my lovely girlfriend for eternally inspiring me is needed. I can only hope for the opportunity to tribute more of my life's work to you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation for Studying Cardiovascular Disease

Cardiovascular disease is the dysfunction of the heart, arteries, and veins that supply oxygen to vital organs, such as the brain, kidney, and liver. Most commonly a result of atherosclerosis, an inflammatory disease leading to arterial stenosis, cardiovascular disease is associated with numerous life-threatening events, such as myocardial infarction, stroke, abdominal aneurysm, and lower limb ischemia. Not surprisingly, cardiovascular disease continues to be the principal cause of morbidity and mortality in industrialized countries [1, 2].

As illustrated by the below description of atherosclerosis pathology, this disease involves the complex interplay of countless biochemical and biomechanical factors, many of which have yet to be identified and understood. As a result, the overlying motivation for this work is to contribute to the incomplete knowledge of cardiovascular disease progression and prevention.

More specifically, this work focuses on identifying the biomechanical precursors of acute atherosclerosis. Treatment options are certainly not scant, ranging from the non-invasive sort, such as diet and behavioral alterations and pharmaceuticals, to the invasive solutions, such as angioplasty, endarterectomy, and bypass grafting. Due to the relatively lower risk and healthcare cost burdens, the non-invasive treatment options are preferred and can potentially prevent, slow, or reverse atherosclerosis development. However, their

efficacy is strongly dependent on *early* and *accurate* means of lesion detection and identification. Furthermore, in the case of unstable, vulnerable plaques, invasive treatment may not be beneficial at all and instead may only introduce additional complications and risks.

The hunt for a reliable means of plaque detection and identification has migrated toward biomechanical analysis of plaque for two main reasons: (i) there is a dearth of data and expertise on the mechanical behavior and properties of soft arterial tissue [3] yet (ii) the biomechanics of plaque matter and plaque rupture can provide invaluable and trustworthy insight into patient-specific atherosclerosis evolution and its severity.

1.2 Pathogenesis of Atherosclerosis

Historically, atherogenesis has been portrayed as simply the accumulation of lipids within the arterial wall because high levels of low-density lipoprotein (LDL) are a major risk factor. This observation, however, is only a small part of the intricate pathological process, and other risk factors do exist. In fact, the disease prospers even in patients with relatively low plasma concentrations of LDL.

Atherosclerosis is an inflammatory disease that occurs in medium to large vessels of the arterial system, and thus the observable tissue lesions are a result of a collective series of molecular and cellular responses. To date, one of the most accepted explanations for its onset is the response-to-injury hypothesis, where endothelial dysfunction resulting from injury alters the normal homeostatic conditions of the endothelium [4]. This triggers an inflammatory response whereby cycles of accumulation and activation of mononuclear cells, migration and proliferation of smooth muscle cells, and formation of fibrous tissue

gradually assemble and enlarge an atherosclerotic lesion. Moreover, if the inflammatory response fails to remove the offending agents and defuse the injury, it becomes a chronic harmful pattern.

Endothelial dysfunction due to injury has multiple origins, namely from those recognized as disease risk factors: elevated and modified LDL; free radicals caused by cigarette smoking, hypertension, and diabetes mellitus; genetic inclination or disorders; elevated homocysteine concentrations; and infectious microorganisms such as herpesviruses and *Chlamydia pneumoniae*. While it is beyond the scope of this thesis to specify the mechanisms by which each of the above risk factors leads to endothelial injury, a brief depiction of how LDL causes damage is informative to the understanding of the subsequent inflammatory process. Once LDL particles are introduced in an artery, they are progressively oxidized and eventually taken up by macrophages to form lipid peroxides—destructive free radicals—and to aid in the accumulation of agents that form foam cells (lipid-laden monocytes and macrophages) [5-7]. Already, the inflammatory response is evident as macrophages attempt to consume and remove modified LDL in order to protect endothelial and smooth muscle cells from harmful free radical formation [8]. In addition to its primary role of injuring the endothelial layer, modified LDL has the added ability to up-regulate gene expression that promotes replication of monocyte-derived macrophages and that introduces new monocytes into the lesion, thus *further* encouraging chronic inflammation [9]. Hypercholesterolemia was specifically chosen as the illustrative injury example not only because it is a very common stimulus of endothelial dysfunction but also because it blurs the line between sources of injury and the ensuing inflammatory response. In fact, a recent emphasis has been on characterizing

atherogenesis as a disease resulting from the dichotomous and linked effects of both hypercholesterolemia *and* inflammation [10].

Increased and modified LDL is one of various injurious triggers of endothelial cell dysfunction. It is the ensuing inflammatory response that plays the determining role of how an atherosclerotic lesion will progress. As a prologue, it should be noted that certain arterial sites, namely downstream of branches, bifurcations, recirculation zones and curvatures, are more prone to plaque formation because of their characteristically altered blood flow. At these locales, the endothelium experiences increased flow disturbance, recirculation, and thus decreased shear stress, serving to negatively modify its permeability and adherence properties toward lipoproteins, leukocytes, and other plasma constituents [11-13]. Therefore, whether due solely to endothelial damage, or the fluid mechanical nature of the arterial site, or more likely a combination of these two phenomena, the precursor of atherosclerotic lesions is formation of a slew of molecules responsible for the *adherence*, *migration*, *proliferation,* and *accumulation* of monocytes and T cells. More specifically, an increase in endothelial permeability, an up-regulation of leukocyte adhesion molecule expression (such as L-selectin and integrins), an up-regulation of endothelial adhesion molecule expression (such as E-selectin and intercellular adhesion molecule 1), and the migration of leukocytes into the artery wall (facilitated by molecules such as oxidized LDL and macrophage colony-stimulating factor) are all observed (Figure 1-1A). In fact, as an aside, to reiterate the difficulty of pinpointing a single culprit in the initiation and promotion of plaque formation, it should be noted that oxidized LDL and its products, oxidized phospholipids and oxysterols, have more than 10 known functions that negatively feed the inflammatory process [10].

Fatty-streak formation is then first detected after the adherent monocytes migrate into the subendothelium, differentiate into machrophages, and consequently internalize trapped lipids to form foam cells (Figure 1-1B). In addition, these macrophages secrete cytokines and growth factors that promote the migration of smooth muscle cells into the lesion. All the while, the newly-formed lesion mediates T-cell activation, cellular proliferation, platelet adherence and aggregation, and continued leukocyte adherence and migration into the artery wall.

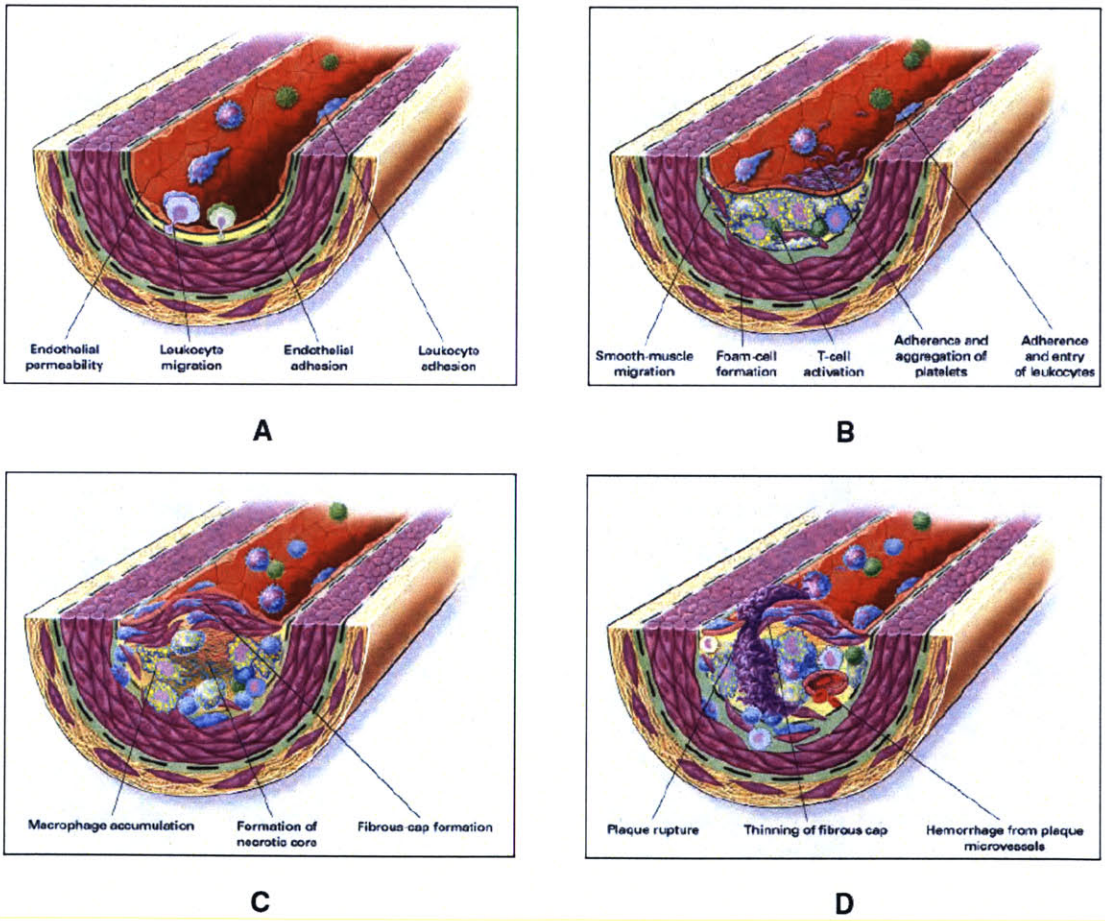


Figure 1-1. Atherogenesis and its progression. A. Initial steps of atherosclerosis following endothelial dysfunction. **B.** Fatty streak formation. **C.** Progression to intermediate and advanced disease. **D.** Unstable fibrocalcific lesion characterized by a large necrotic core and fibrous cap thinning at the lesion shoulders due to locally-concentrated smooth muscle cell loss and proteolytic enzymes release by macrophages. Figures from [4].

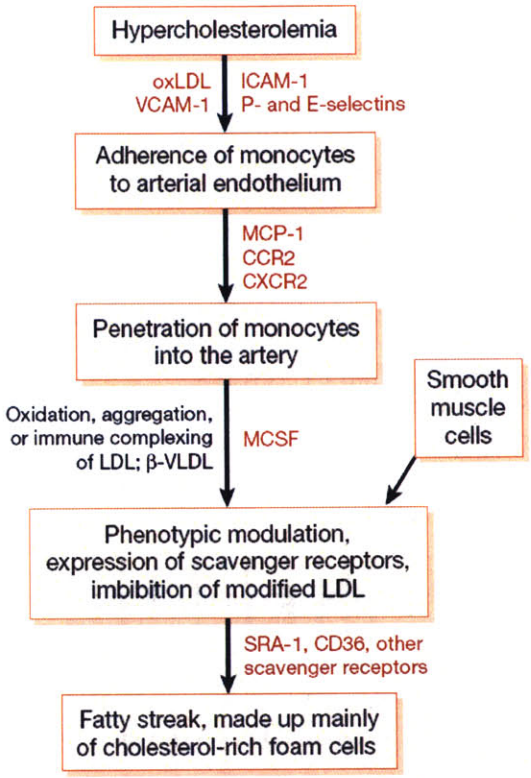


Figure 1-2. Flowchart portraying fatty streak lesion formation. An widely accepted sequence of events leading to fatty streak lesions. Figure from [10].

Typically, the next pathological event occurs when a fibrous cap is formed on top of the lesion that walls it from the lumen, via the release of extracellular matrix components (collagen, laminin, and fibronectin) by the migrant smooth muscle cells (Figure 1-1C). This process signals the onset of an intermediate or advanced lesion, whereby the fibrous cap covers a soft necrotic core—a mixture of leukocytes, lipid, and cellular debris. Meanwhile, the plaque can continue to expand at its shoulders due to macrophage accumulation associated with sustained leukocyte adhesion and entry into the lesion. If the resulting fibrous cap becomes uniformly dense, a *stable* advanced lesion remains; however, commonly the fibrous cap is thinned and weakened resulting in an *unstable* advanced plaque. The macrophages, which play a critical role in virtually every step of

atherogenesis, are responsible: they express matrix metalloproteinases (MMPs) and other proteolytic enzymes that degrade the fibrous wall structural elements. It has been observed that plaques characterized by relatively thin fibrous caps and a high lipid content are mechanically weak and thus prone to rupture [14, 15]. The consequences of rupture include very rapid thrombus formation, intraplaque hemorrhaging, and occlusion of the artery, which not only intensify the injurious inflammatory cycle but also lead to life-threatening events [16, 17] (Figure 1-1D).

1.3 Thesis Goals

Given the complexity and multifaceted nature of atherogenesis, the ability to easily and reliably assess plaque vulnerability in all stages of its pathology would be beneficial to preventing critical cardiac events. One promising avenue is focusing on the *biomechanics* of plaque and plaque rupture.

Understanding the biomechanical progression of plaques to rupture has become of particular interest in light of recent, revealing research. Although it was suspected as early as the 1960s that plaque fissuring could result in acute thrombus formation [18], only recently has plaque rupture been identified and established as a critical aspect of acute cardiac events. In a 1980 autopsy study of human hearts post acute infarction, open cap fissure was observed in a vast majority of the specimens [19], and subsequent studies reported similar trends. These observations are only confirmed with present knowledge of how arteries respond to developing plaques within. As lesions grow, they occupy a larger percentage of the lumen's volume dedicated to blood flow. As a result, arteries compensate via vascular remodeling to preserve the lumen's effective volume; in fact,

histological [20] and non-invasive ultrasound [21] studies of lesions that *gradually* progressed reported both local and generalized vascular dilation. On the other hand, when lesions progress *suddenly*—as a result of complicated or fissuring plaques and consequent, rapid thrombus formation—vascular remodeling is not observed.

With the need to quantify the mechanical behavior of atherosclerotic plaques in the context of physiological stresses established, this thesis focuses on contributing to the difficult task of arterial elasticity imaging. It is widely accepted that the material properties of soft tissue, including arterial tissue, are very specific to the particular specimen or testing setup [3] and vary immensely in the literature (elastic modulus values span four orders of magnitude) [22, 23]. However, it is also accepted that the ability to reliably measure them is important: (i) it provides a map of the locations and magnitudes of high stress concentration (most importantly, high circumferential stress), which have been shown to correspond with regions where plaque rupture tends to transpire [24, 25], (ii) it provides a picture of the mechanical stresses on vascular cells that contribute to the biochemical responses associated with atherogenesis, and (iii) it provides insight into the mechanical behavior of the individual plaque constituents.

Not surprisingly, the mechanical properties of biological tissue are dictated by the structural organization and distribution of its molecular building blocks. This has two consequences. First, diseased tissue will exemplify varied behavior and properties from its corresponding healthy specimen. Second, a truly established groundwork for modeling tissue does not exist. Realistically, a nonlinear, viscoelastic, anisotropic model would capture its behavior most closely. However, quick loading and an appropriate size scale can make a linear elastic, isotropic approximation suitable.

Atherosclerotic arterial plaques are composed of fibrous tissue surrounding a necrotic core of extracellular lipids and debris. For example, the bulk of the matter shown in the histological section of Figure 1-3 is fibrous tissue. The purple lines encircle various lipid pools whereas the green line identifies a calcified plaque. Lipid pools tend to be soft, structurally weak, and highly thrombogenic and thus are major culprits in the occurrence of plaque rupture. Lipid pool size and shape eccentricity have been thoroughly examined, especially within post mortem coronary specimens. Generally, it is found that larger and eccentric lipid pools are associated with critical coronary events [25]. It is not just the geometry but also the material properties of the lipid pool that give rise to harmful stress concentrations in the neighboring fibrous caps. Not surprisingly, lipid-lowering therapy has been shown to produce a significant reduction in cardiac events associated with plaque rupture. This is because lipid-lowering therapy alters the lipid composition (e.g. increasing the concentration of cholesterol monohydrate crystals) such that its stiffness increases, and a stiffer lipid pool is likely to reduce stresses in plaque caps [26]. Mention of plaque cap is important since lipid pool, via its size, shape, and composition, is not likely to trigger plaque rupture without a thin, vulnerable plaque cap. In fact, through structural analysis, thinning of the fibrous cap was shown to dramatically increase peak circumferential stress in plaque [27]. Thus, in many instances of plaque rupture and subsequent thrombosis, a coupling of a lipid-rich region and a thin fibrous cap is observed.

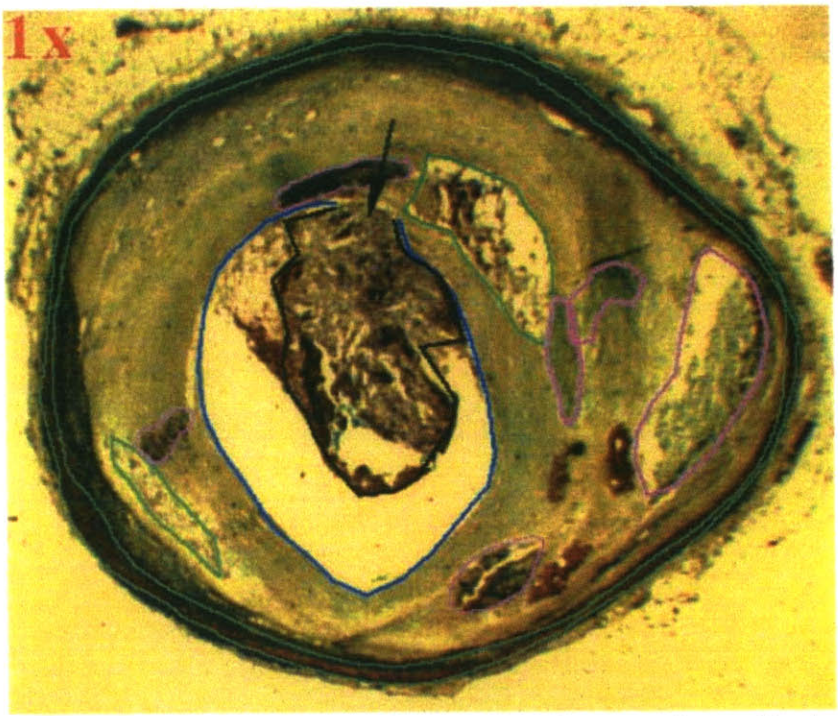


Figure 1-3. Histology of a post mortem coronary artery specimen. Regions of artery wall, fibrous plaque, lipid pool, and calcified plaque were identified in this histological specimen, from which a pre-rupture geometry was reconstructed for modeling and analysis. Figure from [28].

This thesis first identifies and explains the constituents necessary for arterial elasticity imaging. Then it provides a comprehensive look at traditional parameter reconstruction methods for the inverse elasticity problem, specifically the linear perturbation Gauss-Newton method. In addition to presenting a formulation and characterization of this solution, it catalogs some of its major limitations. Finally, it presents an alternative, novel technique for arterial elasticity estimation, a combined finite element modeling (FEM) and genetic algorithm scheme, and discusses its potential and shortcomings.

Chapter 2

Components of Arterial Elasticity Reconstruction

2.1 Arterial Imaging

Assessing tissue mechanical behavior relies on the merger of various efforts. The first is the ability to *image* the arterial tissue in order to (i) obtain its specific geometry and (ii) provide a framework with which to manage elastography experiments. Presented below, for the reader's interest, is a brief overview of some of the relevant invasive and non-invasive imaging modalities used for clinical and/or research study of atherosclerosis.

Angiography

X-ray angiography, the gold standard in coronary artery imaging, is an invasive means of imaging the lumen diameter that involves exposing patients to x-ray radiation and use of a catheter and contrast dye. It is predominately used to detect advanced, occlusive lesions and to visualize their degree of flow obstruction. Because it cannot image the arterial wall and its components, this modality often fails to detect intermediate lesions where vascular remodeling of the artery has compensated for the gradually enlarging and occluding lesion. It should be stressed that x-ray angiography is traditionally only a clinical measure of a vessel's percent stenosis.

Intravascular Ultrasound (IVUS)

IVUS is the only clinically established technique for acquiring real-time cross-sectional images of coronary arteries *in vivo* [29]. It is a catheter-based, and thus invasive, technique that generates cross-sectional images based on the acoustic scattering

properties of the sample. As a result, it allows for visualization of internal plaque structure and reasonable differentiation between tissue components (soft, fibrous, calcified, etc.). It should be noted that identification of certain plaque constituents (e.g. calcified tissue) can be easier than that of others (e.g. lipid and fibrous tissue) [30]. Specifically, IVUS uses high-frequency ultrasound (30-40 MHz) and produces high axial and lateral resolutions of approximately 100μm and 200μm, respectively, with a penetration depth of 4-10mm.

Optical Coherence Tomography (OCT)

OCT is a catheter-based imaging technique, newly developed as an optical analog of IVUS, which harnesses light rather than an acoustic signal [31, 32]. A light source is split into a reference beam and a sample beam. The reference beam is directed toward a mirror whose position is accurately known and the sample beam is directed toward the sample being imaged via a lens. Light returning from both beams is recombined; and based on the ensuing interference fringe the distance traveled by the sample beam can be deduced.

The use and advancement of intravascular OCT is of particular interest to the work in this thesis because, at the expense of depth perception, it offers higher spatial resolution (axial resolutions of 10μm and lateral resolutions of 25μm) and improved contrast of tissue as compared with IVUS. Hence, OCT provides more sensitivity for elastography and elasticity calculations. These advantages are expounded upon later in this thesis.

Surface Magnetic Resonance Imaging (MRI)

Originally exploited for diagnostic purposes, basically as a non-invasive version of angiography, MRI has recently been applied to investigating the arterial wall. Due to improvements in high-resolution fast spin-echo imaging and processing, MRI now offers

in vivo visualization of atherosclerotic plaque structure in human carotid arteries [33]. In fact, several investigators have shown that MRI is capable of differentiating atherosclerotic plaque components, such as lipid pools, fibrous caps, calcification, and acute thrombosis, in human carotid arteries [34-36].

Peripheral Vascular Ultrasound

Peripheral ultrasound is an alternative and lower cost method for non-invasively monitoring atherosclerosis. Clinically, it is combined with Doppler flow imaging to assess the degree of stenosis, plaque morphology, and hemodynamics, key factors in predicting the occurrence of stroke due to carotid atherosclerosis [37, 38]. In addition, non-invasive ultrasound has been shown to distinguish plaque components, such as intraplaque hemorrhage, fibrous regions, and lipid-rich regions, according to echogenecity [39].

2.2 Elastography

Unofficially, physicians have been practicing elastography for many years without the aid of imaging or technologically-advanced experimental apparatus. Instead, they have relied on manual palpation as a diagnostic tool for a variety of diseases based on the fundamental observation that diseases cause changes in tissue mechanical behavior. In fact, soft tissue irregularities, such as local hard nodules in the breast, may indicate the presence of tumors and cancer, and presently the accepted means of detecting these breast lesions is manual palpation [40, 41]. And it is not uncommon for palpation to discover tumors that were undetected by computed tomography (CT), MRI, or ultrasound. However, due to its inherently subjective nature, palpation is hardly an infallible

detection tool; depending on their size and location, lesions do frequently go unnoticed to physicians, particularly those far below the skin surface. As in the above example, diseases commonly lead to a stiffening of local tissue over time due to phenomena like fluid discharges from the vascular system, loss of lymphatic systems, or simply the nature of the disease's pathology. Not surprisingly, in the word atherosclerosis, "sclerosis" means "hardness" and is an identifier for the plaque cap, which is locally hard due to its high concentration of collagen and possible calcification [42].

Elastography was developed to provide a more quantitative and reliable means of assessing tissue elasticity and as a solution to the imaging limitations. Ophir and colleagues [43-45] first conceived of it as a means of estimating strain maps (elastograms) from images of a tissue, or more generally a linear elastic, isotropic material, undergoing static deformation. A simplistic view of the process is illustrated in Figure 2-1. First, the tissue is *externally* compressed and surface imaging, particularly ultrasound, is used to capture the displacing specimen. Next, image processing and cross-correlation techniques are utilized to determine the displacement between pairs of A-lines pre- and post-compression. Finally, the resulting displacement field can be translated into a strain profile via the basic displacement and strain equilibrium equations for static compression.

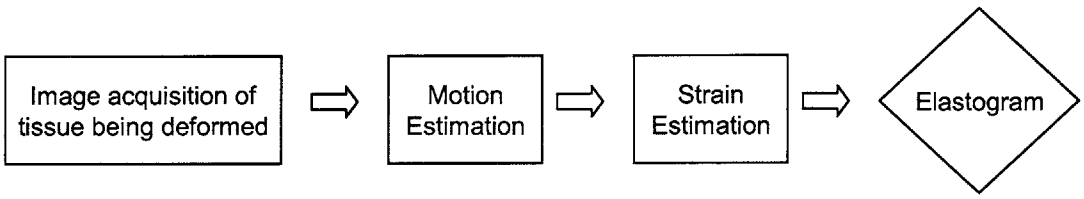


Figure 2-1. Flowchart of elastography process. In order to generate an elastogram—a map of a specimen's strain data—cross-correlation and image-processing techniques are used on images of a specimen pre- and post-compression.

Based on the original, above-described concept, elastography has been expanded to include different mechanical stimuli, imaging modalities, and specimens. De Korte *et al.* [29], for example, harnessed elastography for the moderately successful *in vivo* study of intravascular elasticity. Systemic blood pressure afforded the mechanical excitation and IVUS was used to monitor the subsequent arterial deformations, with the ultimate goal of identifying different plaque components and high strain regions via the elastogram. Other adaptations of elastography include dynamic loading, as opposed to static, by use of vibrations. Despite the added complexity of applying the wave equation as opposed to static equilibrium equations, dynamic loading does not require knowledge of boundary conditions outside the region under investigation. The review article by Greenleaf *et al.* [41] offers a more comprehensive examination of other mechanical excitation approaches, imaging methods, and the governing displacement-strain static and dynamic equations.

2.3 Model Parameter Estimation of Elasticity

Elasticity estimation is best summarized as the effort to replace *qualitative* measures of elasticity with *quantitative* diagnostic tools. Little has been mentioned thus far as to how arterial imaging along with elastography allows for the prediction of material properties, such as the elastic modulus or shear modulus. This section will focus on this endeavor—it provides motivation for applying model parameter estimation, a review of the governing linear elasticity equations, and a brief introduction into its implementation which is the focus of the remainder of the thesis.

2.3.1 Motivation

Parameter estimation is a field of optimization that surfaces in many scientific and engineering problems. Within this field, mixed numerical-experimental methods have become increasingly prevalent [46]. In the mechanical characterization of specimen material properties, for example, the classical methods of uniaxial, biaxial, and torsion tests are able to determine material properties by analytically relating local stress and strain to the applied boundary conditions via constitutive relationships. These methods, however, do not often suffice as they limit the choice of test specimens and applied boundary conditions and are not feasible solutions for commonly encountered complex models. To address these limitations, an alternative, less simplistic approach to obtaining mechanical properties is required—hybrid numerical-experimental parameter estimation.

Deducing the material properties of experimental specimens under realistic boundary conditions is difficult not merely because it does not allow use of the classical methods but because it involves inhomogeneous field data. Thus, knowledge of the boundary loads and conditions is not enough; measurements of field information (displacement, stresses, strains) are necessary. Biomaterials often fall into this category. Researchers have modeled a wide variety of biological tissue, including articular cartilage and the meniscus [47], skin and subcutis [48], myocardial tissue [49], and the intervertebral disk [50], in attempts to describe their mechanical behavior for medical and therapeutic purposes. Nonetheless, knowledge of soft tissue's material properties remains limited [44], partly a result of its specificity to the particular specimen or testing setup [3] and its enormous variability—the elastic moduli of normal soft tissue span four orders of

magnitude [22, 23]. Yet, a better understanding of this property variability in the form of a quantifiable field map of material properties would have invaluable consequences.

As explained in the "Thesis Goals," plaques associated with regions of high stress are vulnerable and prone to rupture. Furthermore, the geometry and makeup of the plaque (particularly the relative locations and sizes of the fibrous cap and lipid pools) along with respective material properties dictate the occurrence of stress concentrations. While elastography is capable of offering an internal distribution of arterial strain or displacement, there is no analogous means of measuring internal stress [40, 51]. This is precisely where model parameter estimation fits: it is a means of deducing the material properties (e.g. elastic modulus) based on imaging and elastography data in order to fully describe the arterial plaque model.

2.3.2 Linear Elasticity Equations

Familiarity with the underlying architecture of the arterial model is key to understanding the upcoming parameter estimation schemes. A forward problem model is one whereby the inputs, boundary conditions, geometry, and assumptions fully describe the model and lead to a direct solution of the outputs. The following is a review of the equations governing the forward problem model.

Consider a body being displaced and that each point on the boundary of the solid is specified either by a stress or displacement. Let $u(x, y, z) = u(\boldsymbol{x})$ denote its displacement field as a function of the spatial coordinate $\boldsymbol{x}$. The resulting strain components are defined by

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right). \tag{2.1}$$

Assuming incompressibility, a constraint can be added to the strain components, expressed as

$$\sum_{k=1}^{3} \varepsilon_{kk} = \frac{\partial u_k}{\partial x_k} = 0 \ . \tag{2.2}$$

An alternative way to define incompressibility constraint is to think about it as the limit of a compressible material:

$$\nabla \cdot u = \lim_{\lambda \to \infty} \left(-\frac{p}{\lambda} \right) = 0 \ , \tag{2.3}$$

where p is the pressure or hydrostatic stress and λ is the second Lamé constant, a material property that approaches ∞ as a material becomes incompressible.

The ensuing constitutive, stress-strain relation for an incompressible, linear elastic solid is

$$\sigma_{ij} = -p\delta_{ij} + 2\mu\varepsilon_{ij} \ , \tag{2.4}$$

where σ_{ij} is a component of the stress tensor, μ is the material shear modulus, and δ_{ij} is the Kronecker delta.

It should be noted that the more common material properties, E and v, the Young's modulus and Poisson's ratio, are related to λ and μ. For example, in the case of an incompressible material, $E = 3\mu$.

Balancing linear momentum over each part of the material gives the equilibrium equations,

$$\frac{\partial \sigma_{ij}}{\partial x_j} + f_j = 0 \ , \tag{2.5}$$

where f_j is the body force per unit volume. In this case, we ignore body force contributions, such as gravity, and proceed. Substituting Equation (2.4) into Equation (2.5), we arrive at the equilibrium equations in terms of strains,

$$-\frac{\partial p}{\partial x_i}+\frac{\partial\left(\mu\varepsilon_{ij}\right)}{\partial x_j}=0\ , \tag{2.6}$$

or displacements,

$$-\frac{\partial p}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\mu\frac{\partial u_j}{\partial x_i}\right)+\frac{\partial}{\partial x_j}\left(\mu\frac{\partial u_i}{\partial x_j}\right)=0\ . \tag{2.7}$$

Finally, a two-dimensional approximation is imposed as a simplification because imaging data is typically obtained in two-dimensional cross-sections of an artery. The two traditional approximations are the plane stress and plane strain assumptions. According to the plane stress approximation, out-of-plane stresses are zero, in other words,

$$\sigma_{xz}=\sigma_{yz}=\sigma_{zz}=0\ . \tag{2.8}$$

This is most valid for modeling bodies that are thin in the z-direction because they are more likely to maintain zero stresses throughout their thickness. According to the plane strain approximation, on the other hand, all out-of-plane strains are zero, or

$$u_z=0;\ \varepsilon_{xz}=\varepsilon_{yz}=\varepsilon_{zz}=0. \tag{2.9}$$

This is most valid for modeling bodies that are infinite in the z-direction, or more realistically for modeling bodies and loads that are close to uniform throughout the z-direction. Although neither approximation realistically describes an arterial image, researchers generally agree the plane strain approximation is more valid [52].

In Chapter 3 of this thesis, the equations presented above will be used to formulate the "inversion equation," the starting point for elastic modulus estimation.

2.3.3 Introduction to Model Parameter Estimation

As stated earlier, arterial imaging and elastography are two key components necessary for material property estimation. A naïve approach to material property estimation needs only these two components; elastography researchers often consider elastograms substitutes for elasticity maps. In other words, from the strain or displacement field generated via elastography, a map of the material properties is inferred [43, 53]. As expected, this technique can only provide approximate measures for tissue properties. Essentially, it correlates regions of high strain (or large displacement) with regions of low stiffness. To overcome the fact that internal stress is unknown, a uniform stress distribution is often imposed. However, this proves to be unrealistic and imprecise because the stress field actually decays from the boundaries where the load is applied into the tissue and generates a 'target hardening artifact' in the image [43]. Several investigations have attempted to compensate for this by imposing an analytical model of stress decay. This approach, however, is frequently complicated by stress concentrations at tissue boundaries [43, 54, 55].

To obtain a more suitable and accurate representation of material properties, a solution to the *inverse problem* is required. Here, the goal is to invert the measured mechanical responses within the framework of a theoretical model of the *forward elasticity problem.* A simplified flowchart outlining the general steps of an *iterative* inverse problem solution is shown in Figure 2-2.

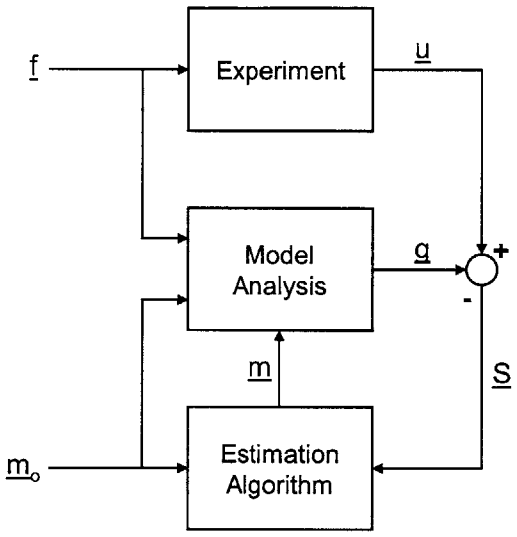


Figure 2-2. Mixed numerical-experimental, iterative solution for inverse problems. Experimental output ***u*** is compared with model output ***g*** to generate a sensitivity matrix ***S*** that aids the estimation algorithm in iterating on model inputs ***m*** until ***u*** and ***g*** match accordingly.

The objective is to minimize the difference between model-predicted (***g***) and experimentally-measured (***u***) displacement responses. A measured internal stress field is not needed since the elasticity is determined simply in terms of this residual that is captured by a sensitivity matrix (***S***).

Chapter 3

Gauss-Newton Method

3.1 Inverse Elasticity Problem

The inverse problem is not restricted to tissue elasticity reconstruction. It is a frequent obstacle in the field of model parameter estimation. In fact, inverse problem strategies have been investigated extensively by researchers in electromagnetics, optics, geophysics, continuum mechanics, and biomechanics [56-59]. As defined above, inverse problems are a subcategory of problems where the measured data are not sufficient to directly solve a forward model or the physical equations for desired unknown parameters. Of interest to bioelectricity research, for example, is the distribution of potentials at the surface of the heart or brain. Given the only available data are a limited number of peripheral potential measurements, an inverse problem approach is the common route.

The elasticity inversion equation is developed from the equilibrium elasticity equations formulated in the previous chapter. Rewriting Equation (2.6) by means of the incompressibility and plane strain assumptions gives two first-order partial differential equations for unknowns, p and μ:

$$-\frac{\partial p}{\partial x}+2\frac{\partial(\varepsilon_{xx}\mu)}{\partial x}+2\frac{\partial(\varepsilon_{xy}\mu)}{\partial y}=0. \tag{3.1}$$

$$-\frac{\partial p}{\partial y}+2\frac{\partial(\varepsilon_{xy}\mu)}{\partial x}-2\frac{\partial(\varepsilon_{xx}\mu)}{\partial y}=0. \tag{3.2}$$

Typically, the pressure terms are eliminated from Equations (3.1) and (3.2) by differentiating the first with respect to y and the second with respect to x and then

distinguished from the general category of Newton's method is explained shortly). Newton's method locates the roots of nonlinear equations and functions by applying a local linear model to the function at the current iterate. In other words, the algorithm fits a tangent line to the function at a given point and by approximating the function's slope with the slope of the line it is able to deduce a root direction.

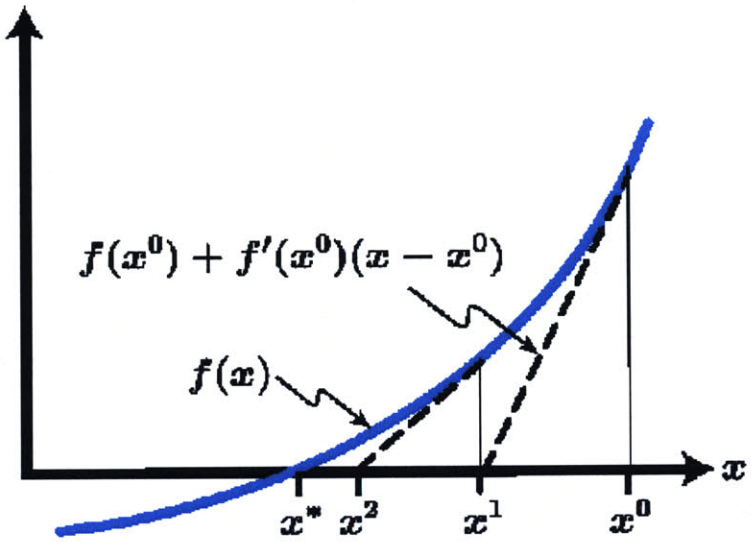


Figure 3-1. 1-D picture of a Newton method solution. By approximating local derivatives at certain points, the Newton method iteratively seeks a nonlinear function's zero.

For example, suppose we have a nonlinear function expressed as

$$F(x) = 0 . \tag{3.8}$$

Then the traditional Newton sequence, or update equation, is given by

$$x_{k+1} = x_k - F'(x_k)^{-1} F(x_k) , \tag{3.9}$$

where k denotes the iteration number (i.e., the k^{th} iterate).

We now develop the Newton sequence for the NLS equations of interest. Equation (3.6) is the nonlinear function, analogous to Equation (3.8), whose roots are the solutions

this simplifying step [62], this is not advantageous experimentally because, at present, hydrostatic pressure cannot be measured from within soft tissues.

Iterative Inversion

An alternative, preferred strategy to direct inversion for solving the inverse elasticity equations was introduced in Figure 2-2. By minimizing the residual between model-computed and measured mechanical responses, unknown model parameters, such as a vector of Young's modulus, can be iteratively fit to the model. This iterative approach, whereby the solution is not expressed in closed form but rather a residual is used to back-solve for certain parameters, has also been harnessed in other endeavors, such as electrical impedance tomography [63] and microwave imaging [64].

As shown in Figure 2-2, this approach requires a simple feedback loop:

(i) Given specimen geometry, boundary conditions, and forcing load ($\boldsymbol{f}$).

(ii) Given experimental mechanical responses ($\boldsymbol{u}$).

(iii) Guess parameter distribution (e.g. $\boldsymbol{\mu}$).

(iv) Build model and compute mechanical responses ($\boldsymbol{g}$) by solving the forward elasticity problem.

(v) Check if residual ($\boldsymbol{g} - \boldsymbol{u}$) is small enough. If criterion is not satisfied, update $\boldsymbol{\mu}$ and return to (iv). If criterion is satisfied, stop iterating.

As illustrated in the above stepwise summary, the iterative scheme requires more than one, and typically many, solutions to the forward problem, making it a far slower and less efficient choice than direct inversion. So why opt for iterative inverse reconstruction? First, it is more robust and therefore can be easily combined with commercially-available finite element software. As a result, the effort of developing it is reduced as computation

is distributed over various platforms. Second, no continuity restrictions are placed on the shear or Young's modulus, allowing jump changes and bound inclusions. Third, and possibly most importantly, iterative solutions are traditionally more stable and less sensitive to errors in the experimental data than direct inversion solutions, which often lead to unstable systems [52].

Iterative solutions for the inverse problem are the focus of this chapter's remainder. In step (v) above, it is not obvious exactly how to update $\boldsymbol{\mu}$ in order to arrive at a good fit, however, clearly this action is the driving force behind the search and its success. Previously, we hinted at the use of a sensitivity matrix to direct the search. The formulations of the necessary equations as well as of this sensitivity matrix are presented below, followed by our version of a solution and some simulation results.

3.2 Formulating the Nonlinear Least Squares Equations

When the solution to the inverse elasticity problem is expressed as one that minimizes the residual between computed and observed mechanical responses, the realm of nonlinear least squares is entered. The general form for a nonlinear least squares (NLS) problem is

Given $F: \quad R^p \to R^q, \quad q \geq p, \quad$ solve

$$\min_{\bar{x} \in R^p} \left\{ \Phi(\bar{x}) = \tfrac{1}{2} \left\| \bar{F}(\bar{x}) \right\|^2 \right\}, \tag{3.1}$$

where $\boldsymbol{F} = (f_1, f_2, \ldots, f_n)^T$ is an objective function that guides the search toward the stationary point, $\boldsymbol{x}^*$, one that satisfies a zero gradient, minima criterion. To attack the NLS problem and as a prelude to introducing the Gauss-Newton method, an understanding of and expressions for both the gradient and Hessian of $\Phi(\boldsymbol{x})$ are needed. They are written as

$$\overline{\nabla}\Phi\left(\overline{x}\right)=\overline{J}\left(\overline{x}\right)^{T}\,\overline{F}\left(\overline{x}\right) \tag{3.2}$$

and

$$\overline{\nabla}^{2}\Phi\left(\overline{x}\right)=\overline{J}\left(\overline{x}\right)^{T}\,\overline{J}\left(\overline{x}\right)+\sum_{i=1}^{n} f_{i}\left(\overline{x}\right)\overline{\nabla}^{2} f_{i}\left(\overline{x}\right), \tag{3.3}$$

where the Jacobian, a matrix of all first-order partial derivatives of a vector-valued function (i.e., its columns are composed of the function's gradients), is defined by

$$J\left(\overline{x}\right)_{ij}=\frac{\partial f_{i}}{\partial x_{j}}\left(\overline{x}\right). \tag{3.4}$$

How to calculate the Jacobian in practice is addressed in the proceeding section.

From the definition of the gradient (Equation (3.2)), we have insight into the geometric interpretation of the stationary point. Since finding the stationary point corresponds to finding where the gradient is zero, $\overline{\nabla}\Phi\left(\overline{x}^{*}\right)=\overline{J}\left(\overline{x}^{*}\right)^{T}\,\overline{F}\left(\overline{x}^{*}\right)=0$, it also corresponds to the point for which $\boldsymbol{F}(\boldsymbol{x}^{*})$ is orthogonal to the columns of $\boldsymbol{J}(\boldsymbol{x}^{*})$. This interpretation confirms the importance of tracking changes in the objective function, $\boldsymbol{F}(\boldsymbol{x})$, as a means of assessing convergence [65]. Not surprisingly, the value of the objective function is traditionally used as an important convergence criterion. The other convergence criteria are listed in the next section when we build a Gauss-Newton method around the NLS problem.

Before presenting the Gauss-Newton method, it is helpful to personalize the general NLS problem to the case of elasticity reconstruction, the main decision being what the objective function should be. As already mentioned, a suitable driving force is the residual between model-predicted and experimentally-measured displacement fields, and

this is the common choice of other researchers interested in iterative inverse elasticity reconstruction [40, 51, 66]. Equation (3.1) can then be rewritten as

Given $g: \; R^p \to R^q, \quad q \geq p, \quad$ solve

$$\min_{\bar{E} \in R^p} \left\{ \Phi(\bar{E}) = \tfrac{1}{2} \left\| \bar{g}(\bar{E}) - \bar{u} \right\|^2 \right\}, \tag{3.5}$$

where $\boldsymbol{E}$ is the Young's modulus distribution, typically a one-dimensional vector ($p = 1$) if concerned with an *isotropic* distribution, $\boldsymbol{g}(\boldsymbol{E})$ is the vector of computed displacements based on the inputted $\boldsymbol{E}$, and $\boldsymbol{u}$ is the vector of measured displacements (both displacement fields lie in one-, two-, or three-dimensional space depending on how many displacement components are provided by the experiment). Once again, the NLS problem is finding the stationary point $\boldsymbol{E}^*$ satisfying a zero gradient, minima criteria. Therefore, expressions for the gradient and Hessian of $\Phi(\boldsymbol{E})$ are necessary:

$$\bar{\nabla}\Phi(\bar{E}) = \bar{J}\left(\bar{g}(\bar{E})\right)^T \left(\bar{g}(\bar{E}) - \bar{u}\right) = \bar{J}_g^T \left(\bar{g}(\bar{E}) - \bar{u}\right), \tag{3.6}$$

where $\boldsymbol{J_g}$ is the Jacobian of $\boldsymbol{g}(\boldsymbol{E})$ with respect to $\boldsymbol{E}$, and

$$\bar{\nabla}^2\Phi(\bar{E}) = \bar{J}_g^T \bar{J}_g + \frac{\partial J_{g\,i}}{\partial E_j}(\bar{E})\left[\bar{I} \otimes \left(\bar{g}(\bar{E}) - \bar{u}\right)\right], \tag{3.7}$$

where $\boldsymbol{I}$ is the identity matrix and $\otimes$ is the Kronecker delta. Note that the final term appearing in Equation (3.7) is identical to that of Equation (3.3), where the latter is merely expressed as a summation.

3.3 Linear Perturbation Gauss-Newton Method

The NLS equations for elasticity are now prepared for a modified Newton's method, typically a Gauss-Newton algorithm (the reason the Gauss-Newton method is

distinguished from the general category of Newton's method is explained shortly). Newton's method locates the roots of nonlinear equations and functions by applying a local linear model to the function at the current iterate. In other words, the algorithm fits a tangent line to the function at a given point and by approximating the function's slope with the slope of the line it is able to deduce a root direction.

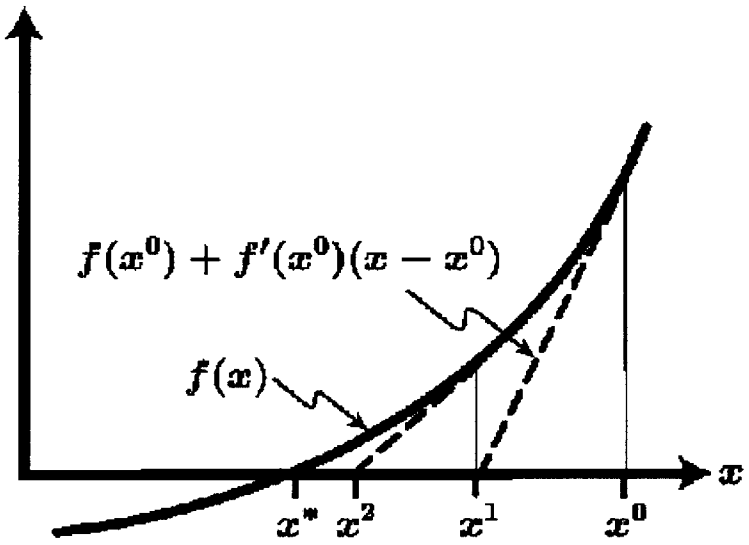


Figure 3-1. 1-D picture of a Newton method solution. By approximating local derivatives at certain points, the Newton method iteratively seeks a nonlinear function's zero.

For example, suppose we have a nonlinear function expressed as

$$F(x) = 0 . \tag{3.8}$$

Then the traditional Newton sequence, or update equation, is given by

$$x_{k+1} = x_k - F'(x_k)^{-1} F(x_k) , \tag{3.9}$$

where k denotes the iteration number (i.e., the k^{th} iterate).

We now develop the Newton sequence for the NLS equations of interest. Equation (3.6) is the nonlinear function, analogous to Equation (3.8), whose roots are the solutions

to the NLS problem. Thus, a Taylor series expansion of it about an arbitrary elasticity distribution $\boldsymbol{E_k}$ reveals

$$\overline{\nabla}\Phi\left(\overline{E}_k\right) \approx \overline{\nabla}\Phi\left(\overline{E}_k\right) + \overline{\nabla}^2\Phi\left(\overline{E}_k\right)\cdot\left(\Delta\overline{E}_k\right), \tag{3.10}$$

where $\Delta\boldsymbol{E_k} = \boldsymbol{E} - \boldsymbol{E_k}$ [51]. In order to minimize Equation (3.5), we demand that Equation (3.10) equals zero and substitute in the expressions from Equations (3.6) and (3.7). Prior to substitution, however, a simplification is made to the Hessian (Equation (3.7)). The second order term is neglected based on the observation that it is typically small relative the first term [63]:

$$\overline{\nabla}^2\Phi\left(\overline{E}\right) \approx \overline{J}_g^T\,\overline{J}_g\,. \tag{3.11}$$

This simplification of the Hessian and its corresponding advantages and disadvantages are a distinguishing characteristic of the Gauss-Newton method from other Newton's methods. The limitations introduced as a result of it are elaborated upon in the following section; however, suffice it to say the reduction yields a *symmetric* and *positive-definite* Hessian, so long as $\boldsymbol{J_g}$ is not singular, and thus a more solvable system. The final Newton update equation for the root direction becomes

$$\Delta\overline{E} = -\left[\overline{\nabla}^2\Phi\left(\overline{E}\right)\right]^{-1}\overline{\nabla}\Phi\left(\overline{E}\right) \tag{3.12}$$

or

$$\overline{E}_{k+1} = \overline{E}_k - \left[\overline{J}_g^T\overline{J}_g\right]^{-1}\left[\overline{J}_g^T\left(\overline{\mathrm{g}}\left(\overline{E}\right) - \overline{u}\right)\right]_k\,. \tag{3.13}$$

An important remaining task is to build $\boldsymbol{J_g}$ (the Jacobian of $\boldsymbol{g}(\boldsymbol{E})$ with respect to $\boldsymbol{E}$). Unfortunately, this is not straightforward because the function $\boldsymbol{g}(\boldsymbol{E})$ is not explicitly known. Instead, FEM is used to transform the inputted model geometry, boundary conditions, forcing load, and material properties into the desired displacement responses.

Thus, unlike the case of nonlinear functions, which possess explicit partial derivatives that satisfy Equation (3.4), an approximation for the Jacobian must be employed. As proposed by Kallel *et al.* [51], a common solution is a finite difference, or linear perturbation, approximation of Equation (3.4). For example, suppose once again we have a function $\boldsymbol{F}(\boldsymbol{x}) = (\boldsymbol{f_1}, \ldots, \boldsymbol{f_n})^T$. Then,

$$J\left(\bar{x}\right)_{ij} = \frac{\partial f_i}{\partial x_j}\left(\bar{x}\right) \approx \frac{f_i\left(x_j + \varepsilon \left|x_j\right| e_j\right) - f_i\left(x_j\right)}{\varepsilon \left|x_j\right|}, \tag{3.14}$$

for "sufficiently" small perturbation ε, where e_j is the unit basis vector in the j^{th} direction. A discussion on the choice of ε is deferred to the following section for it is has important consequences in neutralizing potential errors and limitations in the Gauss-Newton method.

We end this section with a brief note on convergence criteria. Although we are most concerned with the decay of the error between current iterate and solution ($\boldsymbol{E}^*$), it is unattainable because $\boldsymbol{E}^*$ is the target unknown. A common substitute, however, is to monitor the norm of the objective function [67]. In other words, set an error tolerance that must be met by the norm of the objective function:

$$\left\| \bar{F}\left(\bar{x}\right)\right\| \leq \tau_r \left\| \bar{F}\left(\bar{x}_0\right)\right\|, \tag{3.15}$$

where τ_r is some algorithm-specific fraction. In addition to checking the error, it is useful and sometimes crucial to establish a convergence criteria based on the algorithm's step. This ensures that not only is the residual decaying but the search's rate is not stagnating. A simple way of accomplishing this is to set an error tolerance on the norm of the difference between subsequent iterates:

$$\left\| \bar{E}_{k+1} - \bar{E}_k \right\| \leq \tau_s, \tag{3.16}$$

where τ_s is an algorithm-specific value.

3.4 Limitations

Shortcomings associated with gradient-based strategies to solving such NLS problems will be concisely addressed in this section. In addition, we will attempt to identify the consequences of these shortcomings. Most are inherent to the solution strategy, however, some, such as machine precision errors, are a result of precision limitations born of the FE function evaluation or of other software choices.

3.4.1 Computational Cost

The linear perturbation Gauss-Newton method requires computation of a finite difference Jacobian, which can be computationally substantial. Since FEM function evaluations are the only available means of transforming an elasticity distribution into a field of displacements, building the Jacobian requires m (or, more precisely, m+1) function evaluations, where m is the number of points or finite elements that have elastic moduli assigned to them. In other words, the number of m parameter unknowns dictates the number of function evaluations needed to build the Jacobian, and thus the computational cost of doing this.

If one can safely assume the Jacobian is sparse, then the computational cost can be reduced. In this case, it is possible to calculate several columns of the Jacobian via a single function evaluation by taking advantage of the banded Jacobian and *a priori* knowledge of its bandwidth. However, this is a special case scenario, and in the results presented later in this chapter, typically the Jacobian is not sparse.

3.4.2 Machine Precision Errors

For a convergent Gauss-Newton system, it is apt to divide limitations into (i) those that lead to *stagnation* and a resulting small error in solution *accuracy* and (ii) those that affect the *speed* of the nonlinear iteration [68]. The former can be products of certain machine precision errors. Here, the derivatives, assembled in the Jacobian, are accurate and successfully drive the search toward the solution in typical Newton *q-quadratic* fashion. However, once near the solution, stagnation occurs, and it becomes clear the algorithm's iterates cease to gain accuracy. The machine precision errors are, more often than not, introduced via the function evaluation, especially when various software platforms are being integrated. Not surprisingly, the solution can only be as accurate as the results of the function evaluation.

Furthermore, in order to develop a functioning and successful linear perturbation Gauss-Newton method, these precision errors associated with the function evaluation (i.e., FEM) need to be known. Otherwise, perturbation of the elasticity distribution, summarized by Equation (3.14) cannot be accomplished effectively. For example, if the elasticity values are perturbed by an amount less than and thus invisible to the FEM's precision, then they are essentially unperturbed and incapable of predicting a gradient. Choice of ε is thus reliant on the error in the function evaluation, and a general rule for its determination is [68]

$$\varepsilon \geq \sqrt{error_{function\ evaluation}} \ . \tag{3.17}$$

3.4.3 The Ill-Posed Problem

The stagnation issue and inaccuracy associated with function evaluation precision errors are generally benign, especially when compared with the second genre of limitations: those that affect the *speed* of the nonlinear iteration. At the root of this is an ill-conditioned system.

For a general linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$, an ill-conditioned matrix $\boldsymbol{A}$, which produces an ill-posed problem, implies that some (or all) of the equations are numerically linearly dependant. Some upshots of this are a very sensitive and unstable system when subject to noise, rounding errors, and perturbations. It is a common misconception that sensitivity to noise is the only consequence; however, this is not the case. Hadamard, a pioneer in the work of well-posed and ill-posed problems, defined a problem as ill-posed if the solution is not unique *or* if it is not a continuous function of the data (i.e., large sensitivity to noise and small perturbations) [69]. For example, in the case of the inverse elasticity problem, Barbone and Bamber [52] showed analytically that, among other factors, model boundary conditions in the forward incompressible elasticity problem essentially dictate the ill-posedness of the inverse problem. Specifically, application of displacement boundary conditions in the forward problem leads to a *nonunique* inverse problem even devoid of noise, because it provides no information beyond what is already available in the measured displacement field. In order to make the unknown elasticity distribution unique, some knowledge of elasticity or stress must be applied to the forward problem to render its inverse well-posed.

Contrary to Hadamard's belief that ill-posed problems are "artificial" and unlikely to describe physical systems, they appear frequently, especially in the context of inverse problems. A very general formulation of the inverse problem is

$$\int_{\Omega} \text{input} \times \text{system}\, d\Omega = \text{output}, \tag{3.18}$$

where the goal is to deduce either the input or the system based on measurements of the output, which are frequently noisy. Qualitatively speaking, these objectives translate to determining unknown inputs that yield certain output signals or determining the internal structure of a physical system from its measured behavior [70]. The prototypical ill-posed inverse problem is a Fredholm integral equation of the first kind, written as [71]

$$\int_0^1 \overline{K}(s,t)\overline{f}(t)dt = \overline{g}(s), \qquad 0 \le s \le 1, \tag{3.19}$$

where the kernel $\boldsymbol{K}$ and the function $\boldsymbol{g}$ are known and signify the mathematical model and the measured finite outputs, while $\boldsymbol{f}$ is the unknown input function. What makes this problem difficult to solve or ill-posed? It is the immutable link between the kernel $\boldsymbol{K}$ and its inputs *within* the integral that makes the system nontrivial to solve. Integration with $\boldsymbol{K}$ tends to "smooth" the inputs $\boldsymbol{f}$, especially high-frequency components, cusps, or edges. Thus, in the reverse or inverse process, high-frequency components in $\boldsymbol{g}$ will be amplified, making determination of $\boldsymbol{f}$ difficult.

Before addressing numerical techniques to remedy ill-conditioned systems, it is helpful to examine the kernel $\boldsymbol{K}$ and its "smoothing" properties more closely. Singular value expansion (SVE) is the accepted means of studying square integrable kernels, as in the Fredholm integral equation. However, the analogous tool for the analogous matrix equations, which is more appropriate for this thesis, is singular value decomposition (SVD).

SVD of a rectangular or square matrix, $\boldsymbol{A} \in \Re^{m \times n}$, where $m \ge n$, yields

$$\bar{A} = \bar{U}\,\bar{\Sigma}\,\bar{V}^T = \sum_{i=1}^{n} \bar{u}_i \bar{\sigma}_i \bar{v}_i^T \,, \tag{3.20}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices (i.e., their rows are an orthonormal basis and their columns are an orthonormal basis, such that $\boldsymbol{U^T U} = \boldsymbol{V^T V} = \boldsymbol{I}$) and where $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values of $\boldsymbol{A}$. SVD is unique for a given matrix $\boldsymbol{A}$, the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are its sets of orthonormal basis vectors that transform it into a diagonal matrix, and its singular values, which decay gradually to zero ($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$), reveal the rank and ill-conditioning of $\boldsymbol{A}$ (the rank of $\boldsymbol{A}$ is equal to the number of nonzero singular values appearing on the diagonal).

We now focus on how the singular values are a direct measure of the ill-conditioning of $\boldsymbol{A}$. Multiplying Equation (3.20) by $\boldsymbol{v_i}$ and the transpose of Equation (3.20) by $\boldsymbol{u_i}$, expressions similar to an eigenvalue decomposition arise:

$$\bar{A}\bar{v}_i = \sigma_i \bar{u}_i \,, \qquad \left\| \bar{A}\bar{v}_i \right\|_2 = \sigma_i \tag{3.21}$$

and

$$\bar{A}^T \bar{u}_i = \sigma_i \bar{v}_i \,, \qquad \left\| \bar{A}^T \bar{u}_i \right\|_2 = \sigma_i \,. \tag{3.22}$$

Equations (3.21) and (3.22) show that for a given small singular value σ_i, there exists vectors $\boldsymbol{v_i}$ and $\boldsymbol{u_i}$ that are the numerical null vectors of $\boldsymbol{A}$ and $\boldsymbol{A^T}$, respectively. In other words, if $\boldsymbol{A}$ possesses one or more small singular values, then some of its columns are, or are nearly linearly dependent and its rank is nearly deficient, hence an ill-conditioned matrix. The presence of small singular values, especially when compared with the matrix's largest singular value, is a clear indicator of such ill-conditioning.

Another observation associated with decreasing singular values is the increase in sign changes in the corresponding singular vectors $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$. In other words, $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$ tend to

have more sign changes as the σ_i decrease—a characteristic feature of SVD. This oscillatory behavior of the singular vectors helps explains the "smoothing effect" of the kernel $\boldsymbol{K}$. For example, consider SVD of the mapping $\boldsymbol{Ax}$ of an arbitrary vector $\boldsymbol{x}$ [70]:

$$\bar{x} = \sum_{i=1}^{n} \left(\bar{v}_i^T \bar{x} \right) \bar{v}_i \tag{3.23}$$

and

$$\overline{Ax} = \sum_{i=1}^{n} \sigma_i \left(\bar{v}_i^T \bar{x} \right) \bar{u}_i \,. \tag{3.24}$$

It is evident from Equations (3.23) and (3.24) that multiplication with the σ_i, which translates to $\boldsymbol{Ax}$ mapping of the vector $\boldsymbol{x}$, tends to muffle high frequency components of $\boldsymbol{x}$ more so than corresponding low-frequency components. This observation, along with the conclusion that the inverse process would tend to have the opposite, amplifying effect, is consistent with hypothesis presented above regarding properties of the kernel $\boldsymbol{K}$ in the context of inverse problems.

An ill-posed problem, which jeopardizes the uniqueness of the solution, results in a system that is very sensitive to slight perturbations and noise in the data. The result is an ineffective *and* inefficient Jacobian and subsequent Newton root direction. Hence, this potentially detrimental type of error is said to affect the *speed* of the nonlinear iteration. Harking back to the inverse elasticity problem at hand, it is important to note some additional, specific sources of error that make the problem statement even more vulnerable to ill-conditioning.

In a previous section, the Gauss-Newton sequence (or update) equation was derived. An analogy can be made between the result, shown in Equations (3.12) and (3.13), and the general form of a linear set of equations $\boldsymbol{Ax} = \boldsymbol{b}$. In this case, the unknown vector $\boldsymbol{x}$ is

equivalent to $\Delta \boldsymbol{E}_k$ (the update on the elasticity distribution direction), $\boldsymbol{A}$ translates to the Hessian ($-\boldsymbol{J_g}^T \boldsymbol{J_g}$), and $\boldsymbol{b}$ translates to $\boldsymbol{J_g}^T (\boldsymbol{g}(\boldsymbol{E}) - \boldsymbol{u})$. SVD of $\boldsymbol{A}$ revealed that an increase in the matrix's dimensions would yield an increase in the number of small singular values, and thus the potential for a more ill-conditioned system. This is an especially precarious relation for the Hessian at hand, because of the way it is represented:

$$\overline{\nabla}^2 \Phi\left(\overline{E}\right) \approx \overline{J}_{\mathrm{g}}^{\,T}\, \overline{J}_{g}\,. \tag{3.11}$$

Recall that a second order term is omitted in order to arrive at Equation (3.11) because of the difficulty in calculating it and because it is regarded as small relative to the first order term. Additionally, this simplification imposes a symmetric, positive-definite Hessian—a traditionally more versatile matrix to invert with real eigenvalues and hence more physical to understand. However, a negative consequence is immediately evident from SVD of the Jacobian:

$$\overline{J}_g\left(\overline{E}\right) = \overline{U}_g\left(\overline{E}\right)\, \overline{\Sigma}_g\left(\overline{E}\right)\, \overline{V}_g\left(\overline{E}\right)^T, \tag{3.25}$$

where $\boldsymbol{U_g}$ and $\boldsymbol{V_g}$ are the orthogonal matrices and $\boldsymbol{\Sigma_g}$ is a diagonal matrix containing the singular values of $\boldsymbol{J_g}$. It should be noted that when an ill-posed problem is discretized, as is this problem, the inherent difficulties with ill-conditioning carry over to the discrete coefficient matrix, thus compelling a closer investigation into matrices such as $\boldsymbol{J_g}$ (also, this suggests what has already been assumed—that a strong connection between SVE and SVD exists). According to SVD, as the dimensions of $\boldsymbol{J_g}$ increase, so do the number of small singular values, thereby increasing the likelihood of it being ill-conditioned (the degree of ill-conditioning can be estimated by the ratio of the largest singular value to the smallest). The size of $\boldsymbol{J_g}$ depends solely on the number of unknown elasticity modulus values discretized by the model; thus, for accurate and fine distributions, the dimensions

of $\boldsymbol{J_g}$ tend to be large. Moreover, it is not the ill-conditioning of $\boldsymbol{J_g}$ that is of primary concern, but rather the ill-conditioning of the Hessian, decomposed as

$$\overline{\nabla}^2\Phi(\overline{E}) \approx \overline{J}_g^T(\overline{E})\,\overline{J}_g(\overline{E}) = \left(\overline{V}_g(\overline{E})\,\overline{\Sigma}_g(\overline{E})^T\,\overline{U}_g(\overline{E})^T\right)\left(\overline{U}_g(\overline{E})\,\overline{\Sigma}_g(\overline{E})\,\overline{V}_g(\overline{E})^T\right)$$

$$= \overline{V}_g(\overline{E})\,\overline{\Sigma}_g(\overline{E})^T\,\overline{\Sigma}_g(\overline{E})\,\overline{V}_g(\overline{E})^T = \overline{V}_g(\overline{E})\,\overline{\Sigma}_g(\overline{E})^*\,\overline{V}_g(\overline{E})^T. \tag{3.26}$$

As illustrated by Equation (3.26), the ill-conditioning of the Hessian scales roughly with the *square* of the size and ill-conditioning of $\boldsymbol{J_g}$.

Neglect of the second order term in the Hessian's description poses yet another source of ill-conditioning. As Dennis [65] explains, if the residual size (between the solution and the iterate) is too large, then the Gauss-Newton method will not be locally convergent. This is a typical consequence of an ill-conditioned gradient and an ill-posed problem—an inability to produce typical quadratic convergence. The reason the second order term can be at fault is because it is directly proportional to the residual between computed and measured displacements (see Equation (3.7)). In other words, if the iterate is far from the solution, based on a bad initial guess of the solution, this second order term can dominate and produce an erroneous Hessian. Not surprisingly, successful use of a Gauss-Newton method, via simplifying the Hessian, is reliant on small residual sizes.

3.4.4 Local Convergence

The Gauss-Newton method, along with its counterpart calculus- and gradient-based methods, is local in scope. In other words, it seeks optima that are the *best in a neighborhood of the current point* [72]. Often referred to as "hill-climbing" techniques, they rely solely on an expression for a certain function's gradient to climb up to the peak

of its "hill." Once the gradient is zero, or nearly zero, the search ends, having determined the location where the derivatives (slopes) are zero in *all* directions.

Complex, nonlinear functions, and especially those that describe physical phenomena, rarely possess only one maxima or minima. And thus when faced with a distribution of many peaks and valleys, these calculus-based searches often depend highly on the location of the search's starting point.

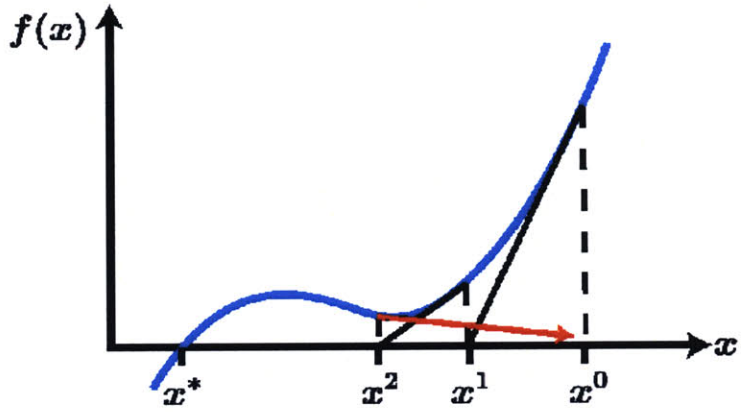


Figure 3-2. 1-D picture of a Newton method failing. In the presence of a more complex function characterized by valleys and local minima, a Newton method is likely to yield erroneous search directions.

Figure 3-2 does not portray a search erroneously falling into a local minima, however, it does provide a 1-D depiction of similar limitations associated with being completely dependent on local derivatives. Due to the starting point, the search was propelled away from the actual, global solution by a local derivative that was close to zero, once again reinforcing the importance of an accurate initial guess to the algorithm's efficiency and success.

Global convergence strategies, which can be vital to ensuring certain problems converge on the correct solution, are discussed at length in the following chapter.

3.5 Regularization

Of the limitations and drawbacks of the Gauss-Newton method, it is the ill-posed problem that can be most pervasive and detrimental, yet also one that can be alleviated. Regularization is the general technique of exploiting *a priori* information in order to (i) stabilize the system and (ii) identify a unique, stable solution, when faced with underdetermined problems [70]. It is accomplished by adding a constraint that must be satisfied in addition to minimizing the original residual norm. Typically, this *a priori* constraint involves the regularized solution or a residual of the regularized solution.

One of the most widely used schemes is Tikhonov regularization, where the objective is to minimize a linear combination of the residual norm and an additional regularization term. Of course the minimization of the residual norm is sacrificed slightly, however, by also minimizing a well-chosen regularization term, a result that is *close* to the solution is hopefully attained. In the discrete nonlinear system of equations introduced in Equation (3.1), a generalized Tikhonov regularization scheme becomes

Given $F: \; R^p \rightarrow R^q, \quad q \geq p, \quad$ solve

$$\min_{\bar{x} \in R^p} \left\{ \frac{1}{2} \left\| \bar{F}(\bar{x}) \right\|_2^2 + \lambda \bar{\Omega}^2(\bar{x}) \right\}, \tag{3.27}$$

where λ is the weighting factor and $\boldsymbol{\Omega}(\boldsymbol{x})$ is the regularization term and a function of the regularized solution, $\boldsymbol{x}$. This latter term is sometimes dubbed the *discrete smoothing norm* because it often takes the form

$$\bar{\Omega}(\bar{x}) = \left\| \bar{L}\bar{x} \right\|_2, \tag{3.28}$$

where the matrix $\boldsymbol{L}$ acts as a smoothing operator on the solution field [70]. Choice of $\boldsymbol{L}$ depends on the desired smoothing effect and includes, but is not exclusive to, identity

matrices, weighted diagonal matrices, and discrete approximations for derivative and Laplacian operators. For example, a common representation for $\boldsymbol{L}$ is a discrete first-order derivative, expressed as

$$\overline{L}_1 = \begin{pmatrix} 1 & -1 & 0 & \\ & \ddots & \ddots & \\ & 0 & 1 & -1 \end{pmatrix}, \tag{3.29}$$

which is used to penalize large gradients between solution values of subsequent discrete elements.

Tikhonov regularization was first developed independently by Phillips [73] and Tikhonov [74] and has become a fundamental formula for ill-conditioned systems, so fundamental that a rigorous discussion of its theory is omitted in this thesis because it is widely available in the literature. However, within the context of the elasticity inverse problem, we illustrate how and why it is effective.

The first and most obvious remedying effect of Tikhonov regularization is a direct consequence of the smoothing norm. Often, solution fields of unregularized ill-posed problems are erratic and discontinuous due to high sensitivity to noise and slight perturbations. However, the elasticity of biological tissue should, for the most part, be continuous and harmonious. The Tikhonov regularization term offers a very direct penalty against solution distributions that does not meet certain user-desired smoothness and inclusion criteria.

To appreciate the more indirect consequences of Tikhonov regularization, it is helpful to carry out the mathematical derivation of including it in the original NLS equations:

Given $g: \; R^p \rightarrow R^q, \quad q \geq p,$ solve

$$\min_{\bar{E}\in R^p}\left\{\Phi(\bar{E})=\tfrac{1}{2}\left\|\bar{g}(\bar{E})-\bar{u}\right\|_2^2+\frac{\lambda}{2}\left\|\overline{LE}\right\|_2^2\right\}. \tag{3.30}$$

Represented in matrix notation, we obtain the relation

$$\Phi(\bar{E})=\frac{1}{2}\left(\bar{g}(\bar{E})-\bar{u}\right)^T\left(\bar{g}(\bar{E})-\bar{u}\right)+\frac{\lambda}{2}\left(\overline{LE}\right)^T\left(\overline{LE}\right), \tag{3.31}$$

which can now be used to determine the gradient with respect to the elasticity distribution, $\boldsymbol{E}$:

$$\bar{\nabla}\Phi(\bar{E})=\bar{J}_g^T\left(\bar{g}(\bar{E})-\bar{u}\right)+\lambda\,\bar{L}^T\overline{LE}. \tag{3.32}$$

Via a process analogous to obtaining Equation (3.13), a Newton update equation based on the above gradient and corresponding Hessian is established:

$$\bar{E}_{k+1}=\bar{E}_k-\left[\bar{J}_g^T\bar{J}_g+\lambda^2\bar{L}^T\bar{L}\right]^{-1}\left[\bar{J}_g^T\left(\bar{g}(\bar{E})-\bar{u}\right)+\lambda\bar{L}^T\overline{LE}\right]_k. \tag{3.33}$$

Of primary note is that the matrix to be inverted is altered. Specifically, the original $\boldsymbol{J_g^T J_g}$ is augmented by a matrix composed of the regularization term, $\boldsymbol{L}$. The benefit lies in the observation that the augmentation matrix appears as $\boldsymbol{L^T L}$, a symmetric, positive-definite matrix. As a result, the diagonal of the original Hessian ($\boldsymbol{J_g^T J_g}$) sees a vast enlargement. Recall that an ill-conditioned Hessian, one that is riddled with clusters of small singular values making its columns numerically linearly dependent, is the root of an ill-posed problem. Hence, we can infer that boosting the diagonal of the Hessian, via addition of a symmetric, positive-definite matrix, also boosts the small singular values on the diagonal of $\boldsymbol{\Sigma}$.

Mathematically, it can also be shown that Tikhonov regularization yields unique solutions by rewriting Equation (3.33) in the form

$$\left(\bar{J}_g^T \bar{J}_g + \lambda^2 \bar{L}^T \bar{L}\right)\Delta\bar{E}_k = \bar{J}_g^T \left(\bar{g}\left(\bar{E}\right) - \bar{u}\right) + \lambda \bar{L}^T \overline{LE}\,. \tag{3.34}$$

It follows that if $N\left(\bar{J}_g\right) \cap N\left(\bar{L}\right) = \bar{0}$ (i.e., if the nullspaces of $\boldsymbol{A}$ and $\boldsymbol{L}$ intersect trivially), then the Tikhonov solution is unique [70].

We now present a brief explanation of how the regularization weighting parameter, λ, is obtained. Correct choice of λ is a nontrivial problem with many proposed solutions, of which the L-curve method (as described below) has become the most popular. According to this selection criterion, λ is the value that maximizes the curvature of the typically L-shaped log-log plot of the regularized solution squared norm, $\left\| \bar{x}_{reg} \right\|^2$, versus the residual vector squared norm, $\left\| \overline{Ax}_{reg} - \bar{b} \right\|^2$. In other words, as expected, it serves as the appropriate balance between the two terms being minimized in Equations (3.27) or (3.30). Furthermore, it has been observed that the value corresponding to the L-curve corner is comparable to the magnitude of the mean squared residual of the unregularized solution [75]. Therefore, to find λ for a particular problem, in many instances, we first attacked the problem with an unregularized Gauss-Newton method, solved for

$$\lambda \approx \frac{1}{m} \sum_{j=1}^{m} \left(g_j\left(\bar{E}\right) - u_j\right)^2, \tag{3.35}$$

and then implemented a corresponding regularized strategy with the newly-determined λ.

3.6 Algorithm Design

Figure 3-3, once again, summarizes the overall process and the constituents necessary for elasticity reconstruction. The three main components are the experimental data, the model, and the estimation algorithm (see Chapter 2 for more details). In this section, we

briefly discuss building finite element (FE) models, leaving any details to the ensuing results sections, and finally introduce the estimation algorithm, boldfaced in the below flowchart.

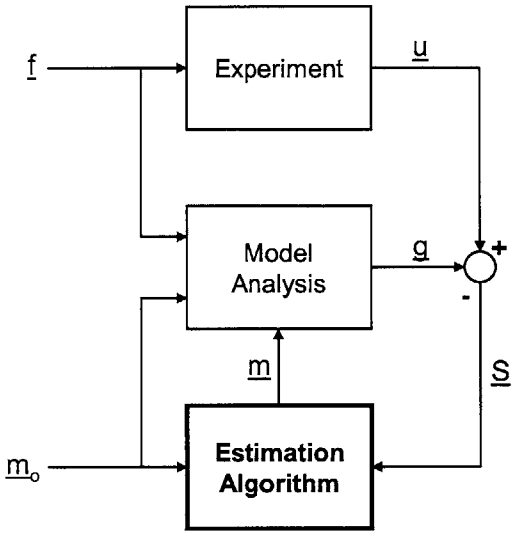


Figure 3-3. Mixed numerical-experimental, iterative solution for inverse problems.

The estimation algorithm was characterized with simplistic, 2-D, square-shaped FE models (see Figure 3-4). The left surface of the square models was constrained in all directions and a forcing load was applied to the right surface in the negative y-direction, mimicking quasi-static deformations. The models were meshed with quadrilateral elements, composed of 4 nodes per element. Inclusions of varying size, form, and Young's modulus were introduced inside the square-shaped models, such that the ratio between the background "tissue matrix" elasticity and inclusion(s) elasticity could be altered. The incompressibility constraint was enforced with a constant Poisson's ratio of 0.499, and, once again, linear elastic, isotropic material properties were assumed.

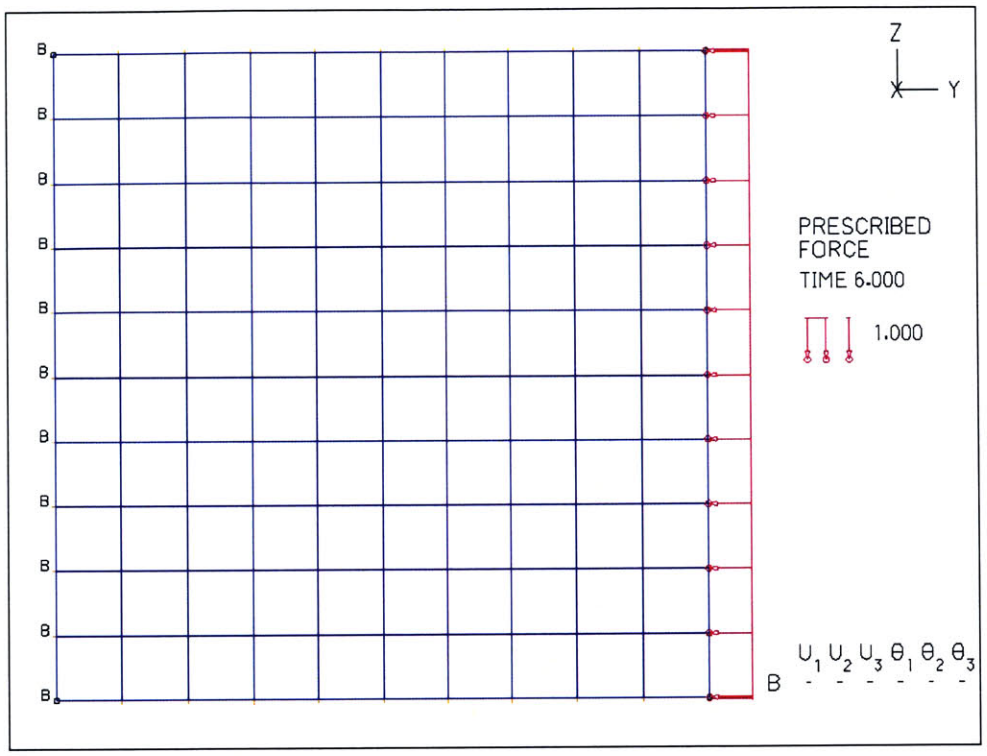


Figure 3-4. Finite element skeleton model used to characterize the Gauss-Newton method. A 2-D square-shaped solid composed of linear elastic, isotropic material properties and 4 node-per-element quadrilateral elements. A load was applied on the right surface in the negative y-direction and the left surface was constrained.

Although, in theory, most FEM software would suffice as a modeling tool, the two software requirements that allow for a completely automated system are, first, the ability to be run in batch mode and, second, easy access to its inputs and outputs. In this case, the FEM program of choice was a commercially-available software package, ADINA [76], which satisfies both requirements. Not only does ADINA provide batch mode commands for use of its programs without user input, but it is also operated by ANSYS-style text scripts, which can be read and written with standard i/o functions. As a result, an automated cycle of manipulating FEM input files, remotely running the models, and extracting necessary data from the output files helped expedite the model analysis portion of the elasticity reconstruction flowchart (see Figure 2-2).

The nomenclature describing the model inputs and outputs is mostly consistent with that introduced during the derivations of the NLS and Gauss-Newton equations. Essentially, the size of the vectors and matrices being manipulated are dictated by the number of model elements, m, and the number of model nodes, n. For example, the elasticity distribution, $\boldsymbol{E} = (E_1, E_2, \ldots, E_m)^T$, is a vector of isotropic Young's modulus values assigned to the m finite elements, while the displacements, $\boldsymbol{g} = (g_1, g_2, \ldots, g_{2n})^T$ and $\boldsymbol{u} = (u_1, u_2, \ldots, u_{2n})^T$, are nodal vectors carrying two-dimensional displacement values. In all FE models, $n > m$ (thus $2n > m$), making the Jacobian a $2n \times m$ rectangular matrix.

Due to lack of real, measured elastography data, characterizing the estimation algorithm involved simulating displacement data by choosing a target "unknown" elasticity distribution, solving the forward problem, and outputting the resulting displacement field. If desired, white Gaussian noise can then be applied to this "experimental" displacement field, and, in the following chapter, this is demonstrated.

The Gauss-Newton method and the FEM-integrating code were prototyped in MATLAB because of its versatility and because it provides both the necessary scientific and i/o functions and commands. Once tested, a far more efficient version was developed in C. The commented MATLAB and C source codes are presented in entirety in Appendices A.1 and A.2.

This thesis does not discuss all the details of the code development, since this process is specific to the software, the programming language, and the programmer. Figure 3-5, however, provides a general overview of the tasks the estimation algorithm is responsible for.

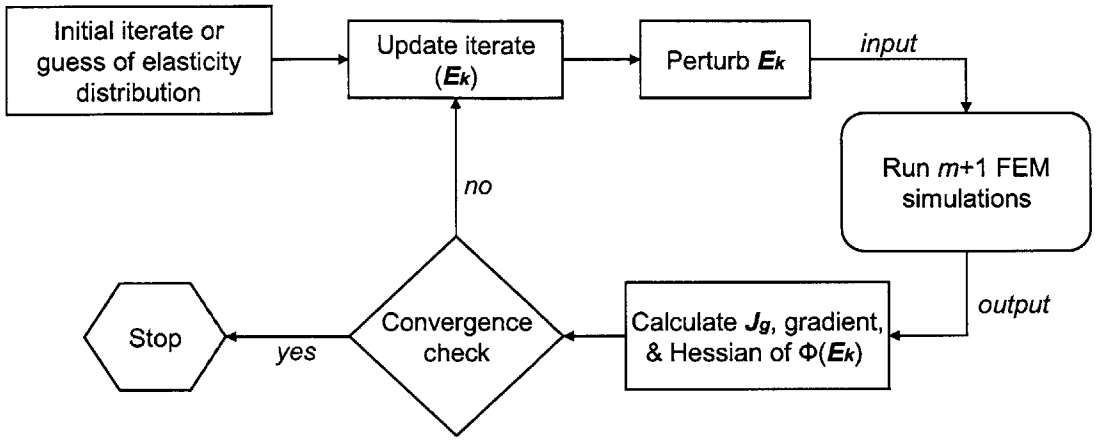


Figure 3-5. Flowchart of a Gauss-Newton method-based reconstruction algorithm. Upon an initial guess of the elasticity distribution (typically a constant distribution), the algorithm estimates the gradients necessary to direct it to a solution by perturbing elasticity distributions.

The algorithm is triggered with an initial iterate, or guess, of the elasticity distribution. Once initiated, it enters the overall iteration loop, emerging only if the *two* convergence criteria (for the norms of the objective function and elasticity step) are satisfied or if the algorithm exceeds a reasonable number of iterations, typically 20 or 25. Within the encompassing loop, the algorithm first updates the current iterate of elasticity distribution based on either the initial guess or the previous iteration's result. Then, with a nested loop, it perturbs each elemental component of the distribution by an amount ε = 5e-4 (a range of 1e-5 < ε > 1e-4 was determined to be the optimal perturbation amount based on Equation (3.17) and the fact that ADINA only outputs 6 significant digits of double precision data to its output files). The current iterate elasticity distribution along with the n perturbed distributions are each written into individual FEM input files (.in files) and run via batch commands of the form

c:\adina\aui\aui.exe -b -m 20mb prob01.in

c:\adina\adina\adina.exe -b -s -m 10mw -M 100mw -t 2 prob01.dat.

For a detailed explanation of ADINA batch command usage, the reader is referred to www.adina.com or [76]. By way of the .in files, ADINA is told to write the desired displacement fields automatically into output files, called .out files. More integration code was written in order to parse these .out files and sequester the displacement data. This displacement data is then used to build the Jacobian, column by column, as well as the other matrix and vector constituents of the Gauss-Newton method. Finally, a new elasticity distribution step is computed, the results are checked with the convergence criteria, and the loop repeats. See Appendices A.1 and A.2 for the source code, a more thorough understanding of how to interface and i/o with FEM software, and a depiction of the numerical methods.

3.7 Results

The elasticity reconstruction system was tested on a subset of three FE skeleton models (subject to the assumptions and model criteria listed in the previous section), each a 2-D square-shaped model and differing in the coarseness of its mesh. Table 3-1 summarizes the mesh sizes associated with each of the three FE model skeletons.

Model	Nodes	Elements	CPU Clock Time Per Iteration
1	25	16	32.5 s
2	49	36	81.2 s
3	121	100	262.8 s

Table 3-1. Finite element mesh specifications for the 3 skeleton models. Average CPU clock times per iteration (using MATLAB and a Pentium 4, 2 GHz with 1.00 GB of RAM) are also listed for the 3 models so one can quantify how much extra computational work goes along with additional iterations. Note that CPU times are almost entirely dedicated to solving the FE models, and this is also true for results in Chapter 4. However, because in the above case the models are essentially equally simplistic, the increase in CPU time per iteration is attributed to the increase in FEM solutions per iteration. The models used in Chapter 4, on the other hand, were more complex and computationally intensive; as a result, we did not list corresponding CPU times because their potential variability with different CPUs and platforms would not be as reflective.

A preliminary investigation of the system's ability to reconstruct homogeneous Young's modulus distributions for the three different-sized models was performed. The specific goals of this investigation were (i) to assess the limits of the algorithm for simple elasticity distributions and (ii) to confirm the hypothesis that systems of larger dimensions tend to yield more singular values with small magnitudes and hence are more ill-posed.

Following these preliminary conclusions, this thesis presents reconstruction results for models in which inclusions of varying size, form, and elasticity were introduced, the goal being to evaluate how the algorithm handles inhomogeneous distributions and how helpful/unhelpful regularization is in this endeavor.

3.7.1 Homogeneous Elasticity Distributions

Model 1

Initial iterate guesses of 25%, 50%, and 100% above the target modulus distribution value (E = 100) were inputted into the algorithm in an effort to reconstruct the target distribution (See Table 3-2).

Initial Iterate [E]	Target Value [E]	Iterations	λ
125	100	5	-
150	100	6	-
200	100	14	-
200	100	10*	4.72×10^{-10}

Table 3-2. Model 1 reconstruction experiments. Initial iterates were assigned as constant elasticity distributions. * indicates the presence of Tikhonov regularization. Note that experimentation included initial iterates above *and* below the target modulus distribution, and since they yielded analogous convergence trends, for simplicity, we chose to present the representative and noteworthy results and patterns.

Due to the combination of a homogeneous distribution and a model of such relatively coarse mesh, the estimation algorithm had little difficulty obtaining the target modulus

distribution. In fact, regularization was not absolutely necessary for initial iterates through 100% above the target Young's modulus value; in other words, the algorithm recognized a unique solution and could converge on its own. Regularization was added, regardless of this observation, to assess its benefit in making convergence more efficient. Via the non-regularized solution, a subsequent regularization weighting parameter (λ) can be estimated. Based on the norm of error between the actual and convergent model-predicted displacement fields (see Section 3.5), a λ of 4.72×10^{-10} was predicted as an appropriate balance between the original residual norm and the regularization term. True to its role, the Tikhonov regularization term successfully reduced the number of iterations necessary for convergence from 14 to 10 for an initial iterate of 200. For all homogeneous modulus distribution cases, a discrete first derivative operator was chosen for the Tikhonov $\boldsymbol{L}$ matrix. However, rather than two non-zero terms (diagonal and non-diagonal) appearing per row, as shown in Equation (3.29), the $\boldsymbol{L}$ matrix was tailored to the FEM connectivity matrix. In other words, the operator $\boldsymbol{L}$ was designed to take discrete first-order derivatives of all neighboring elements, such that each element is "smoothed" by each of its neighboring elements (see Figure 3-6).

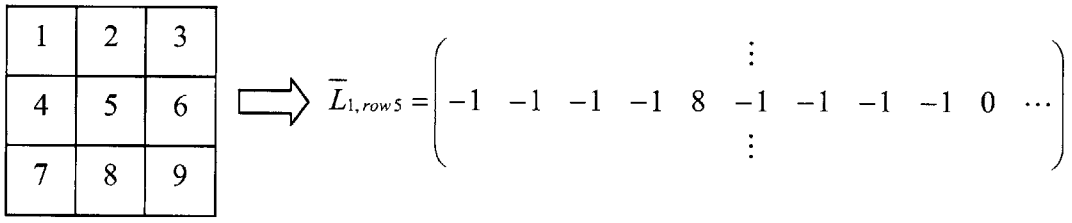


Figure 3-6. A first-order derivative operator based on all neighboring elements. The Tikhonov regularization $\boldsymbol{L}$ matrix was tailored to the FEM connectivity matrix, such that each element was smoothed by all neighboring elements.

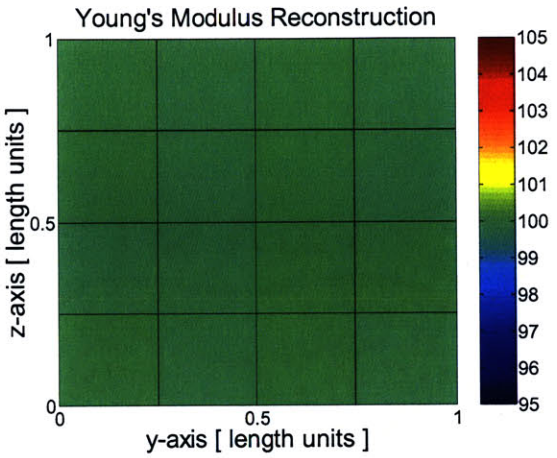


Figure 3-7. Model 1 homogeneous reconstruction results. The experiments of Table 3-2 all yielded accurate homogeneous reconstructions of the target elasticity.

Figure 3-7 is a 2-D colormap of a typical, converged solution for this particular model skeleton. The smooth, homogeneous result indicates successful reconstruction and avoidance of instability through matrix inversion. One important observation from the results of this fairly trivial inverse model problem is that as initial iterates deviated farther and farther from target elasticity values, convergence became more difficult. As forewarned in Section 3.4.3 (The Ill-Posed Problem), neglect of the second order term to abridge and simplify the Hessian can lead to severe ill-conditioning, particularly when iterates are far from the solution. In addition, although not encountered with this model, the Gauss-Newton method is more prone to falling into local minima when iterates are distant from the solution.

The other key observation will be made with the presentation of the finer mesh model results, and it will help predict how severely larger matrix systems ill-condition the inverse problem as a result of increased small singular values. For the purposes of comparison, it should be noted that the number of elements and nodes associated with Model 1 yielded a Jacobian of size $2n \times m$, or 50×16, and a Hessian of $m \times m$, or 16×16. Additionally, it should be noted that a consistent and relatively stringent convergence

criteria, which can be found in Appendix A.1, was mandated across all Gauss-Newton reconstruction experiments. Given data lacking noise, the intent of these experiments was to test the algorithm's limitations and to compare results over various models, model sizes, and distributions, and, to do this, consistent and strict criteria were necessary.

Model 2

Elasticity reconstruction efforts for Model 2, which was meshed with more than double the number of elements of Model 1, were more rigorous. As compared with reconstruction results for Model 1, analogous Model 2 problems consistently required more iterations before converging (see Table 3-3), confirming the hypothesis surrounding larger scale systems.

Initial Iterate [E]	Target Value [E]	Iterations	λ
125	100	12	-
125	100	4*	3.00×10^{-12}
150	100	**NC**	-
150	100	5*	3.00×10^{-11}
200	100	**NC**	-
200	100	10*	2.62×10^{-9}

Table 3-3. Model 2 reconstruction experiments. Initial iterates were assigned as constant elasticity distributions. * indicates the presence of Tikhonov regularization and **NC** indicates no convergence.

Furthermore, devoid of regularization, the algorithm failed to locate solutions for initial iterates that deviated by more than 50% of the target modulus value, where solutions were obtained for Model 1. When equipped with Tikhonov regularization, however, solutions were obtained for initial iterates of 150 and 200 within 5 and 10 iterations, respectively. It should be noted that, although not always explicitly stated, the regularization weighting terms for all these homogeneous distribution problems were estimated based on the description that appears in Section 3.5.

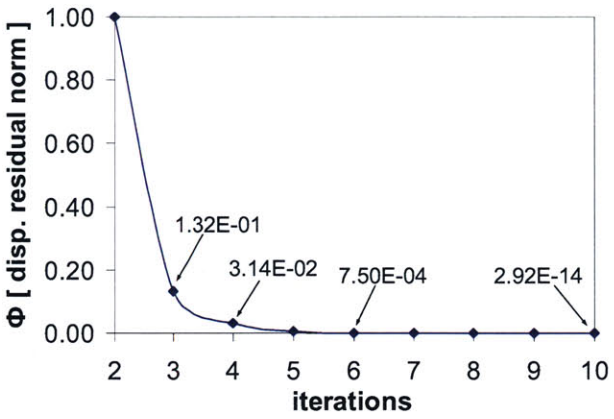


Figure 3-8. Reconstruction displacement error versus iteration number. This convergence plot demonstrates near *q-quadratic* Newton convergence through the relative displacement error.

Figure 3-8 is a Gauss-Newton convergence plot, showing the norm of the displacement field residual being minimized as the Newton search progresses (in particular, this plot tracks the convergence progress of the final search shown in Table 3-3: a regularized search with initial iterate of 200). As explained earlier, the norm of the displacement field residual serves as an appropriate replacement for the error in the solution, since the actual solution is not accessible. Successful Gauss-Newton search histories should display what is known as *q-quadratic* convergence, where the solution residual will be roughly squared with each iteration [68]. Once again, although the solution residual is undeterminable, the nonlinear residual—the displacement field residual—is an insightful, alternative gauge and should also be roughly squared with each iteration. The plot shown in Figure 3-8 does not adhere exactly to the quadratic relationship, however, its trend is not far off and exemplifies very successful Newton convergence. To obtain exact *q-quadratic* convergence, two criteria must be fulfilled: the Jacobian must be well-conditioned and machine precision errors in function evaluations

cannot be prevalent. The convergence history of Figure 3-8 documents a previously ill-conditioned system that was aided with Tikhonov regularization, so the deviation from

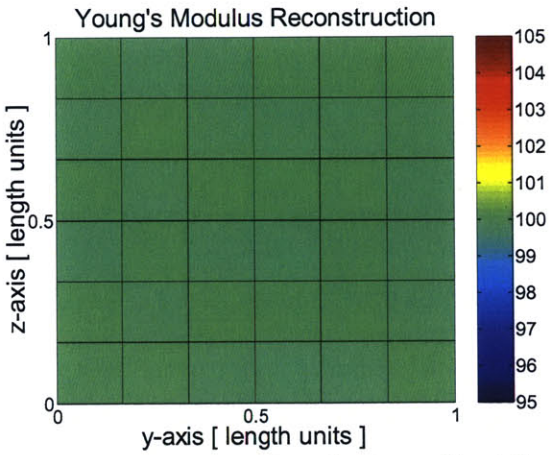


Figure 3-9. Model 2 homogeneous reconstruction results. All converging experiments of Table 3-3 yielded accurate homogeneous reconstructions of the target elasticity.

exact *q-quadratic* convergence, at least for earlier in the iteration history, can be attributed to slight deficiencies in the regularization technique's ability to fully restore conditioning to the ill-posed problem. Furthermore, as noted before, the elasticity reconstruction system suffers from small machine precision errors associated with outputting FEM data that can lead to convergence stagnation late in the iteration history (See Section 3.4.3).

Model 3

Reconstruction results for Model 3 were consistent with the observations that deviant initial iterates and finer mesh sizes (i.e., larger model systems) yield slower and eventually no convergence (see Table 3-4).

Initial Iterate [E]	Target Value [E]	Iterations	λ
125	100	**NC**	-
125	100	4*	7.70×10^{-9}
150	100	**NC**	-
150	100	5*	1.98×10^{-3}
200	100	**NC**	-
200	100	10*	1.96×10^{-3}

Table 3-4. Model 3 reconstruction experiments. Initial iterates were assigned as constant elasticity distributions. * indicates the presence of Tikhonov regularization and **NC** indicates no convergence.

In fact, Tikhonov regularization was required for stability and for engendering unique solutions in all the attempted initial iterates. Furthermore and to no surprise, an increase in the regularization weighting factor, λ, follows when convergence is made more difficult because it is determined solely from the non-regularized, "converged" nonlinear (displacement) residual. Thus, as the problem becomes more ill-posed, convergence becomes challenging, and the non-regularized, "converged" residual worsens. As a result, our expression for λ deems that more weight should be allocated to the smoothing, regularization term. This simplistic model for λ does not always produce the most appropriate or accurate predictions. In the next section, results for models with inclusions are presented and we show that incorporating some intelligence in the choice of λ can help provoke convergence.

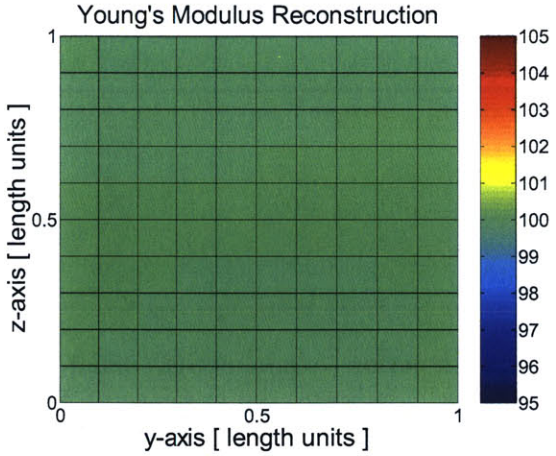


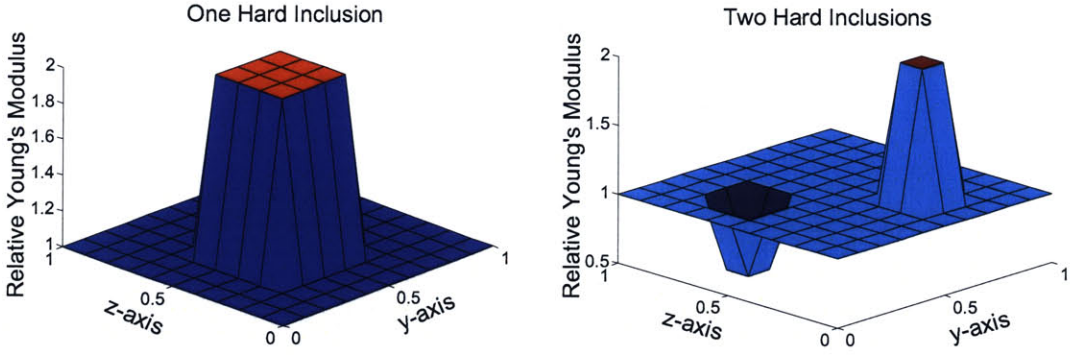
Figure 3-10. Model 3 homogeneous reconstruction results. All converging experiments of Table 3-3 yielded accurate homogeneous reconstructions of the target elasticity.

3.7.2 Inhomogeneous Elasticity Distributions: Inclusions

After confirming the validity of the Gauss-Newton method for the three model skeletons, a variety of inclusions was inserted into the finest mesh of the three (Model 3), in an effort to reconstruct inhomogeneous elasticity distributions. An inclusion is a general term referring to an area of certain shape and size *within* and *distinguishable* from the background material matrix based on its dissimilar elasticity or material properties. The algorithm's ability to locate inclusions by predicting their elasticity is of great interest to atherosclerotic plaque identification. Vulnerable plaques, as described in Chapter 1, tend to possess regions of high stress, or stress concentrations. Furthermore, at the root of many stress concentrations, are the size, relative location, and stiffness of inclusions, such as calcification and lipid pools. A valuable elasticity estimation algorithm, thus, should generate a solution that, based on the inclusions it predicts and builds, identifies the critical regions of high stress.

Hard Inclusions

The first genre of inclusions we experimented with was hard inclusions—those characterized by abrupt changes in elasticity. Figure 3-11 shows 3-D surface plots of square-shaped models, where one hard inclusion was inserted into the first model and assigned a modulus value of double the background and two hard inclusions were added to the second model and assigned modulus values of double and half the background.



A B

Figure 3-11. 3-D surface plots of target hard inclusions. A. One hard inclusion characterized by a sharp change in elasticity. **B.** Two hard inclusions, one with Young's modulus greater than that of background material and one with Young's modulus less than that of the background material.

It is when dealing with such inclusions that one truly appreciates the definition and power of Tikhonov regularization—incorporating *a priori* assumptions about the size and smoothness of the desired solution. One way of incorporating these assumptions is by modifying the regularization matrix $\boldsymbol{L}$ accordingly. In the case of hard inclusions, the transition between material regions is not smooth and thus an effective regularization matrix should also exhibit analogous continuity breaks. For example, the $\boldsymbol{L_1}^*$ matrix shown in Figure 3-12 possesses three distinct regions corresponding to three hypothetical material regions that are presumably separated by sharp changes in elasticity. An alternative, more physical explanation for designing $\boldsymbol{L_1}^*$ this way is to ensure that hard

inclusions are "smoothed" *principally* by the elements that constitute it and to avoid penalties for erroneous smoothing by elements in transitory or other regions.

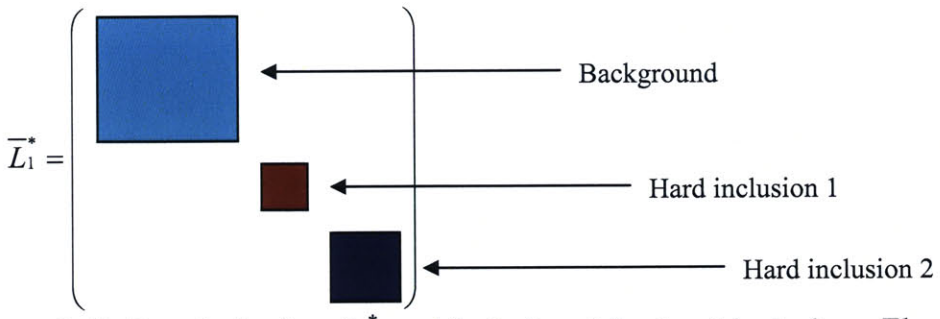


Figure 3-12. Regularization L_1^* matrix designed for hard inclusions. Element values are smoothed not by *all* neighboring elements but by neighboring elements of same material region (e.g. an inclusion).

Table 3-5 summarizes the key results and parameters for a set of four inverse problems: two single-inclusion models (Figure 3-11A) and two double-inclusion models (Figure 3-11B). Similarly to the homogeneous distribution problems, a constant initial elasticity distribution was inputted, leaving the algorithm the task of locating the convergent discrete elasticity values.

Initial Iterate [E]	Hard Inclusions	Iterations	λ
125	1	6	7.22×10^{-9}
175	1	9	7.22×10^{-9}
125	2	8	7.22×10^{-9}
175	2	13	7.22×10^{-9}

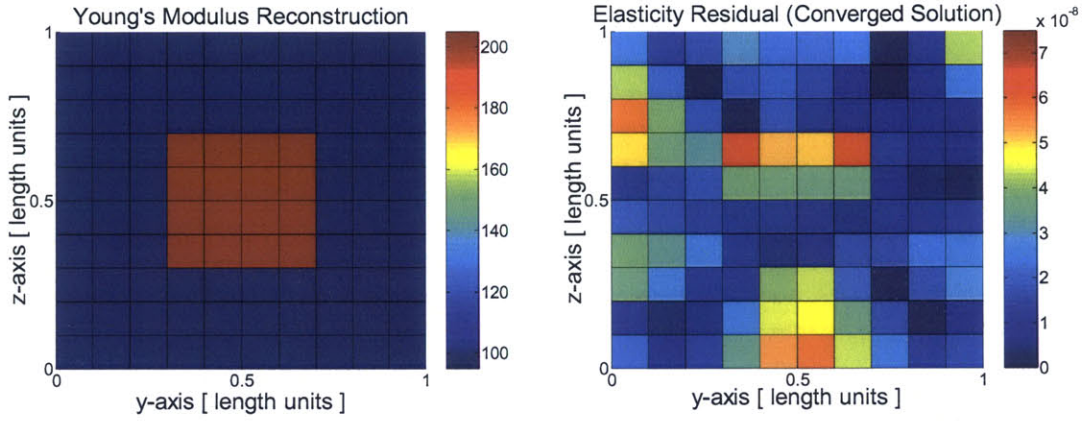
Table 3-5. Reconstruction experiments for one and two hard inclusions. Tikhonov regularization was necessary for all the experiments and λ was determined via Equation (3.35).

Figures 3-13 and 3-14 provide various visual forms of reconstruction results. With the regularization weighting parameter listed in Table 3-5 and model-specific $\boldsymbol{L_1}^*$ matrices, the Gauss-Newton method was highly successful in elasticity estimation of models with hard inclusions. Although in reality target elasticity distributions are not available, for these theoretical, testing experiments, the target values, used to create "actual" displacement fields, were employed to gauge the ability of the method. For example,

Figure 3-13B is a plot of the elemental errors between converged and target modulus values,

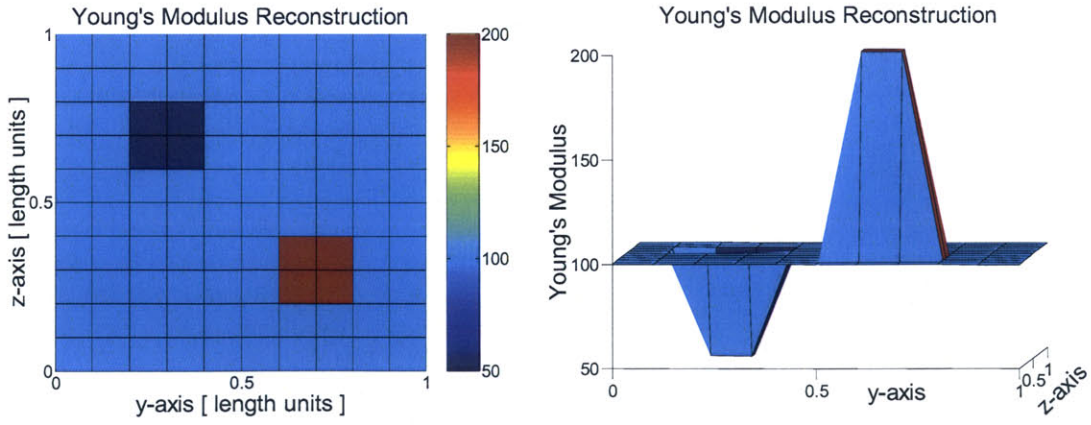
$$\overline{E}_{residual} = \frac{\left|\overline{E}_{converged} - \overline{E}_{target}\right|}{\left\|\overline{E}_{target}\right\|}, \tag{3.36}$$

as per the first experiment listed in Table 3-5. Over the entire model, the discrete, absolute errors remained below 1.00×10^{-7}.



A B

Figure 3-13. Reconstruction result colormaps (one hard inclusion). A. Converged Young's modulus reconstruction solution. **B.** Discrete residual between target and converged elasticity distributions, calculated using Equation (3.36).



A B

Figure 3-14. Reconstruction result colormaps (two hard inclusions). A. Converged Young's modulus reconstruction solution. **B.** 3-D view of the same converged solution.

Spatially Continuous Inclusions

Realistic plaque, arterial, or soft tissue inclusions rarely conform to the abrupt stiffness jumps associated with ideal hard inclusions. Even when the transition from background to inclusion is steep, elastic continuity is nonetheless present due to the inherently continuous nature of soft tissue material properties. As a result, models incorporating spatially continuous inclusions were built and the results of their corresponding inverse problems are presented and discussed in this section.

It is common to model spatially continuous inclusions as normal Gaussian distributions [51]. For an inclusion with spatially increasing elasticity values, the 2-D modified normal Gaussian distribution equation is as follows

$$E(x,y)=f(x,y)=\left[\frac{1}{\beta_1}\left(\frac{1}{\sqrt{2\pi\sigma_x\sigma_y}}\right)e^{-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2}+\frac{(y-\mu_y)^2}{2\sigma_y^2}\right)}\right]+\beta_2, \tag{3.37}$$

where β_1 and β_2 are constants used to resize the graph to a desirable $E_{\max}$ and $E_{\min}$, respectively, σ_x^2 and σ_y^2 are the distribution variances in x and y, and μ_x and μ_y are the x and y mean values. The MATLAB script file written to allocate the normal Gaussian distribution elasticity values to a grid of elements is presented in Appendix A.3. In order to obtain a relatively steep change in elasticity and retain semblance of a background, variances of

$$\sigma_{x,y}^2=\frac{1}{\sqrt{2\pi}} \tag{3.38}$$

were chosen. Resizing constants of $\beta_1 \approx 2.27$ and $\beta_2 = 1$ were used to fix the $E_{\min}$ value at 1 and stipulate that $E_{\max}=2E_{\min}$. Finally, μ_x and μ_y values of 0.5 ensured that the

distribution peak appeared in the center of the square grid. The result is the target elasticity distribution shown in Figure 3-15.

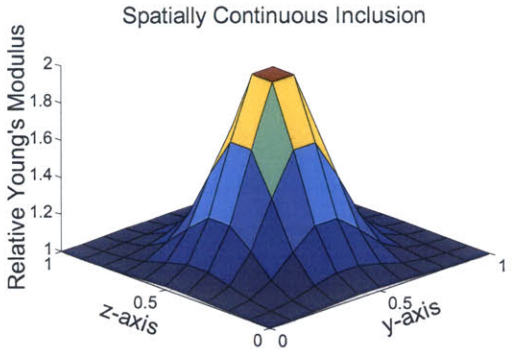


Figure 3-15. 3-D surface plot of target spatially continuous inclusion. Inclusion was generated with the 2-D normal Gaussian distribution equation. The inclusion's peak modulus value reaches 2 times the background modulus value.

As predicted, a more realistic view and model of inclusions must diminish the seemingly omnipotent capacity of regularization techniques. Of the numerous experiments simulated with this normal Gaussian distribution, two are particularly insightful (see Table 3-6). Because of the relatively smoother nature of the spatially continuous inclusions, the $\boldsymbol{L_1}^*$ matrix was abandoned for the original and more traditional $\boldsymbol{L_1}$ first-order derivative operator, depicted in Figure 3-6, forcing the Gauss-Newton method to locate the "boundaries" of inclusions via their stiffness.

Initial Iterate [E]	Iterations	λ
125	3	$\mathbf{1.26\times10^{-9}}$
125	6	1.00×10^{-12}

Table 3-6. Reconstruction experiments for one spatially continuous inclusion. The boldfaced λ indicates the suggestion by Equation (3.35).

For the first experiment listed in Table 3-6, $\lambda = 1.26\times10^{-9}$ was applied *as recommended* by the mean squared displacement residual of the unregularized, convergent solution. In its defense, this method of selecting a regularization weighting parameter produced a system that converged on a nontrivial elasticity distribution within

3 iterations—the solution satisfied both convergence criteria: a sufficiently small Newton step size, $\left\| \overline{E}_{k+1} - \overline{E}_k \right\|$, and displacement residual, $\left\| \overline{g}(\overline{E}) - \overline{u} \right\|$. However, when the elemental elasticity errors between converged and target values were plotted (see Figure 3-16A), it follows that the errors ascend along with the modulus values of the inclusion. In other words, noticeable errors were obtained at the center or maximum modulus value of the inclusion and faded in the background. This result was somewhat troubling, at first, because it was not wholly a result of loose convergence criteria; in fact, the criteria, as described earlier, tended to the more stringent side because of the lack of data noise and experimental complications. Rather, a unique solution to this system existed, whereby an elasticity distribution slightly differing from the target distribution produced a displacement field "numerically identical" to that of the target distribution.

Recall that the role of λ is to balance the influence of the original NLS term and the smoothing term. In the first simulation, the unregularized solution predicted a fairly large λ value, indicating that smoothing was, indeed, needed. However, it was the overpowering influence of the smoothing term that dragged the inclusion's modulus values, especially the larger ones, down toward the background values (see Figure 3-16C). In response, λ was reduced 3 orders of magnitude and a second simulation was attempted. The result was a return to virtually error-free and well-shaped elasticity reconstruction curves (See Figures 3-16B and 3-16D). The conclusion drawn here is that while regularization is needed to stabilize the system and establish uniqueness, appropriately designing it based on *a priori* knowledge of the solution and experimentation can lead to more accurate, converged solutions.

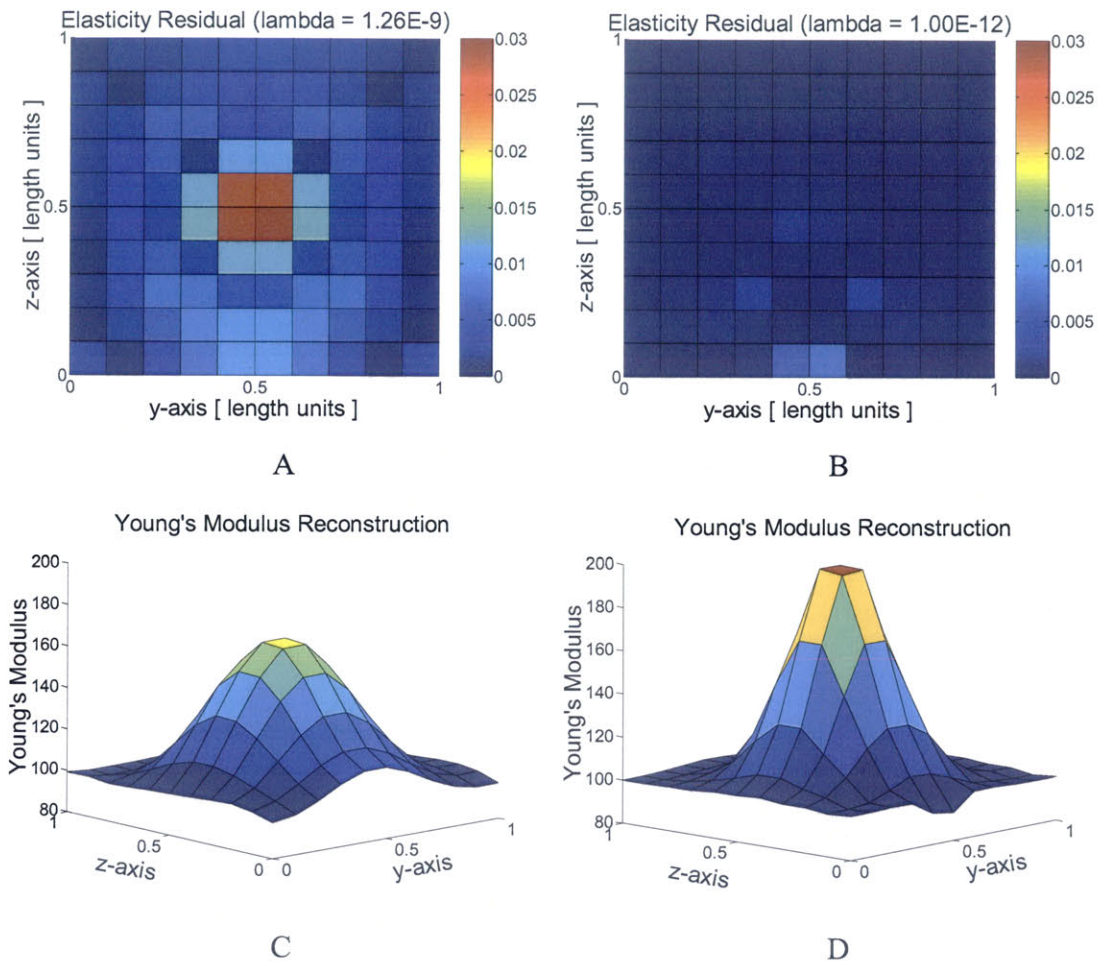


Figure 3-16. Reconstruction result colormaps (spatially continuous inclusion). A. Discrete residual between target and converged elasticity distributions, exhibiting visual errors at the apex of the inclusion. **B.** Discrete elasticity residual for smaller value of λ, exhibiting minimal errors. **C.** 3-D iso view of the reconstruction results reflects a relatively large influence of smoothing. **D.** 3-D iso view of the reconstruction results when smoothing is not as prevalent.

Doubtless, the Gauss-Newton method has the potential to be a powerful diagnostic tool in atherosclerotic plaque identification and evaluation, providing a discrete picture of an artery's elasticity and vulnerability. In fact, its abilities were confirmed in the basic reconstruction problems we built and experimented with. Currently, limitations in the search and difficultieith accurate data acquisition have, to some degree, prevented this

potential from becoming realized. Many of the limitations associated with the Gauss-Newton method, some of which became evident in the presentation of reconstruction results above, stem from a lack of robustness and intelligence in the searching strategy. For instance, in the cases where inclusions take on Young's modulus values of more than a factor of 2 or 3 of the background material (which is a realistic situation in soft tissue and diseased arterial tissue), the algorithm is likely to falter until appropriate regularization parameters are found and used. The next chapter introduces a technique, new to arterial elasticity reconstruction yet known for its robustness, which promises to be helpful.

Chapter 4

A Genetic Algorithm Approach

4.1 An Alternative Strategy

David Goldberg, an early pioneer in the advancement of genetic algorithms, viewed the goals of optimization in slightly different light than the traditional views of improving performance toward some optimal point. Rather, he was concerned with optimization criteria that better served more humanlike decision-making. "We never judge a businessman by an attainment-of-the-best criterion; perfection is all too stern a taskmaster," he states [72]. However, traditional optimization strategies are often built around this naïve and sole focus on the *destination* of the search. A shift toward focusing more on the *process of improvement* rather than the *destination* is more natural and potentially helpful. This way, we concern ourselves with doing better relative to others, which is a more robust and intelligent gauge than locating the best overall solution. Nicely summarized, "It would be nice to be perfect: meanwhile, we can only strive to improve" [72].

Genetic algorithms, developed by John Holland and colleagues, are search methods that simulate biological evolution through naturally occurring genetic operations on chromosomes [77]. They apply the Darwinian principle of survival of the fittest on string structures to build unique searches with elements of both structure *and* randomness. As a result, genetic algorithms tend to be much more efficient than purely random searches, such as the simple random walk, because it intelligently exploits historical information to

speculate on new search points [72]. Recognizing that biological systems are extremely robust, efficient, and flexible, the goal is to mimic them and harness these properties for use in artificial systems. A key advantage to this relatively new subgroup of search algorithms is they have been theoretically and empirically proven to give very robust searches in complex spaces and are, in principle, extremely simple to devise [77].

Categorizing optimization techniques by the tradeoff between efficiency and robustness, conventional calculus-based searches lean heavily in the direction of efficiency while random searches excel in robustness. Genetic algorithms, however, are designed with both features in mind, thus, in principle, they represent powerful hybrids of these two search necessities.

According to Koza [78], the evolutionary process can be summarized by four simple rules:

- Entities have the ability to reproduce.
- A population of such entities exists.
- The population consists of a diversity of these entities.
- The diversity influences the survivability of these entities in a given environment.

These rules are self-evident in nature, in that diversity materializes as variation in chromosomes and that this variation affects an entity's structure and behavior. Furthermore, these variations which translate to differences in fitness, when in the context of an environment, lead to visible changes in entity structure over many generations.

The genetic algorithm (GA) harnesses these simple rules in order to produce analogous, relatively complex effects. Genetic algorithms embark on a predefined initial population of individuals, typically created at random from a field of possible search solutions. Through pseudo genetic operations, such as mating pool selection, crossover reproduction, and mutation, the fittest individuals in the population survive to the next generation and facilitate the proliferation of new individuals. Based on a simple set of rules and parameters dictating how these genetic operations function, the algorithm crawls and learns the surface of the search space on its own in an extremely efficient manner.

This thesis borrows an elementary example from Koza [78] of a GA in use, in hopes that it will provide an illustrative understanding of how and why the algorithm functions.

The Hamburger Restaurant Problem

A GA was applied to optimize a business strategy, specifically, to find the best strategy for a chain of four hamburger restaurants. In this example, each restaurant is faced with three decisions:

- Price of the hamburger: $0.50 or $10
- Choice of drink: wine or cola
- Speed of service: slow or fast

Additionally, it is assumed the manager is unaware of the maximum profit attainable and receives data only in the form of each restaurant's profits per week. In other words, he has no way of quantifying a profit "gradient" associated with changing each variable from the list of decisions above. Instead, he must experiment with the decisions via his

four restaurants in order to understand the environment and how it influences his business.

A first step in implementing a GA is to decide on a representation scheme for the entities, or individuals, in the population. The two typical representation schemes are binary and continuous form. In the former case, entities are translated into binary numbers while, in the latter case, some pattern of continuous numbers is used to represent the individuals. For this problem, the decisions involve a choice between two strategies, and so binary representation is the most appropriate, where each digit of the binary number signifies one of three decisions made. After designing a suitable representation scheme, an initial population of possible (often randomly-chosen) solutions is generated (see Table 4-1).

Restaurant	Price	Drink	Service	Binary representation
1	$10	Cola	Fast	011
2	$10	Wine	Fast	001
3	$0.50	Cola	Slow	110
4	$10	Cola	Slow	010

Table 4-1. Representation scheme for the hamburger restaurant problem.

Next, each individual in this generation must be tested against the unknown environment. To do this, an objective, or fitness, function is developed. Choice of objective function is entirely problem-specific; however, it must reflect an individual's survivability in the environment. For the problem at hand, let the fitness measure merely be the value of each individual's binary representation. This may seem an arbitrary fitness transformation, however, its aptness will be revealed later and this process will become clearer when choosing an objective function for the inverse elasticity problem. With the given objective function, fitness measures are assigned to the above individuals of Generation 0 and consequently a pseudo rank is born (see Table 4-2).

i	String X_i	Fitness $f(X_i)$
1	011	3
2	001	1
3	110	6
4	010	2
	Total	12
	Worst	1
	Average	3.00
	Best	6

Table 4-2. Fitness measure values for the four initial business strategies.

Recovering fitness measures for a population of individuals is analogous to probing the environment for clues that help reveal where the optimal points lie. Equipped with these newfound fitness measures, the GA's next step is to create a mating pool from which offspring for the new generation will arise. Once again, selection of the mating pool is problem-specific, with the stipulation that it involves some function of the fitness measures; in other words, the fitter individuals are more likely to survive than the less fit ones. Section 4-4 discusses "Roulette wheel" mating pool selection and modifications to it. For now, assume the probability an individual is chosen for the mating pool is proportional to the clout of its fitness measure with respect to the rest of the population—this is the GA's form of survival of the fittest. Hence, as illustrated in Table 4-3, a *fitness-biased* probabilistic means of selection is possible.

i	String X_i	Fitness $f(X_i)$	$\frac{f(X_i)}{\sum f(X_i)}$	Mating pool	$f(X_i)$
1	011	3	0.25	011	3
2	001	1	0.08	110	6
3	110	6	0.50	110	6
4	010	2	0.17	010	2
	Total	12			17
	Worst	1			2
	Average	3.00			4.25
	Best	6			6

Table 4-3. One possible mating pool resulting from the fitness values.

Based on this simplified version of mating pool selection, the GA's direction and success is immediately evident. Not only was the population-average fitness measure

increased from 3.00 to 4.25, but the least fit individual was eliminated, effectively narrowing the pool of potential solutions.

On a fitter and more specific pool of parents, crossover reproduction is performed to generate the offspring that will, in part, compose the following generation. This intelligently introduces new, not entirely random, points in the search space to be tested. As with mating pool selection, the parents participating in reproduction are usually selected proportionate to their fitness. The genetic operation of crossover is, like many components of a GA, problem-specific; however, it involves begetting one or more offspring from the string fragments of two parents. In binary representation, this is a trivial task: merely separate the parents' strings at a designated breakpoint and reconstruct the fragments to form offspring, as shown in Figure 4-1.

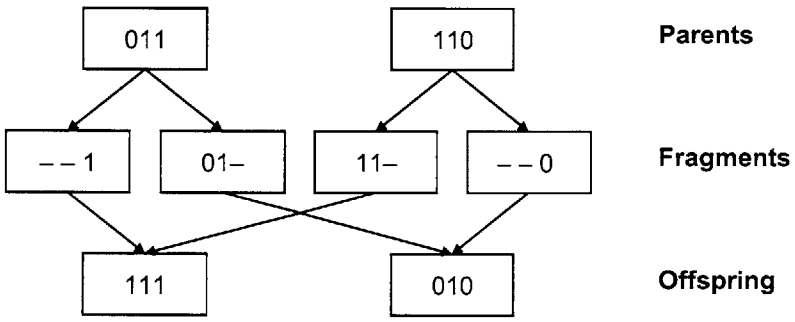


Figure 4-1. Crossover reproduction yielding two offspring from two parents.

The parents used in Figure 4-1 were taken from the mating pool of Table 4-3, and one can immediately see that offspring 111 will be fitter than its parents when it comes time to assign fitness measures and select the mating pool for Generation 1.

The genetic operation mutation offers yet another means of introducing diversity and elements of randomness in the group of points being searched, further reinforcing the robustness of the GA. In general, this can be done by randomly altering 1, 2, ... *N* string digit(s) of a relatively small percentage of the population.

Equipped with these basic genetic operation rules, the GA iterates over several generations of search points until satisfying convergence criteria typically involving the makeup of a population. A discussion of the mathematical foundations behind the success of genetic algorithms, involving similarity template (schemata) theory, is beyond the scope of this this thesis. However, if interested, the reader is referred to [72, 78].

4.2 Problem Statement Revisited

Consider a model with m unknown elasticity parameters, $\overline{E}_m$, and a set of output values $\overline{g}(\overline{E}) = (g_1, \ldots, g_n)^T$ that are to be compared with a set of experimentally-determined values $\overline{u} = (u_1, \ldots, u_n)^T$. The objective is to accurately estimate the model parameters, based on the error between $\boldsymbol{g}$ and $\boldsymbol{u}$. However, in this case, m and n will not take on the variables traditionally assigned to them (i.e., for a Gauss-Newton method, m = model elements and n = degrees of freedom × model nodes). Instead, we will find that m becomes some intelligently reduced value and n is effectively decoupled according to choice of m (see Section 4.4).

Once again, we model incompressible, linear elastic materials, subject to the plane strain assumption, and undergoing static deformations. Additionally, we exploit FEM to transform the given geometry, material properties assigned at model elements, and boundary conditions into the displacement or strain outputs ($\boldsymbol{g}$)

In many situations, the traditional m unknowns can be reduced to $m \ll n$ parameters, yielding a system where the number of parameters being fit or optimized is reasonable given the number of measurements available. This is one way our alternative strategy serves to improve the conditioning of the inverse problem. For instance, Locatelli *et al.*

[79] analogously optimized structural and control parameters in aerospace and precise system smart structures by carrying out a modal reduction of their problem into a 3-mode optimization problem.

In arterial elasticity, *a priori* knowledge of the specimen's geometry and composition can be exploited to help *reduce* and solve the problem, similar to the way regularization was used to condition the Gauss-Newton method. Typically, the geometries are constructed from imaging and then segmented by cardiovascular imaging experts. This segmentation process does depend on the expert's choice of boundary locations and is therefore not an absolute and necessarily consistent procedure. However, as Yabushita *et al.* [80] showed with optical coherence tomography (OCT) of diseased arterial segments, certain imaging criteria and characteristics, such as light backscattering, define different plaque types and components. When compared with correlating histological sections, they found that OCT images are, in fact, capable of accurately characterizing plaque components [80]. And the same is true for other modalities (e.g. IVUS). Consequently, for the parameter estimation problem, a map of the arterial components, as demonstrated by Figure 4-2, is used to accurately and intelligently reduce the problem into a manageable number of parameters.

In addition to remedying the ill-conditioning, model reduction simplifies the solution strategy. A less formulaic approach can be adopted, specifically one that is robust, seeks global minima, handles noise more effectively, and even allows versatility in the material model.

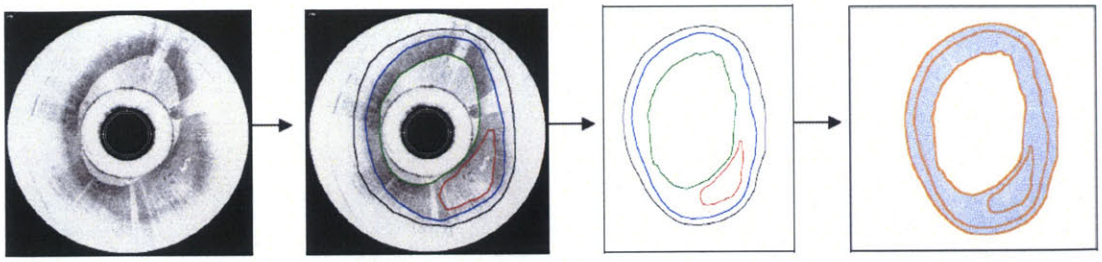


Figure 4-2. From OCT to FEM. OCT image of an arterial cross-section, segmented into distinct components (e.g. plaque morphology), and finally meshed as a finite element model. OCT images courtesy of Brett E. Bouma and the Wellman Center for Photomedicine (Massachusetts General Hospital).

The goal of this investigation was to employ an adaptive technique, in the form of a genetic algorithm, in conjunction with FEM, to formulate a simpler solution for parameter estimation that avoids the problems inherent to discrete gradient-based methods. The overall algorithm would serve as an effective and rapid first pass at elasticity imaging, and its role would be that of a regularization aid or accurate initial guess for traditional inverse problem solutions or, if sufficient, as a replacement.

4.3 System Components

The approach taken was a multiobjective optimization view, quite similar to traditional iterative inversion approaches. With multiple parameters to optimize via several software platforms, the first step was to integrate all the efforts. Langer *et al.* [81], in unrelated work, proposed a similar integration scheme to control all applications via an Application Manager, capable of accommodating other software as well as providing the libraries necessary to manipulate large datasets. The Application Manager's tasks included controlling the inputs and outputs, executing the FEM models, structuring the data, and implementing our optimization method. In the presented work, MATLAB was used to prototype the system because of its ease of use and its interfacing flexibility.

With the abundant number of FEM software packages available to simplify the task of computational modeling, we, once again, harnessed a software platform to model and solve the forward problem and focused instead on implementing a simply integrated system. The commercially-available software package, ADINA [76], was used for FEM.

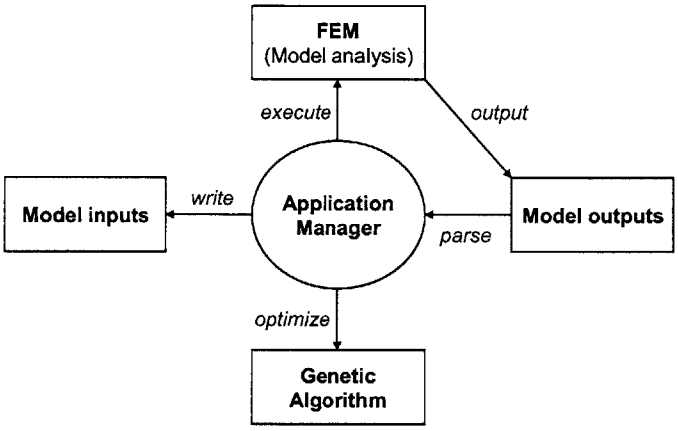


Figure 4-3. Overall combined FEM-GA system architecture. The Application Manager carries out a circular set of tasks, including executing the FEM program and interfacing between the FEM program and the optimization algorithm.

In place of the estimation algorithm from the process diagram of Figure 2-2, we chose to implement a GA on account of this technique's success with relatively small parameter search spaces and with locating global solutions.

Figure 4-3 provides an alternative architectural picture to Figure 2-2 of an elasticity parameter estimation system. Given a segmented geometry and the experimental data, the Application Manager carries out the circular set of tasks briefly discussed above until convergence.

4.4 Algorithm Design

GA design is highly problem-specific; however, as illustrated in Section 4.1, there are underlying principles consistent in every GA skeleton. As a result, writing a GA-based

scheme first entails choosing how to address these principles. The first is the representation scheme and data structure of the population (P) of solutions being searched. In this case, the parameterized mechanical properties (more specifically, the Young's modulus values in our linear, isotropic formulation) of the model make up the search space, and thus their values become the GA individuals.

Each individual in the population has a corresponding fitness value, which quantifies how fit the individual is in comparison to other individuals. In other words, whether it is likely this individual will survive and reproduce. The fitness value is evaluated via an objective, or fitness, function chosen by the designer. Needless to say, choice of the objective function is both specific to the problem and very important as it is the search's driving force. For this elasticity parameter estimation problem, the objective was to minimize the difference between the measured and predicted displacement or strain responses, thus the natural choice of objective function was a simple residual norm function:

$$Objective_i = \tfrac{1}{2}\left\| \bar{e}_{predicted,i} - \bar{e}_{actual,i} \right\|^2, \tag{4.1}$$

where i represents the specific lumped parameter region and $\boldsymbol{e}$ is a general vector representing the values being compared in *that specific* region. It should be noted that when solving iterative inverse elasticity problems, using nodal displacement fields is most preferred because obtaining a *measured* strain field involves differentiating the already noisy *measured* displacements and thus amplifying the experimental noise. In this case, we experimented with the strain field, which was acceptable because, as is commonly done when prototyping, we simulated measured data by solving the forward elasticity problem with the target modulus field and adding white Gaussian noise when

desired. It should also be noted that the strain fields being compared in Equation (4.1) are effective strains, defined as

$$\bar{e} = \sqrt{\bar{e}_{xx}^2 + \bar{e}_{yy}^2 + \tfrac{1}{2}\bar{e}_{yz}^2} \tag{4.2}$$

With Equations (4.1) and (4.2), *each* GA individual is given a raw fitness value associated with *each distinct* parameter region. These raw values are then normalized by the number of elements in the region, so as to not over-compensate regions with a greater number of elements.

$$Fitness_{raw,i} = Objective_i \tag{4.3}$$

$$\tilde{f}_i = Normalized\ Fitness_{raw,i} = \frac{Fitness_{raw,i}}{elements_i} \tag{4.4}$$

After the f_i values are normalized yet again but with respect to the corresponding f_i values for all individuals in the population,

$$f_i = \frac{\tilde{f}_i}{\sum_{indivudiauls} \tilde{f}_{i,indivudual}}, \tag{4.5}$$

the *total fitness value* can be evaluated as a weighted sum of the raw fitness values,

$$F = Fitness_{total} = \sum_i \alpha_i f_i \tag{4.6}$$

We now have a better understanding of what n represents. Though officially it corresponds to the number of strain or displacement values available (generally, 2 × nodes or 3 × nodes, depending on the number of degrees of freedom), the outputted data is filtered through a series of transformations, including Equation (4.2) and, more importantly, a decoupling into specific lumped regions. Once the data is separated into its corresponding i^{th} grouping, it is then reassembled via Equation (4.6) into one total fitness value per GA individual.

The next step in constructing the GA was designing the appropriate genetic operations to take on the individuals. Three typical and fundamental operations are: (i) mating pool selection, (ii) crossover reproduction, and (iii) mutation. A "roulette wheel" selection process is often used for mating pool selection, where the likelihood an individual is chosen is a function of its fitness value. One can imagine constructing a pseudo "roulette wheel" (see Figure 4-4), where the percentages associated with each individual are the probabilities an individual will be chosen.

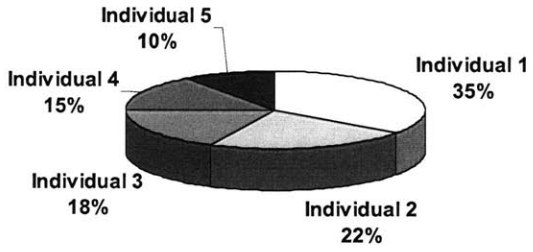


Figure 4-4. Cartoon of roulette wheel selection. An individual is selected into the mating pool according to a probability that is a function of its fitness.

The specific design adopted was an adaptation of this idea. Often fitness values can be quite sensitive to variations in the parameters [82]. Consequently, finding a function that translates an individual's fitness value to an appropriate survival probability is not always trivial. Instead, once the individuals are ranked according to their fitness (the lowest fitness value is given rank 1), a survival probability based on the rank is assigned. This "rank-survival curve" was defined as

$$Survival\ probability = \frac{\left((P+1)-rank\right)^{\gamma}}{\left(1^{\gamma}+2^{\gamma}+\ldots+P^{\gamma}\right)}, \tag{4.7}$$

where P is the population number and γ is a factor that adjusts the curve to favor the fitter individuals more or less. Using this curve, 50% of each generation is selected for the mating pool, in order to populate the subsequent generation ($P_{mp} = \frac{1}{2} P$ where P_{mp} is the mating pool population). The population sizes were allocated according to how expansive or reserved the search needs were (this is described in more detail during the presentation of the results).

From the mating pool, $P_{offspring}=P_{mp}$ offspring are created via a simple crossover function that swaps individuals' parameters in a partly random fashion. These offspring then join their parents to form the next generation of individuals to be evaluated. The algorithm continues this way, learning the search space, until it has located the fittest individual(s).

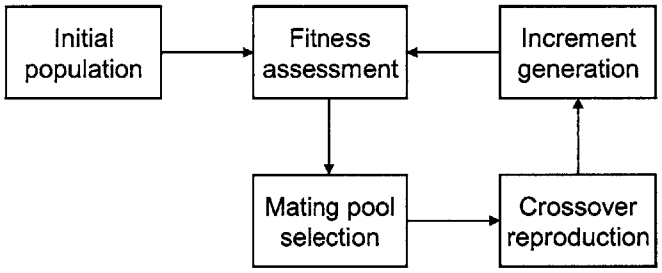


Figure 4-5. Flowchart of GA genetic operations.

The MATLAB source code for the FEM-GA algorithm and its corresponding functions is contained in Appendix A.4.

It should be noted that the genetic mutation operation was not explicitly implemented in this version, though it could prove helpful in locating additional accurate solutions for more rigorous versions. Instead, depending on the size of the initial population, diversity was inherently introduced at the start of each simulation, and thus we were inclined to favor relatively larger initial population sizes. A mutation function that randomly alters a

small percentage of each generation's population was not implemented for two main reasons. First, the prototype was intended to quickly generate results and evaluate the potential efficacy of this solution. The addition of a mutation function would require an extra function evaluation as well as an FEM execution that would slow the simulation. Second, without mutation, we could better handle and visualize the large solution space and the location of the global minimum. Adding an inherently random function, like mutation, would make evaluating how the algorithm traverses the solution space toward the global minimum time consuming.

4.5 Results

The FEM-GA system was applied to several models with different numbers of parameters. To efficiently characterize the algorithm, models representing idealized arteries and plaques were used. Specifically, the models represent two-dimensional arterial cross-sections, with inner radius, $r_i = 0.02$ units, and outer radius, $r_o = 0.05$ units. A pressure of $p = 13332.26$ units, consistent with a realistic arterial pressure of 100mmHg, was applied to the lumens. All the models satisfy the equilibrium equations for incompressible, linear elastic solids undergoing small, quasi-static deformations and were meshed with 9-node quadrilateral elements. Furthermore, as previously indicated, the measured displacement/strain data was generated by solving the forward problem with the target material field and white Gaussian noise was added in certain experiments.

4.5.1 One- and Two-Parameter Models

When applied to the one-parameter model exhibited in Figure 4-6, the algorithm found the value $E_{1converged}$ = 7.46e5 ($E_{1target}$ = 7.50e5, 0.513% error) after one iteration from an initial population of 40 individuals. The initial population was generated as a range of values from E_{min} = 1.00e5 to E_{max} = 1.00e6. Since random diversity was introduced only through the initial population, the choice of population size influences the accuracy of $E_{1converged}$. Thus, the fittest individual in the chosen population size of 40 had E = 7.46e5.

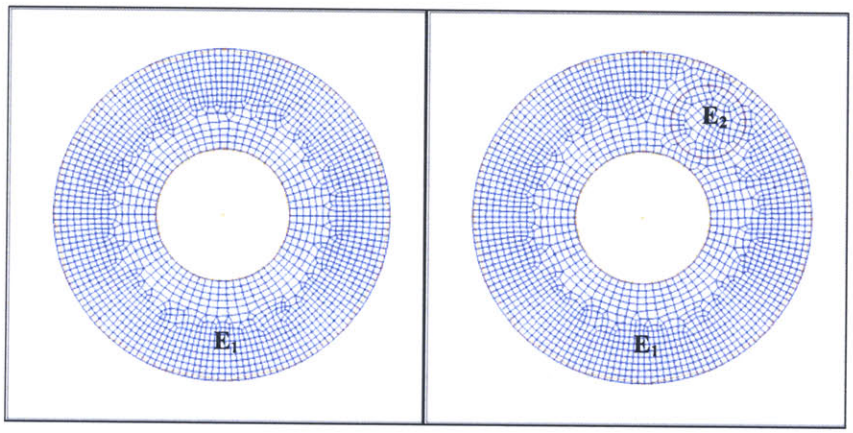


Figure 4-6. One- and two-parameter FE models. One-parameter model consisting of 1844 elements (left panel) and two-parameter model consisting of 1710 elements in E_1 region and 103 elements in E_2 region (right panel).

After confirming the validity of the algorithm on the straightforward one-parameter case, a small inclusion in the vascular model was inserted to introduce another parameter (Figure 4-6, right panel). Two simulations were run, where E_2 was varied to represent simplified versions of actual arterial inclusions. For instance, Simulation 1 corresponds to a hard calcified nodule surrounded by typical arterial fibrous plaque while the inclusion in Simulation 2 characterizes an extensible lipid pool. In each simulation, the algorithm was allowed only 5 iterations and an initial population of 50 individuals to locate the modulus values.

As Table 4-4 summarizes, the simulations were successful in finding the solution space's global minima. The slight errors in modulus values can be attributed to the limited population sizes and the restricted extent to which the algorithm was allowed to search. The reason for the relatively larger errors associated with smaller elastic modulus values (i.e., E_2 modulus of Simulation 2) is elucidated later in this thesis.

Simulation	$E_{1target}$	$E_{2target}$	$E_{1converged}$ (error)	$E_{2converged}$ (error)
1	6.00e5	3.00e7	6.14e5 (2.38 %)	3.02e7 (0.68 %)
2	6.00e5	3.00e3	5.96e5 (0.68 %)	3.39e3 (12.9 %)

Table 4-4. Simulation results for 2 different two-parameter models. Simulation 1 represents a simplified calcified inclusion and Simulation 2 represents a simplified lipid pool. Merely 5 iterations and an initial population of 50 were needed to accurately approximate the elasticity values.

The graphs, illustrated in Figure 4-7, chart the algorithm's search history for Simulation 1, showing only 5 iterations were needed to converge on the 2 correct parameter values. This behavior, where a very diverse initial population eventually develops into few individuals (or, hypothetically, one individual) after subsequent generations, is characteristic of successful genetic algorithms.

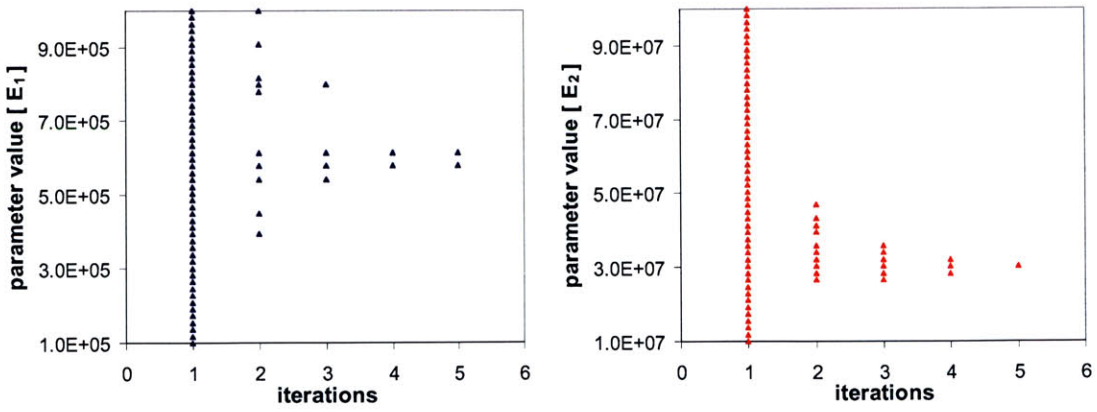


Figure 4-7. Search history for Simulation 1. With an initial population of 50 different individuals, 5 iterations were needed to narrow the search to 1 or 2 individuals. The algorithm then converged on values for E_1 (left panel) and E_2 (right panel).

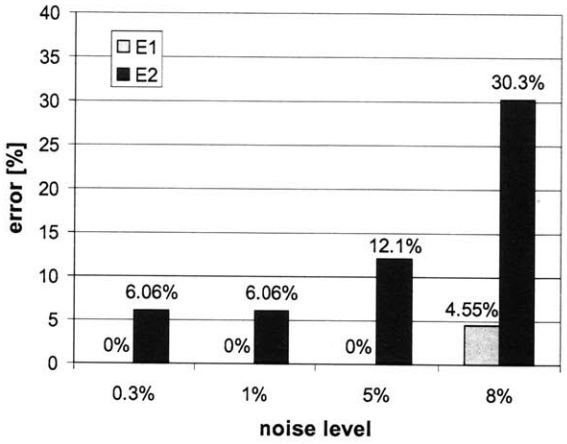


Figure 4-8. Effects of noise on the FEM-GA system. The plot shows errors in $E_{converged}$ values associated with adding 0.3%, 1%, 5%, and 8% white Gaussian noise to 'measured' displacement/strain data. The results indicate the system possesses positive, inherent smoothing abilities due to the lumping of parameters and that regions consisting of larger number of elements (i.e., E_1 region) are better able to handle noise.

We also investigated the algorithm's ability to handle noise in the 'measured' data. We predicted that the system, as a lumped parameter optimizer, would inherently possess built-in smoothing capabilities. Our results support this hypothesis. We added 0.3%, 1%, 5%, and 8% white Gaussian noise to the 'measured' data for a two-parameter, calcified model and ran the simulations with initial population sizes of 100 individuals (Figure 4-8). The results showed that, even when the data is riddled with noise levels of up to 8%, we can obtain fairly accurate and consistent lumped elasticity estimates, a very difficult task for calculus-based, discrete elasticity imaging. Not surprisingly, the errors associated with the E_1 region were consistently lower than those associated with the inclusion because it consisted of approximately 17-fold more elements than the inclusion and was therefore able to smooth the imposed noise to a greater extent. It is important to note, once again, that a small portion of error in all the results is attributed to the selected initial population sizes, making it virtually unfeasible to assign specific fault for observed errors in the results. Devoid of a mutation function, which would add function calls and slow

the process of characterizing the system, we rely on well-chosen initial population sizes to minimize these particular errors.

4.5.2 Three- and Four-Parameter Models

More complicated models incorporating 3 or 4 parameters (Figure 4-9) initially yielded mixed results. With the search space extended to an additional degree of freedom, the solution became more difficult to locate. This was primarily a result of two factors. First, the algorithm had to search yet another degree of freedom with the previous restrictions on the search and initial population size. Second, increasing the number of parameters produces more parameter-parameter interfaces, which potentially increases the dependency between elements. This convolutes the solution curve and can make the global minima a steeper, less forgiving trough.

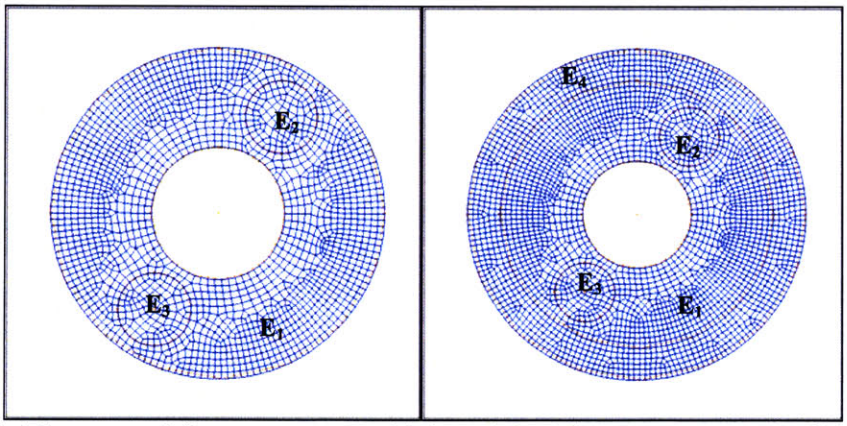


Figure 4-9. Three- and four-parameter FE models. Three-parameter model consisting of 1580, 103, and 102 elements in E_1, E_2, and E_3 regions (left panel) and four-parameter model consisting of 1580, 103, 102, and 1072 elements (right panel).

As Table 4-5 demonstrates for the m=3 and m=4 cases, solution sets for larger parameter optimization problems often exhibited one deviant parameter. This indicates the population size and iteration number restrictions placed on the algorithm restricted the

optimization of all parameters. A more expansive search is necessary to confidently locate better solutions.

	E_{target}	$E_{converged}$ (error) [m=3]	$E_{converged}$ (error) [m=4]
E_1	6.00e5	5.96e5 (0.68 %)	5.82e5 (3.83 %)
E_2	3.00e3	*3.94e3 (31.3 %)*	*6.18e3 (106 %)*
E_3	3.00e7	3.02e7 (0.68 %)	3.18e7 (6.06 %)
E_4	1.00e5	-	1.02e5 (1.82 %)

Table 4-5. Simulation results for three- and four-parameter models. Italicized values illustrate deviant convergence due to a restricted search, indicating the need for a more expansive search.

4.5.3 Expanding and Limiting the Search

To remedy the difficulties encountered for increased parameter problems, the algorithm was adjusted to expand its search and was regularized to limit its search based on physical observations.

We expanded the search in two ways. First, by increasing the population and relaxing the iteration restriction, the algorithm was compelled to search more comprehensively. Second, the "rank-survival curve," Equation (4.7), was used to dictate the extent of the search. Specifically, reducing the factor γ, especially in earlier iterations, relaxes the bias toward fitter individuals. This prevents the algorithm from selecting only the fittest individuals and converging too quickly, thus forcing a more thorough search. Table 4-6 reveals the outcome of the two search expanding techniques. The initial populations of an m=3 and m=4 model were enlarged to 100 individuals. In addition, for the m=4 model, γ was initiated at 0.5 and gradually increased to its typical value of 1.

	E_{target}	$E_{converged}$ (error) [m=3]	$E_{converged}$ (error) [m=4]
E_1	6.00e5	6.00e5 (0 %)	5.91e5 (1.52 %)
E_2	3.00e3	3.00e3 (0 %)	3.18e3 (6.06 %)
E_3	3.00e7	2.91e7 (3.03 %)	3.00e7 (0 %)
E_4	1.00e5	-	1.16e6 (16.4 %)

Table 4-6. Simulation results for three- and four-parameter models (expanded search). In the three-parameter model (m=3), the initial population was increased to 100 individuals and the number of iterations was consequently relaxed. In the four-parameter model (m=4), the initial population was also increased to 100 individual and the value of γ was dynamically increased from a low initial value, further enforcing a more thorough search.

Another technique for maintaining the robustness of this system when more parameters were introduced was intelligently limiting the search based on the physical problem. For these specific models, it was observed that the raw fitness values (f_i) for regions associated with greater parameter (elasticity) values tended to be independent of the other parameter values, while raw fitness values for regions associated with lower elasticity were dependent on the other parameter values. Physically, this occurs because distensible matter is less able to dissipate the energy associated with a pressure or displacement load than rigid matter. Therefore, the strains or displacements in these regions, such as a lipid pool, are highly dependent on the strains or displacements in the model's surrounding regions. As a consequence the algorithm had difficulty converging on modulus values for a lipid pool region when the search was not comprehensive enough.

We harnessed this observation to intelligently calculate individuals' total fitness (F). Rather than equally weighting the raw fitness values from each parameter region, the raw value associated with the stiffest region was emphasized over those of the other regions. As a result, the modulus value for this region was very accurately and quickly pinpointed

in the early part of the search. Subsequently, the remaining parameters, which depend on the accuracy of this solution, were located in following iterations.

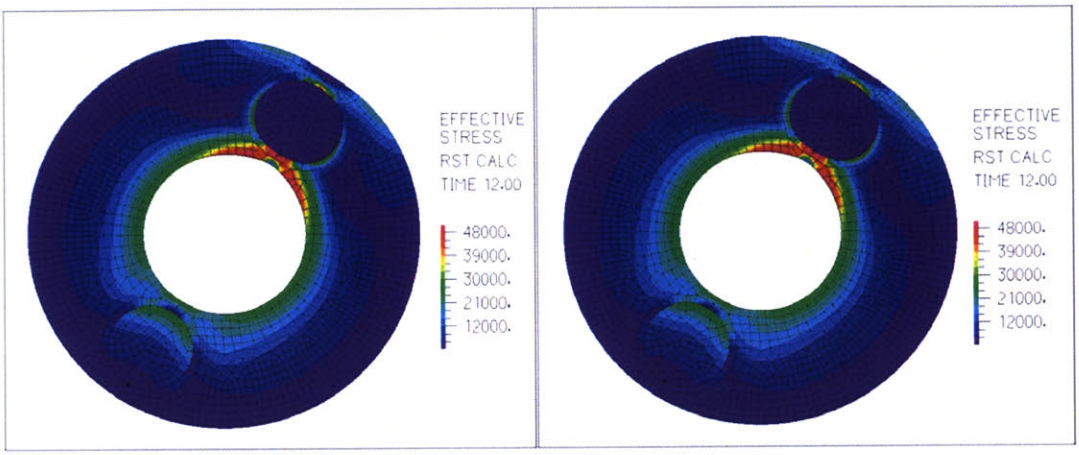


Figure 4-10. Stress distribution residual for three-parameter model. Stress distribution corresponding to target strain field (left panel) and corresponding to converged FEM-GA results from Table 3 (right panel). As expected, stress distributions predicted by FEM-GA system are consistent with the target stress field.

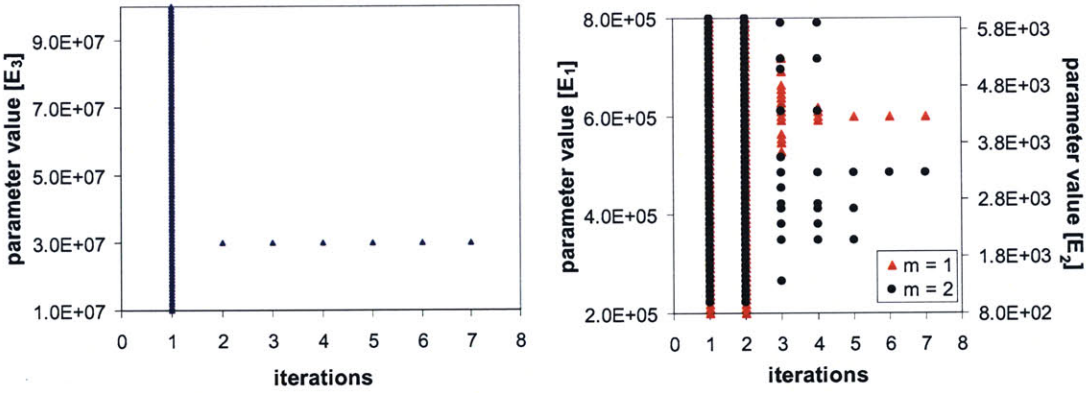


Figure 4-11. Search history for limited search simulation. The algorithm converged on E_3 first because its fitness value was essentially independent of the other parameter values. The remaining parameter values were located in subsequent iterations.

The results after 8 iterations of such a simulation are summarized in Table 4-7 (see also Figure 4-11). The elasticity value of parameter 3, corresponding to the stiffest region, was successfully recovered in 1 iteration, allowing the algorithm to focus on the more dependent parameters in the following iterations. As designed, the succession of

convergence follows the elasticity stiffness: $E_{3\ convergence} \ll E_{1\ convergence} < E_{2\ convergence}$ and $E_{3\ stiffness} \gg E_{1\ stiffness} > E_{2\ stiffness}$. In other words, the greater the elasticity value (i.e., the stiffer the region), the faster the algorithm converged on its modulus.

	E_{target}	$E_{converged}$ (error) [limited search]
E_1	6.00e5	6.00e5 (0 %)
E_2	3.00e3	3.27e3 (9.09 %)
E_3	3.00e7	3.00e7 (0 %)

Table 4-7. Simulation results for three-parameter model (limited search). The search was intelligently limited based on the observation that fitness values for stiffer regions were effectively independent of the elasticity values of other regions. The algorithm was forced to converge on E_3 first thus making it easier to locate the more dependent parameters in subsequent iterations.

The FEM-GA system demonstrated promise, potential, and versatility in preliminary arterial elasticity reconstruction, even in the presence of noise. Its success, however, is dependent on the ability to lump discrete elements of relatively similar value into several larger parameter regions. In the study of atherosclerotic plaques, distinguishing, for instance, a lipid pool or a calcified plaque with distinct boundaries is a common goal and byproduct of imaging and elastography experiments. If we harness this *a priori* knowledge, then we can take advantage of a search scheme more robust and intelligent than traditional gradient-based techniques, in order to generate lumped elasticity values or close initial guesses for traditional techniques.

Chapter 5

Summary & Conclusions

The current lack of knowledge of the pathology and atherogenesis that leads to the industrialized world's leading cause of morbidity and mortality—cardiovascular disease—is the overriding motivation behind the work presented in this thesis. As delineated in Chapter 1, atherogenesis is a complex disease promoted by many linked biochemical and biomechanical pathways. The need for an early and accurate means of lesion identification, especially in the context of potentially vulnerable plaques, cannot be stressed enough. In fact, the use of non-invasive treatment options depends highly on this effort.

The specific focus of this thesis was the study and exploitation of certain biomechanical precursors of acute, vulnerable atherosclerotic plaques. Currently, a lack of data on the mechanical behavior and properties of soft arterial tissue exists [3]. However, it is thought and gradually being confirmed that patient-specific maps of certain mechanical properties within arterial walls can supersede much other pathological expertise in predicting the harmful effects of plaque components on the structure and evolution of a plaque.

The main contribution of this thesis is within the specific enterprise of arterial elasticity reconstruction, which was outlined in Chapter 2. Generally termed model parameter estimation, this venture combines imaging, experimentation, and computational modeling to numerically predict a set of model parameters (i.e., discrete

values of Young's modulus) and complete a forward FEM problem. Imaging comes in many forms with all-ranging outcomes and advantages. Besides providing the arterial geometry necessary to build FE models, it is the scaffold with which elastography experiments are performed. Elastography, the second of the three main components of elasticity reconstruction, is the process of estimating tissue motion and subsequently tissue strain by imaging a specimen under an applied mechanical stimulus. Since no analogous means of providing internal stress distributions are currently available [40, 51], the elastography displacement or strain data is the only gateway into tissue elasticity. Furthermore, this means that for truly accurate elasticity reconstruction, devoid of unnecessary assumptions about the stress distribution, the problem must be formulated as an inverse problem.

The flowchart in Figure 2-2 is a picture of the inverse problem as an iterative undertaking. Although direct inversion is a solution possibility to the inverse elasticity equations, it has been shown to place limitations on the elasticity distribution and can yield unstable systems. As a result, Chapter 3 was dedicated to exploring the Gauss-Newton method as an iterative solution scheme to the inverse elasticity equations formulated as a NLS problem.

Despite both inherent and problem-specific limitations, our version of a linear perturbation Gauss-Newton method was mostly able to reconstruct homogeneous and inhomogeneous elasticity distributions. Faced with larger systems and the addition of both hard and spatially continuous inclusions, an unaided algorithm predictably faltered. Tikhonov regularization terms of assorted $\boldsymbol{L}$ matrices and weighting parameters were successfully included for stability and to encourage unique solutions. It was observed that

solutions do depend on choice of $\boldsymbol{L}$ and λ and consequently that *a priori* knowledge of the distribution is an indispensable asset for more complex models and initial guesses farther from the target solution.

When successful, the Gauss-Newton method provided an irreplaceable view of a specimen's local stiffness. Its reliance on regularization techniques, even for data lacking noise, and its potential computational intensity cannot be ignored though [43]. And when estimates of a specimen's modulus values are inadequate, a global convergence strategy may be beneficial. In Chapter 4 of this thesis, we explored an adaptive scheme, especially well-suited for robust searches.

If the model is segmented into a smaller number of lumped elasticity parameters (as desired by the user), then a combined FEM-GA system can be simply implemented to yield very accurate lumped elasticity values. The design of the algorithm, including the rules for genetic operations and how to interface with FEM, was presented, followed by results for more realistic, arterial models. Even in the presence of noisy data, the FEM-GA system was capable of optimizing the elasticity parameters accurately, in part, due to the inherent smoothing properties of lumping parameters. Advantages of this technique include very simple implementation, global convergence, and nonexistent ill-conditioning because of the reduced number of parameters and because its search is independent of a Jacobian or gradient. Additionally, because it does not depend on a numerical Jacobian, the GA is effectively blind to the formulation of the forward problem. Therefore, we expect future applications of this method with more complex material models and 3-D geometries to be straightforward. Disadvantages, of course, stem from the lack of precision associated with lumped elasticity values.

Future work in arterial elasticity reconstruction can be divided into two categories: (i) further characterizing and improving on the Gauss-Newton and FEM-GA systems that were built for this thesis and (ii) researching alternative and combinatory schemes.

Via collaborative work with Massachusetts General Hospital-based laboratories, images of and elastography data for real blood vessels are currently being generated using multiple imaging modalities. The two, most promising modalities are ultrasound and OCT—ultrasound because of its established reputation of generating elastography data and OCT because of its ability to differentiate plaque components. Once data is obtained, it will be used to test the developed technologies, in an effort to correlate stiffness patterns with pathology and to confirm current predictions of regions of vulnerability. In general, strives will be made to advance the technologies to accommodate more complex geometric and elasticity models. For example, a project is already underway to estimate material parameters for both 3-D models and nonlinear material properties using the FEM-GA system.

Alternatively, future work could be more development-minded. For example, although a functioning Gauss-Newton method was implemented, reducing its computational cost is eventually a necessary step when dealing with larger systems and more complex models. The algorithms appearing in Appendices A.1 and A.2 fall under the class of explicit Gauss-Newton methods because they require explicit representations of the entire Jacobian. Instead, implicit methods, such as Krylov subspace, are able to avoid computing and storing this large matrix by requiring only the matrix vector product of the Jacobian with an arbitrary vector, $\boldsymbol{v}$. In addition, a recently developed improvement of the linear perturbation Gauss-Newton method is the adjoint method [83], which

replaces the typical m+1 FEM function evaluations necessary to approximate the Jacobian with only 2 evaluations. Finally, in order to remedy some of the ill-conditioning and local convergence problems associated with the Gauss-Newton method it may be possible to merge the two technologies presented in this thesis, thereby equipping the Gauss-Newton method with a global convergence strategy that either delivers an accurate initial guess or serves as a pseudo regularization aid.

References

[1] American Heart Association, *Heart and Stroke Facts: 1996. Statistical Supplement*, American Heart Association, Dallas, TX, 1996.

[2] American Heart Association, *Heart Disease and Stroke Statistics–2003 Update*, American Heart Association, Dallas, TX, 2003.

[3] P.D. Richardson, "Biomechanics of plaque rupture: progress, problems, and new frontiers," *Ann Biomed Eng*, vol. 30, no. 4, pp. 524-36, 2002.

[4] R. Ross, "Atherosclerosis - an inflammatory disease," *N Engl J Med*, vol. 340, no. 2, pp. 115-26, 1999.

[5] D. Steinberg, "Low density lipoprotein oxidation and its pathobiological significance," *J Biol Chem*, vol. 272, no. 34, pp. 20963-6, 1997.

[6] J.C. Khoo, E. Miller, F. Pio, D. Steinberg, and J.L. Witztum, "Monoclonal antibodies against LDL further enhance macrophage uptake of LDL aggregates," *Arterioscler Thromb*, vol. 12, no. 11, pp. 1258-66, 1992.

[7] M. Navab, J.A. Berliner, A.D. Watson, S.Y. Hama, M.C. Territo, A.J. Lusis, D.M. Shih, B.J. Van Lenten, J.S. Frank, L.L. Demer, P.A. Edwards, and A.M. Fogelman, "The Yin and Yang of oxidation in the development of the fatty streak. A review based on the 1994 George Lyman Duff Memorial Lecture," *Arterioscler Thromb Vasc Biol*, vol. 16, no. 7, pp. 831-42, 1996.

[8] J. Han, D.P. Hajjar, M. Febbraio, and A.C. Nicholson, "Native and modified low density lipoproteins increase the functional expression of the macrophage class B

scavenger receptor, CD36," *J Biol Chem*, vol. 272, no. 34, pp. 21654-21659, 1997.

[9] M.T. Quinn, S. Parthasarathy, L.G. Fong, and D. Steinberg, "Oxidatively modified low density lipoproteins: a potential role in recruitment and retention of monocyte/macrophages during atherogenesis," *Proc Natl Acad Sci U S A*, vol. 84, no. 9, pp. 2995-8, 1987.

[10] D. Steinberg, "Atherogenesis in perspective: hypercholesterolemia and inflammation as partners in crime," *Nat Med*, vol. 8, no. 11, pp. 1211-7, 2002.

[11] A.I. Gotlieb and B.L. Langille, "The role of rheology in atherosclerotic coronary artery disease," in *Atherosclerosis and Coronary Artery Disease. Vol. 1*, V. Fuster, R. Ross, and E.J. Topol, Eds., pp. 595-606. Lippincott-Raven, Philadelphia, PA, 1996.

[12] T.A. Springer and M.I. Cybulsky, "Traffic signals on endothelium for leukocytes in health, inflammation, and atherosclerosis," in *Atherosclerosis and Coronary Artery Disease. Vol. 1*, V. Fuster, R. Ross, and E.J. Topol, Eds., pp. 511-538. Lippincott-Raven, Philadelphia, PA, 1996.

[13] W.A. Muller, S.A. Weigl, X. Deng, and D.M. Phillips, "PECAM-1 is required for transendothelial migration of leukocytes," *J Exp Med*, vol. 178, no. 2, pp. 449-60, 1993.

[14] P.K. Shah, E. Falk, J.J. Badimon, A. Fernando-Ortiz, A. Mailhac, J.T. Villareal-Levy, J.T. Fallon, J. Regnstrom, and V. Fuster, "Human monocyte-derived macrophages induce collagen breakdown in fibrous caps of atherosclerotic

plaques. Potential role of matrix-degrading metalloproteinases and implications for plaque rupture," *Circulation*, vol. 92, no. 6, pp. 1565-9, 1995.

[15] M.D. Rekhter, G.W. Hicks, D.W. Brammer, H. Hallak, E. Kindt, J. Chen, W.S. Rosebury, M.K. Anderson, P.J. Kuipers, and M.J. Ryan, "Hypercholesterolemia causes mechanical weakening of rabbit atheroma: local collagen loss as a prerequisite of plaque rupture," *Circ Res*, vol. 86, no. 1, pp. 101-8, 2000.

[16] M.J. Davies, P.D. Richardson, N. Woolf, D.R. Katz, and J. Mann, "Risk of thrombosis in human atherosclerotic plaques: role of extracellular lipid, macrophage, and smooth muscle cell content," *Br Heart J*, vol. 69, no. 5, pp. 377-81, 1993.

[17] E. Falk, P.K. Shah, and V. Fuster, "Coronary plaque disruption," *Circulation*, vol. 92, no. 3, pp. 657-671, 1995.

[18] P. Constantinides, "Plaque fissure in human coronary thrombosis," *J Atheroscler Res*, vol. 6, no. pp. 1-17, 1966.

[19] M.J. Davies and T. Thomas, "The pathological basis and microanatomy of occlusive thrombus formation in human coronary arteries," *Philos Trans R Soc London, Ser B*, vol. 294, no. pp. 225-229, 1981.

[20] S. Glagov, E. Weisenberg, C.K. Zarins, R. Stankunavicius, and G.J. Kolettis, "Compensatory enlargement of human atherosclerotic coronary arteries," *N Engl J Med*, vol. 316, no. 22, pp. 1371-5, 1987.

[21] S. Kiechl and J. Willeit, "The natural course of atherosclerosis. Part II: vascular remodeling. Bruneck Study Group," *Arterioscler Thromb Vasc Biol*, vol. 19, no. 6, pp. 1491-8, 1999.

[22] F.A. Duck, *Physical Properties of Tissues - A Comprehensive Reference Book*, Academic Press, Sheffield, United Kingdom, 1990.

[23] A. Sarvazyan, "Shear acoustic properties of soft biological tissues in medical diagnostics," in *Proc Acoust Soc Am*, Ottawa, Canada, 1993, pp. 2329.

[24] G.C. Cheng, H.M. Loree, R.D. Kamm, M.C. Fishbein, and R.T. Lee, "Distribution of circumferential stress in ruptured and stable atherosclerotic lesions. A structural analysis with histopathological correlation," *Circulation*, vol. 87, no. 4, pp. 1179-1187, 1993.

[25] P.D. Richardson, M.J. Davies, and G.V. Born, "Influence of plaque configuration and stress distribution on fissuring of coronary atherosclerotic plaques," *Lancet*, vol. 2, no. 8669, pp. 941-4, 1989.

[26] H.M. Loree, B.J. Tobias, L.J. Gibson, R.D. Kamm, D.M. Small, and R.T. Lee, "Mechanical properties of model atherosclerotic lesion lipid pools," *Arterioscler Thromb*, vol. 14, no. 2, pp. 230-4, 1994.

[27] H.M. Loree, R.D. Kamm, R.G. Stringfellow, and R.T. Lee, "Effects of fibrous cap thickness on peak circumferential stress in model atherosclerotic vessels," *Circ Res*, vol. 71, no. 4, pp. 850-8, 1992.

[28] S.D. Williamson, Y. Lam, H.F. Younis, H. Huang, S. Patel, M.R. Kaazempur-Mofrad, and R.D. Kamm, "On the sensitivity of wall stresses in diseased arteries to variable material properties," *J Biomech Eng*, vol. 125, no. 1, pp. 147-55, 2003.

[29] C.L. de Korte, A.F. van der Steen, E.I. Céspedes, G. Pasterkamp, S.G. Carlier, F. Mastik, A.H. Schoneveld, P.W. Serruys, and N. Bom, "Characterization of plaque

components and vulnerability with intravascular ultrasound elastography," *Phys Med Biol*, vol. 45, no. 6, pp. 1465-75, 2000.

[30] C.L. de Korte, A.F. van der Steen, E.I. Céspedes, and G. Pasterkamp, "Intravascular ultrasound elastography in human arteries: initial experience in vitro," *Ultrasound Med Biol*, vol. 24, no. 3, pp. 401-8, 1998.

[31] J.M. Schmitt, "OCT elastography: imaging microscopic deformation and strain of tissue," *Opt Express*, vol. 3, no. 6, pp. 199-211, 1998.

[32] G.J. Tearney, M.E. Brezinski, B.E. Bouma, S.A. Boppart, C. Pitris, J.F. Southern, and J.G. Fujimoto, "In vivo endoscopic optical biopsy with optical coherence tomography," *Science*, vol. 276, no. 5321, pp. 2037-2039, 1997.

[33] C. Yuan, J.S. Tsuruda, K.N. Beach, C.E. Hayes, M.S. Ferguson, C.E. Alpers, T.K. Foo, and D.E. Strandness, "Techniques for high-resolution MR imaging of atherosclerotic plaque," *J Magn Reson Imaging*, vol. 4, no. 1, pp. 43-9, 1994.

[34] J.F. Toussaint, J.F. Southern, V. Fuster, and H.L. Kantor, "T2-weighted contrast for NMR characterization of human atherosclerosis," *Arterioscler Thromb Vasc Biol*, vol. 15, no. 10, pp. 1533-42, 1995.

[35] J.F. Toussaint, G.M. LaMuraglia, J.F. Southern, V. Fuster, and H.L. Kantor, "Magnetic Resonance Images Lipid, Fibrous, Calcified, Hemorrhagic, and Thrombotic Components of Human Atherosclerosis In Vivo," *Circulation*, vol. 94, no. 5, pp. 932-938, 1996.

[36] M. Shinnar, J.T. Fallon, S. Wehrli, M. Levin, D. Dalmacy, Z.A. Fayad, J.J. Badimon, M. Harrington, E. Harrington, and V. Fuster, "The diagnostic accuracy

of ex vivo MRI for human atherosclerotic plaque characterization," *Arterioscler Thromb Vasc Biol*, vol. 19, no. 11, pp. 2756-61, 1999.

[37] M. Eliasziw, R.F. Smith, N. Singh, D.W. Holdsworth, A.J. Fox, and H.J. Barnett, "Further comments on the measurement of carotid stenosis from angiograms. North American Symptomatic Carotid Endarterectomy Trial (NASCET) Group," *Stroke*, vol. 25, no. 12, pp. 2445-9, 1994.

[38] "Endarterectomy for asymptomatic carotid artery stenosis. Executive Committee for the Asymptomatic Carotid Atherosclerosis Study," *Jama*, vol. 273, no. 18, pp. 1421-8, 1995.

[39] J.M. de Bray, J.M. Baud, and M. Dauzat, "Consensus concerning the morphology and the risk of carotid plaques," *Cerebrovasc Dis*, vol. 7, no. pp. 289-96, 1997.

[40] M.M. Doyley, P.M. Meaney, and J.C. Bamber, "Evaluation of an iterative reconstruction method for quantitative elastography," *Phys Med Biol*, vol. 45, no. 6, pp. 1521-40, 2000.

[41] J.F. Greenleaf, M. Fatemi, and M. Insana, "Selected methods for imaging elastic properties of biological tissues," *Annu Rev Biomed Eng*, vol. 5, no. pp. 57-78, 2003.

[42] M.J. Davies, *Atlas of Coronary Artery Disease*, Lippincott-Raven, Philadelphia, PA, 1998.

[43] J. Ophir, I. Céspedes, H. Ponnekanti, Y. Yazdi, and X. Li, "Elastography: a quantitative method for imaging the elasticity of biological tissues," *Ultrason Imaging*, vol. 13, no. 2, pp. 111-34, 1991.

[44] J. Ophir, I. Céspedes, B. Garra, H. Ponnekanti, Y. Huang, and N. Maklad, "Elastography: ultrasonic imaging of tissue strain and elastic modulus in vivo," *Eur. J. Ultrasound*, vol. 3, no. 1, pp. 49-70, 1996.

[45] I. Cespedes, J. Ophir, H. Ponnekanti, and N. Maklad, "Elastography: elasticity imaging using ultrasound with application to muscle and breast in vivo," *Ultrason Imaging*, vol. 15, no. 2, pp. 73-88, 1993.

[46] O.M.G.C. Op Den Camp, C.W.J. Oomens, F.E. Veldpaus, and J.D. Janssen, "An efficient algorithm to estimate material parameters of biphasic mixtures," *Int J Numer Meth Engng*, vol. 45, no. pp. 1315-1331, 1999.

[47] V.C. Mow, S.C. Kuei, W.M. Lai, and C.G. Armstrong, "Biphasic creep and stress relaxation of articular cartilage in compression? Theory and experiments," *J Biomech Eng*, vol. 102, no. 1, pp. 73-84, 1980.

[48] C.W. Oomens, D.H. van Campen, and H.J. Grootenboer, "A mixture approach to the mechanics of skin," *J Biomech*, vol. 20, no. 9, pp. 877-85, 1987.

[49] J.M.R. Huyghe, *Nonlinear finite element models of the beating left ventricle and the intramyocardial coronary circulation*, Doctoral dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands, 1986.

[50] J.M.A. Snijders, *The tri-phasic mechanics of the intervertebral disc - a theoretical, numerical and experimental analysis*, Doctoral dissertation, University of Limburg, Maastricht, The Netherlands, 1987.

[51] F. Kallel and M. Bertrand, "Tissue elasticity reconstruction using linear perturbation method," *IEEE Trans Med Imaging*, vol. 15, no. pp. 299-313, 1996.

[52] P.E. Barbone and J.C. Bamber, "Quantitative elasticity imaging: what can and cannot be inferred from strain images," *Phys Med Biol*, vol. 47, no. 12, pp. 2147-64, 2002.

[53] Y. Yamakoshi, J. Sato, and T. Satoa, "Ultrasonic imaging of internal vibration of soft tissue under forced vibration," *IEEE Trans Ultrason Ferroelectr Freq Control*, vol. 17, no. pp. 45-53, 1990.

[54] H. Ponnekanti, J. Ophir, and I. Céspedes, "Axial stress distributions between coaxial compressors in elastography: an analytical model," *Ultrasound Med Biol*, vol. 18, no. 8, pp. 667-73, 1992.

[55] H. Ponnekanti, J. Ophir, Y. Huang, and I. Céspedes, "Fundamental mechanical limitations on the visualization of elasticity contrast in elastography," *Ultrasound Med Biol*, vol. 21, no. 4, pp. 533-43, 1995.

[56] R.D.N. Ghosh, *Methods of Inverse Problems in Physics*, CRC, Boca Raton, FL, 1986.

[57] Z. Gao and T. Mura, "Nonelastic strains in solids - an inverse characterization from measured boundary data," *Int J Eng Sci*, vol. 30, no. 1, pp. 55-68, 1992.

[58] A. Zabras, N. Maniatty, and K. Stelson, "Finite element analysis of some inverse elasticity problems," *J Eng Mechanics*, vol. 115, no. 6, pp. 1303-1317, 1989.

[59] J.P. Laible, D. Pflaster, B.R. Simon, N.H. Krag, M. Pope, and L.D. Haugh, "A dynamic material parameter estimation procedure for tissue using a poroelastic finite element model," *J Biomed Eng*, vol. 116, no. pp. 19-29, 1994.

[60] A.R. Skovoroda, S.Y. Emelianov, and M. O'Donnell, "Tissue elasticity reconstruction based on ultrasonic displacement and strain images," *IEEE Trans Ultrason Ferroelectr Freq Control*, vol. 42, no. pp. 747-765, 1995.

[61] C. Sumi, A. Suzuki, and K. Nakayama, "Estimation of shear modulus distribution in soft tissue from strain distribution," *IEEE Trans Biomed Eng*, vol. 42, no. 2, pp. 193-202, 1995.

[62] K.R. Raghavan and A.E. Yagle, "Forward and inverse problems in elasticity imaging of soft-tissues," *IEEE Trans Nucl Sci*, vol. 41, no. pp. 1639-48, 1994.

[63] T.J. Yorkey, J.G. Webster, and W.J. Tompkins, "Comparing reconstruction algorithms for electrical impedance tomography," *IEEE Trans Biomed Eng*, vol. 34, no. 11, pp. 843-52, 1987.

[64] P.M. Meaney, *Microwave imaging for 2-D electrical property distribution profiling*, Doctoral dissertation, Thayer School of Engineering, Dartmouth College, Hanover, NH, 1995.

[65] J.E. Dennis, "Non-linear least squares and equations," in *The State of the Art in Numerical Analysis (Proceedings of the Conference on The State of the Art in Numerical Analysis held at The University of York)*, D. Jacobs, Ed., pp. 269-312. Academic Press, 1977.

[66] M.M. Doyley, J.C. Bamber, T. Shiina, and M.O. Leach, "Reconstruction of elasticity modulus distribution from envelope detected B-mode data," in *IEEE Int Ultrasonics Symp*, IEEE, San Antonio, TX, 1996, pp. 1611-14.

[67] C.T. Kelley, "Iterative methods for solving linear and nonlinear equations," in *Frontiers in Applied Mathematics*, SIAM, Philadelphia, PA, 1995.

[68] C.T. Kelley, *Solving Nonlinear Equations with Newton's Method (Fundamentals of Algorithms)*, SIAM, Philadelphia, PA, 2003.

[69] J. Hadamard, *Lectures on Cauchy's Problem in Linear Partial Differential Equations*, Yale University Press, New Haven, CT, 1923.

[70] P.C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, PA, 1998.

[71] C.W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Pitman, Boston, MA, 1984.

[72] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[73] D.L. Phillips, "A technique for the numerical solution of certain integral equations," *J Assoc Comput Mach*, vol. 9, no. pp. 84-97, 1962.

[74] A.N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method," *Soviet Math Dokl*, vol. 4, no. pp. 1035-1038, 1963.

[75] P.J. Mc Carthy, "Direct analytic model of the L-curve for Tikhonov regularization parameter selection," *Inverse Problems*, vol. 19, no. pp. 643-663, 2003.

[76] *ADINA: Automatic Dynamic Incremental Nonlinear Analysis*. 1986: Watertown, MA.

[77] J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

[78] J.R. Koza, *Genetic Programming: One the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, MA, 1992.

[79] G. Locatelli, H. Langer, M. Müller, and H. Baier, "Simultaneous optimization of actuator placement and structural parameters by mathematical and genetic optimization algorithms," in *Proc of IUTAM Conference Smart Structures and Structronic System*, Magdeburg, Germany, 2000.

[80] H. Yabushita, B.E. Bouma, S.L. Houser, H.T. Aretz, I.K. Jang, K.H. Schlendorf, C.R. Kauffman, M. Shishkov, D.H. Kang, E.F. Halpern, and G.J. Tearney, "Characterization of human atherosclerosis by optical coherence tomography," *Circulation*, vol. 106, no. 13, pp. 1640-5, 2002.

[81] H. Langer, T. Pühlhofer, and H. Baier, "An approach for shape and topology optimization integrating CAD parameterization and evolutionary algorithms," in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, 2002.

[82] L. Yao and W. Sethares, "Nonlinear parameter estimation via the genetic algorithm," *IEEE Trans Sig Proc*, vol. 42, no. 4, pp. 927-935, 1994.

[83] A.A. Oberai, N.H. Gokhale, and G.R. Feijóo, "Solution of inverse problems in elasticity using the adjoint method," *Inverse Problems*, vol. 19, no. pp. 297-313, 2003.

Appendix

A.1 Linear Perturbation Gauss-Newton Method: MATLAB

```
clear all;

%%%%%%%%%% USER-DEFINED PARAMETERS %%%%%%%%%%

newtonDir = 'C:\Documents and Settings\Ahmad Khalil\Desktop\newton_method\';
dataDir = 'C:\Documents and Settings\Ahmad Khalil\Desktop\data_from_alyx\';
cd( newtonDir );

file = 'model_3';

elems = 36;
nodesPerElem = 4;

initialGuess = 150;

% Regularization weighting factor
lambda = 1.0e-012;

%%%%%%%%%% RUN FEM OF INITIAL GUESS (FP) %%%%%%%%%%

Enew( 1:elems,1 ) = initialGuess

skeletonFilePrefix = file;
skeletonInFile = strcat( file,'.in');
fileNumber = num2str( 0 );
filePrefix = strcat( skeletonFilePrefix, fileNumber );
inFile  = strcat( filePrefix, '.in'  );
outFile = strcat( filePrefix, '.out' );

makeInFile( Enew,inFile,filePrefix,skeletonInFile,newtonDir );

% .in file -> .out file
dos(['"C:\Program Files\ADINA\ADINA System 8.1\bin\aui.exe" -b -m 100mb ',inFile]);
dos(['"C:\Program Files\ADINA\ADINA System 8.1\bin\adina.exe" -b -s -m 100mb ',filePrefix]);

% nodeDisp is an (n x 2) matrix = [ y1  z1  ;  y2  z2  ]
% elemNodeMat is an (m x nodesPerElem) matrix = [ node1   node2   node3   node4 ; ... ]
[ nodeDisp,elemNodeMat,nodePos ] = readOutFileNodalFull( outFile,nodesPerElem );
elems = size( elemNodeMat,1 );
nodes = size( nodeDisp,1 );

% Construct connectivity matrix (elemIncidenceMat)
elemIncidenceMat( 1:elems,1:elems ) = zeros;
for i = 1:elems
   for j = 1:nodesPerElem
      for k = 1:elems
         for m = 1:nodesPerElem
```

```
            if elemNodeMat( k,m ) == elemNodeMat( i,j );
               elemIncidenceMat( i,k ) = 1;
            end
         end
      end
   end
end
clear elemNodeMat;

%%%%%%%%%% INITIALIZE VECTORS & MATRICES %%%%%%%%%%

% effNodeDisp is a (2n x 1) vector
effNodeDisp( 1:2*nodes,1 ) = zeros;
for i = 1:1:nodes
   effNodeDisp( i*2-1,1 ) = nodeDisp( i,1 );
   effNodeDisp( i*2,1 ) = nodeDisp( i,2 );
end

deleteAllFiles( filePrefix );

% Initialize Jg (2n x m)
clear Jg; clear Jf;
Jg( 1:2*nodes,1:elems ) = zeros;

% Construct Tikhonov L matrix (m x m)
L( 1:elems,1:elems ) = zeros;
L = diag( -ones( elems,1 ),0 );
for i = 1:elems
   connect = 0;
   for j = 1:elems
      if j ~= i
         if elemIncidenceMat( i,j ) == 1
            connect = connect + 1;
            L( i,j ) = 1;
         end
      end
   end
   L( i,i ) = L( i,i ) * connect;
end

%%%%%%%%%% RUN TARGET .in FILE %%%%%%%%%%

% .in file -> .out file
dos(['"C:\Program Files\ADINA\ADINA System 8.1\bin\aui.exe" -b -m 100mb ',skeletonInFile]);
dos(['"C:\Program Files\ADINA\ADINA System 8.1\bin\adina.exe" -b -s -m 100mb ',skeletonFilePrefix]);
outFile = strcat( skeletonFilePrefix,'.out' );

[ actNodeDisp ] = readOutFileNodal( outFile );

% effectNodeDisp is (2n x 1) vector of 'experimental displacement data'
effActNodeDisp( 1:2*nodes,1 ) = zeros;
for i = 1:1:nodes
   effActNodeDisp( i*2-1,1 ) = actNodeDisp( i,1 );
   effActNodeDisp( i*2,1 ) = actNodeDisp( i,2 );
end
```

```
porFile = strcat( skeletonFilePrefix,'.por' ); dos( ['del ',porFile] );
datFile = strcat( skeletonFilePrefix,'.dat' ); dos( ['del ',datFile] );
resFile = strcat( skeletonFilePrefix,'.res' ); dos( ['del ',resFile] );
modFile = strcat( skeletonFilePrefix,'.mod' ); dos( ['del ',modFile] );
dos( ['del ',outFile] );

%%%%%%%%%% ITERATE %%%%%%%%%%

% 1e-3 > eps > 1e-4 (eps ~ sqrt( error of function evaluation ))
eps = 5e-4;
loop = 1;
iters = 1;
Ecriteria = 5e-2;
fcriteria = 1e-6;
clear E;
maxIters = 20;

while iters < maxIters

   clear Eold; clear Jg; clear Jf;

   Eold = Enew;

   % Negative E values not allowed -> set to initial guess
   for i = 1:elems
      if Eold( i,1 ) < 0
         Eold( i,1 ) = initialGuess;
      end
   end

   % Run FEM of iterate update
   if iters ~= 1
      fileNumber = num2str( 0 );
      filePrefix = strcat( skeletonFilePrefix, fileNumber );
      inFile  = strcat( filePrefix, '.in'  );
      outFile = strcat( filePrefix, '.out' );

      makeInFile( Eold,inFile,filePrefix,skeletonInFile,newtonDir );

      % .in file -> .out file
      dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\aui.exe" -b -m 100mb ', inFile] );
      dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\adina.exe" -b -s -m 100mb ', filePrefix] );

      [ nodeDisp ] = readOutFileNodal( outFile );

      effNodeDisp( 1:2*nodes,1 ) = zeros;
      for i = 1:1:nodes
         effNodeDisp( i*2-1,1 ) = nodeDisp( i,1 );
         effNodeDisp( i*2,1 ) = nodeDisp( i,2 );
      end

      deleteAllFiles( filePrefix );

   end

   % Perturb discrete E values and run perturbed distributions
```

```
    for i = 1:elems

        fileNumber = num2str( i );
        filePrefix = strcat( skeletonFilePrefix, fileNumber );
        inFile  = strcat( filePrefix, '.in'  );
        outFile = strcat( filePrefix, '.out' );
        porFile = strcat( filePrefix, '.por' );
        datFile = strcat( filePrefix, '.dat' );
        resFile = strcat( filePrefix, '.res' );
        modFile = strcat( filePrefix, '.mod' );

        clear Ep;
        Ep = Eold;
        Ep( i,1 ) = Eold( i,1 ) * eps + Eold( i,1 );
        makeInFile( Ep, inFile, filePrefix, skeletonInFile, newtonDir );

        % .in file -> .out file
        dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\aui.exe" -b -m 100mb ', inFile] );
        dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\adina.exe" -b -s -m 100mb ', filePrefix] );

        [ nodeDispPT ] = readOutFileNodal( outFile );

        deleteAllFiles( filePrefix );

        effNodeDispPT( 1:2*nodes,1 ) = zeros;
        for j = 1:1:nodes
           effNodeDispPT( j*2-1,1 ) = nodeDispPT( j,1 );
           effNodeDispPT( j*2,1 ) = nodeDispPT( j,2 );
        end

        % Column by column approximation for Jacobian (Jg)
        Jg( :,i ) = ( effNodeDispPT - effNodeDisp ) / ( Eold( i,1 ) * eps );

    end

    % Jf = Jg' * Jg + Tikhonov regularization term
    Jf = Jg'*Jg + lambda*L(:,:,1)'*L(:,:,1);

    f = Jg' * ( effNodeDisp - effActNodeDisp ) + lambda*L(:,:,1)'*L(:,:,1)*Eold;

    % Newton update equation for new search direction
    Enew = -Jf \ f + Eold

    E( :,iters ) = Enew;

    % Convergence criteria
    if ( ( norm( Enew-Eold ) ) / elems < Ecriteria ) & ( ( norm( f )/elems ) < fcriteria )
       break
    end

    iters = iters + 1
end
```

```
% The function 'readOutFileNodalFull' parses the FEM .out file and returns:

%  1. disp: (n x 2) matrix of model's nodal displacements in y- and z-directions
%  2. elemNodeMat: (m x nodesPerElem) matrix of the nodes associated with each element
%  3. nodePos: (n x 2) matrix of the y- and z-positions for each node

function [ disp,elemNodeMat,nodePos ] = readOutFileNodalFull( filename,nodesPerElem );

% Open the file. If this returns a -1, the file was not opened successfully
fid = fopen( filename );
if fid == -1
  error( 'File not found or permission denied' );
end

iterate = 0;
while( iterate == 0 )
   buffer = fgetl( fid );
   iterate = strncmp( buffer, ' CARD NUMBER 1', 14 );
end

buffer = fgetl( fid );
buffer = fgetl( fid );

inLineTokens = 16;
for i = 1:inLineTokens
   [ toss, buffer ] = strtok( buffer );
end
nodes = str2num( toss );

iterate = 0;
while( iterate ==0 )
   buffer = fgetl( fid );
   iterate = strncmp( buffer, '  GENERATED NODAL DATA',22 );
end
nodeHeader = 7;
for i = 1:nodeHeader-1
   buffer = fgetl( fid );
end

nodePos( 1:nodes,1:2 ) = zeros;
inLineTokens = 16;
for i = 1:nodes
   buffer = fgetl( fid );
   buffer = fgetl( fid );
   for j = 1:inLineTokens
      [ toss,buffer ] = strtok( buffer );
   end
   [ Ypos,buffer ] = strtok( buffer );
   [ Zpos,buffer ] = strtok( buffer );
   nodePos( i,1 ) = str2num( Ypos );
   nodePos( i,2 ) = str2num( Zpos );
end

iterate = 0;
while( iterate == 0 )
   buffer = fgetl( fid );
```

```
   iterate = strncmp( buffer, ' NUMBER OF ELEMENTS',19 );
end

inLineTokens = 18;
for i = 1:inLineTokens
   [ toss, buffer ] = strtok( buffer );
end
elems = str2num( toss );

iterate = 0;
while( iterate == 0 )
   buffer = fgetl( fid );
   iterate = strncmp( buffer, ' E L E M E N T   I N F O R M A T I O N', 39 );
end
elemHeader = 6;
for i = 1:elemHeader-1
   buffer = fgetl( fid );
end
for i = 1:elems
   buffer = fgetl( fid );
   buffer = fgetl( fid );
   inLineTokens = 16;
   for j = 1:inLineTokens-5
      [ toss, buffer ] = strtok( buffer );
   end
   for j = 1:nodesPerElem
      [ toss, buffer ] = strtok( buffer );
      elemNodeMat( i,j ) = str2num( toss );
   end
end

iterate = 0;
while( iterate == 0 )
   buffer = fgetl( fid );
   iterate = strncmp( buffer, ' D I S P L A C E M E N T S',27 );
end
dispHeader = 7;
for i = 1:dispHeader
   buffer = fgetl( fid );
end

for i = 1:nodes
  buffer = fgetl( fid );
  [ nodeCount,buffer ] = strtok( buffer );
  while isempty( nodeCount ) == 1 | isempty( str2num( nodeCount ) ) == 1 | str2num( nodeCount ) ~= i
     buffer = fgetl( fid );
     [ nodeCount,buffer ] = strtok( buffer );
  end
   [ toss,buffer ] = strtok( buffer );
   [ Ytrans,buffer ] = strtok( buffer );
   [ Ztrans,buffer ] = strtok( buffer );
   disp( i,1 ) = str2num( Ytrans );
   disp( i,2 ) = str2num( Ztrans );
end

fclose( fid );
```

```
% The function 'readOutFileNodal' is just a shortened version of 'readOutFileNodalFull'
% It returns only the disp (n x 2) matrix of model's nodal displacements

function [ disp ] = readOutFileNodal( filename );

% Open the file. If this returns a -1, the file was not opened successfully
fid = fopen( filename );
if fid == -1
  error( 'File not found or permission denied' );
end

iterate = 0;
while( iterate == 0 )
   buffer = fgetl( fid );
   iterate = strncmp( buffer, ' CARD NUMBER 1', 14 );
end

buffer = fgetl( fid );
buffer = fgetl( fid );

inLineTokens = 16;
for i = 1:inLineTokens
   [ toss, buffer ] = strtok( buffer );
end
nodes = str2num( toss );

iterate = 0;
while( iterate == 0 )
   buffer = fgetl( fid );
   iterate = strncmp( buffer, ' D I S P L A C E M E N T S',27 );
end
dispHeader = 7;
for i = 1:dispHeader
   buffer = fgetl( fid );
end

for i = 1:nodes
  buffer = fgetl( fid );
  [ nodeCount,buffer ] = strtok( buffer );

  while isempty( nodeCount ) == 1 | isempty( str2num( nodeCount ) ) == 1 | str2num( nodeCount ) ~= i
     buffer = fgetl( fid );
     [ nodeCount,buffer ] = strtok( buffer );
  end
   [ toss,buffer ] = strtok( buffer );
   [ Ytrans,buffer ] = strtok( buffer );
   [ Ztrans,buffer ] = strtok( buffer );
   disp( i,1 ) = str2num( Ytrans );
   disp( i,2 ) = str2num( Ztrans );
end

fclose( fid );
```

```
% The function 'makeInFile' uses a skeleton .in file ('skeletonInFile') to write a new .in file that
% builds the same model but with the given Young's modulus distribution vector (E)

function makeInFile( E, inFile, filePrefix, skeletonInFile, dir )

% Open the files. If this returns a -1, the file was not opeened successfully.
fidR = fopen( skeletonInFile, 'r' );
if fidR == -1
  error( 'File not found or permission denied' );
end

fidW = fopen( inFile, 'w' );
if fidW == -1
  error( 'File not found or permission denied' );
end

iterate = 0;
lineNum1 = 0;

while ( iterate == 0 )
   buffer = fgetl( fidR );
   iterate = strncmp( buffer, 'MATERIAL ELASTIC', 16);
   lineNum1 = lineNum1 + 1;
end

frewind( fidR );

for i = 1 : ( lineNum1 - 1 )
   buffer = fgetl( fidR );
   fprintf( fidW, '%s\n', buffer );
end

n = size( E,1 );

for i = 1 : n

   % Get the 'Material Elastic Name...' line from the skeleton .in file
   buffer = fgetl( fidR );
   % Get the 'Density...Alpha' line from the skeleton .in file
   buffer = fgetl( fidR );
   EndCommandBuffer = fgetl( fidR );

end

for i = 1 : n
   fprintf( fidW, 'MATERIAL ELASTIC NAME=%g E=%14.14e NU=0.499000000000000,\n', i, E( i ) );
   fprintf( fidW, '%s\n', buffer );
   fprintf( fidW, '%s\n', EndCommandBuffer );
end

iterate = 0;

while ( iterate == 0 )
   buffer = fgetl( fidR );
   fprintf( fidW, '%s\n', buffer );
   iterate = strncmp( buffer, 'ADINA OPTIMIZE=SOLVER FILE=,', 28)  ;
```

```
end

buffer = fgetl( fidR );
fprintf( fidW, '"%s%s.dat",\n',dir,filePrefix );

buffer = fgetl( fidR );
while ( buffer ~= ( -1 ) )
   fprintf( fidW, '%s\n', buffer );
   buffer = fgetl( fidR );
end

fclose( fidR );
fclose( fidW );
```

% The function 'deleteAllFiles' deletes all the superfluous ADINA FEM files

```
function deleteAllFiles( filePrefix )

porFile = strcat( filePrefix, '.por' ); dos ( ['del ',porFile] );
datFile = strcat( filePrefix, '.dat' ); dos ( ['del ',datFile] );
resFile = strcat( filePrefix, '.res' ); dos ( ['del ',resFile] );
modFile = strcat( filePrefix, '.mod' ); dos ( ['del ',modFile] );
outFile = strcat( filePrefix, '.out' ); dos ( ['del ',outFile] );
inFile = strcat( filePrefix, '.in' ); dos ( ['del ',inFile] );
```

Sample ADINA .in file of 2-D square-shaped finite element model with 16 elements

```
*
FEPROGRAM PROGRAM=ADINA
*
CONTROL PLOTUNIT=PERCENT VERBOSE=YES ERRORLIM=0 LOGLIMIT=0 UNDO=5,
     PROMPTDE=UNKNOWN AUTOREPA=YES DRAWMATT=YES DRAWTEXT=EXACT,
     DRAWLINE=EXACT DRAWFILL=EXACT AUTOMREB=YES ZONECOPY=NO,
     SWEEPCOI=YES SESSIONS=YES DYNAMICT=YES UPDATETH=YES AUTOREGE=NO,
     ERRORACT=CONTINUE FILEVERS=V81 INITFCHE=NO SIGDIGIT=16,
     AUTOZONE=YES
*
COORDINATES POINT SYSTEM=0
@CLEAR
1 0.00000000000000 0.00000000000000 0.00000000000000 0
2 0.00000000000000 1.00000000000000 0.00000000000000 0
3 0.00000000000000 1.00000000000000 1.00000000000000 0
4 0.00000000000000 0.00000000000000 1.00000000000000 0
@
*
COORDINATES POINT SYSTEM=0
@CLEAR
1 0.00000000000000 0.00000000000000 0.00000000000000 0
2 0.00000000000000 1.00000000000000 0.00000000000000 0
3 0.00000000000000 1.00000000000000 1.00000000000000 0
4 0.00000000000000 0.00000000000000 1.00000000000000 0
@
*
LINE STRAIGHT NAME=1 P1=1 P2=2
*
LINE STRAIGHT NAME=2 P1=2 P2=3
*
LINE STRAIGHT NAME=3 P1=3 P2=4
*
LINE STRAIGHT NAME=4 P1=4 P2=1
*
LINE COMBINED NAME=5 COUPLED=YES RESTRICT=YES
@CLEAR
1
2
3
4
@
*
BODY SHEET NAME=1 LINE=5 DELETE-L=NO
@CLEAR
@
*
SUBDIVIDE EDGE NAME=1 BODY=1 MODE=DIVISIONS NDIV=4,
     RATIO=1.00000000000000 PROGRESS=GEOMETRIC
@CLEAR
2
3
4
@
*
MATERIAL ELASTIC NAME=1 E=100.00000000000000 NU=0.499000000000000,
```

```
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=2 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=3 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=4 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=5 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=6 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=7 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=8 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=9 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=10 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=11 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=12 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=13 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=14 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=15 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
MATERIAL ELASTIC NAME=16 E=100.00000000000000 NU=0.499000000000000,
     DENSITY=0.00000000000000 ALPHA=0.00000000000000 MDESCRIP='NONE'
*
EGROUP TWODSOLID NAME=1 SUBTYPE=STRAIN DISPLACE=DEFAULT,
     STRAINS=DEFAULT MATERIAL=1 INT=DEFAULT RESULTS=FORCES DEGEN=NO,
     FORMULAT=2 STRESSRE=GLOBAL INITIALS=NONE FRACTUR=NO,
     CMASS=DEFAULT STRAIN-F=0 UL-FORMU=DEFAULT PNTGPS=0 NODGPS=0,
     LVUS1=0 LVUS2=0 SED=NO RUPTURE=ADINA INCOMPAT=DEFAULT,
     TIME-OFF=0.00000000000000 POROUS=NO WTMC=1.00000000000000,
     OPTION=NONE DESCRIPT='NONE'
*
GFACE NODES=4 NCOINCID=BOUNDARIES NCTOLERA=1.00000000000000E-05,
     SUBSTRUC=0 GROUP=1 PREFSHAP=QUAD-DIRECT BODY=1 COLLAPSE=NO,
```

```
     SIZE-FUN=0 MIDNODES=CURVED METHOD=DELAUNAY NLAYER=1 NLTABL=0
@CLEAR
1
@
*
GFACE NODES=4 NCOINCID=BOUNDARIES NCTOLERA=1.00000000000000E-05,
     SUBSTRUC=0 GROUP=1 PREFSHAP=QUAD-DIRECT BODY=1 COLLAPSE=NO,
     SIZE-FUN=0 MIDNODES=CURVED METHOD=DELAUNAY NLAYER=1 NLTABL=0
@CLEAR
@
*
EDATA SUBSTRUC=0 GROUP=1
@STARTMODIFY
@CHAROW          1       101       101
1 1 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
2 2 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
3 3 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
4 4 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
5 5 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
6 6 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
7 7 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
8 8 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
9 9 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
10 10 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
11 11 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
12 12 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
13 13 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
14 14 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
15 15 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
16 16 0.00000000000000 0.00000000000000  'DEFAULT'  'DEFAULT',
     0.00000000000000 0.00000000000000 0 0.00000000000000 0
@ENDMODIFY
*
EDATA SUBSTRUC=0 GROUP=1
@STARTMODIFY
@ENDMODIFY
*
BOUNDARIES SUBSTRUC=0
@CLEAR
1  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  ,
'FIXED'  'FIXED'  'FIXED'
2  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  ,
```

```
'FIXED'  'FIXED'  'FIXED'
3  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  ,
'FIXED'  'FIXED'  'FIXED'
4  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  ,
'FIXED'  'FIXED'  'FIXED'
5  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  'FIXED'  ,
'FIXED'  'FIXED'  'FIXED'
@
*
TIMEFUNCTION NAME=1 IFLIB=1 FPAR1=0.00000000000000,
     FPAR2=0.00000000000000 FPAR3=0.00000000000000,
     FPAR4=0.00000000000000 FPAR5=0.00000000000000,
     FPAR6=0.00000000000000
@CLEAR
0.00000000000000 0.00000000000000
6.00000000000000 1.0000000000000
@
*
LOAD FORCE NAME=1 MAGNITUD=-1.00000000000000 FX=0.00000000000000,
     FY=1.00000000000000 FZ=0.00000000000000
*
APPLY-LOAD BODY=1
@CLEAR
1  'FORCE' 1  'EDGE' 2 0 1 0.00000000000000 0 -1 0 1 0  'NO',
     0.00000000000000 0.00000000000000 1 0
@
*
TIMESTEP NAME=DEFAULT
@CLEAR
6 1.00000000000000
@
*
PRINT-STEPS SUBSTRUC=0 REUSE=1
@CLEAR
1 6 6 1
@
*
PRINTNODES FACES SUBSTRUC=0 REUSE=1 BODY=1
@CLEAR
1
@
*
PRINTOUT ECHO=NO PRINTDEF=STRAINS INPUT-DA=0 OUTPUT=SELECTED,
     DISPLACE=YES VELOCITI=NO ACCELERA=NO IDISP=NO ITEMP=NO,
     ISTRAIN=NO IPIPE=NO STORAGE=NO LARGE-ST=NONE
*
ADINA OPTIMIZE=SOLVER FILE=,
'C:\Documents and Settings\Ahmad Khalil\Desktop\model_1.dat',
     FIXBOUND=YES MIDNODE=NO OVERWRIT=YES
*
```

A.2 Linear Perturbation Gauss-Newton Method: C Code

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "matrix_vector.h"

#define FALSE 0
#define TRUE  1

// Declare directory names and skeleton file name //
static char newtonDir[] = "C:\\Documents and Settings\\Ahmad S Khalil\\My
Documents\\MIT\\parameter_estimation\\lin_pert_newton\\";
static char dataDir[] = "C:\\Documents and Settings\\Ahmad S Khalil\\Desktop\\data_from_alyx\\";
static char skeletonFile[] = "square";
static char dotIn[]  = ".in";
static char dotOut[] = ".out";
static char dotRes[] = ".res";
static char dotPor[] = ".por";
static char skeletonInFile[50], inFile[50], outFile[50], resFile[50], porFile[50], filePrefix[50], fileNo[10];

// Declare static variables //
static double *nodalDisp, *nodalDispPert, *actNodalDisp, *Enew, *Eold, *Epert, *f, *dispResidual,
*EResidual;
static double **Jg, **Jg_t, **Jf, **delE, **nodePos;
static int **elemNodeMat;
static int iters = 1;
static int perturb, loop, runActData;

// Inputs & initial guess //
static int nodesPerElem = 4;
static int elems = 36;
static int nodes = 49;
static int eps = 5e-4;
static int initGuess = ;
static int degOfFree = 2;
static int nodesPerElem = 4;

// Declare functions //
void allocateMemory();
void forwardProblem();
void nameFiles();
void makeInFile();
void runInFile();
void readOutFileFull();
void readOutFile();
void deleteFiles();
void gaussj(double **a, int n, double **b, int m);

int main()
{
```

```
int i,j;
double Echeck, fcheck;

loop = FALSE;
perturb = FALSE;
strcat( skeletonInFile,skeletonFile );
strcat( skeletonInFile,dotIn );

allocateMemory();

// Vectorize initial guess of E //
for (i=1;i<elems+1;i++) {
        Enew[i] = initGuess;
}

itoa(0,fileNo,10);
forwardProblem();


loop = TRUE;

////////////////////////////////////
//    Actual Data        //
////////////////////////////////////
runActData = TRUE;
fileNo = "actual";
forwardProblem();
runActData = FALSE;
////////////////////////////////////

// Loop //
while (loop == TRUE) {

        // Eold = Enew //
        for (i=1;i<elems+1;i++) {
                Eold[i] = Enew[i];
        }

        if (iters>1) {
                perturb = FALSE;
                itoa(0,fileNo,10);
                forwardProblem();
        }

        // Perturb values of E one at a time and then run //
        for (i=1;i<elems+1;i++) {
                Epert[i] = Eold[i];
        }
        perturb = TRUE;
        for (i=1;i<elems+1;i++) {

                if (i>1) {
                        Epert[i-1] = Eold[i-1];
                }

                Epert[i] = Eold[i] * eps + Eold[i];
```

```
                    itoa(i,fileNo,10);
                    forwardProblem();

                    // Form Jg matrix columns from displacements of perturbed E values //
                    for (j=1;j<nodes+1;j++) {
                        Jg[j][i] = ( nodalDispPert[j] - nodalDisp[j] ) / ( eps * Eold[i] );
                    }
                }

                Jg_t = transpose(Jg_t,Jg,nodes,elems);

                // Jf = Jg_t * Jg //
                Jf = matrix_matrix_mult(Jf,Jg_t,Jg,elems,nodes);

                for (i=1;i<nodes+1;i++) {
                    dispResidual[i] = nodalDisp[i] - actNodalDisp[i];
                }

                // f = Jg_t * (nodalDisp - actNodalDisp) //
                f = matrix_vector_mult(f,Jg_t,dispResidual,elems,nodes);

                for(i=1;i<elems+1;i++) {
                    delE[i][1] = f[i];
                }

                // Invert Jf and solve Jf \ f //
                gaussj( Jf,elems,delE,1 );

                // Update E distribution in delE direction //
                for(i=1;i<elems+1;i++) {
                    Enew[i] = -delE[i][1] + Eold[i];
                }

                // Check convergence criteria //
                for (i=1;i<elems+1;i++) {
                    EResidual[i] = Enew[i]-Eold[i];
                }
                Echeck = vector_norm( EResidual,elems );
                fcheck = vector_norm( f,elems );
                if ((Echeck/elems < Ecriteria) && (fcheck/elems < fcriteria)) {
                    loop = FALSE;
                    break;
                }

                // Increment //
                iters = iters + 1;
        }

        return 0;
}

void forwardProblem() {

        nameFiles();
```

```
        if (runActData = FALSE) {
                makeInFile();
        }

        runInFile();

        if (loop == FALSE) {
                readOutFileFull();
        } else {
                readOutFile();
        }

        deleteFiles();
}

void allocateMemory() {

        int i;

        // Allocate memory for vectors //
        Enew =                  calloc (elems,sizeof(double));
        Eold =                  calloc (elems,sizeof(double));
        Epert =                 calloc (elems,sizeof(double));
        f =                             calloc (elems,sizeof(double));
        EResidual =             calloc (elems,sizeof(double));

        nodalDisp =             calloc (2*nodes,sizeof(double));
        nodalDispPert = calloc (2*nodes,sizeof(double));
        actNodalDisp =  calloc (2*nodes,sizeof(double));
        dispResidual =  calloc (2*nodes,sizeof(double));

        // Allocate memory for matrices //
        Jg = calloc (nodes,sizeof(double *));
        for (i=1;i<nodes+1;i++) {
                Jg[i] = calloc (elems,sizeof(double));
        }
        Jg_t = calloc (elems,sizeof(double *));
        for (i=1;i<elems+1;i++) {
                Jg[i] = calloc (nodes,sizeof(double));
        }
        Jf = calloc (elems,sizeof(double *));
        for (i=1;i<elems+1;i++) {
                Jf[i] = calloc (elems,sizeof(double));
        }
        delE = calloc (elems,sizeof(double *));
        for (i=1;i<elems+1;i++) {
                delE[i] = calloc (1,sizeof(double));
        }
        nodePos = calloc (nodes,sizeof(double *));
        for (i=1;i<nodes+1;i++) {
                nodePos[i] = calloc (degOfFree,sizeof(double));
        }
        elemNodeMat = calloc (elems,sizeof(int *));
        for (i=1;i<elems+1;i++) {
                elemNodeMat[i] = calloc (nodesPerElem,sizeof(int));
        }
```

```
}

void nameFiles() {

                strcat( filePrefix, skeletonFile );
                strcat( filePrefix,fileNo );
                strcpy(inFile,filePrefix); strcpy(outFile,filePrefix);
                strcpy(resFile,filePrefix);strcpy(porFile,filePrefix);
                strcat( inFile,dotIn );
                strcat( outFile,dotOut );
                strcat( resFile,dotRes );
                strcat( porFile,dotPor );

}


void makeInFile() {

        FILE *rFile, *wFile;
        int i, iterate, lineNum;
        char buffer[256];
        char endCommandBuffer[10];

        rFile = fopen(skeletonInFile,"r");
        if(rFile==NULL) {
                printf("Error: can't open skeleton .in file\n");
  }

        wFile = fopen(inFile,"w");
        if(wFile==NULL) {
                printf("Error: can't create .in file\n");
  }

        iterate = 1;
        lineNum = 0;

        while (iterate != 0) {
                fgets( buffer,4096,rFile );
                iterate = strncmp( buffer,"MATERIAL ELASTIC",16 );
                lineNum = lineNum + 1;
        }

        rewind( rFile );

        for (i=0;i<(lineNum - 1);i++) {
                fgets(buffer,4096,rFile);
                fprintf( wFile, "%s",buffer );
        }

        for (i=0;i<elems;i++) {
                fgets(buffer,4096,rFile);
                fgets(buffer,4096,rFile);
                fgets(endCommandBuffer,4096,rFile);
        }

        if (perturb == TRUE) {
```

```
                for (i=1;i<elems+1;i++) {
                        fprintf( wFile, "MATERIAL ELASTIC NAME=%d E=%14.14f
                                NU=0.499000000,\n", i,Epert[i]);
                        fprintf( wFile, "%s",buffer );
                        fprintf( wFile, "%s",endCommandBuffer );
                }
        } else {
                for (i=1;i<elems+1;i++) {
                        fprintf( wFile, "MATERIAL ELASTIC NAME=%d E=%14.14f
                                NU=0.499000000,\n", i,Eold[i]);
                        fprintf( wFile, "%s",buffer );
                        fprintf( wFile, "%s",endCommandBuffer );
                }
        }

        iterate = 1;

        while (iterate!=0) {
                fgets(buffer,4096,rFile);
                fprintf( wFile, "%s",buffer );
                iterate = strncmp( buffer,"ADINA OPTIMIZE=SOLVER FILE=",28);
        }

        fgets(buffer,4096,rFile);
        fprintf( wFile,"'%s%s.dat',\n",newtonDir,filePrefix );

        fgets(buffer,4096,rFile);
        while (feof(rFile)==0) {
                fprintf( wFile,"%s",buffer);
                fgets(buffer,4096,rFile);
        }

        fclose( wFile );
        fclose( rFile );
}

// NOTE: runInFile() is NOT implementable yet //
void runInFile() {

        // Check which file to run //
        dos( ["'C:\Program Files\ADINA\ADINA System 8.1\bin\aui.exe" -b -m 100mb ',
                skeletonInFile ] );
        dos( ["'C:\Program Files\ADINA\ADINA System 8.1\bin\adina.exe" -b -s -m 100mb ',
                skeletonFilePrefix ] );

}

void readOutFileFull() {

        FILE *rFile;
        int i, j, iterate, lineNum, header, modelNodes, modelElems, inLineTokens;
        char *tok, *yPos, *zPos, *yTrans, *zTrans;
        char buffer[256];

        rFile = fopen(outFile,"r");
        if(rFile==NULL) {
```

```
                printf("Error: can't open skeleton .in file\n");
}

        iterate = 1;
        while (iterate!=0) {
                fgets(buffer,4096,rFile);
                iterate = strncmp(buffer, " CARD NUMBER 1",14);
        }

        fgets(buffer,4096,rFile);
        fgets(buffer,4096,rFile);

        inLineTokens = 16;
        tok = strtok( buffer, " ");
        for (i=1;i<inLineTokens;i++) {
                tok = strtok( NULL, " " );
        }
        modelNodes = atoi(tok);
        if (modelNodes!=nodes) {
                printf( "\nCHECK INITIAL NODES ASSIGNMENT!\n" );
        }

        iterate = 1;
        while (iterate!=0) {
                fgets(buffer,4096,rFile);
                iterate = strncmp(buffer, "  GENERATED NODAL DATA",22);
        }

        header = 7;
        for (i=1;i<header;i++) {
                fgets(buffer,4096,rFile);
        }

        for (i=1;i<nodes+1;i++) {
                fgets(buffer,4096,rFile);
                fgets(buffer,4096,rFile);
                tok = strtok( buffer," ");
                for (j=1;j<inLineTokens;j++) {
                        tok = strtok( NULL, " ");
                }

                yPos = strtok( NULL, " ");
                zPos = strtok( NULL, " ");
                nodePos[i][1] = atof( yPos );
                nodePos[i][2] = atof( zPos );
        }

        iterate = 1;
        while (iterate!=0) {
                fgets(buffer,4096,rFile);
                iterate = strncmp(buffer, " NUMBER OF ELEMENTS",19);
        }

        inLineTokens = 18;
        tok = strtok( buffer, " ");
        for (i=1;i<inLineTokens;i++) {
```

```
        tok = strtok( NULL, " ");
}
modelElems = atoi(tok);
if (modelElems!=elems) {
        printf( "\nCHECK INITIAL ELEMS ASSIGNMENT!\n" );
}

iterate = 1;
while (iterate!=0) {
        fgets(buffer,4096,rFile);
        iterate = strncmp(buffer, " E L E M E N T   I N F O R M A T I O N",39);
}

header = 6;
for (i=1;i<header;i++) {
        fgets(buffer,4096,rFile);
}
inLineTokens = 16;
for (i=1;i<elems+1;i++) {
        fgets(buffer,4096,rFile);
        fgets(buffer,4096,rFile);
        tok = strtok( buffer, " ");
        for (j=1;j<inLineTokens-5;j++) {
                tok = strtok( NULL, " ");
        }
        for (j=1;j<nodesPerElem+1;j++) {
                tok = strtok( NULL, " ");
                elemNodeMat[i][j] = atoi( tok );
        }
}

iterate = 1;
while (iterate!=0) {
        fgets(buffer,4096,rFile);
        iterate = strncmp( buffer," D I S P L A C E M E N T S",27);
}

header = 7;
for (i=1;i<header+1;i++) {
        fgets(buffer,4096,rFile);
}

for (i=1;i<nodes+1;i++) {
        fgets(buffer,4096,rFile);
        tok = strtok( buffer, " ");

        while ( atoi(tok) != i ) {
                fgets(buffer,4096,rFile);
                tok = strtok( buffer, " ");
        }

        tok = strtok( NULL, " ");
        yTrans = strtok( NULL, " ");
        zTrans = strtok( NULL, " ");

        // ACTUAL NODAL DISP //
```

```
                if (perturb == TRUE) {
                        nodalDispPert[2*i-1] = atof( yTrans );
                        nodalDispPert[2*i] = atof( zTrans );
                } else if (runActData = TRUE) {
                        actNodalDisp[2*i-1] = atof( yTrans );
                        actNodalDisp[2*i] = atof( zTrans );
                } else {
                        nodalDisp[2*i-1] = atof( yTrans );
                        nodalDisp[2*i] = atof( zTrans );
                }
        }

        fclose( rFile );

}

void readOutFile () {

        FILE *rFile;
        int i, j, iterate,header;
        char *tok, *yTrans, *zTrans;
        char buffer[256];

        rFile = fopen(outFile,"r");
        if(rFile==NULL) {
                printf("Error: can't open skeleton .in file\n");
  }

        iterate = 1;
        while (iterate!=0) {
                fgets(buffer,4096,rFile);
                iterate = strncmp(buffer, "  D I S P L A C E M E N T S",27);
        }

        header = 7;
        for (i=1;i<header+1;i++) {
                fgets(buffer,4096,rFile);
        }

        for (i=1;i<nodes+1;i++) {
                fgets(buffer,4096,rFile);
                tok = strtok( buffer, " ");

                while ( atoi(tok) != i ) {
                        fgets(buffer,4096,rFile);
                        tok = strtok( buffer, " ");
                }

                tok = strtok( NULL, " ");
                yTrans = strtok( NULL, " ");
                zTrans = strtok( NULL, " ");

                // ACTUAL NODAL DISP //
                if (perturb == TRUE) {
                        nodalDispPert[2*i-1] = atof( yTrans );
                        nodalDispPert[2*i] = atof( zTrans );
```

```
                } else if (runActData = TRUE) {
                        actNodalDisp[2*i-1] = atof( yTrans );
                        actNodalDisp[2*i] = atof( zTrans );
                } else {
                        nodalDisp[2*i-1] = atof( yTrans );
                        nodalDisp[2*i] = atof( zTrans );
                }
        }

        fclose( rFile );
}

// NOTE: deleteFile() is NOT implementable yet //
void deleteFiles() {

}
```

// The function 'gaussj.c' uses Gauss-Jordan elimination to invert the matrix [a]. It was borrowed from *Numerical Recipes in C: The Art of Scientific Computing* by William H. Press //

// NOTE: In order for 'gaussj.c' to function, 'nrutil.h' and 'nrutil.c' are needed (also obtained from *Numerical Recipes in C: The Art of Scientific Computing* by William H. Press //

```
#include <math.h>
#define NRANSI
#include "nrutil.h"
#define SWAP(a,b) {temp=(a);(a)=(b);(b)=temp;}

void gaussj(double **a, int n, double **b, int m)
{
	int *indxc,*indxr,*ipiv;
	int i,icol,irow,j,k,l,ll;
	double big,dum,pivinv,temp;

	indxc=ivector(1,n);
	indxr=ivector(1,n);
	ipiv=ivector(1,n);
	for (j=1;j<=n;j++) ipiv[j]=0;
	for (i=1;i<=n;i++) {
		big=0.0;
		for (j=1;j<=n;j++)
			if (ipiv[j] != 1)
				for (k=1;k<=n;k++) {
					if (ipiv[k] == 0) {
						if (fabs(a[j][k]) >= big) {
							big=fabs(a[j][k]);
							irow=j;
							icol=k;
						}
					} else if (ipiv[k] > 1) nrerror("gaussj: Singular Matrix-1");
				}
		++(ipiv[icol]);
		if (irow != icol) {
			for (l=1;l<=n;l++) SWAP(a[irow][l],a[icol][l])
			for (l=1;l<=m;l++) SWAP(b[irow][l],b[icol][l])
		}
		indxr[i]=irow;
		indxc[i]=icol;
		if (a[icol][icol] == 0.0) nrerror("gaussj: Singular Matrix-2");
		pivinv=1.0/a[icol][icol];
		a[icol][icol]=1.0;
		for (l=1;l<=n;l++) a[icol][l] *= pivinv;
		for (l=1;l<=m;l++) b[icol][l] *= pivinv;
		for (ll=1;ll<=n;ll++)
			if (ll != icol) {
				dum=a[ll][icol];
				a[ll][icol]=0.0;
				for (l=1;l<=n;l++) a[ll][l] -= a[icol][l]*dum;
				for (l=1;l<=m;l++) b[ll][l] -= b[icol][l]*dum;
			}
	}
	for (l=n;l>=1;l--) {
		if (indxr[l] != indxc[l])
```

```
                        for (k=1;k<=n;k++)
                                SWAP(a[k][indxr[l]],a[k][indxc[l]]);
        }
        free_ivector(ipiv,1,n);
        free_ivector(indxr,1,n);
        free_ivector(indxc,1,n);
}
#undef SWAP
#undef NRANSI
```

// **matrix_vector.h** //

```
double **transpose (double **M, int m, int n);
double vector_norm( double *v, int m );
double vector_vector_mult( double *v1, double *v2, int n );
double *vector_const_mult( double *v, double constant, int n );
double *vector_vector_sum( double *v1, double *v2, double factor, int n );
double *matrix_vector_mult( double *result_vector, double **Mat, double *v, int n );
double **matrix_matrix_mult( double **M, double **M1, double **M2, int m, int n );
```

// **matrix_vector.c** //

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "matrix_vector.h"

float **transpose (float **Mt, float **M, int m, int n) {

        int i,j;
        for (i=1;i<n+1;i++) {
                for (j=1;j<m+1;j++) {
                        Mt[i][j] = M[j][i];
                }
        }
        return Mt;
}

float vector_norm( float *v, int m ) {

        float norm_value = 0;
        int i;

        for( i = 1; i< m+1; i++ ) {
                norm_value = norm_value + pow( v[ i ],2 );
        }
        norm_value = sqrt( norm_value );
        return norm_value;
}

float *matrix_vector_mult( float *result_vector, float **Mat, float *v, int m, int n ) {

        int i,j;
        for( i = 1; i < m+1; i++ ) {
                for( j = 1; j < n+1; j++ ) {
                        result_vector[ i ] = result_vector[ i ] + ( Mat[ i ][ j ] * v[ j ] );
                }
        }
        return result_vector;
}

float vector_vector_mult( float *v1, float *v2, int n ) {

        int i;
        float result_value;

        result_value = 0;
```

```
        for( i = 1; i < n+1; i++ ) {
                result_value = v1[ i ] * v2[ i ] + result_value;
        }
        return result_value;
}

float *vector_const_mult( float *v, float constant, int n ) {

        int i;
        for( i = 1; i < n+1; i++ ) {
                v[ i ] = v[ i ] * constant;
        }
        return v;
}

float *vector_vector_sum( float *v1, float *v2, float factor, int n ) {

        int i;
        for( i = 1; i < n+1; i++ ) {
                v1[ i ] = v1[ i ] + factor * v2[ i ];
        }
        return v1;
}

float **matrix_matrix_mult( float **M, float **M1, float **M2, int m, int n ) {

        int i,j;
        float *MColumn;
        float *M2Column;

        M2Column = calloc(n,sizeof(float));
        for (i=1;i<m+1;i++) {
                MColumn = calloc(m,sizeof(float));
                for (j=1;j<m+1;j++) {
                        MColumn[j] = 0;
                }
                for (j=1;j<n+1;j++) {
                        M2Column[j] = M2[j][i];
                }
                MColumn = matrix_vector_mult(MColumn,M1,M2Column,m,n);
                for (j=1;j<m+1;j++) {
                        M[j][i] = MColumn[j];
                }
        }
        return M;
}
```

A.3 Normal Gaussian Distribution: MATLAB

```
% This function makes a 2-D mesh grid based on the number of elements and nodes in a given FEM
% model and then assigns values to the elements based on a 2-D normal Gaussian distribution

nodes = 121;
elems = 100;

step = 1/(sqrt(nodes)-1);
[x,y] = meshgrid(0:step:1,0:step:1);

sigX = (1/(sqrt(2*pi)))^2;
sigY = (1/(sqrt(2*pi)))^2;

% Axes of symmetry
muX = 0.55;
muY = 0.55;

% Such that f_max = 2
beta1 = 2.27105;
beta2 = 1;

% Normal Gaussian Distribution (f_min = 1, f_max = 2, x_min = 0, x_max = 1)
f = ((1/beta1)*(1/(sqrt(2*pi*sigX*sigY))).*exp(-((((x+step)-muX).^2)./(2*sigX^2) + (((y+step)-
muY).^2)./(2*sigY^2))))+beta2;

Fmax = max(max(f))
surface(x,y,f);
xlabel( 'y-axis [length units]' );
ylabel( 'z-axis [length units]' );
zlabel( 'Relative Young"s Modulus' );

E(1,1)  =  f(10,10);
E(2,1)  =  f(10,9);
E(3,1)  =  f(10,8);
E(4,1)  =  f(10,7);
E(5,1)  =  f(10,6);
E(6,1)  =  f(10,5);
E(7,1)  =  f(10,4);
E(8,1)  =  f(10,3);
E(9,1)  =  f(10,2);
E(10,1) =  f(10,1);
E(11,1) =  f(9,1);
E(12,1) =  f(8,1);
E(13,1) =  f(7,1);
E(14,1) =  f(6,1);
E(15,1) =  f(5,1);
E(16,1) =  f(4,1);
E(17,1) =  f(3,1);
E(18,1) =  f(2,1);
E(19,1) =  f(1,1);
E(20,1) =  f(1,2);
```

```
E(21,1) =       f(1,3);
E(22,1) =       f(1,4);
E(23,1) =       f(1,5);
E(24,1) =       f(1,6);
E(25,1) =       f(1,7);
E(26,1) =       f(1,8);
E(27,1) =       f(1,9);
E(28,1) =       f(1,10);
E(29,1) =       f(2,10);
E(30,1) =       f(3,10);
E(31,1) =       f(4,10);
E(32,1) =       f(5,10);
E(33,1) =       f(6,10);
E(34,1) =       f(7,10);
E(35,1) =       f(8,10);
E(36,1) =       f(9,10);
E(37,1) =       f(9,9);
E(38,1) =       f(9,8);
E(39,1) =       f(9,7);
E(40,1) =       f(9,6);
E(41,1) =       f(9,5)
E(42,1) =       f(9,4);
E(43,1) =       f(9,3);
E(44,1) =       f(9,2);
E(45,1) =       f(8,2);
E(46,1) =       f(7,2);
E(47,1) =       f(6,2);
E(48,1) =       f(5,2);
E(49,1) =       f(4,2);
E(50,1) =       f(3,2);
E(51,1) =       f(2,2);
E(52,1) =       f(2,3);
E(53,1) =       f(2,4);
E(54,1) =       f(2,5);
E(55,1) =       f(2,6);
E(56,1) =       f(2,7);
E(57,1) =       f(2,8);
E(58,1) =       f(2,9);
E(59,1) =       f(3,9);
E(60,1) =       f(4,9);
E(61,1) =       f(5,9);
E(62,1) =       f(6,9);
E(63,1) =       f(7,9);
E(64,1) =       f(8,9);
E(65,1) =       f(8,8);
E(66,1) =       f(8,7);
E(67,1) =       f(8,6);
E(68,1) =       f(8,5);
E(69,1) =       f(8,4);
E(70,1) =       f(8,3);
E(71,1) =       f(7,3);
E(72,1) =       f(6,3);
E(73,1) =       f(5,3);
E(74,1) =       f(4,3);
E(75,1) =       f(3,3);
E(76,1) =       f(3,4);
```

```
E(77,1) =       f(3,5);
E(78,1) =       f(3,6);
E(79,1) =       f(3,7);
E(80,1) =       f(3,8);
E(81,1) =       f(4,8);
E(82,1) =       f(5,8);
E(83,1) =       f(6,8);
E(84,1) =       f(7,8);
E(85,1) =       f(7,7);
E(86,1) =       f(7,6);
E(87,1) =       f(7,5);
E(88,1) =       f(7,4);
E(89,1) =       f(6,4);
E(90,1) =       f(5,4);
E(91,1) =       f(4,4);
E(92,1) =       f(4,5);
E(93,1) =       f(4,6);
E(94,1) =       f(4,7);
E(95,1) =       f(5,7);
E(96,1) =       f(6,7);
E(97,1) =       f(6,6);
E(98,1) =       f(6,5);
E(99,1) =       f(5,5);
E(100,1)=       f(5,6);
```

A.4 Finite Element Modeling–Genetic Algorithm System: MATLAB

```
clear all;

format long e;

%%%%%%%%%% USER-DEFINED PARAMETERS %%%%%%%%%%

% Number of parameters
n = 3;

% Number of elements in each parameter region
elements( 1 ) =  1580;
elements( 2 ) =  103;
elements( 3 ) =  102;
totalElements = sum( elements );

initialfilename = 'genetic_algorithm_model';
GAdir = 'C:\Documents and Settings\Ahmad Khalil\My
Documents\parameter_estimation\lumped_parameter_GA\';
ADINAdir = '"C:\Program Files\ADINA\ADINA System 8.1\bin\';
ADINAaui = strcat( ADINAdir, 'aui.exe" -b -m 100mb ' );
ADINA = strcat( ADINAdir, 'adina.exe" -b -s -m 100mb ' );

% Population size for each iteration
pop( 1:10 ) = 50;

% Initial population ranges
maxValue( 1 ) = 1e6;
minValue( 1 ) = 1e5;
maxValue( 2 ) = 1e4;
minValue( 2 ) = 1e3;
maxValue( 3 ) = 1e8;
minValue( 3 ) = 1e7;

% Create initial population
[ Eoffspring ] = initialize( pop( 1 ),n,minValue,maxValue )

%%%%%%%%%% RUN TARGET .in FILE %%%%%%%%%%

iterfilename = strcat( initialfilename,'_');
skeletonFile = strcat( initialfilename,'.in');
skeletonFilePrefix = initialfilename;

dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\aui.exe" -b -m 100mb ', skeletonFile ] );
dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\adina.exe" -b -s -m 100mb ', skeletonFilePrefix ]
);

outfilename = strcat( skeletonFilePrefix,'.out');
```

```
[ actStrains ] = readOutFile( n,outfilename,elements );

deleteAllFiles( skeletonFilePrefix )

% effActualStrains is an (elems x 1) vector of effective strain values
clear effActualStrains;
effActualStrains( totalElements,1 ) = zeros;
for i = 1:totalElements
   effActualStrains( i,1 ) = sqrt( actStrains( i,1 )^2 + actStrains( i,2 )^2 + 0.5 * actStrains( i,3 )^2 );
end
clear actStrains;

%%%%%%%%%% ITERATE %%%%%%%%%%

for iters = 1:size( pop,2 );

   if iters == 1
      numParents = pop( iters );
   else

      %%%%%%%%%% MATING POOL SELECTION %%%%%%%%%%

      clear Eoffspring; clear Eparents; clear survival;

      for j = 1:n
         Efitness( :,n+1+j ) = Efitness( :,n+1+j ) * normalizer( j );
      end

      numParents = pop( iters ) / 2;
      numCross = numParents;
      for i = 1:pop( iters )

         survivalCurve = 1;

         survival( 1 ) = 0;
         survival(pop( iters)-(i-2)) = i^survivalCurve / sum((1:1:pop(iters)).^survivalCurve);

      end

      % Use 'rank-survival curve'
      for i = 1:numParents
         for j = 2:pop( iters ) + 1
            r = rand;
            if r < 0.5 + ( sum( survival( 1:j ) ) )/2 & r > 0.5 - ( sum( survival( 1:j ) ) )/2;
               Eparents( i,1:2*n+1 ) = Efitness( j-1,1:2*n+1 );
               break
            end
         end
      end

      %%%%%%%%%% CROSSOVER REPRODUCTION %%%%%%%%%%

      % Best in population survives to next generation regardless of mating pool selection
      Eparents( numParents,1:2*n+1 ) = Efitness( 1,1:2*n+1 );

      % Crossover function
```

```
   [ Eoffspring ] = randCrossover( n,Eparents,numCross );

end

for i = 1:numParents

   filenumber = num2str( i );
   filedot = '.in';
   fileprefix = strcat( iterfilename, filenumber );
   infilename = strcat( iterfilename, filenumber, filedot );

   % Writes .in file
   makeInFile( n, Eoffspring( i,: ), infilename, fileprefix, skeletonFile, GAdir );

   % .in file -> .out file
   dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\aui.exe" -b -m 100mb ', infilename] );
   dos( ['"C:\Program Files\ADINA\ADINA System 8.1\bin\adina.exe" -b -s -m 100mb ', fileprefix] );

end

for i = 1:numParents

   filenumber = num2str( i );
   filedot = '.out';
   outfilename = strcat( iterfilename, filenumber, filedot );

   % Parse the .out file for strain values
   [ YZstrains ] = readOutFile( n,outfilename,elements );

   deleteAllFiles( fileprefix );

   % Put strains in 1-D vector of effective strains
   clear effPredStrains;
   effPredStrains( totalElements,1 ) = zeros;
   for elems = 1:totalElements
      effPredStrains( elems,1 ) = sqrt(YZstrains( elems,1 )^2 + YZstrains( elems,2 )^2 + ...
         0.5 * YZstrains( elems,3 )^2 );
   end

   diff = effPredStrains - effActualStrains;

   % Decouple objective (difference) values into distinct parameter regions
   diff( 1:elements( 1 ) ) = diff( 1:elements( 1 ) ) / elements( 1 );
   for j = 2:n
      diff( sum( elements( 1:j-1 ) ) + 1 : sum( elements( 1:j ) ) ) =...
         diff( sum( elements( 1:j-1 ) ) + 1 : sum( elements( 1:j ) ) ) / elements( j );
   end

   Eoffspring( i,n+1 ) = 0;
   Eoffspring( i,n+2 ) = norm( diff( 1:elements( 1 ) ) );
   for j = 2:n
      Eoffspring( i,n+1+j ) = norm( diff( sum(elements( 1:j-1 ))+1 : sum( elements( 1:j ))));
   end
end

clear YZstrains;
```

```
clear Efitness;

if iters == 1
   Efitness = Eoffspring;
else
   Efitness( 1:numParents,: ) = Eoffspring( 1:numParents,: );
   Efitness( numParents+1:pop( iters ),: ) = Eparents( 1:numParents,: );
end

% Normalize
clear normalizer;
for j = 1:n
   normalizer( j ) = sum( Efitness( :,n+1+j ) );
end
for j = 1:n
   Efitness( :,n+1+j ) = Efitness( :,n+1+j ) / normalizer( j );
end

% Weighting terms to obtain Total Fitness Values
for j = 1:n
   alpha( j ) = 1;
end

% Total Fitness Values = weighted sum of each parameter's normalized fitness value
Efitness( :,2*n+2 ) = zeros;
for j = 1:n
   Efitness( :,2*n+2 ) = Efitness( :,2*n+2 ) + alpha( j )*Efitness( :,n+1+j );
end

% Normalize
clear sumNormalizer;
sumNormalizer = sum( Efitness( :,2*n+2 ) );
Efitness( :,2*n+2 ) = Efitness( :,2*n+2 ) / sumNormalizer;

[ sortedRawFitness, Eindex ] = sort( Efitness( :,2*n+2 ) );
for i = 1:pop( iters )
   Etemp( i,: ) = Efitness( Eindex( i ),: );
end

Efitness = Etemp;
clear Etemp;

Efitness
if iters == 1
   E( 1:pop( iters ),: ) = Efitness( 1:pop( iters ),: );
else
   E( sum( pop( 1:iters-1 ) ) + 1 : sum( pop( 1:iters ) ),: ) = Efitness( 1:pop( iters ),: );
end
end
Econverged = Efitness( 1,: )
save;
```

```
% The function 'readOutFile' parses the FEM .out file and returns:

% 1. strains: (elems x 3) matrix of YY-, ZZ-, and YZ-strains

function [ strains ] = readOutFile( n,filename,elements );

% Open the file. If this returns a -1, the file was not opened successfully
fid = fopen( filename );
if fid == -1
  error( 'File not found or permission denied' );
end

totalElements = sum( elements );
strains( totalElements,3 ) = zeros;

for i = 1:n

   iterate = 0;

   while( iterate == 0 )
      buffer = fgetl( fid );
      iterate = strncmp( buffer, ' S T R E S S   C A L C U L A T I O N S', 38);
   end

   strainHeader = 12;
   for ii = 1:strainHeader, buffer = fgetl( fid ); end

   if i == 1
      start = 1;
      finish = elements( i );
   else
      start = 0;
      for c = 1:i-1
         start = start + elements( c );
      end
      finish = start + elements( i );
      start = start + 1;
   end

   for ii = start:finish

      totalStrainYY = 0;
      totalStrainZZ = 0;
      totalStrainYZ = 0;

      for iii = 1:9
         buffer = fgetl( fid ); buffer = fgetl( fid ); buffer = fgetl( fid );
         [ strainxx_string, buffer ] = strtok( buffer );
         [ strainyy_string, buffer ] = strtok( buffer );
         [ strainzz_string, buffer ] = strtok( buffer );
         [ strainyz_string, buffer ] = strtok( buffer );
         strainyy = str2num( strainyy_string );
         strainzz = str2num( strainzz_string );
         strainyz = str2num( strainyz_string );
         totalStrainYY = totalStrainYY + strainyy;
         totalStrainZZ = totalStrainZZ + strainzz;
```

```
        totalStrainYZ = totalStrainYZ + strainyz;
      end

      % Elemental strain = average of the strain values for each nodal point
      strains( ii,1 ) = totalStrainYY / 9;
      strains( ii,2 ) = totalStrainZZ / 9;
      strains( ii,3 ) = totalStrainYZ / 9;

      buffer = fgetl( fid ); buffer = fgetl( fid );

    end
end

fclose( fid );
```

```
% The function 'makeInFile' uses a skeleton .in file ('skeleton') to write a new .in file that
% builds the same model but with the given Young's modulus distribution vector (E)

function makeInFile( n, E, filename, fileprefix, skeleton, GAdir )

% Open the file. If this returns a -1, the file was not opened successfully
fidR = fopen( skeleton, 'r' );
if fidR == -1
  error( 'File not found or permission denied' );
end

fidW = fopen( filename, 'w' );
if fidW == -1
  error( 'File not found or permission denied' );
end

iterate = 0;
lineNum1 = 0;

while ( iterate == 0 )
   buffer = fgetl( fidR );
   iterate = strncmp( buffer, 'MATERIAL ELASTIC', 16);
   lineNum1 = lineNum1 + 1;
 end


frewind( fidR );

for i = 1 : ( lineNum1 - 1 )
   buffer = fgetl( fidR );
   fprintf( fidW, '%s\n', buffer );
end

for i = 1 : n

   % Get the 'Material Elastic Name...' line from the skeleton file
   buffer = fgetl( fidR );
   % Get the 'Density...Alpha' line from the skeleton file
   buffer = fgetl( fidR );
   EndCommandBuffer = fgetl( fidR );

end

for i = 1 : n
   fprintf( fidW, 'MATERIAL ELASTIC NAME=%g E=%14.14e NU=0.499000000000000,\n', i, E( i ) );
   fprintf( fidW, '%s\n', buffer );
   fprintf( fidW, '%s\n', EndCommandBuffer );
end

iterate = 0;


while ( iterate == 0 )
   buffer = fgetl( fidR );
   fprintf( fidW, '%s\n', buffer );
   iterate = strncmp( buffer, 'ADINA OPTIMIZE=SOLVER FILE=,', 28)  ;
```

```
end

buffer = fgetl( fidR );
fprintf( fidW, '"%s%s.dat",\n',GAdir,fileprefix );

buffer = fgetl( fidR );
while ( buffer ~= ( -1 ) )
   fprintf( fidW, '%s\n', buffer );
   buffer = fgetl( fidR );
end

fclose( fidR );
fclose( fidW );
```

```
% The function 'randCrossover' uses a simple parameter switching method to create offspring
% from the parents passed in.

function [ Eoffspring ] = randCrossover( n,Eparents,numCross )

numOffspring = size( Eparents,1 );

for i = 1:numCross

   crossPartner( i ) = round( rand*( numOffspring - 0.01 ) + 0.5 );

   while crossPartner( i ) == i
      crossPartner( i ) = round( rand*( numOffspring - 0.01 ) + 0.5 );
   end

   crossNo = round( rand*( n-1-0.01 ) + 0.5 );

   for j = 1:crossNo

      crossLoc( j ) = round( rand*( n-0.01 ) + 0.5 );
      if j > 1
         notYet = 0;
         while notYet == 0;
            for c = 1:j-1
               if crossLoc( j ) == crossLoc( j-c )
                  crossLoc( j ) = round( rand*( n-0.01 ) + 0.5 );
                  break;
               end
               notYet = 1;
            end
         end
      end

      if crossLoc( j ) == 1;
         Eoffspring( i,crossLoc( j ) ) = Eparents( crossPartner( i ),crossLoc( j ) );
         Eoffspring( i,crossLoc( j )+1:n ) = Eparents( i,crossLoc( j )+1:n );
      elseif crossLoc( j ) == n
         Eoffspring( i,crossLoc( j ) ) = Eparents( crossPartner( i ),crossLoc( j ) );
         Eoffspring( i,1:crossLoc( j )-1 ) = Eparents( i,1:crossLoc( j )-1 );
      else
         Eoffspring( i,crossLoc( j ) ) = Eparents( crossPartner( i ),crossLoc( j ) );
         Eoffspring( i,1:crossLoc( j )-1 ) = Eparents( i,1:crossLoc( j )-1 );
         Eoffspring( i,crossLoc( j )+1:n ) = Eparents( i,crossLoc( j )+1:n );
      end
   end
end
```

```
% The function 'initialize' builds a random initial population based on the range of values passed in

function [ Eoffspring ] = initialize( pop,n,minValue,maxValue )

for i = 1:n
   Eoffspring( 1:pop,i ) = linspace( minValue( i ),maxValue( i ),pop )';
end

cpEoffspring = Eoffspring;

if n > 1
   N = n-1;
   randomize = 0;
else
   randomize = 1;
end
swapPos = round( rand*pop );

for i = 1:N
   if randomize == 1
      break
   end
   for j = 1:pop
      if Eoffspring( j,i ) == Eoffspring( j,i+1 )

         while swapPos == 0;
            swapPos = round( rand*pop );
         end

         swap = Eoffspring( j,i );
         Eoffspring( j,i ) = Eoffspring( swapPos,i );
         Eoffspring( swapPos,i ) = swap;

         swapPos = round( rand*pop );

         while swapPos == 0;
            swapPos = round( rand*pop );
         end

         swap = Eoffspring( j,i );
         Eoffspring( j,i ) = Eoffspring( swapPos,i );
         Eoffspring( swapPos,i ) = swap;
      else
         swapPos = round( rand*pop );
         while swapPos == 0;
            swapPos = round( rand*pop );
         end

         swap = Eoffspring( j,i );
         Eoffspring( j,i ) = Eoffspring( swapPos,i );
         Eoffspring( swapPos,i ) = swap;
      end
   end
end
```

% The function 'deleteAllFiles' deletes all the superfluous ADINA FEM files

```
function deleteAllFiles( filePrefix )

porFile = strcat( filePrefix, '.por' ); dos ( ['del ',porFile] );
datFile = strcat( filePrefix, '.dat' ); dos ( ['del ',datFile] );
resFile = strcat( filePrefix, '.res' ); dos ( ['del ',resFile] );
modFile = strcat( filePrefix, '.mod' ); dos ( ['del ',modFile] );
outFile = strcat( filePrefix, '.out' ); dos ( ['del ',outFile] );
inFile = strcat( filePrefix, '.in' ); dos ( ['del ',inFile]
```