Integral-Based Inverse Problem Solutions for DIET Systems

A thesis submitted in partial fulfilment for the Degree of

Master of Engineering (Mechanical)

In the University of Canterbury

by

Samuel J. Houghton

University of Canterbury

2006

Abstract

Magnetic Resonance Elastography (MRE) is an emerging method for non-invasive breast cancer screening. It takes the MRI displacement data output and reconstructs the internal stiffness distribution, where cancerous tissue is approximately five to ten times stiffer than healthy breast tissue. Hence, MRE offers a high contrast solution to this diagnostic problem.

Current MRE methods for reconstructing stiffness use forward simulation based optimization methods that are highly non-linear, non-convex and very heavy computationally. This research develops integral-based inverse problem solutions that reformulate the underlying differential equations in terms of integrals of MRI measured displacement data, and this transforms the problem into a linear, convex optimization. All derivative terms in the formulation are removed by special choice of integration limits, so no smoothing or filtering of the input data is required. The resulting equations can easily be solved by linear least squares requiring very minimal computation.

1D inverse algorithms were developed to provide a proof of concept of the integral-based method. Initially, the complete compressible 2D Navier's equations were used to develop the 2D inverse methods. Reasonable results were achieved with the algorithm successfully identifying a 1cm by 1cm tumour with up to 10% noise, data resolution of 20 measured points per cm and actuation frequencies of 100Hz.

However, for the same input data set, a simplified incompressible 2D model was used as the basis for the final proposed inverse algorithm. This approach significantly improved results by removing ill-conditioned terms from the original formulation. For a 1cm by 1 cm tumour, accurate results were obtained with up to 40% noise, a range of actuation frequencies and very low data resolution of the order of 2 measured points per cm. These results thus indicate that more crude and less expensive data measurement systems could be used to obtain good results.

The methods developed can be readily extended to 3D by applying a similar incompressible integral formulation to the 3D Navier’s equations.

Acknowledgements

I would like to thank the following people:

- Christopher E. Hann for his assistance, patience, mathematical expertise and seemingly endless enthusiasm towards this research project.

- Associate Professor J. Geoffrey Chase for his invaluable advice and guidance, as well as providing me the opportunity to undertake this research

- Dr. Eli Van Houten for his technical assistance and second opinion.

- Ishan Singh-Levett for his friendship and companionship that has made my university experience all the more enjoyable.

- My parents, Phillip and Karen, for their love & support

- All my friends in Christchurch that have helped me maintain some form of sanity throughout my time at the University of Canterbury.

Table of Contents

Part 4: Conclusions

Appendix A: MATLAB code

List of Figures

Part 1

Introduction

1 Introduction

1.1 Breast Cancer

Breast Cancer is a serious health problem amongst women. It is the most common form of female cancer and is the second most fatal cancer among women worldwide [American Cancer Society, 2006]. One in ten women will suffer from breast cancer during their lives and 25% of those who develop it are predicted to die from the disease [NZBCF].

In New Zealand, breast cancer accounts for the most fatalities of all cancers in women and has the fifth highest incidence per 100,000 women of breast cancer in the world [American Cancer Society, 2006]. In 2001, 615 women died from breast cancer and this constituted 4.4% of all female mortalities. There were also 2310 new registrations of breast cancer in the same year [NZHIS, 2001].

A number of factors have been linked to an increased risk of developing breast cancer, including age, daily alcohol consumption and genetics, although only a small percentage (approximately 20%) of occurrences of breast cancer have been known to develop from these known risk factors [Robertson, 2005]. Very little is fully understood regarding the occurrence and development of breast cancer. In particular, there exists no proven preventative intervention or treatment that reverses malignant tumour growth. Therefore, early detection remains the best means of survival [National Breast Cancer Coalition].

Early detection is particularly critical as a patient survival rate is directly related to the size of tumour and degree that it has spread. If the malignant tumour is discovered while it is still localized (it is less than 2cm in size and has not spread to the lymph nodes) the five year survival rate is 95% [Cokkinides et al., 2004]. Hence, there is a great deal of research on new methods enabling cancer detection than is currently possible.

1.2 Existing Methods of Breast Cancer Screening

There are currently a number of methods used to detect breast cancer. These include: Manual Palpation, Mammography, MRI (Magnetic Resonance Imaging) and Ultrasound. Before the introduction of modern screening techniques, manual palpation was the most effective method of cancer detection, and even in modern technology it remains one of the most common forms of detection. This involves a manual physical examination for lumps in the breast tissue, either performed by a medical practitioner or the individual themselves. Manual Palpation is still currently an important method for breast cancer detection and regular self-examination is considered an important preventative measure. However, the size of the cancerous lump has to exceed 1-2 cm in order to be located easily [Hii, 2005] and by this stage the cancerous cells may have spread beyond the breast.

Mammography represents the principal and most effective technology currently available for breast cancer screening. Since its widespread introduction in the 1970's, mortality resulting from breast cancer has significantly decreased [Tabar et al. 2003]. The procedure is performed by compressing each breast between two plastic plates in order to spread the tissue apart. The compression of the breasts ensures that a minimal dose of radiation is required and that little movement occurs during screening. It also provides a much sharper image as a thinner layer is imaged. An X-ray image is then recorded onto radiographic film and must be interpreted by a highly skilled radiologist. The accuracy of this technique is therefore highly dependent on the ability and experience of the radiologist reading the resulting images [Kopans. 1998].

There are a number of drawbacks associated with mammography. Patients report a great level of pain and discomfort during the process of compressing the breast. The patients are also constantly subjected to radiation if regular screening occurs and this provides an additional health risk. However, due to refinements in the procedure (including breast compression) this level of radiation has been significantly reduced in comparison to traditional X-ray methods, but it still restricts the number of years of screening and this makes it available primarily after the age of 50.

The level of discomfort and radiation dose has somewhat tarnished the reputation of the procedure and can lead to low screening compliance rates among eligible women [Chamberlain,

2002]. The large size and capital cost of mammography equipment also restricts the location of screening facilities to urban centres. This too causes lower screening rates, as women living in rural areas are not always able to maintain a regular screening programme due to large travel distances and inflexible scheduling for the procedure [Robertson, 2005].

Contrast-enhanced Magnetic Resonance Imaging (MRI) is a technique that has proved to be very useful at detecting breast cancer and providing further diagnosis [Bone et al. 1998], particularly among woman with very dense breasts. However, the widespread application of this technology is limited by the significant cost and size of the equipment required. It is also not recommended as an initial screening method because it has been known to find abnormalities that are not breast cancer, resulting in an increased number of unnecessary biopsy procedures [National Cancer Institute]. Finally, its slow turnaround time means only a very few women could be screened each day per MRI system, making it impractical large-scale screening.

Ultrasound, also known as sonography, is an imaging method that uses high-frequency sound waves to create an image of the breast. The method involves a medical practitioner moving a handheld probe (transducer) across the surface of the breast and interpreting the image displayed on a computer screen. Although it is a valuable tool for diagnosis alongside a Mammogram as it is widely available, non-invasive and relatively inexpensive, it is not recommended as a primary breast screening method as the image contrast is poor and requires a highly skilled and experienced operator to interpret the output images leading to subjective and inaccurate results [Chamberlain, 2002]. In addition the manual nature of this method brings into doubt the geometric repeatability of the process, which is important when trying to implement a widespread, repeatable breast screening method [Peters et al. 2004].

1.3 Emerging Elastographic Screening Methods

Elastographic techniques concentrate on the high contrast between the elastic properties of the carcinoma and the surrounding healthy breast tissue. Separate studies completed by Krouskop et al. (1998) and Samani et al. (2003) measuring the elastic moduli of human tissue have shown invasive ductal carcinoma, or cancerous tumour, to be approximately an order of magnitude stiffer than fibroglandular tissue from a healthy breast. Currently, there are a number of methods under development that utilise the high contrast in elastic properties.

Magnetic Resonance Elastography (MRE) uses harmonic mechanical displacements measured from an MRI to calculate the internal elasticity distribution using an inverse problem algorithm. This technique may provide an effective means of screening for tumours within the breast without the false positives inherent in current MRI methods. However, this method still requires an MRI system, which is expensive, not transportable, and has very low turnaround.

Digital Image-Based Elasto-Tomography (DIET) is an emerging new technology that uses digital imaging of a sinusoidally actuated breast surface to determine the surface motion of the tissue [Peters et al. 2004, 2005]. The surface motion is then used to reconstruct the three dimensional internal elasticity distribution. The system consists of four distinct steps, which are summarized in Figure 1.1:

1. An actuator positioned on the surface of the breast, which operates at constant frequency and amplitude and induces steady-state sinusoidal motion throughout the breast tissue.

2. High resolution, spatially calibrated cameras are positioned in an array over the breast. These take a sequence of 2D images of reference points on the surface of the breast.

3. The consecutive 2D motion output from the cameras is converted into 3D time varying motion vector for each reference point using image processing algorithms.

4. The 3D internal elasticity distribution of the breast tissue is reconstructed using an inverse algorithm with the surface motion vectors used as input.

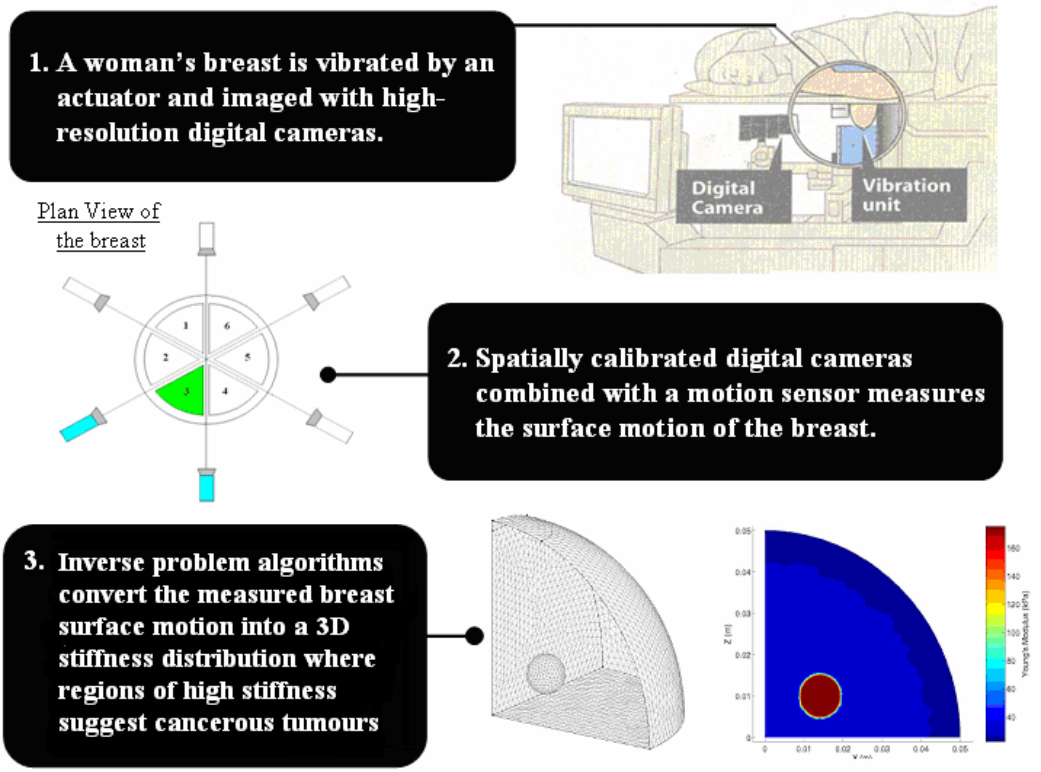

Figure 1.1 – Summary of the DIET process

The DIET system has a number of potential advantages over existing and developing methods of breast cancer screening. These include [Robertson, 2005];

- Transportability
- Low Cost – Equipment is less capitally intensive
- Breast tissue is not exposed to harmful radiation
- More comfortable than a mammogram
- Elastographic contrast between carcinoma and healthy tissue
- Objective interpretation of results means that technique does not require as highly skilled operators as other methods
- Can collate a history of results for comparison and diagnosis
- Suitable for dense breast tissue

These potential benefits mean that the DIET system could increase screening compliance rates, as the portable and low cost nature of the device means that screening is more accessible, as well as eliminating the disincentive for screening that the often uncomfortable mammogram process initiates. The ability of the system to screen breasts with more dense issue gives the ability to

screen younger women, with whom this breast property is characteristic, particularly those with a genetic disposition towards the disease. The elimination of doses of radiation also means that women in the at-risk group of 40-55 years of age, or even younger, can be screened more often, thus increasing the chances of early detection.

1.4 Existing Inverse Problem Solutions

The use of Magnetic Resonance Imaging (MRI) has been investigated for use in cancer detection and diagnosis. The signal generated from carefully aligned nuclear spins is used to create cross-sectional images and obtain displacement data throughout the tissue, at a resolution of approximately 1.52 measurements per $1mm^3$, creating the potential for sub-millimetre detection of tumours. However, to create a stiffness image at the same voxel resolution as the original MR data using traditional global optimization approaches leads to an impracticably large computational problem.

More specifically, for a typical MR dataset of 16 256x256 voxel image slices, the required RAM for the update matrix would be roughly 5000Gb, which is beyond the current state of computer technology to invert and solve [Van Houten et al. 1999]. Breast MRI is also not currently used for routine cancer screening. However, clinical trials are being performed to determine its value in testing women at high risk for breast cancer [National Cancer Institute, 2002]. The limitations of current computer technology necessitate the implementation of innovative inverse problem solution algorithms that are computationally efficient.

Previous work with Elastographic inverse problem solutions has shown that is it possible to solve MR Elastography problems using conventional non-linear optimization by dividing the total problem domain into sub-zones [Van Houten et al. 1999]. An estimated stiffness distribution is found that minimizes the error between measured and computed displacements. Simulations using 3D breast geometry indicate that the algorithm can detect 1cm diameter hard inclusions with 2.5x elasticity contrast to the surrounding tissue [Van Houten et al. 2001]. However, the implementation of these algorithms remains computationally intense.

1.5 Proposed Inverse Problem Solution

This thesis investigates and formulates novel integration-based inverse problem solution algorithms to be applied in Magnetic Resonance Elastography (MRE) and eventually DIET systems. The final algorithm is intended to take measured or estimated global 3D harmonic displacements of the tissue throughout the breast and use it to calculate the internal stiffness distribution. The integral fitting concept is extended from ordinary differential equation models [Hann et al. 2005, 2006] to a partial differential equation model [Peters et al. 2004]. This approach transforms the problem into a linear and convex identification problem with no forward simulations required and involving very minimal computation on a standard personal computer [Hann et al. 2005, 2006].

The closet comparison to the proposed integral-based method is an optimization based method adopted by Van Houten et al. (1999, 2001). Both are model based reconstructions that can operate with unfiltered data. The main advantage of the integral method is that it doesn't require an iterative solution because the best fit is obtained directly using linear least squares.

Chapter 2 of this thesis details the simulation of the 1D Navier equation and integral-based parameter identification to identify the required 1D stiffness distribution. Chapter 3 derives a finite difference formulation of the 2D Navier equation. Chapter 4 presents the integral formulation in the 2D homogeneous case and details the general 2D inverse problem algorithm for the non-homogeneous case. It also briefly outlines a proposed extension to the three dimensional case. Results and discussion for the 1D and 2D inverse problem algorithms implemented on simulated data with random noise added are shown Chapters 5 & 6, respectively. Conclusions and future work are discussed in Chapter 7.

Part 2

Methodology

2 One Dimensional Inverse Problem

2.1 Forward Simulation of One Dimensional Data

Although the displacement response of an actuated breast can not realistically be modelled in a single dimension, it is necessary to first verify the proposed integral-based inverse problem solution in this simple case. The results from this analysis can then be used to identify problems and limitations of the method as well as establish potential frameworks from which higher dimensional simulation models and inverse problem solution algorithms could be developed.

One dimensional motion data is simulated using the one dimensional, time harmonic simplification of Navier's Equation, given by Equations (2.1) – (2.2b). This approximation represents the propagation of shear waves through the one dimensional medium as a result of forced harmonic displacement which is applied transversely to the direction of the medium, akin to the shaking of a piece of string. It ignores the effects of longitudinal compressive waves along the one dimensional medium. When modelling human tissue, this approximation is appropriate as tissue is effectively incompressible and the resulting waves are insignificant in comparison with the shear deformations.

$$v(x,t) = v(x)e^{i\omega t} \tag{2.1}$$

$$\frac{d}{dx}\left(\frac{dv}{dx}G\right) = -\rho\omega^2 v \tag{2.2}$$

$$G = \frac{E}{2+2\nu} \tag{2.2b}$$

where:

G = Shear Modulus (Pa)

ν = Poisson's Ratio

E = Young's Modulus (Pa)

ρ = Tissue Density (kgm^{-3})

v = Displacement perpendicular to model axis (m)

x = Position (m)

ω = Harmonic Actuation Frequency ($rads^{-1}$)

2.1.1 Homogeneous Model

Initially, the model is assumed to be homogeneous in order to simplify Equation (2.2). Specifically, if the Shear Modulus term (G) given by Equation (2.2) is constant, it can be removed from the derivative term, leading to:

$$Gv_{xx} = -\rho\omega^2 v \tag{2.3}$$

For $m+1$ equally spaced points along the global domain $0 \le x \le L$, finite difference approximations are used to numerically compute the second derivative term v_{xx} in Equation (2.3), defined as follows:

$$v_{xx}(x_i) = \frac{v(x_{i-1}) - 2v(x_i) + v(x_{i+1})}{h^2} + O(h^2), \quad i = 1,...,m-1 \tag{2.4}$$

$$h = L/m \tag{2.5}$$

where:

x_i = $ih,\ i = 0,...,m$

h = distance between discrete points (m)

Equation (2.4) represents the central difference approximation. The forward and backward differences need not be applied to the end points, x_0 and x_m, as these are the points at which boundary conditions are applied. This enables a linear system of equations to be formulated in terms of the displacement amplitudes after boundary conditions are applied, as shown in Equations (2.6-2.8).

$$A\underline{v} = b \tag{2.6}$$

Where A is an $m\,x\,m$ matrix and b is an $m\,x1$ vector of the forms:

$$A = \frac{1}{h^2}\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ G & -2G+\rho\omega^2 & G & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & G & -2G+\rho\omega^2 & G & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & G & -2G+\rho\omega^2 & G & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & G & -2G+\rho\omega^2 & G \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$b = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{2.8}$$

In this case, the specified boundary conditions are that one end of the domain was restricted to give zero displacement and the other end was allowed to oscillate with a fixed amplitude of 1mm, giving $v_0 = 0$ and $v_m = 1$. This forced displacement oscillation, as mentioned above, is applied transversely to the direction of the 1D medium.

Equation (2.6) can then be solved for $\underline{v}$ to produce a numerical approximation to the solution $v(x)$ in Equation (2.3). Uniformly distributed noise as a percentage of the geometric mean of the displacement was then added to the simulated data as a means of testing the robustness of the inverse problem algorithm.

2.1.2 Non-Homogeneous Model

A more general formulation of Equation (2.2) is to allow regions of greater stiffness, or cancerous 'lumps,' within the domain. As shown in Equation (2.3) the 1D Navier equation given by Equation (2.2) was significantly simplified by considering the stiffness to be constant

over the whole domain. A similar idea can be used in this non-homogeneous case by assuming the Shear Modulus G in Equation (2.2) is piecewise constant. A stiffness element is then considered 'healthy' for a low stiffness value and 'cancerous' for a high stiffness value. Figure (2.1) shows an example of a piecewise constant Shear Modulus distribution where there is a cancerous lump in the domain.

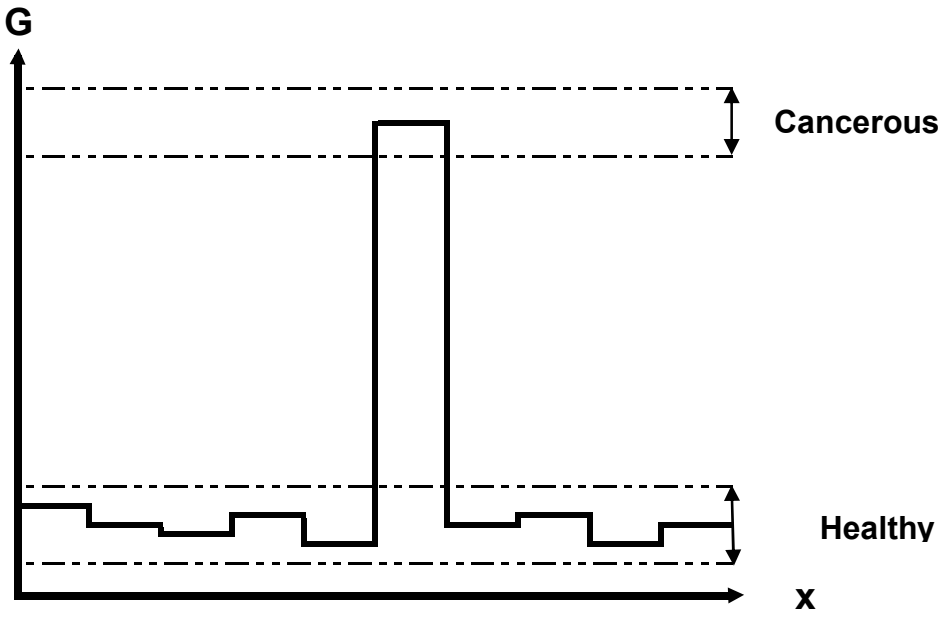

Figure 2.1 – The piecewise constant stiffness distribution of the forward simulation model for the non-homogeneous case

To ensure realistic displacements, elements are linked together by enforcing stress continuity at the boundary between two regions of differing stiffness. The 1D stress equation is defined:

$$\tau = G\gamma = G\frac{dv}{dx} = Gv_x \tag{2.9}$$

Where γ is the Shear Strain and τ is the Shear Stress with units of Nm^{-2}.

For two given elements with differing stiffness G_1 and G_2, continuity is enforced by setting:

$$\tau^- = \tau^+ \tag{2.10}$$

$$G_1 v_x^- = G_2 v_x^+ \tag{2.11}$$

The numerical finite difference approximations to τ^- and τ^+ are defined:

$$\tau^- = G_1 \frac{(3v_i - 4v_{i-1} + v_{i-2})}{h^2} + O(h^2) \tag{2.12}$$

$$\tau^+ \approx G_2 \frac{(-3v_i + 4v_{i+1} - v_{i+2})}{h^2} + O(h^2) \tag{2.13}$$

This gives the following numerical stress continuity condition:

$$G_1(3v_i - 4v_{i-1} + v_{i-2}) \approx G_2(-3v_i + 4v_{i+1} - v_{i+2}) \tag{2.14}$$

where:

G_1 = Shear Modulus (Nm^{-2}) to the left of the stiffness boundary

G_2 = Shear Modulus (Nm^{-2}) to the right of the stiffness boundary

v_x^- = Left Side Derivative of Displacement

v_x^+ = Right Side Derivative of Displacement

The reason for the notation v_x^- and v_x^+ is that the first derivative of displacement (v_x) is not continuous at the boundary between regions of differing stiffness. However the product of the Shear Modulus and the derivative v_x is continuous throughout the domain. Therefore the left hand and right hand derivatives, v_x^- and v_x^+, are using a backward difference and a forwards difference respectively. The magnitude of the error due to the finite difference approximations is kept consistent with previous approximations, in the order of h^2.

Equation (2.3) relating to the 1D Navier Equation is therefore not applied at the centre nodes separating two differing stiffness regions and is subsequently removed from the linear system of equations defined in Equations (2.6-2.8). The corresponding stress continuity condition given by (Equation 2.14) is applied instead. This gives a new linear system of equations:

$$A\underline{v} = b \tag{2.15}$$

Where A is an $m\,x\,m$ matrix similar to Equation (2.6) except in the region of the high stiffness 'lump' which is defined:

$$A = \frac{1}{h^2}\begin{bmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \dots & G_1 & C_1 & G_1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots \\ \dots & G_1 & -4G_1 & 3(G_1+G_2) & -4G_2 & G_2 & \dots & 0 & 0 & 0 & 0 & 0 & \dots \\ \dots & 0 & 0 & G_2 & C_2 & G_2 & \dots & 0 & 0 & 0 & 0 & 0 & \dots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \dots & 0 & 0 & 0 & 0 & 0 & \dots & G_2 & C_2 & G_2 & 0 & 0 & \dots \\ \dots & 0 & 0 & 0 & 0 & 0 & \dots & G_2 & -4G_2 & 3(G_2+G_3) & -4G_3 & G_3 & \dots \\ \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & G_3 & C_3 & G_3 & \dots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{2.16}$$

Where $C_i = -2G_i + \rho\omega^2$ and b is an $m\,x\,1$ vector of the form:

$$b = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{2.17}$$

The solution to Equation (2.15) will then approximate the solution to the non-homogenous 1D Navier's equation given by Equation (2.2) for the case of a piecewise constant Shear Modulus G given by Equation (2.2b). The solutions of Equation (2.6) and Equation (2.15) will converge to the true solution with reduction of the step size h.

2.2 One Dimensional Inverse Problem Solution Algorithms

Three different methods of integration were investigated when creating the one dimensional inverse problem solution algorithms. All methods assume that the stiffness values that are to be calculated are constant along 1cm long segments and the stiffness of a segment is not constrained

to the stiffnesses of neighbouring segments in any way. This representation of breast tissue stiffness distribution effectively limits the resolution of cancer detection to a 1cm long 'tumour'. However as cancerous tissue is significantly stiffer than healthy breast tissue, there is still a possibility of picking up smaller 'tumours' as the average stiffness in a 1cm segment containing cancer will still be greater than the surrounding healthy tissue. Furthermore, once a potential 'tumour' is discovered the tissue stiffness distribution could be refined for greater accuracy.

The first method investigated is a local integration method which calculates the value of the Shear Modulus at each segment using only displacement data from within that particular segment. This means that each stiffness value is calculated independently from the surrounding segments.

The second method involves single integration over the whole displacement dataset to formulate a linear system of equations involving the Shear Modulus of all segments. Thus the interactions between different segments and their effects on displacement is utilised as additional knowledge to create a more effective algorithm. For the single integration method, the fundamental equation is only integrated once and thus requires differentiation of the displacement data and hence is potentially more sensitive to noise.

The third method involves double integrating the differential equation of motion (Equation (2.3)). Thus it incorporates the global knowledge of the stiffness distribution as well as not requiring any differentiation. This results in a more robust method to the effects of noise.

2.2.1 Local Integration

For a constant Shear Modulus G, Equation (2.3) can be rewritten:

$$v_{xx}(x) = -Av \tag{2.18}$$

$$A = \frac{-\rho\omega^2}{G} \tag{2.19}$$

Integrating Equation (2.18) from x_1 to x gives:

$$v_x(x) - v_x(x_1) = -A\int_{x_1}^{x} v(x')dx' \tag{2.20}$$

Integrating Equation (2.20) from x_1 to x gives:

$$\begin{aligned} v(x) &= v(x_1) + v_x(x_1)(x - x_1) - A\int_{x_1}^{x}\int_{x_1}^{x}\left(v(x')dx'\right)dx \\ &= ax + b - A\int_{x_1}^{x}\int_{x_1}^{x}\left(v(x')dx'\right)dx \end{aligned} \tag{2.21}$$

$$a = v_x(x_1) \tag{2.22}$$

$$b = v(x_1) - v_x(x_1)x_1 \tag{2.23}$$

Note that a and b are treated as unknown parameters since they contain derivatives which due to noise may potentially corrupt the solution and hence affect the accuracy of the fitted parameter A.

Equation (2.21) shows that two integrations of Equation (2.18) eliminates derivative terms leaving three unknown coefficients a, b and A. Given for example 10 segments, defined by $\{[x_{i-1}, x_i], i = 1,...,10\}$ with 10 different Shear Modulus values G_i, $i = 1,...,10$, Equations (2.21)-(2.23) and (2.19) become respectively:

$$v(x) = a_i x + b_i - A_i\int_{x_1}^{x}\int_{x_1}^{x}\left(vdx'\right)dx,\ x \in [x_{i-1}, x_i],\ i = 1,...,10 \tag{2.24}$$

$$a_i = v_x(x_{i-1}) \tag{2.25}$$

$$b_i = v(x_{i-1}) - v_x(x_{i-1})x_{i-1} \tag{2.26}$$

$$A_i = \frac{-\rho\omega^2}{G_i} \tag{2.27}$$

By choosing $m > 3$ values of x in each interval $[x_{i-1}, x_i]$, $i = 1,...,10$, Equation (2.24) will produce $10m$ equations in 30 unknowns $\{a_i, b_i, A_i,\ i = 1,...,10\}$. Since $10m > 30$ this is an over-determined system which can be solved by linear least squares and will uniquely determine A_i and hence the stiffness distribution G_i, $i = 1,...,10$.

However note that Equations (2.24-2.27) are only defined locally on each segment, independent of all other segments. Thus the problem of determining the 10 A_i terms is split up into 10 separate problems. Also note that the integral sign is used for simplicity in notation. It should be taken to mean numerical integration in all instances.

2.2.2 Global Single Integration with Derivative Fitting

This second method looks to include the effects of the global motion dataset, as a change of the Shear Modulus or stiffness within a small segment of the system affects the entire motion dataset. Therefore making use of the global motion dataset would be beneficial in increasing the accuracy of the stiffness to be solved for. Integrating Equation (2.2) from 0 to x produces the following equation:

$$v_x(x)G(x) - v_x(0)G(0) = -\rho\omega^2 \int_0^x v(x)dx \tag{2.28}$$

For a given segment $x_{i-1} < x < x_i$ and a piecewise constant Shear Modulus function $G(x)$, shown in Figure (2.1), Equation (2.28) can be written in the form:

$$v_x(x)G_i - v_x(0)G_1 = -\rho\omega^2 \int_0^x v(x)dx,\ x \in [x_{i-1}, x_i] \tag{2.29}$$

Given $m \geq 2$ values of x in each segment an over-determined system of linear equations can be set up in terms of the unknowns $\{G_i,\ i = 1,...,10\}$. However this method requires the derivative of noisy displacement data. Thus a series of quadric polynomials were fitted in each segment to smooth out the noise and enable a reasonably accurate approximation to derivatives. This process is discussed in more detail in the results section of Chapter 5.

2.2.3 Global Double Integration

The third integration method also includes the effects of the global motion dataset but has the advantage of not requiring any derivatives to compute. Thus the method is more robust to the effects of noise. Integrating Equation (2.2) from 0 to x gives:

$$\begin{aligned} &\int_0^x G(x')v_{xx}(x')dx' = -\rho\omega^2 \int_0^x v(x')dx' \\ &G(x)v_x(x) - G(0)v_x(0) = -\rho\omega^2 \int_0^x v(x')dx' \end{aligned} \tag{2.30}$$

Integrating Equation (2.30) from 0 to x gives:

$$\int_0^x G(x')v_x(x')dx' - G(0)v_x(0)x = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \tag{2.31}$$

However for a constant piecewise function of Shear Modulus $G(x)$ defined:

$$G(x) = G_i,\ x_{i-1} \le x \le x_i,\ i = 1,...,10 \tag{2.32}$$

There is no simple immediate simplification of Equation (2.31). Thus to obtain the generalised formula across the whole domain, the formula for the first three stiffness segments are initially derived as follows:

Stiffness 1, $x \in [x_0, x_1]$

$$\begin{aligned} &G_1 \int_0^x v_x(x)dx - G_1 v_x(0)x = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \\ &G_1[v(x) - v(0)] - G_1 v_x(0)x = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \\ &G_1 v(x) - G_1 v(0) - G_1 v_x(0)x = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \end{aligned} \tag{2.33}$$

Thus for the first segment Equation (2.31) becomes:

$$G_1 v(x) + b_1 + ax = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \tag{2.34}$$

where:

$a = -G_1 v_x(0)$

$b_1 = -G_1 v(0)$

Stiffness 2, $x \in [x_1, x_2]$

$$\begin{aligned} & G_1 \int_0^{x1} v_x(x)dx + G_2 \int_{x1}^x v_x(x)dx - G_1 v_x(0)x = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \\ & G_1 [v(x_1) - v(0)] + G_2 [v(x) - v(x_1)] - G_1 v_x(0)x = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \\ & G_2 v(x) + (G_1 [v(x_1) - v(0)] - G_2 v(x_1)) - G_1 v_x(0)x = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \end{aligned} \tag{2.35}$$

Thus for the second segment Equation (2.31) becomes:

$$G_2 v(x) + b_2 + ax = -\rho\omega^2 \int_0^x \left(\int_0^x v(x')dx' \right) dx \tag{2.36}$$

where:

$a = -G_1 v_x(0)$

$b_2 = G_1 [v(x_1) - v(0)] - G_2 v(x_1)$

$\quad = -G_1 v(0) + v(x_1)(G_1 - G_2)$

Stiffness 3, $x \in [x_2, x_3]$

$$\begin{aligned}
&G_1\int_0^{x1} v_x(x)dx + G_2\int_{x1}^{x2} v_x(x)dx + G_3\int_{x2}^{x} v_x(x)dx - G_1 v_x(0)x \\
&\qquad = -\rho\omega^2\int_0^x\left(\int_0^x v(x')dx'\right)dx \\
&G_1[v(x_1)-v(0)] + G_2[v(x_2)-v(x_1)] + G_3[v(x)-v(x_2)] - G_1 v_x(0)x \\
&\qquad = -\rho\omega^2\int_0^x\left(\int_0^x v(x')dx'\right)dx \\
&G_3 v(x) + [G_1[v(x_1)-v(0)] + G_2[v(x_2)-v(x_1)] - G_3 v(x_2)] - G_1 v_x(0)x \\
&\qquad = -\rho\omega^2\int_0^x\left(\int_0^x v(x')dx'\right)dx
\end{aligned} \tag{2.37}$$

Thus for the third segment Equation (2.31) becomes:

$$G_3 v(x) + b_3 + ax = -\rho\omega^2\int_0^x\left(\int_0^x v(x')dx'\right)dx \tag{2.38}$$

where:

$$a = -G_1 v_x(0)$$

$$\begin{aligned}
b_2 &= G_1[v(x_1)-u(0)] + G_2[v(x_2)-u(x1)] - G_3 v(x_2) \\
&= -G_1 + v(x_1)(G_1 - G_2) + v(x_2)(G_2 - G_3)
\end{aligned}$$

Looking at Equations (2.34), (2.36) and (2.38) the general formula for the i^{th} segment can be inferred as follows:

$$G_i v(x) + b_i + ax = -\rho\omega^2\int_0^x\left(\int_0^x v(x')dx'\right)dx \tag{2.39}$$

where:

$$a = -G_1 v_x(0)$$

$$b_i = -G_i v(0) + \sum_{k=2}^{i} v(x_{k-1})[G_{k-1} - G_k]$$

By treating the global constant term a and telescoping terms b_i as extra unknown parameters, it avoids the possibility of error in the derivative $v_x(0)$ or $v(x_{k-1})$ terms corrupting the solution.

Choosing $m \geq 3$ values of x in each interval $[x_{i-1}, x_i]$, $i = 1,...,10$, Equation (2.39) will produce $10m$ equations in 21 unknowns $\{(G_i, b_i, i = 1,...,10), a\}$. Since $10m > 21$ the result is an over-determined system of linear equations that can be solved uniquely for the stiffness distribution $\{G_i, i = 1,...,10\}$ by linear least squares. For example, if $m = 3$ the system of equations can be written:

$$A\underline{\alpha} = b \tag{2.40}$$

$$\alpha = \{(G_i, b_i, i = 1,...,10), a\} \tag{2.41}$$

$$A = \begin{bmatrix} v(x_1^{(1)}) & 1 & 0 & 0 & \dots & 0 & 0 & x_1^{(1)} \\ v(x_1^{(2)}) & 1 & 0 & 0 & \dots & 0 & 0 & x_1^{(2)} \\ v(x_1^{(3)}) & 1 & 0 & 0 & \dots & 0 & 0 & x_1^{(3)} \\ 0 & 0 & v(x_2^{(1)}) & 1 & \dots & 0 & 0 & x_2^{(1)} \\ 0 & 0 & v(x_2^{(2)}) & 1 & \dots & 0 & 0 & x_2^{(2)} \\ 0 & 0 & v(x_2^{(3)}) & 1 & \dots & 0 & 0 & x_2^{(3)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & v(x_{10}^{(1)}) & 1 & x_{10}^{(1)} \\ 0 & 0 & 0 & 0 & \dots & v(x_{10}^{(2)}) & 1 & x_{10}^{(2)} \\ 0 & 0 & 0 & 0 & \dots & v(x_{10}^{(3)}) & 1 & x_{10}^{(3)} \end{bmatrix} \tag{2.42}$$

$$b = -\rho\omega^2 \begin{bmatrix} I(x_1^{(1)}) \\ I(x_1^{(2)}) \\ I(x_1^{(3)}) \\ \vdots \\ I(x_{10}^{(1)}) \\ I(x_{10}^{(2)}) \\ I(x_{10}^{(3)}) \end{bmatrix} \tag{2.43}$$

where:

$$I(x_k^{(i)}) = -\rho\omega^2 \int_0^{x_k^{(i)}} \left(\int_0^{x_k^{(i)}} v(x')dx' \right) dx$$

2.2.4 Mesh Refinement

The global double integral 1D inverse solution algorithm of Section 2.2.3 was further enhanced by incorporating a localised mesh refinement technique. This was performed by initially solving for the stiffness across 1cm intervals then using the resulting stiffness distribution to identify regions with higher stiffness, which may not have been aligned with the discretized grid. For the case of a stiffness region not aligned with the discretized grid or a tumour less than 1cm the calculated stiffness value may contain contributions from both healthy and cancerous tissue. Thus the net stiffness may be less than it should be to easily detect cancer. Therefore regions with greater stiffness than expected for healthy tissue are looked at more closely by a localized refinement of the mesh. These identified regions are each divided into two 0.5cm regions producing another mesh with greater resolution. The advantage of this refinement process is that only the likely cancerous regions are refined thus improving computational efficiency and accuracy. A new over-determined system of linear equations is then formed in a similar way to Section 2.2.3 but now with more unknown stiffness values to solve for. This technique allows identification of 1cm long tumours that do not neatly fit within the discretized domain and also enhances the ability to identify tumours down to a size of 0.5cm.

2.2.5 Numerical Integration

All the integral terms given in Sections 2.2.1-2.2.3 are numerical integrals. The single integrals in Section 2.2.2 are simply an application of the trapezium rule as shown:

$$\begin{aligned} E(x_j) &= \int_{x_0}^{x_j} v(x)dx \\ &= h\left(\frac{v(x_0)+v(x_j)}{2} + \sum_{i=1}^{j-1} v(x_i) \right) \end{aligned} \tag{2.44}$$

Where the notation in Equation (2.44) and (2.47)-(2.48) is described in Figure (2.3).

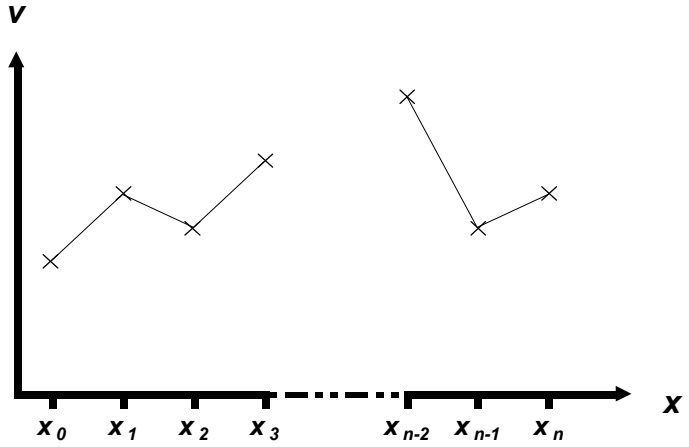

Figure 2.3 – Description of the notation used to define the motion dataset for the purposes of numerical integration

However, the inverse algorithms detailed in Sections 2.2.1 and 2.2.3 both require the use of numerical double integrals, which are detailed as follows:

$$F(x_j) = \int_{x_0}^{x_j}\left(\int_{x_0}^{x} v(x')dx'\right)dx \tag{2.45}$$

However it is not immediately clear how to formulate this in terms of sums. Therefore Equation (2.45) is reformulated as follows:

$$F(x_j) = \int_{x_0}^{x_j} f(x)dx \tag{2.46}$$

$$f(x_k) = \int_{x_0}^{x_k} v(x')dx' \tag{2.47}$$

Equations (2.46) and (2.48) are also evaluated using the trapezium rule. Equations (2.48) and (2.49) represent the trapezium rule formulation of the numerical double integral (Equation (2.45)) in terms of summation terms:

$$F(x_j) = h\left(\frac{f(x_0)+f(x_j)}{2} + \sum_{i=1}^{j-1}\left(f(x_i)\right)\right) \tag{2.48}$$

$$\begin{aligned} f(x_k) &= 0 && k = 0 \\ &= h\left(\frac{v(x_0) + v(x_k)}{2} + \sum_{i=1}^{k-1} \left(v(x_i) \right) \right) && k = 1,\ldots,n \end{aligned} \tag{2.49}$$

This numerical double integral function based upon the trapezium rule can easily be calculated using inbuilt functions in MATLAB. Both the intermediate function $f(x)$ and the final double integral function $F(x)$ are formulated using the cumulative trapezium function, 'cumtrapz'. Therefore Equations (2.48) and (2.49) can be represented in terms of MATLAB code as follows:

```
F=h^2*cumtrapz(cumtrapz(v));
```

3 Forward Simulation of Two Dimensional Motion Data

3.1 Homogeneous Forward Simulation

To obtain global motion data that can be used as input to test the inverse solution algorithm, a forward simulation algorithm was formulated. For proof of concept a simple square domain is used. However, the method can be readily extended to handle any curved domain by creating an appropriate mesh with, for example, triangular elements near the boundary and square elements inside.

The decision to use Finite Difference Approximations (FDAs) as the basis of the simulation model was made primarily due to the computational efficiency and ease of programming that this form of numerical approximation provides. The simple (square) geometry also negated the need to develop a more complicated finite element solver. Given their similarity to the fundamental DIET problem, the two dimensional plane strain, time harmonic approximations of Navier's Equations are used as the fundamental defining equations for the 2D model:

$$\begin{aligned} \frac{\partial}{\partial x}(Gu_x) + \frac{\partial}{\partial y}(Gu_y) + \frac{\partial}{\partial x}\left[(\lambda + G)(u_x + v_y)\right] &= -\rho\omega^2 u \\ \frac{\partial}{\partial x}(Gv_x) + \frac{\partial}{\partial y}(Gv_y) + \frac{\partial}{\partial y}\left[(\lambda + G)(u_x + v_y)\right] &= -\rho\omega^2 v \end{aligned} \tag{3.1}$$

where:

$$G = \frac{E}{2 + 2\nu} \tag{3.2}$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{3.3}$$

where:

u = Displacement in x direction (m)

v = Displacement in y direction (m)

E = Young's Modulus (Nm^{-2})

G = Shear Modulus (Nm^{-2})

ω = Harmonic Actuation Frequency ($rads^{-1}$)

ρ = Tissue Density (kgm^{-3})

ν = Poisson's Ratio

λ = Lamé's Constant

Initially, the model is assumed to have homogeneous material properties to provide a means of verifying against analytical results. It also provides a baseline for comparison when incorporating a region or 'lump' of greater stiffness that models cancerous tissue. In the homogeneous case, the Young's Modulus, E, is constant. Therefore, since the λ and G terms are linearly related to E in Equations (3.2)-(3.3), they are also constant and can be removed from the derivative terms in Equation (3.1) producing:

$$\begin{aligned}(\lambda+2G)u_{xx}+Gu_{yy}+(\lambda+G)v_{xy}&=-\rho\omega^2u\\(\lambda+2G)v_{yy}+Gv_{xx}+(\lambda+G)u_{xy}&=-\rho\omega^2v\end{aligned} \tag{3.4}$$

This approach also extends to the non-homogeneous 'lump' model as the stiffness distribution is either 'healthy', corresponding to low constant stiffness, or 'cancerous', corresponding to high constant stiffness. More specifically, non-homogeneous cases will assume element-wise constant material properties. The stiffness elements are then connected by applying the appropriate stress continuity terms at the boundary of regions of differing stiffness in a similar way to the 1D case in Equations (2.10) and (2.11).

Each of the derivative terms in Equation (3.4) are approximated by a discrete finite difference approximation defined:

$$u_{xx}\approx\frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{h^2} \tag{3.5}$$

$$u_{yy} \approx \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} \tag{3.6}$$

$$v_{xy} \approx \frac{v_{i-1,j-1} - v_{i-1,j+1} - v_{i+1,j-1} + v_{i+1,j+1}}{4h^2} \tag{3.7}$$

$$v_{xx} \approx \frac{v_{i-1,j} - 2v_{i,j} + v_{i+1,j}}{h^2} \tag{3.8}$$

$$v_{yy} \approx \frac{v_{i,j-1} - 2v_{i,j} + v_{i,j+1}}{h^2} \tag{3.9}$$

$$u_{xy} \approx \frac{u_{i-1,j-1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i+1,j+1}}{4h^2} \tag{3.10}$$

Where h is the (typically equal) distance assumed between discrete points in both the x and y direction. Note that different or varying values of h can be readily incorporated.

Due to computer memory constraints, sparse matrix algebra in MATLAB is used to form the system of equations and find the solution. This approach is efficient in this case because the equations relating to each node only contain contributions from the eight surrounding nodes and the node itself. Thus, for grid refinements where the system of equations is expanded, the maximum number of terms per equation does not exceed nine.

3.2 Application of Boundary Conditions

Three different boundary condition models were implemented. All three are shown schematically in Figure 3.1.

1. 'Box Shake' Model - Figure 3.1(a)

- The entire external boundary of the square domain is oscillated at known amplitude in the x direction and zero amplitude in the y direction.

2. 'Edge Effect' Model – Figure 3.1(b)

- The domain is restrained in both the x & y directions on the left hand edge.
- The right hand edge is constrained to oscillate at known amplitude in the x direction and zero amplitude in the y direction.
- The top and bottom horizontal edges of the domain are made 'free' edges, where the constraints enforced on these two edges are zero shear stress along the boundary and zero longitudinal stress normal to the boundary.

3. 'Phantom' Model – Figure 3.1(c)

- Model is similar to the breast phantoms used to simulate data for the DIET project [Peters et al. (2005)], except that shear excitation is used.
- A single edge of the domain is forced to oscillate. This forced harmonic displacement is parallel to the edge.
- The other three edges are 'free' with zero shear stress and zero normal longitudinal stress enforced along these boundaries of the domain.

The known peak to peak amplitude that was used for these boundary conditions was 1mm. This value has been chosen because it corresponds to the capabilities and requirements of the initial DIET prototype system [Peters et al. (2005)].

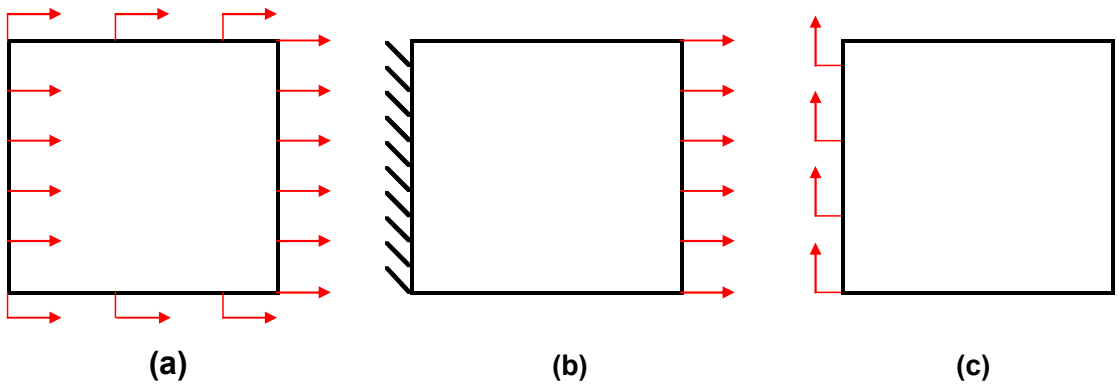

Figure 3.1 – Description of the three types of boundary condition systems implemented; (a) 'Box Shake' model, (b) 'Edge Effect' model and (c) 'Phantom' model.

Both the 'Edge Effect' and 'Phantom' models, shown in Figures 3.1(b) and 3.1(c), require zero stress boundary conditions to be imposed along the 'free' edges. For the 'Edge Effect' model of Figure 3.1(b) the shear stress τ_{xy} and the y direction longitudinal stress σ_y are used along the horizontal boundaries defined [Chou et al. (1967)]:

$$\tau_{xy} = G(u_y + v_x) \tag{3.11}$$

$$\sigma_y = (2G + \lambda)v_y + \lambda u_x \tag{3.12}$$

Finite differences are used to approximate Equations (3.11) and (3.12). Specifically, at the bottom horizontal boundary (B) forward differences are applied perpendicular to the boundary and central differences are applied parallel to the boundary:

$$\tau_{xy}^{(B)} \approx G\frac{(-3u_{i,j} + 4u_{i,j+1} - u_{i,j+2}) + (-v_{i-1,j} + v_{i+1,j})}{2h} = 0 \tag{3.13}$$

$$\sigma_y^{(B)} \approx (2G + \lambda)\frac{(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2})}{2h} + \lambda\frac{(-u_{i-1,j} + u_{i+1,j})}{2h} = 0 \tag{3.14}$$

Similarly, the stresses on the top horizontal boundary (T) are defined:

$$\tau_{xy} \approx G\frac{(3u_{i,j} - 4u_{i,j-1} + u_{i,j-2}) + (-v_{i-1,j} + v_{i+1,j})}{2h} = 0 \tag{3.15}$$

$$\sigma_y \approx (2G + \lambda)\frac{(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2})}{2h} + \lambda\frac{(-u_{i-1,j} + u_{i+1,j})}{2h} = 0 \tag{3.16}$$

The finite difference approximations of Equations (3.11) and (3.12) are all $O(h^2)$ and replace the finite difference derivations of Equation (3.4) at each horizontal boundary node.

Zero shear stress and zero normal longitudinal stress conditions are also enforced at the right vertical edge of the domain for the 'Phantom' model in Figure 3.1(c). The normal longitudinal stress in this case is the x direction longitudinal defined as follows:

$$\sigma_x = (2G + \lambda)u_x + \lambda v_y \tag{3.17}$$

The finite difference approximations to the longitudinal and shear stresses given by Equations (3.11) and (3.12) along the 'free' vertical boundary (V) are defined:

$$\sigma_x^{(V)} \approx (2G + \lambda)\frac{(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j})}{2h} + \lambda\frac{(-v_{i,j-1} + v_{i,j+1})}{2h} = 0 \tag{3.18}$$

$$\tau_{xy}^{(V)} \approx G\frac{(3v_{i,j} - 4v_{i-1,j} + v_{i-2,j}) + (-u_{i,j-1} + u_{i,j+1})}{2h} = 0 \tag{3.19}$$

The 'Phantom' model of Figure 3.1(c) also has 'free' corners, where the unrestrained vertical edge of the domain meets the two unrestrained horizontal edges. In this instance, there is no uniquely defined normal longitudinal stress that can be defined on the corner node. So the x direction longitudinal stress of Equation (3.17) was chosen as a zero stress condition, in addition to the shear stress of Equation (3.11). The finite difference formulas for σ_x and τ_{xy} are thus defined for the top right corner node (R) as follows:

$$\sigma_x^{(R)} \approx (2G + \lambda)\frac{(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j})}{2h} + \lambda\frac{(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2})}{2h} = 0 \tag{3.20}$$

$$\tau_{xy}^{(R)} \approx G\frac{(3v_{i,j} - 4v_{i-1,j} + v_{i-2,j}) + (3u_{i,j} - 4u_{i,j-1} + u_{i,j-2})}{2h} = 0 \tag{3.21}$$

The finite difference formulas for σ_x and τ_{xy} are defined for the bottom right corner node (B) as follows:

$$\sigma_x^{(B)} \approx (2G + \lambda)\frac{(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j})}{2h} + \lambda\frac{(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2})}{2h} = 0 \tag{3.22}$$

$$\tau_{xy}^{(B)} \approx G\frac{(3v_{i,j} - 4v_{i-1,j} + v_{i-2,j}) + (-3u_{i,j} + 4u_{i,j+1} - u_{i,j+2})}{2h} = 0 \tag{3.23}$$

After incorporating the boundary conditions of Equations (3.11)-(3.23) and combining with Equations (3.4)-(3.10) a linear system of equations can be formed in terms of the displacement amplitudes in both Cartesian directions u and v. For reasons of space the full system of equations for each case shown in Figures 3.1(a)-(c) is not given. The system of equations will have a unique solution for each given boundary condition provide the 'measured' data input to validate the inverse solution algorithms.

3.3 Verification of Homogeneous Model

The 2-D time harmonic, plane strain Navier's Equations given by Equation (3.4) has a unique analytical solution across the infinite domain defined:

$$\begin{aligned} u(x,y) = {} & C_1 \cos\left(\frac{\omega\sqrt{\rho}x}{\sqrt{G}}\right) + C_2 \sin\left(\frac{\omega\sqrt{\rho}x}{\sqrt{G}}\right) + C_3 \sin\left(\frac{\omega\sqrt{\rho}x}{\sqrt{\lambda+2G}}\right) + C_4 \cos\left(\frac{\omega\sqrt{\rho}x}{\sqrt{\lambda+2G}}\right) \\ & + C_5 \sin\left(\frac{\omega\sqrt{\rho}y}{\sqrt{G}}\right) + C_6 \cos\left(\frac{\omega\sqrt{\rho}y}{\sqrt{G}}\right) + C_7 \sin\left(\frac{\omega\sqrt{\rho}y}{\sqrt{\lambda+2G}}\right) + C_8 \cos\left(\frac{\omega\sqrt{\rho}x}{\sqrt{\lambda+2G}}\right) \end{aligned} \tag{3.24}$$

$$v(x,y) = \frac{-2}{\omega\sqrt{G\rho}(\lambda+2G)} \begin{pmatrix} \sqrt{G}\sqrt{\lambda+2G}\left(-\frac{C_7\rho\omega^2 x}{2} + \frac{C_{11}\lambda}{2} + C_{11}G\right)\cos\left(\frac{\omega\sqrt{\rho}y}{\sqrt{\lambda+2G}}\right) \\ -\sqrt{G}\sqrt{\lambda+2G}\left(-\frac{C_8\rho\omega^2 x}{2} + \frac{C_{12}\lambda}{2} + C_{12}G\right)\sin\left(\frac{\omega\sqrt{\rho}y}{\sqrt{\lambda+2G}}\right) \\ -\omega\left(\left(G+\frac{\lambda}{2}\right)\left(C_9\sqrt{G\rho} + C_1\omega\rho y\right)\right)\sin\left(\frac{\omega\sqrt{\rho}x}{\sqrt{G}}\right) \\ -\omega\left(\left(G+\frac{\lambda}{2}\right)\left(C_{10}\sqrt{G\rho} - C_2\omega\rho y\right)\right)\cos\left(\frac{\omega\sqrt{\rho}x}{\sqrt{G}}\right) \end{pmatrix} \tag{3.25}$$

By choosing random values of the initial conditions C_i and evaluating the solution of Equations (3.24) and (3.25) on the boundary of the square domain of Figure 3.1, the homogeneous finite difference solution of Equation (3.4) can be verified for accuracy. However, to reproduce the solution of Equations (3.24) and (3.25) numerically it is necessary to slightly change the boundary conditions in the 'Box Shake' model of Figure 3.1(a).

Specifically, the boundary conditions of constant amplitude along each edge of the domain in Figure 3.1(a) are replaced by the values of the known analytical solution which vary continuously along the edges. Thus, the numerical solution should closely approximate the analytical solution within the square sub-domain. This approach provides an effective means for debugging the numerical code because appropriate derivatives could be compared between analytical and numerical models, along with the resulting forward simulated displacement values. It also enables a check of the overall numerical stability and accuracy of the forward simulation software and methods.

3.4 Non-Homogeneous Model

The next step in the formation of a forward simulation algorithm for this study is to accommodate regions of greater stiffness, or cancerous 'lumps,' within the domain. By considering stiffness to be constant across elements, the fundamental differential equations of Equation (3.1) can be simplified considerably to the form of Equation (3.4). For this new non-homogeneous model, this assumption will be maintained, except that the stiffness will now be piecewise constant. In a similar way to the 1D case of Equations (2.10) and (2.11), the different elements are linked via a stress continuity condition. For simplicity the 'lump' is assumed to have a rectangular geometry.

To enforce the stress continuity between regions of differing stiffness, the stress either side of the boundary is approximated numerically. This enforcement is done using finite difference approximations with $O(h^2)$ error similar to Equations (2.12)-(2.14). For the horizontal and vertical boundaries, both normal longitudinal and shear stress are used to enforce the stress continuity. For the vertical boundary, the normal longitudinal stress is represented by the x direction longitudinal stress given in Equation (3.17). For the horizontal boundary, the normal longitudinal stress is represented by the y direction longitudinal stress given in Equation (3.12). The shear stress is given by Equation (3.11).

Figure 3.2(a) illustrates the vertical boundary of the stiffness discontinuity, where the subscripts (L) and (R) denote the elements to the left and right of the boundary respectively. Figure 3.2(b)

illustrates the horizontal boundary where the subscripts (A) and (B) denote the elements above and below the boundary respectively.

The finite difference equations are defined as follows:

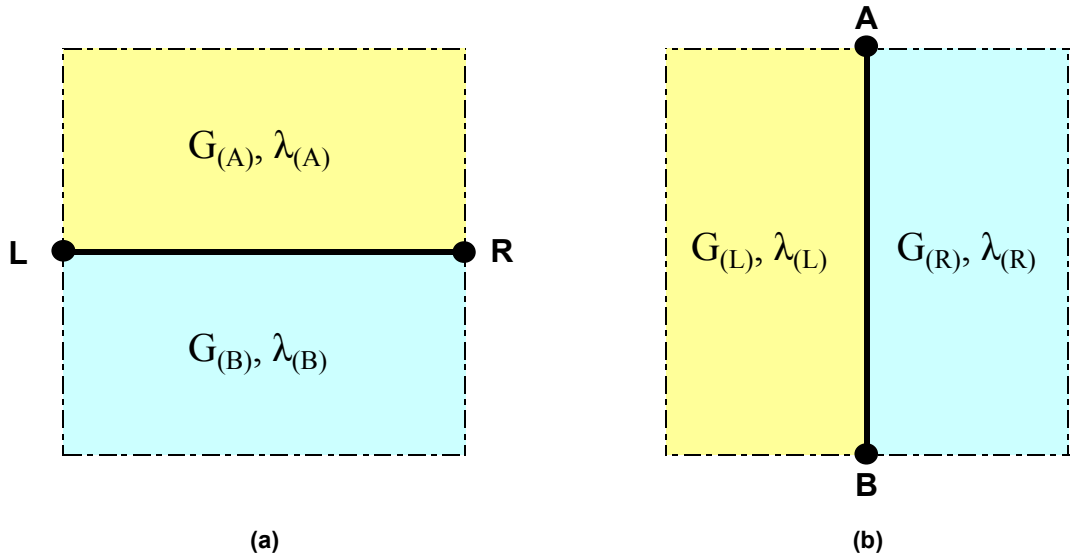

Figure 3.2 – Notation used to describe the Vertical Stiffness Boundary (a) and the Horizontal Stiffness Boundary (b)

Vertical Boundary

$$\begin{aligned}&\sigma_x^- = \sigma_x^+ \\ &(2G_{(L)} + \lambda_{(L)})(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}) + \lambda_{(L)}(-v_{i,j-1} + v_{i,j+1}) \\ &\qquad = (2G_{(R)} + \lambda_{(R)})(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}) + \lambda_{(R)}(-v_{i,j-1} + v_{i,j+1})\end{aligned} \tag{3.26}$$

$$\begin{aligned}&\tau_{xy}^- = \tau_{xy}^+ \\ &G_{(L)}(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j} + -v_{i,j-1} + v_{i,j+1}) \\ &\qquad = G_{(R)}(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j} + -v_{i,j-1} + v_{i,j+1})\end{aligned} \tag{3.27}$$

Horizontal Boundary

$$\begin{aligned}&\sigma_y^- = \sigma_y^+ \\ &(2G_{(B)} + \lambda_{(B)})(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}) + \lambda_{(B)}(-v_{i-1,j} + v_{i+1,j}) \\ &\qquad = (2G_{(A)} + \lambda_{(A)})(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2}) + \lambda_{(A)}(-v_{i-1,j} + v_{i+1,j})\end{aligned} \tag{3.28}$$

$$\begin{aligned}&\tau_{xy}^{-} = \tau_{xy}^{+}\\&G_{(B)}(3u_{i,j} - 4u_{i,j-1} + u_{i,j-2} + -v_{i-1,j} + v_{i+1,j})\\&\qquad = G_{(A)}(-3u_{i,j} + 4u_{i,j+1} - u_{i,j+2} + -v_{i-1,j} + v_{i+1,j})\end{aligned} \tag{3.29}$$

The extreme corner nodes on the boundary of two elements with differing stiffnesses, denoted as 'A' and 'B' in Figure 3.2(a) and 'L' and 'R' in Figure 3.2(b), require additional equations to represent stress continuity due to their geometric position. For the cancerous 'lump' geometry there are four corner nodes, as shown in Figure 3.4.

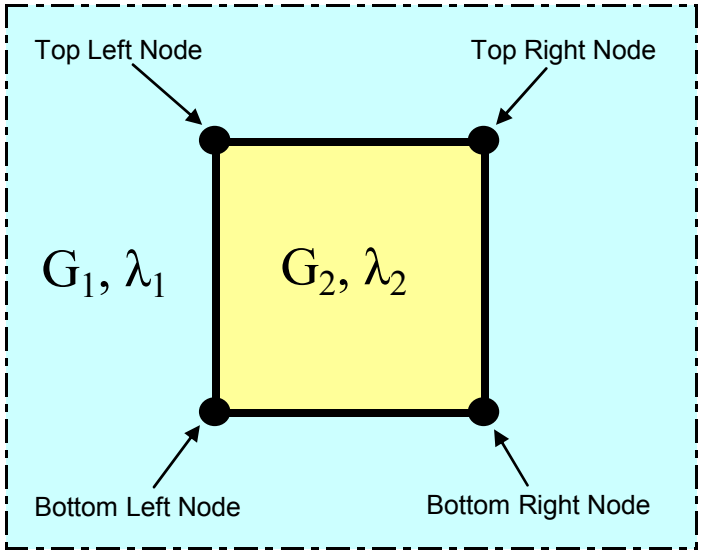

Figure 3.4 – Notation used to Describe the Extreme Nodes along Stiffness Boundary

For these nodes, the x direction and y direction longitudinal stress equations, given by Equations (3.17) and (3.12), are used to enforce stress continuity, since each stress equation is equally valid in this case. The finite difference formulations for the four cases are defined:

Top Left Node

$$\begin{aligned}&\sigma_x^{(1)} = \sigma_x^{(2)}\\&(2G_1 + \lambda_1)(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}) + \lambda_1(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2})\\&\qquad = (2G_2 + \lambda_2)(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}) + \lambda_2(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2})\end{aligned} \tag{3.30}$$

$$\sigma_y^{(1)} = \sigma_y^{(2)}$$
$$(2G_1 + \lambda_1)(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2}) + \lambda_1(3u_{i,j} - 4u_{i-1,j} - u_{i-2,j}) \tag{3.31}$$
$$= (2G_2 + \lambda_2)(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}) + \lambda_2(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j})$$

Top Right Node

$$\sigma_x^{(1)} = \sigma_x^{(2)}$$
$$(2G_1 + \lambda_1)(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}) + \lambda_1(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2}) \tag{3.32}$$
$$= (2G_2 + \lambda_2)(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}) + \lambda_2(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2})$$

$$\sigma_y^{(1)} = \sigma_y^{(2)}$$
$$(2G_1 + \lambda_1)(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2}) + \lambda_1(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}) \tag{3.33}$$
$$= (2G_2 + \lambda_2)(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}) + \lambda_2(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j})$$

Bottom Left Node

$$\sigma_x^{(1)} = \sigma_x^{(2)}$$
$$(2G_1 + \lambda_1)(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}) + \lambda_1(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}) \tag{3.34}$$
$$= (2G_2 + \lambda_2)(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}) + \lambda_2(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2})$$

$$\sigma_y^{(1)} = \sigma_y^{(2)}$$
$$(2G_1 + \lambda_1)(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}) + \lambda_1(3u_{i,j} - 4u_{i-1,j} - u_{i-2,j}) \tag{3.35}$$
$$= (2G_2 + \lambda_2)(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2}) + \lambda_2(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j})$$

Bottom Right Node

$$\sigma_x^{(1)} = \sigma_x^{(2)}$$
$$(2G_1 + \lambda_1)(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}) + \lambda_1(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}) \tag{3.36}$$
$$= (2G_2 + \lambda_2)(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j}) + \lambda_2(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2})$$

$$\sigma_y^{(1)} = \sigma_y^{(2)}$$
$$(2G_1 + \lambda_1)(3v_{i,j} - 4v_{i,j-1} + v_{i,j-2}) + \lambda_1(-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}) \tag{3.37}$$
$$= (2G_2 + \lambda_2)(-3v_{i,j} + 4v_{i,j+1} - v_{i,j+2}) + \lambda_2(3u_{i,j} - 4u_{i-1,j} + u_{i-2,j})$$

3.5 Summary

Combining Equations (3.4)-(3.23) forms a linear system of equations that can be uniquely solved to determine the displacement response. The displacement response is equivalent to the numerical solution of Equations (3.1)-(3.3) with stepwise constant stiffnesses and boundary conditions given by Figures 3.1(a)-(c). Equations (3.26)-(3.37) describe the stress continuity conditions linking elements with different stiffnesses. This allows a 'lump' to be placed in the geometry. This forward solution of a non-homogeneous system is the basis for testing the integral-based inverse algorithms using random added noise to account for measurement and/or model error.

4 Two Dimensional Inverse Problem Solutions

The inverse algorithms developed in Section 2.2 for the 1D Navier Equation formulate equations in terms of integrals of measured displacement amplitudes of sinusoidally actuated elastic material to identify the unknown Young's Modulus distribution of the material. This idea is extended for the 2D case. As with the 1D case, the algorithm is initially formulated for the homogeneous case. The non-homogeneous case is then treated as piecewise homogeneous.

The formulation of the inverse algorithms involves integrating the two dimensional Navier's Equations.

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} = -\rho\omega^2 u \tag{4.1}$$

$$\frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} = -\rho\omega^2 v \tag{4.2}$$

where:

$$\sigma_x = (\lambda + 2G)u_x + \lambda v_y \tag{4.3}$$

$$\sigma_y = (\lambda + 2G)v_y + \lambda u_x \tag{4.4}$$

$$\tau_{xy} = G(v_x + u_y) \tag{4.5}$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{4.6}$$

$$G = \frac{E}{2+2\nu} \tag{4.7}$$

where:

u = x direction displacement (m)

v = y direction displacement (m)

σ_x = Longitudinal Stress in the x direction (Pa)

σ_y = Longitudinal Stress in the y direction (Pa)

τ_{xy} = Shear Stress (Pa)

G = Shear Modulus (Pa)

λ = Lamé's Constant

E = Young's Modulus (Pa)

ν = Poisson's Ratio

ρ = Tissue Density (kgm^{-3})

ω = Harmonic Actuation Frequency ($rads^{-1}$)

Geometrically, for this 2D case, the single integral used in Equation (2.30) for the 1D case is replaced by an area or double integral, along the Cartesian coordinates x and y. Therefore, this double area integral represents a quadruple integral of Equations (4.1) and (4.2).

Choosing the same type of limits of integration as in the 1D case leads to first order derivative terms on the boundary of the rectangular region of interest. Given noisy displacement measurements, this derivative can corrupt the solution. Therefore, these integration limits must be chosen carefully.

4.1 2D Homogeneous Inverse Algorithm

The fundamental (assumed) geometry and coordinate system nomenclature is shown in Figure 4.1. It is applied for all 2D homogeneous inverse algorithms and is extended to the non-homogeneous.

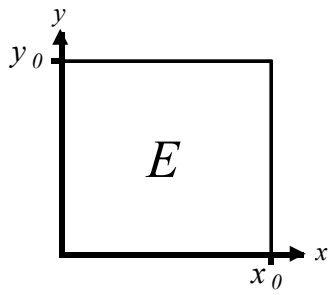

Figure 4.1 – Description of the fundamental geometry and coordinate system nomenclature for the 2D homogeneous inverse algorithms

4.1.1 Initial Inverse Algorithm

Initially, Equations (4.1) and (4.2) are integrated from a fixed base point at the coordinate origin, assuming it represents a 'corner' where all dimensions (x, y) are in the positive valued quadrant.

$$\begin{aligned}&\int_0^{y_0}\int_0^{x_0}\int_0^{y}\int_0^{x}\left(\frac{\partial\sigma_x(x',y')}{\partial x}+\frac{\partial\tau_{xy}(x',y')}{\partial y}\right)dx'dy'dxdy\\&=\int_0^{y_0}\int_0^{x_0}\int_0^{y}\int_0^{x}\left(-\rho\omega^2u(x',y')\right)dx'dy'dxdy\end{aligned}\tag{4.8}$$

$$\begin{aligned}&\int_0^{y_0}\int_0^{x_0}\int_0^{y}\int_0^{x}\left(\frac{\partial\sigma_y(x',y')}{\partial y}+\frac{\partial\tau_{xy}(x',y')}{\partial x}\right)dx'dy'dxdy\\&=\int_0^{y_0}\int_0^{x_0}\int_0^{y}\int_0^{x}\left(-\rho\omega^2v(x',y')\right)dx'dy'dxdy\end{aligned}\tag{4.9}$$

Geometrically, Equations (4.8) and (4.9) can be written in the following form:

$$\begin{aligned}&\int_{Ax_0y_0}\left[\int_{Axy}\left(\frac{\partial\sigma_x(x',y')}{\partial x}+\frac{\partial\tau_{xy}(x',y')}{\partial y}\right)dA_{x'y'}\right]dA_{xy}\\&=\int_{Ax_0y_0}\left[\int_{Axy}\left(-\rho\omega^2u(x',y')\right)dA_{x'y'}\right]dA_{xy}\end{aligned}\tag{4.10}$$

$$\begin{aligned}\int_{Ax_0y_0}\left[\int_{Axy}\left(\frac{\partial\sigma_y(x',y')}{\partial y}+\frac{\partial\tau_{xy}(x',y')}{\partial x}\right)dA_{x'y'}\right]dA_{xy}\\ =\int_{Ax_0y_0}\left[\int_{Axy}\left(-\rho\omega^2 v(x',y')\right)dA_{x'y'}\right]dA_{xy}\end{aligned} \tag{4.11}$$

Where A_{xy} is the rectangular region defined by $\{0 \le x' \le x,\ 0 \le y' \le y\}$ and $dA_{xy} = dxdy$.

The reason for integrating twice over each coordinate x and y is to remove the second order derivatives of u and v that arise from Equations (4.1)-(4.5) and are sensitive to noise. To simplify the formulation of Equations (4.8) and (4.9), or equivalently Equations (4.10) and (4.11), each term is integrated separately and the orders of integration interchanged as required. The primes on the coordinates of x' and y' are dropped for simplicity of notation for all integrals in the remainder of this chapter. After utilising the formulas for σ_x, σ_y and τ_{xy} given by Equations (4.3)-(4.5), the results of the integration of Equation (4.10) and (4.11) are as follows:

$$\begin{aligned}\int_{Ax_0y_0}\int_{Axy}\left(\frac{\partial\sigma_x}{\partial x}\right)dA_{xy}dA_{xy} &= \int_0^{y_0}\int_0^{x_0}\int_0^{y}\int_0^{x}\left(\frac{\partial\sigma_x}{\partial x}\right)dxdydxdy\\ &= \int_0^{y_0}\int_0^{x_0}\int_0^{y}\left(\sigma_x(x,y)-\sigma_x(0,y)\right)dydxdy\\ &= \int_0^{y_0}\int_0^{x_0}\int_0^{y}\lambda\left(v_y(x,y)-v_y(0,y)\right)dydxdy\\ &\quad+\int_0^{y_0}\int_0^{y}\int_0^{x_0}(2G+\lambda)\left(u_x(x,y)-u_x(0,y)\right)dxdydy\\ &= \lambda\int_0^{y_0}\int_0^{x_0}\left(v(x,y)-v(x,0)-v(0,y)+v(0,0)\right)dxdy\\ &\quad+(2G+\lambda)\int_0^{y_0}\int_0^{y}\left(u(x_0,y)-u(0,y)-x_0u(0,y)\right)dydy\end{aligned} \tag{4.12}$$

$$\begin{aligned}
\int_{Ax_0y_0}\int_{Axy}\left(\frac{\partial \tau_{xy}}{\partial y}\right)dA_{xy}\,dA_{xy} &= \int_0^{y_0}\int_0^{x_0}\int_0^{x}\int_0^{y}\left(\frac{\partial \tau_{xy}}{\partial y}\right)dydxdxdy \\
&= \int_0^{y_0}\int_0^{x_0}\int_0^{x}\left(\tau_{xy}(x,y)-\tau_{xy}(x,0)\right)dxdxdy \\
&= \int_0^{y_0}\int_0^{x_0}\int_0^{x} G\left(v_x(x,y)-v_x(x,0)\right)dxdxdy \\
&\quad +\int_0^{x_0}\int_0^{x}\int_0^{y_0} G\left(u_y(x,y)-u_y(x,0)\right)dydxdx \\
&= G\int_0^{y_0}\int_0^{x_0}\left(v(x,y)-v(0,y)-v(x,0)+v(0,0)\right)dxdy \\
&\quad +G\int_0^{x_0}\int_0^{x}\left(u(x,y_0)-u(x,0)-y_0u_y(x,0)\right)dxdx
\end{aligned} \tag{4.13}$$

Note that the constants G and λ have been pulled out of the resulting integrals.

Adding the results of Equations (4.12) and (4.13) and using the formulas for λ and G given by Equations (4.6) and (4.7), Equation (4.8) can be written in the form:

$$\left(f_v + f_u\right)E = \int_0^{y_0}\int_0^{x_0}\int_0^{y}\int_0^{x}\left(-\rho\omega^2 u\right)dxdydxdy \tag{4.14}$$

where f_u and f_v are integral functions of (measured) displacements u and v, defined:

$$\begin{aligned}
f_v &= a_1\int_0^{y_0}\int_0^{x_0}\left(v(x,y)-v(x,0)-v(0,y)+v(0,0)\right)dxdy \\
&= a_1\left(\int_0^{y_0}\int_0^{x_0}v(x,y)dxdy - y_0\int_0^{x_0}v(x,0)dx - x_0\int_0^{y_0}v(0,y)dy + x_0y_0v(0,0)\right)
\end{aligned} \tag{4.15}$$

$$\begin{aligned}
f_u &= a_2\int_0^{y_0}\int_0^{y}\left(u(x_0,y)-u(0,y)-x_0u_x(0,y)\right)dydy \\
&\quad +a_3\int_0^{x_0}\int_0^{x}\left(u(x,y_0)-u(x,0)-y_0u_y(x,0)\right)dxdx
\end{aligned} \tag{4.16}$$

However, it is important to note that Equation (4.16) still requires a derivative of the measured displacement u, as u_x.

where:

$$a_1 = \frac{\lambda+G}{E} = \frac{\nu}{(1+\nu)(1-2\nu)} + \frac{1}{2(1+\nu)} \tag{4.17}$$

$$a_2 = \frac{\lambda + 2G}{E} = \frac{\nu}{(1+\nu)(1-2\nu)} + \frac{1}{1+\nu} \tag{4.18}$$

$$a_3 = \frac{G}{E} = \frac{1}{2(1+\nu)} \tag{4.19}$$

The formulation of the terms that relate to the integration of the second Navier's equation, Equation (4.9), are similar to those in Equation (4.8) and therefore only the end result is shown for simplicity:

$$(g_u + g_v)E = \int_0^{y_0}\int_0^{x_0}\int_0^{y}\int_0^{x}\left(-\rho\omega^2 v\right)dxdydxdy \tag{4.20}$$

where:

$$\begin{aligned} g_u &= a_1\int_0^{y_0}\int_0^{x_0}\left(u(x,y) - u(x,0) - u(0,y) + u(0,0)\right)dxdy \\ &= a_1\left(\int_0^{y_0}\int_0^{x_0} u(x,y)dxdy - y_0\int_0^{x_0} u(x,0)dx - x_0\int_0^{y_0} u(0,y)dy + x_0 y_0 u(0,0)\right) \end{aligned} \tag{4.21}$$

$$\begin{aligned} g_v = a_3\int_0^{y_0}\int_0^{y}\left(v(x_0,y) - v(0,y) - x_0 v_x(0,y)\right)dydy \\ +a_2\int_0^{x_0}\int_0^{x}\left(v(x,y_0) - v(x,0) - y_0 v_y(x,0)\right)dxdx \end{aligned} \tag{4.22}$$

where a_1, a_2 and a_3 are also defined in Equations (4.17)-(4.19).

However, note that Equation (4.22) requires a derivative of the measured displacement v, as v_y.

Equations (4.14) and (4.20) can be used to form an over-determined system of equations by choosing various values of (x_0, y_0) and varying the position of the origin. The details are not shown here, but it is similar to the process that is presented in detail for the non-homogeneous case later in this chapter.

4.1.2 Centred Base Point Inverse Algorithm

The integral formulation of Equations (4.8) and (4.9) given by Equations (4.14) and (4.20) involves computing derivatives of u and v along the boundary, which is difficult using noisy data. The appearance of these derivatives in Equations (4.14) and (4.20) is directly related to the choice of integration limits. In this section, a new integral formulation is presented involving the innovative use of integration limits to eliminate all derivative terms in the resulting equations for the inverse problem.

Consider the following integration of Equation (4.1):

$$\begin{aligned}&\int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(\frac{\partial\sigma_x}{\partial x}+\frac{\partial\tau_{xy}}{\partial y}\right)dxdxdydy\\&\quad=\int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(-\rho\omega^2 u\right)dxdxdydy\end{aligned}\tag{4.23}$$

$$\begin{aligned}&\int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(\frac{\partial\sigma_y}{\partial y}+\frac{\partial\tau_{xy}}{\partial x}\right)dxdxdydy\\&\quad=\int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(-\rho\omega^2 v\right)dxdxdydy\end{aligned}\tag{4.24}$$

With the appropriate change in the orders of integration the first derivative term of Equations (4.23) can be rewritten as follows:

$$\int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(\frac{\partial\sigma_x}{\partial x}\right)dxdxdydy=\int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\left(\sigma_x(x_0+x,y)-\sigma_x(x_0-x,y)\right)dxdydy \tag{4.25}$$

Substituting $z=x_0+x$ and $z=x_0-x$ yields a right hand side:

$$=\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(\int_{x_0}^{2x_0}\sigma_x(x,y)dx-\int_0^{x_0}\sigma_x(x,y)dx\right)dydy \tag{4.25b}$$

Substituting Equation (4.3) for $\sigma_x(x,y)$:

$$\begin{aligned} &= \int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(\int_{x_0}^{2x_0}(\lambda+2G)u_x(x,y)dx - \int_0^{x_0}(\lambda+2G)u_x(x,y)dx\right)dydy \\ &\quad + \int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(\int_{x_0}^{2x_0}\lambda v_y(x,y)dx - \int_0^{x_0}\lambda v_y(x,y)dx\right)dydy \end{aligned} \tag{4.25c}$$

Changing limits in $v_y(x,y)$ integrand:

$$\begin{aligned} &= \int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(\int_{x_0}^{2x_0}(\lambda+2G)u_x(x,y)dx - \int_0^{x_0}(\lambda+2G)u_x(x,y)dx\right)dydy \\ &\quad + \int_{x_0}^{2x_0}\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(\lambda v_y(x,y)\right)dydydx \\ &\quad + \int_0^{x_0}\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(\lambda v_y(x,y)\right)dydydx \end{aligned} \tag{4.25d}$$

Integrating u_x with respect to x and v_y with respect to y:

$$\begin{aligned} &= (\lambda+2G)\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(u(2x_0,y) - 2u(x_0,y) + u(0,y)\right)dydy \\ &\quad + \lambda\int_{x_0}^{2x_0}\int_0^{y_0}\left(v(x,y_0+y) - v(x,y_0-y)\right)dydx \\ &\quad - \lambda\int_0^{x_0}\int_0^{y_0}\left(v(x,y_0+y) - v(x,y_0-y)\right)dydx \end{aligned} \tag{4.25e}$$

Substituting $z = x_0 + x$ and $z = x_0 - x$ and simplifying:

$$\begin{aligned} &= (\lambda+2G)\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(u(2x_0,y) - 2u(x_0,y) + u(0,y)\right)dydy \\ &\quad + \lambda\int_{x_0}^{2x_0}\left(\int_{y_0}^{2y_0}v(x,y)dy - \int_0^{y_0}v(x,y)dy\right)dx \\ &\quad - \lambda\int_0^{x_0}\left(\int_{y_0}^{2y_0}v(x,y)dy - \int_0^{y_0}v(x,y)dy\right)dx \end{aligned} \tag{4.25f}$$

Similarly, the second term in Equation (4.23) can be rewritten:

$$\int_0^{x_0}\int_{x_0-x}^{x_0+x}\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(\frac{\partial\tau_{xy}}{\partial y}\right)dydydxdx = \int_0^{x_0}\int_{x_0-x}^{x_0+x}\int_0^{y_0}\left(\tau_{xy}(x,y_0+y) - \tau_{xy}(x,y_0-y)\right)dydxdx \tag{4.26}$$

Substituting $z = x_0 + x$ and $z = x_0 - x$ yields a right hand side:

$$= \int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(\int_{y_0}^{2y_0}\tau_{xy}(x,y)dy - \int_0^{y_0}\tau_{xy}(x,y)dy\right)dxdx \tag{4.26b}$$

Substituting Equation (4.5) for $\tau_{xy}(x,y)$:

$$\begin{aligned} &= \int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(\int_{y_0}^{2y_0} Gu_y(x,y)dy - \int_0^{y_0} Gu_y(x,y)dy\right)dxdx \\ &\quad + \int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(\int_{y_0}^{2y_0} Gv_x(x,y)dy - \int_0^{y_0} Gv_x(x,y)dy\right)dxdx \end{aligned} \tag{4.26c}$$

Changing limits in $v_y(x,y)$ integrand:

$$\begin{aligned} &= \int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(\int_{y_0}^{2y_0} Gu_y(x,y)dy - \int_0^{y_0} Gu_y(x,y)dy\right)dxdx \\ &\quad + \int_{y_0}^{2y_0}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(Gv_x(x,y)\right)dx\,dxdy + \int_0^{y_0}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(Gv_x(x,y)\right)dx\,dxdy \end{aligned} \tag{4.26d}$$

Integrating u_y with respect to y and v_x with respect to x:

$$\begin{aligned} &= G\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(u(x,2y_0) - 2u(x,y_0) + u(x,0)\right)dxdx \\ &\quad + G\int_{y_0}^{2y_0}\int_0^{x_0}\left(v(x_0+x,y) - v(x_0-x,y)\right)dxdy \\ &\quad - G\int_0^{y_0}\int_0^{x_0}\left(v(x_0+x,y) - v(x_0-x,y)\right)dxdy \end{aligned} \tag{4.26e}$$

Substituting $z = x_0 + x$ and $z = x_0 - x$ and simplifying:

$$\begin{aligned} &= G\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(u(x,2y_0) - 2u(x,y_0) + u(x,0)\right)dxdx \\ &\quad + G\int_{y_0}^{2y_0}\left(\int_{x_0}^{2x_0} v(x,y)dx - \int_0^{x_0} v(x,y)dx\right)dy \\ &\quad - G\int_0^{y_0}\left(\int_{x_0}^{2x_0} v(x,y)dx - \int_0^{x_0} v(x,y)dx\right)dy \end{aligned} \tag{4.26f}$$

Note that Equations (4.25f) and (4.26f) do not have any derivatives of u and v in the results, having eliminated them by the choice of integration limits

Adding Equations (4.25f) and (4.26f) and using the formulas for λ and G given by Equations (4.17)-(4.19), Equation (4.23) can be written without derivatives of u or v.

$$\left(\overline{f}_v + \overline{f}_u\right)E = \int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(-\rho\omega^2 u\right)dxdxdydy \tag{4.27}$$

where:

$$\overline{f_v} = a_1\left(\int_{y_0}^{2y_0}\int_{x_0}^{2x_0} v(x,y)dxdy - \int_{y_0}^{2y_0}\int_{0}^{x_0} v(x,y)dxdy - \int_{0}^{y_0}\int_{x_0}^{2x_0} v(x,y)dxdy + \int_{0}^{y_0}\int_{0}^{x_0} v(x,y)dxdy\right) \quad (4.28)$$

$$\begin{aligned}\overline{f_u} = {} & a_2\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(u(2x_0,y) - 2u(x_0,y) + u(0,y)\right)dydy \\ & + a_3\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(u(x,2y_0) - 2u(x,y_0) + u(x,0)\right)dxdx\end{aligned} \quad (4.29)$$

The formulation of the terms that relate to the integration of Equation (4.24) are similar to those in Equation (4.23) and therefore only the end result is shown for simplicity.

$$\left(\overline{g}_u + \overline{g}_v\right)E = \int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(-\rho\omega^2 v\right)dxdxdydy \quad (4.30)$$

where:

$$\overline{g_u} = a_1\left(\int_{y_0}^{2y_0}\int_{x_0}^{2x_0} u(x,y)dxdy - \int_{y_0}^{2y_0}\int_{0}^{x_0} u(x,y)dxdy - \int_{0}^{y_0}\int_{x_0}^{2x_0} u(x,y)dxdy + \int_{0}^{y_0}\int_{0}^{x_0} u(x,y)dxdy\right) \quad (4.31)$$

$$\begin{aligned}\overline{g_v} = {} & a_2\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(v(x,2y_0) - 2v(x,y_0) + v(x,0)\right)dxdx \\ & + a_3\int_0^{y_0}\int_{y_0-y}^{y_0+y}\left(v(2x_0,y) - 2v(x_0,y) + v(0,y)\right)dydy\end{aligned} \quad (4.32)$$

Similar to Equations (4.14) and (4.20) in the previous section, Equations (4.27) and (4.30) can be used to form an over-determined system of equations by choosing various limits of (x_0, y_0) and varying the position of the origin. The details are not shown here, but it is similar to the process that is presented in detail for the non-homogeneous case later in this chapter.

4.2 Non-Homogeneous Inverse Algorithms

A non-homogeneous inverse algorithm is formulated that is capable of identifying a square tumour within the global domain. It consists of a 2×2 stencil shown in Figure 4.2, which can be mapped across the global domain to produce an over-determined system of linear equations that

inter-relates all of the discretized stiffness regions throughout the domain. The algorithm adopts integration limits similar to that of the homogeneous centred base point inverse algorithm to negate the need to calculate derivatives of the displacement amplitudes. The mapping of the stencil is performed by moving the local base point across the global domain.

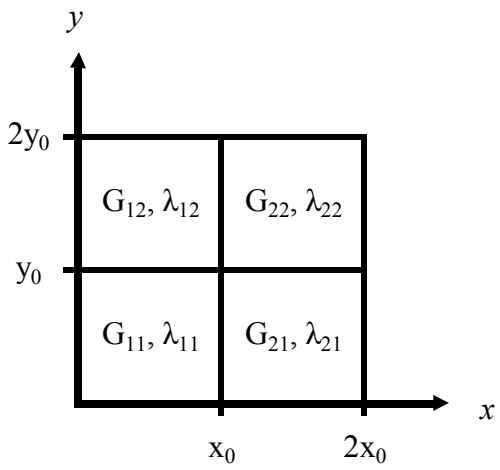

Figure 4.2 – Diagram detailing the local geometric structure and notation for the non-homogeneous inverse algorithm

The integral formulation in this non-homogeneous case follows the same procedure as that of Equations (4.23) and (4.24) in the homogeneous case. In particular, Equations (4.25b) and (4.26b) still hold for the non-homogeneous case since the x and y direction longitudinal stresses, σ_x and σ_y, and the shear stress, τ_{xy}, are continuous over the whole domain. For example, over the sub-domain of Figure 4.2:

$$\begin{aligned}\int_{x_0-x}^{x_0+x}\left(\frac{\partial \sigma_x}{\partial x}\right)dx &= \int_{x_0}^{x_0+x}\left(\frac{\partial \sigma_x}{\partial x}\right)dx + \int_{x_0-x}^{x_0}\left(\frac{\partial \sigma_x}{\partial x}\right)dx \\ &= \sigma_x(x_0+x,y) - \sigma_x{}^{+}(x_0,y) + \sigma_x{}^{-}(x_0,y) - \sigma_x(x_0-x,y) \\ &= \sigma_x(x_0+x,y) - \sigma_x(x_0-x,y)\end{aligned} \tag{4.33}$$

That is, the stress terms either side of the stiffness boundary cancel out due to stress continuity, similarly for σ_y and τ_{xy}.

However, the λ and G terms can no longer be factored straight out of the integrals as in Equations (4.25e) and (4.26e). In this non-homogeneous case, they each take on four constant

values in the sub-domains of Figure 4.2. For simplicity of notation and for each (i, j) element in Figure 4.2, σ_x, σ_y and τ_{xy} are denoted as follows:

$$\begin{aligned} \sigma_x^{ij} &= (2G_{ij} + \lambda_{ij})u_x + \lambda_{ij} v_y \\ \sigma_y^{ij} &= (2G_{ij} + \lambda_{ij})v_y + \lambda_{ij} u_x \qquad (i, j) \in \{1,2\} \\ \tau_{xy}^{ij} &= G_{ij}\left(u_y + v_x\right) \end{aligned} \tag{4.34}$$

Equation (4.25b) can then be broken into the four portions and written in the form:

$$\begin{aligned} &\int_0^{y_0}\int_{y_0}^{y_0+y}\int_{x_0}^{2x_0} \sigma_x^{22}(x,y)dx\,dydy - \int_0^{y_0}\int_{y_0}^{y_0+y}\int_0^{x_0} \sigma_x^{12}(x,y)dx\,dydy \\ &\quad + \int_0^{y_0}\int_{y_0-y}^{y_0}\int_{x_0}^{2x_0} \sigma_x^{21}(x,y)dx\,dydy - \int_0^{y_0}\int_{y_0-y}^{y_0}\int_0^{x_0} \sigma_x^{11}(x,y)dx\,dydy \end{aligned} \tag{4.35}$$

Substituting Equation (4.34) for $\sigma_x^{ij}(x,y)$:

$$\begin{aligned} &= \int_0^{y_0}\int_{y_0}^{y_0+y}\int_{x_0}^{2x_0}\left((2G_{22}+\lambda_{22})u_x(x,y)\right)dx\,dydy + \int_{x_0}^{2x_0}\int_0^{y_0}\int_{y_0}^{y_0+y}\left(\lambda_{22}v_y(x,y)\right)dydydx \\ &\quad - \int_0^{y_0}\int_{y_0}^{y_0+y}\int_0^{x_0}\left((2G_{12}+\lambda_{12})u_x(x,y)\right)dx\,dydy - \int_0^{x_0}\int_0^{y_0}\int_{y_0}^{y_0+y}\left(\lambda_{12}v_y(x,y)\right)dydydx \\ &\quad + \int_0^{y_0}\int_{y_0-y}^{y_0}\int_{x_0}^{2x_0}\left((2G_{21}+\lambda_{21})u_x(x,y)\right)dx\,dydy + \int_{x_0}^{2x_0}\int_0^{y_0}\int_{y_0-y}^{y_0}\left(\lambda_{21}v_y(x,y)\right)dydydx \\ &\quad - \int_0^{y_0}\int_{y_0-y}^{y_0}\int_0^{x_0}\left((2G_{11}+\lambda_{11})u_x(x,y)\right)dx\,dydy - \int_0^{x_0}\int_0^{y_0}\int_{y_0-y}^{y_0}\left(\lambda_{11}v_y(x,y)\right)dydydx \end{aligned} \tag{4.36}$$

Integrating u_x with respect to x and v_y with respect to y:

$$\begin{aligned} &= (2G_{22}+\lambda_{22})\int_0^{y_0}\int_{y_0}^{y_0+y}\left(u(2x_0,y)-u(x_0,y)\right)dydy + \lambda_{22}\int_{x_0}^{2x_0}\int_0^{y_0}\left(v(x,y_0+y)-v(x,y_0)\right)dydx \\ &\quad - (2G_{12}+\lambda_{12})\int_0^{y_0}\int_{y_0}^{y_0+y}\left(u(x_0,y)-u(0,y)\right)dydy - \lambda_{12}\int_{x_0}^{2x_0}\int_0^{y_0}\left(v(x,y_0+y)-v(x,y_0)\right)dydx \\ &\quad + (2G_{21}+\lambda_{21})\int_0^{y_0}\int_{y_0-y}^{y_0}\left(u(2x_0,y)-u(x_0,y)\right)dydy + \lambda_{21}\int_{x_0}^{2x_0}\int_0^{y_0}\left(v(x,y_0)-v(x,y_0-y)\right)dydx \\ &\quad - (2G_{11}+\lambda_{11})\int_0^{y_0}\int_{y_0-y}^{y_0}\left(u(x_0,y)-u(0,y)\right)dydy - \lambda_{11}\int_0^{x_0}\int_0^{y_0}\left(v(x,y_0)-v(x,y_0-y)\right)dydx \end{aligned} \tag{4.37}$$

Substituting $z = x_0 + x$ and $z = x_0 - x$ and simplifying:

$$
\begin{aligned}
&=(2G_{22}+\lambda_{22})\int_0^{y_0}\int_{y_0}^{y_0+y}\big(u(2x_0,y)-u(x_0,y)\big)dydy+\lambda_{22}\int_{x_0}^{2x_0}\int_{y_0}^{2y_0}v(x,y)dydx-\lambda_{22}y_0\int_{x_0}^{2x_0}u(x,y_0)dx\\
&\quad-(2G_{12}+\lambda_{12})\int_0^{y_0}\int_{y_0}^{y_0+y}\big(u(x_0,y)-u(0,y)\big)dydy-\lambda_{12}\int_0^{x_0}\int_{y_0}^{2y_0}v(x,y)dydx+\lambda_{12}y_0\int_0^{x_0}u(x,y_0)dx\\
&\quad+(2G_{21}+\lambda_{21})\int_0^{y_0}\int_{y_0-y}^{y_0}\big(u(2x_0,y)-u(x_0,y)\big)dydy+\lambda_{21}y_0\int_{x_0}^{2x_0}u(x,y_0)dx-\lambda_{21}\int_{x_0}^{2x_0}\int_0^{y_0}v(x,y)dydx\\
&\quad-(2G_{11}+\lambda_{11})\int_0^{y_0}\int_{y_0-y}^{y_0}\big(u(x_0,y)-u(0,y)\big)dydy-\lambda_{11}y_0\int_0^{x_0}u(x,y_0)dx+\lambda_{11}\int_0^{x_0}\int_0^{y_0}v(x,y)dydx
\end{aligned}
\tag{4.38}
$$

Similarly, using the notation of Equation (4.34), Equation (4.26b) can be written in the form:

$$
\begin{aligned}
&\int_0^{x_0}\int_{x_0}^{x_0+x}\int_{y_0}^{2y_0}\tau_{xy}{}^{22}(x,y)dy\,dxdx-\int_0^{x_0}\int_{x_0}^{x_0+x}\int_0^{y_0}\tau_{xy}{}^{21}(x,y)dy\,dxdx\\
&\qquad+\int_0^{x_0}\int_{x_0-x}^{x_0}\int_{y_0}^{2y_0}\tau_{xy}{}^{12}(x,y)dy\,dxdx-\int_0^{x_0}\int_{x_0-x}^{x_0}\int_0^{y_0}\tau_{xy}{}^{11}(x,y)dy\,dxdx
\end{aligned}
\tag{4.39}
$$

Substituting Equation (4.34) for $\tau_{xy}{}^{ij}(x,y)$:

$$
\begin{aligned}
&=\int_0^{x_0}\int_{x_0}^{x_0+x}\int_{y_0}^{2y_0}\big(G_{22}u_y(x,y)\big)dy\,dxdx+\int_{y_0}^{2y_0}\int_0^{x_0}\int_{x_0}^{x_0+x}\big(G_{22}v_x(x,y)\big)dxdxdy\\
&\quad-\int_0^{x_0}\int_{x_0}^{x_0+x}\int_0^{y_0}\big(G_{21}u_y(x,y)\big)dy\,dxdx-\int_0^{y_0}\int_0^{x_0}\int_{x_0}^{x_0+x}\big(G_{21}v_x(x,y)\big)dxdxdy\\
&\quad+\int_0^{x_0}\int_{x_0-x}^{x_0}\int_{y_0}^{2y_0}\big(G_{12}u_y(x,y)\big)dy\,dxdx+\int_{y_0}^{2y_0}\int_0^{x_0}\int_{x_0-x}^{x_0}\big(G_{12}v_x(x,y)\big)dxdxdy\\
&\quad-\int_0^{x_0}\int_{x_0-x}^{x_0}\int_0^{y_0}\big(G_{11}u_y(x,y)\big)dy\,dxdx-\int_0^{y_0}\int_0^{x_0}\int_{x_0-x}^{x_0}\big(G_{11}v_x(x,y)\big)dxdxdy
\end{aligned}
\tag{4.40}
$$

Integrating u_y with respect to y and v_x with respect to x:

$$
\begin{aligned}
&=G_{22}\int_0^{x_0}\int_{x_0}^{x_0+x}\big(u(x,2y_0)-u(x,y_0)\big)dxdx+G_{22}\int_{y_0}^{2y_0}\int_0^{x_0}\big(v(x_0+x,y)-v(x_0,y)\big)dxdy\\
&\quad-G_{21}\int_0^{x_0}\int_{x_0}^{x_0+x}\big(u(x,y_0)-u(x,0)\big)dxdx-G_{21}\int_0^{y_0}\int_0^{x_0}\big(v(x_0+x,y)-v(x_0,y)\big)dxdy\\
&\quad+G_{12}\int_0^{x_0}\int_{x_0-x}^{x_0}\big(u(x,2y_0)-u(x,y_0)\big)dxdx+G_{12}\int_{y_0}^{2y_0}\int_0^{x_0}\big(v(x_0,y)-v(x_0-x,y)\big)dxdy\\
&\quad-G_{11}\int_0^{x_0}\int_{x_0-x}^{x_0}\big(u(x,y_0)-u(x,0)\big)dxdx-G_{21}\int_0^{y_0}\int_0^{x_0}\big(v(x_0,y)-v(x_0-x,y)\big)dxdy
\end{aligned}
\tag{4.41}
$$

Substituting $z=x_0+x$ and $z=x_0-x$ and simplifying:

$$
\begin{aligned}
&= G_{22}\int_0^{x_0}\int_{x_0}^{x_0+x}\big(u(x,2y_0)-u(x,y_0)\big)dxdx + G_{22}\int_{x_0}^{2x_0}\int_{y_0}^{2y_0} v(x,y)dydx - G_{22}x_0\int_{y_0}^{2y_0} v(x_0,y)dy \\
&\quad -G_{21}\int_0^{x_0}\int_{x_0}^{x_0+x}\big(u(x,y_0)-u(x,0)\big)dxdx - G_{21}\int_{x_0}^{2x_0}\int_0^{y_0} v(x,y)dydx + G_{21}x_0\int_0^{y_0} v(x_0,y)dy \\
&\quad +G_{12}\int_0^{x_0}\int_{x_0-x}^{x_0}\big(u(x,2y_0)-u(x,y_0)\big)dxdx + G_{12}\int_0^{x_0}\int_{y_0}^{2y_0} v(x,y)dydx - G_{12}x_0\int_{y_0}^{2y_0} v(x_0,y)dy \\
&\quad -G_{11}\int_0^{x_0}\int_{x_0-x}^{x_0}\big(u(x,y_0)-u(x,0)\big)dxdx - G_{11}\int_0^{x_0}\int_0^{y_0} v(x,y)dydx + G_{11}x_0\int_0^{y_0} v(x_0,y)dy
\end{aligned}
\tag{4.42}
$$

Thus, for the non-homogeneous case, adding Equations (4.38) and (4.42) and collecting common terms involving u and v (measured displacement) gives the derivative free integral formulation of Equation (4.23):

$$
\begin{aligned}
&a_2\int_0^{y_0}\int_{y_0}^{y_0+y}\big(E_{22}\big(u(2x_0,y)-u(x_0,y)\big)+E_{12}\big(u(0,y)-u(x_0,y)\big)\big)dydy \\
&\quad +a_2\int_0^{y_0}\int_{y_0-y}^{y_0}\big(E_{21}\big(u(2x_0,y)-u(x_0,y)\big)+E_{11}\big(u(0,y)-u(x_0,y)\big)\big)dydy \\
&\quad +a_3\int_0^{x_0}\int_{x_0}^{x_0+x}\big(E_{22}\big(u(x,2y_0)-u(x,y_0)\big)+E_{21}\big(u(x,0)-u(x,y_0)\big)\big)dxdx \\
&\quad +a_3\int_0^{x_0}\int_{x_0-x}^{x_0}\big(E_{12}\big(u(x,2y_0)-u(x,y_0)\big)+E_{11}\big(u(x,0)-u(x,y_0)\big)\big)dxdx \\
&\quad +a_3(E_{12}-E_{22})x_0\int_{y_0}^{2y_0} v(x_0,y)dy + a_3(E_{21}-E_{11})x_0\int_0^{y_0} v(x_0,y)dy \\
&\quad +(a_1-a_3)(E_{21}-E_{22})y_0\int_{x_0}^{2x_0} v(x,y_0)dx + (a_1-a_3)(E_{12}-E_{11})y_0\int_0^{x_0} v(x,y_0)dx \\
&\quad +a_1E_{11}\int_0^{x_0}\int_0^{y_0} v(x,y)dydx - a_1E_{21}\int_{x_0}^{2x_0}\int_0^{y_0} v(x,y)dydx \\
&\quad -a_1E_{12}\int_0^{x_0}\int_{y_0}^{2y_0} v(x,y)dydx + a_1E_{22}\int_{x_0}^{2x_0}\int_{y_0}^{2y_0} v(x,y)dydx \\
&= \int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\big(-\rho\omega^2 u\big)dxdxdydy
\end{aligned}
\tag{4.43}
$$

Where a_1, a_2 and a_3 are given by Equations (4.17)-(4.19), but the following identities also hold for $(i,j)\in\{1,2\}$:

$$
a_1 = \frac{\lambda_{ij}+G_{ij}}{E_{ij}} = \frac{\nu}{(1+\nu)(1-2\nu)} + \frac{1}{2(1+\nu)} \tag{4.44}
$$

$$
a_2 = \frac{\lambda_{ij}+2G_{ij}}{E_{ij}} = \frac{\nu}{(1+\nu)(1-2\nu)} + \frac{1}{1+\nu} \tag{4.45}
$$

$$a_3 = \frac{G_{ij}}{E_{ij}} = \frac{1}{2(1+\nu)} \tag{4.46}$$

Note that the stiffness values E_{ij} in Equation (4.43) relate to a local region or 'stencil' involving the four elements shown in Figure 4.2. As this stencil is moved over a larger global domain, the global indices of the element in Figure 4.2 would change. Hence the notation E_{ij} is replaced with $\overline{E}_{ij}$ to denote the local coordinates of stiffness relative to the stencil in Figure 4.2.

Equation (4.43) can therefore be rewritten directly in terms of these more general unknown local stiffness elements, $\overline{E}_{11}$, $\overline{E}_{21}$, $\overline{E}_{12}$ and $\overline{E}_{22}$ as follows:

$$k_{11}\overline{E}_{11} + k_{21}\overline{E}_{21} + k_{12}\overline{E}_{12} + k_{22}\overline{E}_{22} = \int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(-\rho\omega^2 u\right)dxdxdydy \tag{4.47}$$

$$\begin{aligned} k_{11} = {} & a_2\int_0^{y_0}\int_{y_0-y}^{y_0}\left(-u(x_0,y)+u(0,y)\right)dydy + a_3\int_0^{x_0}\int_{x_0-x}^{x_0}\left(-u(x,y_0)+u(x,0)\right)dxdx \\ & -a_3x_0\int_0^{y_0}v(x_0,y)dy - (a_1-a_3)y_0\int_0^{x_0}v(x,y_0)dx + a_1\int_0^{y_0}\int_0^{x_0}v(x,y)dxdy \end{aligned} \tag{4.48}$$

$$\begin{aligned} k_{21} = {} & a_2\int_0^{y_0}\int_{y_0-y}^{y_0}\left(u(2x_0,y)-u(x_0,y)\right)dydy + a_3\int_0^{x_0}\int_{x_0}^{x_0+x}\left(-u(x,y_0)+u(x,0)\right)dxdx \\ & +a_3x_0\int_0^{y_0}v(x_0,y)dy + (a_1-a_3)y_0\int_{x_0}^{2x_0}v(x,y_0)dx - a_1\int_0^{y_0}\int_{x_0}^{2x_0}v(x,y)dxdy \end{aligned} \tag{4.49}$$

$$\begin{aligned} k_{12} = {} & a_2\int_0^{y_0}\int_{y_0}^{y_0+y}\left(-u(x_0,y)+u(0,y)\right)dydy + a_3\int_0^{x_0}\int_{x_0-x}^{x_0}\left(u(x,2y_0)-u(x,y_0)\right)dxdx \\ & +a_3x_0\int_{y_0}^{2y_0}v(x_0,y)dy + (a_1-a_3)y_0\int_0^{x_0}v(x,y_0)dx - a_1\int_{y_0}^{2y_0}\int_0^{x_0}v(x,y)dxdy \end{aligned} \tag{4.50}$$

$$\begin{aligned} k_{22} = {} & a_2\int_0^{y_0}\int_{y_0}^{y_0+y}\left(u(2x_0,y)-u(x_0,y)\right)dydy + a_3\int_0^{x_0}\int_{x_0}^{x_0+x}\left(u(x,2y_0)-u(x,y_0)\right)dxdx \\ & -a_3x_0\int_{y_0}^{2y_0}v(x_0,y)dy - (a_1-a_3)y_0\int_{x_0}^{2x_0}v(x,y_0)dx + a_1\int_{y_0}^{2y_0}\int_{x_0}^{2x_0}v(x,y)dxdy \end{aligned} \tag{4.51}$$

The formulation of the derivative free terms that relate to the integration of Equation (4.24) are similar to those in Equation (4.23) and therefore only the end results in terms of the local stiffness elements, $\overline{E}_{ij}$, are shown for simplicity:

$$m_{11}\overline{E}_{11} + m_{21}\overline{E}_{21} + m_{12}\overline{E}_{12} + m_{22}\overline{E}_{22} = \int_0^{y_0}\int_{y_0-y}^{y_0+y}\int_0^{x_0}\int_{x_0-x}^{x_0+x}\left(-\rho\omega^2 v\right)dxdxdydy \tag{4.52}$$

$$\begin{aligned} m_{11} = {} & a_3\int_0^{y_0}\int_{y_0-y}^{y_0}\left(-v(x_0,y)+v(0,y)\right)dydy + a_2\int_0^{x_0}\int_{x_0-x}^{x_0}\left(-v(x,y_0)+v(x,0)\right)dxdx \\ & -(a_1-a_3)x_0\int_0^{y_0}u(x_0,y)dy - a_3y_0\int_0^{x_0}u(x,y_0)dx + a_1\int_0^{y_0}\int_0^{x_0}u(x,y)dxdy \end{aligned} \tag{4.53}$$

$$\begin{aligned} m_{21} = {} & a_3\int_0^{y_0}\int_{y_0-y}^{y_0}\left(v(2x_0,y)-v(x_0,y)\right)dydy + a_2\int_0^{x_0}\int_{x_0}^{x_0+x}\left(-v(x,y_0)+v(x,0)\right)dxdx \\ & +(a_1-a_3)x_0\int_0^{y_0}u(x_0,y)dy + a_3y_0\int_{x_0}^{2x_0}u(x,y_0)dx - a_1\int_0^{y_0}\int_{x_0}^{2x_0}u(x,y)dxdy \end{aligned} \tag{4.54}$$

$$\begin{aligned} m_{12} = {} & a_3\int_0^{y_0}\int_{y_0}^{y_0+y}\left(-v(x_0,y)+v(0,y)\right)dydy + a_2\int_0^{x_0}\int_{x_0-x}^{x_0}\left(v(x,2y_0)-v(x,y_0)\right)dxdx \\ & +(a_1-a_3)x_0\int_{y_0}^{2y_0}u(x_0,y)dy + a_3y_0\int_0^{x_0}u(x,y_0)dx - a_1\int_{y_0}^{2y_0}\int_0^{x_0}u(x,y)dxdy \end{aligned} \tag{4.55}$$

$$\begin{aligned} m_{22} = {} & a_3\int_0^{y_0}\int_{y_0}^{y_0+y}\left(v(2x_0,y)-v(x_0,y)\right)dydy + a_2\int_0^{x_0}\int_{x_0}^{x_0+x}\left(v(x,2y_0)-v(x,y_0)\right)dxdx \\ & -(a_1-a_3)x_0\int_{y_0}^{2y_0}u(x_0,y)dy - a_3y_0\int_{x_0}^{2x_0}u(x,y_0)dx + a_1\int_{y_0}^{2y_0}\int_{x_0}^{2x_0}u(x,y)dxdy \end{aligned} \tag{4.56}$$

Equations (4.47) and (4.52) relate the Young's Modulus values of four independent stiffness regions in a given stencil, and the u and v displacement within the geometry and local coordinate system that is defined in Figure 4.3. It effectively represents a 2×2 stencil that can be applied to any four discretized stiffness areas of any compatible sizes within a two dimensional global domain.

The two dimensional global domain adopted for the purposes of verifying this algorithm is shown in Figure 4.4. The geometry of this global domain is square as it directly relates to the geometry of the forward simulation model with a total size of $0.1\text{m}\times0.1\text{m}$. However, note that the 2×2 stencil is derived from an equation that makes no assumptions about the boundary conditions applied to the actuated elastic medium. Therefore, the algorithm could be applied to any sinusoidally actuated, 2D plane strain elastic medium with arbitrary boundary conditions and shape. For example Figure 4.5 shows an arbitrary shape that is first approximated by squares. The stencil of Figure 4.3 could then be moved over the domain.

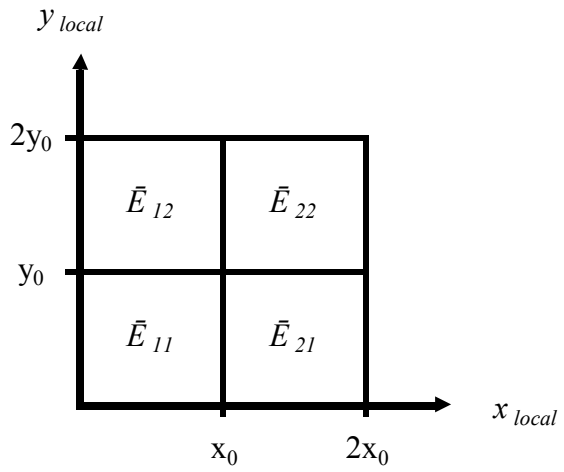

Figure 4.3 – Description of the geometry and local coordinate system of the 2x2 stencil

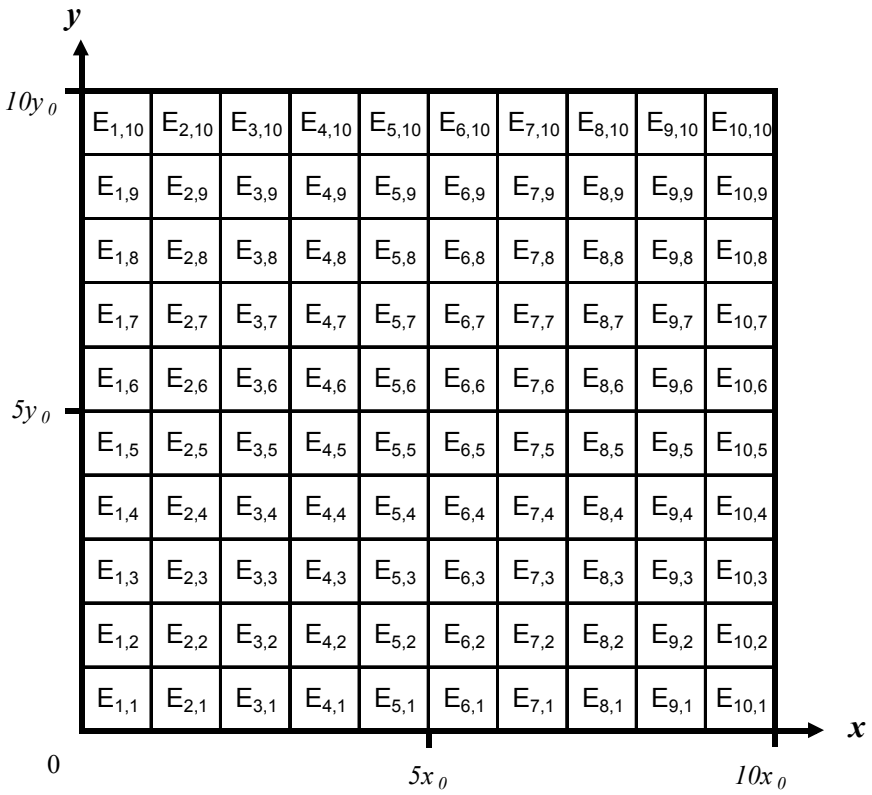

Figure 4.4 – Diagram that shows the discretization of the global domain into a series of elements each with independent stiffness values $E_{i,j}$, $(i,j) \in \{1,2,...,9\}$.

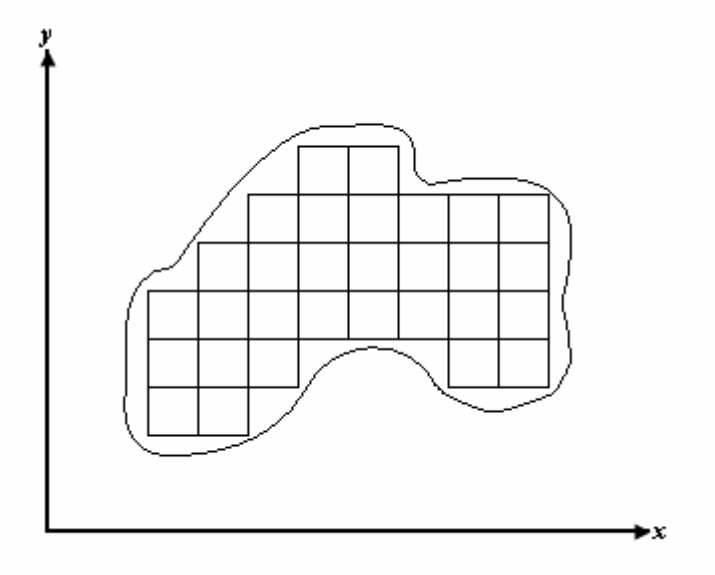

Figure 4.5 – Arbitrary shape approximated by squares in order to apply the stencil shown in Figure 4.2

The 2×2 stencil of Figure 4.3 in global coordinates with respect to Figure 4.4 is shown in Figure 4.6 where $(i,j)\in\{1,2,...,9\}$ and x_0 and y_0 are fixed at 10mm to contain precisely four stiffness elements or regions.

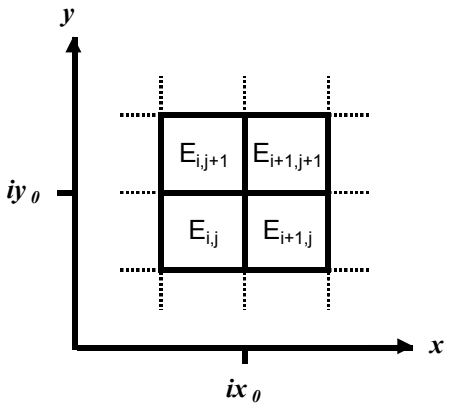

Figure 4.6 – 2x2 Stencil of Figure 4.3 in global coordinates

For a given midpoint $(\bar{x}_i,\bar{y}_j)=(ix_0,jy_0)$ in Figure 4.6, Equations (4.47)-(4.56) can be rewritten in terms of the global coordinates and E_{ij} as defined in Figure 4.4:

$$\begin{aligned} &k_{11}^{(i,j)}E_{i,j}+k_{21}^{(i,j)}E_{i+1,j}+k_{12}^{(i,j)}E_{i,j+1}+k_{22}^{(i,j)}E_{i+1,j+1} \\ &\quad=\int_0^{y_0}\int_{\bar{y}_0-y}^{\bar{y}_0+y}\int_0^{x_0}\int_{\bar{x}_0-x}^{\bar{x}_0+x}\left(-\rho\omega^2 u\right)dxdxdydy \end{aligned} \tag{4.57}$$

where for $(i, j) \in \{1,2,...,n-1\}$:

$$\begin{aligned} k_{11}^{(i,j)} &= a_2 \int_0^{y_0} \int_{\bar{y}_j - y}^{\bar{y}_j} \left(-u(\bar{x}_i, y) + u(\bar{x}_i - x_0, y)\right) dy dy + a_3 \int_0^{x_0} \int_{\bar{x}_i - x}^{\bar{x}_i} \left(-u(x, \bar{y}_j) + u(x, \bar{y}_j - y_0)\right) dx dx \\ &\quad - a_3 x_0 \int_{\bar{y}_j - y_0}^{\bar{y}_j} v(\bar{x}_i, y) dy - (a_1 - a_3) y_0 \int_{\bar{x}_i - x_0}^{\bar{x}_i} v(x, \bar{y}_j) dx + a_1 \int_{\bar{y}_j - y_0}^{\bar{y}_j} \int_{\bar{x}_i - x_0}^{\bar{x}_i} v(x, y) dx dy \end{aligned} \tag{4.58}$$

$$\begin{aligned} k_{21}^{(i,j)} &= a_2 \int_0^{y_0} \int_{\bar{y}_j - y}^{\bar{y}_j} \left(u(\bar{x}_i + x_0, y) - u(\bar{x}_i, y)\right) dy dy + a_3 \int_0^{x_0} \int_{\bar{x}_i}^{\bar{x}_i + x} \left(-u(x, \bar{y}_j) + u(x, \bar{y}_j - y_0)\right) dx dx \\ &\quad + a_3 x_0 \int_{\bar{y}_j - y_0}^{\bar{y}_j} v(\bar{x}_i, y) dy + (a_1 - a_3) y_0 \int_{\bar{x}_i}^{\bar{x}_i + x_0} v(x, \bar{y}_j) dx - a_1 \int_{\bar{y}_j - y_0}^{\bar{y}_j} \int_{\bar{x}_i}^{\bar{x}_i + x_0} v(x, y) dx dy \end{aligned} \tag{4.59}$$

$$\begin{aligned} k_{12}^{(i,j)} &= a_2 \int_0^{y_0} \int_{\bar{y}_j}^{\bar{y}_j + y} \left(-u(\bar{x}_i, y) + u(\bar{x}_i - x_0, y)\right) dy dy + a_3 \int_0^{x_0} \int_{\bar{x}_i - x}^{\bar{x}_i} \left(u(x, \bar{y}_j + y_0) - u(x, \bar{y}_j)\right) dx dx \\ &\quad + a_3 x_0 \int_{\bar{y}_j}^{\bar{y}_j + y_0} v(\bar{x}_i, y) dy + (a_1 - a_3) y_0 \int_{\bar{x}_i - x_0}^{\bar{x}_i} v(x, \bar{y}_j) dx - a_1 \int_{\bar{y}_j}^{\bar{y}_j + y_0} \int_{\bar{x}_i - x_0}^{\bar{x}_i} v(x, y) dx dy \end{aligned} \tag{4.60}$$

$$\begin{aligned} k_{22}^{(i,j)} &= a_2 \int_0^{y_0} \int_{\bar{y}_j}^{\bar{y}_j + y} \left(u(\bar{x}_i + x_0, y) - u(\bar{x}_i, y)\right) dy dy + a_3 \int_0^{x_0} \int_{\bar{x}_i}^{\bar{x}_i + x} \left(u(x, \bar{y}_j + y_0) - u(x, \bar{y}_j)\right) dx dx \\ &\quad - a_3 x_0 \int_{\bar{y}_j}^{\bar{y}_j + y_0} v(\bar{x}_i, y) dy - (a_1 - a_3) y_0 \int_{\bar{x}_i}^{\bar{x}_i + x_0} v(x, \bar{y}_j) dx + a_1 \int_{\bar{y}_j}^{\bar{y}_j + y_0} \int_{\bar{x}_i}^{\bar{x}_i + x_0} v(x, y) dx dy \end{aligned} \tag{4.61}$$

and:

$$\begin{aligned} &m_{11}^{(i,j)} \overline{E}_{i,j} + m_{21}^{(i,j)} \overline{E}_{i+1,j} + m_{12}^{(i,j)} \overline{E}_{i,j+1} + m_{22}^{(i,j)} \overline{E}_{i+1,j+1} \\ &\quad = \int_0^{y_0} \int_{\bar{y}_0 - y}^{\bar{y}_0 + y} \int_0^{x_0} \int_{\bar{x}_0 - x}^{\bar{x}_0 + x} \left(-\rho \omega^2 v\right) dx dx dy dy \end{aligned} \tag{4.62}$$

where for $(i, j) \in \{1,2,...,n-1\}$:

$$\begin{aligned} m_{11}^{(i,j)} &= a_3 \int_0^{y_0} \int_{\bar{y}_j - y}^{\bar{y}_j} \left(-v(\bar{x}_i, y) + v(\bar{x}_i - x_0, y)\right) dy dy + a_2 \int_0^{x_0} \int_{\bar{x}_i - x}^{\bar{x}_i} \left(-v(x, \bar{y}_j) + v(x, \bar{y}_j - y_0)\right) dx dx \\ &\quad - (a_1 - a_3) x_0 \int_{\bar{y}_j - y_0}^{\bar{y}_j} u(\bar{x}_i, y) dy - a_3 y_0 \int_{\bar{x}_i - x_0}^{\bar{x}_i} u(x, \bar{y}_j) dx + a_1 \int_{\bar{y}_j - y_0}^{\bar{y}_j} \int_{\bar{x}_i - x_0}^{\bar{x}_i} u(x, y) dx dy \end{aligned} \tag{4.63}$$

$$m_{21}^{(i,j)} = a_3\int_0^{y_0}\int_{\bar{y}_j-y}^{\bar{y}_j}\left(v(\bar{x}_i+x_0,y)-v(\bar{x}_i,y)\right)dydy + a_2\int_0^{x_0}\int_{\bar{x}_i}^{\bar{x}_i+x}\left(-v(x,\bar{y}_j)+v(x,\bar{y}_j-y_0)\right)dxdx$$
$$+(a_1-a_3)x_0\int_{\bar{y}_j-y_0}^{\bar{y}_j}u(\bar{x}_i,y)dy + a_3y_0\int_{\bar{x}_i}^{\bar{x}_i+x_0}u(x,\bar{y}_j)dx - a_1\int_{\bar{y}_j-y_0}^{\bar{y}_j}\int_{\bar{x}_i}^{\bar{x}_i+x_0}u(x,y)dxdy \quad (4.64)$$

$$m_{12}^{(i,j)} = a_3\int_0^{y_0}\int_{\bar{y}_j}^{\bar{y}_j+y}\left(-v(\bar{x}_i,y)+v(\bar{x}_i-x_0,y)\right)dydy + a_2\int_0^{x_0}\int_{\bar{x}_i-x}^{\bar{x}_i}\left(v(x,\bar{y}_j+y_0)-v(x,\bar{y}_j)\right)dxdx$$
$$+(a_1-a_3)x_0\int_{\bar{y}_j}^{\bar{y}_j+y_0}u(\bar{x}_i,y)dy + a_3y_0\int_{\bar{x}_i-x_0}^{\bar{x}_i}u(x,\bar{y}_j)dx - a_1\int_{\bar{y}_j}^{\bar{y}_j+y_0}\int_{\bar{x}_i-x_0}^{\bar{x}_i}u(x,y)dxdy \quad (4.65)$$

$$m_{22}^{(i,j)} = a_3\int_0^{y_0}\int_{\bar{y}_j}^{\bar{y}_j+y}\left(v(\bar{x}_i+x_0,y)-v(\bar{x}_i,y)\right)dydy + a_2\int_0^{x_0}\int_{\bar{x}_i}^{\bar{x}_i+x}\left(v(x,\bar{y}_j+y_0)-v(x,\bar{y}_j)\right)dxdx$$
$$-(a_1-a_3)x_0\int_{\bar{y}_j}^{\bar{y}_j+y_0}u(\bar{x}_i,y)dy - a_3y_0\int_{\bar{x}_i}^{\bar{x}_i+x_0}u(x,\bar{y}_j)dx + a_1\int_{\bar{y}_j}^{\bar{y}_j+y_0}\int_{\bar{x}_i}^{\bar{x}_i+x_0}u(x,y)dxdy \quad (4.66)$$

Note that Equations (4.57)-(4.66) reduce to Equations (4.47)–(4.56) when $i=j=1$ or equivalently $(\bar{x}_i,\bar{y}_j)=(\bar{x}_1,\bar{y}_1)=(x_0,y_0)$.

For a global domain with n^2 discretized stiffness segments, the 2×2 stencil provides $2(n-1)^2$ equations that relate the displacement amplitudes of the actuated material to the Young's Modulus of each discretized stiffness area. Thus, there are now $2(n-1)^2$ independent equations in n^2 unknowns. For $n\geq4$, this presents an over-determined system of linear equations that can be solved for the unknown material property constants by linear least squares.

For the 10x10 global domain presented in Figure 4.4, $n=10$, and there are 100 unknown stiffness values assigned to each discretized stiffness area and $2(10-1)^2=162$ independent equations that relate these stiffness values to the displacement amplitudes.

4.2.1 Non-Homogeneous Inverse Problem Formulation

To demonstrate the process of forming this over-determined system of equations as a matrix function, Equation (4.57) is applied across a global domain for $n=4$. In this case, there will be $(n-1)^2=9$ equations consisting of Equation (4.57) with all combinations of $(i,j)\in\{1,2,3\}$. For example, choosing $(i,j)=(1,1)$ gives the equation:

$$k_{11}^{(1,1)}E_{1,1} + k_{21}^{(1,1)}E_{2,1} + k_{12}^{(1,1)}E_{1,2} + k_{22}^{(1,1)}E_{2,2} = \int_0^{y_0}\int_{y_1-y}^{y_1+y}\int_0^{x_0}\int_{x_1-x}^{x_1+x}\left(-\rho\omega^2 u\right)dxdxdydy \tag{4.67}$$

Continuing for the other 8 (i, j) combinations, $\{(2,1),(3,1),(1,2),(2,2),(3,2),(1,3),(2,3),(3,3)\}$, the equations can be written in matrix form as follows:

$$A\underline{e} = b \tag{4.68}$$

where:

$$A=\begin{bmatrix}
k_{11}^{(1,1)} & k_{21}^{(1,1)} & 0 & 0 & k_{12}^{(1,1)} & k_{22}^{(1,1)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & k_{11}^{(2,1)} & k_{21}^{(2,1)} & 0 & 0 & k_{12}^{(2,1)} & k_{22}^{(2,1)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & k_{11}^{(3,1)} & k_{21}^{(3,1)} & 0 & 0 & k_{12}^{(3,1)} & k_{22}^{(3,1)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & k_{11}^{(1,2)} & k_{21}^{(1,2)} & 0 & 0 & k_{12}^{(1,2)} & k_{22}^{(1,2)} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & k_{11}^{(2,2)} & k_{21}^{(2,2)} & 0 & 0 & k_{12}^{(2,2)} & k_{22}^{(2,2)} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & k_{11}^{(3,2)} & k_{21}^{(3,2)} & 0 & 0 & k_{12}^{(3,2)} & k_{22}^{(3,2)} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{11}^{(1,3)} & k_{21}^{(1,3)} & 0 & 0 & k_{12}^{(1,3)} & k_{22}^{(1,3)} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{11}^{(2,3)} & k_{21}^{(2,3)} & 0 & 0 & k_{12}^{(2,3)} & k_{22}^{(2,3)} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{11}^{(3,3)} & k_{21}^{(3,3)} & 0 & 0 & k_{12}^{(3,3)} & k_{22}^{(3,3)}
\end{bmatrix} \tag{4.69}$$

$$b = \begin{bmatrix}
\int_0^{y_0}\int_{y_1-y}^{y_1+y}\int_0^{x_0}\int_{x_1-x}^{x_1+x}\left(-\rho\omega^2 u\right)dxdxdydy \\
\int_0^{y_0}\int_{y_1-y}^{y_1+y}\int_0^{x_0}\int_{x_2-x}^{x_2+x}\left(-\rho\omega^2 u\right)dxdxdydy \\
\vdots \\
\int_0^{y_0}\int_{y_{n-1}-y}^{y_{n-1}+y}\int_0^{x_0}\int_{x_{n-1}-x}^{x_{n-1}+x}\left(-\rho\omega^2 u\right)dxdxdydy
\end{bmatrix} \tag{4.70}$$

$$\underline{e} = \left[E_{1,1}, E_{2,1}, E_{3,1}, E_{4,1}, E_{2,1}, E_{2,2}, E_{2,3}, E_{2,4}, E_{3,1}, E_{3,2}, E_{3,3}, E_{3,4}, E_{4,1}, E_{4,2}, E_{4,3}, E_{4,4}\right]^T \tag{4.71}$$

A 2×2 non-homogeneous stencil was found to be the optimal size for the best conditioned global non-homogeneous inverse problem solution. This result occurs because a 1x1 non-homogeneous stencil effectively integrates over only half of each stiffness element. Thus, there are less points to average out the effects of noise in the measured displacements u and v. On the other hand, a 3×3 or 4×4 discretized stencil still end up being formulated across a single 10mm$\times$10mm

element, so there is no net gain. However, these larger stencils introduce more single integral terms that are less accurate than the double integral terms which have a greater low-pass filtering effect on the data. Thus, the effect of using higher order stencils is to decrease the accuracy of the algorithm. Hence in all further cases of testing the algorithm, a 2×2 stencil is used.

4.3 Non-Homogeneous Stress Continuity Constraint Model

The non-homogeneous integral formulation of the previous section was found to work well for excitation frequencies on the order of 100Hz, but not as well for lower frequencies of around 50Hz for the given system parameters chosen for both the simulation and inverse algorithm. This is because the spatial wavelengths of the motion data are inversely proportional to the actuation frequency. Therefore, the proportion of sinusoidal waveforms per discretized element or 'spatial information' decreases with a decrease in the actuation frequency. This loss of information makes the inverse algorithms more sensitive to noise.

To improve the performance of the 2D non-homogeneous inverse algorithm, where the frequency of excitation is low (50 Hz), a constraint model is introduced to the non-homogeneous inverse algorithm. This constraint model enforces stress continuity across each of the boundaries between discretized stiffness elements. Continuity in the shear stress, τ_{xy}, given in Equation (4.5), is enforced along the boundaries as the longitudinal stresses are significantly ill-conditioned. Specifically, $(\lambda + 2G)u_x \approx -\lambda v_y$ and $(\lambda + 2G)v_y \approx -\lambda u_x$ in Equations (4.3) and (4.4) are ill-conditioned, since breast tissue is very close to incompressible with a Poisson's ratio of 0.49.

The calculation of the shear stress, τ_{xy}, requires the calculation of derivatives. To reduce the effects of noise when calculating the shear stress, cubic polynomials are fitted to local vertical and horizontal strips of the motion amplitude dataset. The polynomials are then differentiated and the derivatives are then evaluated at the appropriate position.

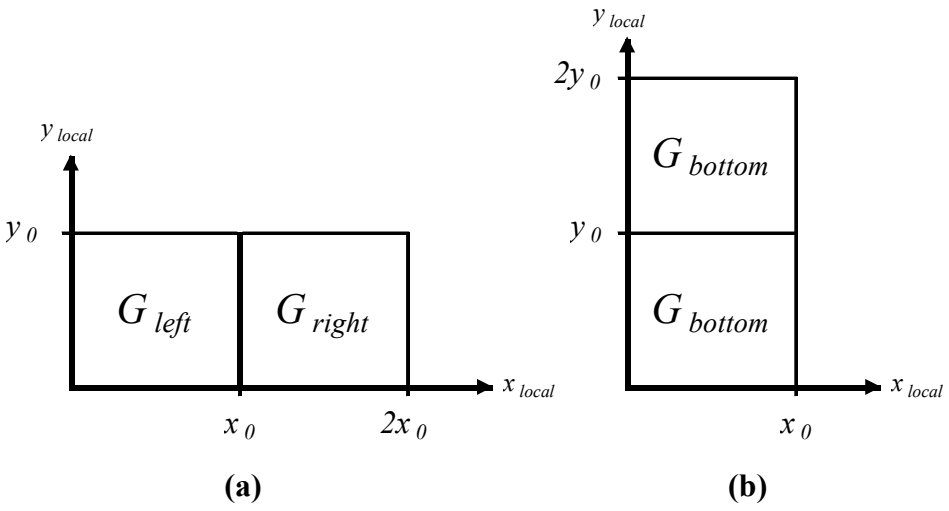

Figure 4.7 – Description of the notation used to describe a vertical stiffness boundary (a) and a horizontal stiffness boundary (b).

Figure 4.7(a) shows two stiffness elements that form a vertical boundary in local (x, y) coordinates. Stress Continuity across the vertical boundary is defined:

$$G_{left}\left(u_y(x_0, y) + v_x(x_0^{-}, y)\right) = G_{right}\left(u_y(x_0, y) + v_x(x_0^{+}, y)\right), \quad 0 \le y \le y_0 \tag{4.72}$$

Where u_y is continuous along the boundary and v_x is not. For simplicity of notation, Equation (4.72) is rewritten in the form:

$$G_{left}\left(u_y + v_x^{-}\right) = G_{right}\left(u_y + v_x^{+}\right) \tag{4.72b}$$

$$\Rightarrow \frac{G_{right}}{G_{left}} = \frac{\left(u_y + v_x^{-}\right)}{\left(u_y + v_x^{+}\right)} \tag{4.72c}$$

$$\Rightarrow \frac{E_{right}}{E_{left}} = \frac{\left(u_y + v_x^{-}\right)}{\left(u_y + v_x^{+}\right)} \tag{4.72d}$$

Equation (4.72d) follows from Equation (4.72c) by using Equation (4.7). Similarly, stress continuity across the horizontal boundary shown in Figure 4.7(b) is defined:

$$G_{bottom}\left(u_y(x, y_0^{\,-}) + v_x(x, y_0)\right) = G_{top}\left(u_y(x, y_0^{\,+}) + v_x(x, y_0)\right), \qquad 0 \le x \le x_0 \qquad (4.73)$$

This is rewritten:

$$G_{bottom}\left(u_y^{\,-} + v_x\right) = G_{top}\left(u_y^{\,+} + v_x\right) \qquad (4.73b)$$

$$\Rightarrow \frac{G_{top}}{G_{bottom}} = \frac{\left(u_y^{\,-} + v_x\right)}{\left(u_y^{\,+} + v_x\right)} \qquad (4.73c)$$

$$\Rightarrow \frac{E_{top}}{E_{bottom}} = \frac{\left(u_y^{\,-} + v_x\right)}{\left(u_y^{\,+} + v_x\right)} \qquad (4.73d)$$

The final stiffness ratio over the vertical boundary in Figure 4.7(a) is formulated as the average of all the ratios along this boundary. For a given data point, the stiffness ratio is defined by Equation (4.72d). Figure 4.8 shows an example of how one such ratio in Equation (4.72d) is calculated for a typical data point denoted by an '×' along the vertical boundary.

Along the line L_v in Figure 4.8, v_x is continuous at x_0. To evaluate the left hand derivative $v_x^{\,-}$ and right hand derivative $v_x^{\,+}$ at x_0, a cubic polynomial is fitted each side of x_0, using the displacement data $\{v(x, \bar{y}),\ 0 \le x \le 2x_0\}$. An example of this calculation is shown in Figure 4.9, where $p_1(x)$ and $p_2(x)$ represent the cubics fitted to the motion data. Thus, $v_x^{\,-} = p_1'(x_0)$ and $v_x^{\,+} = p_2'(x_0)$.

As discussed in Chapter 3, u_y is continuous along all vertical boundary lines in the domain. Hence, u_y can be evaluated at the point $(x_0, \bar{y})$ by fitting a single cubic $q(y)$ along the line L_u in Figure 4.8, which is $\pm 0.5y_0$ from $\bar{y}$, and extends outside the stiffness elements E_{left} and E_{right}. Therefore, $u_y = q'(\bar{y})$. A similar method is used for the horizontal boundary in Figure 4.7(b).

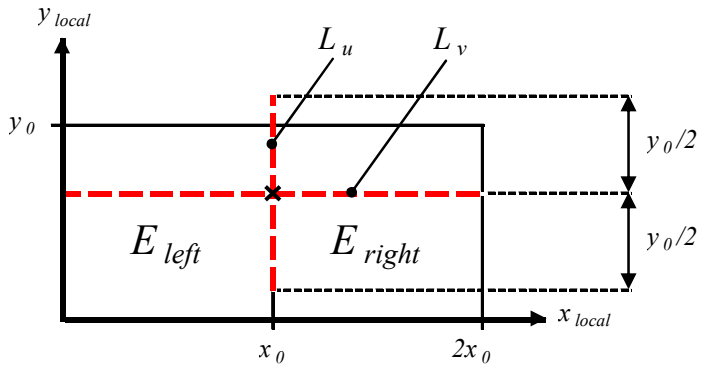

Figure 4.8 – An example of calculating a ratio in Equation (4.72d)

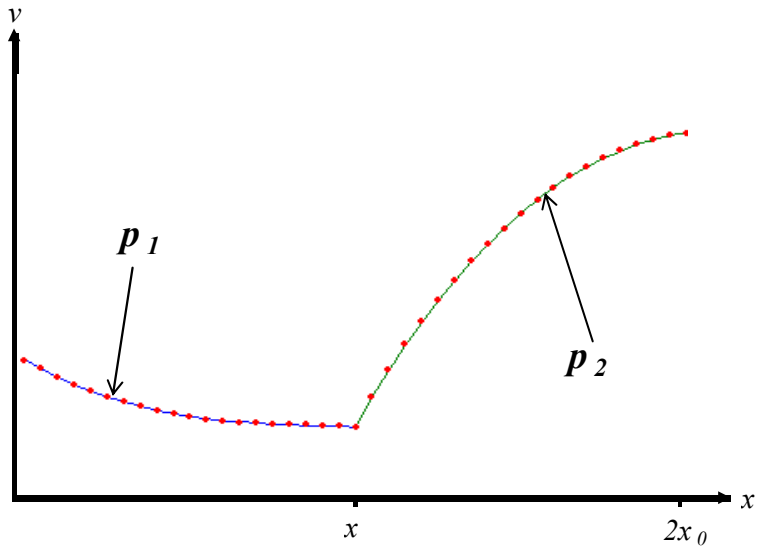

Figure 4.9 – Fitting two cubics to find the left and right hand derivatives v_x^- and v_x^+.

For the global domain of Figure 4.4, there is potentially $90 \times 2 = 180$ final stiffness ratios corresponding to all the vertical and horizontal boundaries. However, on a given stiffness boundary, there is the potential for a number of the ratios for each data point in Equations (4.72d) and (4.73d) to be ill-conditioned and therefore corrupt the final average ratio. To avoid ill-conditioning, several checks are performed.

One cause of ill-conditioning occurs when $u_x + v_y$ comes close to zero in Equations (4.72d) and (4.73d). The first check is to discard the corresponding ratio if the following identities hold:

Horizontal Stiffness Boundary Terms – Equation (4.72d)

$$1 - tolerance < \left| v_x^{+} / - u_y \right| < 1 + tolerance \tag{4.74}$$

$$1 - tolerance < \left| v_x^{-} / - u_y \right| < 1 + tolerance \tag{4.75}$$

Vertical Stiffness Boundary Terms – Equation (4.73d)

$$1 - tolerance < \left| u_y^{+} / - v_x \right| < 1 + tolerance \tag{4.76}$$

$$1 - tolerance < \left| u_y^{-} / - v_x \right| < 1 + tolerance \tag{4.77}$$

The tolerance parameter described above is found from simulation and the optimal value can change slightly for differing boundary conditions and actuation frequencies. However, it is typically in the region of 20% or 0.2.

A second ill-conditioning check is also performed on the derivative terms used to calculate the stiffness ratios. The check is to impose a minimum limit, $sizetol$, on the size of the absolute values of u_y and v_x from Equations (4.72d) and (4.73d) respectively. That is, the stiffness ratio is discarded if the following identities hold:

$$\left| v_x \right|, \left| u_y \right| \le sizetol \tag{4.78}$$

The reason for imposing this limit is that as the value of the derivative terms approaches zero, the ability of the polynomial fitting algorithm to give an accurate value is compromised.

After discarding all of the ill-conditioned terms using Equations (4.74)-(4.78), the average of all the remaining ratios produces a final stiffness ratio that is representative of the ratio of Young's Modulus between the two neighbouring discretized stiffness areas. If there are less than three remaining stiffness ratios per stiffness boundary after discarding the ill-conditioned terms, it is

considered an insufficient number of terms to provide an accurate average. Therefore, in this case, there is no final ratio used for this particular stiffness boundary.

The algorithm then assigns each stiffness element $E_{i,j}$ an average stiffness ratio $q_{i,j}$ defined as follows:

$$q_{i,j} = \frac{\left(v_{left}\right) + \left(v_{right}\right)^{-1} + \left(h_{bottom}\right) + \left(h_{top}\right)^{-1}}{4} \tag{4.79}$$

$$\left(v_{left}\right) = \left(\frac{E_{right}}{E_{left}}\right)_{Left\ Boundary} \tag{4.80}$$

$$\left(v_{right}\right)^{-1} = \left(\frac{E_{right}}{E_{left}}\right)^{-1}_{Right\ Boundary} \tag{4.81}$$

$$\left(h_{bottom}\right) = \left(\frac{E_{top}}{E_{bottom}}\right)_{Bottom\ Boundary} \tag{4.82}$$

$$\left(h_{top}\right)^{-1} = \left(\frac{E_{top}}{E_{bottom}}\right)^{-1}_{Top\ Boundary} \tag{4.83}$$

The terms $\left(v_{right}\right)^{-1}$ and $\left(h_{top}\right)^{-1}$ ensure that the value of $q_{i,j}$ reflects precisely the ratio of $E_{i,j}$ to its surrounding four stiffness values.

If $q_{i,j}$ is close to 1 for a particular stiffness it shows that it is a locally homogenous region. The larger the value increases from 1 the more non-homogeneous the region becomes. To constrain the stiffness values throughout the global domain of Figure 4.4, each $E_{i,j}$ is written as a constant multiple of an unknown stiffness $\hat{E}$:

$$E_{i,j} = q_{i,j}\hat{E} \tag{4.84}$$

where $q_{i,j}$ is defined in Equation (4.79).

For a 1cm tumour at some position in Figure 4.4, $\hat{E}$ can be interpreted as the healthy stiffness value of the region surrounding the tumour. By combining the equality constraints of Equation (4.84) with the integral formulation of Equations (4.57)-(4.66), Young's Modulus can be determined for each stiffness area in Figure 4.4.

Note that if $E_{i,j}$ in Figure 4.10 is a carcinoma surrounded by healthy tissue, the four surrounding elements $E_{i,j}$, $E_{i,j}$, $E_{i,j}$ and $E_{i,j}$ will each contain one high final stiffness boundary ratio along with three ratios close to 1. If the cancer element $E_{i,j}$ is 5 times stiffer than the healthy tissue, $(1+1+1+5)/4 = 2$. Thus the effect of a tumour on the formulation of Equation (4.79) is to slightly lift the stiffness of the four surrounding stiffness elements. However, there will be a significant rise in the stiffness distribution at the element $E_{i,j}$, the only difference is that the rise will be potentially slightly smoother. This behaviour was found to have no significant effect in simulation.

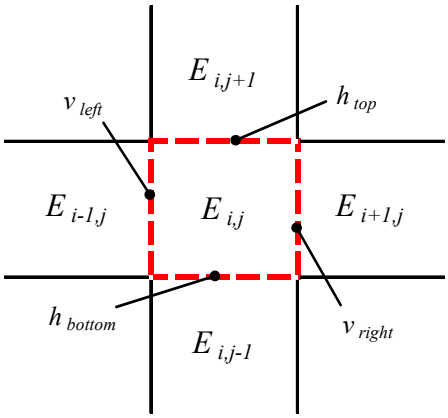

Figure 4.10 – Notation of the stiffness ratios between adjacent stiffness areas for Equation (4.79)

For a given healthy base-line stiffness $\hat{E}$, the overall constraint model is summarized as follows:

$$C\underline{e} = d \tag{4.85}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & -q_{1,1} \\ 0 & 1 & 0 & \cdots & 0 & 0 & -q_{2,1} \\ 0 & 0 & 1 & \cdots & 0 & 0 & -q_{3,1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & -q_{9,10} \\ 0 & 0 & 0 & \cdots & 0 & 1 & -q_{10,10} \end{bmatrix} \tag{4.86}$$

$$\underline{e} = \begin{bmatrix} E_{1,1} \\ E_{2,1} \\ E_{3,1} \\ \vdots \\ E_{1,1} \\ E_{1,1} \\ \hat{E} \end{bmatrix} \tag{4.87}$$

$$d = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.88}$$

This constraint model is combined with the integral formulation of Equations (4.57)-(4.66) to find $\hat{E}$ and thus all $E_{i,j}$ for $(i,j) \in \{1,2,...,10\}$. Note that the current model assumes that the cancerous region coincides precisely with one of the stiffness elements in Figure 4.4. Therefore, in practice the stiffness distribution model would need to be varied in steps to ensure the tumour is as close as possible to matching with a stiffness element. For example, the stiffness distribution of Figure 4.4 could be translated up and across in steps of 0.0025m giving a total of $3^2 + 1 = 10$ combinations including Figure 4.4. However, this method is only included as one possible way of handling low actuation frequencies in the order of 50Hz and further investigations and development into the robustness of this approach is left to future work.

4.4 Summary

Two 2D homogeneous inverse algorithms were developed in order to investigate the different integral methods and provide a framework for the development of a non-homogeneous algorithm. The Initial inverse algorithm of Equations (4.14)-(4.22) adopts an integral method that uses the local origin as the reference with which to integrate from. This method contains derivatives terms that are particularly susceptible to noise and are the primary source of error within the solution. It also contains single integral terms that are not as effective at filtering the noise as the double integral terms.

The centred base point algorithm of Equations (4.27)-(4.32) adopts a unique method of integrating the 2D plain strain Navier's Equations that contains only double integral terms. These terms effectively filter the effects of noise and make the algorithm very robust. Subsequently, the basis of this integral technique is adopted in the non-homogeneous algorithm.

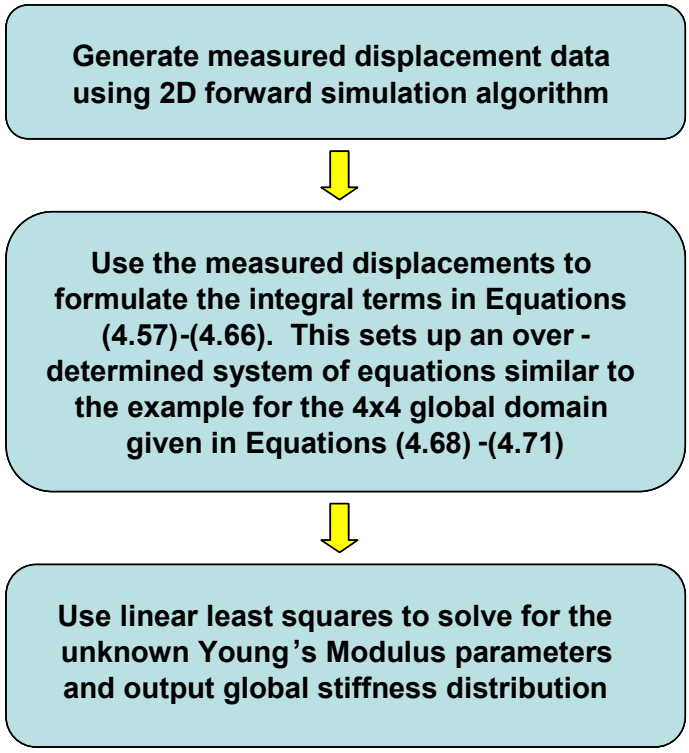

Figure 4.11 – The overall process applied when using the 2D non-homogeneous inverse algorithm

Finally, a 2D non-homogeneous inverse algorithm was then formulated that is capable of identifying a square 1cm square tumour within the square global domain with 10cm sides. It consists of a 2×2 stencil (Figure 4.3) that can be mapped across the global domain to produce an over-determined system of linear equations that inter-relates all of the discretized stiffness elements throughout the domain. The mapping of the stencil is performed by moving the local base point across the global domain. Figure 4.11 shows the overall process of using the 2D non-homogeneous inverse algorithm.

A constraint model, as described in Section 4.3, was formulated that enforces shear stress continuity at the boundary between discretized stiffness elements. It was primarily formulated to increase the ability of the inverse algorithm to identify carcinoma using an actuation frequency of 50Hz, as initial investigations suggest that the method is more sensitive to noise at this lower frequency for the assumed system and modelling parameters that are adopted.

Part 3

Results & Discussion

5 Performance of 1D Inverse Algorithms

5.1 Verification of Forward Simulation Algorithm

The 1D forward simulation model of Section 2.1 is verified using an analytical solution for both the homogeneous and non-homogeneous cases. The analytical solutions are obtained by solving Equation (2.2), with boundary conditions $v(0) = 0$, $v(L) = 0.001$ using MAPLE. Since these analytical solutions are only required for validation of the numerical solutions, the details are not included.

Figure 5.1 shows the convergence of both a homogeneous and non-homogeneous model against their respective analytical solutions. In both instances, the medium is assumed to have a 'healthy' Young's Modulus value of 30kPa. For the non-homogeneous forward simulation, a 1cm long carcinoma or region of greater stiffness is included in the 0.4-0.5cm region of the domain. The value of Young's Modulus for this carcinoma is 300kPa which is ten times greater than the surrounding healthy tissue. The excitation frequency was 100Hz for both instances.

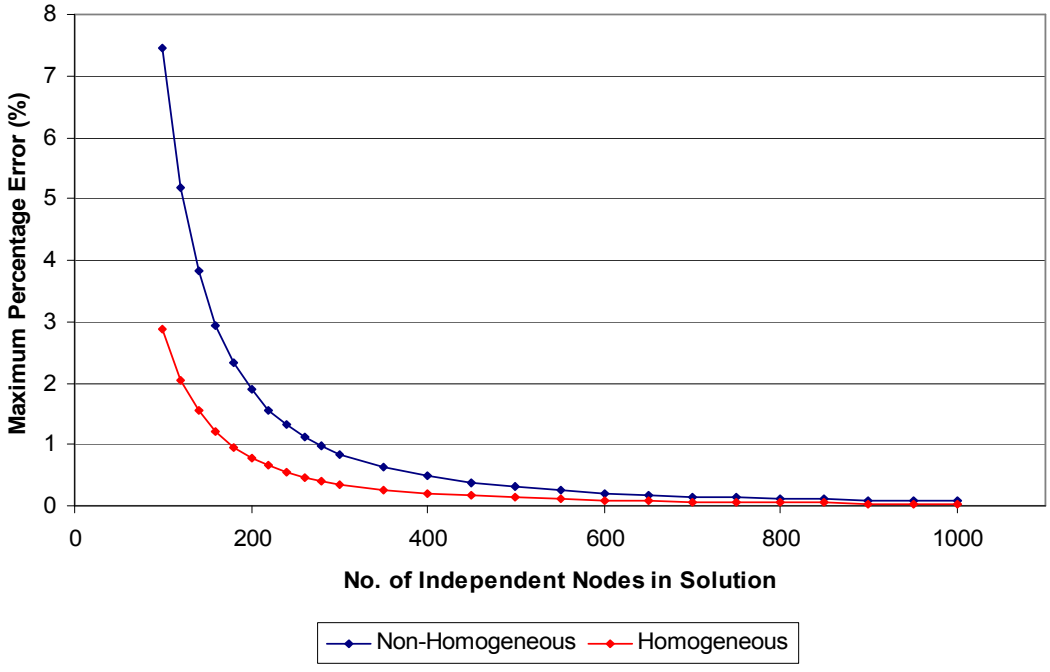

Figure 5.1 - Maximum Error in 1D Simulation Solution compared against Grid Refinement

The 'measured' displacement data used as input into the inverse algorithms was based on a forward model solution using 2000 independent nodes. For the non-homogeneous geometric representation defined above this choice gives a maximum percentage error of 0.0193% between the numerical and analytical simulations.

The Young's Modulus value of 30kPa is the value used for 'healthy' tissue for all simulations in both 1D and 2D. In MRI tests, it has been shown that for 20% precompression of breast tissue and excitation frequency of 4Hz that the Young's Modulus of fat tissue is 24 ± 6 kPa [Krouskop et al,1998]. Fat tissue represents the most significant tissue within a female breast. It has also been shown that Young's Modulus increases with an increase in excitation frequency [Krouskop et al,1998]. As the excitation frequency for the proposed MRE method is expected to exceed 50Hz the upper bound of this stiffness value (30kPa) is chosen as the representative 'healthy' Young's Modulus value.

It has been shown in static analysis performed by Samani et al. (2003) that the Young's Modulus of ductal carcinoma is approximately 6 times greater than the surrounding fat tissue. The dynamic analysis performed by Krouskop et al. (1998) shows that the Young's Modulus of ductal carcinoma is approximately 10 times greater the surrounding fat tissue at an excitation frequency of 4Hz. Therefore, for the analysis of the inverse algorithms, Young's Modulus values representing carcinoma were chosen in the range of 150-300kPa, which is 5-10 times stiffer than healthy tissue.

The other tissue properties required for the forward simulation and subsequent inverse problem algorithm are the density and Poisson's ratio. These values were assumed to be similar to the corresponding properties of water, which is the primary constituent of fat tissue. The values of density and Poisson's ratio are therefore chosen to be 1020kgm^{-3} and 0.49 respectively.

5.2 Comparison of Integral Methods

The three different integral methods of Sections 2.2.1-2.2.3 are evaluated and compared to establish the best method for further development. The global motion data output from the forward simulation requires post-processing before providing the input to the inverse algorithms. In particular, the number of data points per centimetre is reduced to correspond to a level similar

to what can be achieved using current MRI technology. More specifically, 1.5 Tesla MRI machines, which represent the base model of MRI technology, record spatial data at an approximate average of 0.75mm intervals [Haack et al. 1999]. This value represents just greater over 13 points per centimetre. Stronger MRI machines (3 Tesla and greater) are capable of further the density of spatial data points [Haack et al. 1999]. Therefore, a base value of 20 points per centimetre, which is equivalent to the number of points in each discretized segment, is used for initial comparison of integral methods.

Random noise is also added to the forward simulated motion data input. The added noise is calculated as a percentage of the median of the absolute values of the motion dataset. By increasing the noise up to 10% the relative merits of each integral method can be determined. For a Poisson's ratio of $\nu = 0.49$, the Young's Modulus of 30kPa used corresponds to a Shear Modulus of approximately $G = 10.1$kPa Equation (2.2b). To test the algorithm the carcinoma is positioned at the 7^{th} discretized stiffness segment and has stiffness 10 times the healthy value. This location relates to a position of 6-7cm along the 1D global domain shown in Figure 2.1. This location is essentially arbitrary as in general the position doesn't significantly affect the algorithms performance, as discussed later in this Chapter. The excitation frequency chosen is 100Hz.

For each set of parameters are evaluated, 20 separate runs of the algorithm are performed with random noise added to the motion data input each run. This Monte Carlo approach provides a means of quantifying the median and range of each stiffness value reconstructed. Using the cumulative data from the 20 runs a 90% confidence interval is defined for each discretized segment.

5.2.1 Local Inverse Algorithm

The local inverse algorithm of Equations (2.24)-(2.27) is applied. Figure 5.2 shows the 90% confidence interval corresponding to the 20 reconstructed stiffness values for each discretized segment except segment 7, compared against the actual value used in the simulation. The median values are denoted by an $\times$.

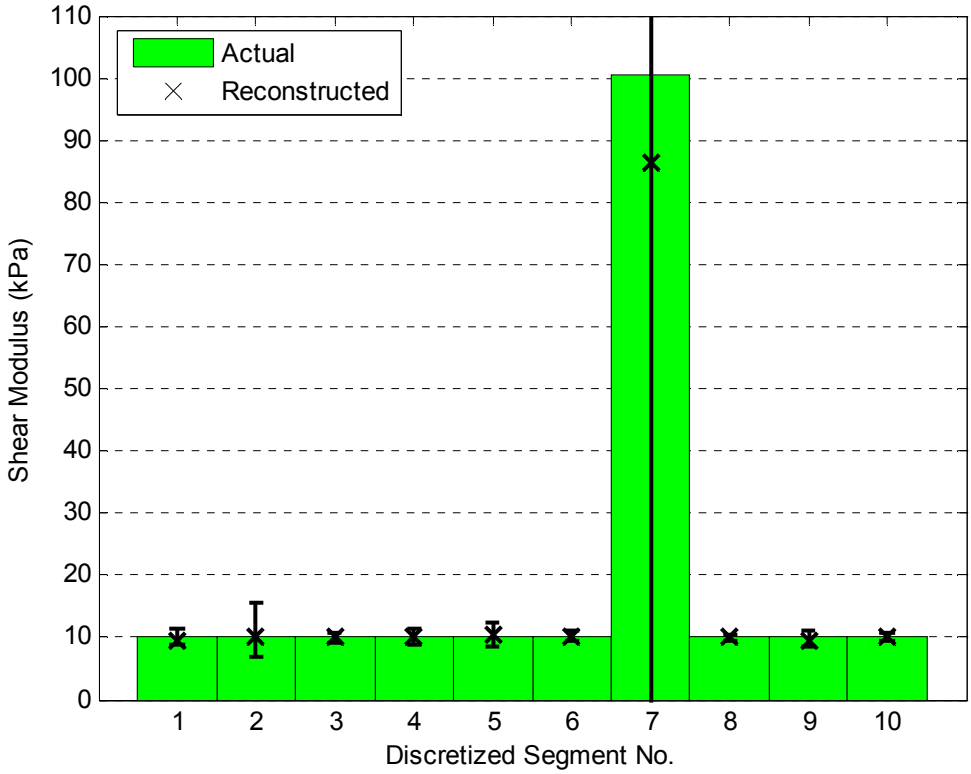

Figure 5.2 – 90% Confidence Interval of the Reconstructed Stiffness Distribution compared against the Actual Distribution from the Local Inverse Algorithm

The complete 90% confidence interval for segment 7 is not shown on Figure 5.2 as it significantly exceeds the Shear Modulus range of interest. The maximum and minimum values of the 90% confidence interval are 1443.5kPa and -324.1kPa respectively. Hence, the reconstruction of the carcinoma stiffness is very inaccurate using this Local inverse algorithm. On the other hand, as can be seen in Figure 5.2, the healthy tissue can be reconstructed accurately.

The reason for the significantly different results between the healthy and cancerous segments is because each discretized segment is treated independent from all the others. Therefore, the calculation of the stiffness relies solely on the dynamics of the motion dataset in each individual segment, which is governed by the spatial period. The low stiffness values, by definition, have a much shorter spatial period. Thus, for the healthy Shear Modulus value of 30kPa, each 1cm discretized segment of motion data will contain approximately 1/3 of the spatial period of the excited tissue, which is sufficient to accurately identify the tissue stiffness even with added noise. However, the motion dataset of the cancerous tissue contains approximately only 1/30 of

the spatial period of the excited tissue for each segment. After adding noise, the small portion of the sinusoid is essentially a straight line which has an infinite spatial period and thus an infinite stiffness. Therefore, in this case, the inverse problem is virtually undefined and it is not possible to accurately identify the stiffness of the carcinoma.

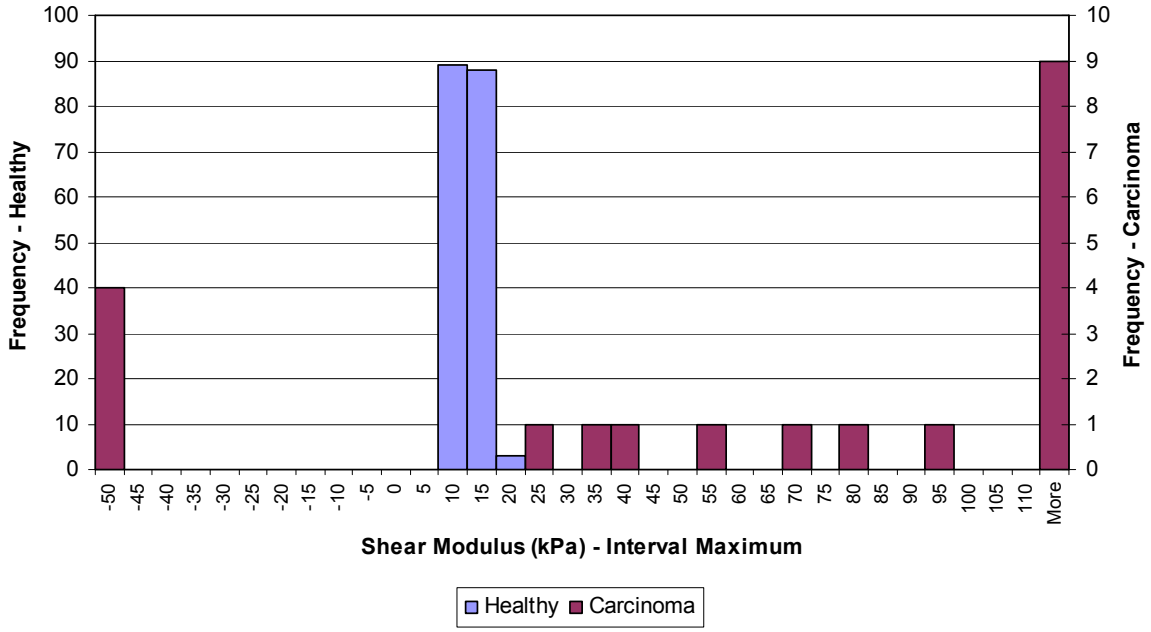

Figure 5.3 – Shear Modulus Distributions of Healthy and Cancerous Tissue from the Local Inverse Algorithm

Although the range of the 90% confidence interval for the cancerous tissue overlaps the intervals of the healthy tissue, Figure 5.3 shows that the distributions do not overlap at any stage. The algorithm is most likely to calculate a large negative stiffness, which is physically impossible, or a large positive stiffness value. Therefore, in most cases the Local inverse algorithm can detect regions where there is a large change of stiffness, but it is not able to calculate the stiffness of the carcinoma accurately.

5.2.2 Global Single Integral Inverse Algorithm

Figure 5.4 shows the median calculated stiffness value and the corresponding 90% confidence interval for each discretized segment compared against the actual value used for simulation for the Global Single Integral inverse algorithm of Equation (2.29), with 10% random noise.

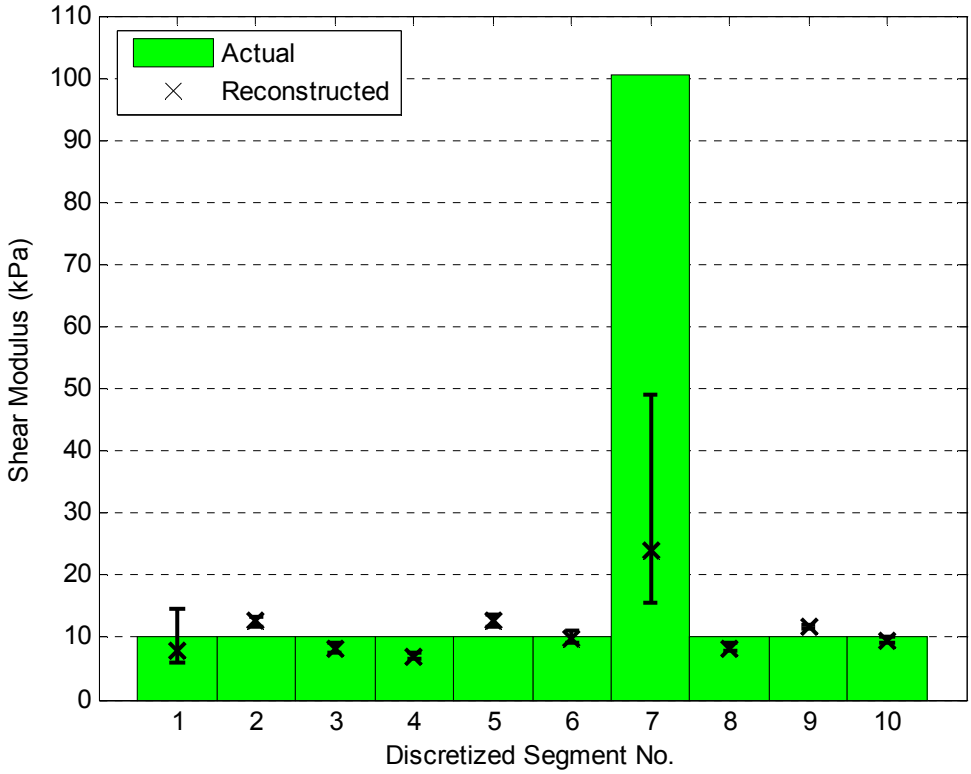

Figure 5.4 – 90% Confidence Interval of the Reconstructed Stiffness Distribution compared against the Actual Distribution from the Global Single Integral Inverse Algorithm

As with the Local inverse algorithm, the calculated stiffness values of healthy tissue are quite accurate and the range is very small for the majority of segments. However, the Shear Modulus of the carcinoma is severely underestimated and the lower limit of the 90% confidence interval of segment 7 coincides with the upper limit of the 90% confidence interval of the healthy tissue in segment 1. The advantage of introducing the use of global information from the motion dataset is offset by using only a single integral formulation, rather than the double integral formulation of the Local inverse algorithm. This behaviour occurs because the single integral formulation requires the calculation of derivative terms, which are sensitive to noise. The Global Single Integral inverse algorithm fits quartic polynomials to the noisy motion dataset to reduce the effects of noise. However, Figure 5.5 verifies that there is still considerable error in the derivative function.

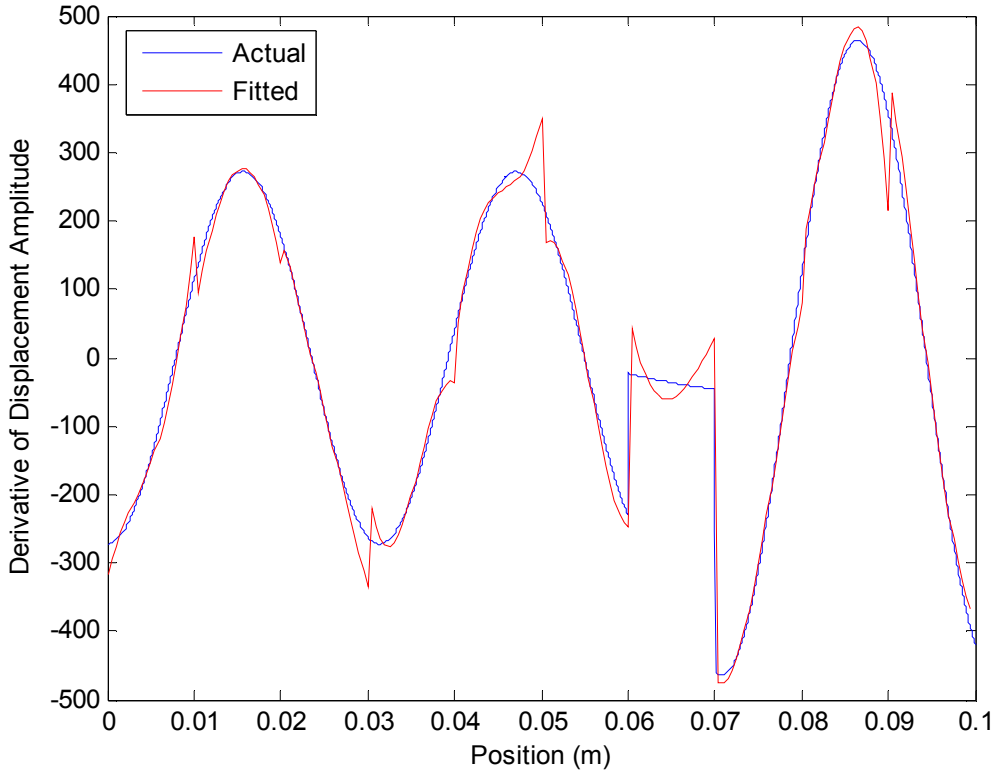

Figure 5.5 – Comparison between the derivative function calculated using polynomial fitting and the actual derivative function

Figure 5.6 shows that there is no positive discrimination between healthy and cancerous stiffness values as the distributions overlap. Although in most cases it is possible to identify a region of greater stiffness, it is not always the case. Thus, there is the possibility that outliers in the reconstruction of healthy tissue could be confused for carcinoma and provide a false positive. There is also the possibility that the calculated stiffness value for the carcinoma falls within the healthy stiffness distribution so that the algorithm does not identify the carcinoma at all.

As mentioned previously, the main problem with the Global Single Integral method is the calculation of the derivative. This derivative is difficult to obtain accurately since it is sensitive to noise. Hence, this method is also not fully suitable, and further integrations are required.

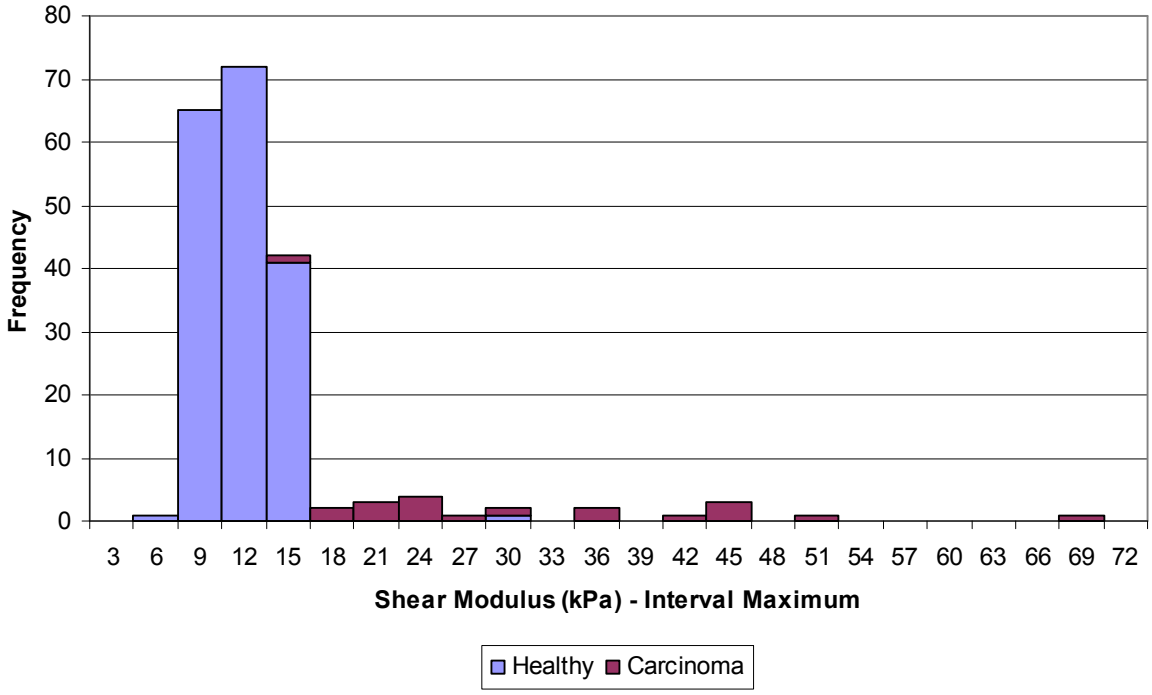

Figure 5.6 – Shear Modulus Distributions of Healthy and Cancerous Tissue from the Global Single Integral Inverse Algorithm

5.2.3 Global Double Integral Inverse Algorithm

Figure 5.7 shows the median calculated stiffness value and the corresponding 90% confidence interval for each discretized segment compared against the actual value used for simulation for the Global Double Integral inverse algorithm of Equations (2.39)-(2.43). The overall performance of the Global Double Integral inverse algorithm is very good. The Shear Moduli calculated for discretized segments with healthy tissue are accurate with the smallest 90% confidence interval range of the three integral methods. The Shear Moduli calculated for the carcinoma segment also had the smallest confidence interval range. The median value of carcinoma stiffness was still underestimated by approximately 30%. However as Figure 5.8 shows, there is still clear positive discrimination between healthy and cancerous stiffness values.

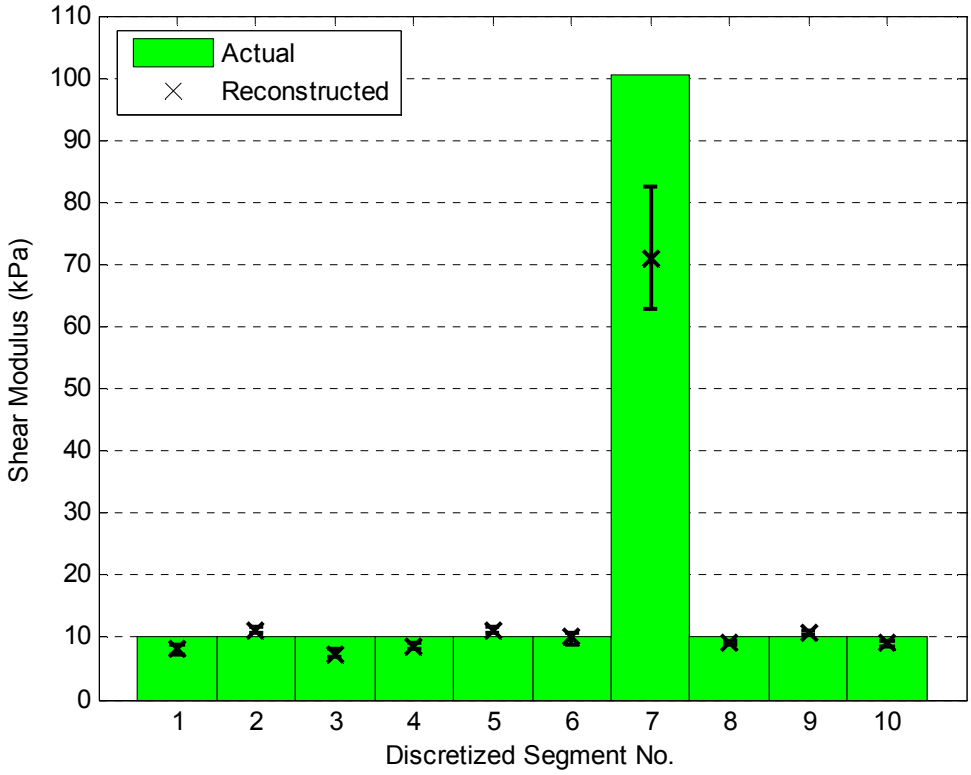

Figure 5.7 – 90% Confidence Interval of the Reconstructed Stiffness Distribution compared against the Actual Distribution from the Global Double Integral Inverse Algorithm

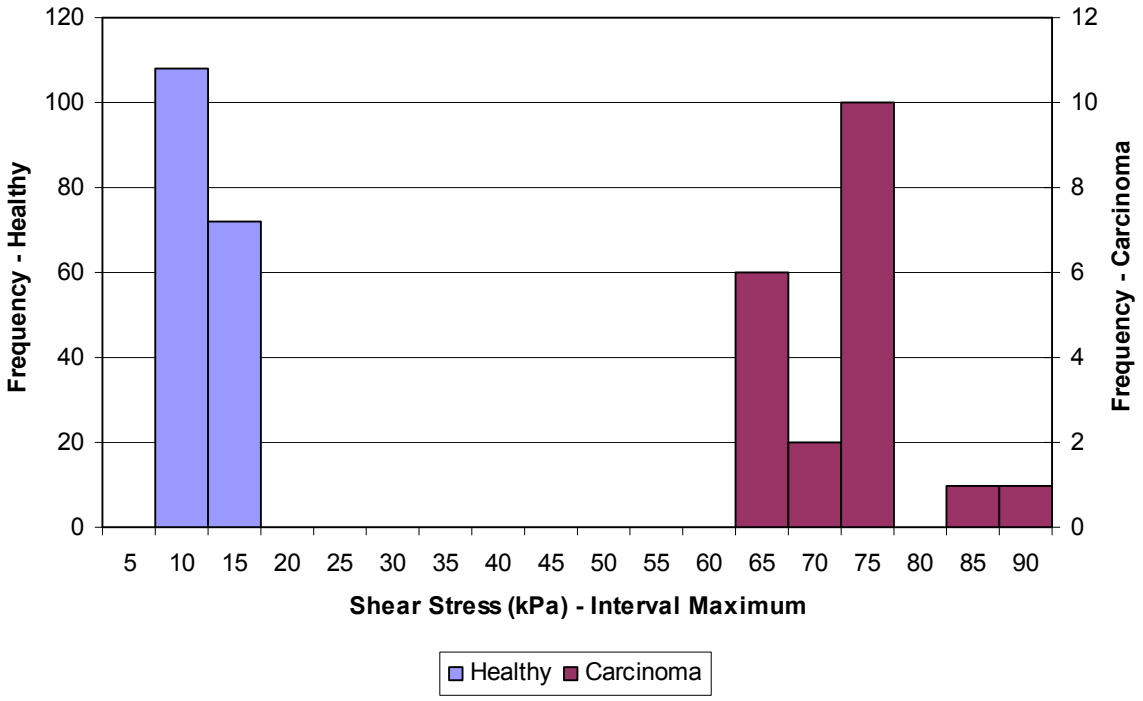

Figure 5.8 – Shear Modulus Distributions of Healthy and Cancerous Tissue from the Global Double Integral Inverse Algorithm

5.2.4 Summary of Evaluation of Integral Methods

From the comparative evaluation of the three different inverse algorithms developed it is clear that the Global Double Integral inverse algorithm performs the best. It successfully identifies both the Shear Moduli of both the carcinoma and healthy tissue with comparatively small 90% confidence interval range. The Local inverse algorithm has potential benefits due to its overall simplicity, but it is only effective at identifying where there is sufficient spatial information within each discretized segment. For example, if the magnitude of Shear Modulus is sufficiently low or the size of the discretized segment is sufficiently large. The single integral formulation was a little better at identifying the actual stiffness value of the carcinoma than the Local inverse algorithm. However, the median carcinoma stiffness was very low and the minimum value of the 90% confidence interval of this stiffness overlapped with the healthy tissue stiffness range. Therefore, the method cannot reliably detect the presence of cancer which is predominantly due to the calculation of derivatives which are sensitive to noise and corrupt the solution.

The main reason for its success is that the Global Double Integral method does not require any calculation of derivatives. Therefore, it is robust to noise, as shown by the good separation of stiffness values between the healthy and cancerous regions in Figures 5.7 and 5.8. Thus, the key point to note is that the integral methods are more robust for this type of problem where no further derivatives need to be calculated. The trade off is that added numerical integrations must be performed, adding some complexity in both the formulation and computation.

5.3 Evaluation of Global Double Integral Inverse Algorithm

The Global Double Integral inverse algorithm is evaluated further to quantify its performance and robustness with the variation of important system parameters. These parameters include:

- The level of noise in the input data
- Frequency of the tissue actuation
- Level of carcinoma stiffness in relation to healthy tissue stiffness
- Resolution or number of points per segment of the input motion data
- Position of the carcinoma within the global domain.

5.3.1 Variation with Level of Random Added Noise

The level of randomly added noise significantly affects the performance of any inverse algorithm. Figure 5.9 shows the performance of the algorithm with respect to the level of uniform random noise added to the motion data input, where the 90% confidence interval for the calculated Shear Modulus values is shown for each point. As the level of noise increases, the Shear Modulus calculated for the carcinoma decreases significantly. The Shear Modulus of the healthy tissue also becomes less accurate.

The reason for the decrease in stiffness for the cancerous region is that for greater levels of noise there is more potential freedom for a fitted model solution of Equation (2.39) to move. In other words, the frequency of potential fitted model curves would increase with increasing noise, which corresponds to a decrease in fitted stiffness. However, there is clear discrimination for all noise levels evaluated.

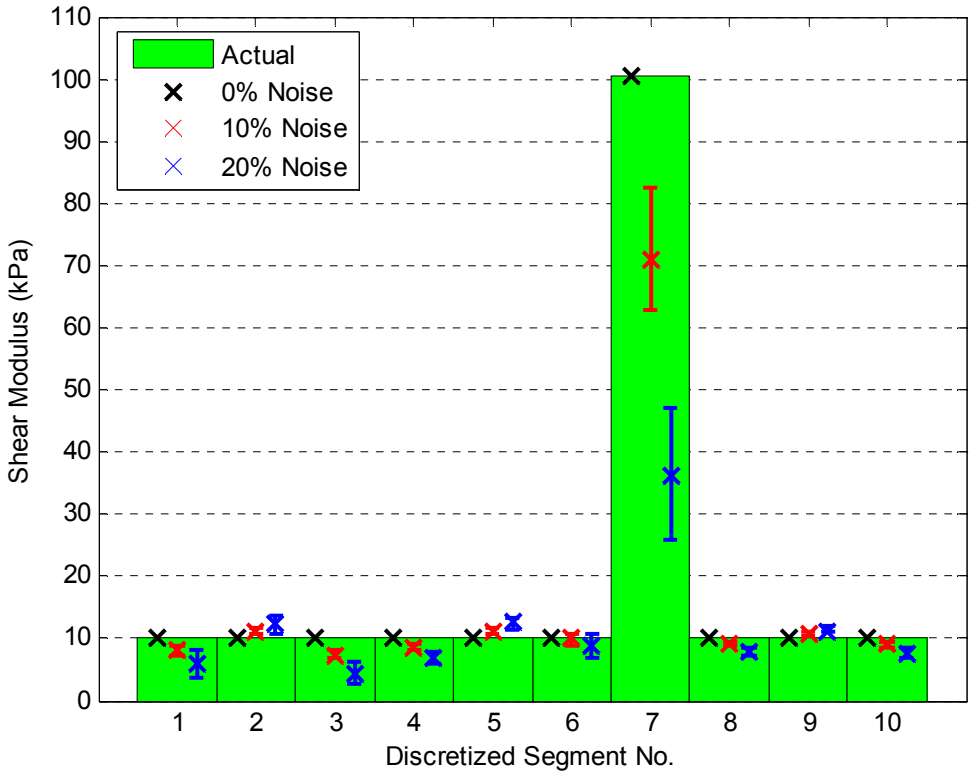

Figure 5.9 – The impact of random added noise on the performance of the Global Double Integral inverse algorithm with 20 points per segment

5.3.2 Variation with Actuation Frequency

The actuation frequency is a significant parameter of the inverse problem as it directly relates to the spatial period of the shear wave induced in the 1D medium. However, the amount the frequency can be increased is limited by both the performance of the actuation system and the frequency of data collection of the MRI system used. The performance of the algorithm with respect to the actuation frequency of the tissue sample is detailed in Figure 5.10.

As the frequency increases, the overall accuracy of the algorithm increases. For example, at 150Hz the carcinoma stiffness calculated is very close to the actual stiffness and the healthy tissue values are very accurate with a small range. In comparison, the algorithm significantly underestimates the carcinoma stiffness, and the healthy tissue stiffness values are relatively inaccurate for the tissue actuated at 50Hz. The clear discrimination of stiffness values, which is clear at 100Hz and 150Hz, is much less obvious at 50Hz.

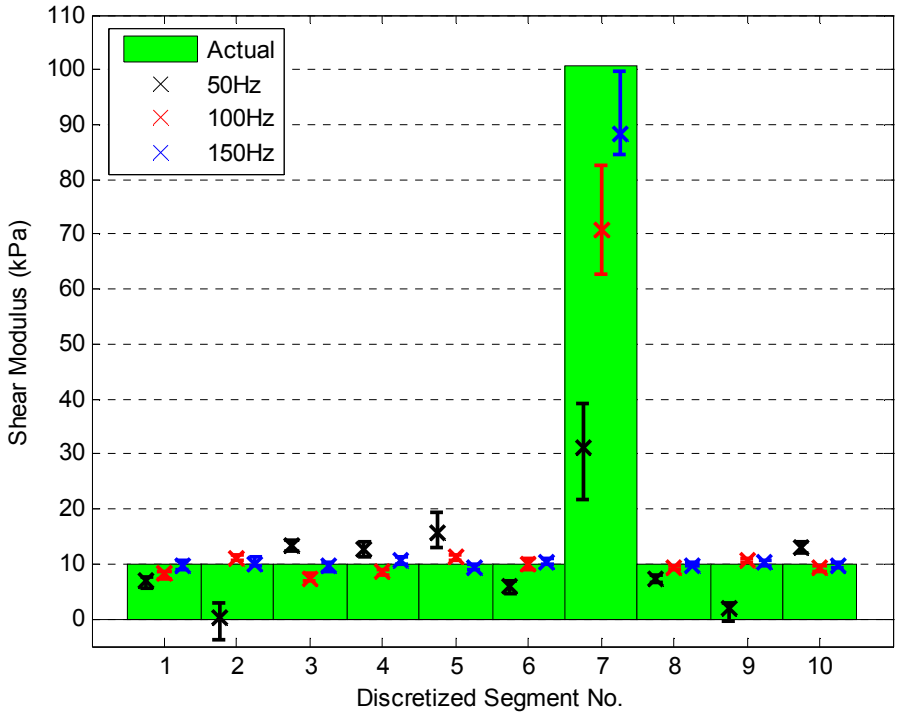

Figure 5.10 – The impact of actuation frequency on the performance of the Global Double Integral inverse algorithm with 10% random added noise and 20 points per segment

The reason for a decreased accuracy in the algorithm for an actuation frequency of 50Hz is because the spatial period of the actuated tissue is inversely proportional to the actuation

frequency. Therefore, for a smaller frequency, the spatial period is larger and thus there is a relatively smaller proportion of the spatial sinusoidal wave contained within each segment. This smaller contrast effectively provides less independent motion information to the inverse algorithm resulting in a less accurate stiffness reconstruction.

This decrease in accuracy of the inverse algorithm with decreasing actuation frequency is dependent on the stiffness and mass parameters assumed for the forward simulation of the motion data. These parameter values directly relate to the natural frequency of the tissue and thus define the elastic response created as input to demonstrate the inverse algorithms. Therefore, a final 3D inverse algorithm needs to be tested for all reasonable ranges of breast tissue stiffness and density to verify its performance with variation of the actuation frequency. These results, however, give an indication of the trends and potential limitations of integration methods that might be encountered with respect to the mechanical properties and dynamic frequency response of the breast.

5.3.3 Variation with Carcinoma Stiffness

Currently, there is no definitive data on the actual stiffness of ductal carcinoma at the range of actuation frequencies proposed using this method. However, significant positive discrimination of stiffness between healthy and cancerous tissue has been found using both static analysis and dynamic tests at actuation frequencies of up to 4Hz [Samani et al, 2003, Krouskop et al, 1998]. Therefore, the performance of the algorithm is quantified using a range based on the limited data available.

Previous analysis of the Global Double Integral inverse algorithm has used a carcinoma stiffness that is 10 times greater than the healthy tissue. Figure 5.11 evaluates the algorithm with carcinoma to healthy tissue stiffness ratio of 5:1, as this level approximately represents the lower limit of this ratio in both static and dynamic testing [Samani et al. 2003, Krouskop et al. 1998]. It shows that the algorithm more accurately reconstructs the tissue distribution for the lower stiffness ratio of 5:1 than for the stiffness ratio of 10:1 using the equivalent analysis shown in Figure 5.9. This result occurs because the spatial period of the carcinoma is reduced, making more spatial information available per segment, which results in identifying the Shear Modulus more accurately.

However, as discussed in Section 5.3.2 this is also dependent on the response defined by the natural frequency. Therefore, the variation of tissue density and healthy tissue stiffness combined with this variation in carcinoma stiffness is required to provide comprehensive analysis of the performance of 1D inverse algorithm.

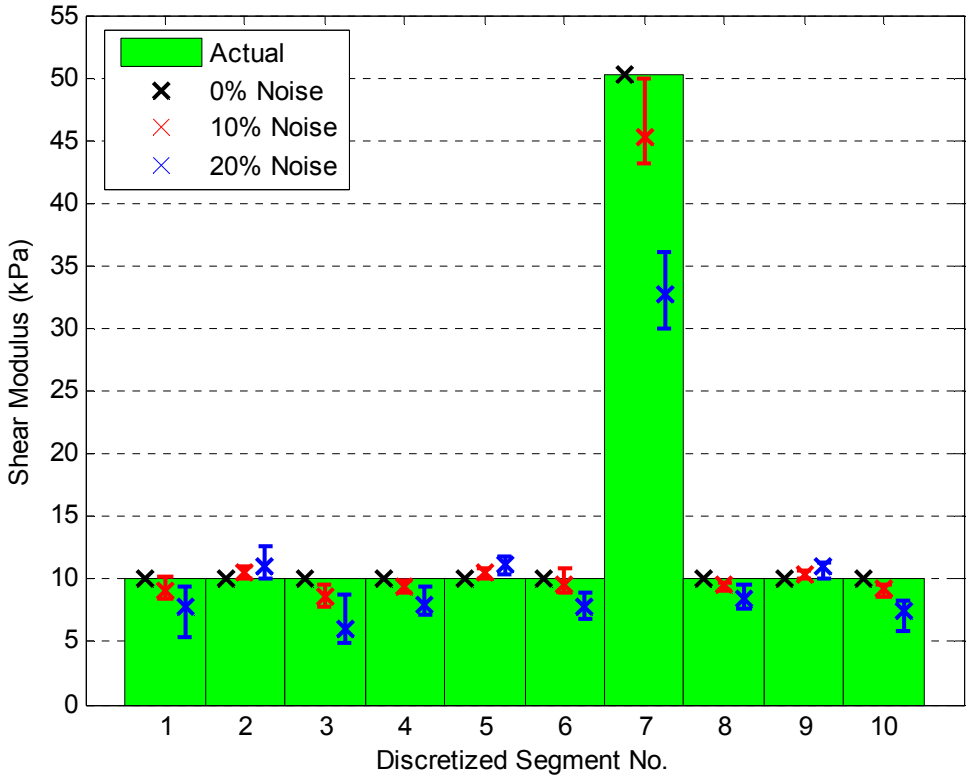

Figure 5.11 – The performance of the Global Double Integral inverse algorithm with a carcinoma to healthy tissue stiffness ratio of 5:1 (10% random added noise and 20 points per segment)

5.3.4 Variation with Data Resolution

The data resolution parameter is limited by the magnetic strength of the MRI system used to produce the motion dataset and is discussed in detail in Section 5.2. Figures 5.12 and 5.13 show the performance of the algorithm with respect to the data resolution of the input motion dataset with 10% and 20% random noise added respectively. It shows the 90% confidence interval of calculated Shear Modulus values for each point.

As the resolution of the input data increases, there is no distinct change in the median value of the Shear Modulus values calculated. However, the range of the 90% confidence interval reduces, particularly for the carcinoma stiffness values. As the number of data points across the

domain increases for the same length, the accuracy of the double integral terms in Equation (2.39) increases along with the number of equations used in the linear least squares solution to Equations (2.40)-(2.43). With more data points the effects of noise are more likely to be filtered or offset, thus reducing the variability of the solution.

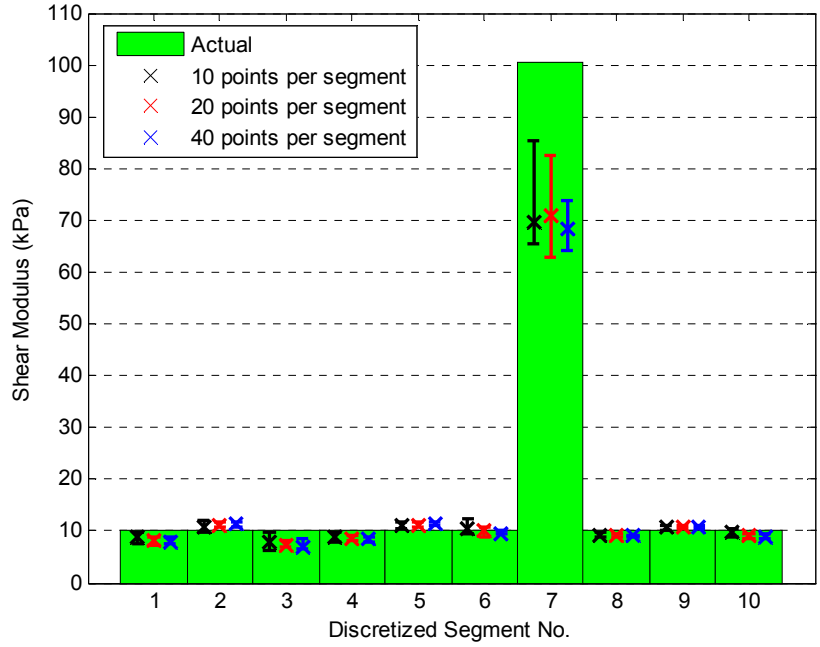

Figure 5.12 – The impact of the number of points per segment on the performance of the Global Double Integral inverse algorithm with 10% random added noise

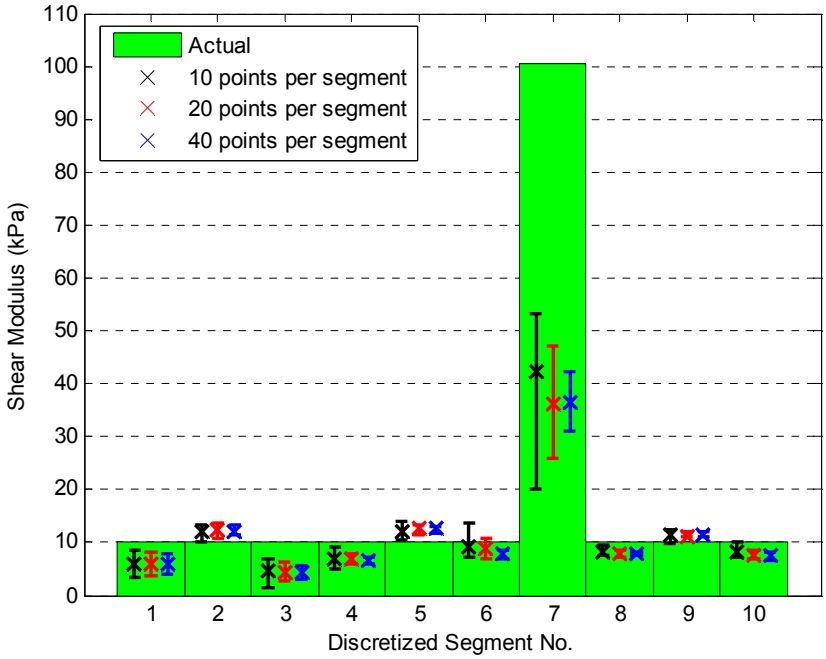

Figure 5.13 – The impact of the number of points per segment on the performance of the Global Double Integral inverse algorithm with 20% random added noise

5.3.5 Variation with Carcinoma Position

The position of the carcinoma can affect the global motion dataset and therefore the performance of the Global Double Integral inverse algorithm. The carcinoma is thus positioned throughout the global domain to analyze the effect on the algorithm. For simplicity in the forward simulation, the carcinoma is assumed to lie in the segments 2-9 in the global domain. In all simulations 10% random noise is added and 20 points are used per segment.

Figure 5.14 shows the results of the fitted carcinoma stiffness over the 8 different segments. Positive discrimination between healthy and cancerous stiffness values is maintained in most cases. However, this result is potentially compromised in position 6 and severely compromised in position 3 where there is no discrimination.

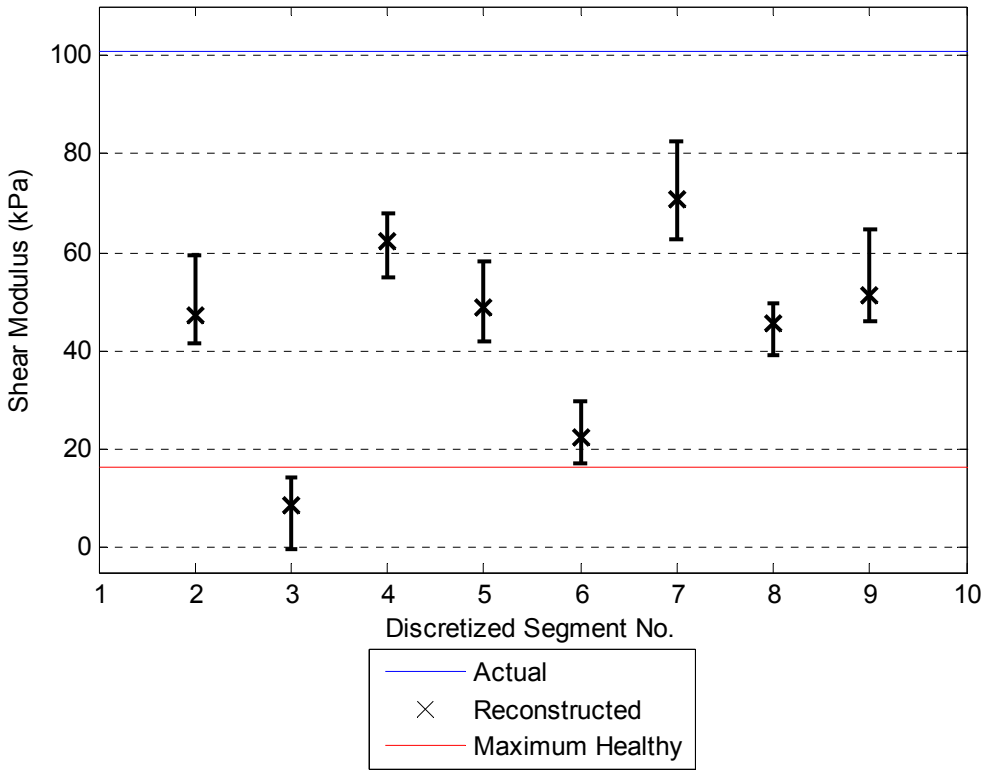

Figure 5.14 – 90% confidence intervals of reconstructed carcinoma stiffness values in all possible positions

In the instance where the carcinoma is positioned at the third discretized segment, the median value of the stiffness calculated directly corresponds to the healthy tissue stiffness. This inaccurate reconstruction of tissue stiffness is caused by the carcinoma being positioned in such a way that it does not significantly affect the global motion dataset, as compared to a

homogeneous solution with Shear Modulus 31.2kPa, as shown in Figure 5.15. Note that both the spatial amplitude and spatial phase are not altered significantly on either side of the carcinoma. Therefore, with the addition of noise, the benefit of using global information to effectively identify a carcinoma within the global domain is diminished in this specific case. This unique effect is less likely to happen in a more realistic 3D model, as shear and longitudinal stress waves propagate in three dimensions. However, this problem can be somewhat remedied by increasing the actuation frequency by 10% as this change sufficiently alters the contrast between the healthy and cancerous global motion datasets.

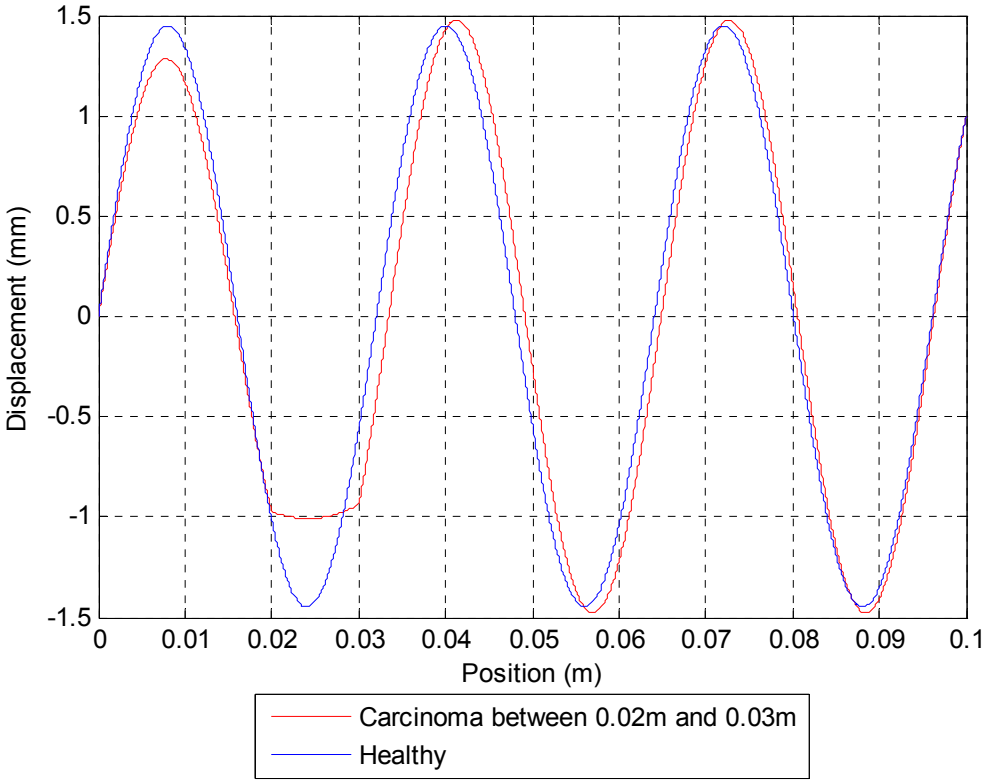

Figure 5.15 – Comparison between cancerous and healthy motion datasets

5.3.6 Mesh Refinement Algorithm

The enhancement of the Global Double Integral inverse algorithm to incorporate mesh refinement was developed in Section 2.2.4 to identify carcinomas that are not aligned with the discretized segments of the domain. It is evaluated by simulating the carcinoma (or region of greater stiffness) of length 1cm, at a position so that it is half in one discretized stiffness segment and half in the adjacent segment. This geometry represents the worst case scenario in terms of alignment of the discretized grid with the carcinoma. Figure 5.16 shows the distribution of

healthy tissue stiffness compared against the maximum carcinoma stiffness calculated for each run.

Although the carcinoma is successfully identified, there is a significant range of fitted healthy tissue stiffness values, which subsequently extends into the carcinoma stiffness distribution, creating false positives within the solution. Therefore, the mesh refinement process as described in Section 2.2.4 is not an effective method for identifying a tumour that is misaligned with the grid. The reason for this result is that reducing segments from 1cm to 0.5cm effectively reduces the amount of spatial information in each segment to the level where it can no longer accurately identify even the healthy stiffness values, at this actuation frequency.

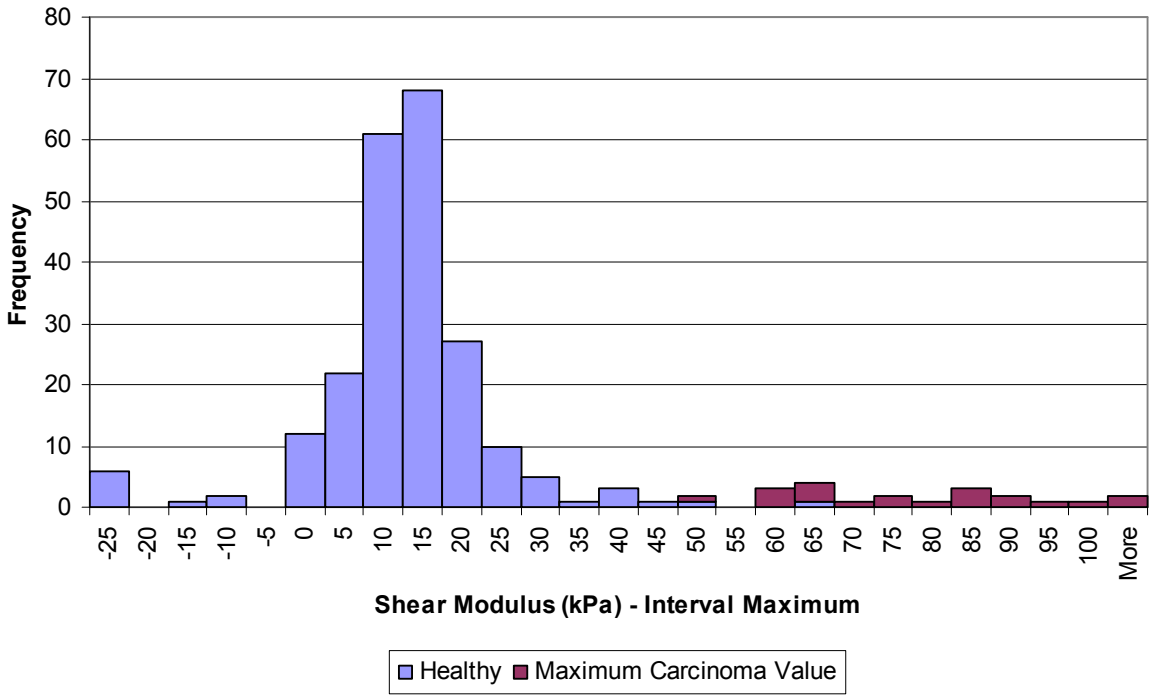

Figure 5.16 - Shear Modulus Distributions of Healthy and Cancerous Tissue from the Global Double Integral Inverse Algorithm using Mesh Refinement

However, one way around this problem that avoids mesh refinement would be to change the alignment of the grid a number of times until it lines up more approximately with the tumour. For example, in order to get a 2mm resolution of position on a carcinoma of 1cm size and for a domain that is 0.1cm long, the global double integral inverse algorithm could be applied on 5 different discretizations of the grid defined as follows:

$$D_1 = \{[0,1],[1,2],\ldots,[9,10]\}$$
$$D_i = \{[0.2i,0.2i+1],[0.2i+1,0.2i+2],...,[0.2i+8,0.2i+9]\}, \quad i = 2,...,5 \tag{5.1}$$

where D_i is the set of discretized segments used for each iteration of the inverse algorithm.

This approach implies that for some discretized grid, D_i, $i = 1,...,5$, the inverse algorithm will be sufficiently aligned with the carcinoma to give a clear indication of the increase in stiffness of the tissue and the position. The fact that the double integral method requires no forward simulation and can identify all the stiffness values with one very fast linear least squares solution ensures the method of Equation (5.1) will not have any significant effect on the overall computational speed or intensity of the algorithm.

5.3.7 Summary

The Global Double Integral inverse algorithm was found to be robust to the effects of random added noise. As the actuation frequency of the tissue simulation is decreased the accuracy of the algorithm also decreases due to the reduction of spatial information. An increase in the data resolution of the motion data input results in reduced variability of the calculated carcinoma stiffness resulting in increased positive discrimination of stiffness values. A reduction of carcinoma stiffness to the lower limits of expected Young's Modulus values, as discussed in Section 5.1 results in increased performance of the algorithm due to the increased spatial information within the carcinoma segment.

These results are for the assumed mass and stiffness system parameters used in forward simulation. These parameters define the natural frequency and therefore the response of breast tissue. Thus, a change in these parameters could alter the dynamic response so that the inverse algorithm is potentially less accurate for certain actuation frequencies.

Successful identification of the carcinoma is not always guaranteed when its position is varied throughout the global domain. This loss of accuracy occurs because the carcinoma can be positioned in this 1D example in such a way that it does not significantly alter the global motion dataset in comparison to a homogeneous model. These cases are therefore geometrically ill-

conditioned and not likely to occur in a physiologically realistic 3D case. However, a 10% variation in actuation frequency can alter the global motion dataset sufficiently for successful identification of the carcinoma, even in this simplified case.

A method of localized mesh refinement was developed as an adaptation to the Global Double Integral inverse algorithm. This method was developed to identify carcinoma not aligned with the discretized grid and carcinoma smaller than the discretized grid. It proved to be relatively unsuccessful as the reduction of grid size can create false positives within the solution as the segment length becomes too small to accurately identify the healthy tissue stiffness due to the lack of spatial information. However, a method was proposed to identify carcinoma not aligned with the discretized grid that involves multiple iterations of the inverse algorithm where the position of the grid is moved incrementally for each iteration, such that the grid will approximately align with the tumour at some iteration.

6 Performance of 2D Inverse Algorithms

6.1 Verification of Forward Simulation Algorithm

6.1.1 Verification against Analytical Homogeneous Solution

The homogeneous forward simulation algorithm is initially verified by comparison against the analytical solution that exists in the infinite domain. Thus, the basic foundations of the algorithm can be checked against an accurate analytical solution before more physically realistic boundary conditions are applied. Figures 6.1 and 6.2 show the convergence of the numerical solution to the analytical solution with actuation frequencies of 50Hz and 100Hz. The Young's Modulus of the healthy tissue is set at 30kPa, as justified in Section 5.1.

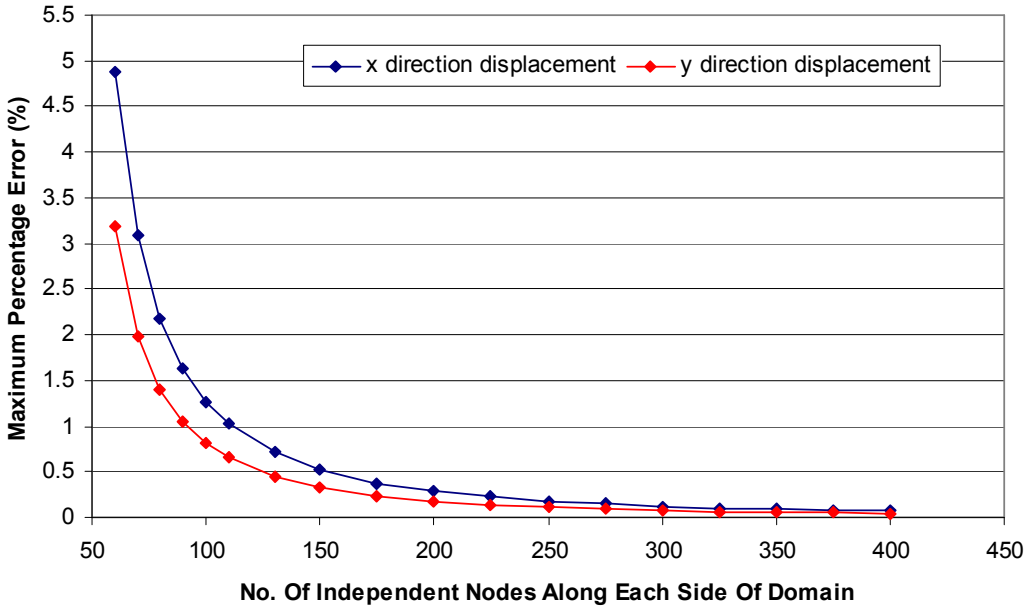

Figure 6.1 - Maximum Error in the Simulated Solution compared against the Analytical Solution at an actuation frequency of 50Hz

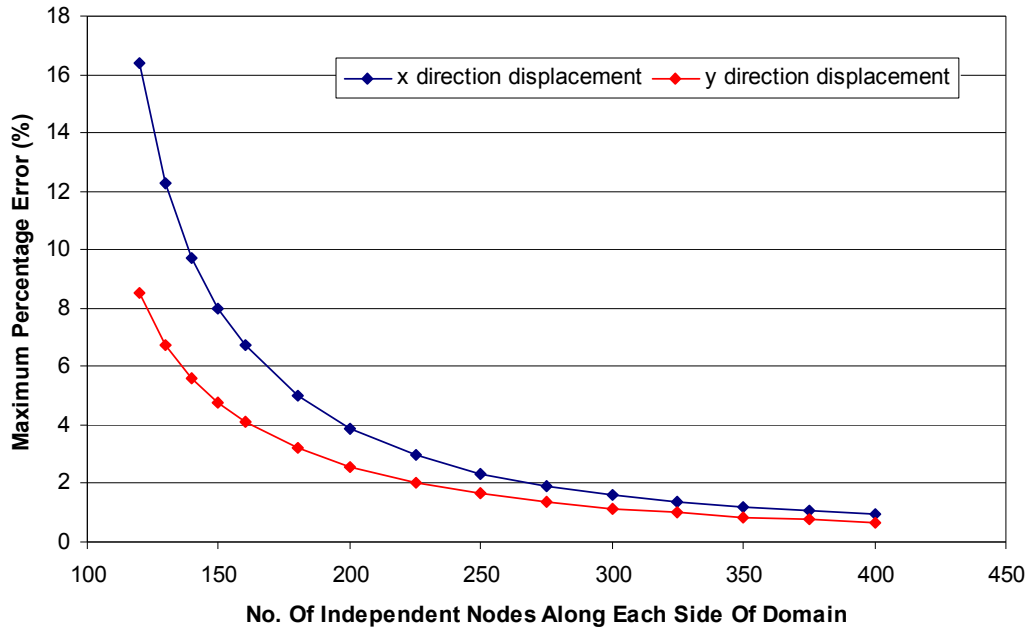

Figure 6.2 - Maximum Error in the Simulated Solution compared against the Analytical Solution at an actuation frequency of 100Hz

The results in Figures 6.1 and 6.2 show that the finite difference based forward simulation algorithm can accurately model the analytical solution. The convergence of the solution at an actuation frequency of 100 Hz is much slower than at 50Hz because the greater actuation frequency produces a more dynamic response with higher spatial frequencies.

6.1.2 Convergence of Forward Simulation Algorithm

For both the homogeneous and non-homogenous case, there exists no known analytical solution for the 2D plane strain problem of Equation (3.1) with arbitrary boundary conditions. Therefore, the numerical forward simulation of these cases has to be verified by relative convergence of the solution with successive grid refinement. In this case, small changes at successive refinements indicate convergence.

Resonance can be a problem with the forward simulation of the 2D plane strain simplification of Navier's equations in Equation (3.1). Specifically, the model does not include damping. The presence of undamped resonance within the motion dataset can severely reduce the ability of the forward simulation model to converge within the maximum resolution of data points, the number

of which is limited by the memory constraints of the computer where the simulation is being run. However, if a resonant model does converge successfully, then the inverse algorithm should still successfully solve for the stiffness as the fundamental dynamics of the model are still satisfied.

Figure 6.3 shows the convergence of the homogeneous model with the 'Phantom' boundary conditions of Section 3.2 applied. The error in the solution is measured based on the maximum percentage difference of the motion dataset between the current mesh and the previous mesh. The Young's Modulus of the healthy tissue is set at 30kPa and the excitation frequency is 100Hz. At 250 nodes the forward simulation converges within 1%, and as the number of nodes increase, the error decreases exponentially. More specifically, after 250 nodes the forward solution becomes mesh independent.

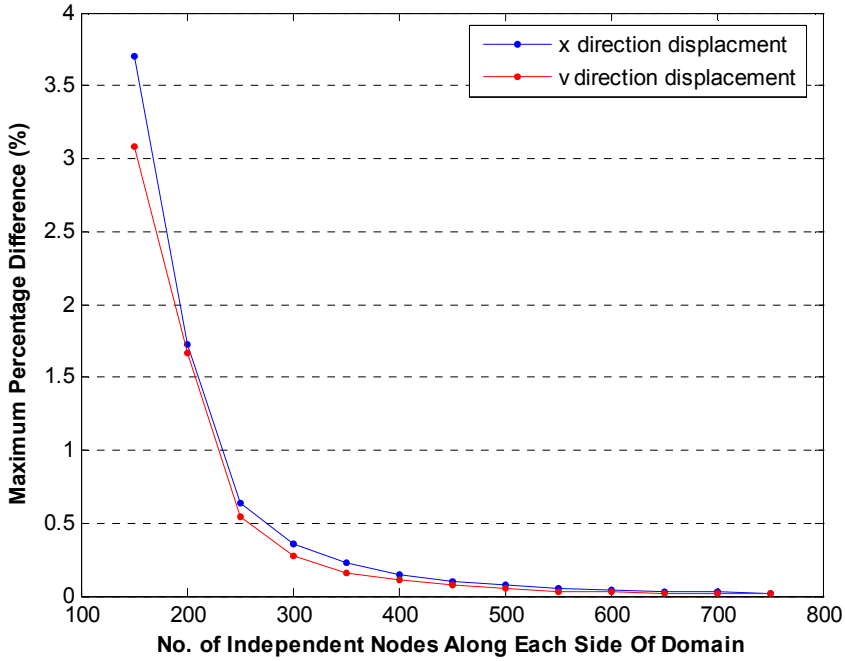

Figure 6.3 – Maximum Percentage Difference of the Motion Dataset between Current Mesh and Previous Mesh for the Homogeneous Model using 'Phantom' Boundary Conditions Actuated at 100Hz

Figure 6.4 shows the convergence of a non-homogeneous model with the 'Phantom' boundary conditions applied. The carcinoma is positioned at (7,9) as described in Figure 4.3 and has a Young's Modulus of 300kPa. This forward simulation model is the primary model used to evaluate the non-homogeneous inverse algorithm.

The convergence of this model is not as clear and consistent as in the homogeneous case. This less clear result is due to the interactions between the carcinoma and healthy tissue. In this case,

it is not possible to perform further mesh refinement to the model to conclusively show that the forward simulation solution has converged and is mesh independent because of memory limitations of the computers available (maximum of 4096Mb of RAM). Therefore, although not conclusive, the model of Figure 6.4 is considered to have converged sufficiently to provide an input to the non-homogeneous inverse problem algorithm.

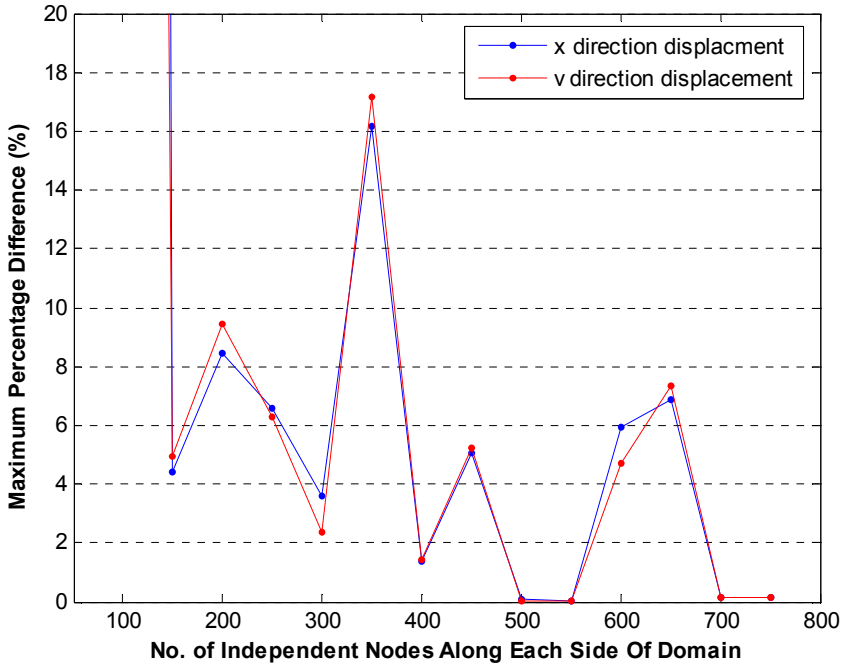

Figure 6.4 – Maximum Percentage Difference of the Motion Dataset between Current Mesh and Previous Mesh for the Non-Homogeneous Model using 'Phantom' Boundary Conditions Actuated at 100Hz

However, this result is not always the case for the non-homogeneous forward simulation solutions. Using excitation frequencies of 50Hz and approximately 100Hz, less than half of all possible geometric cases converged sufficiently. Note that the criterion for convergence is that at least two successive iterations fall below 1% before reaching the maximum of 750 nodes. In particular, it was found that where the forward simulation solution was more likely to converge is when the carcinoma is placed in positions (i,6) and (i,9) of Figure 4.4 where $i = 2,...,9$. Note that for simplicity in the forward simulation code the carcinoma was not placed on the boundary of the global domain corresponding to $i = 1,10$.

Conversely, the solutions don't converge when the carcinoma is placed in positions (i,7) and (i,8) where $i = 2,...,9$. This criterion consists of all the possible carcinoma positions using the given boundary condition models described in Section 3.2. In particular, all other possible geometrics

are permutations of existing ones once symmetry is taken into account. Therefore, only forward simulation models with carcinoma placed at positions (i,6) and (i,9) where $i = 2,...,9$ are used as input to test the inverse algorithm. The convergence of the forward simulation solutions of the non-homogeneous models is relatively inconclusive and a computer with greater memory capacity is required to investigate them comprehensively. In all cases, the inverse algorithms are being tested against the input properties and full convergence or accuracy is not required to test the reconstruction.

As an example, Figure 6.5 shows the solution for the non-homogeneous forward simulation. The 'Phantom' boundary conditions are applied and the tissue is actuated at 100Hz. The carcinoma is located at position (7,9) and is 10 times stiffer than surrounding tissue. The results are qualitatively consistent with finite element results and expected behaviour.

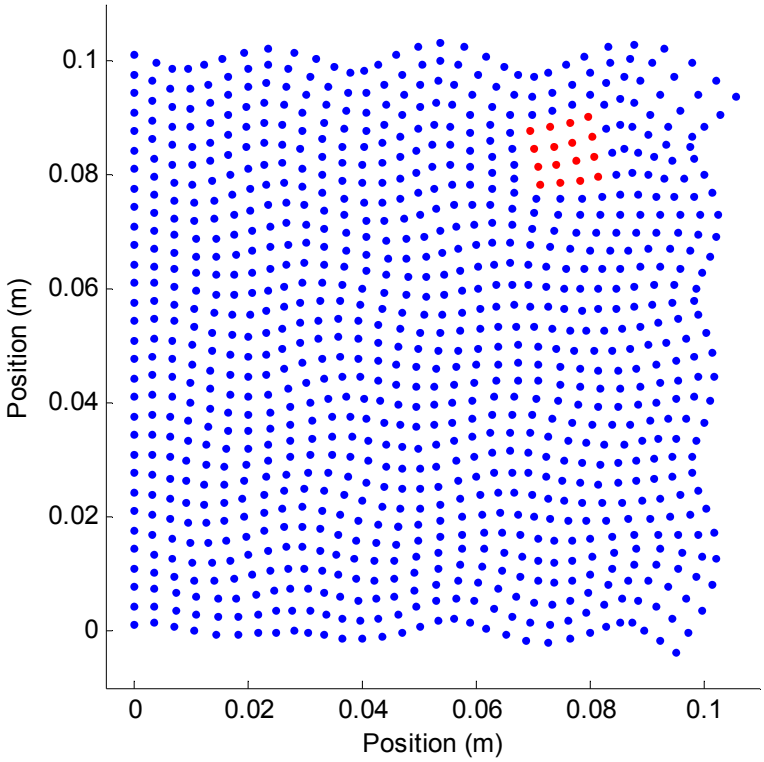

Figure 6.5 – The solution to the 2D forward simulation algorithm at maximum displacement. The carcinoma is shown in red

6.2 Comparison of Homogeneous Inverse Algorithms

The initial inverse algorithm of Equations (4.14)-(4.22) and the centred base point inverse algorithm Equations (4.27)-(4.32), are evaluated and compared to establish the best method for further development. The global motion data output from the forward simulation requires post-processing before providing the input to the inverse algorithms. In particular, the number of data points per centimetre is reduced to correspond to a level similar to what can be achieved using current MRI technology, as discussed in Section 5.2. A base value of 20 points per centimetre, which is equivalent to 20x20 or 400 points per each discretized element, is used for the initial comparison of integral methods. For the forward solution motion data that was calculated using a number of nodes which is not a multiple of 400, cubic spline interpolation was performed to obtain the motion at each 'measured' data point.

The steps used to generate and validate the inverse algorithm are summarized as follows:

1. Random noise is also added to the motion data input in order to determine the robustness of each of the algorithms and provide a means of comparative analysis. This is calculated as a percentage of the median of the absolute values of the respective amplitude displacements from each Cartesian direction u and v.

2. The excitation frequency is 100Hz and the 'Box Shake' boundary condition model of Figure 3.1(a) is used.

3. For each set of parameters that the algorithms are evaluated, 20 separate runs of the algorithm are performed, each with different random noise added to the motion data input. This Monte Carlo approach provides a means of quantifying the median and range of each of the stiffness values reconstructed.

4. Using the cumulative data from the 20 runs a 90% confidence interval is used to evaluate the range of stiffness values calculated for each discretized element.

For both algorithms, an over-determined system of 20 equations in 1 unknown was formulated for the homogeneous case. For the initial algorithm described by Equations (4.14)-(4.22), the

bottom left corner of the sub-domain acts as the integral base point. Therefore, this point is varied to reduce the effects of a potentially ill-conditioned base point corrupting the solution, as shown in Figure 6.6(a). For the centred base point algorithm, the centre of the sub-domain acts as the integral base point. Therefore, this point is also varied as shown in Figure 6.6(b).

For both algorithms, the area of the sub-elements increases successively, as shown in Figure 6.6, until the final sub-element maps onto the entire global domain. These approaches may not represent the optimal methods for defining sub-elements to formulate independent equations. However, they are simplistic and give accurate representations of the performance of each algorithm for comparison.

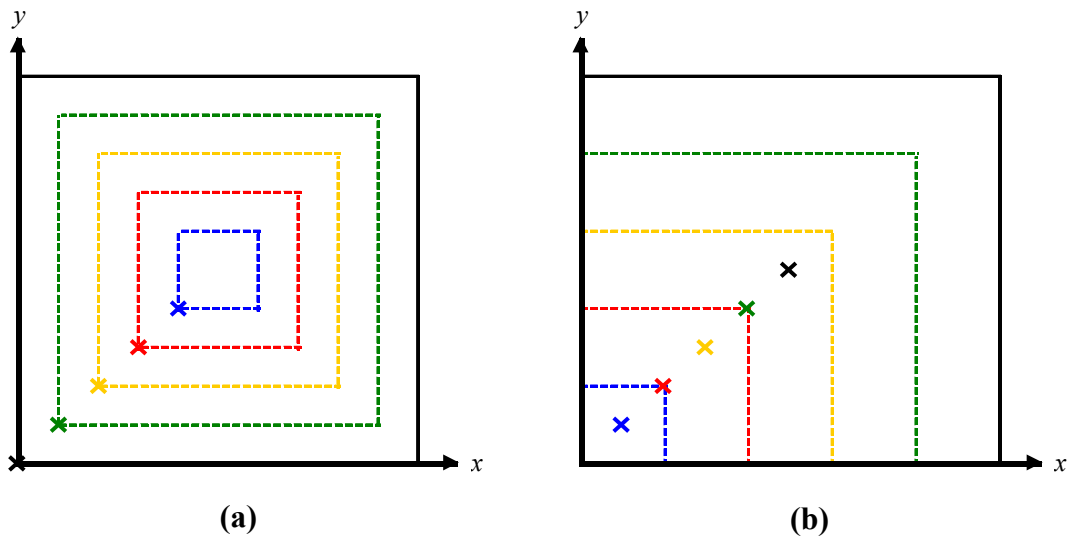

Figure 6.6 – Description of the sub-domains used to generate the integral equations for the Initial inverse algorithm (a) and the Centred Base Point Algorithm (b). The cross, ×, represents the base point and the boundary of the respective sub-domain is represented by a line of the same colour.

Figure 6.7 compares the relative performance of the two homogeneous inverse algorithms for the same given motion data input. As can be seen in Figure 6.7, the initial inverse algorithm is very sensitive to the applied noise. However, the centred base point algorithm shows very little effect to the increase in random added noise within the range shown.

This robustness occurs because the initial inverse algorithm requires the calculation of derivatives within the formulation of the equations. The derivatives are calculated using finite

difference approximations and are very susceptible to noise. These terms thus contribute the most significant proportion of error to the system of equations.

It is possible to calculate the derivative terms more accurately using techniques such as polynomial fitting or a low pass filter. However, they still remain a significant source of error within the solution and would also decrease the computational efficiency of the algorithm. The initial inverse algorithm also contains single integral terms within the formulation of the equations. Simulation has shown that single integral terms do not reduce the effects of noise as significantly as double integrals as they provide less effective low-pass filtering. Thus, their use, in comparison to double integral formulations, introduces further potential relative error into the algorithm.

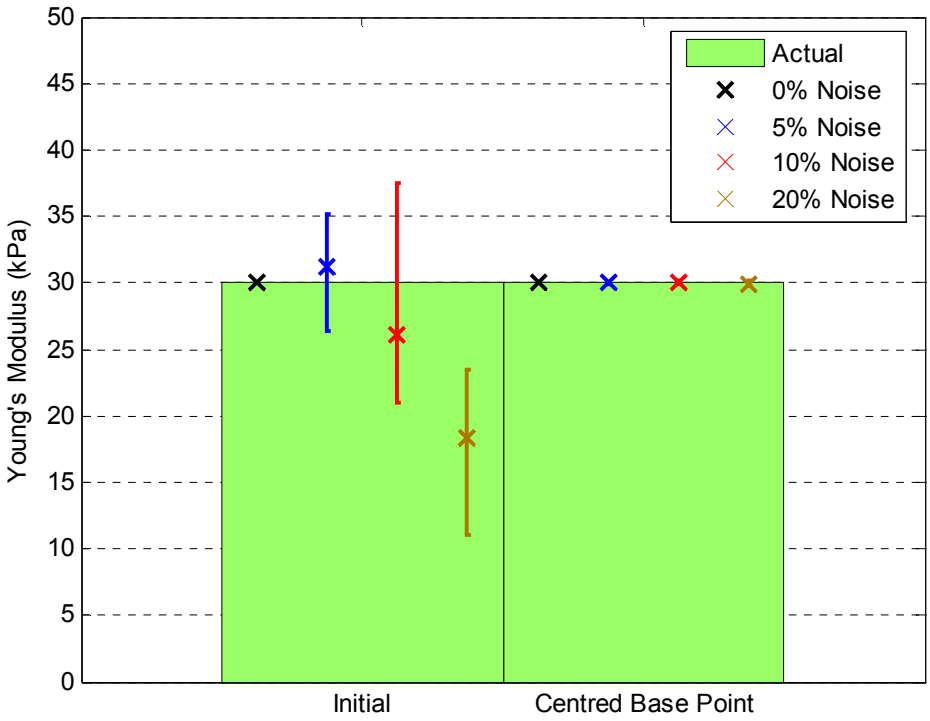

Figure 6.7 – Comparison between the Initial and Centred Base Point Inverse algorithms for the homogeneous case

The centred base point inverse algorithm, by comparison, does not require the calculation of derivatives and all integral terms calculated are double integrals. The algorithm thus much more successfully nullifies the effects of noise and accurately reconstructs the Young's Modulus of the homogeneous tissue. The development of a non-homogeneous inverse algorithm is therefore based on the integral techniques adopted in the centred base point inverse algorithm.

6.3 Evaluation of Non-Homogeneous Inverse Algorithm

The non-homogeneous inverse algorithm uses an integral method based upon the homogeneous centred base point inverse algorithm, as this formulation negates the need to calculate error prone derivative terms. It is evaluated to quantify its performance and robustness with the variation of important system parameters. These parameters are:

- The level of noise in the input data
- Frequency of the tissue actuation
- Level of carcinoma stiffness in relation to healthy tissue stiffness
- Resolution, or number of points per element of the input motion data
- Boundary conditions used to generate the input data
- Position of the carcinoma within the global domain
- The application of the stress continuity constraint model of Section 4.3

For each set of parameters evaluated, 20 separate runs of the algorithm are performed, each with different random noise added to the motion data input. Using the data from the 20 runs, the median and 90% confidence interval are calculated for the reconstructed healthy and cancerous stiffness values.

6.3.1 Variation with Random Added Noise

The level of randomly added noise significantly affects the performance of the inverse algorithm. Figure 6.8 shows the performance of the algorithm with respect to the level of random noise added to the motion data input. The motion input data used is defined by the 'Phantom' boundary conditions, has a carcinoma at position (7,9) and is excited at 100Hz. As the level of noise increases, the median Elastic Modulus calculated for the carcinoma decreases significantly. The confidence interval range of both the healthy and cancerous stiffness values also increase significantly as the level of added noise increases. The results are precisely the same behaviour as was observed for the 1D case in Section 5.2.1.

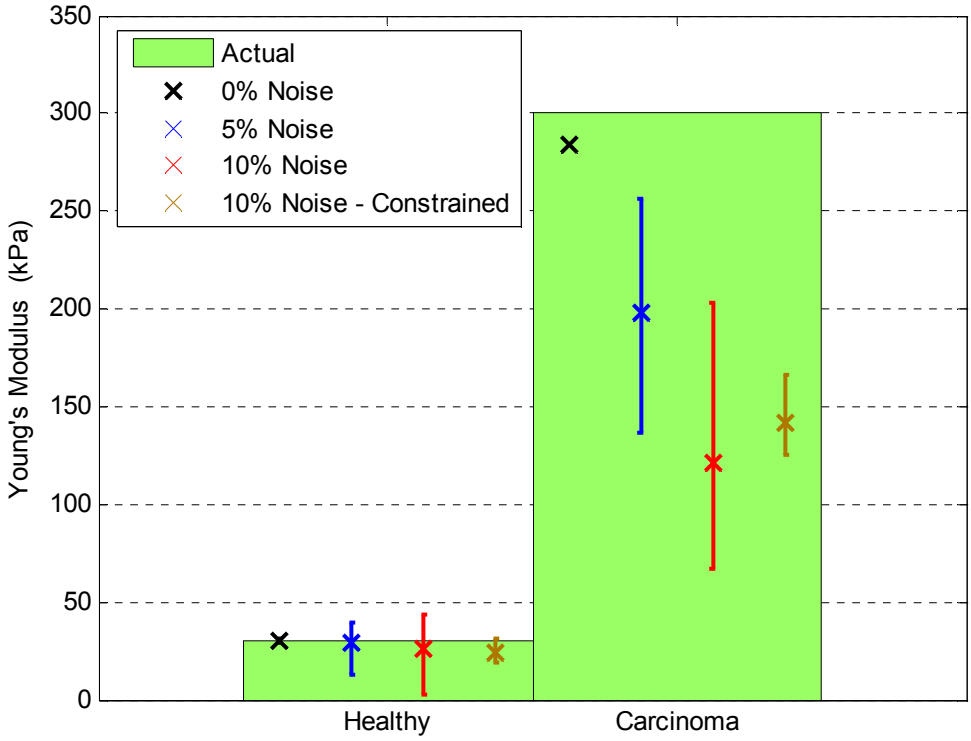

Figure 6.8 – The performance of the non-homogeneous inverse algorithm with random added noise. The motion data input used is defined by the 'Phantom' boundary conditions, has a carcinoma at position (7,9) and is excited at 100Hz.

The application of the simple constraint model based upon shear stress continuity between discretized elements can enhance the solution. Using the unconstrained algorithm, with 10% random added noise, the positive discrimination between healthy and cancerous stiffness values is not significant in comparison with the range of the carcinoma stiffness. However with the application of the constraint model, the accuracy of the carcinoma stiffness is increased and the range of both cancerous and healthy tissue stiffnesses are significantly reduced resulting in clear positive discrimination. This result is also illustrated in Figure 6.8.

6.3.2 Variation with Actuation Frequency

The actuation frequency is a significant parameter of the inverse problem as it directly relates to the spatial period of the shear and longitudinal waves induced in the 2D medium. It is limited by both the performance of the actuation system and the frequency of data collection of the MRI system used. Figure 6.9 shows the performance of the algorithm when the tissue simulation is

actuated at 50Hz. The 90% confidence interval of the calculated Young's Modulus is shown for the healthy tissue and carcinoma.

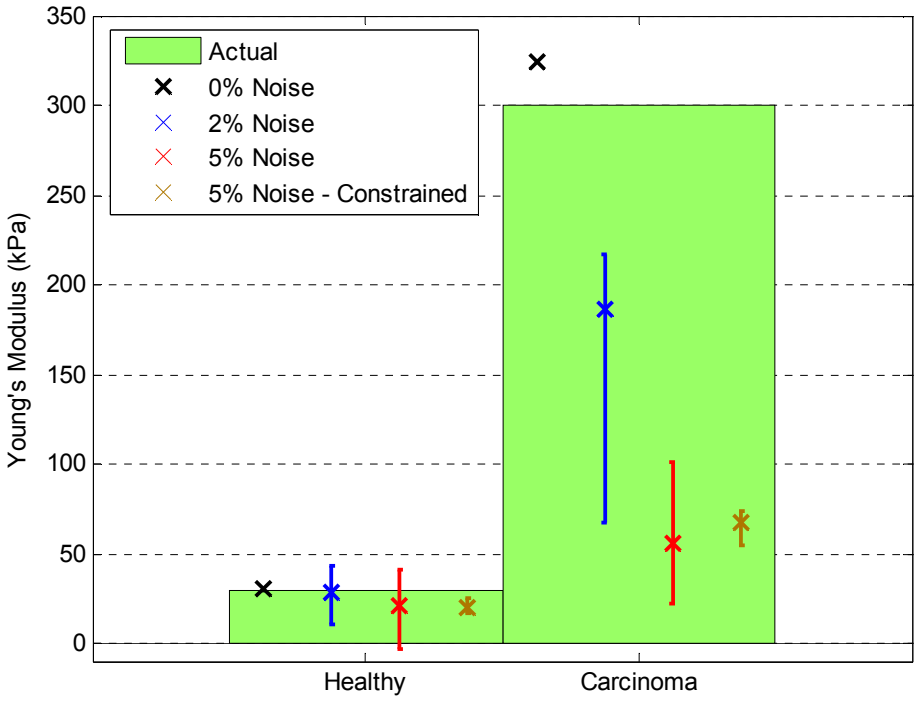

Figure 6.9 – The performance of the non-homogeneous inverse algorithm with random added noise. The motion data input used is defined by the 'Phantom' boundary conditions, has a carcinoma at position (5,6) and is excited at 50Hz.

By comparing the results of Figures 6.8 and 6.9 it can be seen that the non-homogeneous inverse algorithm is much less robust in the presence of noise at 50Hz compared with an actuation frequency of 100Hz. At 50Hz with 5% random added noise there is no positive discrimination between healthy and cancerous stiffness values. However, by using the constraint model, the accuracy of the calculated carcinoma stiffness is increased and the range of both healthy tissue and carcinoma stiffness is reduced. This approach results in positive discrimination of stiffness values and therefore a clear identification of the carcinoma.

The application of the Box Shake system of boundary conditions described in Section 3.2 results in solutions that are significantly more converged than using other boundary conditions. This difference results because this particular set of boundary conditions constrains the tissue more significantly than the other methods described. Therefore, using the Box Shake boundary conditions it is possible to simulate a non-homogeneous dataset at 150Hz that successfully

converges and thus direct comparisons can be made between the results of a greater range of actuation frequencies.

Figure 6.10 shows the performance of the algorithm with respect to actuation frequency for the Box Shake boundary conditions and 5% random added noise. The results show that as the actuation frequency is increased, the accuracy of the algorithm also increases. There is clear positive discrimination between healthy and cancerous tissue values at both 100Hz and 150Hz, with the results at 150Hz having a more accurate median value and a smaller 90% confidence interval range.

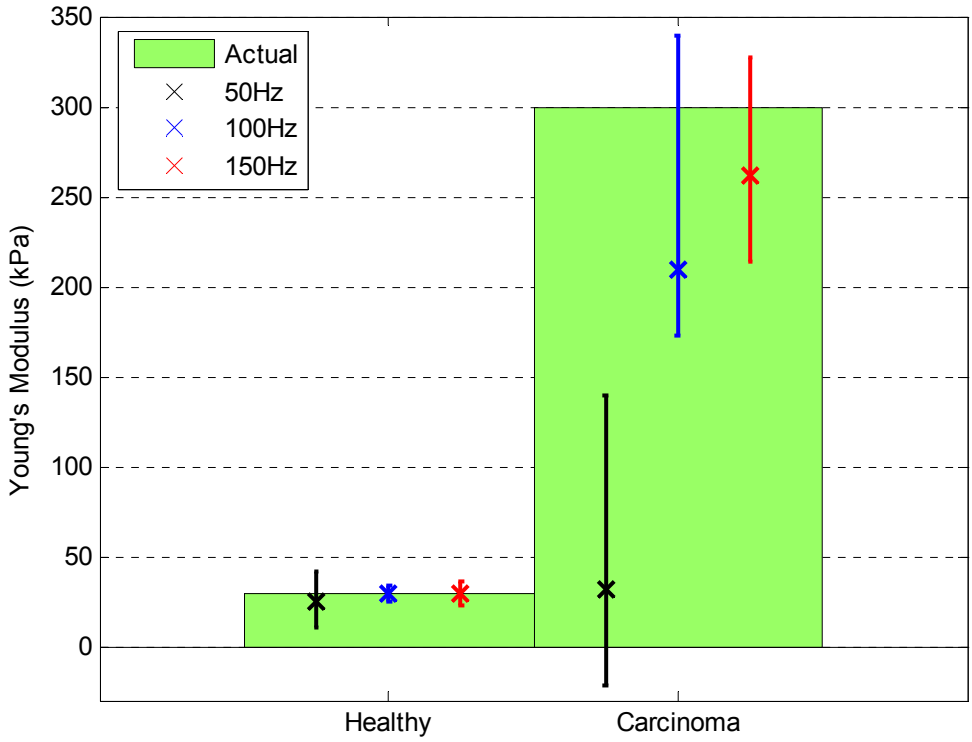

Figure 6.10 - The performance of the non-homogeneous inverse algorithm with variation of actuation frequency. The motion data inputs used are defined by the 'Box Shake' boundary conditions, have the carcinoma at position (7,9) and have 5% random added noise.

The spatial period of the actuated tissue is inversely proportional to the actuation frequency. Therefore, the proportion of sinusoidal waveforms per discretized element or 'spatial information' increases with an increase in the actuation frequency. The performance of the inverse algorithm is thus substantially enhanced. These results are for the assumed stiffness and mass properties of the simulation. Significant variation of these parameters would change the dynamic response and the inverse algorithm performance for that specific case.

6.3.3 Variation with Carcinoma Stiffness

The analysis of the Non-Homogeneous inverse algorithm so far, has used a carcinoma stiffness that is 10 times greater than the healthy tissue. Figure 6.11 evaluates the algorithm with the carcinoma to healthy tissue stiffness ratio of 5:1, as this level approximately represents the lower limit of this ratio in both static and dynamic testing [Samani et al. 2003, Krouskop et al. 1998]. The system parameters used in Figure 6.11 are the same as for Figure 6.8.

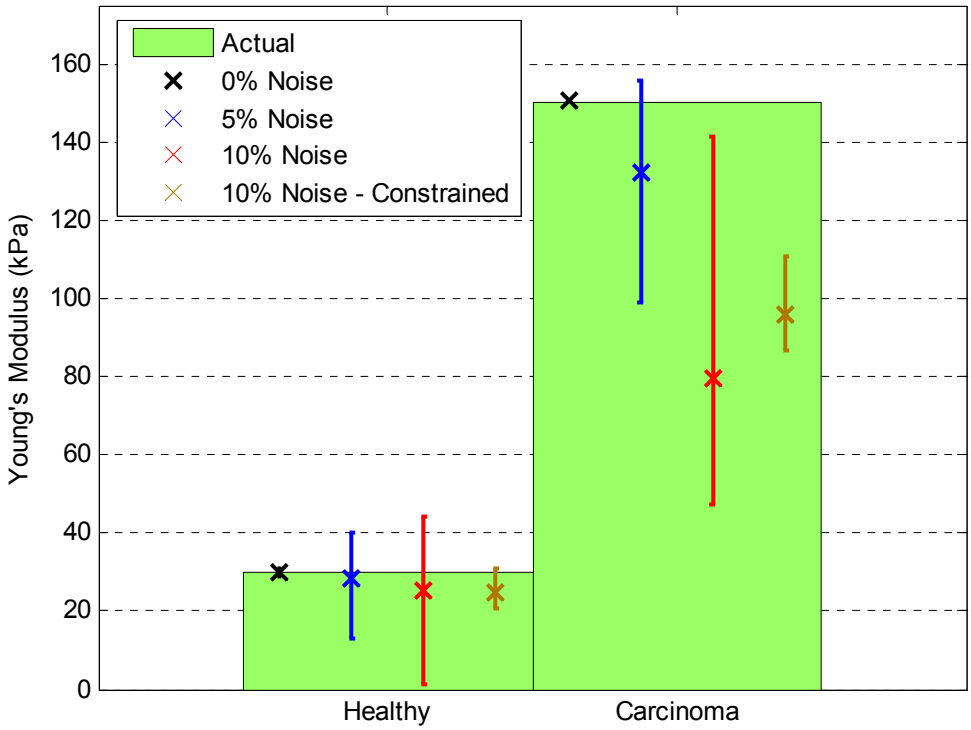

Figure 6.11 - The performance of the non-homogeneous inverse algorithm with carcinoma Young's Modulus of 150kPa and random added noise. The motion data input used is defined by the 'Phantom' boundary conditions, has the carcinoma at position (7,9) and is actuated at 100Hz.

The results show that the algorithm more accurately reconstructs the median healthy and cancerous Elastic Modulus values for the lower stiffness ratio contrast of 5:1 than for the stiffness ratio of 10:1. This result occurs because the spatial period of motion data in a homogeneous element is inversely proportional to the square root of the stiffness. As the spatial period of the carcinoma is reduced there is more spatial information available per segment, which results in identifying the Shear Modulus more accurately. However, the relative positive

discrimination between healthy and cancerous stiffness is very similar to the 10:1 stiffness ratio because the lower stiffness ratio, by definition, effectively reduces the positive discrimination between healthy and cancerous tissue stiffness. Thus, the algorithm behaves similarly within the given 5-10 times stiffness contrast of ductal carcinoma.

6.3.4 Variation with Data Resolution

The data resolution parameter is limited by the magnetic strength of the MRI system used to produce the motion dataset, and is discussed in detail in Section 5.2. Figure 6.12 shows the performance of the algorithm with variation of the data resolution of the input motion dataset and with 5% random added noise. As the resolution of the input data increases, the calculated stiffness values increase in accuracy and the range of the 90% confidence intervals are reduced. With 10 points per segment there is very little positive discrimination between stiffness values, but clear identification of the carcinoma is possible with 20 points per segment and greater. The reason for the increase in accuracy is that the more data points the more the noise will tend to cancel or be filtered by the integrations employed.

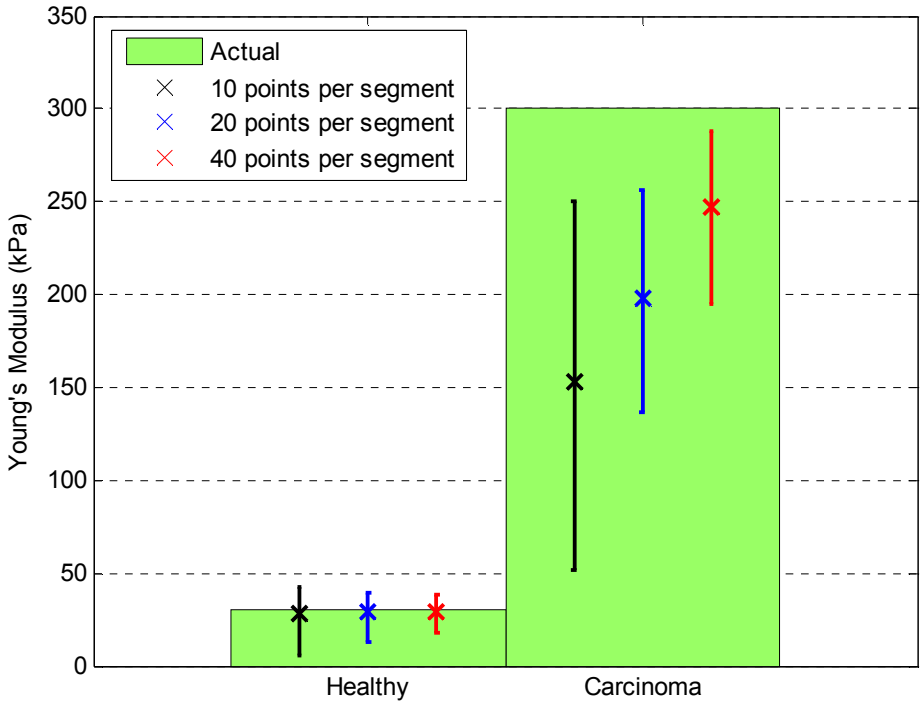

Figure 6.12 – The performance of the non-homogeneous inverse algorithm with variation of data resolution. The motion data input used is defined by the 'Phantom' boundary conditions, has the carcinoma at position (7,9), is actuated at 100Hz and has 5% random added noise.

The performance of the algorithm with respect to data resolution and with 10% random added noise is detailed in Figure 6.13. The results are similar to the 5% noise case of Figure 6.12, but the accuracy and positive discrimination of stiffness values are compromised more due to the effects of the increased noise.

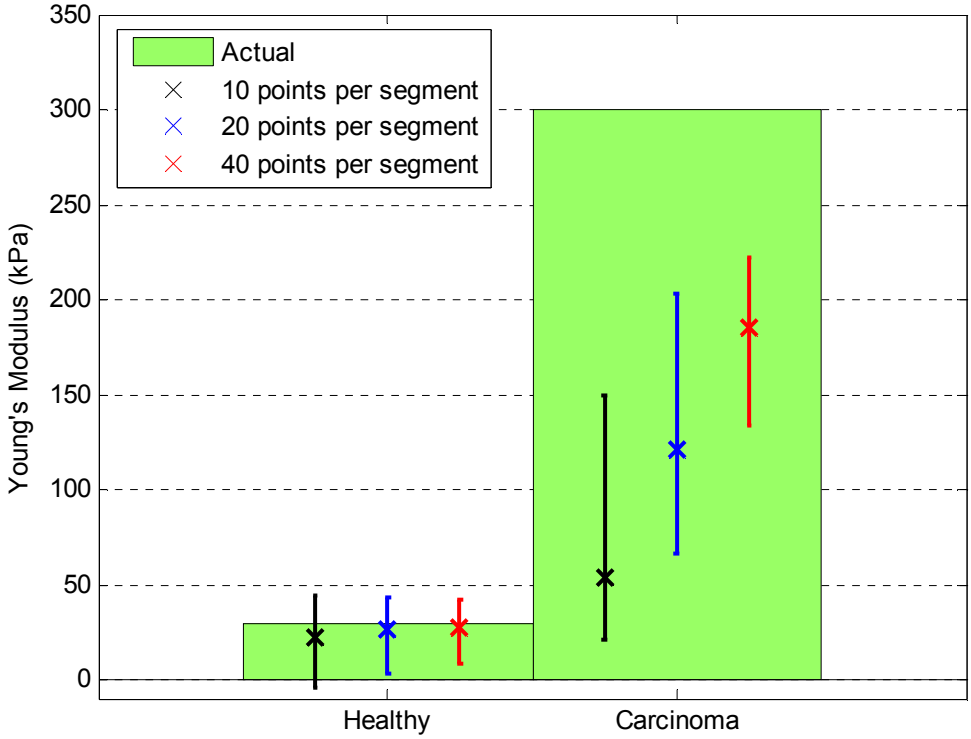

Figure 6.13 - The performance of the non-homogeneous inverse algorithm with variation of data resolution. The motion data input used is defined by the 'Phantom' boundary conditions, has the carcinoma at position (7,9) , is actuated at 100Hz and has 10% random added noise.

6.3.5 Variation with Carcinoma Position and Boundary Conditions

The non-homogeneous inverse algorithm should provide accurate identification of carcinoma within healthy tissue regardless of its position. The performance of the algorithm for different carcinoma positions is thus investigated including the variation in results for different boundary conditions applied to the tissue simulation, as described in Section 3.2. Figures 6.14-6.16 show that although there are deviations in the median and 90% confidence interval range of the stiffness values calculated for each of the combinations of carcinoma position and applied boundary condition, positive discrimination between healthy and cancerous stiffness values is

maintained. This set of results show that the algorithm is universally applicable given the 5% random noise added to the input motion data.

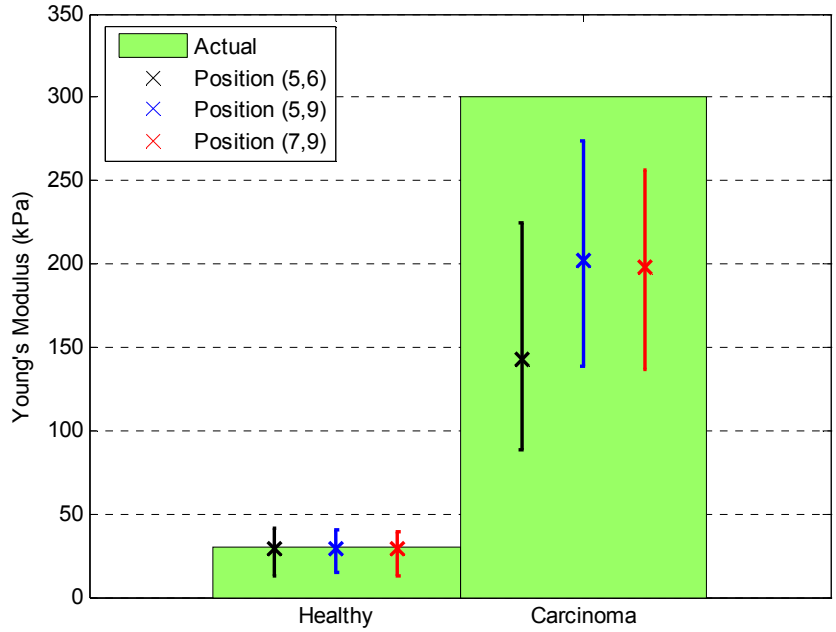

Figure 6.14 – The performance of the non-homogeneous inverse algorithm with variation of carcinoma position. The motion data input used is defined by the 'Phantom' boundary conditions, is actuated at 100Hz and has 5% random added noise.

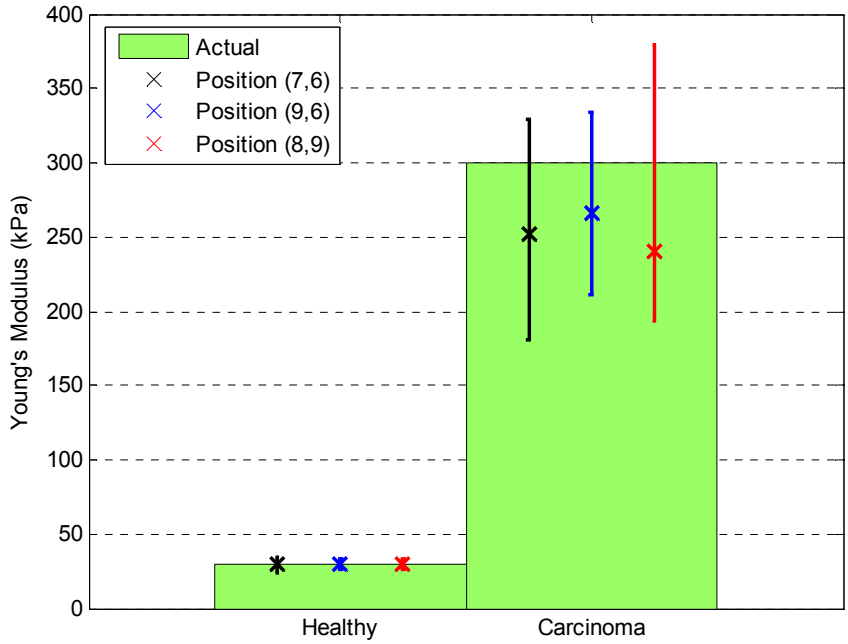

Figure 6.15 – The performance of the non-homogeneous inverse algorithm with variation of carcinoma position. The motion data input used is defined by the 'Box Shake' boundary conditions, is actuated at 100Hz and has 5% random added noise.

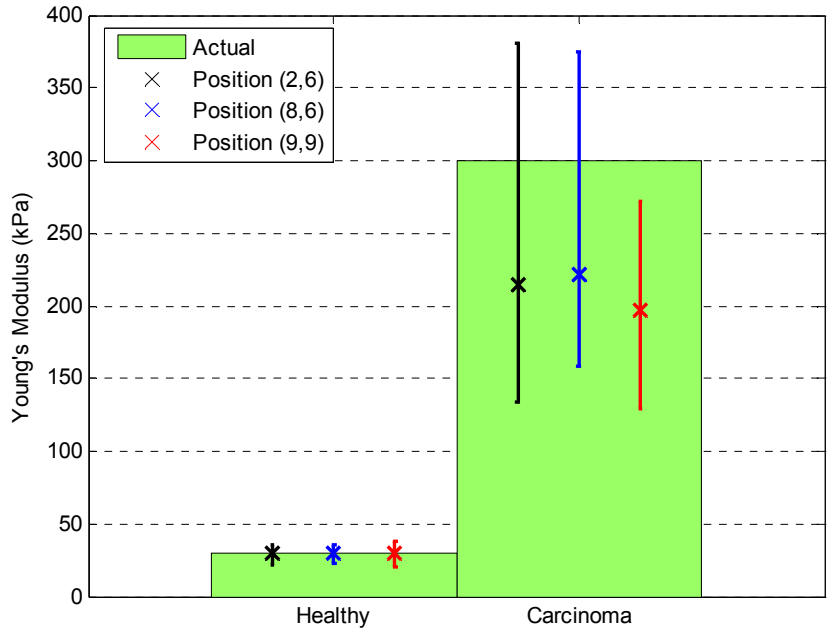

Figure 6.16 – The performance of the non-homogeneous inverse algorithm with variation of carcinoma position. The motion data input used is defined by the 'Edge Effect' boundary conditions, is actuated at 94Hz and has 5% random added noise.

6.4 Carcinoma Identification using 1D Inverse Algorithm

In this section, a different approach to the inverse problem is considered by investigating the ability of a very simplified model to detect cancer. The 1D Global Double Integral inverse algorithm is used to identify a carcinoma within a 2D simulated tissue sample by taking slices of either the x direction or the y direction displacement amplitudes (u or v respectively). The motivation for this approach is to get some idea on the feasibility of using a 2D model to detect cancer from a 3D simulated dataset. Using a 1D model to detect cancer in a 2D model would provide a very crude proof of this concept.

The 1D model only accounts for the shear wave displacement and is very simplistic compared to the 2D model that accounts for the shear and longitudinal waves in two dimensions. However, Figure 6.17 shows that it is still possible to successfully identify a carcinoma by taking a slice of the simulated 2D displacement amplitude solution and using it as input to the 1D non-homogeneous inverse algorithm. Note that the values shown are the absolute values of the

reconstructed Shear Modulus values as the calculated carcinoma stiffness was found to be always negative for this case.

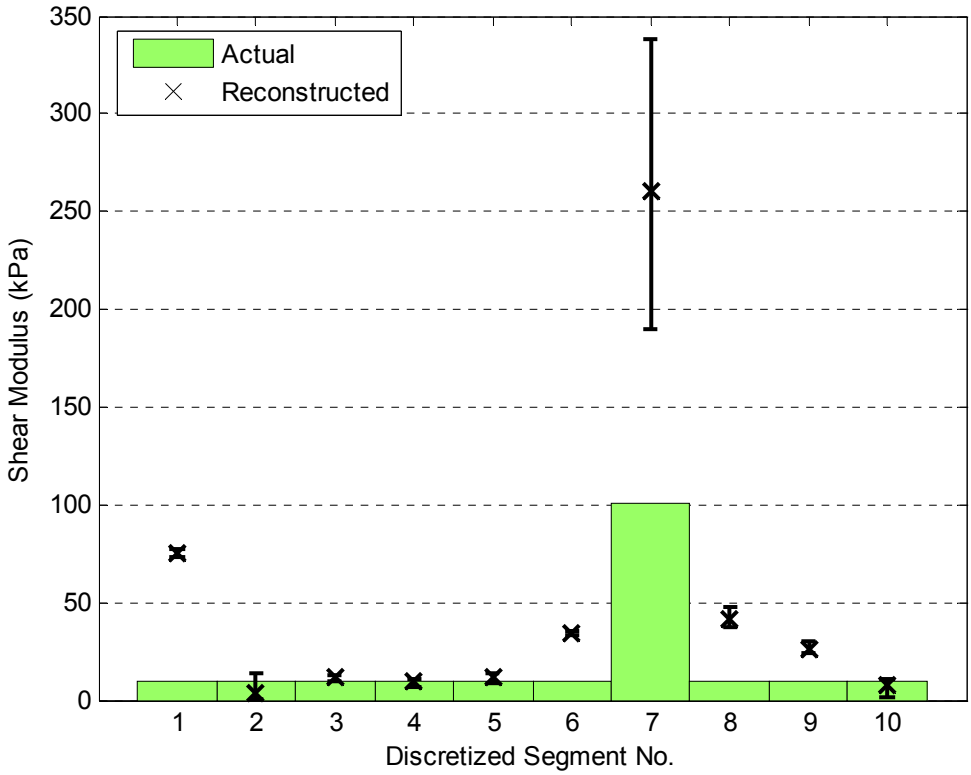

Figure 6.17 – 90% Confidence Interval of the Absolute Values of the Reconstructed Stiffness Distribution using a horizontal slice of the x direction displacement amplitude. The motion data input used is defined by the 'Phantom' boundary conditions, is actuated at 100Hz and has 5% random added noise.

Each 2D motion dataset has four possible inputs to the 1D inverse algorithm, the vertical and horizontal slices of the x direction and the y direction displacement amplitudes. Initial investigations showed that clear positive identification of the carcinoma is possible from at least one of these motion data inputs, and in some cases two or more motion data inputs provided positive results. These investigations include a range of applied boundary conditions and carcinoma positions. The results are not given here as they are very preliminary and not the focus of this thesis. However, they strongly suggest a possible avenue for future development.

6.5 Non-Homogenous, Incompressible Inverse Algorithm

The idea of applying the 1D inverse algorithm to slices of data to act as a simplified 2D inverse algorithm is taken further by modifying the existing 2D non-homogeneous inverse algorithm. The Poisson's ratio of breast tissue is approximately 0.49, which is virtually incompressible. Therefore, the existing 2D inverse algorithm is modified so that it assumes that the elastic material is incompressible. However, the 'measured' data generated from forward simulation is still based on the compressible Navier's equations with $\nu = 0.49$. Incompressibility is equivalent to zero divergence in the displacement field:

$$u_x + v_y = 0 \tag{6.1}$$

The compressible 2D Navier's Equations (4.1) and (4.2) can be written in the form:

$$\frac{\partial}{\partial x}\left(Gu_x\right) + \frac{\partial}{\partial y}\left(Gu_y\right) + \frac{\partial}{\partial x}\left((\lambda + G)(u_x + v_y\right) = -\rho\omega^2 u \tag{6.2}$$

$$\frac{\partial}{\partial x}\left(Gv_x\right) + \frac{\partial}{\partial y}\left(Gv_y\right) + \frac{\partial}{\partial y}\left((\lambda + G)(u_x + v_y\right) = -\rho\omega^2 v \tag{6.3}$$

Substituting Equation (6.1) into Equations (6.2) and (6.3) gives the incompressible Navier's equations defined as follows:

$$\frac{\partial}{\partial x}\left(Gu_x\right) + \frac{\partial}{\partial y}\left(Gu_y\right) = -\rho\omega^2 u \tag{6.4}$$

$$\frac{\partial}{\partial x}\left(Gv_x\right) + \frac{\partial}{\partial y}\left(Gv_y\right) = -\rho\omega^2 v \tag{6.5}$$

Note that Equations (6.4) and (6.5) can also be obtained from Equations (6.2) and (6.3) by the substitution $\lambda = -G$. Thus, the coefficients of the integral terms used for the 2D non-homogeneous inverse algorithm in Equations (4.57)-(4.66) can be reformulated with no further

adaptation required to the existing code. These coefficients, previously defined in Equations (4.44)-(4.46), are now defined, following the substitution of $\lambda = -G$, as follows:

$$a_1 = \frac{\lambda_{ij} + G_{ij}}{E_{ij}} = \frac{-G_{ij} + G_{ij}}{E_{ij}} = 0 \tag{6.6}$$

$$a_2 = \frac{\lambda_{ij} + 2G_{ij}}{E_{ij}} = \frac{-G_{ij} + 2G_{ij}}{E_{ij}} = \frac{G_{ij}}{E_{ij}} = \frac{1}{2(1+\nu)} \tag{6.7}$$

$$a_3 = \frac{G_{ij}}{E_{ij}} = \frac{1}{2(1+\nu)} \tag{6.8}$$

The a_1 coefficient of Equation (6.6) is now zero, thus the incompressible method requires the formulation of fewer terms than the compressible method, and thus less computation.

6.5.1 Variation with Random Added Noise

Figure 6.18 shows the performance of the incompressible 2D non-homogeneous inverse algorithm, using 10 data points per segment and an actuation frequency of 100Hz, with respect to the level of random noise added to the motion data input. The results of Figure 6.18 can be compared to the compressible model for 10 data points per segment in Figure 6.13, as the same geometric conditions and actuation frequency are adopted. At 40% noise there is clear discrimination between healthy and cancerous stiffness values, however for the compressible model there is no clear discrimination at only 10% noise. The decreased accuracy of the reconstructed healthy tissue stiffness at 0% noise is due to modelling error.

The reason for this significantly increased robustness to noise is that the integral equations formulated using the incompressible method exhibit increased numerical stability. Specifically, the double integral terms of $v(x, y)$ in Equation (4.43), given by Equation (6.9) can cancel out creating ill-conditioning in the integral equations for the compressible model case. In other words, for the compressible case, noise could switch the sign of Equation (6.9), thus corrupting the solution. However, for the incompressible case, $a_1 = 0$. Therefore, these ill-conditioned

terms disappear from the incompressible integral formulation of Equations (6.4) and (6.5) as shown in Equation (6.9).

$$a_1\left[\left(E_{11}\int\int v + E_{22}\int\int v\right) - \left(E_{21}\int\int v + E_{12}\int\int v\right)\right] \tag{6.9}$$

Also, for near incompressible materials the value of the coefficient a_3 in Equation (4.44) is approximately 1/50 the size of coefficients a_1 and a_2 from Equations (4.45) and (4.46). Thus, the terms with a coefficient of a_3 do not significantly contribute to the solution, as compared to terms with a coefficient of a_1 or a_2, even if they are relatively accurate. The implementation of the incompressible model therefore eliminates the ill-conditioned terms and provides equal weighting to all integral terms. This change significantly increases the numerical stability and accuracy of the 2D inverse algorithm.

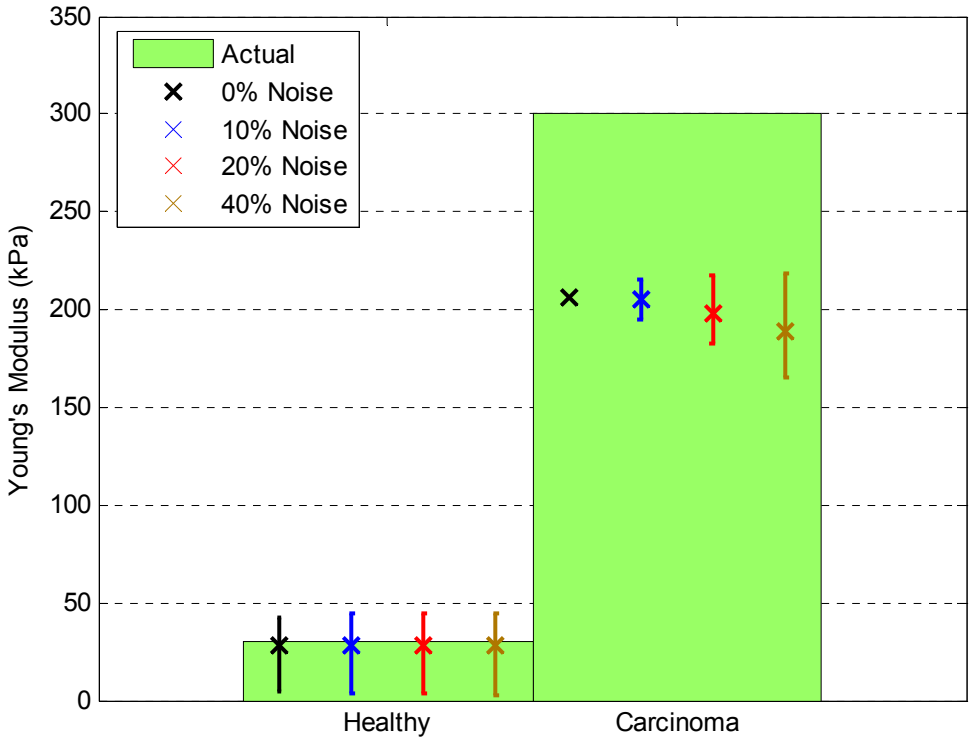

Figure 6.18 – The performance of the incompressible non-homogeneous inverse algorithm with random added noise. The motion data input used is defined by the 'Phantom' boundary conditions, has a carcinoma at position (7,9), uses 10 data points per segment and is excited at 100Hz.

The incompressible method is also performed at an actuation frequency of 50Hz. Figure 6.19 shows that the accuracy of the inverse algorithm also increases significantly at this relatively lower frequency. There is clear discrimination in the results between healthy and cancerous stiffness values with 20% noise in comparison a limit of 2% noise using the compressible method. This increase in accuracy is also attributable to the increased numerical stability of the algorithm.

However, the analysis for the actuation frequency of 50Hz ignores the stiffness distribution along the boundary of the global domain. This choice is made because there is significant modelling error in these regions. Initial simulation has shown that by allowing each divergence term in Equations (6.2) and (6.3) to be a constant piecewise function along the boundary can significantly reduce this modelling error. However, further analysis of this case is left to future work.

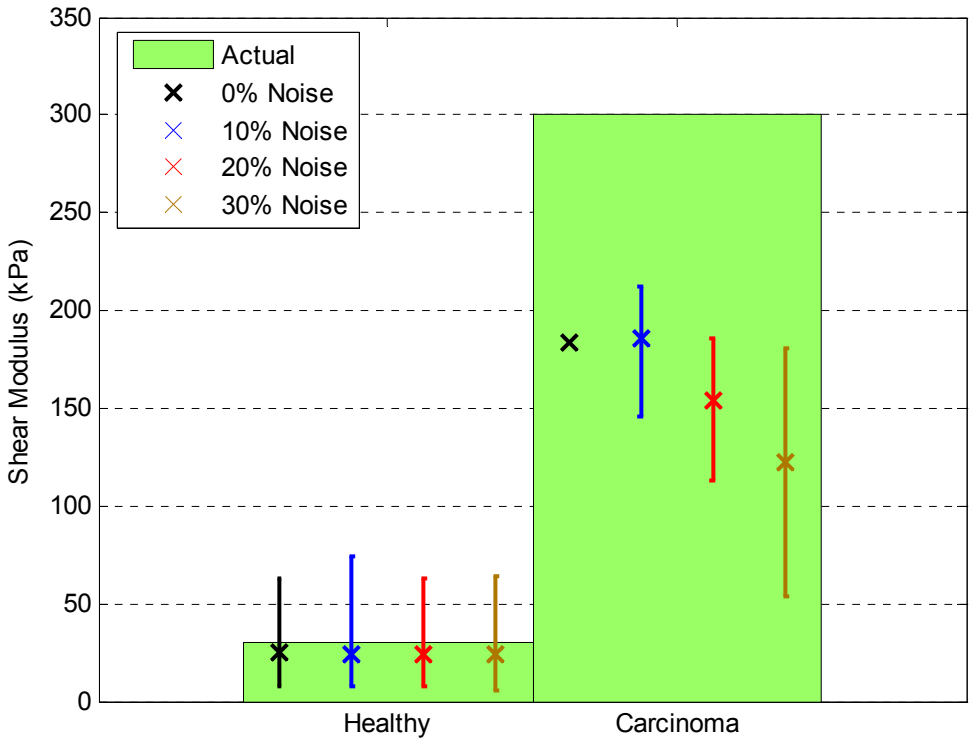

Figure 6.19 – The performance of the incompressible non-homogeneous inverse algorithm with random added noise. The motion data input used is defined by the 'Phantom' boundary conditions, has a carcinoma at position (5,6), uses 10 data points per segment and is excited at 50Hz.

6.5.2 Variation with Data Resolution

Figure 6.20 shows the performance of the incompressible algorithm with variation of the data resolution of the input motion data. The results show that the incompressible method also significantly enhances the robustness of the inverse algorithm to the level of data resolution. There is clear discrimination between healthy and cancerous stiffness values using only 2 points per segment. For a 2D $10\text{cm} \times 10\text{cm}$ global domain, this equates to a total of $21 \times 21 = 441$ global data points. This value is very small in comparison to the number of data points required for the compressible inverse algorithm, which was on the order of $201 \times 201 = 40401$ global data points (equivalent to 20 points per segment).

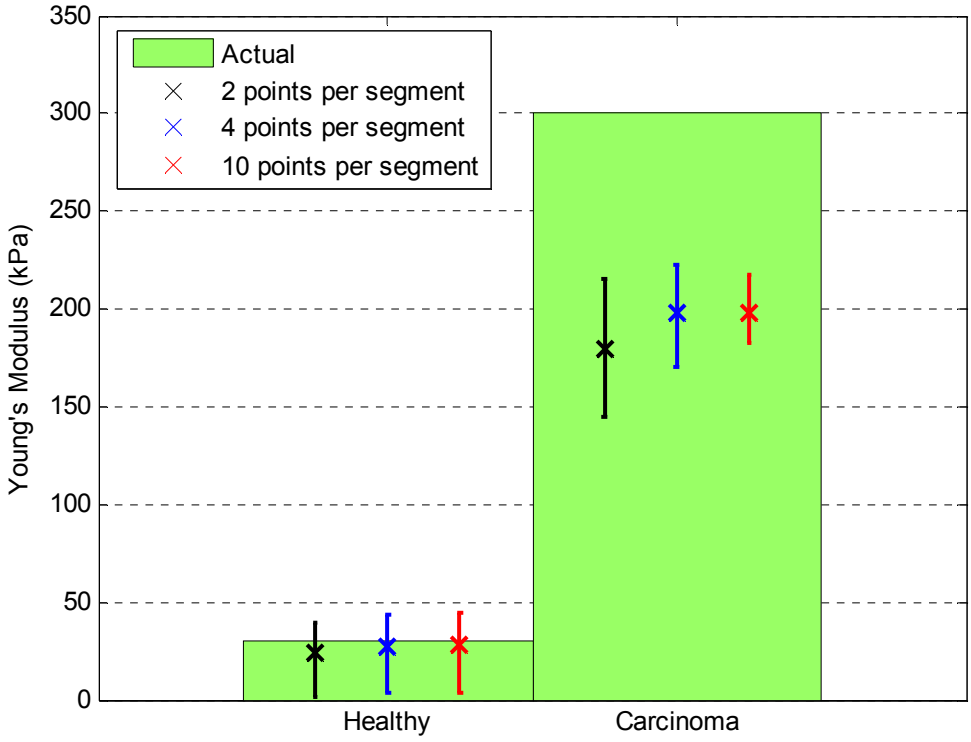

Figure 6.20 – The performance of the incompressible non-homogeneous inverse algorithm with variation of data resolution. The motion data input used is defined by the 'Phantom' boundary conditions, has the carcinoma at position (7,9), is actuated at 100Hz and has 20% random added noise.

The robustness of the incompressible inverse algorithm to both noise and a reduction in the data resolution opens up new possibilities for the measurement of the internal displacements. For example, a relatively cheap and portable Ultrasound device, which is typically very noisy, now presents a potential method of measuring the internal breast tissue displacements and obtaining a realistic reconstruction, instead of MRI.

6.6 Summary

The two 2D homogeneous inverse algorithms were evaluated and their relative performance was compared. It was found that the Centred Base Point method of Equations (4.27)-(4.32) was significantly more accurate than the Initial method of Equations (4.14)-(4.22). This difference occurs because the Initial method requires the calculation of derivatives that are sensitive to noise.

The 2D non-homogeneous inverse algorithm was evaluated by quantifying its accuracy and robustness with the variation of important system parameters. At an actuation frequency of 100Hz the algorithm consistently identifies the carcinoma for a range of positions and applied boundary conditions with up to 5% random added noise. This limit is increased to 10% if the simple shear stress continuity constraint model is adopted. At an actuation frequency of 50Hz the algorithm requires the use of the constraint model to successfully identify the carcinoma with up to 5% random noise added to the motion data input because the lower actuation frequency gives less spatial information per discretized element for this specific set of parameters.

An increase in the data resolution of the motion data increases the accuracy of the reconstructed carcinoma and reduces the 90% confidence interval range resulting in greater positive discrimination of stiffness values. At 10×10 points per element and with an actuation frequency of 100Hz the algorithm does not show clear positive discrimination between healthy and cancerous stiffness values with 5% random added noise. However, with 20×20 points per element and greater, this is no longer an issue and the accuracy continues to increase as the data resolution increases. The algorithm is also shown to produce consistent results within the anticipated range of carcinoma Young's Modulus values.

These results have to be put into perspective, as the algorithm was evaluated for only a limited range of stiffness values and no variation in the density of the tissue. These parameters are used in the simulation of the forward simulated motion displacements and directly relate to the natural frequency and therefore the dynamic response of the breast tissue. Therefore, for a certain combination of these system parameters, there could be little or no displacement response for a specific actuation frequency. This actuation frequency could coincide with one that is intended to be used for carcinoma identification, causing the inverse algorithm to be drowned out by

noise. Thus, further investigations are required into these natural frequency effects for a complete range of mass and stiffness properties when the inverse algorithm is extended to 3D cases.

The 1D Global Double Integral inverse algorithm was used to identify a carcinoma within a 2D simulated tissue sample by taking slices of either the x direction or the y direction displacement amplitudes (u or v respectively) and using it as input to the 1D inverse algorithm. The 1D model only accounts for the shear wave displacement and is very simplistic compared against the 2D model that accounts for the shear and longitudinal waves in two dimensions. However, initial investigations showed that it is possible to successfully identify a carcinoma using this technique.

The accuracy of the 2D non-homogeneous inverse algorithm was improved considerably by assuming the elastic material is incompressible. This simplified approach removes ill-conditioned terms and increases the numerical stability of the integral equations. At an actuation frequency of 100Hz, it became possible to successfully identify carcinoma with a data resolution of down to 2 points per segment and random added noise of 40% and greater. At 50Hz actuation, it is possible to successfully identify carcinoma with up to 20% noise and 10 data points per segment. This increased accuracy and robustness represents a significant improvement on the initial compressible model and opens up the possibility of adopting ultrasound, or similarly noisy measurement device, as a measurement technique instead of MRI.

Part 4

Conclusions

7 Conclusions & Future Work

7.1 1D Inverse Problem Solutions

A non-homogeneous 1D forward simulation algorithm that approximates the 1D Navier's equation is formulated based on finite difference approximations. The method allows step-wise constant spatial variation of stiffness to incorporate carcinoma as regions of significantly greater stiffness than the adjacent healthy tissue. The algorithm is successfully verified against corresponding analytical solutions. The displacement amplitude solution to the forward simulation provides motion data that can then be used as input with random added noise to test the inverse problem algorithms.

Three 1D inverse algorithms were developed based upon different integral methods. The Local Inverse algorithm, is a local integration method that calculates the value of the Shear Modulus at each segment using only displacement data from that particular segment. Thus, each stiffness value is calculated independently from the surrounding segments. The Global Single Integral inverse algorithm involves single integration over the whole displacement dataset to formulate a linear system of equations involving the Shear Modulus of all segments. Thus, the interactions between different segments and their effects on displacement are utilised to create a more effective algorithm. For the single integration method, the fundamental equation is only integrated once and thus requires differentiation of the displacement data. The error in calculating the derivative is reduced by fitting a quartic polynomial to the noisy motion dataset and differentiating this polynomial to create a derivative function. Finally, the Global Double Integral inverse algorithm, involves double integrating the differential equation of motion. Thus, it incorporates the global knowledge of the stiffness distribution, and does not require any noise sensitive, numerical differentiation.

From comparative evaluation of the three different inverse algorithms developed it is clear that the Global Double Integral inverse algorithm performs the best. It accurately identifies the Shear Moduli of both the carcinoma and healthy tissue and provides clear positive discrimination between stiffness values. The Local inverse algorithm has potential benefits due to its overall

simplicity, but is highly dependent on the parameters of the problem. It is effective at identifying regions where there is sufficient spatial information within each discretized segment. For example, if the magnitude of Shear Modulus is sufficiently low or the size of the discretized segment is sufficiently large. It would also be the best method for identifying the Shear Modulus of homogenous material. The Global Single Integral inverse algorithm does not provide significant positive discrimination between healthy and cancerous stiffness values due to the sensitivity to noise involved in numerically calculating derivatives. Thus, the Global Double Integral method most effectively filters the noise through the double integral process to provide the most accurate result utilising all motion data.

The Global Double Integral was evaluated further and was found to be robust to the effects of random, uniformly distributed noise. The main findings are summarized as follows:

- As the actuation frequency of the tissue simulation is decreased the accuracy of the algorithm also decreases due to the reduction of spatial information.

- An increase in the data resolution of the motion data input results in reduced variability of the calculated carcinoma stiffness resulting in increased positive discrimination.

- A reduction of carcinoma stiffness to the lower limits of expected Young's Modulus values results in increased performance of the algorithm due to the increased spatial information within the carcinoma segment.

Note that these results are for the assumed mass and stiffness system parameters used in forward simulation. These parameters define the natural frequency and therefore the response of breast tissue. Thus, a change in these parameters could alter the dynamic response so that the inverse algorithm is no longer accurate for certain actuation frequencies.

Successful identification of the carcinoma is not always guaranteed when its position is varied throughout the global domain. This loss of accuracy occurs because the carcinoma can be positioned in this 1D example such that it does not significantly alter the global motion dataset in comparison to a homogeneous model. These cases are therefore geometrically ill-conditioned, but are not likely to occur in a physiologically realistic 3D case. However, it should be noted

that a realistic 10% variation in actuation frequency can alter the global motion dataset sufficiently for successful identification of the carcinoma, even in this simplified case.

A method of localized mesh refinement was developed as an adaptation to the Global Double Integral inverse algorithm. This method was developed to identify carcinoma not aligned with the discretized grid and carcinoma smaller than the discretized grid. It proved to be relatively unsuccessful as the reduction of grid size can create false positives within the solution as the segment length becomes too small to accurately identify the healthy tissue stiffness due to the lack of spatial information. However, a method was proposed to identify carcinoma not aligned with the discretized grid that involves multiple iterations of the inverse algorithm where the position of the grid is moved incrementally for each iteration, such that the grid will approximately align with the tumour at some iteration.

7.2 2D Inverse Problem Solutions

A non-homogeneous 2D forward simulation algorithm that solves the 2D plane strain Navier's Equation is formulated using finite difference approximations. The algorithm allows step-wise constant variation of stiffness in order to incorporate a carcinoma, which is modelled as a region of stiffness significantly greater than the adjacent healthy tissue. The finite difference approximation to the homogeneous model is successfully verified against the analytical solution that exists in the infinite domain. However, there exist no analytical solutions for models using more realistic boundary conditions and/or have non-homogeneous material properties. Therefore, these models were verified by checking convergence of the solutions for successive grid refinements. The homogeneous models showed clear convergence and grid independence, but this convergence is not as clear for the non-homogeneous model, particularly at high actuation frequencies. An increase in the memory capacity of the computers used would be required to confidently prove convergence of the forward simulation algorithm compared with the 4096Mb of RAM used for the current simulations. The displacement amplitudes generated from the forward simulation provide the 'measured' motion data that is used as input to test the inverse problem algorithms.

Two homogeneous inverse algorithms were developed to find the best framework for the development of a non-homogeneous algorithm. The Initial inverse algorithm adopts an integral

method that uses the local origin as the reference with which to integrate from. This method contains derivatives terms that are particularly susceptible to noise and are the primary source of error within the solution. There are also single integral terms that are not as effective at filtering the noise as the double integral terms. The second method is the Centred Base Point algorithm that adopts a unique set of integration limits, resulting in a formulation that does not require the calculation of derivatives and contains only double integral terms. The algorithm is thus very robust to noise and is used as the basis for a non-homogeneous algorithm.

A 2D non-homogeneous inverse algorithm was then formulated that is capable of identifying a square 1cm square tumour within the square global domain with 10cm sides. The algorithm consists of a 2×2 stencil that can be mapped across the global domain to produce an over-determined system of linear equations that inter-relates all of the discretized stiffness elements throughout the domain. The mapping of the stencil is performed by moving the local base point across the global domain.

This algorithm was evaluated by quantifying its accuracy and robustness with the variation of important system parameters. For an actuation frequency of 100Hz the algorithm consistently identifies the carcinoma for a range of positions and boundary conditions with up to 5% random, uniformly distributed noise. This limit is increased to 10%, with similar results, if the simple shear stress continuity constraint model is adopted. For an actuation frequency of 50Hz the algorithm requires the use of the constraint model to successfully identify the carcinoma with up to 5% random noise added to the motion data input because the lower actuation frequency gives less spatial information per discretized element for this specific set of parameters.

An increase in the data resolution of the motion data increases the accuracy of the reconstructed carcinoma and reduces the 90% confidence interval range resulting in greater positive discrimination of stiffness values. For 10×10 points per element and with an actuation frequency of 100Hz the algorithm does not show clear positive discrimination between healthy and cancerous stiffness values with 5% random added noise. However, with 20×20 points per element and greater, there is clear discrimination and the accuracy continues to increase as the data resolution increases. The algorithm is also shown to produce consistent results within the anticipated range of cancerous Young's Modulus values.

The concept of using a simplified model to identify a tumour was applied by using the 1D Global Double Integral inverse algorithm to identify a carcinoma within a 2D simulated tissue sample. The 1D inverse algorithm was applied to the 2D dataset by using slices of either the x direction or the y direction displacement amplitudes (u or v respectively) as motion data input. The 1D model only accounts for the shear wave displacement and is very simplistic compared against the 2D model that accounts for the shear and longitudinal waves in two dimensions. However, initial investigations showed that it is possible to successfully identify a carcinoma using this technique.

This concept was extended by assuming the elastic material is incompressible and reformulating the 2D non-homogeneous inverse algorithm. This simplified approach removes ill-conditioned terms and increased the numerical stability of the inverse solutions. For an actuation frequency of 100Hz, a carcinoma was successfully identified with a data resolution down to 2 points per segment and noise of 40% and greater. At 50Hz actuation, a carcinoma was identified with up to 20% noise and 10 data points per segment. This increased accuracy and robustness represents a significant improvement on the initial compressible model formulation and opens up the possibility of adopting cheaper and more portable measurements systems like ultrasound, instead of MRI.

7.3 Future Work

The next step is to develop a 3D non-homogeneous inverse algorithm that takes the displacement amplitude motion data of harmonically actuated breast from an MRI machine as input and provides a three dimensional map of the tissue stiffness. The 3D formulation would involve extending the integral based 2×2 stencil from the 2D plain strain non-homogeneous inverse algorithm to a $2\times2\times2$ stencil, based on similar integration techniques applied to the 3D Navier's Equations. The incompressible model could easily be incorporated by setting the divergence of the 3D Navier's equations to zero.

Verification and testing of this algorithm should be first applied on simulated motion data before applying to existing MRI datasets of actuated phantoms with high stiffness. The inverse algorithm could also be extended to incorporate the more complex geometry of the breast by incorporating, for example, triangular elements near the boundary.

A carcinoma was successfully identified in 2D using simplistic models like the 1D inverse algorithm and the incompressible model formulation. However, when identifying carcinoma at an actuation frequency of 50Hz using the incompressible model there are significant erroneous effects in the stiffness distribution along the boundary of the global domain. One possible way around this is to incorporate extra step-wise constant, spatial varying parameters within these regions of high modelling error, to absorb the non-linear effects [Singh-Levett, 2006].

Initial investigations adopted the divergence as a spatially varying parameter and identified this parameter along the boundary of the global domain, as well as the internal stiffness distribution. Preliminary results show this approach could remove the modelling error in the calculated stiffness distribution and thus increase the accuracy of the inverse algorithm. Therefore a similar approach could be taken in 3D where a simplified 'minimal model' that captures the fundamental dynamics of the actuated tissue but reduces the overall complexity and numerical instability. The degree of complexity should only be the amount that gives answers to the key questions regarding the stiffness distribution of the breast tissue, which are the location and approximate stiffness of the carcinoma. The main point is that clear positive discrimination is maintained between healthy and cancerous stiffness values.

7.3.1 Measure of Homogeneity

The 'Measure of Homogeneity' is an alternative approach to identifying a tumour. It further displays the flexibility and power of the current methodology for future applications. The method looks to develop a measure of local homogeneity throughout the domain so that cancer will show up as a large error between the measured displacement and the simulated displacements from an assumed locally homogeneous model. In other words, instead of directly calculating the stiffness of every element in the global domain to detect cancer, a low resolution stiffness distribution is assumed throughout the domain. For example, a piecewise constant stiffness model could be applied over $3\text{cm} \times 3\text{cm}$ elements. This simplified locally homogeneous model can be rapidly and accurately fitted using the non-homogeneous integral method of Section 4.2 and would readily take into account the natural variations in healthy tissue. This stiffness is then used to generate displacements from one forward simulation of the

homogeneous model, and the generated displacements are compared with the measured displacements to detect regions that are highly non-homogeneous suggesting cancer.

However, note that a direct comparison of the homogeneous model generated displacements with the measured data is not sufficient, as a tumour has both local effects and global effects on the displacement. Thus, the algorithm locally translates regions to remove the major global effects and to enhance the local effect of the tumour on its surrounding area. Note also that local regions that are topologically similar would be effectively overlaid. Details and analysis of this approach are left for future work.

An example of this concept is given where the locally homogeneous baseline model is taken to be homogeneous over the whole domain. That is, a single $10\text{cm} \times 10\text{cm}$ element is chosen for simplicity rather than the $3\text{cm} \times 3\text{cm}$ elements described above. This amounts to one Young's Modulus value being fitted throughout the domain, which is achieved using the Centred Base Point formulation of Section 4.1.2. Steps 1-4 in Figure 7.1 show the overall process of this method. Figure 7.2 shows the results of this homogeneity method using the Edge Effect boundary condition forward simulation of Figure 3.1(b), an actuation frequency of 50 Hz and 10% uniformly distributed noise added to the displacements. A 9 point moving average is also applied on the noisy data, before applying the process in Figure 7.1. The tumour is clearly identified showing the future potential of this approach.

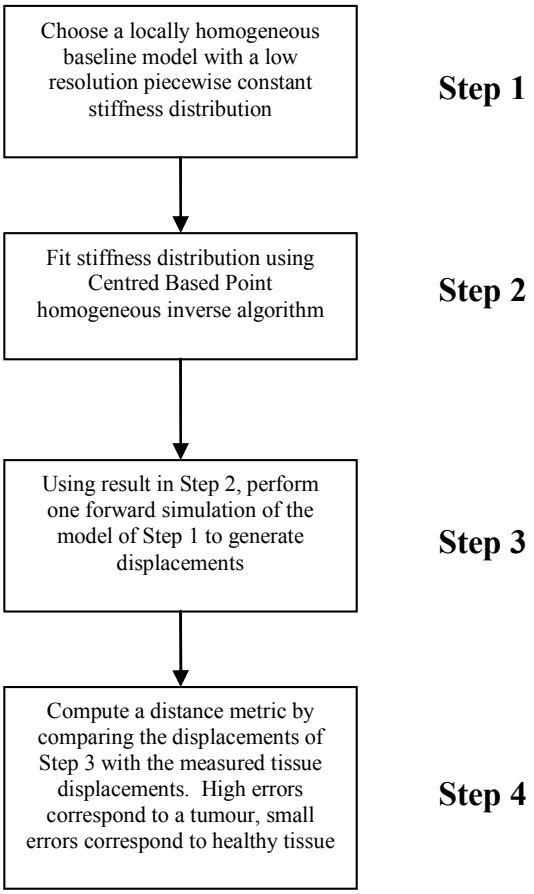

Figure 7.1 – Summary of the Measure of Homogeneity approach to identify tumours

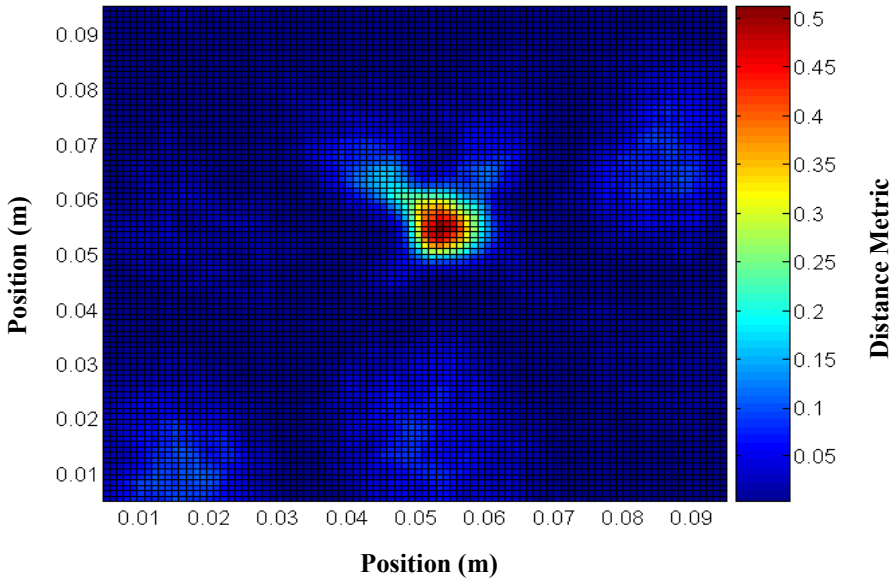

Figure 7.2 – Identifying a 1cm × 1cm tumour using the Measure of Homogeneity approach

8 References

American Cancer Society. (2006): *'Cancer Facts & Figures 2006'*, www.cancer.org

B. Bone, P. Aspelin, L. Bronge, and B. Veress. (1998): *'Contrast-enhanced MR Imaging as a Prognostic Indicator of Breast Cancer,'* Acta Radiol, 39:279-284.

P. C. Chou, N.J. Pagano. (1967): *'Elasticity – Tensor, Dyadic, and Engineering Approaches,'* D. Van Nostrand Co., Inc.

J. Chamberlain. (2002): *'Breastscreen Aotearoa an Independent Review.'*

V.C. Cokkinides, A. Samuels, E.M. Ward, and M.J. Thun. (2004): '*Cancer Prevention & Early Detection Facts & Figures,*' American Cancer Society.

E.M. Haack, R.W. Brown, M.R. Thompson, R. Venkatesan. (1999): *'Magnetic Resonance Imaging: Physical Properties & Sequence Design,'* Wiley-Liss.

C.E. Hann, J.G Chase, J. Lin, T. Lotz, C.V. Doran, and G.M. Shaw. (2005): *'Integral-Based Parameter Identification For Long-Term Dynamic Verification Of A Glucose-Insulin System Model,'* Computer Methods and Programs in Biomedicine, Vol 77(3), pp. 259-270, ISSN: 0169-2607.

C.E. Hann, J.G Chase, and G.M. Shaw (2006): *'Integral-Based Identification of Patient Specific Parameters for a Minimal Cardiac Model,'* Computer Methods and Programs in Biomedicine, 81(2), pp. 181-192, ISSN: 0169-2607.

A. Hii (2005): 'Cluster Tracking Algorithms for a Digital Image-based Elasto-Tomography System.' Masters Thesis, Mechanical Engineering, University of Canterbury.

T.M. Krouskup, T. Wheeler, F. Kallel, B.S. Garra and T. Hall. (1998): '*Elastic Modulli of Breast and Prostate Tissues Under Compression*,' Ultrasonic Imaging, Vol. 20, pp. 260-274.

National Breast Cancer Coalition, Washington D.C. www.stopbreastcancer.org

National Cancer Institute. www.cancer.gov

New Zealand Breast Cancer Foundation (NZBCF). '*Facts on Breast Cancer,'* www.nzbcf.org.nz

New Zealand Health Information Service. (2001): '*Cancer: New Registrations and Deaths 2001,'* www.nzhis.govt.nz

A. Peters, A. Milsant, J. Rouzé, L. Ray, J.G. Chase and E.E.W. Van Houten. (2004): *'Digital Image-based Elasto-Tomography: Proof of concept studies for surface-based mechanical property reconstruction,'* JSME Int. J., Ser. C, 47, pp. 1117-23.

A. Peters, S. Wortmann, R Elliot, M. Staiger, J.G. Chase, and E.E.W. Van Houten (2005): *'Digital Image-based Elasto-Tomography: First Experiments in Surface Based Mechanical Property Estimation of Gelatine Phantoms,'* JSME Int. J., Ser. C, 48, pp. 562-69.

M.B. Robertson (2005): 'Commercialisation and IP Strategies for Digital Image-based Elasto-Tomography (DIET),' Masters in Engineering Management Thesis, University of Canterbury.

A. Samani, J. Bishop, C, Luginbuhl, D.B. Plewes (2003): *'Measuring the elastic modulus of ex vivo small tissue samples,' Phys. Med. Biol.*, 48, pp. 2183-98.

I. Singh-Levett (2006): *'Real-Time Integral Based Structural Health Monitoring,'* Masters Thesis, Mechanical Engineering, University of Canterbury.

L. Tabar, M. Yen, B. Vitak, H. T. Chen, R. A. Smith, and S.W. Duffy.(2003): *'Mammography service screening and mortality in breast cancer patients: 20-year follow-up before and after introduction of screening,'* The Lancet, 361:1405-1410.

E.E.W. Van Houten, KD. Paulsen, M. I Miga, F.E. Kennedy, and J.B. Weaver. (1999): '*An Overlapping Subzone Technique for MR-Based Elastic Property Reconstruction,'* Magnetic Resonance in Medicine 42:779-786, Wiley-Liss, Inc.

E.E.W. Van Houten, KD. Paulsen, M. I Miga, F.E. Kennedy, and J.B. Weaver. (2001): '*Three-Dimensional Subzone-Based Reconstruction Algorithm for MR Elastography'*, Magnetic Resonance in Medicine 45:827-837, Wiley-Liss, Inc.

Appendix A: MATLAB Code

A1: 1D Non-Homogeneous Forward Simulation Algorithm

```
%=========================================================================

% 1D NON-HOMOGENEOUS FORWARD SIMULATION ALGORITHM
% By Samuel J. Houghton
% 2005

% This algorithm uses Finite Difference Approximations to solve for the
% Harmonic Displacement Amplitudes from the 1D Navier's Equation.  It
% incorporates a carcinoma by allowing a step-wise constant stiffness
% distribution and enforces stress continuity at the boundary of segments
% with differing stiffness.

%=========================================================================

clear
clc

tt0 = cputime;

% Defining Model Parameters ================================================

rho = 1020;             % Density (kg/m3)
omega = 100*2*pi;       % Actuation Frequency (rad/s)
mu = 0.49;              % Poisson's Ratio

% Longitudinal Wave Formulation
% A = (rho*omega^2)/(1/(1+mu) + mu/((1+mu)*(1-2*mu)));

% Shear Wave Formulation
A = (rho*omega^2);

len = 2000;             % Total number of Discretized Data Points in Domain
samp = 0.1;             % Length of Domain (m)
h = samp/len;           % Step size

% Define Carcinoma Position
g1 = len*0.2+1;
g2 = len*0.3+1;
g3 = len+1;

E1 = 30000;             % Define Young's Modulus of Healthy Tissue (Pa)

G1 = E1/(2+2*mu);       % Shear Modulus of Healthy Tissue to Left of Carcinoma
G2 = 10*G1;             % Define Shear Modulus of Carcinoma
G3 = G1;                % Shear Modulus of Healthy Tissue to Right of Carcinoma

% Initialize Matrices & Vectors for System of Equations
A1 = A/G1;
A2 = A/G2;
A3 = A/G3;

G = ones(len+1,1);
G(1:g1) = G1;       G(g1+1:g2) = G2;        G(g2+1:g3) = G3;
AA1(1:g1) = A1;     AA1(g1+1:g2) = A2;      AA1(g2+1:g3) = A3;

n = len+1;
K = zeros(n,n);
RHS = zeros(n,1);

% Assembling System of Equations & Solve ===================================
```

```
% 1st Segment Stiffness Terms ------------------------------------------

%Forward Differences
K(1,1) = 2/(h^2) + A1;
K(1,2) = -5/(h^2);
K(1,3) = 4/(h^2);
K(1,4) = -1/(h^2);

%Central Differences
for i = 2:g1-1
   K(i,i-1) = 1/(h^2);
   K(i,i)   = -2/(h^2) + A1;
   K(i,i+1) = 1/(h^2);
end

% 2nd Segment Stiffness Terms ------------------------------------------

% Enforce Stress Continuity
K(g1,g1-2) = G1;
K(g1,g1-1) = -4*G1;
K(g1,g1)   = 3*(G1+G2);
K(g1,g1+1) = -4*G2;
K(g1,g1+2) = G2;

% Central Differences
for i = g1+1:g2-1
   K(i,i-1) = 1/(h^2);
   K(i,i)   = -2/(h^2) + A2;
   K(i,i+1) = 1/(h^2);
end

% 3rd Segment Stiffness Terms ------------------------------------------

% Enforce Stress Continuity
K(g2,g2-2) = G2;
K(g2,g2-1) = -4*G2;
K(g2,g2)   = 3*(G2+G3);
K(g2,g2+1) = -4*G3;
K(g2,g2+2) = G3;

% Backwards Differences
K(g3,g3) = 2/(h^2) + A3;
K(g3,g3-1) = -5/(h^2);
K(g3,g3-2) = 4/(h^2);
K(g3,g3-3) = -1/(h^2);

% Central Differences
for i = g2+1:g3-1
   K(i,i-1) = 1/(h^2);
   K(i,i)   = -2/(h^2) + A3;
   K(i,i+1) = 1/(h^2);
end

% Apply Boundary Conditions ---------------------------------------------

% Boundary Conditions: U(0) = 0, U(n) = 0.001 (displace end of bar by 1mm)
Kbc = K;
Kbc(1,:) = zeros(1,n);
Kbc(1,1) = 1;
Kbc(n,:) = zeros(1,n);
Kbc(n,n) = 1;

RHS(1) = 0;
RHS(g3) = 0.001;

% Solve Forward Simulation ----------------------------------------------

format long
```

```
v = Kbc\RHS;
v = v*1000;

% Post-Processing ========================================================

% Plot Results ----------------------------------------------------------

figure(1)
x = [0:1:(n-1)]*h;
plot(x,v,'color','b')
xlabel('Node')
ylabel('Displacement mm')
grid on

xexact = x;
vexact = v;

% Save Results ----------------------------------------------------------

save fwdsol A G xexact vexact samp

tt1 = cputime - tt0

%-------------------------------------------------------------------------
```

A2: 1D Inverse Algorithm – Local Double Integral Method

```
%=========================================================================

% 1D INVERSE ALGORITHM - LOCAL DOUBLE INTEGRAL METHOD
% By Samuel J. Houghton
% 2005

% This algortihm uses a double integral formualtion of the 1D Navier's
% Equation to provide the basis of an inverse algortihm that solves for the
% stiffness distribution.  The integration is local - that is each segment
% is integrated independently and only incorporates motion data from within
% that segment.

%=========================================================================

clc
clear all

tt0 = cputime;

% Introduce and Define Parameters ========================================

load fwdsol     % Load Forward Simulation Data
Gorig = G;
clear G

ip = 20;        % No. of integration points
seg = 10;       % No. of discretized segments

% Vectors U and x must be of equal lengths => interpolate
xend = samp;
h = samp/seg/ip;
x = [0:h:xend];
v = interp1(xexact,vexact,x);
```

```
% NOISE GENERATOR ========================================================

% Calulating Geometric Mean
sortv = sort(abs(v));
percentilenoise = 50;   %Percentile of Absolute Data (Median - 50%)
propv = sortv(ceil(length(v)*percentilenoise/100));

% Defining Percentage Noise
perror = 10;
f1 = length(v);
rand1 = rand(f1,1) <= 0.5;

% Geometric Mean (Median) Absolute Noise
vN1 = (-1).^rand1*(perror/100)*propv;
vN2 = rand(f1,1);
vN3 = vN1.*vN2;
v  = v + vN3';

% Formulate System of Equations & Solve for Stiffness ====================

for j = 1:10        % Loop for each stiffness segment

    % Define Local Coordinates
    xx1 = (j-1)*0.1*samp;
    xx2 = j*0.1*samp;

    % Define Local Motion Dataset
    xj = x((j-1)*ip+1:j*ip+1);
    vj = v((j-1)*ip+1:j*ip+1);

    % Formulate Double Integral
    dint = h^2*cumtrapz(cumtrapz(vj));

    % Generate matrix of equations in the form A*x = B,
    % where x = [Ebar;a1;b1]
    A1 = zeros(length(dint),3);
    B1 = zeros(length(dint),1);
    A1(:,1) = -dint';
    A1(:,2) = xj';
    A1(:,3) = ones(length(dint),1);

    B1 = vj';

    % Solve using Linear Least Squares
    X1 = lsqlin(A1,B1);

    Gbar(j) = X1(1);
    G(j) = A/Gbar(j);
    a1(j) = X1(2);
    b1(j) = X1(3);

end

% Post-Processing =========================================================

Glong(1) = G(1);
for i = 1:10
    for j = 2:ip+1
        Glong(j+(i-1)*ip) = G(i);
    end
end

figure (1)
plot(x,Glong,xexact,Gorig)
format short
G'./1000
tt1 = cputime-tt0
%-------------------------------------------------------------------------
```

A3: 1D Inverse Algorithm – Global Single Integral Method

```
%=========================================================================

% 1D INVERSE ALGORITHM - GLOBAL SINGLE INTEGRAL METHOD
% By Samuel J. Houghton
% 2005

% This algortihm uses a single integral formualtion of the 1D Navier's
% Equation to provide the basis of an inverse algortihm that solves for the
% stiffness distribution.  The integration is global - that is each the
% formulation incorporates global motion data.  However, the defining
% equation is integrated only once, requiring the calculation of
% derivatives using quardic polynomial fitting.

%=========================================================================

clc
clear all

tt0 = cputime;

% Introduce and Define Parameters ========================================

load fwdsol     % Load Forward Simulation Data
Gorig = G;
x = xexact;
v = vexact;

ip = 20;            % No. of integration points
seg = 10;           % No. of discretized segments
h = samp/seg/ip;    % Step Size

xend = samp;
xnew = [0:h:xend];
vnew = interp1(x,v,xnew);

poly1 = 4;          % Define Polynomial Order for Fitting - Note the
                    % derivative formulations require manual alteration

% NOISE GENERATOR ========================================================

% Calulating Geometric Mean
sortv = sort(abs(vnew));
percentilenoise = 50;   %Percentile of Absolute Data (Median - 50%)
propv = sortv(ceil(length(vnew)*percentilenoise/100));

% Defining Percentage Noise
perror = 10;
f1 = length(vnew);
rand1 = rand(f1,1) <= 0.5;

% Geometric Mean (Median) Absolute Noise
vN1 = (-1).^rand1*(perror/100)*propv;
vN2 = rand(f1,1);
vN3 = vN1.*vN2;
vnew  = vnew + vN3';


% Formulate System of Equations & Solve for Stiffness ====================

% POLYNOMIAL FITTING -----------------------------------------------------

ynew = zeros(1,ip*10);
yderiv = zeros(1,ip*10);
```

```
 % 1st Polynomial Fitting ------------------------------------------------

    coeff1 = polyfit(xnew(1:ip+1),vnew(1:ip+1),poly1);

    ynew(1,1:ip+1) = polyval(coeff1,xnew(1:ip+1));

%     % Cubic Derivative
%     yderiv(1,1:ip+1) = 3*coeff1(1)*xnew(1,1:ip+1).^2...
%         + 2*coeff1(2)*xnew(1,1:ip+1) + coeff1(3);

    % Quardic Derivative
    yderiv(1,1:ip+1) = 4*coeff1(1)*xnew(1,1:ip+1)...
        .^3 + 3*coeff1(2)*xnew(1,1:ip+1).^2 + 2*coeff1(3)*xnew(1,1:ip+1)...
        + coeff1(4);

% Last Polynomial Fitting -------------------------------------------------

    coeff10 = polyfit(xnew(9*ip+1:10*ip+1),vnew(9*ip+1:10*ip+1),poly1);

    ynew(1,9*ip+2:10*ip) = polyval(coeff10,xnew(9*ip+2:10*ip));

%     % Cubic Derivative
%     yderiv(1,9*ip+2:10*ip) = 3*coeff10(1)*xnew(1,9*ip+2:10*ip).^2...
%         + 2*coeff10(2)*xnew(1,9*ip+2:10*ip) + coeff10(3);

    % Quardic Derivative
    yderiv(1,9*ip+2:10*ip) = 4*coeff10(1)*xnew(1,9*ip+2:10*ip)...
        .^3 + 3*coeff10(2)*xnew(1,9*ip+2:10*ip).^2...
        + 2*coeff10(3)*xnew(1,9*ip+2:10*ip) + coeff10(4);

% Rest of Polynomial Fitting ----------------------------------------------

for i = 2:9
    coeffi = polyfit(xnew((i-1)*ip+1:i*ip+1),vnew((i-1)*ip+1:i*ip+1),poly1);

    ynew(1,(i-1)*ip + 2:i*ip+1) = polyval(coeffi,xnew((i-1)*ip + 2:i*ip+1));

%     % Cubic Derivative
%     yderiv(1,(i-1)*ip + 2:i*ip+1) = 3*coeffi(1)*xnew(1,(i-1)*ip...
%         + 2:i*ip+1).^2 + 2*coeffi(2)*xnew(1,(i-1)*ip + 2:i*ip+1) + coeffi(3);

    % Quardic Derivative
    yderiv(1,(i-1)*ip + 2:i*ip+1) = 4*coeffi(1)*xnew(1,(i-1)*ip + 2:i*ip+1)...
        .^3 + 3*coeffi(2)*xnew(1,(i-1)*ip + 2:i*ip+1).^2 + 2*coeffi(3)...
        *xnew(1,(i-1)*ip + 2:i*ip+1) + coeffi(4);
end

% Numerical Derivative - Alternate Derivative Formulation
deltat = xnew(2)-xnew(1);
lxnew = length(xnew);

vderiv(1,1) = 1/(2*deltat)*(-3*vnew(1)+4*vnew(2)-vnew(3));
vderiv(1,lxnew) = 1/(2*deltat)*(3*vnew(lxnew)-4*vnew(lxnew-1)+vnew(lxnew-2));
for i = 2:(length(xnew)-1)
    vderiv(1,i) = 1/(2*deltat)*(vnew(i+1)-vnew(i-1));
end

% Actual Numerical Derivative - Derivative of Non-Noisy Data
deltat2 = x(2)-x(1);
lx = length(x);

vderivactual(1,1) = 1/(2*deltat2)*(-3*v(1)+4*v(2)-v(3));
vderivactual(1,lx) = 1/(2*deltat2)*(3*v(lx)-4*v(lx-1)+v(lx-2));
for i = 2:(length(x)-1)
    vderivactual(1,i) = 1/(2*deltat2)*(v(i+1)-v(i-1));
end

% Derivative Terms Comparison ---------------------------------------------
```

```
figure(1)
plot(xnew(1:lxnew-1),ynew,xnew,vnew,'r')
figure(2)
plot(xnew(1:lxnew-1),yderiv,xnew,vderiv,'r')
figure(3)
plot(xnew,vderiv,'g',x,vderivactual,xnew(1:lxnew-1),yderiv,'r')
xlabel('Position (m)')
ylabel('Derivative of Displacement Amplitude')
legend('Numerical','Actual','Fitted','Location','NorthWest')

x0 = xnew(1:lxnew-1);
vderivactual1 = interp1(x,vderivactual,x0);
vderivactual2 = interp1(x,vderivactual,xnew);

comp1 = abs(yderiv-vderivactual1)/mean(abs(vderivactual1))*100;
comp2 = abs(vderiv-vderivactual2)/mean(abs(vderivactual2))*100;

lim1 = 0.6;
lim2 = lim1 + 0.1;
comp1a = comp1(ip*10*lim1+1:ip*10*lim2);
mean(comp1)
mean(comp2)
max(comp1)
max(comp2)

% Assemble System of Equations ==========================================

AA = zeros(ip*10,10);

FF = h*cumtrapz(vnew(1:end-1));
BB = -A*FF;

for j = 1:10

    xnewj = xnew((j-1)*ip + 1:j*ip);
    vnewj = vnew((j-1)*ip + 1:j*ip);
    yderivj = yderiv((j-1)*ip + 1:j*ip);

    AA((j-1)*ip + 1:j*ip,j) = yderivj';

end

yd0 = yderiv(1);
AA(:,1) = -yd0;

X1 = lsqlin(AA,BB);

% Post-Processing ========================================================

Glong(1) = X1(1);
for i = 1:10
    for j = 2:ip+1
        Glong(j+(i-1)*ip) = X1(i);
    end
end

figure (4)
plot(xnew,Glong,xexact,Gorig)
grid on
%Ylim([0 30000])
format short
G = X1./1000

tt1 = cputime-tt0

%-------------------------------------------------------------------------
```

A4: 1D Inverse Algorithm – Global Double Integral Method with Mesh Refinement

```
%=========================================================================

% 1D INVERSE ALGORITHM - GLOBAL DOUBLE INTEGRAL METHOD
% By Samuel J. Houghton
% 2005

% This algortihm uses a double integral formualtion of the 1D Navier's
% Equation to provide the basis of an inverse algortihm that solves for the
% stiffness distribution.  The integration is global - that is each the
% formulation incorporates global motion data.

% It also incorporates a localized mesh refinement algorithm that attempts
% to identify carcinoma smaller than or not aligned with the original
% discretized grid.  It also adopts the global double integral method.

%=========================================================================

clear all
clc

% Introduce Variables ===================================================

load fwdsol          % Load Forward Simulation Data
Gorig = G;
clear G
B = A;

ip = 20;             % No. of Integration Points
ipa = ip/2;          % No. of Integration Points - Refined Elements
seg = 10;            % No. of Segments
sega = 2*seg;        % No. of Segments (if all segments refined)
hh = samp/ip/seg;    % Step Size
xend = samp;
x = [0:hh:xend];
v = interp1(xexact,vexact,x);    % Generate Motion Dataset
cutoff = 1.2;        % The proportion of which that exceeds the stiffness
                     % threshold before mesh is refined locally in the
                     % particular region

% NOISE GENERATOR =======================================================

% Calulating Geometric Mean
sortv = sort(abs(v));
percentilenoise = 50;   %Percentile of Absolute Data (Median - 50%)
propv = sortv(ceil(length(v)*percentilenoise/100));

% Defining Percentage Noise
perror = 10;
f1 = length(v);
rand1 = rand(f1,1) <= 0.5;

% Geometric Mean (Median) Absolute Noise
vN1 = (-1).^rand1*(perror/100)*propv;
vN2 = rand(f1,1);
vN3 = vN1.*vN2;
v  = v + vN3';

% GLOBAL DOUBLE INTEGRAL METHOD =========================================

% Define & Solve System of Equations ====================================
```

```
% Double Integral Formulation ------------------------------------------

dint1 = cumtrapz(cumtrapz(v))*hh^2;
dint = dint1(1:ip*seg)';

% Form Matrices & Solve -------------------------------------------------

AA = zeros(ip*seg,2*seg+1);
AA(:,1) = x(1:ip*seg)';

for i = 1:seg
    Vx = v(((i-1)*ip+1):i*ip);
    AA(((i-1)*ip+1):i*ip,(i+1)) = Vx;

    AA(((i-1)*ip+1):i*ip,(i+seg+1)) = ones(ip,1);
end

RHS = -B*dint;
xx = lsqlin(AA,RHS);
G = xx(2:seg+1);

% Initial Plotting of Global Double Integral Method Results-----------------

Glong(1) = G(1);
for i = 1:seg
    for j = 2:ip+1
        Glong(j+(i-1)*ip) = G(i);
    end
end
figure(1)
plot(x,Glong,xexact,Gorig)

figure(2)
plot(xexact,vexact)

% ADAPTIVE MESH ALGORITHM ================================================

% Specify Stiffness Threshold

% for i = 1:seg
%     thresh1(i) = G(i)/mean(G);
% end

thresh1 = G./(Gorig(1));

if max(thresh1) >= cutoff

    % Identify Segments That Exceed Stiffness Threshold ------------------

    ind1 = [];
    for i = 1:seg
        if thresh1(i) >= cutoff
            ind1 = [ind1 i];
        end
    end



    % Introduce Variables ------------------------------------------------

    hha = samp/ipa/sega;
    xa = [0:hha:xend];
    va = interp1(x,v,xa);

    % Generate Adjusted x & v vectors in order to refine mesh ------------

    count = 1;
    xb = [];
```

```
hhnew = ones(1,ip*seg+1)*hh;
xb = [0:hh:xend];
hhi = ones(1,hh/hha*ip)*hha;
seg1 = seg;
count = 0;

for i = 1:length(ind1)
    if ind1(i) == 1
        xi = [0:hha:ind1(i)*xend/seg-hha];
        xb = [xi,xb(ip+1:end)];
        hhnew = [hhi,hhnew(ip+1:end)];
        count = count + 1;
    elseif ind1(i) == seg
        xi = [(ind1(i)-1)*xend/seg:hha:ind1(i)*xend/seg];
        xb = [xb(1:(ind1(i)-1)*ip+count*(ipa-ip)),xi];
        hhnew = [hhnew(1:(ind1(i)-1)*ip+count*(ipa-ip)),hhi];
    else
        xi = [(ind1(i)-1)*xend/seg:hha:ind1(i)*xend/seg-hha];
        xb = [xb(1:(ind1(i)-1)*ip+count*(ipa-ip)),xi,xb((ind1(i))*ip...
            +count*(ipa-ip)+1:end)];
        hhnew = [hhnew(1:(ind1(i)-1)*ip+count*(ipa-ip)),hhi,...
            hhnew((ind1(i))*ip+count*(ipa-ip)+1:end)];
        count = count + 1;
    end
    seg1 = seg1 + 1;
end

vb = interp1(x,v,xb);

% Double Integral Formulation ----------------------------------------

sint1b = cumtrapz(xb,vb);
dint1b = cumtrapz(xb,sint1b);
dintb = dint1b(1:end-1)';

% Form & Solve System of Equations -----------------------------------

Cseg = (sega/seg)*(seg1-seg);
Dseg = sega/seg;
Eseg = (sega*ipa)/seg-ip;

AAb = zeros((seg1-Cseg)*ip+Cseg*ipa,2*seg1+1);
AAb(:,1) = xb(1:end-1)';

counta = 1;
countb = 1;

for i = 1:seg

    if i == ind1(counta)

        vai1 = va(Dseg*(i-1)*ipa+1:(Dseg*i-1)*ipa);
        vai2 = va((Dseg*i-1)*ipa+1:Dseg*i*ipa);

        z0 = zeros(size(vai1));
        z1 = ones(size(vai1));

        vai = [vai1' z0';z0' vai2'];
        zi = [z1' z0';z0' z1'];

        AAb((i-1)*ip+(countb-1)*Eseg+1:i*ip+countb*Eseg,i...
            +countb:i+countb+1) = vai;
        AAb((i-1)*ip+(countb-1)*Eseg+1:i*ip+countb*Eseg,i+countb...
            +seg1:i+countb+seg1+1) = zi;

        clear zi

        if counta == length(ind1)
```

```
            countb = countb+(Dseg-1);
            else
            counta = counta+(Dseg-1);
            countb = countb+(Dseg-1);
            end

        else
            vi = v((i-1)*ip+1:i*ip);
            zi = ones(size(vi));

            AAb((i-1)*ip+(countb-1)*Eseg+1:i*ip+(countb-1)*Eseg,i+countb)...
                = vi;
            AAb((i-1)*ip+(countb-1)*Eseg+1:i*ip+(countb-1)*Eseg,i+countb...
                +seg1) = zi;
        end
    end

    RHSb = -B*dintb;

    % Solve Linear Least Squares Problem --------------------------------

    xxb = lsqlin(AAb,RHSb);

    % Post-Processing ========================================================

    % Plotting of Mesh Refinement Results -------------------------------

    Gb = xxb(2:seg1+1);
    Ga = ones(1,sega)*xxb(2);

    ind1a = [];
    rt = sega/seg;
    for i = 1:length(ind1)
        ind1a = [ind1a rt*ind1(i)+[-rt+1:1:0]];
    end

    for i = 1:seg1
        for j = 1:rt
            Ga(j+(i-1)*rt) = Gb(i);
        end
    end
    Gc = Ga;

    count = 0;

    for i = 1:length(ind1)
        lGa = length(Ga);
        Ga = [Ga(1:(rt*ind1(i))-1),Ga((rt*ind1(i)+2):lGa)];
    end

    Glonga(1) = Ga(1);
    for i = 1:sega
        for j = 2:ipa+1
            Glonga(j+(i-1)*ipa) = Ga(i);
        end
    end
    Glonga = Glonga;
    xa = [0:hha:xend];

    figure(4)
    plot(xa,Glonga,xexact,Gorig,x,Glong)
    xlabel('Distance Along Breast (m)')
    ylabel('Modulus of Elasticity (Pa)')

end
format short
Ga'./1000
%-------------------------------------------------------------------------
```

A5: 2D Homogeneous Forward Simulation Algorithm – Infinite Domain Boundary Condition Model

```
%=========================================================================

% 2D HOMOGENEOUS FORWARD SIMULATION ALGORITHM - INFINITE DOMAIN B.C.
% By Samuel J. Houghton
% 2006

% This algorithm uses Finite Difference Approximations to solve for the
% Harmonic Displacement Amplitudes from the 2D Plain Strain Navier's
% Equations.  The medium has homogeneous material properties.

% It adopts Type I boundary conditions along the exterior of the square
% global domain that are a function of an analytical solution that exists
% across the infinite domain.  Therefore the solution of the forward
% simulation algorithm can be verified against analytical results.

%=========================================================================

clear
clc

tt0 = cputime;

% Definition of Variables ================================================

rho = 1020;             % Density (kg/m3)
omega = 2*pi*100;       % Frequency of Oscillation (rad/s)
mu = 0.49;              % Poisson's Ratio

sampi = 0.1;            % Length in the x direction
sampj = 0.1;            % Length in the y direction

gr = 400;               % Grid Refinement
grefinex = gr;
grefiney = gr;
h = sampi/grefinex;     % Step Size
k = h;

ni = gr + 1;
nj = gr + 1;
totaln = ni*nj;         % Total No. of Equations

E = 30000;              % Elastic Modulus (Pa)
G = E/(2+2*mu);
gamma = mu*E/((1+mu)*(1-2*mu));

% Define Coefficients ====================================================

% Crossover Terms
cuv5 = (gamma+G)/(4*h*k);;

% Define Boundary Conditions Based Upon Analytical Solution Across Infinite
%   Domain

kk1 = omega*sqrt(rho/G);
kk2 = omega*sqrt(rho)/sqrt(gamma+2*G);

xx1 = 0;
yy1 = (0:k:sampj)';

xx2 = sampi;
yy2 = (0:k:sampj)';
```

```
xx3 = (0:h:sampi);
yy3 = 0;

xx4 = (0:h:sampi)';
yy4 = sampj;

ux1 = zeros(gr+1,1);
vx1 = zeros(gr+1,1);
ux2 = zeros(gr+1,1);
vx2 = zeros(gr+1,1);
uy1 = zeros(gr-1,1);
vy1 = zeros(gr-1,1);
uy2 = zeros(gr-1,1);
vy2 = zeros(gr-1,1);

for i = 1:gr+1

    Cyy1 = yy1(i);
    Cyy2 = yy2(i);

    Cux1 = sin(kk1*xx1) + sin(kk1*Cyy1) + 2*sin(kk2*Cyy1);
    Cvx1 = 2*(-0.5*(-2*omega^2*rho*xx1 + 4*gamma + 8*G)*sqrt(G)*...
        sqrt(gamma+2*G)*cos(kk2*Cyy1) + omega*(-1*omega*(gamma/2+G)*...
        rho*cos(kk1*xx1)*Cyy1 + 0.5*sin(kk1*xx1)*sqrt(G)*(gamma+2*G)*...
        sqrt(rho)))/(sqrt(G*rho)*(gamma+2*G)*omega);

    Cux2 = sin(kk1*xx2) + sin(kk1*Cyy2) + 2*sin(kk2*Cyy2);
    Cvx2 = 2*(-0.5*(-2*omega^2*rho*xx2 + 4*gamma + 8*G)*sqrt(G)*sqrt(gamma+2*G)...
        *cos(kk2*Cyy2) + omega*(-1*omega*(gamma/2+G)*rho*cos(kk1*xx2)*Cyy2 + ...
        0.5*sin(kk1*xx2)*sqrt(G)*(gamma+2*G)*sqrt(rho)))/(sqrt(G*rho)*...
        (gamma+2*G)*omega);

    ux1(i) = Cux1;
    vx1(i) = Cvx1;
    ux2(i) = Cux2;
    vx2(i) = Cvx2;

end

for j = 2:gr

    Cxx3 = xx3(j);
    Cxx4 = xx4(j);

    Cuy1 = sin(kk1*Cxx3) + sin(kk1*yy3) + 2*sin(kk2*yy3);
    Cvy1 = 2*(-0.5*(-2*omega^2*rho*Cxx3 + 4*gamma + 8*G)*sqrt(G)*sqrt(gamma+2*G)...
        *cos(kk2*yy3) + omega*(-1*omega*(gamma/2+G)*rho*cos(kk1*Cxx3)*yy3 + ...
        0.5*sin(kk1*Cxx3)*sqrt(G)*(gamma+2*G)*sqrt(rho)))/(sqrt(G*rho)*...
        (gamma+2*G)*omega);

    Cuy2 = sin(kk1*Cxx4) + sin(kk1*yy4) + 2*sin(kk2*yy4);
    Cvy2 = 2*(-0.5*(-2*omega^2*rho*Cxx4 + 4*gamma + 8*G)*sqrt(G)*sqrt(gamma+2*G)...
        *cos(kk2*yy4) + omega*(-1*omega*(gamma/2+G)*rho*cos(kk1*Cxx4)*yy4 + ...
        0.5*sin(kk1*Cxx4)*sqrt(G)*(gamma+2*G)*sqrt(rho)))/(sqrt(G*rho)*...
        (gamma+2*G)*omega);

    uy1(j-1) = Cuy1;
    vy1(j-1) = Cvy1;
    uy2(j-1) = Cuy2;
    vy2(j-1) = Cvy2;

end

X1count = 1;
X2count = 1;
Y1count = 1;
Y2count = 1;
```

```
% Define System of Equations ================================================

% Initialize Vectors

iSuxx1 = zeros(3*totaln,1);
iSuyy1 = zeros(3*totaln,1);
iSufreq = zeros(1*totaln,1);
iSBC = zeros(1*totaln,1);

jSuxx1 = zeros(3*totaln,1);
jSuyy1 = zeros(3*totaln,1);
jSvxx1 = zeros(3*totaln,1);
jSvyy1 = zeros(3*totaln,1);
jSufreq = zeros(1*totaln,1);
jSBC = zeros(1*totaln,1);

sSuxx1 = zeros(3*totaln,1);
sSuyy1 = zeros(3*totaln,1);
sSufreq = zeros(3*totaln,1);
sSBC = zeros(3*totaln,1);

iRHSsparse = zeros(2*(grefinex+grefiney),1);
jRHSsparse = zeros(2*(grefinex+grefiney),1);
sRHSsparsex = zeros(2*(grefinex+grefiney),1);
sRHSsparsey = zeros(2*(grefinex+grefiney),1);

iCsparse = zeros(4*totaln,1);
jCsparse = zeros(4*totaln,1);
sCsparse = zeros(4*totaln,1);

% Initialize Counts

RHScount = 1;
Suxx1count = 1;
Suyy1count = 1;
Sufreqcount = 1;
SBCcount = 1;
Ccount = 1;

% Loop for Each Discrete Node to Formulate Appropriate Equations ----------

for q = 1:totaln

    % Specify Coordinates of Node of Interest
    j = ceil(q/ni);
    if rem(q,ni) ~= 0
        i = q - ni*floor(q/ni);
    else
        i = ni;
    end

    if i == 1       % Left Vertical Boundary - Type I BCs

        iqBC = q;
        jqBC = q;
        sqBC = 1;
        iRHS = q;
        jRHS = 1;
        sRHSx = ux1(X1count);
        sRHSy = vx1(X1count);

        iBC(SBCcount,1) = iqBC;
        jBC(SBCcount,1) = jqBC;
        sBC(SBCcount,1) = sqBC;

        iRHSsparse(RHScount,1) = iRHS;
        jRHSsparse(RHScount,1) = jRHS;
        sRHSsparsex(RHScount,1) = sRHSx;
```

```
        sRHSsparsey(RHScount,1) = sRHSy;

        X1count = X1count + 1;
        RHScount = RHScount + 1;
        SBCcount = SBCcount + 1;

    elseif i == ni  % Right Vertical Boundary - Type I BCs

        iqBC = q;
        jqBC = q;
        sqBC = 1;
        iRHS = q;
        jRHS = 1;
        sRHSx = ux2(X2count);
        sRHSy = vx2(X2count);

        iBC(SBCcount,1) = iqBC;
        jBC(SBCcount,1) = jqBC;
        sBC(SBCcount,1) = sqBC;

        iRHSsparse(RHScount,1) = iRHS;
        jRHSsparse(RHScount,1) = jRHS;
        sRHSsparsex(RHScount,1) = sRHSx;
        sRHSsparsey(RHScount,1) = sRHSy;

        X2count = X2count + 1;
        RHScount = RHScount + 1;
        SBCcount = SBCcount + 1;

    elseif j == 1   % Bottom Horizontal Boundary - Type I BCs

        iqBC = q;
        jqBC = q;
        sqBC = 1;
        iRHS = q;
        jRHS = 1;
        sRHSx = uy1(Y1count);
        sRHSy = vy1(Y1count);

        iBC(SBCcount,1) = iqBC;
        jBC(SBCcount,1) = jqBC;
        sBC(SBCcount,1) = sqBC;

        iRHSsparse(RHScount,1) = iRHS;
        jRHSsparse(RHScount,1) = jRHS;
        sRHSsparsex(RHScount,1) = sRHSx;
        sRHSsparsey(RHScount,1) = sRHSy;

        Y1count = Y1count + 1;
        RHScount = RHScount + 1;
        SBCcount = SBCcount + 1;
    elseif j == nj  % Top Horizontal Boundary - Type I BCs

        iqBC = q;
        jqBC = q;
        sqBC = 1;
        iRHS = q;
        jRHS = 1;
        sRHSx = uy2(Y2count);
        sRHSy = vy2(Y2count);

        iBC(SBCcount,1) = iqBC;
        jBC(SBCcount,1) = jqBC;
        sBC(SBCcount,1) = sqBC;

        iRHSsparse(RHScount,1) = iRHS;
        jRHSsparse(RHScount,1) = jRHS;
        sRHSsparsex(RHScount,1) = sRHSx;
```

```
        sRHSsparsey(RHScount,1) = sRHSy;

        Y2count = Y2count + 1;
        RHScount = RHScount + 1;
        SBCcount = SBCcount + 1;

    else

        % Apply Finite Differnce Approximations to Centre Nodes -----------

        % Suxx1/Svyy1 ------------------------------------------------------

        iqSuxx1 = ones(3,1)*q;
        jqSuxx1 = [q-1 q q+1]';
        sqSuxx1 = [(gamma+2*G)/h^2 -2*(gamma+2*G)/h^2 (gamma+2*G)/h^2]';

        jqSvyy1 = [q-ni q q+ni]';

        iSuxx1(Suxx1count:Suxx1count+2,1) = iqSuxx1;
        jSuxx1(Suxx1count:Suxx1count+2,1) = jqSuxx1;
        sSuxx1(Suxx1count:Suxx1count+2,1) = sqSuxx1;

        jSvyy1(Suxx1count:Suxx1count+2,1) = jqSvyy1;

        Suxx1count = Suxx1count + 3;

        % Suyy1/Svxx1 ------------------------------------------------------

        iqSuyy1 = ones(3,1)*q;
        jqSuyy1 = [q-ni q q+ni]';
        sqSuyy1 = [G/h^2 -2*G/h^2 G/h^2]';

        jqSvxx1 = [q-1 q q+1]';

        iSuyy1(Suyy1count:Suyy1count+2,1) = iqSuyy1;
        jSuyy1(Suyy1count:Suyy1count+2,1) = jqSuyy1;
        sSuyy1(Suyy1count:Suyy1count+2,1) = sqSuyy1;

        jSvxx1(Suyy1count:Suyy1count+2,1) = jqSvxx1;

        Suyy1count = Suyy1count + 3;

        % Sufreq -----------------------------------------------------------

        iqSufreq = q;
        jqSufreq = q;
        sqSufreq = rho*omega^2;

        iSufreq(Sufreqcount,1) = iqSufreq;
        jSufreq(Sufreqcount,1) = jqSufreq;
        sSufreq(Sufreqcount,1) = sqSufreq;
        Sufreqcount = Sufreqcount + 1;

        % Cross-Over Terms-------------------------------------------------

        iq = ones(4,1)*q;
        jq = [(q-ni-1) (q-ni+1) (q+ni-1) (q+ni+1)]';
        sq = [cuv5 -cuv5 -cuv5 cuv5]';

        iCsparse(Ccount:Ccount+3,1) = iq;
        jCsparse(Ccount:Ccount+3,1) = jq;
        sCsparse(Ccount:Ccount+3,1) = sq;

        Ccount = Ccount + 4;

        %------------------------------------------------------------------
    end
end
```

```
iSuxx1 = iSuxx1(find(iSuxx1));
iSuyy1 = iSuyy1(find(iSuyy1));
iSufreq = iSufreq(find(iSufreq));
iBC = iBC(find(iBC));

jSuxx1 = jSuxx1(find(jSuxx1));
jSuyy1 = jSuyy1(find(jSuyy1));
jSvyy1 = jSvyy1(find(jSvyy1));
jSvxx1 = jSvxx1(find(jSvxx1));
jSufreq = jSufreq(find(iSufreq));
jBC = jBC(find(jBC));

sSuxx1 = sSuxx1(find(sSuxx1));
sSuyy1 = sSuyy1(find(sSuyy1));
sSufreq = sSufreq(find(sSufreq));
sBC = sBC(find(sBC));

% Convert i, j and s Vectors to Sparse Matrix Format ----------------------

Suxx1 = sparse(iSuxx1,jSuxx1,sSuxx1,ni*nj,ni*nj);
Suyy1 = sparse(iSuyy1,jSuyy1,sSuyy1,ni*nj,ni*nj);
Svyy1 = sparse(iSuxx1,jSvyy1,sSuxx1,ni*nj,ni*nj);
Svxx1 = sparse(iSuyy1,jSvxx1,sSuyy1,ni*nj,ni*nj);
Sufreq = sparse(iSufreq,jSufreq,sSufreq,ni*nj,ni*nj);
SBC = sparse(iBC,jBC,sBC,ni*nj,ni*nj);

Sx = (Suxx1+Suyy1+Sufreq+SBC);
Sy = (Svyy1+Svxx1+Sufreq+SBC);

iRHSsparse = [iRHSsparse; (iRHSsparse+totaln)];
jRHSsparse = [jRHSsparse; jRHSsparse];
sRHSsparse = [sRHSsparsex; sRHSsparsey];

Srhs = sparse(iRHSsparse,jRHSsparse,sRHSsparse,2*ni*nj,1);

iCsparse = iCsparse(find(iCsparse));
jCsparse = jCsparse(find(jCsparse));
sCsparse = sCsparse(find(sCsparse));

Sc = sparse(iCsparse,jCsparse,sCsparse,ni*nj,ni*nj);

% Combine Matrices & Solve System of Equations-----------------------------

Sfull = [Sx Sc; Sc Sy];
a0 = Sfull\Srhs;
a1 = full(a0);

% Post-Processing ============================================================

u1 = a1(1:totaln);
v1 = a1(totaln+1:2*totaln);

u2 = u1;
v2 = v1;

U0 = [];
V0 = [];

for i = 1:nj

    U0 = [U0; u2((nj-i)*ni+1:(nj-i+1)*ni)'];
    V0 = [V0; v2((nj-i)*ni+1:(nj-i+1)*ni)'];

end

tt1 = cputime-tt0
```

```
% Analytical Comparison ------------------------------------------------

Au = zeros(totaln,1);
Av = zeros(totaln,1);
u_xx = zeros(totaln,1);
u_yy = zeros(totaln,1);
v_xx = zeros(totaln,1);
v_yy = zeros(totaln,1);
u_xy = zeros(totaln,1);
v_xy = zeros(totaln,1);

count = 1;

for j = 1:nj
    for i = 1:ni

    xx = (i-1)*h;
    yy = (j-1)*k;

    cAu = sin(kk1*xx) + sin(kk1*yy) + 2*sin(kk2*yy);
    cAv = 2*(-0.5*(-2*omega^2*rho*xx + 4*gamma + 8*G)*sqrt(G)*...
        sqrt(gamma+2*G)*cos(kk2*yy) + omega*(-1*omega*(gamma/2+G)*...
        rho*cos(kk1*xx)*yy + 0.5*sin(kk1*xx)*sqrt(G)*(gamma+2*G)*...
        sqrt(rho)))/(sqrt(G*rho)*(gamma+2*G)*omega);

    Au(count)= cAu;
    Av(count)= cAv;

    u_xx(count) = -sin(kk1*xx)*omega^2*rho/G;

    u_yy(count) = -sin(kk1*yy)*omega^2*rho/G...
        - 2*sin(kk2*yy)*omega^2*rho/(gamma+2*G);

    v_xx(count) = 2*(yy*omega^3*(gamma/2+G)*rho^2*cos(kk1*xx)/G...
        - 1/2*sin(kk1*xx)*omega^2*rho^(3/2)*(gamma+2*G)/sqrt(G))...
    /(sqrt(G*rho)*(gamma+2*G));

    v_yy(count) = ((-2*omega^2*rho*xx+4*gamma+8*G)*cos(kk2*yy)...
        *omega*sqrt(rho))/(gamma+2*G)^(3/2);

    v_xy(count) = 2*(-omega^3*rho^(3/2)*sqrt(G)*sin(kk2*yy)...
        + (omega^3*(gamma/2+G)*rho^(3/2)*sin(kk1*xx))/sqrt(G))...
        /(sqrt(G)*sqrt(rho)*(gamma+2*G)*omega);

    count = count + 1;
    end
end

% u_xxtest = (Suxx1 + 0*Suxx2 + 0*Suxx3)*Au/(gamma+2*G) - u_xx;
% u_yytest = (Suyy1 + Suyy2 + Suyy3)*Au/G - u_yy;
% v_xxtest = (Suxx1 + Suxx2 + Suxx3)*Av/(gamma+2*G) - v_xx;
% v_yytest = (Suyy1 + Suyy2 + Suyy3)*Av/G - v_yy;
% u_xytest = Sc*Au/(gamma+G) - u_xy;
% v_xytest = Sc*Av/(gamma+G) - v_xy;
%
% u_xx_approx = (Suxx1 + 0*Suxx2 + 0*Suxx3)*Au/(gamma+2*G);
% u_yy_approx = (Suyy1 + Suyy2 + Suyy3)*Au/G;
% v_xx_approx = (Suxx1 + Suxx2 + Suxx3)*Av/(gamma+2*G);
% v_yy_approx = (Suyy1 + Suyy2 + Suyy3)*Av/G;
% u_xy_approx = Sc*Au/(gamma+G);
% v_xy_approx = Sc*Av/(gamma+G);

AU0 = [];
AV0 = [];

for i = 1:nj

    AU0 = [AU0; Au((nj-i)*ni+1:(nj-i+1)*ni)'];
```

```
    AV0 = [AV0; Av((nj-i)*ni+1:(nj-i+1)*ni)'];

end

% Plot Comparative Results & Evaluate Error -------------------------------

% Surface Plot comparing x direction displacment amplitudes
figure(1)
subplot(1,2,1)
surf(U0)
title('U Disp - Numerical')
shading interp
subplot(1,2,2)
surf(AU0)
title('U Disp - Analytical')
shading interp

% Surface Plot comparing y direction displacment amplitudes
figure(2)
subplot(1,2,1)
surf(V0)
title('V Disp - Numerical')
shading interp
subplot(1,2,2)
surf(AV0)
title('V Disp - Analytical')
shading interp

paxis1 = 0:1:totaln-1;

EU0 = (U0-AU0)./(max(max(abs(AU0))))*100;
EV0 = (V0-AV0)./(max(max(abs(AV0))))*100;

% Calculating Maximum Error
emaxU0 = max(max(abs(U0-AU0)))/(mean(mean(abs(AU0))))*100
emaxV0 = max(max(abs(V0-AV0)))/(mean(mean(abs(AV0))))*100

nU0 = [];
nV0 = [];

for i = 1:nj
    nU0 = [nU0; u2(1:ni)'];
    nV0 = [nV0; v2(1:ni)'];

    u2 = u2((ni+1):length(u2));
    v2 = v2((ni+1):length(v2));
end

U0 = [];
V0 = [];

for i = 1:nj
    U0 = [U0; u1(1:ni)'];
    V0 = [V0; v1(1:ni)'];

    u1 = u1((ni+1):length(u1));
    v1 = v1((ni+1):length(v1));

end

xx1 = (0:h:sampi);
yy1 = (0:k:sampj)';
[X,Y] = meshgrid(xx1,yy1);

% 2D Dot Plot
figure(3)
hold on
for i = 1:length(xx1)
```

```
    plot(X(:,i) + U0(:,i)./1000,Y(:,i) + V0(:,i)./1000,'.');
end

save Infinite100Hz-200-2323T U0 V0 omega

%-------------------------------------------------------------------------
```

A6: 2D Non-Homogeneous Forward Simulation Algorithm – Phantom Boundary Condition Model

```
%=========================================================================

% 2D NON-HOMOGENEOUS FORWARD SIMULATION ALGORITHM - PHANTOM B.C.
% By Samuel J. Houghton
% 2006

% This algorithm uses Finite Difference Approximations to solve for the
% Harmonic Displacement Amplitudes from the 2D Plain Strain Navier's
% Equations.  The medium has pieceweise constant stiffness, with stress
% continuity enforced at the boundary of regions of differing stiffness.
% The carcinoma can be any rectangular shape positioned away from the
% boundary of the global domain.

% This model adopts boundary conditions where the global domain is
% excited by shear (/or longitudinal) sinusoidal vibrations with constant
% displacment amplitudes along a single boundary of the square global
% domain.  This is done using Type I BCs.

% The other 3 boundaries of the global domain are modelled as free
% boundaries with zero shear and longitudinal stresses being enforced.
% These are Type II BCs with derivative terms being approximated by Finite
% Differences.

% This boundary condition model represents the primary model used for the
% analysis of the inverse algorithms and the 'Box Shake' and 'Edge Effect'
% BC models can easily be created by adapting this model where appropriate.

%=========================================================================

clear
clc

tt0 = cputime;

% Definition of Variables ================================================

rho = 1020;          % Density (kg/m3)
omega = 2*pi*50;     % Actuation Frequency (rad/s)
poisson = 0.49;      % Poisson's Ratio

% Specify Displacement Amplitudes along Boundary
uamp0 = 0.000;           % u displacement (m)
vamp0 = 0.001;           % v displacement (m)

% Size of Global Domain (m)
sampi = 0.1;
sampj = 0.1;

gr = 200;                % Grid Refinement - !!Must be divisible by 10!!
grefinex = gr;
grefiney = gr;
h = sampj/grefinex;      % Step Size
```

```
k = sampi/grefiney;

nj = sampj/h + 1;
ni = sampi/k + 1;
totaln = ni*nj;          % Total No. of Equations

E1 = 30000;              % Define Young's Modulus of Healthy Tissue
E2 = 10*30000;           % Define Young's Modulus of Carcinoma

G1 = E1/(2+2*poisson);
gamma1 = poisson*E1/((1+poisson)*(1-2*poisson));
G2 = E2/(2+2*poisson);
gamma2 = poisson*E2/((1+poisson)*(1-2*poisson));

V = poisson;

% Geometrically Define Carcinoma Position --------------------------------

x1coord = .02;
x2coord = .03;
y1coord = .02;
y2coord = .03;

% Use Geometric Carcinoma Position to Identify Type of Equations applied to
% each individual node ---------------------------------------------------

% Corners numbered clockwise with top left being no. 1
ic1 = x1coord/sampi*grefinex+1;
jc1 = y1coord/sampj*grefiney+1;
corner1 = [ic1 jc1];
ic2 = x2coord/sampi*grefinex+1;
jc2 = y1coord/sampj*grefiney+1;
corner2 = [ic2 jc2];
ic3 = x2coord/sampi*grefinex+1;
jc3 = y2coord/sampj*grefiney+1;
corner3 = [ic3 jc3];
ic4 = x1coord/sampi*grefinex+1;
jc4 = y2coord/sampj*grefiney+1;
corner4 = [ic4 jc4];

x1eqn = x1coord/sampi*grefinex+1;
x2eqn = x2coord/sampi*grefinex+1;
y1eqn = y1coord/sampj*grefiney+1;
y2eqn = y2coord/sampj*grefiney+1;

yeqnbound = y1eqn:1:y2eqn;
xeqnbound = x1eqn:1:x2eqn;

lenxeqnb = length(xeqnbound);
lenyeqnb = length(yeqnbound);

Tq = zeros(lenxeqnb*lenyeqnb,1);
for i = 1:lenyeqnb
    for j = 1:lenxeqnb
        Tq((i-1)*lenxeqnb+j,1) = (yeqnbound(i)-1)*ni+xeqnbound(j);
    end
end

Tqcorner1 = Tq(1);
Tqcorner2 = Tq(lenxeqnb);
Tqcorner3 = Tq(lenyeqnb*lenxeqnb);
Tqcorner4 = Tq((lenyeqnb-1)*lenxeqnb+1);

Tqtophoriedge = Tq(2:lenxeqnb-1,1);
Tqbothoriedge = Tq((lenyeqnb-1)*lenxeqnb+2:lenyeqnb*lenxeqnb-1);

Tqleftvertedge = zeros(lenyeqnb-2,1);
Tqrighvertedge = zeros(lenyeqnb-2,1);
```

```
for i = 1:lenyeqnb-2
    Tqleftvertedge(i,1) = Tq(i*lenxeqnb+1);
    Tqrighvertedge(i,1) = Tq((i+1)*lenxeqnb);
end

Tqcentre = zeros(lenyeqnb*lenxeqnb-2*(lenyeqnb+lenxeqnb-2),1);
for i = 1:lenyeqnb-2
    Tqcentre((i-1)*(lenxeqnb-2)+1:i*(lenxeqnb-2),1) = Tq(i*lenxeqnb+2:(i+1)*lenxeqnb-
1);
end

% Define System of Equations ================================================

nterm = 1;

cc1 = 2*G1 + gamma1;
cc2 = 2*G2 + gamma2;

% Crossover Terms
Hcuv5 = (gamma1+G1)/(4*h*k)/nterm;
Tcuv5 = (gamma2+G2)/(4*h*k)/nterm;

% Initialize Vectors

iSuxx1 = zeros(3*totaln,1);
iSuyy1 = zeros(3*totaln,1);
iSufreq = zeros(1*totaln,1);
iSXBC = zeros(1*totaln,1);
iSYBC = zeros(1*totaln,1);

jSuxx1 = zeros(3*totaln,1);
jSuyy1 = zeros(3*totaln,1);
jSvxx1 = zeros(3*totaln,1);
jSvyy1 = zeros(3*totaln,1);
jSufreq = zeros(1*totaln,1);
jSXBC = zeros(1*totaln,1);
jSYBC = zeros(1*totaln,1);

sSuxx1 = zeros(3*totaln,1);
sSuyy1 = zeros(3*totaln,1);
sSufreq = zeros(3*totaln,1);
sSXBC = zeros(3*totaln,1);
sSYBC = zeros(3*totaln,1);

iRHSsparse = zeros(grefiney+1,1);
jRHSsparse = zeros(grefiney+1,1);
sRHSsparsex = zeros(grefiney+1,1);
sRHSsparsey = zeros(grefiney+1,1);

iCXsparse = zeros(4*totaln,1);
jCXsparse = zeros(4*totaln,1);
sCXsparse = zeros(4*totaln,1);

iCYsparse = zeros(4*totaln,1);
jCYsparse = zeros(4*totaln,1);
sCYsparse = zeros(4*totaln,1);

% Initialize Counts

RHScount = 1;
Suxx1count = 1;
Suyy1count = 1;
Sufreqcount = 1;
SXBCcount = 1;
SYBCcount = 1;
CXcount = 1;
CYcount = 1;
```

```
tophoricount = 1;
bothoricount = 1;
leftvertcount = 1;
righvertcount = 1;
centrecount = 1;

% Loop for Each Discrete Node to Formulate Appropriate Equations ----------

for q = 1:totaln

    % Specify Coordinates of Node of Interest
    i = ceil(q/ni);
    if rem(q,ni) ~= 0
        j = q - ni*floor(q/ni);
    else
        j = ni;
    end

    % Apply Boundary Conditions ------------------------------------------

    if j == 1       % Left Vertical Boundary - Type I BCs
        iqx = q;
        jqx = q;
        iqy = q;
        jqy = q;
        sqx = 1;
        sqy = 1;

        iRHS = q;
        jRHS = 1;
        sRHSx = uamp0;
        sRHSy = vamp0;

        iRHSsparse(RHScount,1) = iRHS;
        jRHSsparse(RHScount,1) = jRHS;
        sRHSsparsex(RHScount,1) = sRHSx;
        sRHSsparsey(RHScount,1) = sRHSy;

        iXBC(SXBCcount,1) = iqx;
        iYBC(SYBCcount,1) = iqy;
        jXBC(SXBCcount,1) = jqx;
        jYBC(SYBCcount,1) = jqy;
        sXBC(SXBCcount,1) = sqx;
        sYBC(SYBCcount,1) = sqy;

        RHScount = RHScount + 1;
        SXBCcount = SXBCcount + 1;
        SYBCcount = SYBCcount + 1;

    elseif j == nj  % Right Vertical Boundary - Type II BCs

        if i == 1   % Bottom Right Corner Node - Type II BCs

            % Theta X Equation
            iqx = ones(3,1)*q;
            jqx = [q q-1 q-2]';
            sqx = [3*(2*G1+gamma1) -4*(2*G1+gamma1) 1*(2*G1+gamma1)]';
            iqCx = ones(3,1)*q;
            jqCx = [q q+ni q+2*ni];
            sqCx = [-3*gamma1 4*gamma1 -1*gamma1]';

            % Shear Stress Equation
            iqy = ones(3,1)*q;
            jqy = [q q-1 q-2]';
            sqy = [3 -4 1]';
            iqCy = ones(3,1)*q;
            jqCy = [q q+ni q+2*ni]';
            sqCy = [-3 4 -1]';
```

```
    iXBC(SXBCcount:SXBCcount+2,1) = iqx;
    jXBC(SXBCcount:SXBCcount+2,1) = jqx;
    sXBC(SXBCcount:SXBCcount+2,1) = sqx;

    iYBC(SYBCcount:SYBCcount+2,1) = iqy;
    jYBC(SYBCcount:SYBCcount+2,1) = jqy;
    sYBC(SYBCcount:SYBCcount+2,1) = sqy;

    iCXsparse(CXcount:CXcount+2,1) = iqCx;
    jCXsparse(CXcount:CXcount+2,1) = jqCx;
    sCXsparse(CXcount:CXcount+2,1) = sqCx;

    iCYsparse(CYcount:CYcount+2,1) = iqCy;
    jCYsparse(CYcount:CYcount+2,1) = jqCy;
    sCYsparse(CYcount:CYcount+2,1) = sqCy;

    SXBCcount = SXBCcount + 3;
    SYBCcount = SYBCcount + 3;
    CXcount = CXcount + 3;
    CYcount = CYcount + 3;

elseif i == ni  % Top Right Corner Node - Type II BCs

    % Theta X Equation
    iqx = ones(3,1)*q;
    jqx = [q q-1 q-2]';
    sqx = [3*(2*G1+gamma1) -4*(2*G1+gamma1) 1*(2*G1+gamma1)]';
    iqCx = ones(3,1)*q;
    jqCx = [q q-ni q-2*ni];
    sqCx = [3*gamma1 -4*gamma1 1*gamma1]';

    % Shear Stress Equation
    iqy = ones(3,1)*q;
    jqy = [q q-1 q-2]';
    sqy = [3 -4 1]';
    iqCy = ones(3,1)*q;
    jqCy = [q q-ni q-2*ni]';
    sqCy = [3 -4 1]';

    iXBC(SXBCcount:SXBCcount+2,1) = iqx;
    jXBC(SXBCcount:SXBCcount+2,1) = jqx;
    sXBC(SXBCcount:SXBCcount+2,1) = sqx;

    iYBC(SYBCcount:SYBCcount+2,1) = iqy;
    jYBC(SYBCcount:SYBCcount+2,1) = jqy;
    sYBC(SYBCcount:SYBCcount+2,1) = sqy;

    iCXsparse(CXcount:CXcount+2,1) = iqCx;
    jCXsparse(CXcount:CXcount+2,1) = jqCx;
    sCXsparse(CXcount:CXcount+2,1) = sqCx;
    iCYsparse(CYcount:CYcount+2,1) = iqCy;
    jCYsparse(CYcount:CYcount+2,1) = jqCy;
    sCYsparse(CYcount:CYcount+2,1) = sqCy;

    SXBCcount = SXBCcount + 3;
    SYBCcount = SYBCcount + 3;
    CXcount = CXcount + 3;
    CYcount = CYcount + 3;

else

    % Theta X Equation
    iqx = ones(3,1)*q;
    jqx = [q q-1 q-2]';
    sqx = [3*(2*G1+gamma1) -4*(2*G1+gamma1) 1*(2*G1+gamma1)]';
    iqCx = ones(2,1)*q;
    jqCx = [q-ni q+ni];
    sqCx = [-gamma1 gamma1]';
```

```
            % Shear Stress Equation
            iqy = ones(3,1)*q;
            jqy = [q q-1 q-2]';
            sqy = [3 -4 1]';
            iqCy = ones(2,1)*q;
            jqCy = [q-ni q+ni]';
            sqCy = [-1 1]';

            iXBC(SXBCcount:SXBCcount+2,1) = iqx;
            jXBC(SXBCcount:SXBCcount+2,1) = jqx;
            sXBC(SXBCcount:SXBCcount+2,1) = sqx;

            iYBC(SYBCcount:SYBCcount+2,1) = iqy;
            jYBC(SYBCcount:SYBCcount+2,1) = jqy;
            sYBC(SYBCcount:SYBCcount+2,1) = sqy;

            iCXsparse(CXcount:CXcount+1,1) = iqCx;
            jCXsparse(CXcount:CXcount+1,1) = jqCx;
            sCXsparse(CXcount:CXcount+1,1) = sqCx;

            iCYsparse(CYcount:CYcount+1,1) = iqCy;
            jCYsparse(CYcount:CYcount+1,1) = jqCy;
            sCYsparse(CYcount:CYcount+1,1) = sqCy;

            SXBCcount = SXBCcount + 3;
            SYBCcount = SYBCcount + 3;
            CXcount = CXcount + 2;
            CYcount = CYcount + 2;
        end

    elseif i == 1   % Bottom Horizontal Boundary - Type II BCs

        % Shear Stress Equation
        iqx = ones(3,1)*q;
        jqx = [q q+ni q+2*ni]';
        sqx = [-3 4 -1]';
        iqCx = ones(2,1)*q;
        jqCx = [q-1 q+1]';
        sqCx = [-1 1]';

        % Theta Y Equation
        iqy = ones(3,1)*q;
        jqy = [q q+ni q+2*ni]';
        sqy = [-3*cc1 4*cc1 -cc1]';
        iqCy = ones(2,1)*q;
        jqCy = [q-1 q+1]';
        sqCy = [-gamma1 gamma1]';

        iXBC(SXBCcount:SXBCcount+2,1) = iqx;
        jXBC(SXBCcount:SXBCcount+2,1) = jqx;
        sXBC(SXBCcount:SXBCcount+2,1) = sqx;

        iYBC(SYBCcount:SYBCcount+2,1) = iqy;
        jYBC(SYBCcount:SYBCcount+2,1) = jqy;
        sYBC(SYBCcount:SYBCcount+2,1) = sqy;

        iCXsparse(CXcount:CXcount+1,1) = iqCx;
        jCXsparse(CXcount:CXcount+1,1) = jqCx;
        sCXsparse(CXcount:CXcount+1,1) = sqCx;

        iCYsparse(CYcount:CYcount+1,1) = iqCy;
        jCYsparse(CYcount:CYcount+1,1) = jqCy;
        sCYsparse(CYcount:CYcount+1,1) = sqCy;

        SXBCcount = SXBCcount + 3;
        SYBCcount = SYBCcount + 3;
        CXcount = CXcount + 2;
        CYcount = CYcount + 2;
```

```
elseif i == ni  % Top Horizontal Boundary - Type II BCs

    % Shear Stress Equation
    iqx = ones(3,1)*q;
    jqx = [q q-ni q-2*ni]';
    sqx = [3 -4 1]';
    iqCx = ones(2,1)*q;
    jqCx = [q-1 q+1]';
    sqCx = [-1 1]';

    % Theta Y Equation
    iqy = ones(3,1)*q;
    jqy = [q q-ni q-2*ni]';
    sqy = [3*cc1 -4*cc1 cc1]';
    iqCy = ones(2,1)*q;
    jqCy = [q-1 q+1]';
    sqCy = [-gamma1 gamma1]';

    iXBC(SXBCcount:SXBCcount+2,1) = iqx;
    jXBC(SXBCcount:SXBCcount+2,1) = jqx;
    sXBC(SXBCcount:SXBCcount+2,1) = sqx;

    iYBC(SYBCcount:SYBCcount+2,1) = iqy;
    jYBC(SYBCcount:SYBCcount+2,1) = jqy;
    sYBC(SYBCcount:SYBCcount+2,1) = sqy;

    iCXsparse(CXcount:CXcount+1,1) = iqCx;
    jCXsparse(CXcount:CXcount+1,1) = jqCx;
    sCXsparse(CXcount:CXcount+1,1) = sqCx;

    iCYsparse(CYcount:CYcount+1,1) = iqCy;
    jCYsparse(CYcount:CYcount+1,1) = jqCy;
    sCYsparse(CYcount:CYcount+1,1) = sqCy;

    SXBCcount = SXBCcount + 3;
    SYBCcount = SYBCcount + 3;
    CXcount = CXcount + 2;
    CYcount = CYcount + 2;

    % Apply Stress Continuity Equations to Stiffness Boundary ---------

elseif q == Tqcorner1

    % Equation 1
    iqx = ones(5,1)*q;
    jqx = [q-2 q-1 q q+1 q+2]';
    sqx = [cc1 -4*cc1 3*(cc1+cc2) -4*cc2 cc2]';

    iqCx = ones(5,1)*q;
    jqCx = [q-2*ni q-ni q q+ni q+2*ni]';
    sqCx = [gamma1 -4*gamma1 3*(gamma1+gamma2) -4*gamma2 gamma2]';

    % Equation2
    iqy = ones(5,1)*q;
    jqy = [q-2*ni q-ni q q+ni q+2*ni]';
    sqy = [cc1 -4*cc1 3*(cc1+cc2) -4*cc2 cc2]';

    iqCy = ones(5,1)*q;
    jqCy = [q-2 q-1 q q+1 q+2]';
    sqCy = [gamma1 -4*gamma1 3*(gamma1+gamma2) -4*gamma2 gamma2]';

    % Add terms to vectors
    iXBC(SXBCcount:SXBCcount+4,1) = iqx;
    jXBC(SXBCcount:SXBCcount+4,1) = jqx;
    sXBC(SXBCcount:SXBCcount+4,1) = sqx;

    iYBC(SYBCcount:SYBCcount+4,1) = iqy;
    jYBC(SYBCcount:SYBCcount+4,1) = jqy;
```

```
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;

        iCXsparse(CXcount:CXcount+4,1) = iqCx;
        jCXsparse(CXcount:CXcount+4,1) = jqCx;
        sCXsparse(CXcount:CXcount+4,1) = sqCx;

        iCYsparse(CYcount:CYcount+4,1) = iqCy;
        jCYsparse(CYcount:CYcount+4,1) = jqCy;
        sCYsparse(CYcount:CYcount+4,1) = sqCy;

        CXcount = CXcount + 5;
        CYcount = CYcount + 5;

    elseif q == Tqcorner2

        % Equation 1
        iqx = ones(5,1)*q;
        jqx = [q-2 q-1 q q+1 q+2]';
        sqx = [cc2 -4*cc2 3*(cc2+cc1) -4*cc1 cc1]';

        iqCx = ones(5,1)*q;
        jqCx = [q-2*ni q-ni q q+ni q+2*ni]';
        sqCx = [gamma1 -4*gamma1 3*(gamma1+gamma2) -4*gamma2 gamma2]';

        % Equation 2
        iqy = ones(5,1)*q;
        jqy = [q-2*ni q-ni q q+ni q+2*ni]';
        sqy = [cc1 -4*cc1 3*(cc1+cc2) -4*cc2 cc2]';

        iqCy = ones(5,1)*q;
        jqCy = [q-2 q-1 q q+1 q+2]';
        sqCy = [gamma2 -4*gamma2 3*(gamma2+gamma1) -4*gamma1 gamma1]';

        % Add terms to vectors
        iXBC(SXBCcount:SXBCcount+4,1) = iqx;
        jXBC(SXBCcount:SXBCcount+4,1) = jqx;
        sXBC(SXBCcount:SXBCcount+4,1) = sqx;

        iYBC(SYBCcount:SYBCcount+4,1) = iqy;
        jYBC(SYBCcount:SYBCcount+4,1) = jqy;
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;

        iCXsparse(CXcount:CXcount+4,1) = iqCx;
        jCXsparse(CXcount:CXcount+4,1) = jqCx;
        sCXsparse(CXcount:CXcount+4,1) = sqCx;

        iCYsparse(CYcount:CYcount+4,1) = iqCy;
        jCYsparse(CYcount:CYcount+4,1) = jqCy;
        sCYsparse(CYcount:CYcount+4,1) = sqCy;

        CXcount = CXcount + 5;
        CYcount = CYcount + 5;


    elseif q == Tqcorner3

        % Equation 1
        iqx = ones(5,1)*q;
        jqx = [q-2 q-1 q q+1 q+2]';
        sqx = [cc2 -4*cc2 3*(cc2+cc1) -4*cc1 cc1]';

        iqCx = ones(5,1)*q;
```

```
        jqCx = [q-2*ni q-ni q q+ni q+2*ni]';
        sqCx = [gamma2 -4*gamma2 3*(gamma2+gamma1) -4*gamma1 gamma1]';

        % Equation2
        iqy = ones(5,1)*q;
        jqy = [q-2*ni q-ni q q+ni q+2*ni]';
        sqy = [cc2 -4*cc2 3*(cc2+cc1) -4*cc1 cc1]';

        iqCy = ones(5,1)*q;
        jqCy = [q-2 q-1 q q+1 q+2]';
        sqCy = [gamma2 -4*gamma2 3*(gamma2+gamma1) -4*gamma1 gamma1]';

        % Add terms to vectors
        iXBC(SXBCcount:SXBCcount+4,1) = iqx;
        jXBC(SXBCcount:SXBCcount+4,1) = jqx;
        sXBC(SXBCcount:SXBCcount+4,1) = sqx;

        iYBC(SYBCcount:SYBCcount+4,1) = iqy;
        jYBC(SYBCcount:SYBCcount+4,1) = jqy;
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;

        iCXsparse(CXcount:CXcount+4,1) = iqCx;
        jCXsparse(CXcount:CXcount+4,1) = jqCx;
        sCXsparse(CXcount:CXcount+4,1) = sqCx;

        iCYsparse(CYcount:CYcount+4,1) = iqCy;
        jCYsparse(CYcount:CYcount+4,1) = jqCy;
        sCYsparse(CYcount:CYcount+4,1) = sqCy;

        CXcount = CXcount + 5;
        CYcount = CYcount + 5;

    elseif q == Tqcorner4

        % Equation 1
        iqx = ones(5,1)*q;
        jqx = [q-2 q-1 q q+1 q+2]';
        sqx = [cc1 -4*cc1 3*(cc1+cc2) -4*cc2 cc2]';

        iqCx = ones(5,1)*q;
        jqCx = [q-2*ni q-ni q q+ni q+2*ni]';
        sqCx = [gamma2 -4*gamma2 3*(gamma2+gamma1) -4*gamma1 gamma1]';

        % Equation2
        iqy = ones(5,1)*q;
        jqy = [q-2*ni q-ni q q+ni q+2*ni]';
        sqy = [cc2 -4*cc2 3*(cc2+cc1) -4*cc1 cc1]';

        iqCy = ones(5,1)*q;
        jqCy = [q-2 q-1 q q+1 q+2]';
        sqCy = [gamma1 -4*gamma1 3*(gamma1+gamma2) -4*gamma2 gamma2]';

        % Add terms to vectors
        iXBC(SXBCcount:SXBCcount+4,1) = iqx;
        jXBC(SXBCcount:SXBCcount+4,1) = jqx;
        sXBC(SXBCcount:SXBCcount+4,1) = sqx;

        iYBC(SYBCcount:SYBCcount+4,1) = iqy;
        jYBC(SYBCcount:SYBCcount+4,1) = jqy;
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;

        iCXsparse(CXcount:CXcount+4,1) = iqCx;
```

```
        jCXsparse(CXcount:CXcount+4,1) = jqCx;
        sCXsparse(CXcount:CXcount+4,1) = sqCx;

        iCYsparse(CYcount:CYcount+4,1) = iqCy;
        jCYsparse(CYcount:CYcount+4,1) = jqCy;
        sCYsparse(CYcount:CYcount+4,1) = sqCy;

        CXcount = CXcount + 5;
        CYcount = CYcount + 5;

    elseif q == Tqtophoriedge(tophoricount)

        % Shear Stress Continuity Equation
        iqx = ones(5,1)*q;
        jqx = [q-2*ni q-ni q q+ni q+2*ni]';
        sqx = [G1 -4*G1 3*(G1+G2) -4*G2 G2]';

        iqCx = ones(2,1)*q;
        jqCx = [q-1 q+1]';
        sqCx = [(-G1+G2) (G1-G2)]';

        % Normal Stress Continuity Equation
        iqy = ones(5,1)*q;
        jqy = [q-2*ni q-ni q q+ni q+2*ni]';
        sqy = [cc1 -4*cc1 3*(cc1+cc2) -4*cc2 cc2]';

        iqCy = ones(2,1)*q;
        jqCy = [q-1 q+1]';
        sqCy = [(-gamma1+gamma2) (gamma1-gamma2)]';

        % Add terms to vectors
        iXBC(SXBCcount:SXBCcount+4,1) = iqx;
        jXBC(SXBCcount:SXBCcount+4,1) = jqx;
        sXBC(SXBCcount:SXBCcount+4,1) = sqx;

        iYBC(SYBCcount:SYBCcount+4,1) = iqy;
        jYBC(SYBCcount:SYBCcount+4,1) = jqy;
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;

        iCXsparse(CXcount:CXcount+1,1) = iqCx;
        jCXsparse(CXcount:CXcount+1,1) = jqCx;
        sCXsparse(CXcount:CXcount+1,1) = sqCx;

        iCYsparse(CYcount:CYcount+1,1) = iqCy;
        jCYsparse(CYcount:CYcount+1,1) = jqCy;
        sCYsparse(CYcount:CYcount+1,1) = sqCy;

        CXcount = CXcount + 2;
        CYcount = CYcount + 2;

        if tophoricount == length(Tqtophoriedge)
        else
            tophoricount = tophoricount + 1;
        end

    elseif q == Tqbothoriedge(bothoricount)

        % Shear Stress Continuity Equation
        iqx = ones(5,1)*q;
        jqx = [q-2*ni q-ni q q+ni q+2*ni]';
        sqx = [G2 -4*G2 3*(G2+G1) -4*G1 G1]';

        iqCx = ones(2,1)*q;
        jqCx = [q-1 q+1]';
        sqCx = [(-G2+G1) (G2-G1)]';
```

```
        % Normal Stress Contniuity Equation
        iqy = ones(5,1)*q;
        jqy = [q-2*ni q-ni q q+ni q+2*ni]';
        sqy = [cc2 -4*cc2 3*(cc2+cc1) -4*cc1 cc1]';

        iqCy = ones(2,1)*q;
        jqCy = [q-1 q+1]';
        sqCy = [(-gamma2+gamma1) (gamma2-gamma1)]';

        % Add terms to vectors
        iXBC(SXBCcount:SXBCcount+4,1) = iqx;
        jXBC(SXBCcount:SXBCcount+4,1) = jqx;
        sXBC(SXBCcount:SXBCcount+4,1) = sqx;

        iYBC(SYBCcount:SYBCcount+4,1) = iqy;
        jYBC(SYBCcount:SYBCcount+4,1) = jqy;
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;

        iCXsparse(CXcount:CXcount+1,1) = iqCx;
        jCXsparse(CXcount:CXcount+1,1) = jqCx;
        sCXsparse(CXcount:CXcount+1,1) = sqCx;

        iCYsparse(CYcount:CYcount+1,1) = iqCy;
        jCYsparse(CYcount:CYcount+1,1) = jqCy;
        sCYsparse(CYcount:CYcount+1,1) = sqCy;

        CXcount = CXcount + 2;
        CYcount = CYcount + 2;

        if bothoricount == length(Tqbothoriedge)
        else
            bothoricount = bothoricount + 1;
        end

    elseif q == Tqleftvertedge(leftvertcount)

        % Normal Stress Continuity Equation
        iqx = ones(5,1)*q;
        jqx = [q-2 q-1 q q+1 q+2]';
        sqx = [cc1 -4*cc1 3*(cc1+cc2) -4*cc2 cc2]';

        iqCx = ones(2,1)*q;
        jqCx = [q-ni q+ni]';
        sqCx = [(-gamma1+gamma2) (gamma1-gamma2)]';
        % Shear Stress Continuity Equation
        iqy = ones(5,1)*q;
        jqy = [q-2 q-1 q q+1 q+2]';
        sqy = [G1 -4*G1 3*(G1+G2) -4*G2 G2]';

        iqCy = ones(2,1)*q;
        jqCy = [q-ni q+ni]';
        sqCy = [(-G1+G2) (G1-G2)]';

        % Add terms to vectors
        iXBC(SXBCcount:SXBCcount+4,1) = iqx;
        jXBC(SXBCcount:SXBCcount+4,1) = jqx;
        sXBC(SXBCcount:SXBCcount+4,1) = sqx;

        iYBC(SYBCcount:SYBCcount+4,1) = iqy;
        jYBC(SYBCcount:SYBCcount+4,1) = jqy;
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;
```

```
        iCXsparse(CXcount:CXcount+1,1) = iqCx;
        jCXsparse(CXcount:CXcount+1,1) = jqCx;
        sCXsparse(CXcount:CXcount+1,1) = sqCx;

        iCYsparse(CYcount:CYcount+1,1) = iqCy;
        jCYsparse(CYcount:CYcount+1,1) = jqCy;
        sCYsparse(CYcount:CYcount+1,1) = sqCy;

        CXcount = CXcount + 2;
        CYcount = CYcount + 2;

        if leftvertcount == length(Tqleftvertedge)
        else
            leftvertcount = leftvertcount + 1;
        end

    elseif q == Tqrighvertedge(righvertcount)

        % Normal Stress Continuity Equation
        iqx = ones(5,1)*q;
        jqx = [q-2 q-1 q q+1 q+2]';
        sqx = [cc2 -4*cc2 3*(cc2+cc1) -4*cc1 cc1]';

        iqCx = ones(2,1)*q;
        jqCx = [q-ni q+ni]';
        sqCx = [(-gamma2+gamma1) (gamma2-gamma1)]';

        % Shear Stress Continuity Equation
        iqy = ones(5,1)*q;
        jqy = [q-2 q-1 q q+1 q+2]';
        sqy = [G2 -4*G2 3*(G2+G1) -4*G1 G1]';

        iqCy = ones(2,1)*q;
        jqCy = [q-ni q+ni]';
        sqCy = [(-G2+G1) (G2-G1)]';

        % Add terms to vectors
        iXBC(SXBCcount:SXBCcount+4,1) = iqx;
        jXBC(SXBCcount:SXBCcount+4,1) = jqx;
        sXBC(SXBCcount:SXBCcount+4,1) = sqx;

        iYBC(SYBCcount:SYBCcount+4,1) = iqy;
        jYBC(SYBCcount:SYBCcount+4,1) = jqy;
        sYBC(SYBCcount:SYBCcount+4,1) = sqy;

        SXBCcount = SXBCcount + 5;
        SYBCcount = SYBCcount + 5;
        iCXsparse(CXcount:CXcount+1,1) = iqCx;
        jCXsparse(CXcount:CXcount+1,1) = jqCx;
        sCXsparse(CXcount:CXcount+1,1) = sqCx;

        iCYsparse(CYcount:CYcount+1,1) = iqCy;
        jCYsparse(CYcount:CYcount+1,1) = jqCy;
        sCYsparse(CYcount:CYcount+1,1) = sqCy;

        CXcount = CXcount + 2;
        CYcount = CYcount + 2;

        if righvertcount == length(Tqrighvertedge)
        else
            righvertcount = righvertcount + 1;
        end

    elseif q == Tqcentre(centrecount)
```

```
        % Apply Finite Differnce Approximations to Centre Carcinoma Nodes -

        % Suxx1/Svyy1 ----------------------------------------------------

        iqSuxx1 = ones(3,1)*q;
        jqSuxx1 = [q-1 q q+1]';
        sqSuxx1 = [(gamma2+2*G2)/nterm/h^2 -2*(gamma2+2*G2)/nterm/h^2
(gamma2+2*G2)/nterm/h^2]';

        jqSvyy1 = [q-ni q q+ni]';

        iSuxx1(Suxx1count:Suxx1count+2,1) = iqSuxx1;
        jSuxx1(Suxx1count:Suxx1count+2,1) = jqSuxx1;
        sSuxx1(Suxx1count:Suxx1count+2,1) = sqSuxx1;

        jSvyy1(Suxx1count:Suxx1count+2,1) = jqSvyy1;

        Suxx1count = Suxx1count + 3;

        % Suyy1/Svxx1 ----------------------------------------------------

        iqSuyy1 = ones(3,1)*q;
        jqSuyy1 = [q-ni q q+ni]';
        sqSuyy1 = [G2/nterm/h^2 -2*G2/nterm/h^2 G2/nterm/h^2]';

        jqSvxx1 = [q-1 q q+1]';

        iSuyy1(Suyy1count:Suyy1count+2,1) = iqSuyy1;
        jSuyy1(Suyy1count:Suyy1count+2,1) = jqSuyy1;
        sSuyy1(Suyy1count:Suyy1count+2,1) = sqSuyy1;

        jSvxx1(Suyy1count:Suyy1count+2,1) = jqSvxx1;

        Suyy1count = Suyy1count + 3;

        % Sufreq ---------------------------------------------------------
        iqSufreq = q;
        jqSufreq = q;
        sqSufreq = rho*omega^2/nterm;

        iSufreq(Sufreqcount,1) = iqSufreq;
        jSufreq(Sufreqcount,1) = jqSufreq;
        sSufreq(Sufreqcount,1) = sqSufreq;

        Sufreqcount = Sufreqcount + 1;

        % Cross-Over Terms------------------------------------------------

        iq = ones(4,1)*q;
        jq = [(q-ni-1) (q-ni+1) (q+ni-1) (q+ni+1)]';
        sq = [Tcuv5 -Tcuv5 -Tcuv5 Tcuv5]';

        iCXsparse(CXcount:CXcount+3,1) = iq;
        jCXsparse(CXcount:CXcount+3,1) = jq;
        sCXsparse(CXcount:CXcount+3,1) = sq;
        iCYsparse(CYcount:CYcount+3,1) = iq;
        jCYsparse(CYcount:CYcount+3,1) = jq;
        sCYsparse(CYcount:CYcount+3,1) = sq;

        CXcount = CXcount + 4;
        CYcount = CYcount + 4;

        if centrecount == length(Tqcentre)
        else
            centrecount = centrecount + 1;
        end

    else
```

```
        % Apply Finite Differnce Approximations to Centre Healthy Nodes ---

        % Suxx1/Svyy1 ------------------------------------------------------

        iqSuxx1 = ones(3,1)*q;
        jqSuxx1 = [q-1 q q+1]';
        sqSuxx1 = [(gamma1+2*G1)/nterm/h^2 -2*(gamma1+2*G1)/nterm/h^2
(gamma1+2*G1)/nterm/h^2]';

        jqSvyy1 = [q-ni q q+ni]';

        iSuxx1(Suxx1count:Suxx1count+2,1) = iqSuxx1;
        jSuxx1(Suxx1count:Suxx1count+2,1) = jqSuxx1;
        sSuxx1(Suxx1count:Suxx1count+2,1) = sqSuxx1;

        jSvyy1(Suxx1count:Suxx1count+2,1) = jqSvyy1;

        Suxx1count = Suxx1count + 3;

        % Suyy1/Svxx1 ------------------------------------------------------

        iqSuyy1 = ones(3,1)*q;
        jqSuyy1 = [q-ni q q+ni]';
        sqSuyy1 = [G1/nterm/h^2 -2*G1/nterm/h^2 G1/nterm/h^2]';

        jqSvxx1 = [q-1 q q+1]';

        iSuyy1(Suyy1count:Suyy1count+2,1) = iqSuyy1;
        jSuyy1(Suyy1count:Suyy1count+2,1) = jqSuyy1;
        sSuyy1(Suyy1count:Suyy1count+2,1) = sqSuyy1;

        jSvxx1(Suyy1count:Suyy1count+2,1) = jqSvxx1;

        Suyy1count = Suyy1count + 3;

        % Sufreq -----------------------------------------------------------
        iqSufreq = q;
        jqSufreq = q;
        sqSufreq = rho*omega^2/nterm;

        iSufreq(Sufreqcount,1) = iqSufreq;
        jSufreq(Sufreqcount,1) = jqSufreq;
        sSufreq(Sufreqcount,1) = sqSufreq;

        Sufreqcount = Sufreqcount + 1;

        % Cross-Over Terms-------------------------------------------------

        iq = ones(4,1)*q;
        jq = [(q-ni-1) (q-ni+1) (q+ni-1) (q+ni+1)]';
        sq = [Hcuv5 -Hcuv5 -Hcuv5 Hcuv5]';

        iCXsparse(CXcount:CXcount+3,1) = iq;
        jCXsparse(CXcount:CXcount+3,1) = jq;
        sCXsparse(CXcount:CXcount+3,1) = sq;
        iCYsparse(CYcount:CYcount+3,1) = iq;
        jCYsparse(CYcount:CYcount+3,1) = jq;
        sCYsparse(CYcount:CYcount+3,1) = sq;

        CXcount = CXcount + 4;
        CYcount = CYcount + 4;

    end
end

iSuxx1 = iSuxx1(find(iSuxx1));
iSuyy1 = iSuyy1(find(iSuyy1));
iSufreq = iSufreq(find(iSufreq));
```

```
iXBC = iXBC(find(iXBC));
iYBC = iYBC(find(iYBC));

jSuxx1 = jSuxx1(find(jSuxx1));
jSuyy1 = jSuyy1(find(jSuyy1));
jSvyy1 = jSvyy1(find(jSvyy1));
jSvxx1 = jSvxx1(find(jSvxx1));
jSufreq = jSufreq(find(iSufreq));
jXBC = jXBC(find(jXBC));
jYBC = jYBC(find(jYBC));

sSuxx1 = sSuxx1(find(sSuxx1));
sSuyy1 = sSuyy1(find(sSuyy1));
sSufreq = sSufreq(find(sSufreq));
sXBC = sXBC(find(iXBC));
sYBC = sYBC(find(iYBC));

% Convert i, j and s Vectors to Sparse Matrix Format ---------------------

Suxx1 = sparse(iSuxx1,jSuxx1,sSuxx1,ni*nj,ni*nj);
Suyy1 = sparse(iSuyy1,jSuyy1,sSuyy1,ni*nj,ni*nj);

Svyy1 = sparse(iSuxx1,jSvyy1,sSuxx1,ni*nj,ni*nj);
Svxx1 = sparse(iSuyy1,jSvxx1,sSuyy1,ni*nj,ni*nj);

Sufreq = sparse(iSufreq,jSufreq,sSufreq,ni*nj,ni*nj);
SXBC = sparse(iXBC,jXBC,sXBC,ni*nj,ni*nj);
SYBC = sparse(iYBC,jYBC,sYBC,ni*nj,ni*nj);

Sx = (Suxx1+Suyy1+Sufreq+SXBC);
Sy = (Svyy1+Svxx1+Sufreq+SYBC);

iRHSsparse = [iRHSsparse; iRHSsparse+totaln];
jRHSsparse = [jRHSsparse; jRHSsparse];
sRHSsparse = [sRHSsparsex; sRHSsparsey];

Srhs = sparse(iRHSsparse,jRHSsparse,sRHSsparse,2*ni*nj,1);

iCXsparse = iCXsparse(find(iCXsparse));
jCXsparse = jCXsparse(find(jCXsparse));
sCXsparse = sCXsparse(find(iCXsparse));

Scx = sparse(iCXsparse,jCXsparse,sCXsparse,ni*nj,ni*nj);

iCYsparse = iCYsparse(find(iCYsparse));
jCYsparse = jCYsparse(find(jCYsparse));
sCYsparse = sCYsparse(find(iCYsparse));

Scy = sparse(iCYsparse,jCYsparse,sCYsparse,ni*nj,ni*nj);
% Clear Unnecessary Variables --------------------------------------------

clear Svxx1 Svyy1 Suyy1 Suxx1 Sufreq SXBC SYBC

clear sCXsparse jCXsparse iCXsparse
clear sCYsparse jCYsparse iCYsparse
clear iSXBC iSYBC jSXBC jSYBC sSXBC sSYBC
clear iSuxx1 jSuxx1 sSuxx1
clear iSuyy1 jSuyy1 sSuyy1
clear iSvxx1 jSvxx1 sSvxx1
clear iSvyy1 jSvyy1 sSvyy1
clear iSufreq jSufreq sSufreq

% Combine Matrices & Solve System of Equations---------------------------

Sfull = [Sx Scx; Scy Sy];

clear Sx Scx Scy Sy
```

```
a0 = Sfull\Srhs;
a1 = full(a0);

% Post-Processing ========================================================

u1 = a1(1:totaln);
v1 = a1(totaln+1:2*totaln);

U0 = [];
V0 = [];

for i = 1:nj
    U0 = [U0; u1(1:ni)'];
    V0 = [V0; v1(1:ni)'];

    u1 = u1((ni+1):length(u1));
    v1 = v1((ni+1):length(v1));
end

xx1 = (0:h:sampi);
yy1 = (0:k:sampj)';
[X,Y] = meshgrid(xx1,yy1);
xcount = 1;
ycount = 1;
[eX1,eY1] = meshgrid(xeqnbound,yeqnbound);

% 2D Dot Plot
figure(1)
clf
hold on
for i = 1:length(xx1)
        plot(X(:,i)+U0(:,i),Y(:,i) + V0(:,i),'b.');
end


% 2D Dot Plot to include Carcinoma Position in RED
% figure(1)
% clf
% hold on
% for i = 1:length(xx1)
%     for j = 1:length(yy1)
%
%         if i == xeqnbound(xcount)
%
%             if j == yeqnbound(ycount)
%                 plot(X(j,i)+U0(j,i),Y(j,i) + V0(j,i),'r.');
%                 ycount = ycount + 1;
%             else
%                 plot(X(j,i)+U0(j,i),Y(j,i) + V0(j,i),'b.');
%             end
%
%         else
%             plot(X(j,i)+U0(j,i),Y(j,i) + V0(j,i),'b.');
%         end
%
%         if ycount > length(yeqnbound)
%             ycount = 1;
%             xcount = xcount + 1;
%         end
%
%         if xcount > length(xeqnbound)
%             xcount = 1;
%         end
%
%     end
% end

% Surface Plots of x & y Displacement Amplitudes
```

```
figure(2)
subplot(1,2,1)
title('Displacement in X direction');
surf(X,Y,U0)
shading interp
subplot(1,2,2)
title('Displacement in Y direction');
surf(X,Y,V0)
shading interp

%-------------------------------------------------------------------------

% save Phantom50Hz-200-2323T U0 V0 omega

tt1 = cputime - tt0

%-------------------------------------------------------------------------
```

A7: 2D Homogeneous Inverse Algorithm – Initial Method

```
%=========================================================================

% 2D HOMOGENEOUS INVERSE ALGORITHM -INITIAL METHOD
% By Samuel J. Houghton
% 2006

% This inverse algorithm determines the Young's Modulus of 2D plain strain
% homogeneous elastic medium using an Origin Base Point Double Integral
% method

%=========================================================================

clear all
close all
clc

load BoxShake100Hz-700-homo

% Define Parameters ======================================================

samp = 0.1;                 % Size of Global Domain
mu = 0.49;                  % Poisson's Ratio
rho = 1020;                 % Density (kg/m3)
sU0 = size(U0);
h_old = samp/(sU0(1)-1);

ip = 200;                   % No. of Integration Points
noeqn = 10;                 % No. of Equations for Each Navier's Equation

h = samp/(ip);              % Step Size
gr = ip;

a1 = mu/((1+mu)*(1-2*mu)) + 1/(1+mu);
b1 = 1/(2+2*mu);
c1 = mu/((1+mu)*(1-2*mu)) + 1/(2+2*mu);

[X Y] = meshgrid(0:h:samp);

% Interpolate Dataset to Required Data Resolution
[XI YI] = meshgrid(0:h_old:samp);

U1 = interp2(XI,YI,U0,X,Y,'cubic');
V1 = interp2(XI,YI,V0,X,Y,'cubic');
```

```
clear XI YI U0 V0

% NOISE GENERATOR ========================================================

% Rearranging Terms for Noise Calculations
u1 = zeros((gr+1)^2,1);
v1 = zeros((gr+1)^2,1);

for i = 1:gr+1

    u1((i-1)*(gr+1)+1:i*(gr+1),1) = U1(i,:)';
    v1((i-1)*(gr+1)+1:i*(gr+1),1) = V1(i,:)';

end

% Calulating Geometric Mean
sortu = sort(abs(u1));
sortv = sort(abs(v1));
percentilenoise = 50;   %Percentile of Absolute Data (Median - 50%)
propu = sortu(ceil(length(u1)*percentilenoise/100));
propv = sortv(ceil(length(v1)*percentilenoise/100));

% Defining Percentage Noise
perror = 20;
f1 = length(u1);
rand1 = rand(f1,1) <= 0.5;
rand2 = rand(f1,1) <= 0.5;

% Geometric Mean (Median) Absolute Noise
uN1 = (-1).^rand1*(perror/100)*propu;
uN2 = rand(f1,1);
uN3 = uN1.*uN2;
u1  = u1 + uN3;
vN1 = (-1).^rand1*(perror/100)*propv;
vN2 = rand(f1,1);
vN3 = vN1.*vN2;
v1  = v1 + vN3;

%=========================================================================

% Rearranging Terms for Inverse Algorithm
nU0 = [];
nV0 = [];

for i = 1:gr+1

    nU0 = [nU0; u1(1:gr+1)'];
    nV0 = [nV0; v1(1:gr+1)'];
    u1 = u1((gr+2):length(u1));
    v1 = v1((gr+2):length(v1));

end

U = nU0;
V = nV0;

clear uN1 uN2 uN3 vN1 vN2 vN3 v1 u1 rand1 rand2 nV0 nU0


% Formulate System of Equations ===========================================

% menu1 = menu('Method of Domain Selection','Origin BP','Centred BP')
% BP stands for 'Base Point'
menu1 = 2;

% Loop for Each Equation ------------------------------------------------

for j = 1:noeqn
```

```
ex0 = j*gr/noeqn + 1;
ey0 = j*gr/noeqn + 1;

x0 = j*h*gr/noeqn;
y0 = j*h*gr/noeqn;

% Determination of Local Domain ---------------------------------------

if menu1 == 1

Ua = U(1:ey0,1:ex0);
Va = V(1:ey0,1:ex0);

elseif menu1 == 2

cen = (gr/2)+1;

Ua = U(cen-(ey0-1)/2:cen+(ey0-1)/2,cen-(ex0-1)/2:cen+(ex0-1)/2);
Va = V(cen-(ey0-1)/2:cen+(ey0-1)/2,cen-(ex0-1)/2:cen+(ex0-1)/2);

end

% CALCULATE U INTEGRAL TERMS ============================================

%  Uxx - Double Integral Term

uF1int1 = trapz(cumtrapz(Ua(:,ex0) - Ua(:,1)))*h^2;
Ux1 = (-3*Ua(:,1) + 4*Ua(:,2) - Ua(:,3))/(2*h);
uF2int1 = trapz(cumtrapz(Ux1))*h^2;

intUxx = uF1int1 - x0*uF2int1;

% Uyy - Double Integral Term

uG1int1 = trapz(cumtrapz(Ua(ey0,:) - Ua(1,:)))*h^2;
Uy1 = (-3*Ua(1,:) + 4*Ua(2,:) - Ua(3,:))/(2*h);
uG2int1 = trapz(cumtrapz(Uy1))*h^2;

intUyy = uG1int1 - y0*uG2int1;

% Vxy - Double Integral Term

uH1int1 = trapz(trapz(Va))*h^2;
uH2int1 = x0*trapz(Va(:,1))*h;
uH3int1 = y0*trapz(Va(1,:))*h;
uH4int1 = x0*y0*Va(1,1);

intVxy = uH1int1 - uH2int1 - uH3int1 + uH4int1;

% U - Double Integral Term

uI1int1 = cumtrapz(Ua)*h;
uI1int2 = trapz(uI1int1)*h;
uI1int3 = cumtrapz(uI1int2)*h;
intU = trapz(uI1int3)*h;

% CALCULATE V INTEGRAL TERMS ============================================

%  Vxx - Double Integral Term

vF1int1 = trapz(cumtrapz(Va(:,ex0) - Va(:,1)))*h^2;
Vx1 = (-3*Va(:,1) + 4*Va(:,2) - Va(:,3))/(2*h);
vF2int1 = trapz(cumtrapz(Vx1))*h^2;

intVxx = vF1int1 - x0*vF2int1;

% Vyy - Double Integral Term
```

```
    vG1int1 = trapz(cumtrapz(Va(ey0,:) - Va(1,:)))*h^2;
    Vy1 = (-3*Va(1,:) + 4*Va(2,:) - Va(3,:))/(2*h);
    vG2int1 = trapz(cumtrapz(Vy1))*h^2;

    intVyy = vG1int1 - y0*vG2int1;

    % Uxy - Double Integral Term

    vH1int1 = trapz(trapz(Ua))*h^2;
    vH2int1 = x0*trapz(Ua(:,1))*h;
    vH3int1 = y0*trapz(Ua(1,:))*h;
    vH4int1 = x0*y0*Ua(1,1);

    intUxy = vH1int1 - vH2int1 - vH3int1 + vH4int1;

    % V - Double Integral Term

    vI1int1 = cumtrapz(Va)*h;
    vI1int2 = trapz(vI1int1)*h;
    vI1int3 = cumtrapz(vI1int2)*h;
    intV = trapz(vI1int3)*h;

    % Save Parameters ========================================================

    ULHS(j,1) = (a1*intUxx + b1*intUyy + c1*intVxy);
    VLHS(j,1) = (b1*intVxx + a1*intVyy + c1*intUxy);
    URHS(j,1) = -rho*omega^2*intU;
    VRHS(j,1) = -rho*omega^2*intV;

end

% SOLVE INTEGRAL EQUATION ================================================

LHS = [ULHS; VLHS];
RHS = [URHS; VRHS];

E1 = lsqlin(LHS,RHS)

E2 = RHS./LHS

percentageerror = abs(30000-E1)/30000*100

% ------------------------------------------------------------------------
```

A8: 2D Homogeneous Inverse Algorithm – Centred Base Point Method

```
%=========================================================================

% 2D HOMOGENEOUS INVERSE ALGORITHM -CENTRED BASE POINT METHOD
% By Samuel J. Houghton
% 2006

% This inverse algorithm determines the Young's Modulus of 2D plain strain
% homogeneous elastic medium using an Centred Base Point Double Integral
% method which uses innovative integration limits to eliminate derivative
% terms.

%=========================================================================

clear all
close all
```

```
clc

load BoxShake100Hz-700-homo

% Interpolate to Desired Data Resolution  ================================

samp = 0.1;                 % Size of Global Domain
mu = 0.49;                  % Poisson's Ratio
rho = 1020;                 % Density (kg/m3)
sU0 = size(U0);
h_old = samp/(sU0(1)-1);

ip = 200;                   % No. of Integration Points
noeqn = 10;                 % No. of Equations for Each Navier's Equation

h = samp/(ip);              % Step Size
gr = samp/h;

a1 = mu/((1+mu)*(1-2*mu)) + 1/(1+mu);
b1 = 1/(2+2*mu);
c1 = mu/((1+mu)*(1-2*mu)) + 1/(2+2*mu);

[X Y] = meshgrid(0:h:samp);

% Interpolate Dataset to Required Data Resolution
[XI YI] = meshgrid(0:h_old:samp);

U1 = interp2(XI,YI,U0,X,Y,'cubic');
V1 = interp2(XI,YI,V0,X,Y,'cubic');

clear XI YI U0 V0

% NOISE GENERATOR ========================================================

% Rearranging Terms for Noise Calculations
u1 = zeros((gr+1)^2,1);
v1 = zeros((gr+1)^2,1);

for i = 1:gr+1

    u1((i-1)*(gr+1)+1:i*(gr+1),1) = U1(i,:)';
    v1((i-1)*(gr+1)+1:i*(gr+1),1) = V1(i,:)';

end

% Calulating Geometric Mean
sortu = sort(abs(u1));
sortv = sort(abs(v1));
percentilenoise = 50;   %Percentile of Absolute Data (Median - 50%)
propu = sortu(ceil(length(u1)*percentilenoise/100));
propv = sortv(ceil(length(v1)*percentilenoise/100));

% Defining Percentage Noise
perror = 0;
f1 = length(u1);
rand1 = rand(f1,1) <= 0.5;
rand2 = rand(f1,1) <= 0.5;

% Geometric Mean (Median) Absolute Noise
uN1 = (-1).^rand1*(perror/100)*propu;
uN2 = rand(f1,1);
uN3 = uN1.*uN2;
u1  = u1 + uN3;
vN1 = (-1).^rand1*(perror/100)*propv;
vN2 = rand(f1,1);
vN3 = vN1.*vN2;
v1  = v1 + vN3;
```

```
%=========================================================================

% Rearranging Terms for Inverse Algorithm
nU0 = [];
nV0 = [];

for i = 1:gr+1

    nU0 = [nU0; u1(1:gr+1)'];
    nV0 = [nV0; v1(1:gr+1)'];

    u1 = u1((gr+2):length(u1));
    v1 = v1((gr+2):length(v1));

end

U = nU0;
V = nV0;

clear uN1 uN2 uN3 vN1 vN2 vN3 v1 u1 rand1 rand2 nV0 nU0


% Formulate System of Equations ===========================================

% Loop for Each Equation ------------------------------------------------

for j = 1:noeqn

    ex0 = j*gr/noeqn + 1;
    ey0 = j*gr/noeqn + 1;

    x0 = j*h*gr/noeqn;
    y0 = j*h*gr/noeqn;

    % Determination of Local Domain ---------------------------------------

    Ua = U(1:ey0,1:ex0);
    Va = V(1:ey0,1:ex0);

    % CALCULATE U INTEGRAL TERMS ==========================================

    %  Uxx - Double Integral Term

    uF1int1 = Ua(:,1) - 2*Ua(:,ceil(ex0/2)) + Ua(:,ex0);

    uF2int1 = flipud(uF1int1(1:ceil(ex0/2)));
    uF2int2 = cumtrapz(uF2int1)*h;

    uF3int1 = uF1int1(ceil(ex0/2):ex0);
    uF3int2 = cumtrapz(uF3int1)*h;

    uF4int1 = uF3int2 + uF2int2;
    intUxx = trapz(uF4int1)*h;

    % Uyy - Double Integral Term

    uG1int1 = Ua(1,:) - 2*Ua(ceil(ey0/2),:) + Ua(ey0,:);

    uG2int1 = fliplr(uG1int1(1:ceil(ey0/2)));
    uG2int2 = cumtrapz(uG2int1)*h;

    uG3int1 = uG1int1(ceil(ey0/2):ey0);
    uG3int2 = cumtrapz(uG3int1)*h;

    uG4int1 = uG3int2 + uG2int2;
    intUyy = trapz(uG4int1)*h;

    % Vxy - Double Integral Term
```

```
uH1int1 = Va(ceil(ey0/2):ey0,ceil(ex0/2):ex0);
uH1int2 = Va(1:ceil(ey0/2),ceil(ex0/2):ex0);
uH1int3 = Va(ceil(ey0/2):ey0,1:ceil(ex0/2));
uH1int4 = Va(1:ceil(ey0/2),1:ceil(ex0/2));

uH2int1 = uH1int1 - uH1int2 - uH1int3 + uH1int4;
uH2int2 = trapz(uH2int1)*h;
intVxy = trapz(uH2int2)*h;

% U - Double Integral Term

uI1int1 = flipud(Ua(1:ceil(ey0/2),:));
uI1int2 = Ua(ceil(ey0/2):ey0,:);

uI2int1 = cumtrapz(uI1int1)*h;
uI2int2 = cumtrapz(uI1int2)*h;

uI3int1 = trapz(uI2int1+uI2int2)*h;

uI4int1 = fliplr(uI3int1(1:ceil(ex0/2)));
uI4int2 = uI3int1(ceil(ex0/2):ex0);

uI5int1 = cumtrapz(uI4int1)*h;
uI5int2 = cumtrapz(uI4int2)*h;

intU = trapz(uI5int1+uI5int2)*h;

% CALCULATE V INTEGRAL TERMS ========================================

%  Vxx - Double Integral Term

vF1int1 = Va(:,1) - 2*Va(:,ceil(ex0/2)) + Va(:,ex0);

vF2int1 = flipud(vF1int1(1:ceil(ex0/2)));
vF2int2 = cumtrapz(vF2int1)*h;

vF3int1 = vF1int1(ceil(ex0/2):ex0);
vF3int2 = cumtrapz(vF3int1)*h;

vF4int1 = vF3int2 + vF2int2;
intVxx = trapz(vF4int1)*h;

% Vyy - Double Integral Term

vG1int1 = Va(1,:) - 2*Va(ceil(ey0/2),:) + Va(ey0,:);

vG2int1 = fliplr(vG1int1(1:ceil(ey0/2)));
vG2int2 = cumtrapz(vG2int1)*h;

vG3int1 = vG1int1(ceil(ey0/2):ey0);
vG3int2 = cumtrapz(vG3int1)*h;

vG4int1 = vG3int2 + vG2int2;
intVyy = trapz(vG4int1)*h;

% Uxy - Double Integral Term

vH1int1 = Ua(ceil(ey0/2):ey0,ceil(ex0/2):ex0);
vH1int2 = Ua(1:ceil(ey0/2),ceil(ex0/2):ex0);
vH1int3 = Ua(ceil(ey0/2):ey0,1:ceil(ex0/2));
vH1int4 = Ua(1:ceil(ey0/2),1:ceil(ex0/2));

vH2int1 = vH1int1 - vH1int2 - vH1int3 + vH1int4;
vH2int2 = trapz(vH2int1)*h;
intUxy = trapz(vH2int2)*h;

% V - Double Integral Term
```

```
    vI1int1 = flipud(Va(1:ceil(ey0/2),:));
    vI1int2 = Va(ceil(ey0/2):ey0,:);

    vI2int1 = cumtrapz(vI1int1)*h;
    vI2int2 = cumtrapz(vI1int2)*h;

    vI3int1 = trapz(vI2int1+vI2int2)*h;

    vI4int1 = fliplr(vI3int1(1:ceil(ex0/2)));
    vI4int2 = vI3int1(ceil(ex0/2):ex0);

    vI5int1 = cumtrapz(vI4int1)*h;
    vI5int2 = cumtrapz(vI4int2)*h;

    intV = trapz(vI5int1+vI5int2)*h;

    % Save Parameters

    ULHS(j,1) = (a1*intUxx + b1*intUyy + c1*intVxy);
    VLHS(j,1) = (b1*intVxx + a1*intVyy + c1*intUxy);
    URHS(j,1) = -rho*omega^2*intU;
    VRHS(j,1) = -rho*omega^2*intV;

end

% SOLVE INTEGRAL EQUATION ================================================

LHS = [ULHS; VLHS];
RHS = [URHS; VRHS];

E1 = lsqlin(LHS,RHS)

E2 = RHS./LHS

percentageerror = abs(30000-E1)/30000*100

% ------------------------------------------------------------------------
```

A9: 2D Non-Homogeneous Inverse Algorithm with Constraint Model

```
%==========================================================================

% 2D NON-HOMOGENOUS INVERSE ALGORITHM with CONSTRAINT MODEL
% By Samuel J. Houghton
% 2006

% This inverse algorithm determines the Young's Modulus of 2D plain strain
% elastic medium with a piecewise constant stiffness distribution.  It
% adopts a 2x2 stencil that is fitted across the global domain to form a
% system of over-determined equations that solve for the Young's Modulus of
% each discretized area element.

% A relatively simple constraint model is also introduced that uses the
% shear stress equations to provide ratios of stiffness between adjacent
% elements, which in turn are used to formulate a representative stiffness
% ratio for each element compared with an arbitrary stiffness.  It works
% for a 1x1 size carcinoma only.

%==========================================================================

clear all
close all
clc
```

```
tt1 = cputime;

load Phantom100Hz-700-6789T.mat      % Load Forward Simulation Data

% Define Parameters ======================================================

samp = 0.1;                 % Size of Global Domain
mu = 0.49;                  % Poisson's Ratio
rho = 1020;                 % Density (kg/m3)
sU0 = size(U0);
h_old = samp/(sU0(1)-1);

ip = 200;                   % No. of Integration Points
noeqn = 10;                 % No. of Equations for Each Navier's Equation

constraintmodel = 0;        % Perform constraint model if equal to 1

nb = 10;                    % nb x nb discrete stiffness elements
h = samp/(ip*nb);           % Step Size
gr = samp/h;

a1 = mu/((1+mu)*(1-2*mu)) + 1/(1+mu);
b1 = 1/(2+2*mu);
c1 = mu/((1+mu)*(1-2*mu)) + 1/(2+2*mu);
d1 = mu/((1+mu)*(1-2*mu));

% Contraint Model Parameters ---------------------------------------------

sizetol = 0.04;
tol2 = 0.2;
cutoff = 3;

% Phantom     50Hz sizetol=0.04, tol2=0.3, cutoff=4
% Edge Effect 50Hz sizetol=0.04, tol2=0.2, cutoff=3
% Box Shake   50Hz sizetol=0.02, tol2=0.2, cutoff=3

% Low Pass Filter Design -------------------------------------------------


% % Low Pass Data Filter
% [bfilt,afilt]=butter(4,0.2,'low');
% U = filtfilt(bfilt,afilt,U);
% V = filtfilt(bfilt,afilt,V);

% Interpolate Dataset to Required Data Resolution ------------------------

[X Y] = meshgrid(0:h:samp);
[XI YI] = meshgrid(0:h_old:samp);

U1 = interp2(XI,YI,U0,X,Y,'cubic');
V1 = interp2(XI,YI,V0,X,Y,'cubic');

clear XI YI U0 V0

% NOISE GENERATOR ========================================================

% Rearranging Terms for Noise Calculations
u1 = zeros((gr+1)^2,1);
v1 = zeros((gr+1)^2,1);

for i = 1:gr+1

    u1((i-1)*(gr+1)+1:i*(gr+1),1) = U1(i,:)';
    v1((i-1)*(gr+1)+1:i*(gr+1),1) = V1(i,:)';

end
```

```
% Calulating Geometric Mean
sortu = sort(abs(u1));
sortv = sort(abs(v1));
percentilenoise = 50;   %Percentile of Absolute Data (Median - 50%)
propu = sortu(ceil(length(u1)*percentilenoise/100));
propv = sortv(ceil(length(v1)*percentilenoise/100));

% Defining Percentage Noise
perror = 0;
f1 = length(u1);
rand1 = rand(f1,1) <= 0.5;
rand2 = rand(f1,1) <= 0.5;

% Geometric Mean (Median) Absolute Noise
uN1 = (-1).^rand1*(perror/100)*propu;
uN2 = rand(f1,1);
uN3 = uN1.*uN2;
u1  = u1 + uN3;
vN1 = (-1).^rand1*(perror/100)*propv;
vN2 = rand(f1,1);
vN3 = vN1.*vN2;
v1  = v1 + vN3;

%=========================================================================

% Rearranging Terms for Inverse Algorithm
nU0 = [];
nV0 = [];

for i = 1:gr+1

    nU0 = [nU0; u1(1:gr+1)'];
    nV0 = [nV0; v1(1:gr+1)'];

    u1 = u1((gr+2):length(u1));
    v1 = v1((gr+2):length(v1));

end

U = nU0;
V = nV0;
clear uN1 uN2 uN3 vN1 vN2 vN3 v1 u1 rand1 rand2 nV0 nU0 U1 V1

% CALCULATE COEFFICIENT TERMS ===========================================

% 2x2 Global Integration Terms ==========================================

% Area Integral Terms ---------------------------------------------------

count = 1;
cu5 = zeros(nb*nb,1);
cv5 = zeros(nb*nb,1);

for j = 1:nb
    for i = 1:nb

        Uij = U((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);
        Vij = V((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);

        cu5ij = (h^2)*trapz(trapz(Uij));
        cv5ij = (h^2)*trapz(trapz(Vij));

        cu5(count,1) = cu5ij;
        cv5(count,1) = cv5ij;

        count = count + 1;
    end
end
```

```
cu5 = cu5*c1;
cv5 = cv5*c1;

% Single Integral Line Terms - Vertical -----------------------------------

count = 1;
cu4 = zeros(nb*(nb-1),1);
cv4 = zeros(nb*(nb-1),1);

for j = 1:nb
    for i = 1:(nb-1)

        Uij = U((j-1)*gr/nb+1:j*gr/nb+1,i*gr/nb+1);
        Vij = V((j-1)*gr/nb+1:j*gr/nb+1,i*gr/nb+1);

        cu4ij = h*trapz(Uij);
        cv4ij = h*trapz(Vij);

        cu4(count,1) = cu4ij;
        cv4(count,1) = cv4ij;

        count = count + 1;

    end
end

cu4 = cu4*d1*samp/nb;
cv4 = cv4*b1*samp/nb;

% Single Integral Line Terms - Horizontal ---------------------------------

count = 1;
cu3 = zeros(nb*(nb-1),1);
cv3 = zeros(nb*(nb-1),1);

for j = 1:(nb-1)
    for i = 1:nb

        Uij = U(j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);
        Vij = V(j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);

        cu3ij = h*trapz(Uij);
        cv3ij = h*trapz(Vij);

        cu3(count,1) = cu3ij;
        cv3(count,1) = cv3ij;

        count = count + 1;

    end
end

cu3 = cu3*b1*samp/nb;
cv3 = cv3*d1*samp/nb;

% Horizontal Double Integral Line Terms - x0-x ----------------------------

count = 1;
Ncu2 = zeros(nb*nb-1,1);
Ncv2 = zeros(nb*nb-1,1);

for j = 1:nb+1
    for i = 1:nb-1

        Uij = U((j-1)*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);
        Vij = V((j-1)*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);
```

```
        Ncu2ij = (h^2)*trapz(cumtrapz(fliplr(Uij)));
        Ncv2ij = (h^2)*trapz(cumtrapz(fliplr(Vij)));

        Ncu2(count,1) = Ncu2ij;
        Ncv2(count,1) = Ncv2ij;

        count = count + 1;
    end
end

Ncu2 = Ncu2*b1;
Ncv2 = Ncv2*a1;

% Horizontal Double Integral Line Terms - x0+x ---------------------------

count = 1;
Pcu2 = zeros(nb*nb-1,1);
Pcv2 = zeros(nb*nb-1,1);

for j = 1:nb+1
    for i = 1:nb-1

        Uij = U((j-1)*gr/nb+1,i*gr/nb+1:(i+1)*gr/nb+1);
        Vij = V((j-1)*gr/nb+1,i*gr/nb+1:(i+1)*gr/nb+1);

        Pcu2ij = (h^2)*trapz(cumtrapz(Uij));
        Pcv2ij = (h^2)*trapz(cumtrapz(Vij));

        Pcu2(count,1) = Pcu2ij;
        Pcv2(count,1) = Pcv2ij;

        count = count + 1;
    end
end

Pcu2 = Pcu2*b1;
Pcv2 = Pcv2*a1;



% Vertical Double Integral Line Terms - y0-y ------------------------------

count = 1;
Ncu1 = zeros(nb*nb-1,1);
Ncv1 = zeros(nb*nb-1,1);

for j = 1:nb-1
    for i = 1:nb+1

        Uij = U((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb+1);
        Vij = V((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb+1);

        Ncu1ij = (h^2)*trapz(cumtrapz(flipud(Uij)));
        Ncv1ij = (h^2)*trapz(cumtrapz(flipud(Vij)));

        Ncu1(count,1) = Ncu1ij;
        Ncv1(count,1) = Ncv1ij;

        count = count + 1;
    end
end

Ncu1 = Ncu1*a1;
Ncv1 = Ncv1*b1;

% Vertical Double Integral Line Terms - y0+y ------------------------------

count = 1;
```

```
Pcu1 = zeros(nb*nb-1,1);
Pcv1 = zeros(nb*nb-1,1);

for j = 1:nb-1
    for i = 1:nb+1

        Uij = U(j*gr/nb+1:(j+1)*gr/nb+1,(i-1)*gr/nb+1);
        Vij = V(j*gr/nb+1:(j+1)*gr/nb+1,(i-1)*gr/nb+1);

        Pcu1ij = (h^2)*trapz(cumtrapz(Uij));
        Pcv1ij = (h^2)*trapz(cumtrapz(Vij));

        Pcu1(count,1) = Pcu1ij;
        Pcv1(count,1) = Pcv1ij;

        count = count + 1;

    end
end

Pcu1 = Pcu1*a1;
Pcv1 = Pcv1*b1;

% RHS Area Integral Terms ------------------------------------------------

count = 1;
cuRHS = zeros((nb-1)^2,1);
cvRHs = zeros((nb-1)^2,1);

ex0 = 2*gr/nb+1;
ey0 = 2*gr/nb+1;

for j = 1:(nb-1)
    for i = 1:(nb-1)

        Uij = U((j-1)*gr/nb+1:(j+1)*gr/nb+1,(i-1)*gr/nb+1:(i+1)*gr/nb+1);
        Vij = V((j-1)*gr/nb+1:(j+1)*gr/nb+1,(i-1)*gr/nb+1:(i+1)*gr/nb+1);

        u1a = flipud(Uij(1:ceil(ey0/2),:));
        u1b = Uij(ceil(ey0/2):ey0,:);
        u2a = cumtrapz(u1a)*h;
        u2b = cumtrapz(u1b)*h;
        u3a = trapz(u2a+u2b)*h;
        u4a = fliplr(u3a(1:ceil(ex0/2)));
        u4b = u3a(ceil(ex0/2):ex0);
        u5a = cumtrapz(u4a)*h;
        u5b = cumtrapz(u4b)*h;
        cuRHSij = trapz(u5a+u5b)*h;

        v1a = flipud(Vij(1:ceil(ey0/2),:));
        v1b = Vij(ceil(ey0/2):ey0,:);
        v2a = cumtrapz(v1a)*h;
        v2b = cumtrapz(v1b)*h;
        v3a = trapz(v2a+v2b)*h;
        v4a = fliplr(v3a(1:ceil(ex0/2)));
        v4b = v3a(ceil(ex0/2):ex0);
        v5a = cumtrapz(v4a)*h;
        v5b = cumtrapz(v4b)*h;
        cvRHSij = trapz(v5a+v5b)*h;

        cuRHS(count,1) = cuRHSij;
        cvRHS(count,1) = cvRHSij;

        count = count + 1;

    end
end
```

```
% Stress Conitinuity Terms ================================================

if constraintmodel == 1

% Vertical Boundary Shear Stress Continuity Terms -------------------------

i = 1;
j = 1;
countnb = 1;
counteqn = 1;
check1 = 1;
SSvertcomp = zeros(nb*(nb-1),1);

xx0 = 0:h:samp/nb;
yy0 = 0:h:samp/nb;

ey0 = gr/nb+1;

for count = 1:nb*(nb-1)

    countSS = 1;
    SSvert1 = [];

    for count2 = 1:gr/nb+1

        Vleft      = V(count2+(j-1)*gr/nb,(i-1)*gr/nb+1:i*gr/nb+1);
        Vleftpoly  = polyfit(xx0,Vleft,3);
        Vxleftij   = 3*Vleftpoly(1)*xx0(end)^2 + 2*Vleftpoly(2)*xx0(end)...
            + Vleftpoly(3);

        Vright     = V(count2+(j-1)*gr/nb,i*gr/nb+1:(i+1)*gr/nb+1);
        Vrightpoly = polyfit(xx0,Vright,3);
        Vxrightij  = Vrightpoly(3);

        if j == 1
            if count2 < gr/nb/2+1
                Uij = U((j-1)*gr/nb+1:j*gr/nb+1,i*gr/nb+1)';
                Upoly = polyfit(yy0,Uij,3);
                Uycentreij = 3*Upoly(1)*yy0(count2)^2+2*Upoly(2)...
                    *yy0(count2)+Upoly(3);
            else
                Uij = U((j-1)*gr/nb-gr/nb/2+count2:j*gr/nb-gr/nb/...
                    2+count2,i*gr/nb+1)';
                Upoly = polyfit(yy0,Uij,3);
                Uycentreij = 3*Upoly(1)*yy0(ceil(ey0/2))^2+2*Upoly(2)...
                    *yy0(ceil(ey0/2))+Upoly(3);
            end
        elseif j == nb
            if count2 > gr/nb/2
                Uij = U((j-1)*gr/nb+1:j*gr/nb+1,i*gr/nb+1)';
                Upoly = polyfit(yy0,Uij,3);
                Uycentreij = 3*Upoly(1)*yy0(count2)^2+2*Upoly(2)...
                    *yy0(count2)+Upoly(3);
            else
                Uij = U((j-1)*gr/nb-gr/nb/2+count2:j*gr/nb-gr/nb/2+...
                    count2,i*gr/nb+1)';
                Upoly = polyfit(yy0,Uij,3);
                Uycentreij = 3*Upoly(1)*yy0(ceil(ey0/2))^2+2*Upoly(2)...
                    *yy0(ceil(ey0/2))+Upoly(3);
            end
        else
            Uij = U((j-1)*gr/nb-gr/nb/2+count2:j*gr/nb-gr/nb/2+count2,i...
                *gr/nb+1)';
            Upoly = polyfit(yy0,Uij,3);
            Uycentreij = 3*Upoly(1)*yy0(ceil(ey0/2))^2 + 2*Upoly(2)...
                *yy0(ceil(ey0/2)) + Upoly(3);
        end
```

```
        SScoeff1 = Uycentreij + Vxleftij;
        SScoeff2 = Uycentreij + Vxrightij;
        SSvert0  = SScoeff1/SScoeff2;

        SSvert0_a(count2,count)  = SScoeff1/SScoeff2;
        Uycentre_a(count2,count) = Uycentreij;
        Vxleft_a(count2,count)   = Vxleftij;
        Vxright_a(count2,count)  = Vxrightij;

        if abs(Vxleftij/-Uycentreij)>1-tol2 & abs(Vxleftij/-Uycentreij)...
                <1+tol2
        elseif abs(Vxrightij/-Uycentreij)>1-tol2 & abs(Vxrightij/...
                -Uycentreij)<1+tol2
        elseif SSvert0<0
        elseif abs(Uycentreij)<sizetol
            if abs(Vxleftij)<sizetol & abs(Vxrightij)<sizetol
            else
                SSvert1(countSS) = SSvert0;
                countSS = countSS + 1;
            end
        else
            SSvert1(countSS) = SSvert0;
            countSS = countSS + 1;
        end
    end

    if length(SSvert1) < cutoff
        SSvert1 = [];
    end

    if isempty(SSvert1) == 0

        SSvertcomp(count,1) = mean(SSvert1);
        vertcheck(count,1:2) = [count length(SSvert1)];

        counteqn = counteqn + 2;
    end

    if count == check1*(nb-1)
        j = j + 1;
        i = 1;
        countnb = countnb + 2;
        check1 = check1 + 1;
    else
        i = i + 1;
        countnb = countnb + 1;
    end
end

sSSvert = size(SSvertcomp)
SSvertRHS = zeros(sSSvert(1),1)*100;

% Horizontal Boundary Shear Stress Continuity Terms ----------------------

i = 1;
j = 1;
countnb = 1;
counteqn = 1;
check1 = 1;
SShoricomp = zeros(nb*(nb-1),1);

xx0 = 0:h:samp/nb;
yy0 = 0:h:samp/nb;

ex0 = gr/nb+1;

for count = 1:nb*(nb-1)
```

```
countSS = 1;
SShori1 = [];

for count2 = 1:gr/nb+1

    Ubot     = U((j-1)*gr/nb+1:j*gr/nb+1,count2+(i-1)*gr/nb)';
    Ubotpoly = polyfit(yy0,Ubot,3);
    Uybotij  = 3*Ubotpoly(1)*yy0(end)^2 + 2*Ubotpoly(2)*yy0(end)...
        + Ubotpoly(3);

    Utop     = U(j*gr/nb+1:(j+1)*gr/nb+1,count2+(i-1)*gr/nb)';
    Utoppoly = polyfit(yy0,Utop,3);
    Uytopij  = Utoppoly(3);

    if i == 1
        if count2 < gr/nb/2+1
            Vij = V(j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);
            Vpoly = polyfit(xx0,Vij,3);
            Vxcentreij = 3*Vpoly(1)*xx0(count2)^2+2*Vpoly(2)...
                *xx0(count2)+Vpoly(3);
        else
            Vij = V(j*gr/nb+1,(i-1)*gr/nb-gr/nb/2+count2:i*gr/nb-gr/...
                nb/2+count2);
            Vpoly = polyfit(xx0,Vij,3);
            Vxcentreij = 3*Vpoly(1)*xx0(ceil(ex0/2))^2+2*Vpoly(2)...
                *xx0(ceil(ex0/2))+Vpoly(3);
        end
    elseif i == nb
        if count2 > gr/nb/2
            Vij = V(j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);
            Vpoly = polyfit(xx0,Vij,3);
            Vxcentreij = 3*Vpoly(1)*xx0(count2)^2+2*Vpoly(2)...
                *xx0(count2)+Vpoly(3);
        else
            Vij = V(j*gr/nb+1,(i-1)*gr/nb-gr/nb/2+count2:i*gr/nb-gr...
                /nb/2+count2);
            Vpoly = polyfit(xx0,Vij,3);
            Vxcentreij = 3*Vpoly(1)*xx0(ceil(ex0/2))^2+2*Vpoly(2)...
                *xx0(ceil(ex0/2))+Vpoly(3);
        end
    else
        Vij = V(j*gr/nb+1,(i-1)*gr/nb-gr/nb/2+count2:i*gr/nb-gr/nb...
            /2+count2);
        Vpoly = polyfit(xx0,Vij,3);
        Vxcentreij = 3*Vpoly(1)*xx0(ceil(ex0/2))^2+2*Vpoly(2)...
            *xx0(ceil(ex0/2))+Vpoly(3);
    end

    SScoeff1 = Vxcentreij + Uybotij;
    SScoeff2 = Vxcentreij + Uytopij;
    SShori0  = SScoeff1/SScoeff2;

    SShori0_a(count2,count)  = SShori0;
    Vxcentre_a(count2,count) = Vxcentreij;
    Uybot_a(count2,count)    = Uybotij;
    Uytop_a(count2,count)    = Uytopij;

    if abs(Uybotij/-Vxcentreij)>1-tol2 & abs(Uybotij/-Vxcentreij)...
            <1+tol2
    elseif abs(Uytopij/-Vxcentreij)>1-tol2 & abs(Uytopij...
            /-Vxcentreij)<1+tol2
    elseif SShori0<0
    elseif abs(Vxcentreij)<sizetol
        if abs(Uybotij)<sizetol & abs(Uytopij)<sizetol
        else
            SShori1(countSS) = SShori0;
            countSS = countSS + 1;
        end
```

```
        else
            SShori1(countSS) = SShori0;
            countSS = countSS + 1;
        end
    end

    if length(SShori1) < cutoff
        SShori1 = [];
    end

    if isempty(SShori1) == 0

        SShoricomp(count,1) = mean(SShori1);
        horicheck(count,1:2) = [count length(SShori1)];

        counteqn = counteqn + 2;
    end

    if count == check1*nb
        j = j + 1;
        i = 1;
        countnb = countnb + 1;
        check1 = check1 + 1;
    else
        i = i + 1;
        countnb = countnb + 1;
    end
end

sSShori = size(SShoricomp)
SShoriRHS = zeros(sSShori(1),1);

% Creating Equality Matrix Based Upon Averaging Techniques ----------------

avgval = zeros(1,100);

countbot = 1;
counttop = 1;
countleft  = 1;
countright = 1;
counteqn = 1;
countvert = 0;
checkvert = 1;

indexbot = 2:nb-1;
indextop = (2:nb-1)+(nb-1)*nb;
indexleft = (nb+1):nb:(nb-2)*nb+1;
indexright = 2*nb:nb:(nb-1)*nb;

for i = 1:nb^2
    if i == 1

        q0 = [SSvertcomp(1) SShoricomp(1)];
        findq0 = find(q0);

        if any(q0) == 1
            q1 = [1/SSvertcomp(1) 1/SShoricomp(1)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

    elseif i == indexbot(countbot)

        q0 = [SSvertcomp(i-1) SSvertcomp(i) SShoricomp(i)];
        findq0 = find(q0);
```

```
        if any(q0) == 1
            q1 = [SSvertcomp(i-1) 1/SSvertcomp(i) 1/SShoricomp(i)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

        if countbot == length(indexbot)
        else
            countbot = countbot + 1;
        end

    elseif i == nb

        q0 = [SSvertcomp(nb-1) SShoricomp(nb)];
        findq0 = find(q0);

        if any(q0) == 1
            q1 = [SSvertcomp(nb-1) 1/SShoricomp(nb)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

    elseif i == indexleft(countleft)

        q0 = [SSvertcomp(i-countleft) SShoricomp(i-nb) SShoricomp(i)];
        findq0 = find(q0);

        if any(q0) == 1
            q1 = [1/SSvertcomp(i-countleft) SShoricomp(i-nb)...
                1/SShoricomp(i)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

        if countleft == length(indexleft)
        else
            countleft = countleft + 1;
        end

    elseif i == indexright(countright)

        q0 = [SSvertcomp(i-countright-1) SShoricomp(i-nb) SShoricomp(i)];
        findq0 = find(q0);

        if any(q0) == 1
            q1 = [SSvertcomp(i-countright-1) SShoricomp(i-nb)...
                1/SShoricomp(i)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

        if countright == length(indexright)
        else
            countright = countright + 1;
        end

    elseif i == nb*(nb-1)+1

        q0 = [SSvertcomp((nb-1)^2+1) SShoricomp((nb-1)^2)];
        findq0 = find(q0);
```

```
        if any(q0) == 1
            q1 = [1/SSvertcomp((nb-1)^2+1) SShoricomp((nb-1)^2)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

    elseif i == indextop(counttop)

        q0 = [SSvertcomp(i-nb) SSvertcomp(i-nb+1) SShoricomp(i-nb)];
        findq0 = find(q0);

        if any(q0) == 1
            q1 = [SSvertcomp(i-nb) 1/SSvertcomp(i-nb+1) SShoricomp(i-nb)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

        if counttop == length(indextop)
        else
            counttop = counttop + 1;
        end

    elseif i == nb^2

        q0 = [SSvertcomp((nb-1)*nb) SShoricomp((nb-1)*nb)];
        findq0 = find(q0);

        if any(q0) == 1
            q1 = [SSvertcomp((nb-1)*nb) SShoricomp((nb-1)*nb)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

    else

        q0 = [SSvertcomp(i-countvert-1) SSvertcomp(i-countvert)...
            SShoricomp(i-nb) SShoricomp(i)];
        findq0 = find(q0);

        if any(q0) == 1
            q1 = [SSvertcomp(i-countvert-1) 1/SSvertcomp(i-countvert)...
                SShoricomp(i-nb) 1/SShoricomp(i)];
            q1 = q1(findq0);
            avgval(counteqn,i)   = 1;
            avgvalcomp(counteqn) = mean(q1);
            counteqn = counteqn + 1;
        end

        if i == check1*(nb-1)
            countvert = countvert+1;
            check1 = check1 + 1;
        end

    end
end

avgval = [avgval -avgvalcomp'];

savgval = size(avgval);
avgvalRHS = zeros(savgval(1),1);
```

```
end

% 1cm x 1cm Local Integration Terms ====================================

% Area Integral Terms ---------------------------------------------------

count = 1;
lcu3 = zeros(4*nb*nb,1);
lcv3 = zeros(4*nb*nb,1);

for j = 1:2*nb
    for i = 1:2*nb

        Uij = U((j-1)*gr/nb/2+1:j*gr/nb/2+1,(i-1)*gr/nb/2+1:i*gr/nb/2+1);
        Vij = V((j-1)*gr/nb/2+1:j*gr/nb/2+1,(i-1)*gr/nb/2+1:i*gr/nb/2+1);

        lcu3ij = (h^2)*trapz(trapz(Uij));
        lcv3ij = (h^2)*trapz(trapz(Vij));

        lcu3(count,1) = lcu3ij;
        lcv3(count,1) = lcv3ij;

        count = count + 1;
    end
end

lcu3 = lcu3*c1;
lcv3 = lcv3*c1;

% Horizontal Double Integral Terms --------------------------------------

count = 1;
lcu2 = zeros(2*nb^2+nb,1);
lcv2 = zeros(2*nb^2+nb,1);

ex0 = gr/nb+1;
ey0 = gr/nb+1;

for j = 1:2*nb+1
    for i = 1:nb
        Uij = U((j-1)*gr/nb/2+1,(i-1)*gr/nb+1:i*gr/nb+1);
        Vij = V((j-1)*gr/nb/2+1,(i-1)*gr/nb+1:i*gr/nb+1);

        u1a = fliplr(Uij(1:ceil(ex0/2)));
        u1b = Uij(ceil(ex0/2):ex0);
        u2a = cumtrapz(u1a)*h;
        u2b = cumtrapz(u1b)*h;
        lcu2ij = trapz(u2a+u2b)*h;

        v1a = fliplr(Vij(1:ceil(ex0/2)));
        v1b = Vij(ceil(ex0/2):ex0);
        v2a = cumtrapz(v1a)*h;
        v2b = cumtrapz(v1b)*h;
        lcv2ij = trapz(v2a+v2b)*h;

        lcu2(count,1) = lcu2ij;
        lcv2(count,1) = lcv2ij;

        count = count + 1;
    end
end

lcu2 = lcu2*b1;
lcv2 = lcv2*a1;

% Vertical Double Integral Terms ----------------------------------------

count = 1;
```

```
lcu1 = zeros(2*nb^2+nb,1);
lcv1 = zeros(2*nb^2+nb,1);

for j = 1:nb
    for i = 1:2*nb+1

        Uij = U((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb/2+1);
        Vij = V((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb/2+1);

        u1a = flipud(Uij(1:ceil(ey0/2)));
        u1b = Uij(ceil(ey0/2):ey0);
        u2a = cumtrapz(u1a)*h;
        u2b = cumtrapz(u1b)*h;
        lcu1ij = trapz(u2a+u2b)*h;

        v1a = flipud(Vij(1:ceil(ey0/2)));
        v1b = Vij(ceil(ey0/2):ey0);
        v2a = cumtrapz(v1a)*h;
        v2b = cumtrapz(v1b)*h;
        lcv1ij = trapz(v2a+v2b)*h;

        lcu1(count,1) = lcu1ij;
        lcv1(count,1) = lcv1ij;

        count = count + 1;
    end
end

lcu1 = lcu1*a1;
lcv1 = lcv1*b1;

% RHS Area Intergal Terms ------------------------------------------------

count = 1;
lcuRHS = zeros(nb^2,1);
lcvRHs = zeros(nb^2,1);

for j = 1:nb
    for i = 1:nb

        Uij = U((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);
        Vij = V((j-1)*gr/nb+1:j*gr/nb+1,(i-1)*gr/nb+1:i*gr/nb+1);

        u1a = flipud(Uij(1:ceil(ey0/2),:));
        u1b = Uij(ceil(ey0/2):ey0,:);
        u2a = cumtrapz(u1a)*h;
        u2b = cumtrapz(u1b)*h;
        u3a = trapz(u2a+u2b)*h;
        u4a = fliplr(u3a(1:ceil(ex0/2)));
        u4b = u3a(ceil(ex0/2):ex0);
        u5a = cumtrapz(u4a)*h;
        u5b = cumtrapz(u4b)*h;
        lcuRHSij = trapz(u5a+u5b)*h;

        v1a = flipud(Vij(1:ceil(ey0/2),:));
        v1b = Vij(ceil(ey0/2):ey0,:);
        v2a = cumtrapz(v1a)*h;
        v2b = cumtrapz(v1b)*h;
        v3a = trapz(v2a+v2b)*h;
        v4a = fliplr(v3a(1:ceil(ex0/2)));
        v4b = v3a(ceil(ex0/2):ex0);
        v5a = cumtrapz(v4a)*h;
        v5b = cumtrapz(v4b)*h;
        lcvRHSij = trapz(v5a+v5b)*h;

        lcuRHS(count,1) = lcuRHSij;
        lcvRHS(count,1) = lcvRHSij;
```

```
        count = count + 1;

    end
end

% Clear Unnecessary Variables ---------------------------------------------

clear cu5ij cv5ij cu4ij cv4ij cu3ij cv3ij Ncu2ij Ncv2ij Pcu2ij Pcv2ij ...
    Ncu1ij Ncv1ij Pcu1ij Pcv1ij cuRHSij cvRHSij

clear lcu3ij lcv3ij lcu2ij lcv2ij lcu1ij lcv1ij lcuRHSij lcvRHSij

clear u1a u1b u2a u2b u3a u4a u4b u5a u5b
clear v1a v1b v2a v2b v3a v4a v4b v5a v5b

% DEFINE SYSTEM OF EQUATIONS ===============================================

% 1x1 Squares ===============================================================

Au1 = zeros(nb^2,nb^2);
Av1 = zeros(nb^2,nb^2);

count1 = 1;
count2 = 1;
count3 = 1;
check1 = 1;

for i = 1:nb^2

    if i == check1*nb

        Ucoeff1 = lcu1(count1)-2*lcu1(count1+1)+lcu1(count1+2)...
            + lcu2(count2)-2*lcu2(count2+nb)+lcu2(count2+2*nb)...
            + lcv3(2*count3-1)-lcv3(2*count3)-lcv3(2*(count3+nb)-1)...
            +lcv3(2*(count3+nb));
        Vcoeff1 = lcv1(count1)-2*lcv1(count1+1)+lcv1(count1+2)...
            + lcv2(count2)-2*lcv2(count2+nb)+lcv2(count2+2*nb)...
            + lcu3(2*count3-1)-lcu3(2*count3)-lcu3(2*(count3+nb)-1)...
            +lcu3(2*(count3+nb));

        Au1(i,i) = Ucoeff1;
        Av1(i,i) = Vcoeff1;

        count1 = count1 + 3;
        count2 = count2 + nb + 1;
        count3 = count3 + nb + 1;
        check1 = check1 + 1;

    else

        Ucoeff1 = lcu1(count1)-2*lcu1(count1+1)+lcu1(count1+2)...
            + lcu2(count2)-2*lcu2(count2+nb)+lcu2(count2+2*nb)...
            + lcv3(2*count3-1)-lcv3(2*count3)-lcv3(2*(count3+nb)-1)...
            +lcv3(2*(count3+nb));
        Vcoeff1 = lcv1(count1)-2*lcv1(count1+1)+lcv1(count1+2)...
            + lcv2(count2)-2*lcv2(count2+nb)+lcv2(count2+2*nb)...
            + lcu3(2*count3-1)-lcu3(2*count3)-lcu3(2*(count3+nb)-1)...
            +lcu3(2*(count3+nb));

        Au1(i,i) = Ucoeff1;
        Av1(i,i) = Vcoeff1;

        count1 = count1 + 2;
        count2 = count2 + 1;
        count3 = count3 + 1;

    end
end
```

```
% 2x2 Squares ============================================================

% Fundamental Equations ---------------------------------------------------

Au2 = zeros((nb-1)^2,nb^2);
Av2 = zeros((nb-1)^2,nb^2);

count2 = 1;
count3 = 1;
check1 = 1;

for i = 1:(nb-1)^2

    if i == check1*(nb - 1)

        Ucoeff1 = Ncu1(count3)   - Ncu1(count3+1) + Ncu2(i)        ...
            - Ncu2(nb-1+i) - cv3(count2)   - cv4(i)       + cv5(count2);
        Ucoeff2 = Ncu1(count3+2) - Ncu1(count3+1) + Pcu2(i)        ...
            - Pcu2(nb-1+i) + cv3(count2+1) + cv4(i)       - cv5(count2+1);
        Ucoeff3 = Pcu1(count3)   - Pcu1(count3+1) + Ncu2(2*nb-2+i)...
            - Ncu2(nb-1+i) + cv3(count2)   + cv4(nb-1+i) - cv5(count2+nb);
        Ucoeff4 = Pcu1(count3+2) - Pcu1(count3+1) + Pcu2(2*nb-2+i)...
            - Pcu2(nb-1+i) - cv3(count2+1) - cv4(nb-1+i) + cv5(count2+nb+1);

        Vcoeff1 = Ncv1(count3)   - Ncv1(count3+1) + Ncv2(i)        ...
            - Ncv2(nb-1+i) - cu3(count2)   - cu4(i)       + cu5(count2);
        Vcoeff2 = Ncv1(count3+2) - Ncv1(count3+1) + Pcv2(i)        ...
            - Pcv2(nb-1+i) + cu3(count2+1) + cu4(i)       - cu5(count2+1);
        Vcoeff3 = Pcv1(count3)   - Pcv1(count3+1) + Ncv2(2*nb-2+i)...
            - Ncv2(nb-1+i) + cu3(count2)   + cu4(nb-1+i) - cu5(count2+nb);
        Vcoeff4 = Pcv1(count3+2) - Pcv1(count3+1) + Pcv2(2*nb-2+i)...
            - Pcv2(nb-1+i) - cu3(count2+1) - cu4(nb-1+i) + cu5(count2+nb+1);

        Au2(i,count2:count2+1) = [Ucoeff1 Ucoeff2];
        Au2(i,count2+nb:count2+nb+1) = [Ucoeff3 Ucoeff4];
        Av2(i,count2:count2+1) = [Vcoeff1 Vcoeff2];
        Av2(i,count2+nb:count2+nb+1) = [Vcoeff3 Vcoeff4];

        count2 = count2 + 2;
        count3 = count3 + 3;

        check1 = check1 + 1;

    else

        Ucoeff1 = Ncu1(count3)   - Ncu1(count3+1) + Ncu2(i)        ...
            - Ncu2(nb-1+i) - cv3(count2)   - cv4(i)       + cv5(count2);
        Ucoeff2 = Ncu1(count3+2) - Ncu1(count3+1) + Pcu2(i)        ...
            - Pcu2(nb-1+i) + cv3(count2+1) + cv4(i)       - cv5(count2+1);
        Ucoeff3 = Pcu1(count3)   - Pcu1(count3+1) + Ncu2(2*nb-2+i)...
        - Ncu2(nb-1+i) + cv3(count2)   + cv4(nb-1+i) - cv5(count2+nb);
        Ucoeff4 = Pcu1(count3+2) - Pcu1(count3+1) + Pcu2(2*nb-2+i)...
        - Pcu2(nb-1+i) - cv3(count2+1) - cv4(nb-1+i) + cv5(count2+nb+1);

        Vcoeff1 = Ncv1(count3)   - Ncv1(count3+1) + Ncv2(i)        ...
            - Ncv2(nb-1+i) - cu3(count2)   - cu4(i)       + cu5(count2);
        Vcoeff2 = Ncv1(count3+2) - Ncv1(count3+1) + Pcv2(i)        ...
            - Pcv2(nb-1+i) + cu3(count2+1) + cu4(i)       - cu5(count2+1);
        Vcoeff3 = Pcv1(count3)   - Pcv1(count3+1) + Ncv2(2*nb-2+i)...
            - Ncv2(nb-1+i) + cu3(count2)   + cu4(nb-1+i) - cu5(count2+nb);
        Vcoeff4 = Pcv1(count3+2) - Pcv1(count3+1) + Pcv2(2*nb-2+i)...
            - Pcv2(nb-1+i) - cu3(count2+1) - cu4(nb-1+i) + cu5(count2+nb+1);

        Au2(i,count2:count2+1) = [Ucoeff1 Ucoeff2];
        Au2(i,count2+nb:count2+nb+1) = [Ucoeff3 Ucoeff4];
        Av2(i,count2:count2+1) = [Vcoeff1 Vcoeff2];
        Av2(i,count2+nb:count2+nb+1) = [Vcoeff3 Vcoeff4];
```

```
        count2 = count2 + 1;
        count3 = count3 + 1;

    end
end

% Solve System of Equations Using Linear Least Squares ====================

rhsscale = (-rho*omega^2);
x = ones(nb^2,1)*30000;

% 1x1 ---------------------------------------------------------------------

A1 = [Au1;Av1];
A1 = A1./rhsscale;
bb1 = [lcuRHS;lcvRHS];
qq1 = sum(A1,2);
% E1 = bb1./qq1;
E1 = lsqlin(A1,bb1);

check3 = norm(A1*x-bb1);

% 2x2 ---------------------------------------------------------------------

A2 = [Au1;Av1;Au2;Av2];
A2 = A2./(-rho*omega^2);
bb2 = [lcuRHS;lcvRHS;cuRHS;cvRHS];
lb = zeros(nb^2,1);

E2 = lsqlin(A2,bb2,[],[],[],[],[],[]);

% Post-Processing ==========================================================

% Comparative Line Plot ---------------------------------------------------

figure(1)
plot(E2)
hold on

% check2 = A2*x-bb2;
qq2 = sum(A2,2);

xx=0:h:0.02;
qq = 1:(nb-1)^2;

if constraintmodel == 1

CC = [Au2./rhsscale;Av2./rhsscale];
dd = [cuRHS;cvRHS];

sCC = size(CC);
CC = [CC zeros(sCC(1),1)];

% Aineq = [SSvert;SShori];
% bineq = [SSvertRHS;SShoriRHS];

lbc = zeros(nb^2+1,1);

E2SS = lsqlin(CC,dd,[],[],avgval,avgvalRHS,[],[]);

figure(1)
plot(E2SS,'r')
hold off

end

% Colour Patch Plot Setup -------------------------------------------------
```

```
e1 = E2;
E1 = [];

for i = 1:nb
    E1 = [E1; e1(1:nb)'];
    e1 = e1((nb+1):length(e1));
end

pfactor = 50;

for j = 1:nb
    for i = 1:nb
        E1long((1:pfactor)+pfactor*(j-1),(1:pfactor)+pfactor*(i-1))...
            = E1(j,i);
    end
end

xi = samp/nb/pfactor:samp/nb/pfactor:samp;
yi = samp/nb/pfactor:samp/nb/pfactor:samp;

[X Y] = meshgrid(xi,yi);

Cmin = 2.0*10^4;
Cmax = 3.3*10^5;
load MyColourmaps

% Plot Colour Patch - 2x2 ------------------------------------------------
figure(2)
surf(X,Y,0*E1long,E1long,'CDataMapping','scaled')
title('Unconstrained Model')
colormap(lump)
caxis([Cmin Cmax])
view([0 90])
shading flat
colorbar

% Plot Colour Patch - 2x2 Model with Constraint Model --------------------
if constraintmodel == 1

e0 = E2SS;
E0 = [];

for i = 1:nb
    E0 = [E0; e0(1:nb)'];
    e0 = e0((nb+1):length(e0));
end

pfactor = 50;

for j = 1:nb
    for i = 1:nb
        E0long((1:pfactor)+pfactor*(j-1),(1:pfactor)+pfactor*(i-1))...
            = E0(j,i);
    end
end

figure(3)
surf(X,Y,0*E0long,E0long,'CDataMapping','scaled')
title('Constrained Model')
colormap(lump)
caxis([Cmin Cmax])
view([0 90])
shading flat
colorbar

end
```

```
tt2 = cputime-tt1

%-------------------------------------------------------------------------
```