Finite Element Analysis in Inverse Problems of Biomechanical Imaging

Abstract

Finite element method (FEM) is widely used in engineering computation. This report describes the use of FEM in inverse problems of Biomechanical Imaging (BMI). A computationally intensive but robust-to-noise iterative approach is used in this analysis and FEM is used in every iteration. In this project, experimental data from collaborators was preprocessed and visualized to a format that can be fed to the inverse problem solver named NLACE. The theory of NLACE and the preprocessing for finite element computation and were interpreted and the output of NLACE was postprocessed, visualized as a plot of shear modulus reconstruction and it was brought up at the end.

Introduction

Generally speaking, cancerous tissue tends to become stiff due to changes in the microscopic level. Microscopic organization changes often leads to macroscopic changes in tissue properties, which can be easier to quantify. Biomechanical imaging (BMI) is used to image the mechanical properties of tissue and infer tissue pathology. BMI generally includes the following steps:

1) subject the tissue to a deformation,
2) measure the displacement field in the entire domain and
3) compute the shear modulus by solving the inverse problem.

Inverse problem is the counterpart of forward problem. Instead of obtaining deformations with known material properties, the inverse problem deals with the opposite way, i.e. using known deformations to find the material properties. Inverse problems are generally ill-posed in nature, therefore the results obtained from inverse solver are very sensitive to the noise in the input data. In order to have better solutions, iterative finite element analysis is performed and effective minimization method is used to obtain a stable solution.

Theoretical Foundation of NLACE

NLACE uses L-BFGS-B optimization algorithm to solve the inverse problem. Gradient-based algorithm, such as L-BFGS-B, generally costs the most when evaluating the gradient vector. It typically needs N+1 solves to find the gradient vector in an N parameters inverse problem. Now with the adjoint equations, NLACE only needs 2 solves to calculate the gradient vector, which improves the efficiency of the optimization algorithm significantly. It is noted that FEM is used to perform the two solves and eventually find the gradient vector. The gradient vector is then carried to the objective function to see if it is minimized enough. If not, the process goes back to the two finite element solves and evaluate a

new gradient vector. This continues iteratively until the objective function is minimized under the constraint of the equations of equilibrium for a linear, incompressible material. Following will review the derivations in a brief and simplified manner and more detailed discussion and examples can be found in [1].

In the inverse solving process, the difference between the measured and predicted displacement fields is minimized. The measured and predicted displacement fields are assumed to satisfy the equation below, which describing the equilibrium of an isotropic, linear-elastic solid undergoing small, quasi-static deformations.

Strong formulation

The strong form of the forward problem is given as follows: given

(i) the Lame parameters, μ (x), in the entire spatial domain Ω,
(ii) assume homogeneous Dirichlet boundary conditions denoted by Γ_g.

find the displacement denoted by u(x) that satisfy the equations of equilibrium for a linear, incompressible material given by

$$-\nabla(\mu\nabla u) = f, \quad \text{in } \Omega \tag{1}$$

$$u = g, \quad \text{on } \Gamma_g \tag{2}$$

In the equations above, f is the forcing function and $\Omega \subset \mathcal{R}^s$ represents the interior of a body whose boundary is$\partial\Omega = \overline{\Gamma_g}$. Eq. (1) represents the balance of linear momentum within an elastic solid. Eq. (2) specifies the displacement on the boundaryΓ_g.

Weak formulation

An equivalent weak formulation of (1) – (2) is as follows: find $u \in \mathcal{S}$ such that

$$a(w, u; \mu) = (w, f) \quad \forall w \in \mathcal{V}, \tag{3}$$

where $a(\cdot,\cdot;\mu)$ and $(\cdot,\cdot)$ are typical bilinear forms as described in [2]. The weighting function space $\mathcal{V}$ and the trial solution space $\mathcal{S}$ are defined as

$$\mathcal{V} = \{w | w \in H^1, w = 0 \text{ on } \Gamma_g\} \tag{4}$$

$$\mathcal{S} = \{u | u \in H^1, u = g \text{ on } \Gamma_g\} \tag{5}$$

Note that u is a function of μ. The equivalency of strong form and weak form is nontrivial and is not discussed here. Details can be referred to [2].

Galerkin's approximation

Finite-dimensional subspaces are introduced here to solve Eq. (3) numerically. A superscript h denotes the finite-dimensional counterpart of a continuous quantity. In order to obtain the Galerkin approximation, function $u^h \in \mathcal{S}^h$ is constructed by $u^h = v^h + g^h$ and to each member $v^h \in \mathcal{V}^h$. This leads to the Galerkin approximation: find$v^h \in \mathcal{V}^h$, such that

$$a(w^h, v^h; \mu^h) = (w^h, f) - a(w^h, g^h; \mu^h) \quad \forall w^h \in \mathcal{V}^h, \tag{6}$$

In this case, the function g^h is constructed from finite element shape functions with nodes on the Dirichlet boundary conditions. Thus Eq. (6) is solved with finite element method. u^h, w^h and μ^h are represented as a linear combination of continuous, piece-wise bilinear finite-element shape functions. Isoparametric element is used here to simplify programming process, meaning the same shape function is used for u^h andμ^h. Below take u^h as an example:

$$u^h = \sum_{a=1}^{n} u_a N_a(x), \tag{7}$$

where n is the number of nodes in the finite element model and N_a is the typical shape function.

Inverse elasticity problem

The inverse problem is defined as follows: given boundary data g, and the measured displacement field $u^m \text{ in } \Omega$, find the distribution of shear modulus μ(x) and a displacement field u, such that Eq. (1) – (2) hold, and $u^m - u = 0$. This problem may be posed as a minimization problem as follows: find μ(x) such that the functional

$$\pi(\mu) = \frac{1}{2}||u - u^m||_\Omega^2 + \frac{\alpha}{2}||\mu||_b^2 \tag{8}$$

is minimized subject to the constraint that u satisfies the forward elasticity problem. $||\cdot||_\Omega^2$ represents the L_2 norm in Ω, and $||\cdot||_b^2$ represents an appropriate norm chosen for regularizing the solution. The parameter α is the Tikhonov parameter and is chosen according to the theory of residues due to Morozov ([3], [4]). Generally gradient-based algorithms, such as steepest descent, BFGS and DFP, are used to solve this type of problem [5]. The most expensive step in these algorithms is usually to evaluate the gradient vector. NLACE uses the adjoint equations to reduce the computation cost. Following will introduce the efficient algorithm used in NLACE computing the gradient vector.

Gradient evaluation

First comes the Lagrangian L,

$$L(u, w, \mu) = \pi(\mu) + a(w, u; \mu) - (w, f) \tag{9}$$

where w plays the role of a Lagrangian multiplier. The differential of L is given by

$$\delta L = D_u L \cdot \delta u + D_w L \cdot \delta w + D_\mu L \cdot \delta \mu \tag{10}$$

Here we set $D_w L \cdot \delta w = 0, \forall w \in \mathcal{V}$, yielding:

$$a(\delta w, u; \mu) = (\delta w, f), \quad \forall \delta w \in \mathcal{V} \tag{11}$$

which implies u satisfies the original weak form of the forward elasticity problem. Thus, we have

$$\delta L = \delta \pi \tag{12}$$

If we further set $D_u L \cdot \delta u = 0, \forall u \in \mathcal{V}$, again yielding:

$$a(w, \delta u; \mu) = -(\delta u, u - u^m), \quad \forall \delta u \in \mathcal{V} \tag{13}$$

With above assumptions of w and u, we have

$$\delta \pi = \delta L = D_\mu a(w, u; \mu) \cdot \delta \mu + \alpha(\mu, \delta \mu)_b \tag{14}$$

If we use Galerkin approximation on Eq. (14), yielding

$$\delta \pi = D_\mu a\left(w^h, u^h; \mu^h\right) \cdot \delta \mu^h + \alpha(\mu^h, \delta \mu^h)_b \tag{15}$$

where $u^h = v^h + g^h$, which satisfies

$$a\left(\delta w^h, v^h; \mu^h\right) = \left(\delta w^h, f\right) - a\left(\delta w^h, g^h; \mu^h\right) \quad \forall \delta w^h \in \mathcal{V}^h, \tag{16}$$

$$a\left(\delta u^h, w^h; \mu^h\right) = -\left(\delta u^h, (u^h - u^m)\right) \quad \forall \delta u^h \in \mathcal{V}^h, \tag{17}$$

Thus, we can simply write

$$\delta \pi = G \cdot \delta \mu \tag{18}$$

where G is the gradient vector to be solved and is given by

$$G_a = a\left(w^h, u^h; N_a\right) + \alpha(\mu^h, N_a)_b \tag{19}$$

Now it is easy to use the above relations to construct an algorithm as follows:

(i) Solve (16) to evaluate u^h.
(ii) Solve (17) to evaluate w^h.
(iii) Use u^h and w^h in (19) to compute G.

The above algorithm involves two finite element solves in (i) and (ii), which is in contrast to the N+1 solves in straightforward calculation of the gradient vector. Following is an example solved by NLACE.

Preprocess

In the experiment, quasi-static force was applied uniformly upon the top face of a phantom cube. The cube was placed on a plane, thus the boundary condition can be considered as that the bottom face was constrained in y direction. Excluding top and bottom faces, another face out of the other four faces was selected for data collection. On that face, displacements of 16 X 40 = 640 nodes were initially recorded. Moreover, data of certain nodes along the two vertical edges were unavailable due to ineffective transducer collection. The displacements of those nodes were recorded as void, showing as NaN in MatLab. After removing these voids, the data becomes 12 X 40 = 480 in total. By applying quasi-static load, 197 loading steps were performed in total, while in this case, only the first 10 loading steps were used, which is good enough for NLACE to produce meaningful results. The experiment is illustrated in Figure 1 and the raw data collected by our project collaborators are displayed in Table 1.

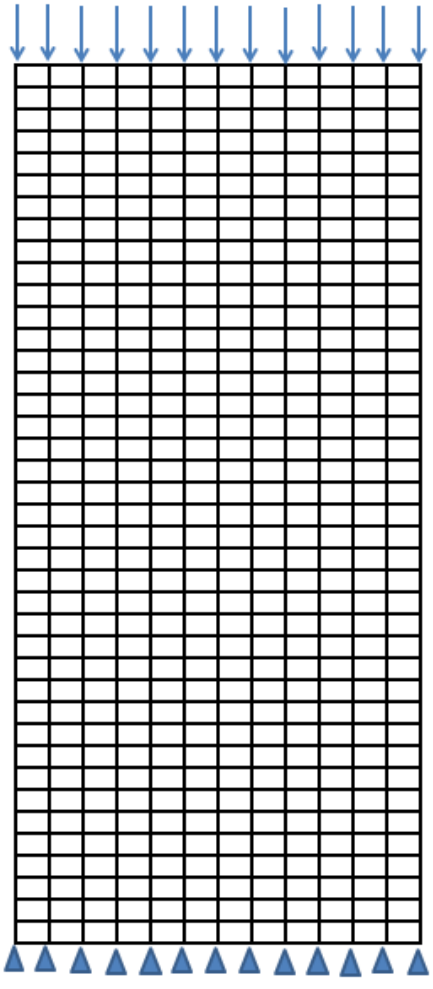

Figure 1 Experiment illustration

Table 1 Displacement Field

Loading Step	X [mm]	Y [mm]	Z [mm]	X [mm]	Y [mm]	Z [mm]	…
1	-22	-60	-0.97641	-19	-60	-0.72549	…
2	-21.9897	-59.8867	-0.85234	-18.9904	-59.8995	-0.66299	…
3	-21.953	-59.8531	-0.92127	-18.9546	-59.8629	-0.7255	…
4	-21.9565	-59.7616	-0.56325	-18.9635	-59.7771	-0.41827	…
5	-21.906	-59.7129	-0.81539	-18.9166	-59.7198	-0.66172	…
6	-21.8822	-59.6069	-0.78251	-18.9001	-59.626	-0.6738	…
…	…	…	…	…	…	…	…

Although transducer gave the displacements along three directions x, y and z, the data along z-direction is observed to be just oscillating and does not change much. Thus it is legitimate to ignore it and reduce the problem from 3D to 2D, which was intentionally done during experiment for the ease of calculation. In this model, only translation in x and y direction is considered thus the maximum DOF per node is 2.

Those data was fed into a MatLab subroutine *create_input_direct.m*, generating an input file *data2D.in* for the solver NLACE. In this file, nodal coordinates, element connectivity and boundary conditions were defined and are partly displayed in the following due to limited space.

Nodal Coordinates

As mentioned before, after removing invalid nodes along both edges, there are 12 X 40 = 480 nodes. The nodes are numbered from bottom to top and from left to right. Table 2 displays Row1 and Row40.

Table 2 Nodal Coordinates

Row#	Node#	X Coord.	Y Coord.	Row#	Node#	X Coord.	Y Coord.
Row1	1	-16	-60	…	…	…	…
	2	-13	-60	Row40	469	-16	57
	3	-10	-60		470	-13	57
	4	-7	-60		471	-10	57
	5	-4	-60		472	-7	57
	6	-1	-60		473	-4	57
	7	2	-60		474	-1	57
	8	5	-60		475	2	57
	9	8	-60		476	5	57
	10	11	-60		477	8	57
	11	14	-60		478	11	57
	12	17	-60		479	14	57
…	…	…	…		480	17	57

Element Connectivity

Element connectivity was illustrated in Figure 2 and Table 3. Element was numbered in the convention of counterclockwise. There are (12-1) X (40-1) = 429 elements in total.

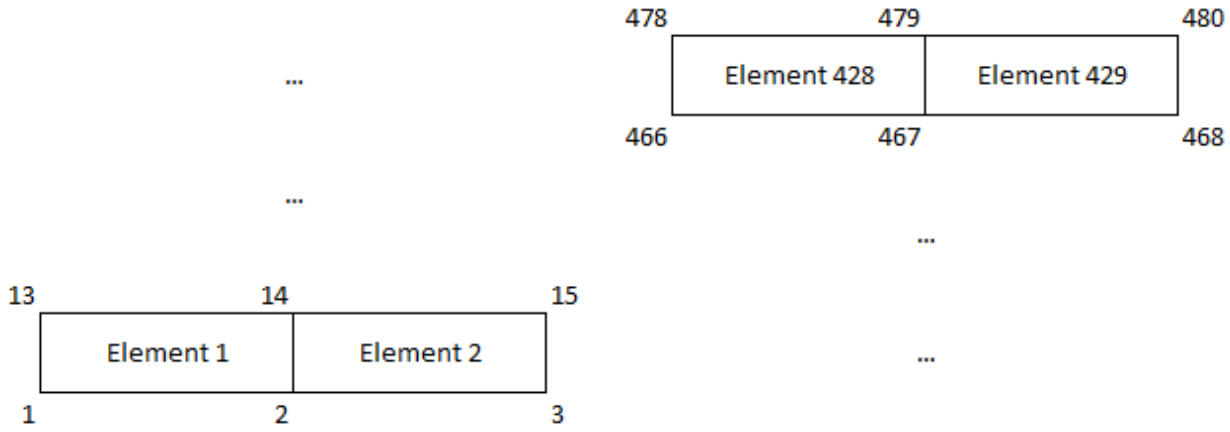

Figure 2 Element 1, 2, 428 and 429

Table 3 Element Topology

Row#	Element#	Node#				Row#	Element#	Node#			
	1	1	2	14	13	…	…	…	…	…	…
	2	2	3	15	14		419	457	458	470	469
	3	3	4	16	15		420	458	459	471	470
	4	4	5	17	16		421	459	460	472	471
	5	5	6	18	17		422	460	461	473	472
Row1	6	6	7	19	18		423	461	462	474	473
	7	7	8	20	19	Row40	424	462	463	475	474
	8	8	9	21	20		425	463	464	476	475
	9	9	10	22	21		426	464	465	477	476
	10	10	11	23	22		427	465	466	478	477
	11	11	12	24	23		428	466	467	479	478
…	…	…	…	…	…		429	467	468	480	479

Boundary Conditions

Boundary conditions is illustrated in Figure 1 and displayed in Table 4. Dirichlet condition = 1 means active and corresponding displacement would be enforced. When Dirichlet condition is inactive (Dirichlet Cond. = 0), homogeneous Neumann condition would be used instead of enforcing a displacement with Dirichlet condition.

Table 4 Boundary Conditions

Node#	Dirichlet Cond.	Lateral Disp.	Dirichlet Cond.	Axial Disp.	Node#	Dirichlet Cond.	Lateral Disp.	Dirichlet Cond.	Axial Disp.
1	1	0.1964	1	0.6652	469	0	0.6768	1	3.4345
2	1	0.1448	1	0.6194	470	0	0.6736	1	3.4095
3	1	0.0964	1	0.5841	471	0	0.6478	1	3.3688
4	1	0.0573	1	0.5684	472	0	0.5632	1	3.3314
5	1	0.0220	1	0.5608	473	0	0.4494	1	3.2990
6	1	-0.0173	1	0.5588	474	0	0.3295	1	3.2690
7	1	-0.0580	1	0.5526	475	0	0.2297	1	3.2525
8	1	-0.0965	1	0.5420	476	0	0.1615	1	3.2410
9	1	-0.1351	1	0.5367	477	0	0.1134	1	3.2280
10	1	-0.1771	1	0.5288	478	0	0.0746	1	3.2094
11	1	-0.2207	1	0.5272	479	0	0.0352	1	3.1983
12	1	-0.2688	1	0.5327	480	0	-0.0032	1	3.1686

All the other information in the input file is related to the optimization algorithm in the inverse analysis and they are purposely omitted here because of the limited space and the fact that the focus of this report is about FEM.

Postprocess

The output from NLACE can be viewed by Paraview as the following figures show. Figure 3 shows the displacements and strains in x and y direction generated roughly by MatLab. The purpose of generating these images with MatLab first is to provide a first and intuitive impression and to make sure that the data fed to NLACE is solvable. Figure 4 is the shear modulus reconstruction generated by NLACE and postprocessed by Paraview, displaying the single inclusion inside the phantom cube. This case took loading step 1 as the undeformed stage (reference frame). In order to make the inclusion more obvious and the noisy spot on the edge less obvious, cases with different reference frames, with averaging frames, and with corresponding scaled regularization factors will be investigated in the future trial cases.

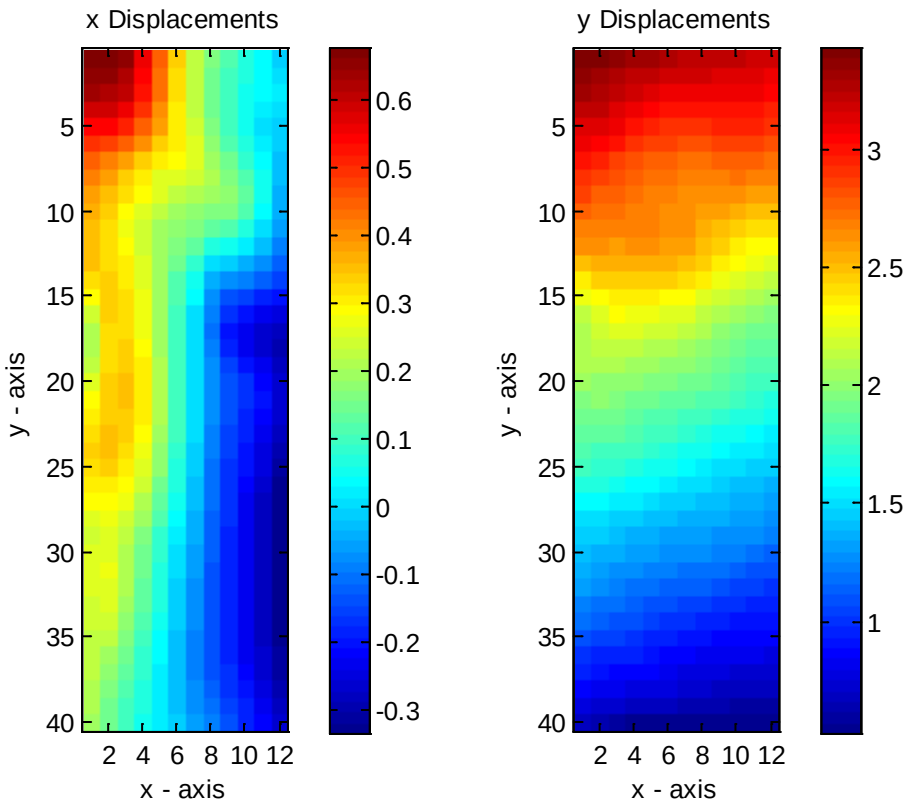

Figure 3 Measured displacements along x and y directions

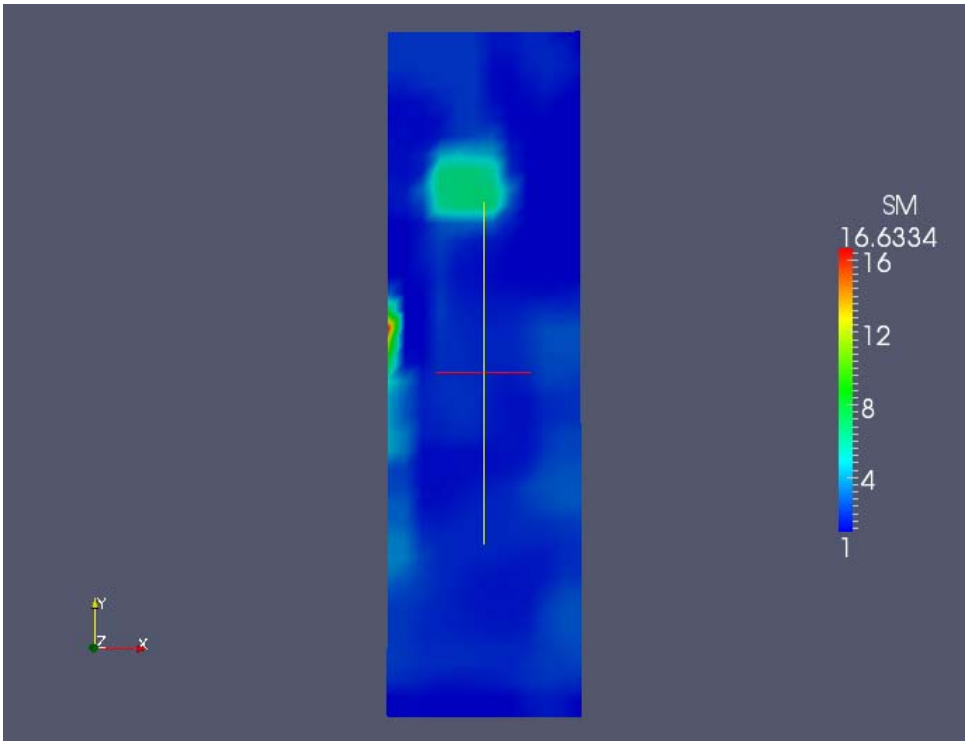

Figure 4 Shear modulus reconstruction

Summary

This report investigates the finite element analysis in inverse problems of Biomechanical Imaging. First the theory of NLACE was briefly introduced. The core algorithm in NLACE was reviewed in this part. Then experimental data of a single inclusion phantom was fed into the MatLab subroutine to generate the input file including parameters for the optimization algorithm and finite element formulation, such as nodal coordinates, element connectivity and boundary conditions. Data preprocess of FEM was reviewed in this part. Shear modulus reconstruction was finally plotted with the output of NLACE visualized by Paraview.

References

[1] Oberai, A. A., Gokhale, N. H., and Feijoo, G. R. 2003. Solution of inverse problems in elasticity imaging using the adjoint method. *Inverse Problems*, 19, 297–313.

[2] Hughes, T. J. R. 2000. The Finite Element Method—Linear Static and Dynamic Finite Element Analysis (Mineola, NY: Dover).

[3] Colton, D. and Kress, R. 1998. Inverse Acoustic and Electromagnetic Scattering Theory, 2nd edition (Berlin: Springer).

[4] Isakov, V. 1998. Inverse Problems for Partial Differential Equations, 1st edition (New York: Springer).

[5] Gill, P. E., Murray, W. and Wright, M. H. 2000. Practical Optimization, 12th edition (London: Academic).