Discretization of the Hellinger-Reissner Variational Form of Linear Elasticity Equations

Kevin A. Sweet
Portland State University
Discretization of the Hellinger-Reissner Variational Form of Linear Elasticity Equations

Kevin Sweet
sweetkev@pdx.edu
August 23, 2019

Abstract
This paper addresses the derivation of the Hellinger-Reissner Variational Form from the strong form of a system of linear elasticity equations that are used in relation to geological phenomena. The problem is discretized using finite element discretization as described in [1]. This allowed the creation of a program that was used to run tests on various domains. The resultant displacement vectors for tested domains are shown at the end of the paper.

1 Background
1.1 Motivation
Linear elasticity is used to describe small, reversible deformations to a structure. This particular set of linear elasticity equations has an application to geological phenomena, such as landslides, but I was unable to meet with the geologist associated with the project to learn the exact nature of the equations’ application.

1.2 Previous Work
The stabilzied mixed finite element method used to discretize the linear elasticity equations are introduced and described in [1]. The work that is done to get from the strong form of the equations to the discretized form of the Hellinger-Reissner variational form is also described in [2], which is a paper that was completed with this project last year. This paper will additionally include information about computing the linear system that is developed.

2 Methodology
2.1 Deriving the Hellinger-Reissner Variational Form
The work in the two following sections is described in [1] and was also done during the previous summer as a part of this project, detailed in [2].

2.1.1 Linear Elasticity Equations
Let $\Omega \subset \mathbb{R}^3$ be a domain with the boundary $\partial \Omega$. The strong form of the problem is to find a displacement vector $\vec{u} : \Omega \to \mathbb{R}^3$ and symmetric stress tensor $\sigma : \Omega \to \mathbb{R}^{3 \times 3}_{\text{sym}}$ such that:

$$
\begin{cases}
A\sigma = \epsilon(\vec{u}) & \text{in } \Omega \\
-(\nabla \cdot \sigma) = f & \text{in } \Omega
\end{cases}
$$

(1)
where $\mathbb{R}^{3\times 3}_{\text{sym}}$ are the symmetric $3 \times 3$ matrices, the function $f : \Omega \to \mathbb{R}^3$ is given, and the symmetric gradient $\epsilon(\vec{u}) : \Omega \to \mathbb{R}^{3\times 3}_{\text{sym}}$ and compliance tensor $A\sigma : \Omega \to \mathbb{R}^{3\times 3}_{\text{sym}}$ are defined by

\[
\epsilon(\vec{u}) = \frac{1}{2} (\nabla \vec{u} + (\nabla \vec{u})^T) \tag{2}
\]

\[
A\sigma = \frac{1}{2\mu} \left( \sigma - \frac{\lambda}{3\lambda + 2\mu} \text{tr}(\sigma) I \right) \tag{3}
\]

Next, the equations from (1) are multiplied by test functions $\vec{v} : \Omega \to \mathbb{R}^3$ and $\tau : \Omega \to \mathbb{R}^{3\times 3}_{\text{sym}}$ and integrated over the domain, resulting in the following:

\[
\begin{cases}
\int_{\Omega} A\sigma : \tau \, dx = \int_{\Omega} \epsilon(\vec{u}) : \tau \, dx \\
- \int_{\Omega} (\nabla \cdot \sigma) \cdot \vec{v} \, dx = \int_{\Omega} f \cdot \vec{v} \, dx
\end{cases} \tag{4}
\]

The double dot product of two matrices $A, B \in \mathbb{R}^{3\times 3}$ is defined by

\[
A : B = \sum_{i,j=1}^{3} a_{ij} b_{ji} \tag{5}
\]

which is equivalent to $\sum_{i=1}^{3} a_i b_i$ with symmetric matrices, such as those in this problem.

### 2.1.2 Integration by Parts

Using integration by parts, the following equation is derived:

\[
\int_{\partial \Omega} \vec{v} \cdot (\tau \vec{n}) \, ds = \int_{\Omega} \nabla \vec{v} : \tau \, dx + \int_{\Omega} \vec{v} \cdot (\nabla \cdot \tau) \, dx \tag{6}
\]

where $\vec{v} : \Omega \to \mathbb{R}^3$ and $\tau : \Omega \to \mathbb{R}^{3\times 3}_{\text{sym}}$. When the right hand side of the first equation in (4) is applied, it results in the equation,

\[
\int_{\partial \Omega} \vec{u} \cdot (\tau \vec{n}) \, ds = \int_{\Omega} \epsilon(\vec{u}) : \tau \, dx + \int_{\Omega} \vec{u} \cdot (\nabla \cdot \tau) \, dx . \tag{7}
\]

### 2.1.3 The Hellinger-Reissner Variational Form

Applying the equation above to (4) and rearranging terms gives, with the assumption $\vec{u} = 0$ on $\partial \Omega$,

\[
\begin{align*}
\int_{\Omega} A\sigma : \tau \, dx + \int_{\Omega} (\nabla \cdot \tau) \cdot \vec{u} \, dx &= 0 \\
- \int_{\Omega} (\nabla \cdot \sigma) \cdot \vec{v} \, dx &= \int_{\Omega} f \cdot \vec{v} \, dx
\end{align*} \tag{8}
\]

This is equivalent to,

\[
\begin{align*}
a(\sigma, \tau) + b(\tau, \vec{u}) &= 0 \\
-b(\sigma, \vec{v}) &= \int_{\Omega} f \cdot \vec{v} \, dx
\end{align*}
\]

where

\[
\begin{align*}
a(\sigma, \tau) &= \int_{\Omega} A\sigma : \tau \, dx \quad , \quad b(\tau, \vec{v}) = \int_{\Omega} (\nabla \cdot \tau) \cdot \vec{v} \, dx
\end{align*}
\]
The problem is now to find \( \vec{u} \in L^2(\Omega; \mathbb{R}^3) \) and \( \sigma \in H(\text{div}, \Omega; \mathbb{R}^{3\times3}_{\text{sym}}) \) such that

\[
\begin{align*}
\left\{ \begin{array}{l}
 a(\sigma, \tau) + b(\tau, \vec{u}) = 0 \\
 -b(\sigma, \vec{v}) = \int_{\Omega} f \cdot \vec{v} \, dx
\end{array} \right.
\end{align*}
\]

for all \( \vec{v} \in L^2(\Omega; \mathbb{R}^3) \) and \( \tau \in H(\text{div}, \Omega; \mathbb{R}^{3\times3}_{\text{sym}}) \), where

\[
L^2(\Omega; \mathbb{R}^3) = \left\{ \vec{v} : \Omega \to \mathbb{R}^3 : \int_{\Omega} \vec{v} \cdot \vec{v} \, dx < \infty \right\},
\]

\[
H(\text{div}, \Omega; \mathbb{R}^{3\times3}_{\text{sym}}) = \left\{ \tau : \Omega \to \mathbb{R}^{3\times3}_{\text{sym}} : \int_{\Omega} \tau : \tau \, dx < \infty \right\}.
\]

2.2 Discretization of the Hellinger-Reissner Variational Form

2.2.1 Stabilized Mixed Finite Element Method

The method described in [1] will now be applied. Let \( \Omega \subset \mathbb{R}^3 \) be a polyhedron partitioned into a tetrahedral mesh \( T = \{ K \} \), with \( F = \{ F \} \) as the set of all triangular faces in the mesh. The goal is to find \((\sigma_h, \vec{u}_h) \in \Sigma_h \times V_h\) such that

\[
\begin{align*}
\left\{ \begin{array}{l}
 a(\sigma_h, \tau_h) + b(\tau_h, \vec{u}_h) = 0 \\
 -b(\sigma_h, \vec{v}_h) + c(\vec{u}_h, \vec{v}_h) = \int_{\Omega} f \cdot \vec{v}_h \, dx
\end{array} \right.
\end{align*}
\]

for all \( \tau_h \in \Sigma_h \) and \( \vec{v}_h \in V_h \). The finite dimensional spaces

\[
\Sigma_h \subset H(\text{div}, \Omega; \mathbb{R}^{3\times3}_{\text{sym}}), \quad V_h \subset L^2(\Omega; \mathbb{R}^3)
\]

are defined in the next subsection, and the jump stabilization term is defined as

\[
c(\vec{u}_h, \vec{v}_h) = \sum_{F \in F} h_F \int_F [\vec{u}_h] : [\vec{v}_h]
\]

where \( h_F \) is the longest edge on \( F \). For a face on the boundary of \( \Omega \),

\[
[\vec{w}] := \frac{1}{2} \left( \vec{n} \vec{w}_F^T + \vec{n} \vec{w}_F^T \right)
\]

where \( \vec{n} \) is the outward normal vector to the face. For an edge that is on the interior of the mesh,

\[
[\vec{w}] := \frac{1}{2} \left( \vec{n}_+ (\vec{n}_+)^T + \vec{n}_+ (\vec{n}_+)^T + \vec{n}_- (\vec{n}_-)^T + \vec{n}_- (\vec{n}_-)^T \right)
\]

where \( K_+ \) and \( K_- \) are adjacent tetrahedra, and \( \vec{n}_+ \) and \( \vec{n}_- \) are their respective outward normal vectors on the face \( F \).

2.2.2 Defining \( \Sigma_h \) and \( V_h \)

The finite dimensional spaces for \( \vec{u}_h \) and \( \sigma_h \) are defined by,

\[
\Sigma_h = \{ \tau \in C(\Omega; \mathbb{R}^{3\times3}_{\text{sym}}) : \tau|_K \in \mathbb{P}_1(K; \mathbb{R}^{3\times3}_{\text{sym}}) \text{ for all } K \in T \}
\]

\[
V_h = \{ \vec{v} : \Omega \to \mathbb{R}^3 : \vec{v}|_K \in \mathbb{R}^3 \text{ for all } K \in T \}.
\]

The basis used for \( V_h \) is

\[
\{ \phi_{K,j} : K \in T \text{ and } 1 \leq j \leq 3 \} \text{ where, } \phi_{K,j}(x) = \begin{cases} e_j, & x \in K \\ 0, & x \notin K \end{cases}
\]
which has the dimension $3NT$, where $NT$ is the number of tetrahedra in the mesh. This basis will be numbered $\phi_i: 1 \leq i \leq 3NT$, where $i = 3(K - 1) + j$. The two spaces from the definition of $\Sigma_h$ are given by

$$C(\Omega; \mathbb{R}^{3\times3}_{\text{sym}}) = \{ \alpha : \Omega \to \mathbb{R}^{3\times3}_{\text{sym}} : \alpha_{ij} \in C(\Omega; \mathbb{R}) \} ,$$

$$P_1(K; \mathbb{R}^{3\times3}_{\text{sym}}) = \{ \alpha : K \to \mathbb{R}^{3\times3}_{\text{sym}} : \alpha_{ij} \in P_1(K; \mathbb{R}) \} .$$

A basis of the space $\mathbb{R}^{3\times3}_{\text{sym}}$ is given by

$$S_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, S_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, S_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

$$S_4 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, S_5 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, S_6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} .$$

Each $\alpha \in \Sigma_h$ is given by a linear function $\ell_i$ that is associated with a single vertex $z_j$, that is defined by its values at adjacent vertices. This is given by

$$\ell_i(z_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} , \quad 1 \leq i, j \leq 4 .$$

A formula that gives $\ell_i$ is

$$\ell_i(x) = 1 - \frac{(x - z_i) \cdot \vec{n}_i}{h_i}$$

where $h_i$ is the perpendicular distance from $z_i$ to its opposite face and $\vec{n}_i$ is the face’s outward normal. From this, a basis for $P_1(K; \mathbb{R}^{3\times3}_{\text{sym}})$ is

$$\ell_i S_j : 1 \leq i \leq 4, 1 \leq j \leq 6$$

with a dimension of 24. $\sigma_h(z)$ is given for each vertex $z$ in the mesh, so $\dim(\Sigma_h) = 6NV$ where $NV$ is the number of vertices in the mesh. This basis will be denoted $\psi_k: 1 \leq k \leq 6NT$ where $k = 6(z - 1) + j$.

### 2.2.3 The Linear System

Now, a linear system can be developed to solve for the stress tensor $\sigma_h$ and the displacement vector $\vec{u}$. The equations from (10) hold for all $\tau_h \in \Sigma_h$ and $\vec{u}_h \in V_h$ if and only if,

$$\begin{cases} a(\sigma_h, \psi_i) + b(\psi_i, \vec{u}_h) = 0, & 1 \leq i \leq 6NV \\ -b(\sigma_h, \phi_k) + c(\vec{u}_h, \phi_k) = \int_{\Omega} f \cdot \phi_d dx, & 1 \leq k \leq 3NT \end{cases} .$$

Because

$$\vec{u}_h = \sum_{j=1}^{3NT} y_j \phi_j \quad \sigma_h = \sum_{j=1}^{6NV} x_j \psi_j$$

for some coefficient vectors $x \in \mathbb{R}^{3NT}, y \in \mathbb{R}^{6NV}$, (14) is equivalent to: find $\begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^{6NV+3NT}$ such that,

$$\begin{pmatrix} A & B^T \\ -B & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ f \end{pmatrix}$$

(16)
where \( A \in \mathbb{R}^{6_NV \times 6_NV}, B \in \mathbb{R}^{3_NT \times 6_NV}, C \in \mathbb{R}^{3_NT \times 3_NT} \) are given by
\[
\begin{align*}
a_{ij} &= a(\psi_j, \psi_i), \quad 1 \leq i, j \leq 6_NV \\
b_{ij} &= b(\psi_j, \phi_i), \quad 1 \leq i \leq 3_NT \quad 1 \leq j \leq 6_NV \\
c_{ij} &= c(\phi_j, \phi_i), \quad 1 \leq i, j \leq 3_NT
\end{align*}
\] (17)
and \( f \in \mathbb{R}^{3_NT} \) is given by
\[
f_k = \int_{\Omega} f \cdot \phi_k \, dx, \quad 1 \leq k \leq 3_NT.
\]

2.3 Computing the Linear System

Here begins work presented in neither [1] nor [2]. The global matrix \( G \), defined as
\[
\begin{pmatrix}
A & B^T \\
-B & C
\end{pmatrix}
\]
is a sparse matrix. To compute the solution to the linear system, \( G \) is put into a compressed sparse row (CSR) format. The matrix is split into three arrays, the values array, which contains every non-zero entry in row-major order, the columns array, which contains the column number for each non-zero entry, and the row index, which gives the cumulative number of non-zero entries row by row. Their dimensions are as follows:
\[
\begin{align*}
\text{dim(values)} &= \text{nnz} \\
\text{dim(columns)} &= \text{nnz} \\
\text{dim(row index)} &= \text{6_NT} + 3_NT + 1
\end{align*}
\]
where \( \text{nnz} \) is the number of non-zero entries in \( G \). To allocate these arrays, matrices \( A, B, \) and \( C \) were analyzed separately. The sum of the number of non-zeroes for each matrix, which is dependent on \( NV \) and \( NT \), gives \( \text{nnz} \) (i.e. \( \text{nnz} = \text{nnz}_A + 2 \text{nnz}_B + \text{nnz}_C \)). To fill the global values array, local \( A_K \) and \( B_K \) matrices were computed over each tetrahedron \( K \), whose values were then added to the global values matrix.

2.3.1 Matrix \( A \)

Each matrix \( A_K \) has four \( \ell \) functions, one for each vertex, associated with it. Every vertex combines with each of six components of the \( \mathbb{R}^{3 \times 3}_{\text{sym}} \) basis, \( S \). Hence, \( A_K \) is a \( 24 \times 24 \) (or \( 4(6) \times 4(6) \)) matrix. This covers every non-zero value of the global matrix, because a function \( \psi_i \) can only have non-zero results when interacting with a function \( \psi_j \) associated with the same vertex or an adjacent vertex, which looping over each tetrahedron covers. For a single entry in \( A_K \),
\[
a_{6(i-1)+m, 6(j-1)+n} = a(\ell_i S_m, \ell_j S_n) = \int_K A(\ell_i S_m) : (\ell_j S_n) \, dx \quad 1 \leq i, j \leq 4 \quad 1 \leq m, n \leq 6.
\]
The rightmost expression simplifies to
\[
\frac{1}{2\mu} \left( S_m : S_n - \frac{\lambda}{3\lambda + 2\mu} \text{tr}(S_m) I : S_n \right) \int_K \ell_i \ell_j \, dx
\]
by using the equation from (3). The two parts of the of the expression result in the following values:
\[
\begin{align*}
\frac{1}{2\mu} \left( S_m : S_n - \frac{\lambda}{3\lambda + 2\mu} \text{tr}(S_m) I : S_n \right) &= \begin{cases} \\
\frac{\lambda + \mu}{\mu(3\lambda + 2\mu)}, & 1 \leq m, n \leq 3 \text{ and } m = n \\
\frac{\lambda}{2\mu(3\lambda + 2\mu)}, & 1 \leq m, n \leq 3 \text{ and } m \neq n \\
\frac{1}{n}, & 4 \leq m, n \leq 6 \text{ and } m = n \\
0, & \text{else}
\end{cases} \\
\int_K \ell_i \ell_j &= \begin{cases} \\
\frac{|K_i|}{|K|}, & i = j \\
\frac{|K_j|}{|K|}, & i \neq j
\end{cases}
\end{align*}
\]
where \( |K| \) is the volume of the tetrahedron. Using this, a Kronecker Product can be created between the two parts of the expression to give the local matrix \( A_K \), as seen below:

\[
A_K = \frac{|K|}{20} \begin{pmatrix}
2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 \\
1 & 1 & 2 & 1 \\
1 & 1 & 1 & 2
\end{pmatrix} 
\otimes
\begin{pmatrix}
\frac{\lambda + \mu}{\mu(3\lambda + 2\mu)} & \frac{-2\mu}{\lambda(3\lambda + 2\mu)} & \frac{-2\mu}{\mu(3\lambda + 2\mu)} & 0 & 0 & 0 \\
\frac{-2\mu}{\mu(3\lambda + 2\mu)} & \frac{\lambda}{\mu(3\lambda + 2\mu)} & \frac{-2\mu}{\mu(3\lambda + 2\mu)} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{\mu} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{\mu}
\end{pmatrix}
\]

This gives the values to be input into matrix \( A \) from each tetrahedron. The structure of the matrix is also revealed from this endeavor, allowing the total number of non-zero entries to be calculated. For the first three basis functions of \( \mathbb{R}^{3\times3}_{\text{sym}} \), associated with a vertex \( z \) on the mesh, the number of non-zero entries in \( A \) is equal to \( 3v \) and for \( S_4-6 \) the number of non-zero entries is \( v \), where \( v \) is the number of interacting vertices with \( z \), including itself. Thus,

\[
nnz_A = \sum_{z=1}^{NV} 12v_z.
\]

### 2.3.2 Matrix \( B \)

For each matrix \( B_K \), each vertex function combines with each \( \mathbb{R}^{3\times3}_{\text{sym}} \) basis, and then interacts with the three \( \phi \) functions over tetrahedra. Therefore, the matrix \( B_K \) is a \( 3 \times 24 \) matrix. Again, looping over each tetrahedron returns all the values for the matrix \( B \). For a single entry in \( B_K \),

\[
b_{i,6m+1+j} = b(\ell_j S_m, \phi_i) = \int_{\Omega} (\nabla \cdot \ell_j S_m) \cdot \phi_i \, dx \quad 1 \leq i \leq 3, \quad 1 \leq j \leq 4, \quad 1 \leq m \leq 6.
\]

When simplified, the rightmost expression equals \( |K|(S_m \nabla \ell_j) \cdot \phi_i \).

This results in the following entries in \( B_K \) for a single \( \ell_j \):

\[
[K] \begin{pmatrix}
\frac{\partial \ell_j}{\partial x} & 0 & 0 & \frac{\partial \ell_j}{\partial y} & 0 & 0 \\
0 & \frac{\partial \ell_j}{\partial y} & 0 & \frac{\partial \ell_j}{\partial z} & 0 & 0 \\
0 & 0 & \frac{\partial \ell_j}{\partial z} & 0 & \frac{\partial \ell_j}{\partial x} & 0
\end{pmatrix}.
\]

In addition to giving every value of the matrix \( B \), the structure is also given. For each \( \phi \) function, there are three non-zero entries with each vertex on the tetrahedron. Hence,

\[
nnz_B = \sum_{K=1}^{NT} 3(3)(4) = \sum_{K=1}^{NT} 36.
\]

### 2.3.3 Matrix \( C \)

The matrix \( C \) is computed with values determined the the faces in the tetrahedral mesh. To develop the matrix, a loop over faces determined the values for each of the three \( \phi \) functions over a face for a single
tetrahedron and, if applicable, its adjacent tetrahedron. For $\phi$ functions over $K$ and its adjacent tetrahedron $\hat{K}$,

\[
[\phi_{K,i}] : [\phi_{K,j}] = \begin{cases}
\frac{1}{2}(\delta_{ij} + (e_i^T n_+)(e_j^T n_+)), & i = j \\
\frac{1}{2}(e_i^T n_+)(e_j^T n_+), & i \neq j
\end{cases}
\]

\[
[\phi_{\hat{K},i}] : [\phi_{\hat{K},j}] = \begin{cases}
-\frac{1}{2}(\delta_{ij} + (e_i^T n_+)(e_j^T n_+)), & i = j \\
-\frac{1}{2}(e_i^T n_+)(e_j^T n_+), & i \neq j
\end{cases}
\]

$1 \leq K, \hat{K} \leq NT \quad 1 \leq i, j \leq 3$.

Because each $\phi$ function interacts with every $\phi$ function on its own tetrahedron or adjacent tetrahedron, there are only non-zero entries for interacting tetrahedra. Thus,

\[nnz_C = \sum_{K=1}^{NT} 9t_K\]

where $t_K$ is the number tetrahedra that interact with tetrahedron $K$.

This, with $nnz_A$ and $nnz_B$, gives the total number of non-zero entries in the global matrix, which gives the dimensions of the values and columns arrays. The work done to give the structure of the three pieces also gives the exact location of each non-zero value in the matrix, which allows the arrays to be filled appropriately.

3 Results

The code that was developed did not match the reductions in error as described in [1], however it did show similar behavior. More work would need to be done to identify why the code does not produce accurate results, but the current results over the Fichera Corner will be shown below. First, the two tables showing the error derived from calculations over the unit cube $\Omega = (0,1)^3$ with $\lambda = .3$ and $\mu = .35$, and whose exact $\bar{u}$ is

\[
\bar{u}(x) = \bar{u}(x_1, x_2, x_3) = \begin{pmatrix} 2^4 \\ 2^5 \\ 2^6 \end{pmatrix} x_1(1-x_1)x_2(1-x_2)x_3(1-x_3)
\]

from which $\sigma$ and $f$ can be calculated.
Table 1: Convergence of approximations of the stress tensor and displacement vector from [1].

<table>
<thead>
<tr>
<th>$h$</th>
<th>$|\sigma - \sigma_h|_{H(\text{div}),A}$</th>
<th>order</th>
<th>$|\vec{u}_h|_C$</th>
<th>order</th>
<th>$|\vec{u} - \vec{u}_h|_0$</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{-1}$</td>
<td>4.1723E+00</td>
<td>—</td>
<td>4.0747E-01</td>
<td>—</td>
<td>2.4720E-01</td>
<td>—</td>
</tr>
<tr>
<td>$2^{-2}$</td>
<td>2.3595E+00</td>
<td>0.82</td>
<td>3.5554E-01</td>
<td>0.20</td>
<td>1.7403E-01</td>
<td>0.51</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>1.2849E+00</td>
<td>0.88</td>
<td>2.5527E-01</td>
<td>0.48</td>
<td>1.1168E-01</td>
<td>0.64</td>
</tr>
<tr>
<td>$2^{-4}$</td>
<td>6.8023E-01</td>
<td>0.92</td>
<td>1.5243E-01</td>
<td>0.74</td>
<td>6.3889E-02</td>
<td>0.81</td>
</tr>
<tr>
<td>$2^{-5}$</td>
<td>3.5167E-01</td>
<td>0.95</td>
<td>8.3310E-02</td>
<td>0.87</td>
<td>3.4309E-02</td>
<td>0.90</td>
</tr>
</tbody>
</table>

Table 2: Convergence of approximations of the stress tensor and displacement vector from our code.

<table>
<thead>
<tr>
<th>$h$ (arbitrary units)</th>
<th>$|\sigma - \sigma_h|_{H(\text{div}),A}$</th>
<th>order</th>
<th>$|\vec{u}_h|_C$</th>
<th>order</th>
<th>$|\vec{u} - \vec{u}_h|_0$</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.93E+00</td>
<td>—</td>
<td>6.36E-01</td>
<td>—</td>
<td>2.99E-01</td>
<td>—</td>
</tr>
<tr>
<td>.5</td>
<td>4.71E+00</td>
<td>0.33</td>
<td>5.58E-01</td>
<td>0.19</td>
<td>2.44E-01</td>
<td>0.29</td>
</tr>
<tr>
<td>.25</td>
<td>3.42E+00</td>
<td>0.46</td>
<td>4.64E-01</td>
<td>0.27</td>
<td>1.56E-01</td>
<td>0.64</td>
</tr>
<tr>
<td>.125</td>
<td>2.66E+00</td>
<td>0.36</td>
<td>3.00E-01</td>
<td>0.62</td>
<td>8.74E-02</td>
<td>0.83</td>
</tr>
</tbody>
</table>

The code that we are currently using clearly does not match the convergence shown in [1].
Figure 1: The Fichera Corner cut to view different magnitudes of $\mathbf{U}_h$, as calculated by our code.
Acknowledgments

• The REU Site is supported by the National Science Foundation under grant no. 1758006.
• My faculty mentor for this project was Dr. Jeff Ovall

References
