5-2014

An Applied Functional and Numerical Analysis of a 3-D Fluid-Structure Interactive PDE

Thomas J. Clark
University of Nebraska-Lincoln, s-tclark15@math.unl.edu

AN APPLIED FUNCTIONAL AND NUMERICAL ANALYSIS OF A 3-D FLUID-STRUCTURE INTERACTIVE PDE.

by

Thomas J. Clark

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Mathematics

Under the Supervision of Professor George Avalos

Lincoln, Nebraska

May, 2014

AN APPLIED FUNCTIONAL AND NUMERICAL ANALYSIS OF A 3-D FLUID-STRUCTURE INTERACTIVE PDE.

Thomas J. Clark, Ph.D.

University of Nebraska, 2014

Adviser: George Avalos

We will present qualitative and numerical results on a partial differential equation (PDE) system which models a certain fluid-structure dynamics. In Chapter 1, the wellposedness of this PDE model is established by means of constructing for it a nonstandard semigroup generator representation; this representation is essentially accomplished by an appropriate elimination of the pressure. This coupled PDE model involves the Stokes system which evolves on a three dimensional domain $\mathcal{O}$ being coupled to a fourth order plate equation, possibly with rotational inertia parameter $\rho > 0$, which evolves on a flat portion Ω of the boundary of $\mathcal{O}$. The coupling on Ω is implemented via the Dirichlet trace of the Stokes system fluid variable – and so the no-slip condition is necessarily not in play – and via the Dirichlet boundary trace of the pressure, which essentially acts as a forcing term on this elastic portion of the boundary. We note here that inasmuch as the Stokes fluid velocity does not vanish on Ω, the pressure variable cannot be eliminated by the classic Leray projector; instead, the pressure is identified as the solution of a certain elliptic boundary value problem. Eventually, wellposedness of this fluid-structure dynamics is attained through a certain nonstandard variational ("inf-sup") formulation. Chapter 1 also includes two abstract results. The first qualitative result shows that zero is in the resolvent set of the operator which generates the C_0-semigroup in the wellposedness argument. The second establishes the backward uniqueness property for the fluid-structure system.

Subsequently, in Chapter 2 we show how our constructive proof of wellposedness naturally gives rise to a certain mixed finite element method for numerically

approximating solutions of this fluid-structure dynamics. This method is demonstrated for a certain test problem in the $\rho = 0$ case. In addition, error estimates for the rate of convergence of the numerical method are provided and a test problem is solved to demonstrate the efficacy of the numerical code.

GRANT INFORMATION

NSF-DMS 0908476 partially supported this research.

NSF-DMS 0838463 partially supported this research through an MCTP Traineeship.

Contents

List of Figures

List of Tables

Chapter 1

Mathematical Analysis of a Fluid-Structure Interation

1.1 Introduction

In this work we investigate a PDE model of a particular Fluid-Structure Interaction of interest in the mathematical literature. Fluid-structure interactions have numerous applications modeling a variety of physical phenomena including blood flow, vibrations in elastic membranes, and aerodynamics of wings and structures such as bridges and tall buildings and the motion of cellular organisms in a fluid; see [19] for an engineering perspective. The model of interest here was explored previously in [14] and the references therein. It has some similar features to fluid-structure models of the eye. For example occular pressure is important physically and the coupling in our model is achieved through the pressure term. The techniques used in this work are applicable to many other models as well. Chapter 1 focuses on the abstract mathematical results we demonstrate for the model including wellposedness of the system, spectral information, and the backward uniqueness property. Chapter 2 describes how the particular method of proof employed in Chapter 1 can be leveraged to derive a numerical method for

generating an approximate solution of the PDE system. The Appendix contains the MATLAB code which is referenced in Chapter 2.

1.2 The PDE and Setting for Wellposedness

One of the main objectives in this work is to provide a proof for semigroup wellposedness with respect to the fluid-structure partial differential equation (PDE) model considered in [14] - see also [13] and [15]. The proof here will be wholly different than that originally given in [14], and has the virtue of giving insight into a mixed finite element method (FEM) formulation so as to numerically approximate the solution of the fluid and structure variables. A numerical analysis involving this fluid-structure FEM will constitute the second part of this work. Throughout, we will consider situations in which either the "Kirchhoff" plate PDE is in place or its irrotational version to describe the structural component of the fluid-structure model (only the irrotational version is considered in [14]). The geometrical situation will be identical to that in [14]. We state it here verbatim: $\mathcal{O} \subset \mathbb{R}^3$ will be a bounded domain with smooth boundary. Moreover, $\partial\mathcal{O} = \bar{\Omega} \cup \bar{S}$, with $\Omega \cap S = \emptyset$, and specifically

$$\Omega \subset \{x = (x_1, x_2, 0)\}\,, \text{ and surface } S \subset \{x = (x_1, x_2, x_3) : x_3 \leq 0\}\,.$$

In consequence, if $\nu(x)$ denotes the exterior unit normal vector to $\partial\mathcal{O}$, then

$$\nu|_{\Omega} = [0, 0, 1]\,. \tag{1.1}$$

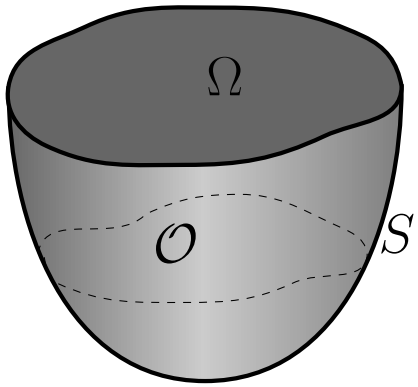


Figure 1.1: The Fluid-Structure Geometry.

With "rotational inertia parameter" $\rho \geq 0$, the PDE model is as follows, in solution variables $w(x,t)$, $u(x,t) = [u^{(1)}(x,t), u^{(2)}(x,t), u^{(3)}(x,t)]$, and $p(x,t)$:

$$w_{tt} - \rho \Delta w_{tt} + \Delta^2 w = p|_\Omega \quad \text{in } \Omega \times (0,T); \tag{1.2}$$

$$w = \frac{\partial w}{\partial \nu} = 0 \quad \text{on } \partial\Omega; \tag{1.3}$$

$$u_t - \Delta u + \nabla p = 0 \quad \text{in } \mathcal{O} \times (0,T); \tag{1.4}$$

$$\text{div}(u) = 0 \quad \text{in } \mathcal{O} \times (0,T); \tag{1.5}$$

$$u = \vec{0} \text{ on } S \quad \text{and } u = [u^{(1)}, u^{(2)}, u^{(3)}] = [0,0,w_t] \quad \text{on } \Omega, \tag{1.6}$$

with initial conditions

$$[w(0), w_t(0), u(0)] = [w_1, w_2, u_0] \in \mathbf{H}_\rho. \tag{1.7}$$

(So when $\rho = 0$, the irrotational plate dynamics are in play; when $\rho > 0$ we have instead the full Kirchhoff plate.) Here, the space of initial data $\mathbf{H}_\rho$ is defined as follows: Let

$$\mathcal{H}_{\text{fluid}} = \left\{ f \in \mathbf{L}^2(\mathcal{O}) : \text{div}(f) = 0; \;\; f \cdot \nu|_S = 0 \right\}, \tag{1.8}$$

$$\widehat{L}^2(\Omega) = \{ w \in L^2(\Omega) \; : \; \int_\Omega w \, d\Omega = 0 \}, \tag{1.9}$$

and

$$W_\rho = \begin{cases} \widehat{L}^2(\Omega), \ \text{ if } \rho = 0, \\ \\ H_0^1(\Omega) \cap \widehat{L}^2(\Omega), \ \text{ if } \rho > 0. \end{cases} \tag{1.10}$$

Therewith, we then set

$$\begin{aligned} \mathbf{H}_\rho = \Big\{ [\omega_1, \omega_2, f] \in \Big[H_0^2(\Omega) \cap \widehat{L}^2(\Omega)\Big] \times W_\rho \times \mathcal{H}_{\text{fluid}} \\ \text{with } \ f \cdot \nu|_\Omega = [0,0,f^3] \cdot [0,0,1] = \omega_2 \Big\}. \end{aligned} \tag{1.11}$$

Moreover, let $A_D : D(A_D) \subset L^2(\Omega) \to L^2(\Omega)$ be given by

$$A_D g = -\Delta g, \quad D(A_D) = H^2(\Omega) \cap H_0^1(\Omega). \tag{1.12}$$

If we subsequently make the denotation for all $\rho \geq 0$,

$$P_\rho = I + \rho A_D, \ \ D(P_\rho) = \begin{cases} L^2(\Omega), \ \text{ if } \rho = 0, \\ D(A_D), \ \text{ if } \rho > 0, \end{cases} \tag{1.13}$$

then the mechanical PDE component (1.2)-(1.3) can be written as

$$P_\rho w_{tt} + \Delta^2 w = p|_\Omega \ \text{ on } (0,T).$$

Using the characterization from [25] that

$$D(P_\rho^{\frac{1}{2}}) = \begin{cases} L^2(\Omega), \ \text{ if } \rho = 0, \\ H_0^1(\Omega), \ \text{ if } \rho > 0, \end{cases}$$

then we can endow the Hilbert space $\mathbf{H}_\rho$ with a norm-inducing inner product

$$\Big(\big[\omega_1, \omega_2, f\big], \big[\tilde{\omega}_1, \tilde{\omega}_2, \tilde{f}\big] \Big)_{\mathbf{H}_\rho} = (\Delta\omega_1, \Delta\tilde{\omega}_1)_\Omega + (P_\rho^{\frac{1}{2}}\omega_2, P_\rho^{\frac{1}{2}}\tilde{\omega}_2)_\Omega + (f, \tilde{f})_{\mathcal{O}},$$

where $(\cdot,\cdot)_\Omega$ and $(\cdot,\cdot)_\mathcal{O}$ are the L^2-inner products on their respective geometries.
We note here, as there was in [14], the necessity for imposing that the wave initial displacement and the wave velocity each have zero mean average. To see this: Invoking the boundary condition (1.6) and the fact that normal vector $\nu = [0,0,1]$ on Ω, we have then by Green's formula, that for all $t \geq 0$,

$$\int_\Omega w_t(t)d\Omega = \int_\Omega u^{(3)}(t)d\Omega = \int_{\partial\mathcal{O}} u(t)\cdot\nu\, d\partial\mathcal{O} = 0. \tag{1.14}$$

And so we have necessarily,

$$\int_\Omega w(t)\, d\Omega = \int_\Omega w_1 d\Omega, \text{ for all } t \geq 0.$$

This accounts for the choice of the structural finite energy space components for $\mathbf{H}_\rho$, in (1.11). The zero-integral condition is place on the plate displacement term w as well so that the all zero state can be a solution to the PDE system. (Some results in [14] allow for $w \in H_0^2(\Omega)$ only.)
As we said, our proof of wellposedness hinges upon demonstrating the existence of a modeling C_0-semigroup $\left\{e^{\mathcal{A}_\rho t}\right\}_{t\geq 0} \subset \mathcal{L}(\mathbf{H}_\rho)$, for appropriate generator
$\mathcal{A}_\rho : D(\mathcal{A}_\rho) \subset \mathbf{H}_\rho \to \mathbf{H}_\rho$. Subsequently, by means of this family, the solution to (1.2)-(1.7), for initial data $[w_1, w_2, u_0] \in \mathbf{H}_\rho$, will then of course be given via the relation

$$\begin{bmatrix} w(t) \\ w_t(t) \\ u(t) \end{bmatrix} = e^{\mathcal{A}_\rho t} \begin{bmatrix} w_1 \\ w_2 \\ u_0 \end{bmatrix} \in C([0,T];\mathbf{H}_\rho). \tag{1.15}$$

Our particular choice here of generator $\mathcal{A}_\rho : D(\mathcal{A}_\rho) \subset \mathbf{H}_\rho \to \mathbf{H}_\rho$ is dictated by the following consideration:
If $p(t)$ is a viable pressure variable for (1.2)-(1.7), then pointwise in time $p(t)$ necessarily

satisfies the following boundary value problem:

$$\Delta p = 0 \ \text{ in } \mathcal{O}; \tag{1.16}$$

$$\frac{\partial p}{\partial \nu} + P_\rho^{-1} p = P_\rho^{-1}\Delta^2 w + \Delta u^{(3)}\big|_\Omega \ \text{ on } \Omega; \tag{1.17}$$

$$\frac{\partial p}{\partial \nu} = \Delta u \cdot \nu|_S \ \text{ on } S. \tag{1.18}$$

To show the validity of (1.16)-(1.18): Taking the divergence of both sides of (1.4) and using the divergence free condition in (1.5) yields equation (1.16). Moreover, dotting both sides of (1.4) with the unit normal vector ν, and then subsequently taking the resulting trace on S will yield the boundary condition (1.18). (Implicitly, we are also using the fact that $u = \vec{0}$ on S.)

Finally, we consider the particular geometry which is in play (where $\nu = [0, 0, 1]$ on Ω to establish (1.17)). Using the equation (1.2) and boundary condition (1.6), we have on Ω

$$\begin{aligned} P_\rho^{-1}\Delta^2 w &= -w_{tt} + P_\rho^{-1} p\big|_\Omega \\ &= -\frac{d}{dt}[0, 0, w_t] \cdot \nu + P_\rho^{-1} p\big|_\Omega \\ &= -\left[u_t \cdot \nu\right]_\Omega + P_\rho^{-1} p\big|_\Omega \\ &= -\left[\Delta u \cdot \nu\right]_\Omega + \frac{\partial p}{\partial \nu}\bigg|_\Omega + P_\rho^{-1} p\big|_\Omega , \end{aligned}$$

which gives (1.17).

The boundary value problem (1.16)-(1.18) can be solved through the agency of the following "Robin" maps R_ρ and $\tilde{R}_\rho$: We define

$$R_\rho g = f \Leftrightarrow \left\{ \Delta f = 0 \ \text{ in } \mathcal{O};\ \frac{\partial f}{\partial \nu} + P_\rho^{-1} f = g \ \text{ on } \Omega;\ \frac{\partial f}{\partial \nu} = 0 \text{ on } S \right\}. \tag{1.19}$$

$$\tilde{R}_\rho g = f \Leftrightarrow \left\{ \Delta f = 0 \ \text{ in } \mathcal{O};\ \frac{\partial f}{\partial \nu} + P_\rho^{-1} f = 0 \ \text{ on } \Omega;\ \frac{\partial f}{\partial \nu} = g \text{ on } S \right\}. \tag{1.20}$$

Therewith, we have that,

$$R_\rho \in \mathcal{L}\big(H^{-\frac{1}{2}}(\Omega), H^1(\mathcal{O})\big);\ \ \tilde{R}_\rho \in \mathcal{L}\big(H^{-\frac{1}{2}}(S), H^1\mathcal{O})\big). \tag{1.21}$$

(See e.g. [31]. We are also using implicity the fact that P_ρ^{-1} is positive definite, self-adjoint on Ω, and moreover manifests elliptic regularity.)
Therewith, the pressure variable $p(t)$, as necessarily the solution of (1.16)-(1.18), can be written pointwise in time as

$$p(t) = G_{\rho,1}(w(t)) + G_{\rho,2}(u(t)), \tag{1.22}$$

where

$$G_{\rho,1}(w) = R_\rho(P_\rho^{-1}\Delta^2 w); \tag{1.23}$$

$$G_{\rho,2}(u) = R_\rho(\Delta u^{(3)}\big|_\Omega) + \tilde{R}_\rho(\Delta u \cdot \nu|_S). \tag{1.24}$$

These relations suggest the following choice for the generator $\mathcal{A}_\rho : D(\mathcal{A}_\rho) \subset \mathbf{H}_\rho \to \mathbf{H}_\rho$. We set

$$\mathcal{A}_\rho \equiv \begin{bmatrix} 0 & I & 0 \\ -P_\rho^{-1}\Delta^2 + P_\rho^{-1}G_{\rho,1}\big|_\Omega & 0 & P_\rho^{-1}G_{\rho,2}\big|_\Omega \\ -\nabla G_{\rho,1} & 0 & \Delta - \nabla G_{\rho,2} \end{bmatrix}; \tag{1.25}$$

$$\text{with } D(\mathcal{A}_\rho) = \big\{\, [w_1, w_2, u] \in \ \mathbf{H}_\rho \text{ satisfying :}$$

$$\text{(a) } w_1 \in \mathcal{S}_\rho \equiv \begin{cases} H^4(\Omega) \cap H_0^2(\Omega), & \text{if } \rho = 0; \\ H^3(\Omega) \cap H_0^2(\Omega), & \text{if } \rho > 0; \end{cases} \tag{1.26}$$

$$\text{(b) } w_2 \in H_0^2(\Omega),\ u \in \mathbf{H}^2(\mathcal{O});$$

$$\text{(c) } u = \vec{0} \text{ on } S \text{ and } u = [0, 0, w_2] \text{ on } \Omega\big\}, \tag{1.27}$$

(c.f. the generator and domain described in an earlier version of [14] in arXiv:1109.4324.) (Note that as $\Delta u \in \mathbf{L}^2(\mathcal{O})$ and $\operatorname{div}(\Delta u) = 0$, then by Theorem 1.2, p. 9 in [34], we have the trace regularity

$$\Delta u \cdot \nu\big|_{\partial\mathcal{O}} \in H^{-\frac{1}{2}}(\partial\mathcal{O}); \tag{1.28}$$

and so the pressure term

$$p \equiv G_{\rho,1}(w_1) + G_{\rho,2}(u) \in H^1(\mathcal{O}). \tag{1.29}$$

In short the domain of $\mathcal{A}_\rho : D(\mathcal{A}_\rho) \subset \mathbf{H}_\rho \to \mathbf{H}_\rho$ is well-defined.)

In what follows, we will have need of solution maps for certain inhomogeneous Stokes flows. To wit: For given $\phi \in H^{\frac{3}{2}}(\Omega)$, let $[\tilde{f}(\phi), \tilde{\pi}(\phi)] \in \mathbf{H}^2(\mathcal{O}) \times H^1(\mathcal{O})/\mathbb{R}$ solve

$$\begin{cases} \lambda \tilde{f} - \Delta \tilde{f} + \nabla \tilde{\pi} = 0 \ \text{ in } \mathcal{O}, \\ \operatorname{div}(\tilde{f}) = \dfrac{1 \cdot \int_\Omega \phi \, d\Omega}{\operatorname{meas}(\mathcal{O})} \ \text{ in } \mathcal{O}, \\ \tilde{f}\big|_S = [0,0,0] \ \text{ on } S, \\ \tilde{f}\big|_\Omega = [0,0,\phi] \ \text{ on } \Omega. \end{cases} \tag{1.30}$$

We note that the classic compatibility condition for solvability is satisfied, and that pressure variable $\tilde{\pi}$ is uniquely defined up to a constant; see e.g., Theorem 2.4, p. 31 of [34]. Then by Agmon-Douglis-Nirenberg, we have
$\left[\tilde{f}, \tilde{\pi}\right] \in \mathcal{L}(H^{\frac{3}{2}}(\Omega), \mathbf{H}^2(\mathcal{O}) \times H^1(\mathcal{O})/\mathbb{R})$, with

$$\|\tilde{f}(\phi)\|_{\mathbf{H}^2(\mathcal{O})} + \|\tilde{\pi}(\phi)\|_{\frac{H^1(\mathcal{O})}{\mathbb{R}}} \leq C \|\phi\|_{H^{\frac{3}{2}}(\Omega)}. \tag{1.31}$$

(see Proposition 2.2, p. 33 of [34]).

In a similar way, we define for fluid data $u^* \in \mathbf{L}^2(\mathcal{O})$, the solution variables

$[\tilde{\mu}(u^*), \tilde{q}(u^*)] \in \mathbf{H}^2(\mathcal{O}) \times H^1(\mathcal{O})/\mathbb{R}$, where $[\tilde{\mu}, \tilde{q}]$ solve

$$\begin{cases} \lambda\tilde{\mu} - \Delta\tilde{\mu} + \nabla\tilde{q} = u^* & \text{in } \mathcal{O}, \\ \operatorname{div}(\tilde{\mu}) = 0 & \text{in } \mathcal{O}, \\ \tilde{\mu}\big|_{\partial\mathcal{O}} = \vec{0} & \text{on } \partial\mathcal{O}. \end{cases} \tag{1.32}$$

Again by Agmon-Douglis-Nirenberg we have $[\tilde{\mu}, \tilde{q}] \in \mathcal{L}(\mathbf{L}^2(\mathcal{O}), \mathbf{H}^2(\mathcal{O}) \times H^1(\mathcal{O})/\mathbb{R})$, with

$$\|\tilde{\mu}(u^*)\|_{\mathbf{H}^2(\mathcal{O})} + \|\tilde{q}(u^*)\|_{\frac{H^1(\mathcal{O})}{\mathbb{R}}} \leq C\|u^*\|_{\mathbf{L}^2(\mathcal{O})}. \tag{1.33}$$

These two fluid maps will be invoked in the proof of Theorem 1 below, which will yield solvability of the following resolvent equation for $\lambda > 0$: Namely, for given $[w_1^*, w_2^*, u^*] \in \mathbf{H}_\rho$, $[w_1, w_2, u] \in D(\mathcal{A}_\rho)$ solves

$$(\lambda I - \mathcal{A}_\rho) \begin{bmatrix} w_1 \\ w_2 \\ u \end{bmatrix} = \begin{bmatrix} w_1^* \\ w_2^* \\ u^* \end{bmatrix}. \tag{1.34}$$

After setting the pressure variable $p = G_{\rho,1}(w_1) + G_{\rho,2}(u)$ in (1.25), the resolvent

equation (1.34) is equivalent to the following system:

$$\lambda w_1 - w_2 = w_1^* \qquad \text{in } \Omega; \tag{1.35}$$

$$\lambda w_2 + P_\rho^{-1}\Delta^2 w_1 - P_\rho^{-1} p\big|_\Omega = w_2^* \qquad \text{in } \Omega; \tag{1.36}$$

$$w_1 = \frac{\partial w_1}{\partial \nu} = 0 \qquad \text{on } \partial\Omega; \tag{1.37}$$

$$\lambda u - \Delta u + \nabla p = u^* \qquad \text{in } \mathcal{O}; \tag{1.38}$$

$$\operatorname{div}(u) = 0 \qquad \text{in } \mathcal{O}; \tag{1.39}$$

$$u\big|_S = [0,0,0] \qquad \text{on } S; \tag{1.40}$$

$$u\big|_\Omega = [0,0,w_2] \quad \text{in } \Omega. \tag{1.41}$$

In particular, it will be seen in Theorem 1 that the solution variable w_1 in (1.34) can be recovered through finding the unique solution pair $[w_1, \tilde{c}] \in H_0^2(\Omega) \times \mathbb{R}$ which solves

$$\begin{cases} a_\lambda(w_1, \phi) + b(\phi, \tilde{c}) = \mathbb{F}(\phi) & \forall \phi \in H_0^2(\Omega), \\ b(w_1, r) = 0 & \forall r \in \mathbb{R}; \end{cases} \tag{1.42}$$

where:

$$\begin{aligned} a_\lambda(\psi,\phi) &= \lambda^2(P_\rho^{1/2}\psi, P_\rho^{1/2}\phi)_\Omega + (\Delta\psi, \Delta\phi)_\Omega + \lambda(\nabla\tilde{f}(\psi), \nabla\tilde{f}(\phi))_{\mathcal{O}} + \lambda^2(\tilde{f}(\psi), \tilde{f}(\phi))_{\mathcal{O}}, \\ &\qquad \forall\, \psi \text{ and } \phi \in H_0^2(\Omega); \\ b(\phi, r) &= -r\int_\Omega \phi \, d\Omega, \quad \forall \phi \in H_0^2(\Omega) \text{ and } r \in \mathbb{R}; \\ \mathbb{F}(\phi) &= (\nabla\tilde{f}(w_1^*), \nabla\tilde{f}(\phi))_{\mathcal{O}} + \lambda(\tilde{f}(w_1^*), \tilde{f}(\phi))_{\mathcal{O}} - (\nabla\tilde{\mu}(u^*), \nabla\tilde{f}(\phi))_{\mathcal{O}} \\ &\qquad - \lambda(\tilde{\mu}(u^*), \tilde{f}(\phi))_{\mathcal{O}} + (u^*, \tilde{f}(\phi))_{\mathcal{O}} + (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega, \quad \forall \phi \in H_0^2(\Omega). \end{aligned} \tag{1.43}$$

Theorem 1.

(i) The operator $\mathcal{A}_\rho : D(\mathcal{A}_\rho) \subset \mathbf{H}_\rho \to \mathbf{H}_\rho$ is maximal dissipative. Therefore by the

Lumer-Phillips Theorem it generates a C_0-semigroup of contractions $\left\{e^{\mathcal{A}_\rho t}\right\}_{t\geq 0}$ on $\mathbf{H}_\rho$.

(ii) Let $\lambda > 0$ and $[w_1^*, w_2^*, u^*] \in \mathbf{H}_\rho$ be given. (By part (i), there exists $[w_1, w_2, u] \in D(\mathcal{A}_\rho)$ which solves (1.34).) Then the structural solution component w_1 and constant component $\tilde{c}$ of the associated pressure term p can be characterized as the solution pair of the variational system (1.42). Subsequently the remaining unknown terms are given by

$$\begin{aligned} w_2 &= \lambda w_1 - w_1^*, \\ u &= \tilde{f}(\lambda w_1 - w_1^*) + \tilde{\mu}(u^*), \\ p &= \tilde{\pi}(\lambda w_1 - w_1^*) + \tilde{q}(u^*) + \tilde{c}, \end{aligned} \tag{1.44}$$

(after utilizing the maps $\tilde{f}$ and $\tilde{\pi}$ in (1.30) and $\tilde{\mu}$ and $\tilde{q}$ in (1.32)).

1.3 Proof of Theorem 1

1.3.1 Proof of Dissipativity

Let $[w_1, w_2, u] \in D(\mathcal{A}_\rho)$ be given. Therewith, we set the pressure

$$\pi_0 \equiv G_{\rho,1}(w_1) + G_{\rho,2}(u). \tag{1.45}$$

(So by (1.23)-(1.24), (1.21) and (1.29), we have that $\pi_0 \in H^1(\mathcal{O})$.) We have then, upon using the definition of the domain in (1.27),

$$\begin{aligned}
&\left(\mathcal{A}_\rho \begin{bmatrix} w_1 \\ w_2 \\ u \end{bmatrix}, \begin{bmatrix} w_1 \\ w_2 \\ u \end{bmatrix} \right)_{\mathbf{H}_\rho} \\
&= (\Delta w_2, \Delta w_1)_\Omega + (-\Delta^2 w_1 + \pi_0|_\Omega, w_2)_\Omega + (\Delta u - \nabla \pi_0, u)_{\mathcal{O}} \\
&= (\Delta w_2, \Delta w_1)_\Omega + (\nabla \Delta w_1, \nabla w_2)_\Omega + (\pi_0|_\Omega (0,0,1), (u^{(1)}, u^{(2)}, w_2))_\Omega \\
&\quad - (\nabla u, \nabla u)_{\mathcal{O}} + \left\langle \frac{\partial u}{\partial \nu}, u \right\rangle_\Omega - \langle \pi_0 \nu, u \rangle_\Omega \\
&= (\Delta w_2, \Delta w_1)_\Omega - (\Delta w_1, \Delta w_2)_\Omega - (\nabla u, \nabla u)_{\mathcal{O}} \\
&\quad + \left(\begin{bmatrix} \partial_{x_3} u^{(1)} \\ \partial_{x_3} u^{(2)} \\ \partial_{x_3} u^{(3)} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ u^{(3)} \end{bmatrix} \right)_\Omega \\
&= -2i \mathrm{Im} \, (\Delta w_1, \Delta w_2)_\Omega - \|\nabla u\|^2_{\mathcal{O}}, \qquad (1.46)
\end{aligned}$$

where in the last step, we have used we have used $u = \vec{0}$ on S, $\mathrm{div}(u) = 0$ and $[u^{(1)}, u^{(2)}, u^{(3)}] = [0, 0, w_2]$ on Ω. This establishes dissipativity.

1.3.2 Maximality

In what follows we will make use of the Babuška-Brezzi Theorem. We state it here directly from p. 116 of [28].

Theorem 2 (Babuška-Brezzi)**.** *Let Σ, M be Hilbert spaces and $a : \Sigma \times \Sigma \to \mathbb{R}$, $b : \Sigma \times M \to \mathbb{R}$, bilinear forms which are continuous. Let*

$$Z = \{\sigma \in \Sigma \,\big|\, b(\sigma, q) = 0, \quad \textit{for every } q \in M\}. \qquad (1.47)$$

Assume that $a(\cdot,\cdot)$ is Z-elliptic, i.e. there exists a constant $\alpha > 0$ such that

$$a(\sigma,\sigma) \geq \|\sigma\|_{\Sigma}^2, \tag{1.48}$$

for every $\sigma \in Z$. Assume further that there exists a constant $\beta > 0$ such that

$$\sup_{\tau\in\Sigma} \frac{b(\tau,q)}{\|\tau\|_{\Sigma}} \geq \beta\|q\|_M, \quad \textit{for every } q \in M. \tag{1.49}$$

Then if $\kappa \in \Sigma$ and $\ell \in M$, there exists a unique pair $(\sigma,p) \in \Sigma \times M$ such that

$$a(\sigma,\tau) + b(\tau,p) = (\kappa,\tau), \quad \textit{for every } \tau \in \Sigma. \tag{1.50}$$

$$b(\sigma,q) = (\ell,q), \quad \textit{for every } q \in M. \tag{1.51}$$

For $\lambda > 0$, we will show that $\text{Range}(\lambda I - \mathcal{A}_\rho) = \mathbf{H}_\rho$. To this end, let $[w_1^*, w_2^*, u^*] \in \mathbf{H}_\rho$ be given. We must find $[w_1, w_2, u] \in \mathcal{D}(\mathcal{A}_\rho) \subset \mathbf{H}_\rho$ which solves

$$(\lambda I - \mathcal{A}_\rho)\begin{bmatrix} w_1 \\ w_2 \\ u \end{bmatrix} = \begin{bmatrix} w_1^* \\ w_2^* \\ u^* \end{bmatrix}. \tag{1.52}$$

Using the structural component (1.35) and (1.36), we then have the boundary value problem

$$\begin{cases} \lambda^2 w_1 + P_\rho^{-1}\Delta^2 w_1 - P_\rho^{-1}p\big|_\Omega = \lambda w_1^* + w_2^* \text{ in } \Omega, \\ w_1\big|_{\partial\Omega} = \frac{\partial w_1}{\partial \nu}\big|_{\partial\Omega} = 0 \ \text{ on } \partial\Omega, \end{cases} \tag{1.53}$$

as well as the fluid system

$$\begin{cases} \lambda u - \Delta u + \nabla p = u^* & \text{in } \mathcal{O}, \\ \operatorname{div}(u) = 0 & \text{in } \mathcal{O}, \\ u\big|_S = [0,0,0] & \text{on } S, \\ u\big|_\Omega = [0,0,\lambda w_1 - w_1^*] & \text{on } \Omega. \end{cases} \tag{1.54}$$

Since $[w_1, w_2, u] \in \mathbf{H}_\rho$, we also have that

$$\int_\Omega w_1 \, d\Omega = 0. \tag{1.55}$$

With this compatibilty condition in mind, by way of "decoupling" the systems (1.53) and (1.54), we proceed as follows. We apply P_ρ to the mechanical equation in (1.53) and then multiply by test function $\phi \in H_0^2(\Omega)$. Subsequently integrating over Ω then yields

$$\left([\lambda^2 P_\rho + \Delta^2] w_1 - p|_\Omega \, , \phi\right)_\Omega = (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega$$

(where $(\cdot,\cdot)_\Omega$ might also indicate the duality pairing between $H_0^2(\Omega)$ and $H^{-2}(\Omega)$). Upon integration by parts and using $\phi\big|_{\partial\Omega} = \dfrac{\partial \phi}{\partial \nu}\Big|_{\partial\Omega} = 0$, we then have

$$\lambda^2 (P_\rho w_1, \phi)_\Omega + (\Delta w_1, \Delta \phi)_\Omega + \left(\frac{\partial u}{\partial \nu} - p\nu, \begin{bmatrix} 0 \\ 0 \\ \phi \end{bmatrix} \right)_\Omega = (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega \, . \tag{1.56}$$

(Note that in obtaining this expression we have again used $\operatorname{div}(u) = 0$ on Ω and normal vector $\nu|_\Omega = [0,0,1]$.)

Now, from (1.54) and (1.55), we can use the fact that in terms of the maps in (1.30) and

(1.32) above, we can write fluid variables u and p of (1.53)-(1.54) as

$$\begin{cases} u = \tilde{f}(\lambda w_1 - w_1^*) + \tilde{\mu}(u^*), \\ p = \tilde{\pi}(\lambda w_1 - w_1^*) + \tilde{q}(u^*) + \tilde{c}, \end{cases} \tag{1.57}$$

for some (to be determined) constant $\tilde{c}$, justifying (1.44).

Applying this representation to (1.56), subsequently integrating by parts, and using the mapping in (1.30), we then have for every $\phi \in H_0^2(\Omega)$,

$$\lambda^2(P_\rho w_1, \phi)_\Omega + (\Delta w_1, \Delta\phi)_\Omega + \left(\nabla u, \nabla \tilde{f}(\phi)\right)_{\mathcal{O}} + \left(\Delta u, \tilde{f}(\phi)\right)_{\mathcal{O}} - \left(p, \mathrm{div}[\tilde{f}(\phi)]\right)_{\mathcal{O}} - \left(\nabla p, \tilde{f}(\phi)\right)_{\mathcal{O}}$$
$$= (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega \,.$$

Using now the representations in (1.57), we have then

$$\lambda^2(P_\rho^{1/2} w_1, P_\rho^{1/2}\phi)_\Omega + (\Delta w_1, \Delta\phi)_\Omega + (\nabla \tilde{f}(\lambda w_1 - w_1^*), \nabla \tilde{f}(\phi))_{\mathcal{O}} + (\nabla \tilde{\mu}(u^*), \nabla \tilde{f}(\phi))_{\mathcal{O}}$$
$$+ (\Delta \tilde{f}(\lambda w_1 - w_1^*), \tilde{f}(\phi))_{\mathcal{O}} + (\Delta \tilde{\mu}(u^*), \tilde{f}(\phi))_{\mathcal{O}} - (\nabla \tilde{\pi}(\lambda w_1 - w_1^*), \tilde{f}(\phi))_{\mathcal{O}}$$
$$- (\nabla \tilde{q}(u^*), \tilde{f}(\phi))_{\mathcal{O}} - (\tilde{\pi}(\lambda w_1 - w_1^*), \mathrm{div}(\tilde{f}(\phi)))_{\mathcal{O}} - (\tilde{q}(u^*), \mathrm{div}(\tilde{f}(\phi)))_{\mathcal{O}} - \tilde{c}(1, \mathrm{div}(\tilde{f}(\phi)))_{\mathcal{O}}$$
$$= (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega \,.$$

Now using (1.30) and (1.32) as well as the fact that $\tilde{\pi}, \tilde{q} \in \frac{L^2(\mathcal{O})}{\mathbb{R}}$, we rewrite this expression as

$$\lambda^2(P_\rho^{1/2} w_1, P_\rho^{1/2}\phi)_\Omega + (\Delta w_1, \Delta\phi)_\Omega + \lambda(\nabla \tilde{f}(w_1), \nabla \tilde{f}(\phi))_{\mathcal{O}} + \lambda^2(\tilde{f}(w_1), \tilde{f}(\phi))_{\mathcal{O}} - \tilde{c}\int_\Omega \phi \, d\Omega$$
$$= (\nabla \tilde{f}(w_1^*), \nabla \tilde{f}(\phi))_{\mathcal{O}} + \lambda(\tilde{f}(w_1^*), \tilde{f}(\phi))_{\mathcal{O}} - (\nabla \tilde{\mu}(u^*), \nabla \tilde{f}(\phi))_{\mathcal{O}}$$
$$- \lambda(\tilde{\mu}(u^*), \tilde{f}(\phi))_{\mathcal{O}} + (u^*, \tilde{f}(\phi))_{\mathcal{O}} + (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega \,. \tag{1.58}$$

This variational relation and the constraint (1.55) establish now the characterization of the range condition (1.52) with the mixed variational problem (1.42). This

characterization, along with (1.35) and (1.57), establishes Theorem 1(ii).

By way of establishing wellposedness of (1.42): The bilinear forms $a_\lambda(\cdot,\cdot)$ and $b(\cdot,\cdot)$ are readily seen to be continuous. In addition a_λ is $H_0^2(\Omega)$-elliptic. The existence of a unique pair $[w_1, \tilde{c}] \in H_0^2(\Omega) \times \mathbb{R}$ which solves (1.42) will follow the Babuška-Brezzi Theorem if we establish the following "inf-sup"condition, for some positive constant β:

$$\sup_{\phi \in H_0^2(\Omega)} \frac{b(\phi, r)}{\|\phi\|_{H_0^2(\Omega)}} \geq \beta |r|, \quad \forall r \in \mathbb{R}. \tag{1.59}$$

To this end, consider the function $\xi \in H^4(\Omega) \cap H_0^2(\Omega)$ which solves

$$\Delta^2 \xi = 1 \text{ in } \Omega; \ \ \xi|_{\partial\Omega} = \left.\frac{\partial \xi}{\partial \nu}\right|_{\partial\Omega} = 0 \text{ on } \partial\Omega. \tag{1.60}$$

By Green's Formula we then have

$$\int_\Omega \xi \cdot 1 \, d\Omega = \int_\Omega \xi \Delta^2 \xi \, d\Omega = \int_\Omega \Delta\xi \Delta\xi \, d\Omega. \tag{1.61}$$

Therewith, for given scalar $r \in \mathbb{R}$, let $\eta \equiv -\text{sgn}(r)\xi$. Then

$$\begin{aligned} \sup_{\phi \in H_0^2(\Omega)} \frac{b(\phi, r)}{\|\phi\|_{H_0^2(\Omega)}} &\geq \frac{-r \int_\Omega \eta \, d\Omega}{\|\eta\|_{H_0^2(\Omega)}} \\ &= \frac{|r| \int_\Omega \xi \, d\Omega}{\|\xi\|_{H_0^2(\Omega)}} \\ &= \|\xi\|_{H_0^2(\Omega)} |r|, \end{aligned} \tag{1.62}$$

after using (1.61). This gives (1.59), with inf-sup constant $\beta = \|\xi\|_{H_0^2(\Omega)}$. The existence of a unique pair $[w_1, \tilde{c}]$ which solves (1.42) now follows from Theorem 2.

Note in particular that

$$w_1 \in \widehat{L}^2(\Omega)\,, \tag{1.63}$$

from the second equation of (1.42). In turn, we recover w_2, u, and p via

$$\begin{cases} w_2 \equiv \lambda w_1 - w_1^* \in H_0^2(\Omega) \cap \widehat{L}^2(\Omega), \\ u \equiv \tilde{f}(\lambda w_1 - w_1^*) + \tilde{\mu}(u^*) \in \mathbf{H}^2(\mathcal{O}) \cap \mathcal{H}_{\text{fluid}}, \\ p \equiv \tilde{\pi}(\lambda w_1 - w_1^*) + \tilde{q}(u^*) + \tilde{c} \in H^1(\mathcal{O}). \end{cases} \tag{1.64}$$

From (1.30) and (1.32),

$$\text{the variables } u \text{ and } p \text{ solve the Stokes system (1.54).} \tag{1.65}$$

Moreover, from (1.58) and (1.30) we have

$$\begin{aligned} &\lambda^2(P_\rho^{1/2} w_1, P_\rho^{1/2}\phi)_\Omega + (\Delta w_1, \Delta\phi)_\Omega + \lambda(u, \tilde{f}(\phi))_\mathcal{O} + (\nabla u, \nabla \tilde{f}(\phi))_\mathcal{O} - (p, \operatorname{div}(\tilde{f}(\phi)))_\mathcal{O} \\ &= \quad (u^*, \tilde{f}(\phi))_\mathcal{O} + (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega \qquad \forall \phi \in H_0^2(\Omega). \end{aligned}$$

Noting the regularity afforded by (1.64) we can integrate by parts and pass the adjoint of $P_\rho^{1/2}$ to achieve,

$$\begin{aligned} &\lambda^2(P_\rho w_1, \phi)_\Omega + (\Delta w_1, \Delta\phi)_\Omega + (\lambda u - \Delta u + \nabla p, \tilde{f}(\phi))_\mathcal{O} + \left\langle \frac{\partial u}{\partial \nu} - p\nu, \tilde{f}(\phi) \right\rangle_{\partial\mathcal{O}} \\ &= \quad (u^*, \tilde{f}(\phi))_\mathcal{O} + (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega \qquad \forall \phi \in H_0^2(\Omega). \end{aligned}$$

As variables u and p solve the Stokes system (1.54), we thus attain the relation

$$\lambda^2(P_\rho w_1, \phi)_\Omega + (\Delta w_1, \Delta\phi)_\Omega - (p|_\Omega, \phi)_\Omega = (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega, \qquad \forall \phi \in H_0^2(\Omega).$$

(We have also implicitly used (1.30) and the remark after (1.56).) In particular, this

holds true for $\phi \in \mathcal{D}(\Omega)$. Thus we have the distributional relation

$$(\lambda^2 P_\rho w_1 + \Delta^2 w_1 - p|_\Omega - P_\rho[\lambda w_1^* + w_2^*], \phi)_\Omega = 0 \quad \forall \phi \in \mathcal{D}(\Omega),$$

and so we infer that

$$w_1 \text{ satisfies (1.53).} \tag{1.66}$$

Subsequently, we infer by elliptic theory that, as required by the definition of the fluid-structure operator $\mathcal{A}_\rho : D(\mathcal{A}_\rho) \subset \mathbf{H}_\rho \to \mathbf{H}_\rho$,

$$w_1 \in \mathcal{S}_\rho, \tag{1.67}$$

where the (displacement) space is as given in (1.26). Finally, because u and $u^* \in \mathcal{H}_{\text{fluid}}$, we have *a fortiori* from (1.54),

$$\Delta p = 0 \text{ in } \mathcal{O} \text{ and } \frac{\partial p}{\partial \nu}\Big|_S = \Delta u \cdot \nu\big|_S \text{ on } S. \tag{1.68}$$

Moreover, from (1.64) and (1.53), $\lambda w_2 + P_\rho^{-1}\Delta^2 w_1 - P_\rho^{-1} p\big|_\Omega = w_2^*$ which implies that in Ω, (since $[w_1^*, w_2^*, u^*] \in \mathbf{H}_\rho$)

$$\begin{aligned} P_\rho^{-1}\Delta^2 w_1 &= P_\rho^{-1} p\big|_\Omega - \lambda w_2 + w_2^* \\ &= P_\rho^{-1} p\big|_\Omega - \lambda u \cdot \nu\big|_\Omega + u^* \cdot \nu\big|_\Omega \\ &= P_\rho^{-1} p\big|_\Omega - \Delta u \cdot \nu\big|_\Omega + \nabla p \cdot \nu\big|_\Omega \end{aligned} \tag{1.69}$$

where in the last equality, (1.54) was again invoked. Thus, from (1.68) and (1.69), we

have that the pressure variable p, obtained by Theorem 2, solves

$$\begin{cases} \Delta p = 0 & \text{in } \mathcal{O}, \\ \frac{\partial p}{\partial \nu} + P_\rho^{-1} p = P_\rho^{-1}\Delta^2 w_1 + \Delta u^{(3)}\big|_\Omega & \text{in } \Omega, \\ \frac{\partial p}{\partial \nu} = \Delta u \cdot \nu\big|_S & \text{on } S. \end{cases}$$

As such,

$$p = G_{\rho,1}(w_1) + G_{\rho,2}(u), \tag{1.70}$$

where $G_{\rho,i}$ are as given in (1.23) and (1.24). (Note that we are implicitly using the critical regularity

$$\Delta u \cdot \nu\big|_{\partial\mathcal{O}} \in H^{-\frac{1}{2}}(\partial\mathcal{O}), \tag{1.71}$$

from (1.28).)

From (1.63), (1.64), (1.65), (1.66), (1.67), and (1.70), we have that the constructed variables $[w_1, w_2, u]$ belong to $D(\mathcal{A})$ and solve the resolvent equation (1.52). This concludes the proof of Theorem 1. $\square$

1.4 $\lambda = 0$ is in the Resolvent Set $\rho(\mathcal{A}_\rho)$

In Theorem 1 we were able to show that every positive real number λ is in the resolvent set of $\mathcal{A}_\rho$, that is the set of $\lambda \in \mathbb{C}$ for which $\lambda I - \mathcal{A}_\rho$ is boundedly invertible on the finite energy space $\mathbf{H}_\rho$. In addition to this, we can show the following result directly. Note that the primary reason this proof is different from that in the previous section is that except for the constant in the pressure term the fluid is uncoupled from the plate dynamics and can be solved immediately. (The boundary term $[0, 0, \varpi_2] = [0, 0, \varpi_1^*]$ is known.) For this reason, Theorem 2 is not invoked in this section. The interested reader can find additional spectral results as well as the implications thereof on the decay of energy in [2].

Theorem 3. *The operator $\mathcal{A}_\rho$ is boundedly invertible on $\mathbf{H}_\rho$; that is zero is in the resolvent set of $\mathcal{A}_\rho$.*

Before proving the theorem we first need a simple Proposition, in the same style as Lemma 2 of [3].

Proposition 4. *Suppose a function $\mu \in \mathbf{L}^2(\mathcal{O})$ and pair $(\varrho, h) \in H^1(\mathcal{O}) \times \mathbf{L}^2(\mathcal{O})$ satisfy the relation*

$$-\Delta\mu + \nabla\varrho = h, \tag{1.72}$$

where $\operatorname{div}(\mu) = \operatorname{div}(h) = 0$. *Then* $\Delta\mu \cdot \nu\big|_{\partial\mathcal{O}} \in \mathbf{H}^{-\frac{1}{2}}(\partial\mathcal{O})$, *with the estimate*

$$\left\|\Delta\mu \cdot \nu\big|_{\partial\mathcal{O}}\right\|_{\mathbf{H}^{-\frac{1}{2}}(\partial\mathcal{O})} \leq C\left[\|\varrho\|_{H^1(\mathcal{O})} + \|h\|_{\mathbf{L}^2(\mathcal{O})}\right]. \tag{1.73}$$

Proof of Proposition 4. Since μ and h are each divergence free, the H^1-function ϱ is harmonic. Consequently, we have by elliptic theory the estimate

$$\left\|\frac{\partial\varrho}{\partial\nu}\right\|_{H^{-\frac{1}{2}}(\partial\mathcal{O})} \leq C\,\|\varrho\|_{H^1(\mathcal{O})}. \tag{1.74}$$

(see e.g., p. 71, Theorem 3.8.1 of [27]). In turn, we can take the dot product of both sides of (1.72) with respect to the unit normal vector, and subsequently take the boundary trace so as to have

$$\begin{aligned}\left\|\Delta\mu \cdot \nu\big|_{\partial\mathcal{O}}\right\|_{\mathbf{H}^{-\frac{1}{2}}(\partial\mathcal{O})} &\leq C\left[\left\|\frac{\partial\varrho}{\partial\nu}\right\|_{H^{-\frac{1}{2}}(\partial\mathcal{O})} + \left\|h\cdot\nu\big|_{\partial\mathcal{O}}\right\|_{H^{-\frac{1}{2}}(\mathcal{O})}\right]\\ &\leq C\left[\|\varrho\|_{H^1(\mathcal{O})} + \|h\|_{\mathbf{L}^2(\mathcal{O})}\right]\end{aligned}$$

(recall that since $\operatorname{div}(h) = 0$, then $h\cdot\nu\big|_{\partial\mathcal{O}}$ is well-defined in $H^{-\frac{1}{2}}(\partial\mathcal{O})$; (see e.g., [34]). This concludes the proof of Proposition 4. $\square$

Proof of Theorem 3.

Let data $[\varpi_1^*, \varpi_2^*, \upsilon^*] \in \mathbf{H}_\rho$ be given. We look for $[\varpi_1, \varpi_2, \upsilon] \in D(\mathcal{A}_\rho)$ which solves

$$\mathcal{A}_\rho \begin{bmatrix} \varpi_1 \\ \varpi_2 \\ \upsilon \end{bmatrix} = \begin{bmatrix} \varpi_1^* \\ \varpi_2^* \\ \upsilon^* \end{bmatrix}. \tag{1.75}$$

To this end, we must search for $[\varpi_1, \varpi_2, \upsilon]$ in $D(\mathcal{A}_\rho)$ and $\pi_0 \in H^1(\mathcal{O})$ which solve

$$\varpi_2 = \varpi_1^* \ \text{ in } \Omega; \tag{1.76}$$

$$P_\rho^{-1}\Delta^2 \varpi_1 - P_\rho^{-1}\pi_0\big|_\Omega = -\varpi_2^* \ \text{ in } \Omega; \tag{1.77}$$

$$\varpi_1 = \frac{\partial \varpi_1}{\partial n} = 0 \ \text{ on } \partial\Omega; \tag{1.78}$$

$$\Delta \upsilon - \nabla \pi_0 = \upsilon^* \ \text{ in } \mathcal{O}; \tag{1.79}$$

$$\text{div}(\upsilon) = 0 \ \text{ in } \mathcal{O}; \tag{1.80}$$

$$\upsilon|_\Omega = [0, 0, \varpi_2] \ \text{ on } \Omega; \ \ \upsilon\big|_S = 0 \ \text{ on } S. \tag{1.81}$$

Moreover, we must justify that the pressure variable π_0 in this system is given by the expression

$$\pi_0 = G_{\rho,1}(\varpi_1) + G_{\rho,2}(\upsilon), \tag{1.82}$$

where the $G_{\rho,i}$ are given by (1.23) and (1.24), respectively (in line with the explicit form of $\mathcal{A}_\rho$ in (1.25)-(1.27)).

(1) The Plate Velocity. From (1.76), the velocity component ϖ_2 is immediately resolved.

(2) The Fluid Velocity. We next consider the Stokes system (1.79)-(1.81). From (1.76) and (1.81) we have $\upsilon\big|_{\partial\mathcal{O}}$ satisfies

$$\int_{\partial\mathcal{O}} \upsilon \cdot \nu d\partial\mathcal{O} = \int_\Omega \begin{bmatrix} 0 \\ 0 \\ \upsilon^3 \end{bmatrix} \cdot \nu d\Omega = \int_\Omega \varpi_2 d\Omega = \int_\Omega \varpi_1^* d\Omega = 0 \tag{1.83}$$

(as $[\varpi_1^*, \varpi_2^*, \upsilon^*] \in \mathbf{H}_\rho$). Since this compatibility condition is satisfied and data $\{\upsilon^*, \varpi_1^*\} \in \mathbf{L}^2(\mathcal{O}) \times H_0^2(\Omega)$, we can find a unique (fluid and pressure) pair $(\upsilon, q_0) \in [\mathbf{H}^2(\mathcal{O}) \cap \mathcal{H}_{\text{fluid}}] \times \frac{\mathbf{H}^1(\mathcal{O})}{\mathbb{R}}$, which solve

$$\Delta \upsilon - \nabla q_0 = \upsilon^* \text{ in } \mathcal{O}; \tag{1.84}$$

$$\text{div}(\upsilon) = 0 \text{ in } \mathcal{O}; \tag{1.85}$$

$$\upsilon = [0, 0, \varpi_1^*] \text{ on } \Omega; \ \upsilon = \vec{0} \text{ on } S. \tag{1.86}$$

In fact, via the mappings/estimates in (1.30)-(1.31) and (1.32)-(1.33) we have

$$\begin{aligned} &\upsilon = \tilde{f}(\varpi_1^*) + \tilde{\mu}(\upsilon^*); \\ &q_0 = \tilde{\pi}(\varpi_1^*) + \tilde{q}(\upsilon^*); \\ &\|\upsilon\|_{\mathbf{H}^2(\mathcal{O}) \cap \mathcal{H}_{\text{fluid}}} + \|q_0\|_{\frac{\mathbf{H}^1(\mathcal{O})}{\mathbb{R}}} \leq C \left[\|\upsilon^*\|_{\mathcal{H}_{\text{fluid}}} + \|\varpi_1^*\|_{H_0^2(\Omega)} \right]. \end{aligned} \tag{1.87}$$

(3) The Mechanical Displacement. Subsequently, with the obtained pressure variable q_0 in (1.84), we consider the plate component BVP (1.77)-(1.78). By ellipticity and elliptic regularity - see e.g., [31] - there exists a solution $\hat{\varpi}_1 \in \mathcal{S}_\rho$ to the system

$$\begin{aligned} \Delta^2 \hat{\varpi}_1 &= q_0|_\Omega - P_\rho \varpi_2^* \text{ in } \Omega; \\ \hat{\varpi}_1 &= \frac{\partial \hat{\varpi}_1}{\partial \nu} = 0 \text{ in } \partial\Omega \end{aligned} \tag{1.88}$$

(here the space $\mathcal{S}_\rho$ is as in (1.26)). Moreover, we have estimate

$$\begin{aligned} \|\hat{\varpi}_1\|_{\mathcal{S}_\rho} &\leq C \left(\left\| q_0|_\Omega \right\|_{H^{\frac{1}{2}}(\Omega)} + \|P_\rho \varpi_2^*\|_{W_\rho^{-1}(\Omega)} \right) \\ &\leq C \left\| [\varpi_1^*, \varpi_2^*, \upsilon^*] \right\|_{\mathbf{H}_\rho}, \end{aligned} \tag{1.89}$$

where

$$W_\rho^{-1}(\Omega) = \begin{cases} L^2(\Omega), & \text{if } \rho = 0 \\ H^{-1}(\Omega), & \text{if } \rho > 0 \end{cases}.$$

(By way of obtaining the last inequality in (1.89) we have also invoked Sobolev Trace Theory, (1.87) and the containment $P_\rho \in \mathcal{L}(W_\rho, W_\rho^{-1})$.) Now if, as in [14], we let Π denote the orthogonal projection of $H_0^2(\Omega)$ onto $H_0^2(\Omega)/\mathbb{R}$ - orthogonal with respect to the inner product $[\omega_1, \omega_2] \to (\Delta\omega_1, \Delta\omega_2)_\Omega$ - then one can readily show that its orthogonal complement $I - \Pi$ can be characterized as

$$\begin{aligned} &(I - \Pi)H_0^2(\Omega) = \text{Span}\{\xi\}, \\ &\quad \text{where } \left\{\Delta^2\xi = 1 \text{ in } \Omega, \ \ \xi\big|_{\partial\Omega} = \tfrac{\partial\xi}{\partial n} = 0 \text{ on } \partial\Omega\right\}. \end{aligned} \tag{1.90}$$

(see Remark 2.1, p. 6, of [14]). With these projections, we then set

$$\begin{aligned} \varpi_1 &= \Pi\hat{\varpi}_1; \\ \pi_0 &= q_0 - \Delta^2(I - \Pi)\hat{\varpi}_1. \end{aligned} \tag{1.91}$$

With this assignment of variables, by (1.88) and $\hat{\varpi}_1 = \Pi\hat{\varpi}_1 + (I - \Pi)\hat{\varpi}_1$, we will have that the newly obtained ϖ_1 solves (1.77)-(1.78). (And of course since π_0 and q_0 differ only by a constant, then the pair (υ, π_0) also solves (1.79)-(1.81).) Moreover, from elliptic theory, (1.87) and (1.89), we have the estimate

$$\begin{aligned} \|\varpi_1\|_{\mathcal{S}_\rho} + \|\pi_0\|_{H^1(\mathcal{O})} &\leq C\left(\left\|\Delta^2(I - \Pi)\hat{\varpi}_1\right\|_{L^2(\Omega)} + \|q_0\|_{\frac{H^1(\mathcal{O})}{\mathbb{R}}} + \|P_\rho\varpi_2^*\|_{W_\rho^{-1}(\Omega)}\right) \\ &\leq C\,\|[\varpi_1^*, \varpi_2^*, \upsilon^*]\|_{\mathbf{H}_\rho} \end{aligned} \tag{1.92}$$

(implicity we are also using the fact that $\Delta^2(I - \Pi) \in \mathcal{L}(H_0^2(\Omega), \mathbb{R})$, by the Closed Graph Theorem.)

(4) Resolution of the Pressure. At this point we invoke Proposition 4 and (1.87) to have

the following trace regularity for fluid velocity of (1.84)-(1.86):

$$\begin{aligned} \| \Delta \upsilon \cdot \nu|_{\Omega} \|_{H^{-\frac{1}{2}}(\partial \mathcal{O})} &\leq C \left[\|q_0\|_{H^1(\mathcal{O})} + \|\upsilon^*\|_{\mathbf{L}^2(\mathcal{O})} \right] \\ &\leq C \left[\|\upsilon^*\|_{\mathcal{H}_{\text{fluid}}} + \|\varpi_1^*\|_{H_0^2(\Omega)} \right]. \end{aligned} \tag{1.93}$$

Consequently we have that the pressure variable π_0 of (1.76)-(1.81) - given explicitly in (1.91) - solves *a fortiori*

$$\Delta \pi_0 = 0 \ \text{ in } \mathcal{O} \tag{1.94}$$

$$\frac{\partial \pi_0}{\partial \nu} + P_{\rho}^{-1} \pi_0 = P_{\rho}^{-1} \Delta^2 \varpi_1 + \Delta \upsilon^3 \big|_{\Omega} \ \text{ on } \Omega \tag{1.95}$$

$$\frac{\partial \pi_0}{\partial \nu} = \Delta \upsilon \cdot \nu \big|_S \ \text{ on } S. \tag{1.96}$$

In fact: Applying the divergence operator to both sides of (1.79) and using $\text{div}(\upsilon) = \text{div}(\upsilon^*) = 0$, we have (1.94). Moreover, dotting both sides of (1.79) with respect to the normal vector, and subsequently taking the boundary trace to the portion S, we have then (1.96) (implicitly we are also using $\upsilon^* \cdot \nu \big|_S = 0$, as $[\varpi_1^*, \varpi_2^*, \upsilon^*] \in \mathbf{H}_{\rho}$). Finally, as $\upsilon^* \cdot \nu \big|_{\Omega} = \varpi_2^*$, as $[\varpi_1^*, \varpi_2^*, \upsilon^*] \in \mathbf{H}_{\rho}$, we have from (1.77)

$$\begin{aligned} P_{\gamma}^{-1} \pi_0 \big|_{\Omega} &= \varpi_2^* + P_{\rho}^{-1} \Delta^2 \varpi_1 \\ &= \Delta \upsilon \cdot \nu \big|_{\Omega} - \nabla \pi_0 \cdot \nu \big|_{\Omega} + P_{\rho}^{-1} \Delta^2 \varpi_1, \end{aligned}$$

which gives the boundary condition (1.95). Necessarily then, the constructed pressure term must be given by the expression

$$\pi_0 = G_{\rho,1}(\varpi_1) + G_{\rho,2}(\upsilon) \in H^1(\mathcal{O}). \tag{1.97}$$

(where the right hand side is well-defined by (1.93)).

Finally, we collect: (i) (1.84)-(1.86) and (1.87) - for the fluid variable υ; (ii) (1.88), (1.76),

(1.89), (1.91) and (1.92) - for the respective structure and pressure variables $[\varpi_1, \varpi_2]$ and π_0; (iii) (1.93), (1.92) and (1.97)- for the characterization of the pressure term π_0. In this way we have obtained the solution of (1.76)-(1.82) in $D(\mathcal{A}_\rho)$, for arbitrary $\mathbf{H}_\rho$, thus establishing the maximality of $\mathcal{A}_\rho$ and concluding the proof. $\square$

1.5 Backward Uniqueness

In this section we show that the PDE model in (1.2)-(1.6) satisfies the backward uniqueness property, that is the solution to the system with non-trivial initial data cannot evolve to the trivial solution; specifically $e^{\mathcal{A}_\rho T}\Phi = \vec{0} \Rightarrow \Phi = \vec{0}$. This property has implications in the area of PDE control theory, but is also interesting in its own right. See for example [4] where a different fluid-structure interaction is studied. In that paper the authors write: "Backward uniqueness is trivial for s.c. *groups*; simple to prove for s.c. *analytic* semigroups; and it may be patently false, as is the case of nilpotent semigroups. The latter case may very well arise from physically significant models." Establishing the backward uniqueness property is very much at home in the case of *thermoelastic* plate systems, see e.g. [30] and [35]; the models therein share features with the system of interest here. The desire to solve such systems motivated the abstract Theorem 5 used below.

Consider again the PDE model in (1.2)-(1.6) with "rotational inertia parameter" $\rho \geq 0$, and solution variables $w(x,t)$, $u(x,t) = [u^{(1)}(x,t), u^{(2)}(x,t), u^{(3)}(x,t)]$, and $p(x,t)$. The initial conditions are given as in (1.7) by

$$[w(0), w_t(0), u(0)] = [w_1^*, w_2^*, u_0^*] \in \mathbf{H}_\rho. \tag{1.98}$$

The main result here deals with establishing the so-called backward uniqueness property for the contraction C_0-semigroup associated with the PDE model (1.2)-(1.7). To this end, we will make use of the following abstract resolvent criterion for backward uniqueness.

Theorem 5. *(See [30], Theorem 3.1, p. 225.) Let A be the infinitesimal generator of a strongly continuous semigroup e^{At} in a Banach space X. Assume that there exist constants $a \in (\pi/2, \pi)$, $r_0 > 0$, and $C > 0$, such that*

$$\left\|\mathcal{R}(re^{\pm ia}; A)\right\|_{\mathcal{L}(X)} \equiv \left\|(re^{\pm ia}I - A)^{-1}\right\|_{\mathcal{L}(X)} \leq C,$$

for all $r \geq r_0$. Then the backward uniqueness property holds true; that is, $e^{AT}x_0 = 0$ for $T > 0$, $x_0 \in X$, implies $x_0 = 0$.

By way of applying Theorem 5 to the problem at hand, we consider the following resolvent relation with complex parameter $\lambda = \alpha + i\beta$, which is formally a "frequency domain" version of (1.2)-(1.7): With pre-image $[\omega_1, \omega_2, \mu] \in D(\mathcal{A}_\rho)$ and associated pressure variable $p \in H^1(\mathcal{O})$, we consider the following fluid-structure model with data $[\omega_1^*, \omega_2^*, \mu^*] \in \mathbf{H}_\rho$:

$$(\lambda I - \mathcal{A}_\rho) \begin{bmatrix} \omega_1 \\ \omega_2 \\ \mu \end{bmatrix} = \begin{bmatrix} \omega_1^* \\ \omega_2^* \\ \mu^* \end{bmatrix} \in \mathbf{H}_\rho. \tag{1.99}$$

With $\lambda = \alpha + i\beta$, and the definition of the domain $D(\mathcal{A}_\rho)$, we have then the fluid-structure PDE system,

$$\begin{cases} \omega_2 = \lambda\omega_1 - \omega_1^* \ \text{ in } \Omega \\ (\alpha^2 - \beta^2)\,\omega_1 + 2i\alpha\beta\omega_1 + P_\rho^{-1}\Delta^2\omega_1 - P_\rho^{-1}p\big|_\Omega = \omega_2^* + \lambda\omega_1^* \ \text{ in } \Omega \\ \omega_1|_{\partial\Omega} = \frac{\partial\omega_1}{\partial n}\big|_{\partial\Omega} = 0 \ \text{ on } \partial\Omega \end{cases} \tag{1.100}$$

$$\begin{cases} \lambda\mu - \Delta\mu + \nabla p = \mu^* \ \text{ in } \mathcal{O} \\ \operatorname{div}(\mu) = 0 \ \text{ in } \mathcal{O} \\ \mu = 0 \text{ on } S; \ \ \mu = [0, 0, \lambda\omega_1 - \omega_1^*] \ \text{ in } \Omega. \end{cases} \tag{1.101}$$

At this point, we will henceforth impose that $\lambda = \alpha + i\beta$ should obey the following

criteria:

Criterion 1: $\lambda = \alpha + i\beta = |\lambda|\, e^{\pm i\vartheta}$, for fixed $\vartheta \in \left(\frac{3\pi}{4}, \pi\right)$. (And so on either of these two rays, we have $0 < |\tan\vartheta| < 1$, $|\beta| = |\alpha|\,|\tan\vartheta|$, $|\lambda|^2 = \alpha^2 + \beta^2 = \alpha^2(1 + \tan^2\vartheta)$.)

Criterion 2: $|\alpha| > 0$ is sufficiently large.

Our main result can now be stated as follows:

Theorem 6. *(i) With respect to the resolvent relation (1.99), or the equivalent fluid-structure PDE (1.100)-(1.101), assume both Criteria 1 and 2 are satisfied. Then for all $\rho \geq 0$, the solution $[\omega_1, \omega_2, \mu] \in D(\mathcal{A}_\rho)$ obeys the following bound, which is uniform for all $\lambda = \alpha + i\beta = |\lambda|\, e^{\pm i\vartheta}$, with fixed $\vartheta \in \left(\frac{3\pi}{4}, \pi\right)$, and $|\alpha| > 0$ sufficiently large:*

$$\left\| \begin{bmatrix} \omega_1 \\ \omega_2 \\ \mu \end{bmatrix} \right\|^2_{\mathbf{H}_\rho} \leq C_\vartheta \left\| \begin{bmatrix} \omega_1^* \\ \omega_2^* \\ \mu^* \end{bmatrix} \right\|^2_{\mathbf{H}_\rho}. \tag{1.102}$$

(ii) In consequence, this estimate and Theorem 5 gives the inference that the fluid-structure C_0-contraction semigroup $\left\{e^{\mathcal{A}_\rho t}\right\}_{t\geq 0}$ satisfies the backward uniqueness property: Namely, if for given $T > 0$ and $[w_1^*, w_2^*, u_0^*] \in \mathbf{H}_\rho$, one has

$$e^{\mathcal{A}_\rho T} \begin{bmatrix} w_1^* \\ w_2^* \\ u_0^* \end{bmatrix} = \vec{0}, \text{ then necessarily } \begin{bmatrix} w_1^* \\ w_2^* \\ u_0^* \end{bmatrix} = \vec{0}.$$

Remark 7. *Unlike the coupled PDE examples in [30], [4], [5], we have not derived here a rate of decay, which is uniform with respect to $Re(\lambda)$, for the fluid-structure solution $[\omega_1, \omega_2, \mu]$ of (1.99). We are not certain that showing such a decay is actually possible. However, by Theorem 6 the uniform bound (1.102) suffices to establish the aforesaid backward uniquess property.*

Remark 8. *In the course of proof, the reader should surmise that for the rotational inertial case $\rho > 0$, one will in fact have the uniform estimate (1.102) for any rays along the angle $\vartheta \in \left(\frac{\pi}{2}, \pi\right)$. In the $\rho = 0$ case, the proof requires that $1 - \tan^2 \vartheta \neq 0$.*

Before moving on to the proof of the main result we state here a few propositions that are needed throughout the proof.

Proposition 9 (Young's Inequality)**.** *For given $p, q \geq 1$ satisfying $1/p + 1/q = 1$, $\epsilon > 0$ and $a, b > 0$ we have*

$$ab \leq \frac{(\epsilon a)^p}{p} + \frac{1}{q}\left(\frac{b}{\epsilon}\right)^q.$$

Thus for small $\epsilon > 0$ there is a (possibly large) constant C_ϵ such that

$$ab \leq \epsilon a^p + C_\epsilon b^q.$$

The following proposition provides the machinery of interpolation between Hilbert spaces. See for example [25] for a proof.

Proposition 10 (Interpolation)**.** *Given $s_1 > s_2$, and $0 < \theta < 1$, then for $w \in H^{(1-\theta)s_1 + \theta s_2}(\Omega)$ we have*

$$\|w\|_{(1-\theta)s_1+\theta s_2} \leq C\|w\|_{s_1}^{1-\theta}\|w\|_{s_2}^{\theta},$$

where $\|\cdot\|_s$ indicates the norm on the Hilbert space H^s.

In the proof that follows we will also have the need of the following boundary trace inequality (see e.g., Theorem 1.6.6 of [10], p. 37).

Proposition 11 (Sobolev Boundary Trace Inequality)**.** — *Let D be a smooth domain of $\mathbb{R}^n$, $n \geq 2$, with sufficiently smooth boundary ∂D. Then there is a positive constant C*

such that

$$\|f\|_{\partial D} \leq C \|f\|_D^{\frac{1}{2}} \|f\|_{H^1(D)}^{\frac{1}{2}} \text{ for every } f \in H^1(D). \tag{1.103}$$

$$\left\| \frac{\partial f}{\partial \nu} \right\|_{\partial D} \leq C \|f\|_{H^1(D)}^{\frac{1}{2}} \|f\|_{H^2(D)}^{\frac{1}{2}} \text{ for every } f \in H^2(D). \tag{1.104}$$

The proof of Theorem 6 will ultimately depend on the appropriate use of four basic relations:

(i) Taking the $D(P_\rho^{\frac{1}{2}})$-inner product of both sides of the structural PDE in (1.100) with ω_1, intergrating by parts and subsequently taking the real part of the result, we have

$$\alpha^2(1-\tan^2\vartheta)\left\|P_\rho^{\frac{1}{2}}\omega_1\right\|_{L^2(\Omega)}^2 + \|\Delta\omega_1\|_{L^2(\Omega)}^2 = \mathrm{Re}\left(p|_\Omega, \omega_1\right)_\Omega + \mathrm{Re}\left(\omega_2^* + \lambda\omega_1^*, \omega_1\right)_\Omega, \tag{1.105}$$

(after also using implicitly the Criterion 1).

(ii) We take the $\mathbf{L}^2(\mathcal{O})$-inner product of both sides of the fluid PDE in (1.101) with μ. After integrating by parts and then taking the imaginary and real parts respectively of the resulting relation we have for $|\alpha| > 0$,

$$\beta\|\mu\|_{\mathcal{O}}^2 = -\mathrm{Im}\left(p|_\Omega, \lambda\omega_1 - \omega_1^*\right)_\Omega + \mathrm{Im}\left(\mu^*, \mu\right)_{\mathcal{O}}; \tag{1.106}$$

$$\alpha\|\mu\|_{\mathcal{O}}^2 + \|\nabla\mu\|_{\mathcal{O}}^2 = -\mathrm{Re}\left(p|_\Omega, \lambda\omega_1 - \omega_1^*\right)_\Omega + \mathrm{Re}\left(\mu^*, \mu\right)_{\mathcal{O}}. \tag{1.107}$$

(iii) Lastly, we take the $\mathbf{H}_\rho$-inner product of both sides of the resolvent equation (1.99) with respect to solution variables $[\omega_1, \omega_2, \mu]$. This gives, upon integrating and taking the real part of the results:

$$\alpha\left\|\begin{bmatrix}\omega_1\\ \omega_2\\ \mu\end{bmatrix}\right\|_{\mathbf{H}_\rho}^2 = \|\nabla\mu\|_{\mathcal{O}}^2 + \mathrm{Re}\left(\begin{bmatrix}\omega_1^*\\ \omega_2^*\\ \mu^*\end{bmatrix}, \begin{bmatrix}\omega_1\\ \omega_2\\ \mu\end{bmatrix}\right)_{\mathbf{H}_\rho}. \tag{1.108}$$

(See the argument leading to (1.46) for details.)

In view of the right hand side of the relations (1.105) and (1.106), it is evidently necessary to scrutinize the "interface" term $(p|_\Omega\,,\omega_1)_\Omega$. Indeed, the estimation of this term will constitute the bulk of the effort in this work. By way of attaining a useful estimation, we will need to consider the explicit representation of the pressure term. Recall from (1.22) that the pressure is given by

$$p = G_{\rho,1}(\omega_1) + G_{\rho,2}(\mu). \tag{1.109}$$

Therefore,

$$(p|_\Omega\,,\omega_1)_\Omega = \left(G_{\rho,1}(\omega_1)|_\Omega\,,\omega_1\right)_\Omega + \left(G_{\rho,2}(\mu)|_\Omega\,,\omega_1\right)_\Omega. \tag{1.110}$$

We will proceed now to estimate each inner product on the right hand side of (1.110).

1.5.1 Analysis of the Term $\left(G_{\rho,2}(\mu)|_\Omega\,,\omega_1\right)_\Omega$

We recall from (1.24) that

$$G_{\rho,2}(\mu) = R_\rho(\,\Delta\mu^3\big|_\Omega) + \tilde{R}_p(\,\Delta\mu\cdot\nu|_S). \tag{1.111}$$

With the right hand side of (1.111) in mind we define the positive, self-adjoint operator $B_\rho : D(B_\rho) \subset L^2(\mathcal{O}) \to L^2(\mathcal{O})$ by

$$B_\rho f = -\Delta f \ \text{ in } \mathcal{O};\ D(B_\rho) = \left\{ f \in H^2(\mathcal{O}) : \frac{\partial f}{\partial \nu} + P_\rho^{-1} f = 0 \ \text{ on } \Omega;\ \frac{\partial f}{\partial \nu} = 0 \ \text{ on } S \right\}. \tag{1.112}$$

Therewith one can can readily compute the respective adjoints of $R_\rho \in \mathcal{L}(H^{-\frac{1}{2}}(\Omega), H^1(\mathcal{O}))$, $\tilde{R}_\rho \in \mathcal{L}(H^{-\frac{1}{2}}(S), H^1(\mathcal{O}))$, $B_\rho R_\rho \in \mathcal{L}(H^{-\frac{1}{2}}(\Omega), [H^1(\mathcal{O})]')$ and

$B_\rho \tilde{R}_\rho \in \mathcal{L}(H^{-\frac{1}{2}}(S), [H^1(\mathcal{O})]')$, as,

$$R_\rho^* f = B_\rho^{-1} f\big|_\Omega \text{ for all } f \in [H^1(\mathcal{O})]'; \tag{1.113}$$

$$\tilde{R}_\rho^* f = B_\rho^{-1} f\big|_S \text{ for all } f \in [H^1(\mathcal{O})]'; \tag{1.114}$$

$$R_\rho^* B_\rho f = f|_\Omega \text{ for all } f \in [H^1(\mathcal{O})]; \tag{1.115}$$

$$\tilde{R}_\rho^* B_\rho f = f|_S \text{ for all } f \in [H^1(\mathcal{O})] \tag{1.116}$$

Indeed, to show (1.113): Given $g \in H^{\frac{1}{2}}(\Omega)$ and $f \in [H^1(\mathcal{O})]'$, we have

$$\begin{aligned}
(R_\rho g, f)_\mathcal{O} &= (R_\rho g, (-\Delta) B_\rho^{-1} f)_\mathcal{O} \\
&= (\nabla R_\rho g, \nabla B_\rho^{-1} f)_\mathcal{O} - (R_\rho g, \frac{\partial}{\partial \nu} B_\rho^{-1} f)_\Omega + 0 \\
&= (-\Delta R_\rho g, B_\rho^{-1} f)_\mathcal{O} + (\frac{\partial}{\partial \nu} R_\rho g, B_\rho^{-1} f)_\Omega + (R_\rho g, P_\rho^{-1} B_\rho^{-1} f)_\Omega \\
&= (\frac{\partial}{\partial \nu} R_\rho g + P_\rho^{-1} R_\rho g, B_\rho^{-1} f)_\Omega \\
&= (g, B_\rho^{-1} f)_\Omega.
\end{aligned}$$

For (1.114): Let $g \in H^{-\frac{1}{2}}(S)$ and $f \in [H^1(\mathcal{O})]'$ be given; then

$$\begin{aligned}
(\tilde{R}_\rho g, f)_\mathcal{O} &= (\tilde{R}_\rho g, (-\Delta) B_\rho^{-1} f)_\mathcal{O} \\
&= (\nabla \tilde{R}_\rho g, \nabla B_\rho^{-1} f)_\mathcal{O} - (\tilde{R}_\rho g, \frac{\partial}{\partial \nu} B_\rho^{-1} f)_\Omega + 0 \\
&= (-\Delta \tilde{R}_\rho g, B_\rho^{-1} f)_\mathcal{O} + (\frac{\partial}{\partial \nu} \tilde{R}_\rho g, B_\rho^{-1} f)_{\partial\mathcal{O}} + (\tilde{R}_\rho g, P_\rho^{-1} B_\rho^{-1} f)_\Omega \\
&= (g, B_\rho^{-1} f)_S - (P_\rho^{-1} \tilde{R}_\rho g, B_\rho^{-1} f)_\Omega + (\tilde{R}_\rho g, P_\rho^{-1} B_\rho^{-1} f)_\Omega \\
&= (g, B_\rho^{-1} f)_S.
\end{aligned}$$

The relations (1.115)-(1.116) follow from the now established (1.113) and (1.114) coupled with the fact that B_ρ is self-adjoint.

With the relations (1.113)-(1.116) in hand, we produce from (1.111) the following:

$$\begin{aligned}\left(G_{\rho,2}(\mu)|_{\Omega}\,,\omega_1\right)_{\Omega} &= \left(R_{\rho}^{*}B_{\rho}\left[R_{\rho}(\Delta\mu^3\big|_{\Omega}) + \tilde{R}_{p}(\Delta\mu\cdot\nu|_{S})\right],\omega_1\right)_{\Omega}\\ &= \left(\left[R_{\rho}(\Delta\mu^3\big|_{\Omega}) + \tilde{R}_{p}(\Delta\mu\cdot\nu|_{S})\right],B_{\rho}R_{\rho}\omega_1\right)_{\mathcal{O}}\\ &= \left(\Delta\mu^3\big|_{\Omega}\,,R_{\rho}\omega_1|_{\Omega}\right)_{\Omega} + \left(\Delta\mu\cdot\nu|_{S}\,,R_{\rho}\omega_1|_{S}\right)_{S}\\ &= \left(\Delta\mu\cdot\nu,R_{\rho}\omega_1\right)_{\partial\mathcal{O}}.\end{aligned}$$

Invoking now Green's Formula – and simultaneously using the fact that the fluid term $\Delta\mu$ is divergence free – yields

$$\left(G_{\rho,2}(\mu)|_{\Omega}\,,\omega_1\right)_{\Omega} = \left(\Delta\mu,\nabla R_{\rho}\omega_1\right)_{\mathcal{O}}. \tag{1.117}$$

Following this up with Green's First Identity, we have then

$$\left(G_{\rho,2}(\mu)|_{\Omega}\,,\omega_1\right)_{\Omega} = -\left(\nabla\mu,\nabla(\nabla R_{\rho}\omega_1)\right)_{\mathcal{O}} + \left\langle\frac{\partial\mu}{\partial\nu},\,\nabla R_{\rho}\omega_1|_{\partial\mathcal{O}}\right\rangle_{\partial\mathcal{O}}. \tag{1.118}$$

With this representation in hand we will establish the following estimate.

Lemma 12. *For all $\rho\geq 0$,*

$$\begin{aligned}\left|\left(G_{\rho,2}(\mu)|_{\Omega}\,,\omega_1\right)_{\Omega}\right| &\leq C_{\epsilon}\left|\alpha\right|^{\frac{3}{2}+2\delta}\left\|\omega_1\right\|^2_{D(P_{\rho}^{\frac{1}{2}})} + \frac{\epsilon}{|\alpha|}\left\|\nabla\mu\right\|^2_{\mathcal{O}}\\ &\quad + \left(\epsilon + \frac{C_{\epsilon}}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\left(\,\left\|[\omega_1,\omega_2,\mu]\right\|^2_{\mathbf{H}_0} + \left\|[\omega_1^{*},\omega_2^{*},\mu^{*}]\right\|^2_{\mathbf{H}_0}\right).\end{aligned} \tag{1.119}$$

Proof of Lemma 12:

To handle the first term on the right hand side of (1.118): Using the regularity of the

map $R_\rho \in \mathcal{L}\left(H^{\frac{1}{2}}(\Omega), H^2(\mathcal{O})\right)$, followed by interpolation, we have

$$\begin{aligned}\left|(\nabla\mu, \nabla(\nabla R_\rho \omega_1))_{\mathcal{O}}\right| &\leq C \left\|\nabla\mu\right\|_{\mathcal{O}} \left\|\omega_1\right\|_{H^{\frac{1}{2}}(\Omega)} \\ &\leq C \left\|\nabla\mu\right\|_{\mathcal{O}} \left\|\omega_1\right\|_{\Omega}^{\frac{3}{4}} \left\|\omega_1\right\|_{H^2(\Omega)}^{\frac{1}{4}} \\ &= C\frac{\sqrt{|\alpha|}}{\sqrt{|\alpha|}} \left\|\nabla\mu\right\|_{\mathcal{O}} \left\|\omega_1\right\|_{\Omega}^{\frac{3}{4}} \left\|\omega_1\right\|_{H^2(\Omega)}^{\frac{1}{4}}.\end{aligned}$$

Applying Young's Inequality with $(p = 2, q = 2)$ and then $(p = \frac{4}{3}, q = 4)$ yields

$$\begin{aligned}\left|(\nabla\mu, \nabla(\nabla R_\rho \omega_1))_{\mathcal{O}}\right| &\leq \frac{\epsilon}{|\alpha|} \left\|\nabla\mu\right\|_{\mathcal{O}}^2 + C_\epsilon |\alpha| \left\|\omega_1\right\|_{\Omega}^{\frac{3}{2}} \left\|\omega_1\right\|_{H^2(\Omega)}^{\frac{1}{2}} \\ &\leq \frac{\epsilon}{|\alpha|} \left\|\nabla\mu\right\|_{\mathcal{O}}^2 + \epsilon \left\|\omega_1\right\|_{H^2(\Omega)}^2 + C_\epsilon |\alpha|^{\frac{4}{3}} \left\|\omega_1\right\|_{D(P_\rho^{\frac{1}{2}})}^2. \end{aligned} \tag{1.120}$$

The second term on the right hand side of (1.118) is a more delicate matter; in fact the analysis becomes a dichotomy with respect to $\rho = 0$ and $\rho > 0$. Both cases will make use of Proposition 11.

Estimating the term $\left|\left\langle \frac{\partial\mu}{\partial\nu}, \nabla R_\rho \omega_1|_{\partial\mathcal{O}} \right\rangle_{\partial\mathcal{O}}\right|$ **for** $\rho = 0$.

To start: We will need the following positive definite, self-adjoint operator $\mathring{\mathbf{A}} : D(\mathring{A}) \subset L^2(\Omega) \to L^2(\Omega)$, defined by

$$\mathring{\mathbf{A}}\varpi = \Delta^2\varpi, \;\; D(\mathring{\mathbf{A}}) = H^4(\Omega) \cap H_0^2(\Omega). \tag{1.121}$$

Then this operator obeys the following "analyticity" estimate for all $s > 0$:

$$\left\|\mathring{\mathbf{A}}^\eta \mathcal{R}(-s; \mathring{\mathbf{A}})\right\|_{\mathcal{L}(L^2(\Omega))} \leq \frac{C}{(1+s)^{1-\eta}}, \text{ for all } \eta \in [0, 1] \tag{1.122}$$

(see e.g., the expression (5.15) in [29], p. 115). With this operator in hand, then in the

present case $\rho = 0$ the structural equation in (1.105) can be written as

$$\left[\alpha^2(1-\tan^2\vartheta) + \mathring{\mathbf{A}}\right]\omega_1 = -2i\alpha\beta\omega_1 + p|_\Omega + \omega_2^* + \lambda\omega_1^*.$$

Applying thereto the operator $\mathring{\mathbf{A}}^\eta \mathcal{R}(-\alpha^2(1-\tan^2\vartheta); \mathring{\mathbf{A}})$ gives then

$$\mathring{\mathbf{A}}^\eta\omega_1 = \mathcal{R}(-\alpha^2(1-\tan^2\vartheta); \mathring{\mathbf{A}})\left[2i\alpha\beta\omega_1 - p|_\Omega - \omega_2^* - \lambda\omega_1^*\right].$$

Subsequently applying the estimate (1.122), we then have for $0 \leq \eta \leq 1$ and $|\alpha| > 0$ sufficiently large,

$$\begin{aligned}\left\|\mathring{\mathbf{A}}^\eta\omega_1\right\|_\Omega &\leq \frac{C}{(1+\alpha^2(1-\tan^2\vartheta))^{1-\eta}}\left[|\alpha\beta|\,\|\omega_1\|_\Omega + \|p|_\Omega\|_\Omega + \|\omega_2^* + \lambda\omega_1^*\|_\Omega\right] \\ &\leq C_\vartheta\,|\alpha|^{2\eta}\,\|\omega_1\|_\Omega + \frac{C_\vartheta}{|\alpha|^{1-2\eta}}\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right). \qquad (1.123)\end{aligned}$$

In obtaining this estimate, we have used $|\beta| = |\alpha|\,|\tan\vartheta|$, the resolvent equation (1.99), and the expression (1.109).

With (1.123) in hand, we now estimate the second term on the right hand side of (1.118): Using the trace inequality (1.104), the fact that $R_0 \in \mathcal{L}(H^\delta(\Omega), H^{\frac{3}{2}+2\delta}(\mathcal{O}))$, and the Sobolev Boundary Trace Theorem, we have for $|\alpha| > 0$ sufficiently large,

$$\begin{aligned}\left|\left\langle\frac{\partial\mu}{\partial\nu}, \nabla R_0\omega_1|_{\partial\mathcal{O}}\right\rangle_{\partial\mathcal{O}}\right| &\leq \left\|\frac{\partial\mu}{\partial\nu}\right\|_{\partial\mathcal{O}}\|\nabla R_0\omega_1|_{\partial\mathcal{O}}\|_{\partial\mathcal{O}} \\ &\leq C\left\|\frac{\partial\mu}{\partial\nu}\right\|_{\partial\mathcal{O}}\|R_0\omega_1|_{\partial\mathcal{O}}\|_{H^{\frac{3}{2}+2\delta}(\mathcal{O})} \\ &\leq C\,\|\nabla\mu\|_{\mathcal{O}}^{\frac{1}{2}}\,\|\mu\|_{\mathbf{H}^2(\mathcal{O})}^{\frac{1}{2}}\,\|\omega_1\|_{H^{2\delta}(\Omega)} \\ &\leq C\,\|\nabla\mu\|_{\mathcal{O}}^{\frac{1}{2}}\,\|[\omega_1,\omega_2,\mu]\|_{D(\mathcal{A}_0)}^{\frac{1}{2}}\,\|\omega_1\|_{H^{2\delta}(\Omega)} \\ &\leq C\sqrt{|\alpha|}\,\|\nabla\mu\|_{\mathcal{O}}^{\frac{1}{2}}\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right)^{\frac{1}{2}}\|\omega_1\|_{H^{2\delta}(\Omega)}, \qquad (1.124)\end{aligned}$$

where again we have implicitly used the resolvent relation (1.99). Using now the characterization

$$H^{2\delta}(\Omega) \approx D(\mathring{\mathbf{A}}^{\frac{\delta}{2}}) \text{ for } 0 < \delta < \frac{1}{4},$$

which can be inferred from the definition of the domain in (1.121) and [25], we have upon applying (1.123) to the right hand side of (1.124),

$$\begin{aligned}\left|\left\langle \frac{\partial \mu}{\partial \nu}, \nabla R_0 \omega_1|_{\partial\mathcal{O}} \right\rangle_{\partial\mathcal{O}}\right| &\leq C\sqrt{|\alpha|}\,\|\nabla\mu\|_{\mathcal{O}}^{\frac{1}{2}} \left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right)^{\frac{1}{2}} \times \\ &\qquad \left[|\alpha|^{\delta}\,\|\omega_1\|_{\Omega} + \frac{1}{|\alpha|^{1-\delta}}\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right)\right] \\ &\leq \frac{C}{|\alpha|^{\frac{1}{4}}}\,\|\nabla\mu\|_{\mathcal{O}}^{\frac{1}{2}}\,|\alpha|^{\frac{3}{4}+\delta}\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right)^{\frac{1}{2}} \times \\ &\qquad \left[\|\omega_1\|_{\Omega} + \frac{1}{|\alpha|}\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right)\right].\end{aligned}$$

Applying Young's Inequality twice with $(p=4, q=\frac{4}{3})$ and then $(p=3, q=\frac{3}{2})$ yields,

$$\begin{aligned}\left|\left\langle \frac{\partial \mu}{\partial \nu}, \nabla R_0 \omega_1|_{\partial\mathcal{O}} \right\rangle_{\partial\mathcal{O}}\right| &\leq \frac{\epsilon}{|\alpha|}\,\|\nabla\mu\|_{\mathcal{O}}^2 + C_\epsilon\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right)^{\frac{2}{3}} \times \\ &\qquad |\alpha|^{1+\frac{4\delta}{3}}\left[\|\omega_1\|_{\Omega} + \frac{1}{|\alpha|}\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}\right)\right]^{\frac{4}{3}} \\ &\leq \frac{\epsilon}{|\alpha|}\,\|\nabla\mu\|_{\mathcal{O}}^2 + \epsilon\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0}^2 + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}^2\right) \\ &\qquad + C_\epsilon\,|\alpha|^{\frac{3}{2}+2\delta}\,\|\omega_1\|_{\Omega}^2 + \frac{C_\epsilon}{|\alpha|^{\frac{1}{2}-2\delta}}\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0}^2 + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}^2\right).\end{aligned} \tag{1.125}$$

Estimating the term $\left|\left\langle \frac{\partial\mu}{\partial\nu}, \nabla R_\rho \omega_1|_{\partial\mathcal{O}}\right\rangle_{\partial\mathcal{O}}\right|$ for $\rho > 0$.

Using again the trace estimate (1.104), the fact that $R_\rho \in \mathcal{L}(H^{\frac{1}{2}}(\Omega), H^2(\mathcal{O}))$, and the

Sobolev Boundary Trace Theorem, we have

$$\begin{aligned}
\left|\left\langle \frac{\partial \mu}{\partial \nu}, \nabla R_\rho \omega_1|_{\partial\mathcal{O}} \right\rangle_{\partial\mathcal{O}}\right| &\leq \left\| \frac{\partial \mu}{\partial \nu} \right\|_{\partial\mathcal{O}} \left\| \nabla R_\rho \omega_1|_{\partial\mathcal{O}} \right\|_{\partial\mathcal{O}} \\
&\leq C \left\|\nabla \mu\right\|_{\mathcal{O}}^{\frac{1}{2}} \left\|\mu\right\|_{\mathbf{H}^2(\mathcal{O})}^{\frac{1}{2}} \left\|R_\rho \omega_1\right\|_{H^2(\mathcal{O})} \\
&\leq C \left\|\nabla \mu\right\|_{\mathcal{O}}^{\frac{1}{2}} \left\|\mu\right\|_{\mathbf{H}^2(\mathcal{O})}^{\frac{1}{2}} \left\|\omega_1\right\|_{H^{\frac{1}{2}}(\Omega)} \\
&\leq C \left\|\nabla \mu\right\|_{\mathcal{O}}^{\frac{1}{2}} \left\|\mu\right\|_{\mathbf{H}^2(\mathcal{O})}^{\frac{1}{2}} \left\|\omega_1\right\|_{\Omega}^{\frac{1}{2}} \left\|\omega_1\right\|_{H^1(\Omega)}^{\frac{1}{2}}.
\end{aligned}$$

Combining this with the resolvent relation (1.99) and the fluid boundary condition in

(1.101) we have then for $|\alpha| > 0$ sufficiently large,

$$
\begin{aligned}
&\left| \left\langle \frac{\partial \mu}{\partial \nu}, \nabla R_\rho \omega_1 |_{\partial \mathcal{O}} \right\rangle_{\partial \mathcal{O}} \right| \\
&\leq C \left\| \nabla \mu \right\|_{\mathcal{O}}^{\frac{1}{2}} \left\| \mu \right\|_{\mathbf{H}^2(\mathcal{O})}^{\frac{1}{2}} \left\| \frac{1}{\lambda} \left(\mu^3 \big|_\Omega + \omega_1^* \right) \right\|_\Omega^{\frac{1}{2}} \left\| \omega_1 \right\|_{H^1(\Omega)}^{\frac{1}{2}} \\
&\leq C \left\| \nabla \mu \right\|_{\mathcal{O}}^{\frac{1}{2}} \left\| [\omega_1, \omega_2, \mu] \right\|_{D(\mathcal{A}_\rho)}^{\frac{1}{2}} \left\| \frac{1}{\lambda} \left(\mu^3 \big|_\Omega + \omega_1^* \right) \right\|_\Omega^{\frac{1}{2}} \left\| \omega_1 \right\|_{H^1(\Omega)}^{\frac{1}{2}} \\
&\leq C_\vartheta \left\| \nabla \mu \right\|_{\mathcal{O}}^{\frac{1}{2}} \left(\left\| [\omega_1, \omega_2, \mu] \right\|_{\mathbf{H}_\rho} + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho} \right)^{\frac{1}{2}} \times \\
&\qquad \left(\left\| \mu \right\|_{\partial \mathcal{O}} + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho} \right)^{\frac{1}{2}} \left\| \omega_1 \right\|_{H^1(\Omega)}^{\frac{1}{2}} \\
&= C_\vartheta \frac{|\alpha|^{\frac{1}{4}}}{|\alpha|^{\frac{1}{4}}} \left\| \nabla \mu \right\|_{\mathcal{O}}^{\frac{1}{2}} \left(\left\| [\omega_1, \omega_2, \mu] \right\|_{\mathbf{H}_\rho} + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho} \right)^{\frac{1}{2}} \times \\
&\qquad \left(\left\| \mu \right\|_{\partial \mathcal{O}} + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho} \right)^{\frac{1}{2}} \left\| \omega_1 \right\|_{H^1(\Omega)}^{\frac{1}{2}} \\
&\leq \frac{\epsilon}{|\alpha|} \left\| \nabla \mu \right\|_{\mathcal{O}}^2 + |\alpha|^{\frac{1}{3}} \left(\left\| [\omega_1, \omega_2, \mu] \right\|_{\mathbf{H}_\rho} + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho} \right)^{\frac{2}{3}} \times \\
&\qquad \left(\left\| \mu \right\|_{\partial \mathcal{O}} + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho} \right)^{\frac{2}{3}} \left\| \omega_1 \right\|_{H^1(\Omega)}^{\frac{2}{3}} \\
&\leq \frac{\epsilon}{|\alpha|} \left\| \nabla \mu \right\|_{\mathcal{O}}^2 + \epsilon \left(\left\| [\omega_1, \omega_2, \mu] \right\|_{\mathbf{H}_\rho}^2 + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho}^2 \right) \\
&\qquad + C_\epsilon |\alpha|^{\frac{1}{2}} \left(\left\| \mu \right\|_{\partial \mathcal{O}} + \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho} \right) \left\| \omega_1 \right\|_{H^1(\Omega)} \\
&\leq C_\epsilon |\alpha|^{\frac{1}{2}} \left\| \mu \right\|_{\partial \mathcal{O}} \left\| \omega_1 \right\|_{H^1(\Omega)} + C_\epsilon |\alpha| \left\| \omega_1 \right\|_{H^1(\Omega)}^2 + \\
&\qquad \frac{\epsilon}{|\alpha|} \left\| \nabla \mu \right\|_{\mathcal{O}}^2 + \epsilon \left\| [\omega_1, \omega_2, \mu] \right\|_{\mathbf{H}_\rho}^2 + 2\epsilon \left\| [\omega_1^*, \omega_2^*, \mu^*] \right\|_{\mathbf{H}_\rho}^2 ,
\end{aligned}
$$

where we used Young's Inequality first with $(p = 4, q = \frac{4}{3})$ and then with $(p = 2, q = 2)$. Invoking the interpolation inequality (1.103), and applying Young's Inequality with

$(p=\frac{4}{3}, q=4)$ and then $(p=\frac{3}{2}, q=3)$ yields now for sufficiently large $|\alpha|>1$

$$
\begin{aligned}
&\left|\left\langle \frac{\partial \mu}{\partial \nu}, \nabla R_\rho \omega_1|_{\partial\mathcal{O}} \right\rangle_{\partial\mathcal{O}}\right| \\
&\leq C_\epsilon |\alpha|^{\frac{1}{2}} \|\mu\|_{\mathcal{O}}^{\frac{1}{2}} \|\nabla\mu\|_{\mathcal{O}}^{\frac{1}{2}} \|\omega_1\|_{H^1(\Omega)} + C_\epsilon |\alpha| \|\omega_1\|_{H^1(\Omega)}^2 \\
&\qquad + \frac{\epsilon}{|\alpha|} \|\nabla\mu\|_{\mathcal{O}}^2 + \epsilon \|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 + 2\epsilon \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2 \\
&= C_\epsilon \frac{|\alpha|^{\frac{3}{4}}}{|\alpha|^{\frac{1}{4}}} \|\mu\|_{\mathcal{O}}^{\frac{1}{2}} \|\nabla\mu\|_{\mathcal{O}}^{\frac{1}{2}} \|\omega_1\|_{H^1(\Omega)} + C_\epsilon |\alpha| \|\omega_1\|_{H^1(\Omega)}^2 \\
&\qquad + \frac{\epsilon}{|\alpha|} \|\nabla\mu\|_{\mathcal{O}}^2 + \epsilon \|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 + 2\epsilon \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2 \\
&\leq C_\epsilon |\alpha| \|\mu\|_{\mathcal{O}}^{\frac{2}{3}} \|\omega_1\|_{H^1(\Omega)}^{\frac{4}{3}} + C_\epsilon |\alpha| \|\omega_1\|_{H^1(\Omega)}^2 + \frac{2\epsilon}{|\alpha|} \|\nabla\mu\|_{\mathcal{O}}^2 \\
&\qquad + \epsilon \|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 + 2\epsilon \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2 \\
&\leq C_\epsilon |\alpha|^{\frac{3}{2}} \|\omega_1\|_{H^1(\Omega)}^2 + \frac{2\epsilon}{|\alpha|} \|\nabla\mu\|_{\mathcal{O}}^2 + 2\epsilon \|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 + 2\epsilon \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2 . \qquad (1.126)
\end{aligned}
$$

Upon a rescaling of the small parameter $\epsilon>0$, we have then the estimate for $\rho>0$,

$$
\begin{aligned}
&\left|\left\langle \frac{\partial \mu}{\partial \nu}, \nabla R_\rho \omega_1|_{\partial\mathcal{O}} \right\rangle_{\partial\mathcal{O}}\right| \\
&\leq C_\epsilon |\alpha|^{\frac{3}{2}} \|\omega_1\|_{H^1(\Omega)}^2 + \frac{\epsilon}{|\alpha|} \|\nabla\mu\|_{\mathcal{O}}^2 + \epsilon \|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 + \epsilon \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2 . \qquad (1.127)
\end{aligned}
$$

Combining now (1.118), (1.120), (1.125), and (1.127), and taking $|\alpha|>0$ sufficiently large, we have finally for all $\rho\geq 0$,

$$
\begin{aligned}
\left|\left(G_{\rho,2}(\mu)|_\Omega, \omega_1\right)_\Omega\right| &\leq C_\epsilon |\alpha|^{\frac{3}{2}+2\delta} \|\omega_1\|_{D(P_\rho^{\frac{1}{2}})}^2 + \frac{\epsilon}{|\alpha|} \|\nabla\mu\|_{\mathcal{O}}^2 \\
&\quad + \left(\epsilon + \frac{C_\epsilon}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\left(\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_0}^2 + \|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_0}^2 \right). \qquad (1.128)
\end{aligned}
$$

This completes the proof of Lemma 12.

1.5.2 Analysis of the Term $\left(G_{\rho,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega$

Recall the definition of $G_{\rho,1}$ from (1.23)

$$G_{\rho,1}(\omega_1) = R_\rho(P_\rho^{-1}\Delta^2\omega_1). \tag{1.129}$$

With respect to this term we will establish the following estimate:

Lemma 13. *For $\rho \geq 0$,*

$$\left|\left(\left(G_{\rho,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega\right| \leq C_\epsilon |\alpha|^{\frac{3}{2}} \|\omega_1\|^2_{D(P_\rho^{\frac{1}{2}})} + \epsilon\left(\|[\omega_1, \omega_2, \mu]\|^2_{\mathbf{H}_\rho} + \|[\omega_1^*, \omega_2^*, \mu^*]\|^2_{\mathbf{H}_\rho}\right). \tag{1.130}$$

Proof of Lemma 13:

As before the proof of this estimate will entail a dichotomy between the $\rho = 0$ and $\rho > 0$ cases.

Analysis of the term $\left|\left(G_{\rho,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega\right|$ for $\rho = 0$

In this case, we have from (1.129) and the expressions in (1.113) and (1.115),

$$\begin{aligned}\left(G_{0,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega &= \left(R_0\Delta^2\omega_1\big|_\Omega, \omega_1\right)_\Omega \\ &= \left(\Delta^2\omega_1, [R_0\omega_1]_\Omega\right)_\Omega.\end{aligned} \tag{1.131}$$

An integration by parts to right hand side then gives

$$\left(G_{0,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega = \left\langle \frac{\partial\Delta\omega_1}{\partial n}, [R_0\omega_1]_\Omega \right\rangle_{\partial\Omega} - \left(\nabla\Delta\omega_1, \nabla\left[R_0\omega_1\right]_\Omega\right). \tag{1.132}$$

To estimate the first term on the right hand side of (1.132): Using the trace estimate

(1.104), the fact that $R_\rho \in \mathcal{L}(L^2(\Omega), H^{\frac{3}{2}}(\mathcal{O}))$, and the Sobolev Trace Theorem, we have

$$\begin{aligned}
\left|\left\langle \frac{\partial \Delta \omega_1}{\partial n}, [R_0\omega_1]_\Omega \right\rangle_{\partial\Omega}\right| &\leq \left\| \frac{\partial \Delta \omega_1}{\partial n} \right\|_{\partial\Omega} \|[R_0\omega_1]_\Omega\|_{\partial\Omega} \\
&\leq C \|\Delta\omega_1\|^{\frac{1}{2}}_{H^1(\Omega)} \|\Delta\omega_1\|^{\frac{1}{2}}_{H^2(\Omega)} \|R_0\omega_1\|_{H^{\frac{3}{2}}(\mathcal{O})} \\
&\leq C \|\omega_1\|^{\frac{1}{2}}_{H^3(\Omega)} \|\omega_1\|^{\frac{1}{2}}_{H^4(\Omega)} \|\omega_1\|_\Omega \\
&\leq C \|\omega_1\|^{\frac{1}{4}}_{H^2(\Omega)} \|\omega_1\|^{\frac{1}{4}}_{H^4(\Omega)} \|\omega_1\|^{\frac{1}{2}}_{H^4(\Omega)} \|\omega_1\|_\Omega \\
&\leq C \|\omega_1\|^{\frac{1}{4}}_{H^2(\Omega)} \|[\omega_1, \omega_2, \mu]\|^{\frac{3}{4}}_{D(\mathcal{A}_0)} \|\omega_1\|_\Omega .
\end{aligned}$$

Using once more the resolvent relation (1.99) and Young's Inequality twice with $(p = 8, q = \frac{8}{7})$ and $(p = \frac{7}{4}, q = \frac{7}{3})$, we have for $|\alpha| > 0$ sufficiently large,

$$\begin{aligned}
\left|\left\langle \frac{\partial \Delta \omega_1}{\partial n}, [R_0\omega_1]_\Omega \right\rangle_{\partial\Omega}\right| &\leq C |\alpha|^{\frac{3}{4}} \|\omega_1\|^{\frac{1}{4}}_{H^2(\Omega)} \left(\|[\omega_1, \omega_2, \mu]\|_{\mathbf{H}_0} + \|[\omega_1^*, \omega_2^*, \mu^*]\|_{\mathbf{H}_0} \right)^{\frac{3}{4}} \|\omega_1\|_\Omega \\
&\leq \epsilon \|\omega_1\|^2_{H^2(\Omega)} + |\alpha|^{\frac{6}{7}} \left(\|[\omega_1, \omega_2, \mu]\|_{\mathbf{H}_0} + \|[\omega_1^*, \omega_2^*, \mu^*]\|_{\mathbf{H}_0} \right)^{\frac{6}{7}} \|\omega_1\|^{\frac{8}{7}}_\Omega \\
&\leq C_\epsilon |\alpha|^{\frac{3}{2}} \|\omega_1\|^2_\Omega + 2\epsilon \left(\|[\omega_1, \omega_2, \mu]\|^2_{\mathbf{H}_0} + \|[\omega_1^*, \omega_2^*, \mu^*]\|^2_{\mathbf{H}_0} \right) . \quad (1.133)
\end{aligned}$$

To estimate the second term on the right hand side of (1.132): Using $R_\rho \in \mathcal{L}(L^2(\Omega), H^{\frac{3}{2}}(\mathcal{O}))$, and the Sobolev Boundary Trace Inequality, we have

$$\begin{aligned}
|(\nabla\Delta\omega_1, \nabla [R_0\omega_1]_\Omega)| &\leq \|\nabla\Delta\omega_1\|_\Omega \|\nabla [R_0\omega_1]_\Omega\|_\Omega \\
&\leq \|\nabla\Delta\omega_1\|_\Omega \|[R_0\omega_1]_\Omega\|_{H^1(\Omega)} \\
&\leq C \|\omega_1\|_{H^3(\Omega)} \|R_0\omega_1\|_{H^{\frac{3}{2}}(\mathcal{O})} \\
&\leq C \|\omega_1\|^{\frac{1}{2}}_{H^2(\Omega)} \|\omega_1\|^{\frac{1}{2}}_{H^4(\Omega)} \|\omega_1\|_\Omega \\
&\leq C \|\omega_1\|^{\frac{1}{2}}_{H^2(\Omega)} \|[\omega_1, \omega_2, \mu]\|^{\frac{1}{2}}_{D(\mathcal{A}_0)} \|\omega_1\|_\Omega \\
&\leq C\sqrt{|\alpha|} \|\omega_1\|^{\frac{1}{2}}_{H^2(\Omega)} \left(\|[\omega_1, \omega_2, \mu]\|_{\mathbf{H}_0} + \|[\omega_1^*, \omega_2^*, \mu^*]\|_{\mathbf{H}_0} \right)^{\frac{1}{2}} \|\omega_1\|_\Omega ,
\end{aligned}$$

after again using the resolvent relation (1.99), and taking $|\alpha| > 0$ sufficently large.

Proceeding via Young's Inequality with $(p = \frac{3}{2}, q = 3)$ yields

$$\begin{aligned} |(\nabla\Delta\omega_1, \nabla\left[R_0\omega_1\right]_\Omega)| &\leq \epsilon\left(\|[\omega_1,\omega_2,\mu]\|^2_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|^2_{\mathbf{H}_0}\right) + C_\epsilon\left|\alpha\right|^{\frac{2}{3}}\|\omega_1\|^{\frac{2}{3}}_{H^2(\Omega)}\|\omega_1\|^{\frac{4}{3}}_{\Omega} \\ &\leq C_\epsilon\left|\alpha\right|\|\omega_1\|^2_\Omega + 2\epsilon\left(\|[\omega_1,\omega_2,\mu]\|^2_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|^2_{\mathbf{H}_0}\right). \end{aligned} \tag{1.134}$$

Applying (1.133) and (1.134) to the right hand side of (1.132) (and rescaling parameter $\epsilon > 0$) now gives

$$\left|(G_{0,1}(\omega_1)|_\Omega, \omega_1)_\Omega\right| \leq C_\epsilon\left|\alpha\right|^{\frac{3}{2}}\|\omega_1\|^2_\Omega + \epsilon\left(\|[\omega_1,\omega_2,\mu]\|^2_{\mathbf{H}_0} + \|[\omega_1^*,\omega_2^*,\mu^*]\|^2_{\mathbf{H}_0}\right). \tag{1.135}$$

Analysis of the term $\left|\left(G_{\rho,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega\right|$ for $\rho > 0$

Again from (1.129) and the expressions in (1.113) and (1.115), we have

$$\begin{aligned} (G_{\rho,1}(\omega_1)|_\Omega, \omega_1)_\Omega &= (R_\rho P_\rho^{-1}\Delta^2\omega_1\big|_\Omega, \omega_1)_\Omega \\ &= \left(P_\rho^{-1}\Delta^2\omega_1, [R_\rho\omega_1]_\Omega\right)_\Omega. \end{aligned} \tag{1.136}$$

At this point we reinvoke the positive definite, self-adjoint operator $\mathring{\mathbf{A}} : D(\mathring{A}) \subset L^2(\Omega) \to L^2(\Omega)$ in (1.121). In this connection, we recall the following characterizations (see [25]):

$$D(\mathring{\mathbf{A}}^\eta) \approx \begin{cases} \left\{\varpi \in H^{4\eta}(\Omega) : \varpi|_{\partial\Omega} = 0\right\}, & \text{for } \frac{1}{8} < \eta < \frac{3}{8} \\ \left\{\varpi \in H^{4\eta}(\Omega) : \varpi|_{\partial\Omega} = \frac{\partial\varpi}{\partial n}\big|_{\partial\Omega} = 0\right\}, & \text{for } \frac{3}{8} < \eta \leq 1. \end{cases} \tag{1.137}$$

Proceeding from (1.136) we have

$$\begin{aligned} ((G_{\rho,1}(\omega_1)|_\Omega, \omega_1)_\Omega &= \left(P_\rho^{-1}\mathring{\mathbf{A}}\omega_1, [R_\rho\omega_1]_\Omega\right)_\Omega \\ &= \left(\mathring{\mathbf{A}}^{\frac{5}{8}+\epsilon}\omega_1, \mathring{\mathbf{A}}^{\frac{3}{8}-\epsilon}P_\rho^{-1}\left[R_\rho\omega_1\right]_\Omega\right)_\Omega. \end{aligned} \tag{1.138}$$

Using in part the fact that $\mathring{\mathbf{A}}^{\frac{3}{8}-\epsilon}P_\rho^{-1}R_\rho^*B_\rho R_\rho \in \mathcal{L}(L^2(\Omega))$ – where again operator $R_\rho^*B_\rho$ has the characterization in (1.115) – an estimation of right hand side then gives

$$\begin{aligned}\left|\left(\left(G_{\rho,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega\right| &\leq \left\|\mathring{\mathbf{A}}^{\frac{5}{8}+\epsilon}\omega_1\right\|_\Omega \left\|\mathring{\mathbf{A}}^{\frac{3}{8}-\epsilon}P_\rho^{-1}\left[R_\rho\omega_1\right]_\Omega\right\|_{\partial\Omega} \\ &\leq C\left\|\omega_1\right\|^{\frac{1}{2}-4\epsilon}_{D(\mathring{\mathbf{A}}^{\frac{1}{2}})}\left\|\omega_1\right\|^{\frac{1}{2}+4\epsilon}_{D(\mathring{\mathbf{A}}^{\frac{3}{4}})}\left\|\omega_1\right\|_\Omega \\ &\leq C\left\|\omega_1\right\|^{\frac{1}{2}-4\epsilon}_{D(\mathring{\mathbf{A}}^{\frac{1}{2}})}\left\|[\omega_1,\omega_2,\mu]\right\|^{\frac{1}{2}+4\epsilon}_{D(\mathcal{A}_\rho)}\left\|\omega_1\right\|_{H^1(\Omega)}.\end{aligned}$$

Using once more the resolvent relation (1.99) and Young's Inequality first with $(p=\frac{4}{1-8\epsilon}, q=\frac{4}{3+8\epsilon})$ then with $(p=\frac{3+8\epsilon}{2}, q=\frac{3+8\epsilon}{1+8\epsilon})$, we have for $|\alpha|>0$ sufficiently large,

$$\begin{aligned}&\left|\left(\left(G_{\rho,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega\right| \\ &\leq C\left|\alpha\right|^{\frac{1}{2}+4\epsilon}\left\|\omega_1\right\|^{\frac{1}{2}-4\epsilon}_{D(\mathring{\mathbf{A}}^{\frac{1}{2}})}\left(\left\|[\omega_1,\omega_2,\mu]\right\|_{\mathbf{H}_\rho}+\left\|[\omega_1^*,\omega_2^*,\mu^*]\right\|_{\mathbf{H}_\rho}\right)^{\frac{1}{2}+4\epsilon}\left\|\omega_1\right\|_{H^1(\Omega)} \\ &\leq \epsilon\left\|\omega_1\right\|^2_{D(\mathring{\mathbf{A}}^{\frac{1}{2}})}+C_\epsilon\left|\alpha\right|^{\frac{2+16\epsilon}{3+8\epsilon}}\left(\left\|[\omega_1,\omega_2,\mu]\right\|_{\mathbf{H}_\rho}+\left\|[\omega_1^*,\omega_2^*,\mu^*]\right\|_{\mathbf{H}_\rho}\right)^{\frac{2+16\epsilon}{3+8\epsilon}}\left\|\omega_1\right\|^{\frac{4}{3+8\epsilon}}_{H^1(\Omega)} \\ &\leq C_\epsilon\left|\alpha\right|^{1+8\epsilon}\left\|\omega_1\right\|^2_{H^1(\Omega)}+2\epsilon\left(\left\|[\omega_1,\omega_2,\mu]\right\|^2_{\mathbf{H}_\rho}+\left\|[\omega_1^*,\omega_2^*,\mu^*]\right\|^2_{\mathbf{H}_\rho}\right), \qquad (1.139)\end{aligned}$$

after using once more the characterization (1.137).

Combining (1.135) and (1.139), we have then for all $\rho\geq 0$ and $|\alpha|>0$ sufficiently large,

$$\left|\left(\left(G_{\rho,1}(\omega_1)|_\Omega, \omega_1\right)_\Omega\right| \leq C_\epsilon\left|\alpha\right|^{\frac{3}{2}}\left\|\omega_1\right\|^2_{D(P_\rho^{\frac{1}{2}})}+\epsilon\left(\left\|[\omega_1,\omega_2,\mu]\right\|^2_{\mathbf{H}_\rho}+\left\|[\omega_1^*,\omega_2^*,\mu^*]\right\|^2_{\mathbf{H}_\rho}\right). \quad (1.140)$$

This completes the proof of Lemma 13.

1.5.3 Proof of Theorem 6

We now combine Lemma 12 and Lemma 13 together. Applying the estimates (1.119) and (1.130) to the right hand side of the expression (1.109), and using the resolvent relation $\lambda\omega_1=\omega_2+\omega_1^*$ yield the following result:

Lemma 14. *For $\rho > 0$, the solution variables $[\omega_1, \omega_2, \mu]$ of the resolvent equation obey the following estimate, for $|\alpha| > 0$ sufficiently large:*

$$\begin{aligned}\left|\left|\left(p|_\Omega, \omega_1\right)_\Omega\right|\right| &\leq C_\epsilon |\alpha|^{\frac{3}{2}+2\delta} \|\omega_1\|^2_{D(P_\rho^{\frac{1}{2}})} + \frac{\epsilon}{|\alpha|} \|\nabla\mu\|^2_{\mathcal{O}} \\ &\quad + \left(\epsilon + \frac{C_\epsilon}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\left(\|[\omega_1, \omega_2, \mu]\|^2_{\mathbf{H}_\rho} + \|[\omega_1^*, \omega_2^*, \mu^*]\|^2_{\mathbf{H}_\rho}\right) \\ &\leq \frac{\epsilon}{|\alpha|} \|\nabla\mu\|^2_{\mathcal{O}} + \left(\epsilon + \frac{C_{\epsilon,\vartheta}}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\left(\|[\omega_1, \omega_2, \mu]\|^2_{\mathbf{H}_\rho} + \|[\omega_1^*, \omega_2^*, \mu^*]\|^2_{\mathbf{H}_\rho}\right).\end{aligned} \tag{1.141}$$

In completing the proof of Theorem 6, we should always bear in mind that Criteria 1 and 2 are imposed upon complex parameter $\lambda = \alpha + i\beta$. For clarity of exposition the proof is divided into a few steps.

Step 1. We apply the estimate (1.141) to the right hand side of (1.106), so as to have

$$\begin{aligned}|\beta| \|\mu\|^2_{\mathcal{O}} &= \left|-\mathrm{Im}\left(p|_\Omega, \lambda\omega_1 - \omega_1^*\right)_\Omega + \mathrm{Im}\left(\mu^*, \mu\right)_{\mathcal{O}}\right| \\ &\leq C_\vartheta |\alpha| \left|\mathrm{Im}\left(p|_\Omega, \omega_1\right)_\Omega\right| + \left|\mathrm{Im}\left(p|_\Omega, \omega_1^*\right)_\Omega\right| + \left|\mathrm{Im}\left(\mu^*, \mu\right)\right|_{\mathcal{O}} \\ &\leq \left|\mathrm{Im}\left(p|_\Omega, \omega_1^*\right)_\Omega\right| + \epsilon C_\vartheta^* \|\nabla\mu\|^2_{\mathcal{O}} \\ &\quad + C_\vartheta^* |\alpha| \left(\epsilon + \frac{C_\epsilon}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\left(\|[\omega_1, \omega_2, \mu]\|^2_{\mathbf{H}_\rho} + C_\epsilon |\alpha|^{\frac{1}{2}+2\delta} \|[\omega_1^*, \omega_2^*, \mu^*]\|^2_{\mathbf{H}_\rho}\right).\end{aligned} \tag{1.142}$$

Now for the first term on the right hand side of (1.142): Since the datum ω_1^* satisfies the compatibility condition $\int_\Omega \omega_1^* \, d\Omega = 0$, then there is a function $\varphi(\omega_1^*) \in \mathbf{H}^1(\mathcal{O})$ which

solves

$$\begin{aligned} \operatorname{div}(\varphi) &= 0 \ \text{ in } \mathcal{O}; \\ \varphi &= 0 \ \text{ in } S; \\ \varphi &= [0,0,\omega_1^*] \ \text{ in } \Omega, \end{aligned} \tag{1.143}$$

with the estimate

$$\|\nabla\varphi\|_{\mathcal{O}} \leq C \|\omega_1^*\|_{H^{\frac{1}{2}}(\Omega)} \tag{1.144}$$

(see e.g., p. 9 of [23]). With this solution variable $\varphi(\omega_1^*)$ in hand, and by virtue of the geometry in play, we then have

$$\begin{aligned} (p|_{\Omega}, \omega_1^*)_{\Omega} &= -\left(\frac{\partial\mu}{\partial\nu}, \varphi\right)_{\partial\mathcal{O}} + (p\nu, \varphi)_{\partial\mathcal{O}} \\ &\quad - (\nabla\mu, \nabla\varphi)_{\mathcal{O}} - (\Delta\mu, \varphi)_{\mathcal{O}} + (\nabla p, \varphi)_{\partial\mathcal{O}} + 0 \\ &= -(\nabla\mu, \nabla\varphi)_{\mathcal{O}} - \lambda(\mu, \varphi)_{\mathcal{O}} + (\mu^*, \varphi)_{\mathcal{O}}. \end{aligned}$$

We have then upon estimating this term, with the use of the estimate (1.144), and for large $|\alpha| > 0$

$$\left|(p|_{\Omega}, \omega_1^*)_{\Omega}\right| \leq \epsilon\left(\|\nabla\mu\|_{\mathcal{O}}^2 + |\alpha|\,\|\mu\|_{\mathcal{O}}^2\right) + |\alpha|\, C_{\epsilon}\, \|[\omega_1^*, \omega_2^*, \mu^*]\|_{\mathbf{H}_\rho}^2. \tag{1.145}$$

Applying this estimate to the right hand side of (1.142) now yields (after a rescaling of $\epsilon > 0$)

$$\begin{aligned} |\beta|\,\|\mu\|_{\mathcal{O}}^2 &\leq \epsilon C_{\vartheta}^*\,\|\nabla\mu\|_{\mathcal{O}}^2 + C_{\vartheta}^*\,|\alpha|\left(\epsilon + \frac{C_{\epsilon}}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\|[\omega_1, \omega_2, \mu]\|_{\mathbf{H}_\rho}^2 \\ &\quad + C_{\epsilon,\vartheta}\,|\alpha|\,\|[\omega_1^*, \omega_2^*, \mu^*]\|_{\mathbf{H}_\rho}^2, \end{aligned} \tag{1.146}$$

where above, positive constant C_{ϑ}^* is independent of parameter $\epsilon > 0$.

Step 2: We invoke the relation (1.107):

$$\|\nabla\mu\|_{\mathcal{O}}^2 = |\alpha|\,\|\mu\|_{\mathcal{O}}^2 - \mathrm{Re}\,(p|_\Omega\,, \lambda\omega_1 - \omega_1^*)_\Omega + \mathrm{Re}\,(\mu_0^*, \mu)_{\mathcal{O}}\,.$$

Applying the estimates (1.146), (1.141), and (1.145) to right hand side now gives

$$\begin{aligned}\|\nabla\mu\|_{\mathcal{O}}^2 \leq \epsilon C_\vartheta^*\,\|\nabla\mu\|_{\mathcal{O}}^2 + C_\vartheta^*\,|\alpha|\left(\epsilon + \frac{C_\epsilon}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 \\ + C_{\epsilon,\vartheta}\,|\alpha|\,\|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2\,. \qquad (1.147)\end{aligned}$$

Step 3: We apply the estimate (1.147) to the right hand side of (1.108). This gives

$$\begin{aligned}|\alpha|\left\|\begin{bmatrix}\omega_1\\ \omega_2\\ \mu\end{bmatrix}\right\|_{\mathbf{H}_\rho}^2 &= \left|\|\nabla\mu\|_{\mathcal{O}}^2 + \mathrm{Re}\left(\begin{bmatrix}\omega_1^*\\ \omega_2^*\\ \mu^*\end{bmatrix}, \begin{bmatrix}\omega_1\\ \omega_2\\ \mu\end{bmatrix}\right)_{\mathbf{H}_\rho}\right| \\ &\leq \epsilon C_\vartheta^*\,\|\nabla\mu\|_{\mathcal{O}}^2 + C_\vartheta^*\,|\alpha|\left(\epsilon + \frac{C_\epsilon}{|\alpha|^{\frac{1}{2}-2\delta}}\right)\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 \\ &\quad + C_{\epsilon,\vartheta}\,|\alpha|\,\|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2\,. \qquad (1.148)\end{aligned}$$

Step 4: Taking $\epsilon > 0$ sufficently small, (with again positive constant C_ϑ^* being

independent of parameter $\epsilon > 0$) we have

$$(1 - \epsilon C^*_\vartheta) \, |\alpha| \left\| \begin{bmatrix} \omega_1 \\ \omega_2 \\ \mu \end{bmatrix} \right\|^2_{\mathbf{H}_\rho}$$

$$\leq \epsilon C^*_\vartheta \, \|\nabla \mu\|^2_{\mathcal{O}} + C_{\epsilon,\vartheta} \frac{|\alpha|}{|\alpha|^{\frac{1}{2}-2\delta}} \left\| \begin{bmatrix} \omega_1 \\ \omega_2 \\ \mu \end{bmatrix} \right\|^2_{\mathbf{H}_\rho} + C_{\epsilon,\vartheta} \, |\alpha| \left\| \begin{bmatrix} \omega_1^* \\ \omega_2^* \\ \mu^* \end{bmatrix} \right\|^2_{\mathbf{H}_\rho},$$

and so

$$|\alpha| \left\| \begin{bmatrix} \omega_1 \\ \omega_2 \\ \mu \end{bmatrix} \right\|^2_{\mathbf{H}_\rho}$$

$$\leq \frac{\epsilon C^*_\vartheta}{1 - \epsilon C^*_\vartheta} \, \|\nabla \mu\|^2_{\mathcal{O}} + C_{\epsilon,\vartheta} \frac{|\alpha|}{|\alpha|^{\frac{1}{2}-2\delta}} \left\| \begin{bmatrix} \omega_1 \\ \omega_2 \\ \mu \end{bmatrix} \right\|^2_{\mathbf{H}_\rho} + C_{\epsilon,\vartheta} \, |\alpha| \left\| \begin{bmatrix} \omega_1^* \\ \omega_2^* \\ \mu^* \end{bmatrix} \right\|^2_{\mathbf{H}_\rho}. \tag{1.149}$$

Step 5: We return to the estimate (1.147). Applying (1.149) thereto gives for $\epsilon > 0$ sufficently small,

$$\|\nabla \mu\|^2_{\mathcal{O}} \leq \frac{\epsilon \widetilde{C}_\vartheta}{1 - \epsilon C^*_\vartheta} \, \|\nabla \mu\|^2_{\mathcal{O}} + \frac{C_{\epsilon,\vartheta} \, |\alpha|}{|\alpha|^{\frac{1}{2}-2\delta}} \, \|[\omega_1, \omega_2, \mu]\|^2_{\mathbf{H}_\rho} + C_{\epsilon,\vartheta} \, |\alpha| \, \|[\omega_1^*, \omega_2^*, \mu^*]\|^2_{\mathbf{H}_\rho}, \tag{1.150}$$

where positive constants $C^*_\vartheta, \widetilde{C}_\vartheta$ do not depend upon $\epsilon > 0$. Taking $\epsilon > 0$ small enough we have now

$$\left(1 - \frac{\epsilon \widetilde{C}_\vartheta}{1 - \epsilon C^*_\vartheta}\right) \|\nabla \mu\|^2_{\mathcal{O}} \leq \frac{C_{\epsilon,\vartheta} \, |\alpha|}{|\alpha|^{\frac{1}{2}-2\delta}} \, \|[\omega_1, \omega_2, \mu]\|^2_{\mathbf{H}_\rho} + C_{\epsilon,\vartheta} \, |\alpha| \, \|[\omega_1^*, \omega_2^*, \mu^*]\|^2_{\mathbf{H}_\rho}, \tag{1.151}$$

whence we obtain

$$\|\nabla\mu\|_{\mathcal{O}}^2 \leq \frac{C_{\epsilon,\vartheta}\,|\alpha|}{|\alpha|^{\frac{1}{2}-2\delta}}\,\|[\omega_1,\omega_2,\mu]\|_{\mathbf{H}_\rho}^2 + C_{\epsilon,\vartheta}\,|\alpha|\,\|[\omega_1^*,\omega_2^*,\mu^*]\|_{\mathbf{H}_\rho}^2\,. \tag{1.152}$$

Step 6: We finish the proof by applying the estimate (1.152) to the right hand side of (1.149). This gives

$$|\alpha|\left\|\begin{bmatrix}\omega_1\\ \omega_2\\ \mu\end{bmatrix}\right\|_{\mathbf{H}_\rho}^2 \leq C_{\epsilon,\vartheta}\frac{|\alpha|}{|\alpha|^{\frac{1}{2}-2\delta}}\left\|\begin{bmatrix}\omega_1\\ \omega_2\\ \mu\end{bmatrix}\right\|_{\mathbf{H}_\rho}^2 + C_{\epsilon,\vartheta}\,|\alpha|\left\|\begin{bmatrix}\omega_1^*\\ \omega_2^*\\ \mu^*\end{bmatrix}\right\|_{\mathbf{H}_\rho}^2\,. \tag{1.153}$$

Taking now $|\alpha|$ so large that $1-\frac{C_{\epsilon,\vartheta}}{|\alpha|^{\frac{1}{2}-2\delta}} > \frac{1}{2}$; i.e.,

$$|\alpha| > (2C_{\epsilon,\vartheta})^{\frac{2}{1-4\delta}}\,,$$

we have finally

$$\frac{|\alpha|}{2}\left\|\begin{bmatrix}\omega_1\\ \omega_2\\ \mu\end{bmatrix}\right\|_{\mathbf{H}_\rho}^2 \leq C_{\epsilon,\vartheta}\,|\alpha|\left\|\begin{bmatrix}\omega_1^*\\ \omega_2^*\\ \mu^*\end{bmatrix}\right\|_{\mathbf{H}_\rho}^2\,, \tag{1.154}$$

which gives the uniform bound (1.102). This completes the proof of Theorem 6.

Chapter 2

Numerical Analysis

This chapter describes how the maximality argument in Chapter 1 can be utilized to approximate solutions to the fluid-structure PDE under consideration. In particular, the numerical method outlined here solves the static problem resulting from the resolvent equations (1.35)-(1.41), but this approach in principle can be modified to solve the time dependent problem in a similar way to what was done in [3] and [20] via the framework set down in [32]. Namely, the solution of the static problem can be exploited via the exponential formula for the semi-group to generate a time dependent solution.

$$\begin{bmatrix} w(t) \\ w_t(t) \\ u(t) \end{bmatrix} = e^{\mathcal{A}t} \begin{bmatrix} w_1 \\ w_2 \\ u_0 \end{bmatrix} = \lim_{n\to\infty} \left(I - \frac{t}{n}\mathcal{A}\right)^{-n} \begin{bmatrix} w_1 \\ w_2 \\ u_0 \end{bmatrix}, \text{ for } \begin{bmatrix} w_1 \\ w_2 \\ u_0 \end{bmatrix} \in \mathbf{H}_\rho.$$

By setting $\lambda = n/t$ this equation can be written as

$$(\lambda I - \mathcal{A})^n \begin{bmatrix} w(t) \\ w_t(t) \\ u(t) \end{bmatrix} = \lambda^n \begin{bmatrix} w_1 \\ w_2 \\ u_0 \end{bmatrix}$$

and solved using the numerical scheme set forth in the maximality argument. By fixing n large enough to approximate the exponential semigroup operator, the numerical scheme can then be used to deliver the solution of the time dependent system at any time t.
We will outline here a certain numerical implementation of the finite element method (FEM) and provide convergence results for the approximation with respect to "mesh parameter" h. Finally, we will provide an explicit model problem as a numerical example.

2.1 Finite Element Formulation

The finite element method is a numerical implementation of the Ritz-Galerkin method over a specific set of basis functions defined on a mesh of the domain. In this case the fluid domain is divided into tetrahedra and the plate domain into triangles. Basis functions are associated to points in the mesh and the system is solved in this finite dimensional setting via a matrix/vector equation, see e.g. [6]. We will demonstrate in this section that the discrete FEM formulation of (1.35)-(1.41) is well-posed.
In what follows, the three dimensional body $\mathcal{O}$ will be taken to be a polyhedron. Given a positive (and small) parameter h of discretization, we let $\{e_\ell\}_{\ell=1}^{N_h}$ be an FEM "triangulation" of $\mathcal{O}$, where each element e_ℓ is a tetrahedron (and so, among other properties, $\bigcup_{\ell=1}^{N_h} e_\ell = \mathcal{O}$, see [6] and Figure 2.21 below).

(A) Relative to the "triangulation" of $\mathcal{O}$, V_h will denote the classic $\mathbf{H}^1$-conforming FEM finite dimensional subspace such that

$$\mu_h \in V_h \Rightarrow \mu_h\big|_{e_\ell} \in [\mathbb{P}^2]^3,\ V_h \subset \mathbf{H}_0^1(\mathcal{O}),\ \ V_h \not\subset \mathbf{H}^2(\mathcal{O});\ \ V_h \subset [\mathcal{C}(\bar{\Omega})]^3,\ \ V_h \not\subset [\mathcal{C}^1(\bar{\Omega})]^3. \tag{2.1}$$

(See [6].) Subsequently to handle the inhomogeneity we specify the set

$$\tilde{V}_h = \Big\{ \mu_h + \gamma_0^+(\xi) \in \mathbf{H}^1(\mathcal{O}) : \mu_h \in V_h \text{ and } \gamma_0^+(\xi)\Big|_{\partial\mathcal{O}} = \begin{cases} \vec{0}, & \text{on } S \\ [0,0,\xi], & \text{for } \xi \in H^2(\Omega), \end{cases} \Big\}. \tag{2.2}$$

(B) In addition Π_h will denote the L^2-FEM finite dimensional subspace for the pressure variable defined by

$$\Pi_h = \big\{ q_h \in \frac{L^2(\mathcal{O})}{\mathbb{R}} \cap \mathcal{C}(\bar{\mathcal{O}}) : \ \forall \ell = 1, ..., N_h; \ q_h\big|_{e_\ell} \in \mathbb{P}^1 \big\} \tag{2.3}$$

(see [16] and [6]).

Moreover, we let $\{\tilde{e}_\ell\}_{\ell=1}^{\tilde{N}_h}$ be a FEM triangulation of the two dimensional polygonal region Ω, where each element $\tilde{e}_\ell$ is a triangle.

(C) Similarly, X_h will denote a conforming FEM subspace such that

$$\psi_h \in X_h \Rightarrow \psi_h\big|_{\tilde{e}_\ell} \in \mathbb{P}^5, \ X_h \subset H_0^2(\Omega), \ \ X_h \not\subset H^3(\Omega); \ \ X_h \subset \mathcal{C}^1(\bar{\Omega}), \ \ X_h \not\subset \mathcal{C}^2(\bar{\Omega}) \tag{2.4}$$

(see e.g. [16] and [33] for details of the explicit construction of these piecewise polynomials. As such, the basis functions which generate X_h are "conforming", relative to fourth-order boundary value problems.)

For the spaces V_h, Π_h and X_h described above we will have need of the following discrete estimates relative to mesh parameter h:

(A′) In regard to the $\mathbf{H}^1(\mathcal{O})$-conforming FEM space V_h in (2.1) we have the following estimate: For $\mu \in \mathbf{H}^2(\mathcal{O}) \cap \mathbf{H}_0^1(\mathcal{O})$,

$$\min_{\mu_h \in V_h} \|\mu - \mu_h\|_{\mathbf{H}_0^1(\mathcal{O})} \leq Ch|\mu|_{2,\mathcal{O}}. \tag{2.5}$$

(See Theorem 5.6, p. 224, of [6].)

(B′) Similarly, in regard to the finite dimensional space Π_h, we have the discrete estimate: For $q \in H^1(\mathcal{O})/\mathbb{R}$,

$$\min_{q_h \in \Pi_h} \|q - q_h\|_{L^2(\mathcal{O})} \leq Ch\|q\|_{H^1(\mathcal{O})}. \tag{2.6}$$

(See e.g., Corollary 1.128, p. 70, of [21].)

(C′) Finally, with regard to the FEM space X_h in (2.4), we have the following discrete estimates:

(i) For $\psi \in H^4(\Omega) \cap H_0^2(\Omega)$,

$$\min_{\psi_h \in X_h} \|\psi - \psi_h\|_{H_0^2(\Omega)} \leq Ch^2|\psi|_{4,\Omega}, \tag{2.7}$$

(ii) For $\psi \in H^3(\Omega) \cap H_0^2(\Omega)$,

$$\min_{\psi_h \in X_h} \|\psi - \psi_h\|_{H_0^2(\Omega)} \leq Ch|\psi|_{3,\Omega}. \tag{2.8}$$

(See estimate (5.82), p. 225, of [6].)

The goal here is to find a finite dimensional approximation $[w_{1h}, w_{2h}, u_h] \in X_h \times X_h \times \tilde{V}_h$ to the solution $[w_1, w_2, u] \in D(\mathcal{A}_\rho)$ of (1.52), as well as an approximation p_h of the associated fluid pressure p. We shall see that these particular FEM subspaces are chosen with a view of satisfying the (discrete) Babuška-Brezzi condition relative to a mixed variational formulation, a formulation which is wholly analogous to that in (1.42) for the static fluid-structure PDE system (1.35)-(1.41). We further note that, by way of satisfying said inf-sup condition, it is indispensable that the structural component space X_h be H^2-conforming (see (2.4)). In addition, this mixed variational formulation for the coupled problem (1.35)-(1.41), like the mixed method for uncoupled Stokes or

Navier-Stokes flow, allows for the implementation of approximating fluid basis functions (in V_h) which are not divergence free (see [11]).

In line with the maximality argument of Section 1.3.2, the initial task in the present finite dimensional setting is to numerically resolve the structural solution component of the PDE system (1.35)-(1.41). Namely, with reference to the bilinear and linear functionals $a_\lambda(\cdot,\cdot), b(\cdot,\cdot)$ and $\mathbb{F}(\cdot)$ of (1.43), the present discrete problem is to find $[w_{1h}, \tilde{c}_h] \in X_h \times \mathbb{R}$ which solve:

$$\begin{cases} a_\lambda(w_{1h}, \psi_h) + b(\psi_h, \tilde{c}_h) = \mathbb{F}(\psi_h), & \forall \psi_h \in X_h, \\ b(w_{1h}, r) = 0, & \forall r \in \mathbb{R}. \end{cases} \tag{2.9}$$

Assuming this variational problem can be solved uniquely, w_{2h} is immediately resolved via the relation

$$w_{2h} = \lambda w_{1h} - w_1^* \tag{2.10}$$

(cf. (1.35)). Subsequently we can recover fluid and pressure approximations u_h and p_h from the discrete solution pair $[w_{1h}, \tilde{c}_h] \in X_h \times \mathbb{R}$ of (2.9). Indeed, to this end we will invoke the classic mixed variational formulation for Stokes flow, so as to approximate the fluid maps $[\tilde{f}(\cdot), \tilde{\pi}(\cdot)]$ and $[\tilde{\mu}(\cdot), \tilde{q}(\cdot)]$ of (1.30) and (1.32), respectively. (See [11].) Let bilinear forms $\tilde{\mathbf{a}}_\lambda(\cdot,\cdot) : \mathbf{H}_0^1(\mathcal{O}) \times \mathbf{H}_0^1(\mathcal{O}) \to \mathbb{R}$ and $\tilde{\mathbf{b}}(\cdot,\cdot) : \mathbf{H}_0^1(\mathcal{O}) \times \mathbf{L}^2(\mathcal{O})/\mathbb{R} \to \mathbb{R}$ be defined respectively as follows:

$$\tilde{\mathbf{a}}_\lambda(\mu, \varphi) = \lambda(\mu, \varphi)_{\mathcal{O}} + (\nabla\mu, \nabla\varphi)_{\mathcal{O}}, \quad \forall \mu, \varphi \in \mathbf{H}_0^1(\mathcal{O}); \tag{2.11}$$

$$\tilde{\mathbf{b}}(\mu, q) = -(\mathrm{div}(\mu), q)_{\mathcal{O}}, \qquad \forall \mu \in \mathbf{H}_0^1(\mathcal{O}), q \in \frac{L^2(\mathcal{O})}{\mathbb{R}}. \tag{2.12}$$

Moreover, we define the standard Sobolev trace map $\gamma_0 : \mathbf{H}^k(\mathcal{O}) \to \mathbf{H}^{k-1/2}(\partial\mathcal{O})$, for

$k = 1, 2, 3, \ldots$. That is for $f \in [\mathcal{C}^\infty(\bar{\mathcal{O}})]^3$,

$$\gamma_0(f) = f\big|_{\partial\mathcal{O}}.$$

Since $\gamma_0(\cdot)$ is continuous and surjective, then for any $\phi \in H^{k-\frac{1}{2}}(\Omega)$ we have the existence and uniqueness of an element in $\mathbf{H}^k(\mathcal{O})$, denoted here as $\gamma_0^+(\phi)$, which satisfies

$$\gamma_0\gamma_0^+(\phi) = \begin{cases} \vec{0} & \text{on } S, \\ [0, 0, \phi] & \text{on } \Omega. \end{cases} \tag{2.13}$$

Therewith, the classic mixed FEM for (1.30) is given as follows: With subspaces V_h and Π_h as given in (2.1) and (2.3) respectively, and given $\phi \in H^{1/2}(\Omega)$, find the unique pair $[\tilde{f}_{0h}(\phi), \tilde{\pi}_h(\phi)] \in V_h \times \Pi_h$ such that

$$\tilde{\mathbf{a}}_\lambda(\tilde{f}_{0h}, \varphi_h) + \tilde{\mathbf{b}}(\varphi_h, \tilde{\pi}_h) = -\tilde{\mathbf{a}}_\lambda(\gamma_0^+(\phi), \varphi_h) \qquad \forall \varphi_h \in V_h, \tag{2.14}$$

$$\tilde{\mathbf{b}}(\tilde{f}_{0h}, \varrho_h) = -\left[\frac{\int_\Omega \phi \, d\Omega}{\text{meas}(\mathcal{O})}\right] \int_{\mathcal{O}} \varrho_h \, d\mathcal{O} - \tilde{\mathbf{b}}(\gamma_0^+(\phi), \varrho_h) \quad \forall \varrho_h \in \Pi_h. \tag{2.15}$$

Likewise, the classic mixed FEM for (1.32) is given as follows: For given $u^* \in H^{-1}(\mathcal{O})$, find the unique pair $[\tilde{\mu}_h(u^*), \tilde{q}_h(u^*)] \in V_h \times \Pi_h$ such that

$$\tilde{\mathbf{a}}_\lambda(\tilde{\mu}_h, \varphi_h) + \tilde{\mathbf{b}}(\varphi_h, \tilde{q}_h) = (u^*, \varphi_h)_{\mathcal{O}} \quad \forall \varphi_h \in V_h; \tag{2.16}$$

$$\tilde{\mathbf{b}}(\tilde{\mu}_h, \varrho_h) = 0 \qquad \forall \varrho_h \in \Pi_h. \tag{2.17}$$

By the Babuška-Brezzi Theorem, the two discrete variational formulations (2.14)-(2.15) and (2.16)-(2.17) are well-posed; see [11]. (In particular, with the so-called Taylor-Hood formulation in place–i.e., fluid approximation space V_h consists of piecewise quadratic functions, and pressure approximation space Π_h consists of piecewise linear functions–then the aforesaid inf-sup condition is satisfied *uniformly* in parameter h.)

With the approximating solution maps (2.14)-(2.17) in place and assuming the structural component approximation $[w_{1h}, \tilde{c}_h]$ is known, we then set

$$u_h = \tilde{f}_{0h}(\lambda w_{1h} - w_1^*) + \gamma_0^+(\lambda w_{1h} - w_1^*) + \tilde{\mu}_h(u^*); \tag{2.18}$$

$$p_h = \tilde{\pi}_h(\lambda w_{1h} - w_1^*) + \tilde{q}_h(u^*) + \tilde{c}_h, \tag{2.19}$$

(c.f. (1.44).)

Now in regard to the variational problem in (2.9), one will in fact have unique solvability of this discrete problem, via the Babuška-Brezzi Theorem, provided that the following inf-sup condition is satisfied:

$$\sup_{\phi_h \in X_h} \frac{b(\phi_h, r)}{\|\phi_h\|_{H_0^2(\Omega)}} \geq \beta_h |r|, \ \forall r \in \mathbb{R}. \tag{2.20}$$

But what is more, in order to ensure stability and ultimately convergence of the numerical solutions obtained by our particular FEM, it is indispensable that the "discrete" inf-sup condition (2.20) be *uniform* of parameter $h > 0$ (at least for h small enough).

In fact we have the following result:

Lemma 15. *Let the bilinear form $b(\cdot, \cdot) : H_0^2(\Omega) \times \mathbb{R} \to \mathbb{R}$ be as defined in (1.43). Then for parameter $h > 0$ small enough one has the "inf-sup" estimate*

$$\sup_{\phi_h \in X_h} \frac{b(\phi_h, r)}{\|\phi_h\|_{H_0^2(\Omega)}} \geq C|r|, \ \forall r \in \mathbb{R}. \tag{2.21}$$

where $C = \|\xi\|_{H_0^2(\Omega)} - \epsilon$, and ξ is the solution of the boundary value problem (1.60). Here, $\epsilon > 0$ can be taken arbitrarily small.

Proof of Lemma 15. We resurrect the elliptic variable $\xi \in H^4(\Omega) \cap H_0^2(\Omega)$ from the

earlier maximality argument. Namely, ξ solves

$$\Delta^2 \xi = 1 \text{ in } \Omega; \quad \xi\big|_{\partial\Omega} = \frac{\partial \xi}{\partial \nu}\Big|_{\partial\Omega} = 0 \text{ on } \partial\Omega.$$

Then by Green's First Identity we have that ξ solves the following variational problem for all $\psi \in H_0^2(\Omega)$:

$$\begin{aligned} (1, \psi)_\Omega &= (\Delta^2 \xi, \psi)_\Omega \\ &= (\Delta \xi, \Delta \psi)_\Omega, \text{ for all } \psi \in H_0^2(\Omega). \end{aligned} \tag{2.22}$$

Let now $\xi_h \in X_h$ denote the "energy projection" of ξ on X_h. That is, ξ_h satisfies the following discrete variational problem:

$$(\Delta \xi_h, \Delta \psi_h)_\Omega = (1, \psi_h)_\Omega \ \ \forall\, \psi_h \in X_h. \tag{2.23}$$

The existence and uniqueness of the discrete solution $\xi_h \in X_h$ follows from the Lax-Milgram Theorem, see e.g., [6], [16]. Applying the discrete estimate (2.7) to the respective variational problems (2.22) and (2.23), we then have

$$|\Delta(\xi - \xi_h)|_\Omega \leq Ch^2. \tag{2.24}$$

With these ingredients, ξ and ξ_h, we then have

$$\begin{aligned} \sup_{\phi_h \in X_h} \frac{b(\phi_h, r)}{\|\phi_h\|_{H_0^2(\Omega)}} &\geq \frac{-r \int_\Omega [-\text{sgn}(r)] \xi_h \, d\Omega}{\|\xi_h\|_{H_0^2(\Omega)}} \\ &= \frac{|r| \int_\Omega \xi_h \cdot 1 \, d\Omega}{\|\xi_h\|_{H_0^2(\Omega)}} \\ &= |r| \big\|\xi_h\big\|_{H_0^2(\Omega)}, \end{aligned} \tag{2.25}$$

after using (2.23) above. Continuing, we then have

$$\begin{aligned}\sup_{\phi_h \in X_h} \frac{b(\phi_h, r)}{\|\phi_h\|_{H_0^2(\Omega)}} &\geq |r| \big\| \xi - (\xi - \xi_h) \big\|_{H_0^2(\Omega)} \\ &\geq |r| \big[\|\xi\|_{H_0^2(\Omega)} - \|\xi - \xi_h\|_{H_0^2(\Omega)} \big] \\ &\geq |r| \big[\|\xi\|_{H_0^2(\Omega)} - Ch^2 \big], \end{aligned} \tag{2.26}$$

after using the estimate (2.24). Taking step size parameter

$$h < \sqrt{\frac{\epsilon}{C}} \tag{2.27}$$

now completes the proof. $\square$

2.2 Regularity of Solution on Polyhedral Domain

The wellposedness arguments in Chapter 1 implicitly assume the domain is smooth. However the finite element setup assumes that the fluid chamber $\mathcal{O}$ is a polyhedron and the plate Ω is a polygon. The only thing that could be affected by this change is the elliptic regularity needed to guarantee that the solution of (1.34) achieves the regularity of $D(\mathcal{A})$. To address this we consider the following two results.

First, for the Stokes problems in (1.30) and (1.32) we refer to [17] which gives the necessary regularity (possibly modulo small $\epsilon > 0$ - see Theorem 18) for convex polyhedral domains with interior angles $\theta \leq 120^\circ$. (The result covers more possible geometries, but the domain of interest here is a convex polyhedron.)

Second, for the pressure problems in (1.19)-(1.20) the needed H^1-regularity follows from Lax-Milgram, so the polyhedral domain does not affect this result.

Finally, for the biharmonic problem in (1.53) we refer to [7] which handles polygonal domains. Because the result imposes a condition on the allowable interior angles in the domain we reproduce the result here. (Note that only the clamped conditions are in play

here so the result is not given in its full generality.)

Theorem 16. *Let Ω be a bounded polygonal domain. Assume that for fixed $k = 1$ the inner angle θ is smaller than $180°$, or for $k = 0$ the inner angle θ is smaller than $126.283696...°$ at each critical boundary point. Then for any given $f \in H^{-k}(\Omega)$ the weak solution of*

$$\Delta^2 w = f \quad in\ \Omega,\ w = \frac{\partial w}{\partial \nu} = 0 \quad on\ \partial\Omega \tag{2.28}$$

has the regularity $w \in H_0^2(\Omega) \cap H^{4-k}(\Omega)$ and satisfies the a priori estimate

$$\|w\|_{H^{4-k}(\Omega)} \leq C\|f\|_{H^{-k}}(\Omega). \tag{2.29}$$

Note that the cube geometry for the test problem in Section 2.6 satisfies the requirements of Theorem 16.

2.3 Error Estimates

We now turn to deriving an estimate on the error to be expected from implementing the discrete formulation described in Section 2.1. In particular we will produce the rate at which approximate solution should converge to the true solution with respect to the mesh parameter h.

In what follows we will have need of the following result in [21], which will not be stated here in its full generality (see [21], Lemma 2.44, p. 104).

Lemma 17. *With reference to the quantities in Theorem 2 above, let Σ_h be a subspace of Σ, and let M_h be a subspace of M. Suppose further that bilinear form $a : \Sigma \times \Sigma \to \mathbb{R}$ is Σ-elliptic; that is, $\exists\, \alpha > 0$ such that*

$$a(\sigma, \sigma) \geq \alpha\|\sigma\|_{\Sigma}^2 \quad \forall \sigma \in \Sigma. \tag{2.30}$$

Also, assume that the following "discrete inf-sup" condition is satisfied: $\exists\, \beta_h > 0$ *such that*

$$\inf_{q_h \in M_h} \sup_{\tau_h \in \Sigma_h} \frac{b(\tau_h, q_h)}{\|\tau_h\|_\Sigma \|q_h\|_M} \geq \beta_h, \tag{2.31}$$

where $\beta_h > 0$ *may depend upon subspaces* Σ_h *and* M_h*. Let moreover* $(\sigma_h, p_h) \in \Sigma_h \times M_h$ *solve the following (approximating) variational problem:*

$$\begin{cases} a(\sigma_h, \tau_h) + b(\tau_h, p_h) = (\kappa, \tau_h) & \forall \tau_h \in \Sigma_h, \\ b(\sigma_h, q_h) = (\ell, q_h) & \forall q_h \in M_h. \end{cases} \tag{2.32}$$

(Note that the existence and uniqueness of the solution pair (σ_h, p_h) *follows from Theorem 2, in view of (2.30) and (2.31).) Then one has the following error estimates:*

$$\|\sigma - \sigma_h\|_\Sigma \leq c_{1h} \inf_{\varsigma_h \in \Sigma_h} \|\sigma - \varsigma_h\|_\Sigma, + c_{2h} \inf_{q_h \in M_h} \|p - q_h\|_M \tag{2.33}$$

$$\|p - p_h\|_M \leq c_{3h} \inf_{\varsigma_h \in \Sigma_h} \|\sigma - \varsigma_h\|_\Sigma + c_{4h} \inf_{q_h \in M_h} \|p - q_h\|_M, \tag{2.34}$$

with $c_{1h} = (1 + \frac{\|a\|}{\alpha_h})(1 + \frac{\|b\|}{\beta_h})$*,* $c_{2h} = \frac{\|b\|}{\alpha_h}$*; moreover if* $M = M_h$ *one can take* $c_{2h} = 0$*,* $c_{3h} = c_{1h}\frac{\|a\|}{\beta_h}$*, and* $c_{4h} = 1 + \frac{\|b\|}{\beta_h} + c_{2h}\frac{\|a\|}{\beta_h}$*.*

Concerning the efficacy of our FEM for numerically approximating the fluid-structure system (1.35)-(1.41), we have the following:

Theorem 18. *Suppose that the fluid chamber* $\mathcal{O}$ *is a convex polyhedron with interior angles* $\theta \leq 120^\circ$ *and the structure domain* Ω *satisfies the conditions in Theorem 16. Let* $h > 0$ *be the parameter of discretization which gives rise to the FEM subspaces* V_h*,* Π_h*, and* X_h *of (2.1), (2.3), and (2.4), respectively. Also let* $[w_1^*, w_2^*, u^*] \in \mathbf{H}_0$ *be fixed. With respect to the solution variables* $[w_1, w_2, u, p] \in D(\mathcal{A}_\rho) \times H^1(\mathcal{O})$ *of (1.35)-(1.41) and their FEM approximations* $[w_{1h}, w_{2h}, u_h, p_h]$*, as given by (2.9) - (2.10) and (2.18)-(2.19), we have the following rates of convergence:*

(i) *(a) If* $\rho = 0$,

$$\|w_1 - w_{1h}\|_{H_0^2(\Omega)} \leq C_\lambda h^2 \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_0}. \tag{2.35}$$

(b) If $\rho > 0$,

$$\|w_1 - w_{1h}\|_{H_0^2(\Omega)} \leq C_\lambda h \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.36}$$

(ii) *(a′) If* $\rho = 0$,

$$\|w_2 - w_{2h}\|_{H_0^2(\Omega)} \leq C_\lambda h^2 \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_0}. \tag{2.37}$$

(b′) If $\rho > 0$,

$$\|w_2 - w_{2h}\|_{H_0^2(\Omega)} \leq C_\lambda h \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.38}$$

(iii) *For* $\rho \geq 0$,

$$\|u - u_h\|_{\mathbf{H}^1(\mathcal{O})} \leq C_\lambda h^{1-\epsilon} \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}, \tag{2.39}$$

In the special case that $div(\gamma_0^+(\lambda w_{1h} - w_1^*))$ *and* $div(\gamma_0^+(\lambda w_1 - w_1^*))$ *are zero at the singular points of* Ω *– see [17] –, then one actually may take* $\epsilon = 0$.

(iv) *For* $\rho \geq 0$,

$$\|p - p_h\|_{L^2(\mathcal{O})} \leq C_\lambda h^{1-\epsilon} \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.40}$$

Again, if $div(\gamma_0^+(\lambda w_{1h} - w_1^*))$ *and* $div(\gamma_0^+(\lambda w_1 - w_1^*))$ *are zero at the singular points of* Ω *one may have* $\epsilon = 0$.

Proof of Theorem 18. We first establish parts (i) and (ii) together. Here we will combine Lemma 17 with the bilinear forms in (1.43). We take in Lemma 17

$$a(\cdot, \cdot) \equiv a_\lambda(\cdot, \cdot) : H_0^2(\Omega) \times H_0^2(\Omega) \to \mathbb{R},$$

so $\alpha = 1$ and $\|a\| = \|a_\lambda\|_{\mathcal{L}\left([H_0^2(\Omega)]^2, \mathbb{R}\right)} \leq C_\lambda$. Moreover, we as before set

$$b(\cdot, \cdot) : H_0^2(\Omega) \times \mathbb{R} \to \mathbb{R} \text{ as in (1.43)}$$

so $\|b\|_{\mathcal{L}\left(H_0^2(\Omega)\times\mathbb{R},\mathbb{R}\right)} \leq C_\lambda$.

In addition, by Lemma 15, we can take

$$\beta_h \equiv C, \ \ \forall h > 0.$$

Subsequently, with reference to the variational system (1.42) and the approximating system (2.9) we have from (2.33)

$$\|w_1 - w_{1h}\|_{H_0^2(\Omega)} \leq C_\lambda \inf_{\psi_h \in X_h} \|w_1 - \psi_h\|_{H_0^2(\Omega)}; \tag{2.41}$$

(note that that second term from the right hand side of (2.33) is zero in this case because $M_h = M = \mathbb{R}$ in this case). Appealing now to estimates (2.7), (2.8), and Theorem 1(i) we have the following error estimates:

(a) If $\rho = 0$,

$$\begin{aligned} \|w_1 - w_{1h}\|_{H_0^2(\Omega)} &\leq C_\lambda h^2 |w_1|_{4,\Omega}, \\ &\leq C_\lambda h^2 \left\| [w_1^*, w_2^*, u^*] \right\|_{\mathbf{H}_0}. \end{aligned} \tag{2.42}$$

(b) If $\rho > 0$, then

$$\begin{aligned} \|w_1 - w_{1h}\|_{H_0^2(\Omega)} &\leq C_\lambda h |w_1|_{3,\Omega}, \\ &\leq C_\lambda h \left\| [w_1^*, w_2^*, u^*] \right\|_{\mathbf{H}_\rho}. \end{aligned} \tag{2.43}$$

This establishes Theorem 18(i). (Note that the regularity assumed above follows for polygonal domains from the references in Section 2.2. In view of (2.10), Theorem 18(ii) follows directly.

To achieve the estimates for the fluid variables, we first note that the error in the constant component of the pressure, given by the variational system (1.42), satisfies:

(a') If $\rho = 0$, we have upon combining (2.34) and (2.42)

$$|c - \tilde{c}_h| \leq C_\lambda h^2 \big\| [w_1^*, w_2^*, u^*] \big\|_{\mathbf{H}_0}. \tag{2.44}$$

(Implicity we are taking in (2.34) [finite dimensional] $\mathbb{R} = M_h$.)

(b') If $\rho > 0$, we have upon combining (2.34) and (2.43)

$$|c - \tilde{c}_h| \leq C_\lambda h \big\| [w_1^*, w_2^*, u^*] \big\|_{\mathbf{H}_\rho}. \tag{2.45}$$

(After again noting as above that in this case $M_h = M = \mathbb{R}$.)

Now we will again invoke Lemma 17, with respect to the Stokes component (1.38)-(1.41), with therein

$$a(\cdot, \cdot) \equiv \tilde{\mathbf{a}}_\lambda(\cdot, \cdot) : \mathbf{H}_0^1(\mathcal{O}) \times \mathbf{H}_0^1(\mathcal{O}) \to \mathbb{R} \text{ as given in (2.11).} \tag{2.46}$$

Consequently we can take $\alpha \equiv 1$ and $\|a\| = \|\tilde{\mathbf{a}}_\lambda\|_{\mathcal{L}\left([\mathbf{H}_0^1(\mathcal{O})]^2, \mathbb{R}\right)} \leq C_\lambda$. Moreover we take

$$b(\cdot, \cdot) \equiv \tilde{\mathbf{b}}(\cdot, \cdot) : \mathbf{H}_0^1(\mathcal{O}) \times \frac{L^2(\mathcal{O})}{\mathbb{R}} \to \mathbb{R}, \text{ as given in (2.12).} \tag{2.47}$$

Then $\|b\| = \|\tilde{\mathbf{b}}\|_{\mathcal{L}\left(\mathbf{H}_0^1(\mathcal{O}) \times \frac{L^2(\mathcal{O})}{\mathbb{R}}, \mathbb{R}\right)} \leq C$.

In addition, since the respective fluid and pressure spaces V_h and Π_h are piecewise quadratic and piecewise linear–i.e. the so-called Taylor-Hood formulation–then for $h > 0$ small enough we can take

$$\beta_h = \beta^*,$$

independent of small $h > 0$. (See e.g., Lemma 4.23, p. 193 of [21].)

Here, without loss of generality, we assume for all $h > 0$ that $\mathrm{div}(\gamma_0^+(\lambda w_{1h} - w_1^*))$ and $\mathrm{div}(\gamma_0^+(\lambda w_1 - w_1^*))$ are zero at the singular points of Ω – see [17] – else the estimate is degraded by small $\epsilon > 0$. Thus with u and u_h as given in (1.44) and (2.18) respectively,

we then have

$$\begin{aligned}\|u-u_h\|_{\mathbf{H}^1(\mathcal{O})} &\leq \|\tilde{f}(\lambda w_1 - w_1^*) - \tilde{f}_{0h}(\lambda w_{1h} - w_1^*) - \gamma_0^+(\lambda w_{1h} - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} \\ &\quad + \|\tilde{\mu}(u^*) - \tilde{\mu}_h(u^*)\|_{\mathbf{H}^1(\mathcal{O})} \\ &\leq \|\tilde{f}(\lambda w_1 - w_1^*) - \tilde{f}_{0h}(\lambda w_{1h} - w_1^*) - \gamma_0^+(\lambda w_{1h} - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} \\ &\quad + Ch\big(|u|_{2,\mathcal{O}} + \|p\|_{H^1(\mathcal{O})}\big) \end{aligned} \tag{2.48}$$

after using estimate (2.33) followed by (2.5) and (2.6).

Concerning the first term on the right hand side of (2.48) we have further

$$\begin{aligned}&\|\tilde{f}(\lambda w_1 - w_1^*) - \tilde{f}_{0h}(\lambda w_{1h} - w_1^*) - \gamma_0^+(\lambda w_{1h} - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} \\ &\quad \leq \|\tilde{f}(\lambda w_1 - w_1^*) - \tilde{f}_{0h}(\lambda w_1 - w_1^*) - \gamma_0^+(\lambda w_1 - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} \\ &\qquad + \|\gamma_0^+(\lambda w_1 - w_1^*) - \gamma_0^+(\lambda w_{1h} - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} + \|\tilde{f}_{0h}(\lambda w_1 - w_1^*) - \tilde{f}_{0h}(\lambda w_{1h} - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} \end{aligned} \tag{2.49}$$

We now estimate these terms one at at time.

Appealing again to the estimates in (2.5) and (2.6) via (2.33) (as well as to the regularity given in (1.31)) we have

$$\begin{aligned}&\big\|[\tilde{f}(\lambda w_1 - w_1^*) - \gamma_0^+(\lambda w_1 - w_1^*)] - \tilde{f}_{0h}(\lambda w_1 - w_1^*)\big\|_{\mathbf{H}^1(\mathcal{O})} \\ &\quad \leq Ch|\tilde{f}(\lambda w_1 - w_1^*) - \gamma_0^+(\lambda w_1 - w_1^*)|_{2,\mathcal{O}} \\ &\quad \leq Ch\|\lambda w_1 - w_1^*\|_{H^2(\Omega)} \end{aligned} \tag{2.50}$$

By the continuity of the right inverse of $\gamma_0 : \mathbf{H}^1(\mathcal{O}) \to \mathbf{H}^{1/2}(\partial\mathcal{O})$ and the estimate (2.42)

or (2.43), we also have

$$\|\gamma_0^+(\lambda w_1 - w_1^*) - \gamma_0^+(\lambda w_{1h} - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} \leq C_\lambda \|w_1 - w_{1h}\|_{H^1(\Omega)}$$
$$\leq C_\lambda h \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.51}$$

From (1.31) and (2.5) as well as either (2.42) or (2.43) we also have

$$\|\tilde{f}_{0h}(\lambda w_1 - w_1^*) - \tilde{f}_{0h}(\lambda w_{1h} - w_1^*)\|_{\mathbf{H}^1(\mathcal{O})} \leq C_\lambda \|w_1 - w_{1h}\|_{H^2(\Omega)}$$
$$\leq C_\lambda h \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.52}$$

Applying (2.49)-(2.52) to the right hand side of (2.48) (as well as considering the continuous dependence of the data inherent in Theorem 1(i),) we then have

$$\|u - u_h\|_{\mathbf{H}^1(\mathcal{O})} \leq C_\lambda h \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}, \tag{2.53}$$

which proves Theorem 18 (iii).

Finally, for the error in the fluid pressure term: by (1.44) and (2.19) we have

$$\|p - p_h\|_{L^2(\mathcal{O})} \leq \|\tilde{\pi}(\lambda w_1 - w_1^*) - \tilde{\pi}_h(\lambda w_{1h} - w_1^*)\|_{L^2(\mathcal{O})} + \|\tilde{q}(u^*) - \tilde{q}_h(u^*)\|_{L^2(\mathcal{O})}$$
$$+ C_\lambda h \big\|[w_1^*, w_2^*, u^*]\|_{\mathbf{H}_\rho} \tag{2.54}$$

after using (2.45). Proceeding just as in the proof of Theorem 18(iii) above results in the asserted estimate (2.67). This completes the proof of Theorem 18. $\square$

2.3.1 Further Error Estimates for the Implemented Problem

Looking closely at the bilinear form a_λ and the functional $\mathbb{F}$ which are defined in (1.43) and employed in (2.9) reveals that in order to implement the FEM to solve (2.9) one must be able to generate fluid solutions $\tilde{f}(\phi)$ and $\tilde{\mu}(\phi)$ for functions $\phi \in H_0^2(\Omega)$. These

cannot be known exactly and so every invocation of $\tilde{f}$ and $\tilde{\mu}$ inherently requires a corresponding numerical solution of a Stokes problem on the fluid domain. This means that in fact a_λ and $\mathbb{F}$ are not being computed exactly as was implicitly assumed in Theorem 18. However, as we will demonstrate below, this approximation does not affect the error estimates given in Theorem 18. This situation is wholly analogous to classical FEM implementation wherein error estimates typically do not take into account error accrued by numerical integration.

Note also that if the error on the structure terms w_1 and $\tilde{c}$ remains unaffected (see Theorem 18 (i) and (ii)) then the error in the fluid and pressure terms (see Theorem 18 (iii) and (iv)) will remain unaffected without further argument.

As before, let w_1 be the unique solution to (1.42); we seek to derive estimates on the error $\|w_1 - \hat{w}_{1h}\|_{H_0^2(\Omega)}$ of the finite element solution $\hat{w}_{1h}$ to the version of (2.9) as it is actually implemented numerically. More specifically let $[\hat{w}_{1h}, \hat{c}_h]$ solve:

$$\begin{cases} a_{\lambda h}(\hat{w}_{1h}, \psi_h) + b(\psi_h, \hat{c}_h) = \mathbb{F}_h(\psi_h), & \forall \psi_h \in X_h, \\ b(\hat{w}_{1h}, r) = 0, & \forall r \in \mathbb{R}. \end{cases} \tag{2.55}$$

where:

$$\begin{aligned} a_{\lambda h}(\psi, \phi) &= \lambda^2 (P_\rho^{1/2}\psi, P_\rho^{1/2}\phi)_\Omega + (\Delta\psi, \Delta\phi)_\Omega + \lambda(\nabla \tilde{f}_h(\psi), \nabla \tilde{f}_h(\phi))_{\mathcal{O}} + \lambda^2 (\tilde{f}_h(\psi), \tilde{f}_h(\phi))_{\mathcal{O}}, \\ &\qquad \forall\, \psi \text{ and } \phi \in H_0^2(\Omega); \\ b(\phi, r) &= -r \int_\Omega \phi \, d\Omega, \quad \forall \phi \in H_0^2(\Omega) \text{ and } r \in \mathbb{R}; \\ \mathbb{F}_h(\phi) &= (\nabla \tilde{f}_h(w_1^*), \nabla \tilde{f}_h(\phi))_{\mathcal{O}} + \lambda(\tilde{f}_h(w_1^*), \tilde{f}_h(\phi))_{\mathcal{O}} - (\nabla \tilde{\mu}_h(u^*), \nabla \tilde{f}_h(\phi))_{\mathcal{O}} \\ &\qquad - \lambda(\tilde{\mu}_h(u^*), \tilde{f}_h(\phi))_{\mathcal{O}} + (u^*, \tilde{f}_h(\phi))_{\mathcal{O}} + (P_\rho(\lambda w_1^* + w_2^*), \phi)_\Omega, \quad \forall \phi \in H_0^2(\Omega). \end{aligned} \tag{2.56}$$

and $\tilde{f}_h$ and $\tilde{\mu}_h$ simply refer to numerically implemented solutions of (1.30) and (1.32) respectively. We note here that again employing a combination of Lemma 17 and

Theorem 5.6, p. 224 of [6], for mesh parameter h we have the following $\mathbf{L}^2$-error estimates for $\tilde{f}_h$ and $\tilde{\mu}_h$:

$$\|\tilde{f}(\phi) - \tilde{f}_h(\phi)\|_{\mathbf{L}^2(\mathcal{O})} \leq Ch^2 \|\tilde{f}(\phi)\|_{\mathbf{H}^2(\mathcal{O})}, \tag{2.57}$$

$$\|\tilde{\mu}(u^*) - \tilde{\mu}_h(u^*)\|_{\mathbf{L}^2(\mathcal{O})} \leq Ch^2 \|\tilde{\mu}(u^*)\|_{\mathbf{H}^2(\mathcal{O})}. \tag{2.58}$$

As before we define

$$\hat{w}_{2h} = \lambda \hat{w}_{1h} - w_1^*. \tag{2.59}$$

Finally the fluid and pressure variables are given through

$$\hat{u}_h = \tilde{f}_{0h}(\lambda \hat{w}_{1h} - w_1^*) + \gamma_0^+(\lambda \hat{w}_{1h} - w_1^*) + \tilde{\mu}_h(u^*); \tag{2.60}$$

$$\hat{p}_h = \tilde{\pi}_h(\lambda \hat{w}_{1h} - w_1^*) + \tilde{q}_h(u^*) + \tilde{c}_h^*, \tag{2.61}$$

We have then the following corollary to Theorem 18.

Corollary 19. *Suppose that the fluid chamber $\mathcal{O}$ is a convex polyhedron with interior angles $\theta \leq 120^\circ$ and the structure domain Ω satisfies the conditions in Theorem 16. Let $h > 0$ be the parameter of discretization which gives rise to the FEM subspaces V_h, Π_h, and X_h of (2.1), (2.3), and (2.4), respectively. Also let $[w_1^*, w_2^*, u^*] \in \mathbf{H}_0$ be fixed. Let $h > 0$ be the parameter of discretization which gives rise to the FEM subspaces V_h, Π_h, and X_h of (2.1), (2.3), and (2.4), respectively. With respect to the solution variables $[w_1, w_2, u, p] \in D(\mathcal{A}_\rho) \times H^1(\mathcal{O})$ of (1.35)-(1.41) and their FEM approximations $[\hat{w}_{1h}, \hat{w}_{2h}, \hat{u}_h, \hat{p}_h]$, as given by (2.55), (2.59) and (2.60)-(2.61), we have the following rates of convergence:*

(i) (a) If $\rho = 0$,

$$\|w_1 - \hat{w}_{1h}\|_{H_0^2(\Omega)} \leq C_\lambda h^2 \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_0}. \tag{2.62}$$

(b) If $\rho > 0$,

$$\|w_1 - \hat{w}_{1h}\|_{H_0^2(\Omega)} \leq C_\lambda h \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.63}$$

(ii) *(a′) If* $\rho = 0$,

$$\|w_2 - \hat{w}_{2h}\|_{H_0^2(\Omega)} \leq C_\lambda h^2 \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_0}. \tag{2.64}$$

(b′) If $\rho > 0$,

$$\|w_2 - \hat{w}_{2h}\|_{H_0^2(\Omega)} \leq C_\lambda h \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.65}$$

(iii) For $\rho \geq 0$,

$$\|u - \hat{u}_h\|_{\mathbf{H}^1(\mathcal{O})} \leq C_\lambda h^{1-\epsilon} \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}, \tag{2.66}$$

In the special case that $div(\gamma_0^+(\lambda w_{1h} - w_1^*))$ *and* $div(\gamma_0^+(\lambda w_1 - w_1^*))$ *are zero at the singular points of* Ω *– see [17] –, then one actually may take* $\epsilon = 0$.

(iv) For $\rho \geq 0$,

$$\|p - \hat{p}_h\|_{L^2(\mathcal{O})} \leq C_\lambda h^{1-\epsilon} \big\|[w_1^*, w_2^*, u^*]\big\|_{\mathbf{H}_\rho}. \tag{2.67}$$

Again, if $div(\gamma_0^+(\lambda w_{1h} - w_1^*))$ *and* $div(\gamma_0^+(\lambda w_1 - w_1^*))$ *are zero at the singular points of* Ω *one may have* $\epsilon = 0$.

Proof of Corollary 19. We will demonstrate (i) only as the remaining parts all follow directly by parallel arguments to those in Theorem 18 once (i) is in hand. By the triangle inequality, $\|w_1 - \hat{w}_{1h}\|_{H_0^2(\Omega)} \leq \|w_1 - w_{1h}\|_{H_0^2(\Omega)} + \|w_{1h} - \hat{w}_{1h}\|_{H_0^2(\Omega)}$. The error in the first term is given in Theorem 18, so it remains to control the error arising from the perturbation inherent in $a_{\lambda h}$ and $\mathbb{F}_h$ via the second term $\|w_{1h} - \hat{w}_{1h}\|_{H_0^2(\Omega)}$. Since the norm in $H_0^2(\Omega)$ is dictated by the laplacian, we have in particular for the solution

$w_{1h} \in X_h$ to (2.9):

$$\begin{aligned}
\|\hat{w}_{1h} - w_{1h}\|^2_{H_0^2(\Omega)} &\leq a_{\lambda h}(\hat{w}_{1h} - w_{1h}, \hat{w}_{1h} - w_{1h}) \\
&= a_{\lambda h}(\hat{w}_{1h}, \hat{w}_{1h} - w_{1h}) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - w_{1h}) \\
&= \mathbb{F}_h(\hat{w}_{1h} - w_{1h}) - b(\hat{w}_{1h} - w_{1h}, \hat{c}_h) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - w_{1h}) \\
&= [a_\lambda(w_{1h}, \hat{w}_{1h} - w_{1h}) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - \psi)] \\
&\quad + [\mathbb{F}_h(\hat{w}_{1h} - w_{1h}) - \mathbb{F}(\hat{w}_{1h} - w_{1h})] + [-b(\hat{w}_{1h} - w_{1h}, \hat{c}_h) \\
&\quad + b(\hat{w}_{1h} - w_{1h}, \tilde{c}_h)]
\end{aligned}$$

after applying (2.55) followed by (2.9) and rearranging terms. Taking the absolute value of the right hand side then gives

$$\begin{aligned}
\|\hat{w}_{1h} - w_{1h}\|^2_{H_0^2(\Omega)} &\leq |a_\lambda(w_{1h}, \hat{w}_{1h} - w_{1h}) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - w_{1h})| \\
&\quad + |\mathbb{F}_h(\hat{w}_{1h} - w_{1h}) - \mathbb{F}(\hat{w}_{1h} - w_{1h})| + |b(\hat{w}_{1h} - w_{1h}, \hat{c}_h - c_h)| \\
&= |a_\lambda(w_{1h}, \hat{w}_{1h} - w_{1h}) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - w_{1h})| \\
&\quad + |\mathbb{F}_h(\hat{w}_{1h} - w_{1h}) - \mathbb{F}(\hat{w}_{1h} - w_{1h})|
\end{aligned}$$

after applying (2.9) and (2.55) to $b(\hat{w}_{1h} - w_{1h}, c_h - \hat{c}_h)$. We have to handle the last line term by term via the definitions in (1.42)-(1.43) and (2.55)-(2.56). We first consider the term $|a_\lambda(w_{1h}, \hat{w}_{1h} - w_{1h}) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - w_{1h})|$. Expanding the terms via the definitions

in (1.43) and (2.56) gives:

$$
\begin{aligned}
&|a_\lambda(w_{1h}, \hat{w}_{1h} - w_{1h}) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - w_{1h})| \\
&= \big|\lambda(\nabla \tilde{f}(w_{1h}), \nabla \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} + \lambda^2(\tilde{f}(w_{1h}), \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\quad - \lambda(\nabla \tilde{f}_h(w_{1h}), \nabla \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - \lambda^2(\tilde{f}_h(w_{1h}), \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&= \big|\lambda(\nabla[\tilde{f}(w_{1h}) - \tilde{f}_h(w_{1h})], \nabla \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} + \lambda^2(\tilde{f}(w_{1h}) - \tilde{f}_h(w_{1h}), \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\quad + \lambda(\nabla \tilde{f}_h(w_{1h}), \nabla[\tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h})])_{\mathcal{O}} \\
&\quad + \lambda^2(\tilde{f}_h(w_{1h}), \tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&= \big|\lambda(\tilde{f}_h(w_{1h}) - \tilde{f}(w_{1h}), \Delta \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} + \lambda^2(\tilde{f}(w_{1h}) - \tilde{f}_h(w_{1h}), \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\quad + \lambda(\Delta \tilde{f}_h(w_{1h}), \tilde{f}_h(\hat{w}_{1h} - w_{1h}) - \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\quad + \lambda^2(\tilde{f}_h(w_{1h}), \tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&\leq C_\lambda\big[\|\tilde{f}_h(w_{1h}) - \tilde{f}(w_{1h})\|_{\mathbf{L}^2(\mathcal{O})}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)} \\
&\quad + \|w_{1h}\|_{H_0^2(\Omega)}\|\tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})}\big]
\end{aligned}
$$

The last line follows from using Hölder's inequality and the continuity of the solution map $\tilde{f}$. Now we apply the $\mathbf{L}^2(\mathcal{O})$ error estimate for the Stokes problem $\tilde{f}$ given in (2.57) and the inequality (1.31) to achieve

$$
\begin{aligned}
&|a_\lambda(w_{1h}, \hat{w}_{1h} - w_{1h}) - a_{\lambda h}(w_{1h}, \hat{w}_{1h} - w_{1h})| \\
&\leq C_\lambda h^2\big[\|w_{1h}\|_{H_0^2(\Omega)}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)} + \|w_{1h}\|_{H_0^2(\Omega)}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)}\big].
\end{aligned} \tag{2.68}
$$

The terms in $|\mathbb{F}_h(\hat{w}_{1h} - w_{1h}) - \mathbb{F}(\hat{w}_{1h} - w_{1h})|$ are handled in similar fashion. First recall

that

$$
\begin{aligned}
&|\mathbb{F}_h(\hat{w}_{1h} - w_{1h}) - \mathbb{F}(\hat{w}_{1h} - w_{1h})| \\
&\qquad = \big|(\nabla \tilde{f}_h(w_1^*), \nabla \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - (\nabla \tilde{f}(w_1^*), \nabla \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\qquad\quad + \lambda(\tilde{f}_h(w_1^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - \lambda(\tilde{f}(w_1^*), \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\qquad\quad - (\nabla \tilde{\mu}_h(u^*), \nabla \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} + (\nabla \tilde{\mu}(u^*), \nabla \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\qquad\quad - \lambda(\tilde{\mu}_h(u^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} + \lambda(\tilde{\mu}(u^*), \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\qquad\quad + (u^*, \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - (u^*, \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big|.
\end{aligned}
$$

We handle these terms one by one below. For the first term we add and subtract the cross term $(\nabla \tilde{f}(w_1^*), \nabla \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}$, rearrange terms and integrate by parts (noting the zero boundary condition and regularity given by (1.31)) and then apply (2.57).

$$
\begin{aligned}
&\big|(\nabla \tilde{f}_h(w_1^*), \nabla \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - (\nabla \tilde{f}(w_1^*), \nabla \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&\quad = \big|(\nabla(\tilde{f}_h(w_1^*) - \tilde{f}(w_1^*)), \nabla \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} \\
&\qquad - (\nabla \tilde{f}(w_1^*), \nabla(\tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h})))_{\mathcal{O}}\big| \\
&\quad = \big|(\tilde{f}_h(w_1^*) - \tilde{f}(w_1^*), \Delta \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - (\Delta \tilde{f}(w_1^*), \tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&\quad \leq \|\tilde{f}_h(w_1^*) - \tilde{f}(w_1^*)\|_{\mathbf{L}^2(\mathcal{O})} \|\Delta \tilde{f}_h(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\qquad + \|\Delta \tilde{f}(w_1^*)\|_{\mathbf{L}^2(\mathcal{O})} \|\tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\quad \leq Ch^2 \|\tilde{f}(w_1^*)\|_{\mathbf{H}^2(\mathcal{O})} \|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)} + Ch^2 \|w_1^*\|_{H_0^2(\Omega)} \|\tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{H}^2(\mathcal{O})} \\
&\quad \leq Ch^2 \|w_1^*\|_{H_0^2(\Omega)} \|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)}.
\end{aligned} \tag{2.69}
$$

In the second term we add and subtract the cross term $\lambda(\tilde{f}(w_1^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}$ and

proceed as above to the following:

$$
\begin{aligned}
&\big|\lambda(\tilde{f}_h(w_1^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - \lambda(\tilde{f}(w_1^*), \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&\quad = \big|\lambda(\tilde{f}_h(w_1^*) - \tilde{f}(w_1^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - \lambda(\tilde{f}(w_1^*), \tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&\quad \leq \lambda\|(\tilde{f}_h(w_1^*) - \tilde{f}(w_1^*)\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}_h(\hat{w}_{1h} - w_{1h}))\|_{\mathbf{L}^2(\mathcal{O})} \\
&\qquad + \lambda\|\tilde{f}(w_1^*)\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\quad \leq C_\lambda h^2\|\tilde{f}(w_1^*)\|_{\mathbf{H}^2(\mathcal{O})}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)} + C_\lambda h^2\|w_1^*\|_{H_0^2(\Omega)}\|\tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{H}^2(\mathcal{O})} \\
&\quad \leq C_\lambda h^2\|w_1^*\|_{H_0^2(\Omega)}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)}. \qquad (2.70)
\end{aligned}
$$

In the third term, after adding and subtracting the cross term $(\nabla\tilde{\mu}_h(u^*), \nabla\tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}$ we integrate by parts and use Hölder's inequality, (1.33) and the estimate (2.57) to yield:

$$
\begin{aligned}
&\big|\big(\nabla\tilde{\mu}_h(u^*), \nabla\tilde{f}_h(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}} - \big(\nabla\tilde{\mu}(u^*), \nabla\tilde{f}(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}}\big| \\
&\quad = \big|\big(\nabla\tilde{\mu}_h(u^*), \nabla(\tilde{f}_h(\hat{w}_{1h} - w_{1h}) - \tilde{f}(\hat{w}_{1h} - w_{1h}))\big)_{\mathcal{O}} \\
&\qquad - \big(\nabla(\tilde{\mu}(u^*) - \tilde{\mu}_h(u^*)), \nabla\tilde{f}(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}}\big| \\
&\quad = \big|\big(\Delta\tilde{\mu}_h(u^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h}) - \tilde{f}(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}} \\
&\qquad - \big(\tilde{\mu}(u^*) - \tilde{\mu}_h(u^*), \Delta\tilde{f}(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}}\big| \\
&\quad \leq \|\Delta\tilde{\mu}_h(u^*)\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}_h(\hat{w}_{1h} - w_{1h}) - \tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\qquad + \|\tilde{\mu}(u^*) - \tilde{\mu}_h(u^*)\|_{\mathbf{L}^2(\mathcal{O})}\|\Delta\tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\quad \leq Ch^2\|u^*\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{H}^2(\mathcal{O})} + Ch^2\|\tilde{\mu}(u^*)\|_{\mathbf{H}^2(\mathcal{O})}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)} \\
&\quad \leq Ch^2\|u^*\|_{\mathbf{L}^2(\mathcal{O})}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)}. \qquad (2.71)
\end{aligned}
$$

For the fourth term we add and subtract $\lambda(\tilde{\mu}_h(u^*), \tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}$, use Hölder's

inequality, the estimates (2.57)-(2.58), and the regularity in (1.33) to derive:

$$
\begin{aligned}
&\big|\lambda\big(\tilde{\mu_h}(u^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}} - \lambda\big(\tilde{\mu}(u^*), \tilde{f}(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}}\big| \\
&\quad = \big|\lambda\big(\tilde{\mu_h}(u^*), \tilde{f}_h(\hat{w}_{1h} - w_{1h}) - \tilde{f}(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}} - \lambda\big(\tilde{\mu}(u^*) - \tilde{\mu}_h(u^*), \tilde{f}(\hat{w}_{1h} - w_{1h})\big)_{\mathcal{O}}\big| \\
&\quad \leq \lambda\|\tilde{\mu_h}(u^*)\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}_h(\hat{w}_{1h} - w_{1h}) - \tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\qquad + \lambda\|\tilde{\mu}(u^*) - \tilde{\mu}_h(u^*)\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\quad \leq C_\lambda h^2\|u^*\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} + C_\lambda h^2\|\tilde{\mu}(u^*)\|_{\mathbf{H}^2(\mathcal{O})}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)} \\
&\quad \leq C_\lambda h^2\|u^*\|_{\mathbf{L}^2(\mathcal{O})}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)}. \qquad (2.72)
\end{aligned}
$$

Finally, on the fifth term it remains to apply Hölder's inequality, the estimate (2.57) and the continuous dependence on data from (1.31) to give:

$$
\begin{aligned}
&\big|(u^*,\tilde{f}(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}} - (u^*, \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&\qquad = \big|(u^*, \tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h}))_{\mathcal{O}}\big| \\
&\qquad \leq \|u^*\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}(\hat{w}_{1h} - w_{1h}) - \tilde{f}_h(\hat{w}_{1h} - w_{1h})\|_{\mathbf{L}^2(\mathcal{O})} \\
&\qquad \leq Ch^2\|u^*\|_{\mathbf{L}^2(\mathcal{O})}\|\tilde{f}(\hat{w}_{1h} - w_{1h})\|_{\mathbf{H}^2(\mathcal{O})} \\
&\qquad \leq Ch^2\|u^*\|_{\mathbf{L}^2(\mathcal{O})}\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)}. \qquad (2.73)
\end{aligned}
$$

Combining (2.68) - (2.73) together and dividing through by $\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)}$ yields

$$
\|\hat{w}_{1h} - w_{1h}\|_{H_0^2(\Omega)} \leq C_\lambda h^2\big\|[u^*, w_1^*, w_2^*]\big\|_{\mathbf{H}_\rho}. \qquad (2.74)
$$

Thus the error in the numerically implemented solution $\hat{w}_{1h}$ is no worse than that given in Theorem 18 for w_{1h}. This establishes (i) which together with the remark above completes the proof of Corollary 19. $\square$

2.4 FEM Elements

The goal of this section is to explain some of the important properties of the finite element basis functions used in the numerical scheme as well as some mechanics of how they are employed in the finite element method.

2.4.1 Argyris Elements

We begin with the H^2-conforming Argyris elements which are used on the structure domain Ω. This weak formulation takes place over $H_0^2(\Omega)$ and thus the most natural choice for the discretization is a set of H^2-conforming elements, see [33]; for a MATLAB implementation see [18]. We use the quintic Argyris basis functions because they are the lowest order H^2 conforming elements available and they ensure wellposedness of the discrete formulation of (1.42) as was shown in Section 2.1. The Argyris basis functions have 21 degrees of freedom(DOF) for each triangle in the mesh, namely Lagrange DOF for function values at each vertex, Hermite DOF for $\partial/\partial x$ and $\partial/\partial y$ at each vertex, Argyris DOF for $\partial^2/\partial x^2$, $\partial^2/\partial x \partial y$, and $\partial^2/\partial y^2$ at each vertex and one DOF at each edge midpoint for the normal derivative.

The numerical construction of the basis functions in the Matlab code follows the abstract setup in [33] closely as our numerical implementation did. The more interested reader can see [18] and [22] for a more detailed MATLAB focused implementation. Here we include a relatively brief outline. The Argyris elements are $\mathcal{C}^1$ across the boundaries of the elements; that is basis function values, 1st derivatives and 2nd derivatives all must match at the vertices and the normal derivatives must match at the edge midpoints. To achieve this the basis functions are constructed in steps. First twenty-one shape functions are built on the reference triangle with vertices $(0,0), (1,0)$, and $(0,1)$, see Figure 2.1 below. (Note that the diagram uses the standard notation of "$\cdot$" indicating the function value, "$\circ$" indicating the first derivatives, "$\bigcirc$" indicating the second

derivatives, and "↗" indicating the normal derivative at a point.)

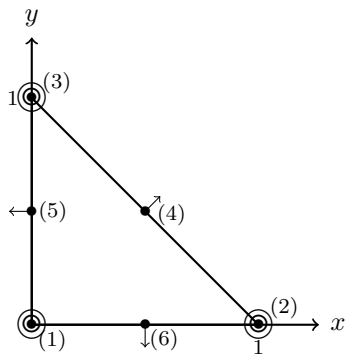


Figure 2.1: The Argyris Reference Element.

The shape functions are polynomials of degree 5 and are listed in Table 2.1.

$$\psi_1 = 1 - 10x^3 - 10y^3 + 15x^4 - 30x^2y^2 + 15y^4 - 6x^5 + 30x^3y^2 + 30x^2y^3 - 6y^5$$

$$\psi_2 = x - 6x^3 - 11xy^2 + 8x^4 + 10x^2y^2 + 18xy^3 - 3x^5 + x^3y^2 - 10x^2y^3 - 8xy^4$$

$$\psi_3 = y - 11x^2y - 6y^3 + 18x^3y + 10x^2y^2 + 8y^4 - 8x^4y - 10x^3y^2 + x^2y^3 - 3y^5$$

$$\psi_4 = 0.5x^2 - 1.5x^3 + 1.5x^4 - 1.5x^2y^2 - 0.5x^5 + 1.5x^3y^2 + x^2y^3$$

$$\psi_5 = xy - 4x^2y - 4xy^2 + 5x^3y + 10x^2y^2 + 5xy^3 - 2x^4y - 6x^3y^2 - 6x^2y^3 - 2xy^4$$

$$\psi_6 = 0.5y^2 - 1.5y^3 - 1.5x^2y^2 + 1.5y^4 + x^3y^2 + 1.5x^2y^3 - 0.5y^5$$

$$\psi_7 = 10x^3 - 15x^4 + 15x^2y^2 + 6x^5 - 15x^3y^2 - 15x^2y^3$$

$$\psi_8 = -4x^3 + 7x^4 - 3.5x^2y^2 - 3x^5 + 3.5x^3y^2 + 3.5x^2y^3$$

$$\psi_9 = -5x^2y + 14x^3y + 18.5x^2y^2 - 8x^4y - 18.5x^3y^2 - 13.5x^2y^3$$

$$\psi_{10} = 0.5x^3 - x^4 + 0.25x^2y^2 + 0.5x^5 - 0.25x^3y^2 - 0.25x^2y^3$$

$$\psi_{11} = x^2y - 3x^3y - 3.5x^2y^2 + 2x^4y + 3.5x^3y^2 + 2.5x^2y^3$$

$$\psi_{12} = 1.25x^2y^2 - 0.75x^3y^2 - 1.25x^2y^3$$

$$\psi_{13} = 10y^3 + 15x^2y^2 - 15y^4 - 15x^3y^2 - 15x^2y^3 + 6y^5$$

$$\psi_{14} = -5xy^2 + 18.5x^2y^2 + 14xy^3 - 13.5x^3y^2 - 18.5x^2y^3 - 8xy^4$$

$$\psi_{15} = -4y^3 - 3.5x^2y^2 + 7y^4 + 3.5x^3y^2 + 3.5x^2y^3 - 3y^5$$

$$\psi_{16} = 1.25x^2y^2 - 1.25x^3y^2 - 0.75x^2y^3$$

$$\psi_{17} = xy^2 - 3.5x^2y^2 - 3xy^3 + 2.5x^3y^2 + 3.5x^2y^3 + 2xy^4$$

$$\psi_{18} = 0.5y^3 + 0.25x^2y^2 - y^4 - 0.25x^3y^2 - 0.25x^2y^3 + 0.5y^5$$

$$\psi_{19} = -16x^2y + 32x^3y + 32x^2y^2 - 16x^4y - 32x^3y^2 - 16x^2y^3$$

$$\psi_{20} = \sqrt{2}(-8x^2y^2 + 8x^3y^2 + 8x^2y^3)$$

$$\psi_{21} = -16xy^2 + 32x^2y^2 + 32xy^3 - 16x^3y^2 - 32x^2y^3 - 16xy^4$$

Table 2.1: Argyris Shape Functions for the Plate.

The shape functions each specify exactly one of the degrees of freedom illustrated in Figure 2.1. For example, ψ_1 satisfies $\psi_1(0,0) = 1$, but $\psi_1(1,0) = 0$ and $\psi_1(0,1) = 0$. Moreover, the first and second derivatives $\partial_x, \partial_y, \partial_{xx}, \partial_{xy}$, and ∂_{yy} of ψ_1 are all zero at the three vertices and the normal derivative $\partial/\partial\nu$ of ψ_1 is zero at the three edge midpoints.

More generally, we define the following linear functionals for $\phi \in \mathcal{C}^2(\Omega)$, (as in [18])

$$\begin{aligned}
\mathcal{L}_i^0(\phi) &= \phi(\mathbf{x}_i), \\
\mathcal{L}_i^x(\phi) &= \phi_x(\mathbf{x}_i), \\
\mathcal{L}_i^y(\phi) &= \phi_y(\mathbf{x}_i), \\
\mathcal{L}_i^{xx}(\phi) &= \phi_{xx}(\mathbf{x}_i), \\
\mathcal{L}_i^{xy}(\phi) &= \phi_{xy}(\mathbf{x}_i), \\
\mathcal{L}_i^{yy}(\phi) &= \phi_{yy}(\mathbf{x}_i), \\
\mathcal{L}_i^{\nu}(\phi) &= \frac{\partial}{\partial \nu}\phi(\mathbf{m}_i),
\end{aligned}$$

where $\mathbf{x}_i$ correspond to mesh triangle vertices and $\mathbf{m}_i$ correspond to mesh triangle midpoints. On the reference triangle $\mathbf{x}_0 = (0,0)$, $\mathbf{x}_1 = (1,0)$, and $\mathbf{x}_2 = (0,1)$; $\mathbf{m}_0 = (.5,0)$, $\mathbf{m}_1 = (.5,.5)$, $\mathbf{m}_2 = (0,.5)$. For convenience we make the enumeratation

$$\begin{aligned}
\mathcal{L}_0^0 &= \mathcal{L}_1, & \mathcal{L}_1^0 &= \mathcal{L}_7, & \mathcal{L}_2^0 &= \mathcal{L}_{13}, \\
\mathcal{L}_0^x &= \mathcal{L}_2, & \mathcal{L}_1^x &= \mathcal{L}_8, & \mathcal{L}_2^x &= \mathcal{L}_{14}, \\
\mathcal{L}_0^y &= \mathcal{L}_3, & \mathcal{L}_1^y &= \mathcal{L}_9, & \mathcal{L}_2^y &= \mathcal{L}_{15}, \\
\mathcal{L}_0^{xx} &= \mathcal{L}_4, & \mathcal{L}_1^{xx} &= \mathcal{L}_{10}, & \mathcal{L}_2^{xx} &= \mathcal{L}_{16}, \\
\mathcal{L}_0^{xy} &= \mathcal{L}_5, & \mathcal{L}_1^{xy} &= \mathcal{L}_{11}, & \mathcal{L}_2^{xy} &= \mathcal{L}_{17}, \\
\mathcal{L}_0^{yy} &= \mathcal{L}_6, & \mathcal{L}_1^{yy} &= \mathcal{L}_{12}, & \mathcal{L}_2^{yy} &= \mathcal{L}_{18},
\end{aligned}$$

and

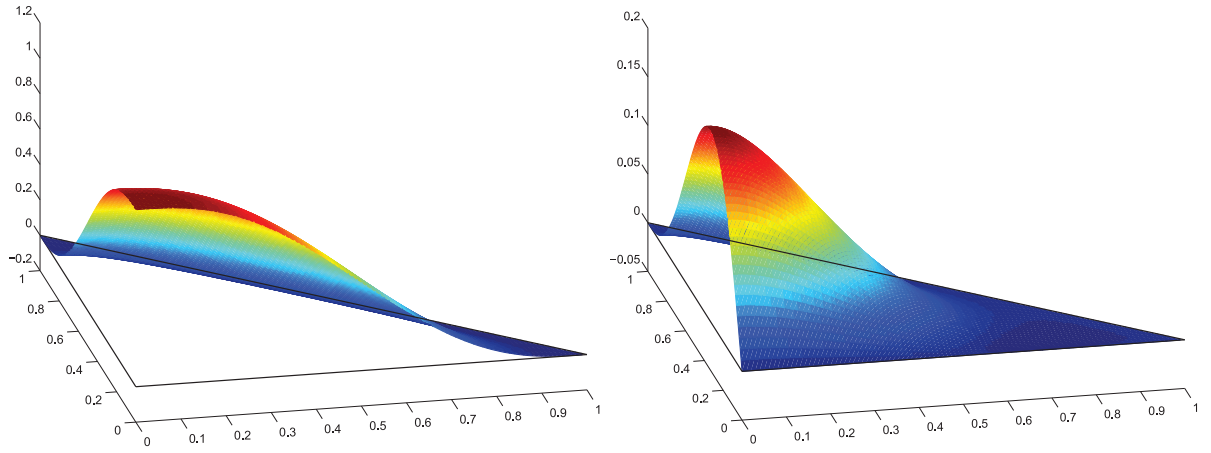
$$\mathcal{L}_0^{\nu} = \mathcal{L}_{19}, \qquad \mathcal{L}_1^{\nu} = \mathcal{L}_{20}, \qquad \mathcal{L}_2^{\nu} = \mathcal{L}_{21}.$$

These represent the degrees of freedom described above. With this enumeration the shape functions satisfy $\mathcal{L}_i(\psi_j) = \delta_{ij}$ for $1 \leq i, j \leq 21$ where δ_{ij} is the Kronecker delta

function. For clarity when applied to the shape functions this enumeration implies that:

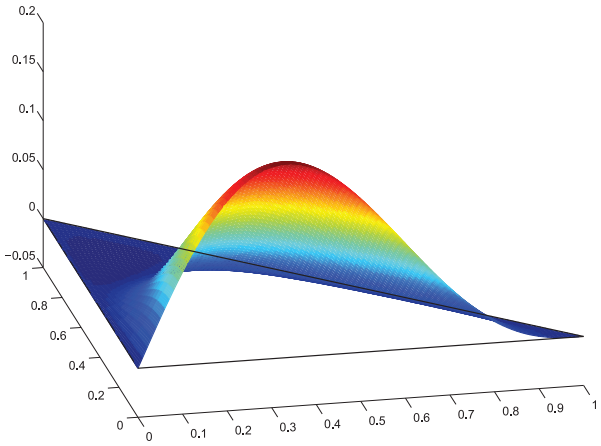
$$\begin{aligned}
\psi_1(0,0) &= 1, & \partial_x\psi_2(0,0) &= 1, & \partial_y\psi_3(0,0) &= 1,\\
\partial_{xx}\psi_4(0,0) &= 1, & \partial_{xy}\psi_5(0,0) &= 1, & \partial_{yy}\psi_6(0,0) &= 1,\\
\psi_7(1,0) &= 1, & \partial_x\psi_8(1,0) &= 1, & \partial_y\psi_9(1,0) &= 1,\\
\partial_{xx}\psi_{10}(1,0) &= 1, & \partial_{xy}\psi_{11}(1,0) &= 1, & \partial_{yy}\psi_{12}(1,0) &= 1,\\
\psi_{13}(0,1) &= 1, & \partial_x\psi_{14}(0,1) &= 1, & \partial_y\psi_{15}(0,1) &= 1,\\
\partial_{xx}\psi_{16}(0,1) &= 1, & \partial_{xy}\psi_{17}(0,1) &= 1, & \partial_{yy}\psi_{18}(0,1) &= 1,\\
\partial_\nu\psi_{19}(.5,0) &= 1, & \partial_\nu\psi_{20}(.5,.5) &= 1, & \partial_\nu\psi_{21}(0,.5) &= 1,
\end{aligned}$$

and that the value of these functions are zero for any other evaluation other than the ones shown above. The graphs of these shape functions over the reference domain are plotted in the figures below.

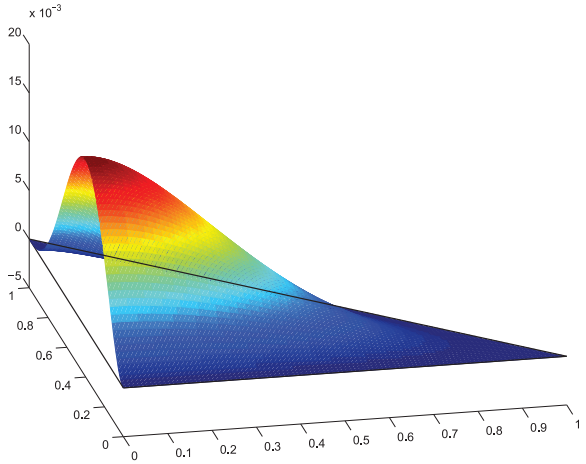


(a) ψ_1

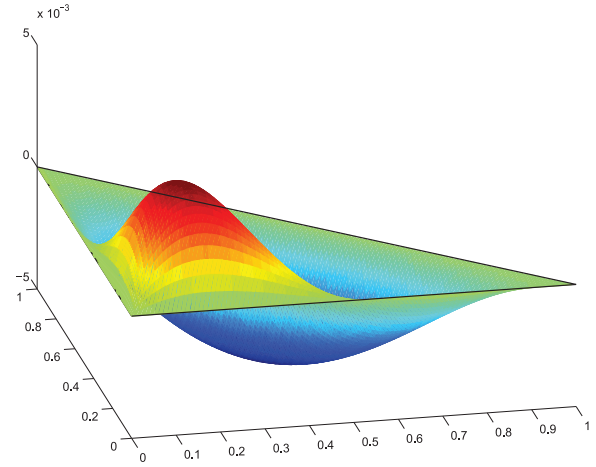
(b) ψ_2



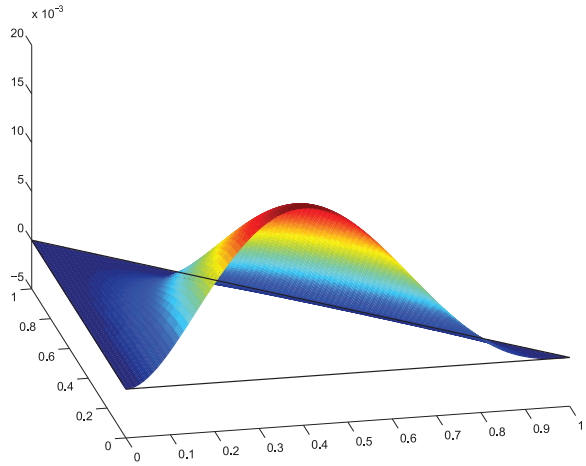
(c) ψ_3



(d) ψ_4

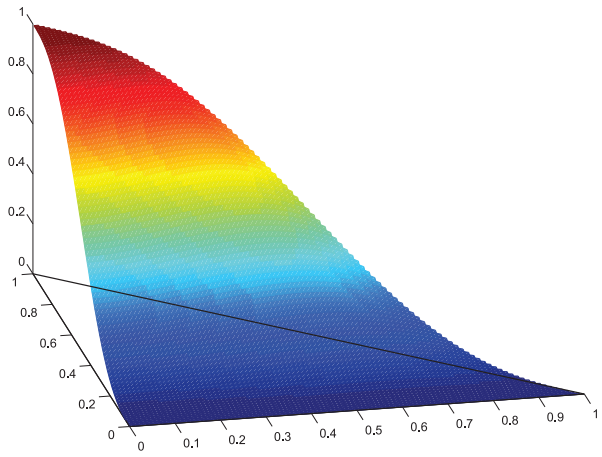


(e) ψ_5

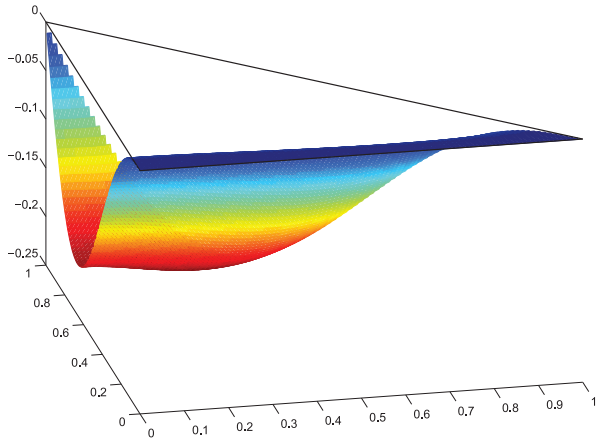


(f) ψ_6

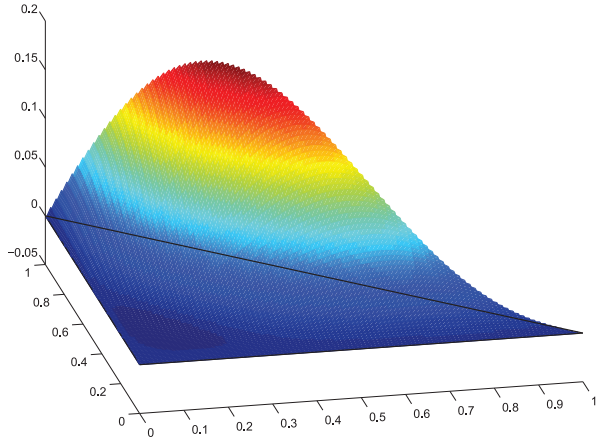
Figure 2.2: The Argyris shape functions for the vertex (0,0).



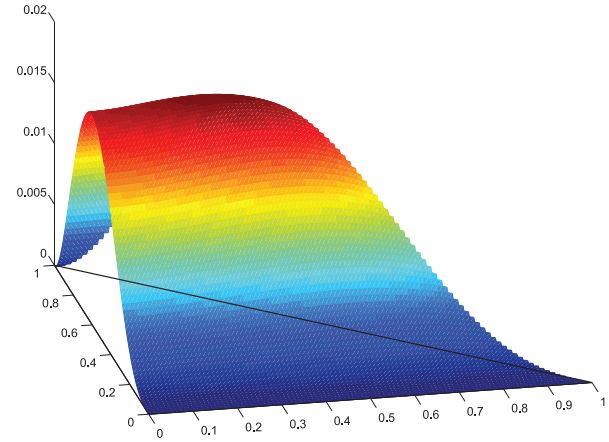
(a) ψ_7



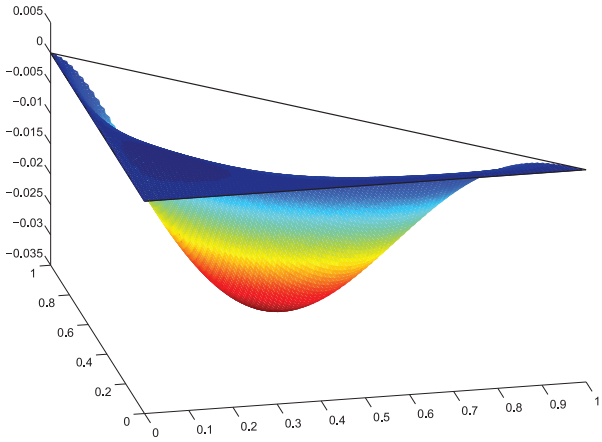
(b) ψ_8



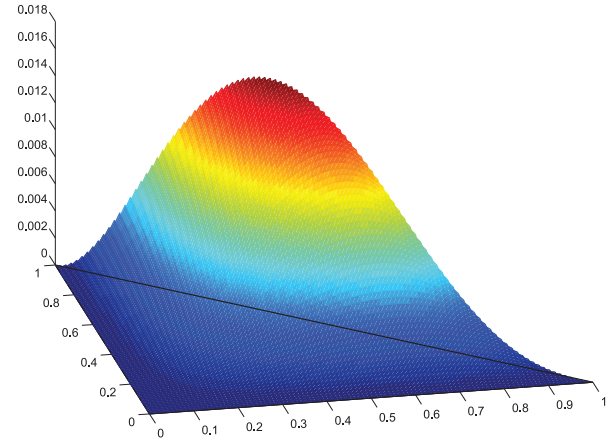
(c) ψ_9



(d) ψ_{10}

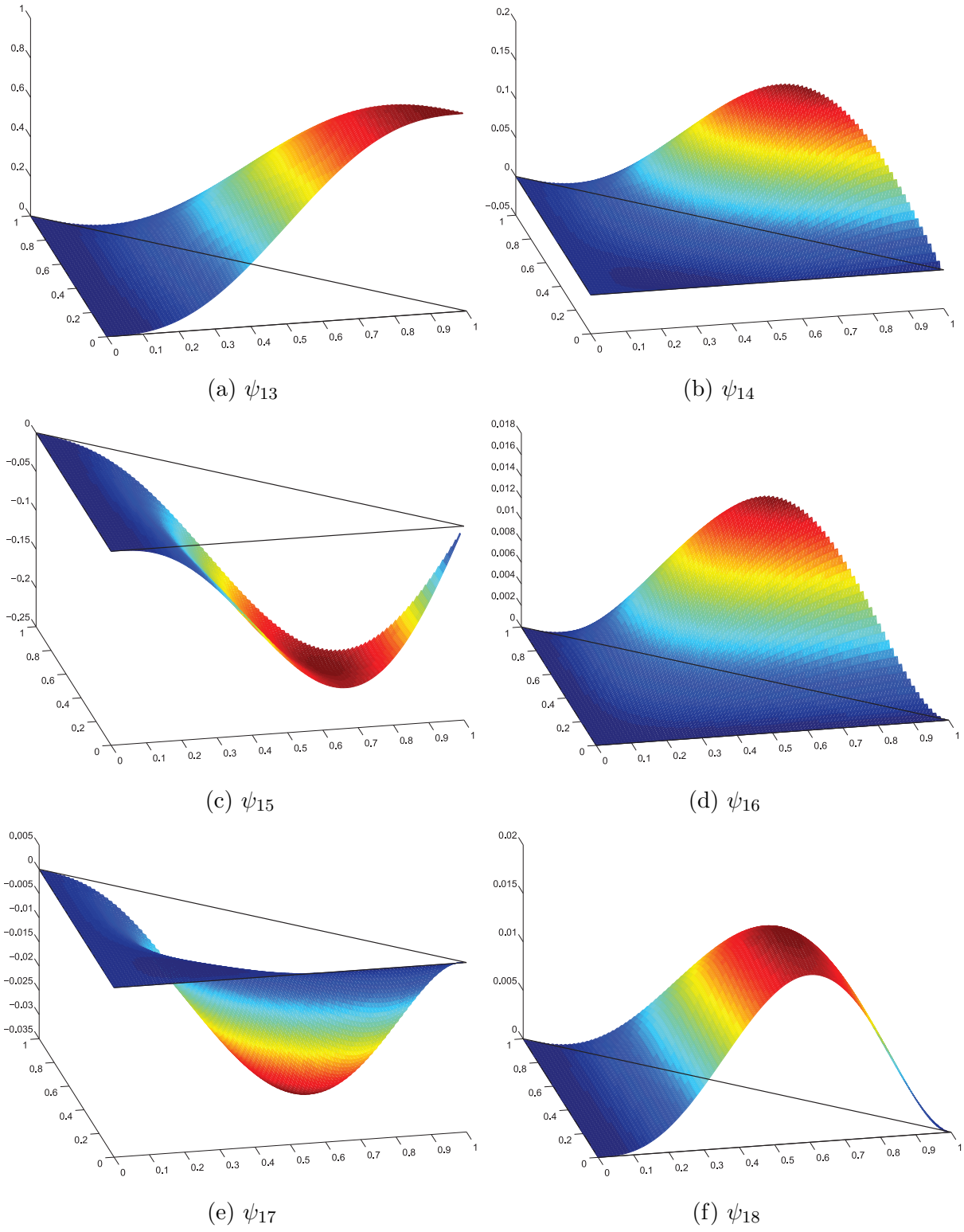


(e) ψ_{11}



(f) ψ_{12}

Figure 2.3: The Argyris shape functions for the vertex (1,0).

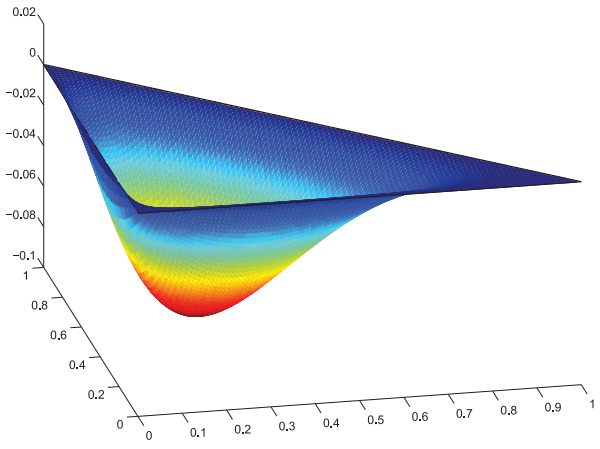


(a) ψ_{13} (b) ψ_{14}

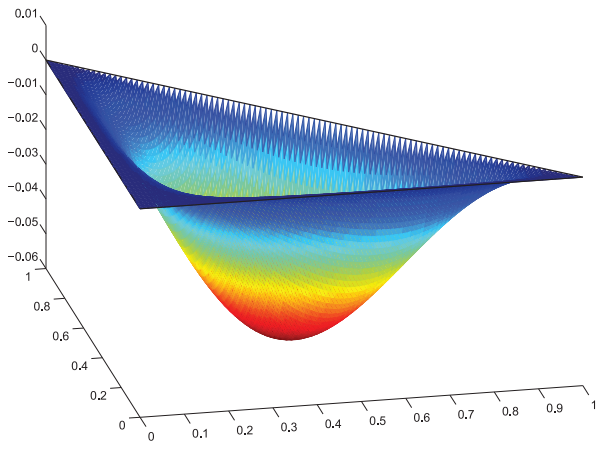
(c) ψ_{15} (d) ψ_{16}

(e) ψ_{17} (f) ψ_{18}

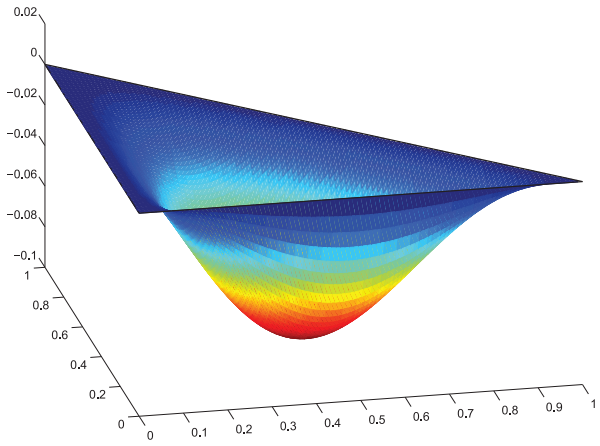
Figure 2.4: The Argyris shape functions for the vertex (0,1).



(a) ψ_{19}



(b) ψ_{20}



(c) ψ_{21}

Figure 2.5: The Argyris shape functions for midpoints (.5,0), (.5,.5) and (0,.5).

With the shape functions in hand, the second step is to create the Argyris basis functions for the points in a mesh of the geometry of interest (in this case the plate Ω). Namely, for each mesh vertex six basis functions (corresponding to the function value, the two first derivatives, and the three 2nd derivatives) and for each edge midpoint in the mesh one basis function (corresponding to the normal derivative at that point). These basis functions are defined piecewise for each triangle in the mesh of which the point is a member. The basis functions are created from affine transformations of the shape functions defined on the reference triangle. The key difficulty is that affine transformations do not preserve the values of the derivatives of the shape functions.

Many avoid using the Argyris functions because of this difficulty but it is necessary to this mixed formulation that the basis functions be H^2-conforming. It is critical to their definition that the derivatives of the Argyris basis functions must take on particular values (1 or 0) at each mesh node. To achieve this, each basis function is composed of a linear combination of transformed shape functions. This has to be done for each basis function, piecewise for each triangle in the mesh. Thus once the mesh is created, a data structure must be built to store the coefficients of these linear combinations; the program **BasisConstant.m** accomplishes this task in our numerical implementation.

Again, following [33] closely, we will outline this procedure below. First, we need to be able to compute the derivatives of the transformed basis functions. To do accomplish this, let T_R denote the reference triangle and T_k denote a particular triangle in the mesh with vertices $(x_0, y_0), (x_1, y_1)$, and (x_2, y_2). Then the affine transformation $\mathbf{x}_k : T_R \to T_k$ given by

$$\mathbf{x}_k(\tilde{x}, \tilde{y}) = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} \tag{2.75}$$

maps $(0, 0)$ to x_0, y_0, $(1, 0)$ to (x_1, y_1) and $(0, 1)$ to (x_2, y_2), and hence maps T_R onto T_k. The vertices in the mesh are ordered such that the Jacobian matrix

$$J_k = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix} = D\mathbf{x}_k / D\tilde{\mathbf{x}},$$

for $\tilde{\mathbf{x}} \in T_R$, is positive. (We used the open source program GMSH which automatically does this.) Thus we can transform a function $\psi \in \mathbb{P}^5(T_k)$ to $\tilde{\psi} \in \mathbb{P}^5(T_R)$ on the reference element via

$$\tilde{\psi} = \psi \circ \mathbf{x}_k. \tag{2.76}$$

Since we can compute the derivatives of the shape functions $\tilde{\psi}$ on T_R directly using calculus, we find the derivatives of the transformed shape functions by using the chain

rule. In fact, the first derivatives are given by

$$\begin{bmatrix} \partial_x \psi(\mathbf{x}) \\ \partial_y \psi(\mathbf{x}) \end{bmatrix} = J_k^{-T} \begin{bmatrix} \partial_x \tilde{\psi}(\tilde{\mathbf{x}}) \\ \partial_y \tilde{\psi}(\tilde{\mathbf{x}}) \end{bmatrix} ; \ \mathbf{x} = \mathbf{x}_k(\tilde{\mathbf{x}}), \tag{2.77}$$

and moreover the second derivatives are given by

$$\begin{bmatrix} \partial_{xx} \psi(\mathbf{x}) \\ \partial_{xy} \psi(\mathbf{x}) \\ \partial_{yy} \psi(\mathbf{x}) \end{bmatrix} = A_k^{-1} \begin{bmatrix} \partial_{xx} \tilde{\psi}(\tilde{\mathbf{x}}) \\ \partial_{xy} \tilde{\psi}(\tilde{\mathbf{x}}) \\ \partial_{yy} \tilde{\psi}(\tilde{\mathbf{x}}) \end{bmatrix} ; \ \mathbf{x} = \mathbf{x}_k(\tilde{\mathbf{x}}), \tag{2.78}$$

where

$$A_k^{-1} = \frac{1}{\det^2(J_k)} \begin{bmatrix} J_{22}^2 & -2J_{21}J_{22} & J_{21}^2 \\ -J_{12}J_{22} & (J_{11}J_{22} + J_{12}J_{21}) & -J_{11}J_{21} \\ J_{12}^2 & -2J_{11}J_{12} & J_{11}^2 \end{bmatrix}; \tag{2.79}$$

where J_{ij} is the value of the ij^{th} entry of J_k; see [33] for details.

Given a particular triangle in the mesh, these matrices are easily constructed and used to compute the values of the derivatives of the transformed shape functions. Thus for each shape function $\psi_1 - \psi_{21}$, we can define for each triangle T_k in the mesh, the transformed shape functions $\psi_{k_1} - \psi_{k_{21}}$. Recall that in this enumeration, $\psi_1 - \psi_6$ correspond to node 1, $\psi_7 - \psi_{12}$ correspond to node 2, and $\psi_{13} - \psi_{18}$ correspond to node 3, while $\psi_{19} - \psi_{21}$ give the three edge midpoint normal derivatives. Recall also from Section 2.4.1 that $\mathcal{L}_1 - \mathcal{L}_{21}$ correspond to the Argyris degrees of freedom, e.g. $\mathcal{L}_6(\psi) = \frac{\partial^2}{\partial y^2}\psi(0,0)$. With this in mind, for each triangle T_k in the mesh we compute the following 21×21 matrix

$$C_k = \begin{bmatrix} \mathcal{L}_1\psi_{k_1} & \dots & \mathcal{L}_1\psi_{k_{21}} \\ \vdots & \ddots & \vdots \\ \mathcal{L}_{21}\psi_{k_1} & \dots & \mathcal{L}_{21}\psi_{k_{21}} \end{bmatrix}. \tag{2.80}$$

Then the coefficients $\alpha^i = [\alpha_1, \dots, \alpha_{21}]^T$ of the i^{th} transformed shape function can be

retrieved by solving

$$C_k \alpha^i = e_i \tag{2.81}$$

where e_i is the i^{th} standard basis element in $\mathbb{R}^{21}$. The matrix C_k is sparse with columns having between 3 and 6 non-zero entries. The entries of C_k are easily computed by taking the matrix product $L_k B$ where L_k is a 21×24 matrix which transforms the values of the shape functions to the element T_k; B is 24×21 matrix that stores the values of the function values (including derivatives) of the shape functions on the reference element. Namely,

$$B = \begin{bmatrix} I_{18} & Z \\ M_1 & M_2 \end{bmatrix} \tag{2.82}$$

where I_{18} is an 18×18 identity matrix, Z is an 18×3 zero matrix, the matrices M_1 and M_2 which give the partial derivatives ∂_x and ∂_y at the edge midpoints $(.5, 0)$, $(.5, .5)$ and $(0, .5)$ are given by

$$M_1 = \begin{bmatrix} \frac{-15}{8} & \frac{-7}{16} & 0 & \frac{-1}{32} & 0 & 0 & \frac{15}{8} & \frac{-7}{16} & 0 & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{15}{16} & \frac{-7}{32} & \frac{7}{32} & \frac{1}{64} & \frac{-1}{32} & \frac{1}{64} & \frac{-15}{16} & \frac{-7}{32} & \frac{7}{32} & \frac{-1}{64} & \frac{1}{32} & \frac{-1}{64} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-15}{16} & \frac{7}{32} & \frac{-7}{32} & \frac{-1}{64} & \frac{1}{32} & \frac{-1}{64} & \frac{15}{16} & \frac{7}{32} & \frac{-7}{32} & \frac{1}{64} & \frac{-1}{32} & \frac{1}{64} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-15}{8} & 0 & \frac{-7}{16} & 0 & 0 & \frac{-1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{15}{8} & 0 & \frac{-7}{16} & 0 & 0 & \frac{1}{32} \end{bmatrix}, \tag{2.83}$$

and

$$M_2 = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.84}$$

The first 18 rows simply evaluate $\mathcal{L}_i(\psi_j)$, but the last 6 rows evaluate $\partial_x\psi_j(.5,0)$, $\partial_y\psi_j(.5,0)$, $\partial_x\psi_j(.5,.5)$, $\partial_y\psi_j(.5,.5)$, $\partial_x\psi_j(0,.5)$, and $\partial_y\psi_j(0,.5)$ from top to bottom. The transformation matrix L_k is given in block form as

$$\begin{bmatrix} L1 & 0 & 0 & 0 \\ 0 & L1 & 0 & 0 \\ 0 & 0 & L1 & 0 \\ 0 & 0 & 0 & L2 \end{bmatrix}, \tag{2.85}$$

where the 6×6 block matrix $L1$ is given by

$$L1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & J_k^{-T} & 0 \\ 0 & 0 & A_k^{-1} \end{bmatrix}, \tag{2.86}$$

and the 3×6 block matrix $L2$ is given by

$$L2 = \begin{bmatrix} \nu_1^T J_k^{-T} & 0 & 0 \\ 0 & \nu_2^T J_k^{-T} & 0 \\ 0 & 0 & \nu_3^T J_k^{-T} \end{bmatrix}, \tag{2.87}$$

where $\nu_1^T = [0,-1]$, $\nu_2^T = [\sqrt{2}/2, \sqrt{2}/2]$, and $\nu_3^T = [-1,0]$. The block nature of these matrices can be exploited in the code to reduce the computation time required to solve

(2.81) for each element in the mesh, although the time to run this section of the code is insignificant compared to the remainder.

One final comment is with respect to implementing boundary conditions with these elements. The $H_0^2(\Omega)$ boundary conditions would force Lagrange and even Hermite basis functions to be all zero on the boundary, but since Argyris functions specify the values of the second derivatives this is not necessarily the case. More specifically the clamped condition does not necessarily impose conditions on ϕ_{xx} and ϕ_{yy}. If the geometry is simple, as in our case, these conditions can be handled in a more ad hoc fashion as was done here. More specifically on the left and right sides of the square domain $[0,1] \times [0,1]$ only the ϕ_{xx} degrees of freedom remain while on the top and bottom sides only the ϕ_{yy} basis functions are in play. Since the domain in our numerical example was a square this was easily imposed. This is more difficult for general domains though one can enforce the conditions variationally. This is done by imposing the boundary conditions using a method akin to Lagrange multipliers, see e.g., [22], [9]. This is possibly one of the reasons for why the Argyris basis functions appear to be scarcely used. However, for PDE models such as this one which require H^2-conforming elements one has little choice.

2.4.2 Fluid Basis Elements

To solve the Stokes problem on the fluid domain we use the Taylor-Hood ($\mathbb{P}^2/\mathbb{P}^1$) elements. The fluid velocity is approximated over quadratic Lagrange elements defined on the reference tetrahedron. The shape functions are listed in Table 2.2 below. The nodes labeled with local node ordering are displayed in Figure 2.6.

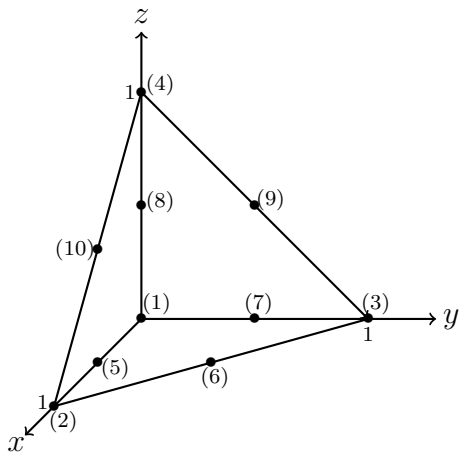


Figure 2.6: The Fluid Velocity Reference Element.

$\varphi_1 = 2\big(1-(x+y+z)\big)\big(1/2-(x+y+z)\big)$
$\varphi_2 = 2x(x-1/2)$
$\varphi_3 = 2y(y-1/2)$
$\varphi_4 = 2z(z-1/2)$
$\varphi_5 = 4x\big(1-(x+y+z)\big)$
$\varphi_6 = 4xy$
$\varphi_7 = 4y\big(1-(x+y+z)\big)$
$\varphi_8 = 4z\big(1-(x+y+z)\big)$
$\varphi_9 = 4yz$
$\varphi_{10} = 4xz$

Table 2.2: Fluid Velocity Shape Functions.

The pressure basis functions are constructed from linear Lagrange elements and are given in Table 2.3 below. The nodes labeled with local node ordering are displayed in Figure 2.7.

$\eta_1 = 1 - (x + y + z)$
$\eta_2 = x$
$\eta_3 = y$
$\eta_4 = z$

Table 2.3: Fluid Pressure Shape Functions.

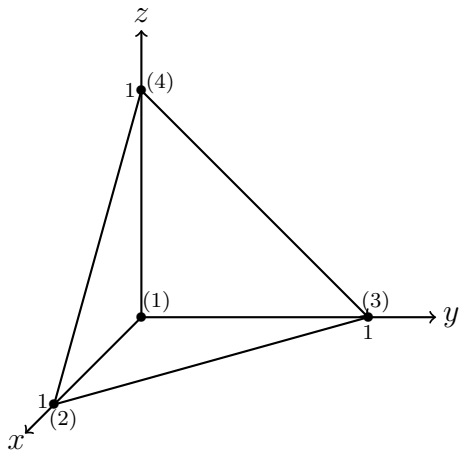


Figure 2.7: The Fluid Pressure Reference Element.

2.5 Matrix Computations

Here we give a brief description of how the mixed variational formulations described earlier are implemented numerically. For example, to solve the mixed problem in (2.9), let as before X_h describe the H^2-conforming space of Argyris elements described in Section 2.4.1. Suppose for a particular mesh this space has dimension N_h with basis $\{\psi_i\}_{i=1}^{N_h}$. Then the approximate solution w_{1h} can be written as $w_{1h} = \sum_{i=1}^{N_h} \alpha_i \psi_i$. Then for

each $\psi_j \in X_h$ and $r \in \mathbb{R}$ we have

$$\sum_{i=1}^{N_h} \alpha_i a_\lambda(\psi_i, \psi_j) + b(\psi_j, \tilde{c}_h) = \mathbb{F}(\psi_j); \tag{2.88}$$

$$\sum_{i=1}^{N_h} \alpha_i b(\psi_i, r) = 0, \tag{2.89}$$

with a_λ, b and $\mathbb{F}$ defined as in (1.43). Repeating this for each $\psi_j \in X_h$ creates a linear system which can be expressed in the form

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \tilde{c}_h \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ 0 \end{bmatrix}. \tag{2.90}$$

Here $A = A_{ij} = a_\lambda(\psi_j, \psi_i)$, $B = B_i = b(\psi_i, 1)$, and $\mathbf{F}_i = \mathbb{F}(\psi_i)$. The matrix formulations of (2.14)-(2.15) and (2.16)-(2.17) are done in an analogous way.

Building the matrices which correspond to the mixed variational formulations in (2.9), (2.14)-(2.15), and (2.16)-(2.17) requires a number of integrals to be evaluated numerically, e.g. the value of $a_\lambda(\phi_i, \phi_j)$. These computations are done via numerical integration of the basis functions. Moreover, the basis functions are defined via linear transformations of the shape functions defined on the reference triangle and tetrahedron. Thus for example if $\Omega = \bigcup_k T_k$ we have

$$(\phi_i, \phi_j)_\Omega = \int_\Omega \phi_i \phi_j d\Omega = \sum_k \int_{T_k} \phi_i \phi_j d\Omega.$$

The final calculation is simplified by the fact that the integrand is zero over many of the elements T_k. For the remaining ones this computation is transferred to an integral over the reference triangle. Thus explicit formulas for the basis functions need never be known. For the domain Ω the resulting integrals are computed using quadrature with evaluation points and weights in Table 2.4(values are rounded in the Table, but are precise to 18 digits in the MATLAB code). This 28 point quadrature rule from [12] is

exact for polynomials of degree 11 or less and thus is exact for a product of two quintic Argyris basis functions.

x_i	y_i	w_i
0.3333	0.3333	0.044
0.948	0.026	0.0044
0.026	0.948	0.0044
0.026	0.026	0.0044
0.8114	0.0943	0.019
0.0943	0.8114	0.019
0.0943	0.0943	0.019
0.0107	0.4946	0.0094
0.4946	0.0107	0.0094
0.4946	0.4946	0.0094
0.5853	0.2073	0.0361
0.2073	0.5853	0.0361
0.2073	0.2073	0.0361
0.1222	0.4389	0.0347
0.4389	0.1222	0.0347
0.4389	0.4389	0.0347
0.6779	0.0448	0.0205
0.6779	0.2772	0.0205
0.0448	0.6779	0.0205
0.0448	0.2772	0.0205
0.2772	0.6779	0.0205
0.2772	0.0448	0.0205
0.8589	0.0	0.0037
0.8589	0.1411	0.0037
0.0	0.8589	0.0037
0.0	0.1411	0.0037
0.1411	0.8589	0.0037
0.1411	0.0	0.0037

Table 2.4: Quadrature Points and Weights for Reference Triangle.

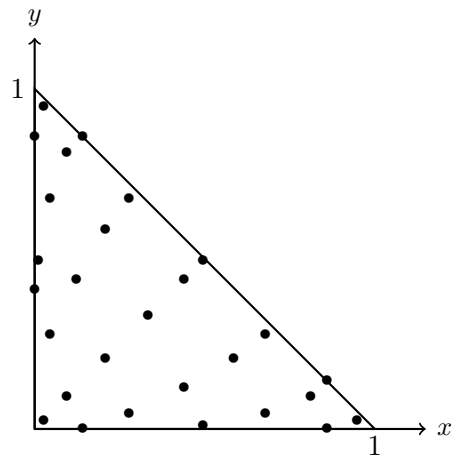


Figure 2.8: Location of Quadrature Points in Reference Triangle.

For the domain $\mathcal{O}$ the quadrature points and weights for the reference tetrahedron are listed in Table 2.5, see [1]. Here an 11 point quadrature is used because the fluid basis functions are quadratic and thus precision for polynomials of degree four or less is sufficient; the points are plotted in Figure 2.9

x_i	y_i	z_i	w_i
0.25	0.25	0.25	-0.0132
0.7857	0.0714	0.0714	0.0076
0.0714	0.7857	0.0714	0.0076
0.0714	0.0714	0.7857	0.0076
0.0714	0.0714	0.0714	0.0076
0.1006	0.1006	0.3994	0.0249
0.1006	0.3994	0.1006	0.0249
0.1006	0.3994	0.3994	0.0249
0.3994	0.1006	0.1006	0.0249
0.3994	0.1006	0.3994	0.0249
0.3994	0.3994	0.1006	0.0249

Table 2.5: Quadrature Points and Weights for Reference Tetrahedron.

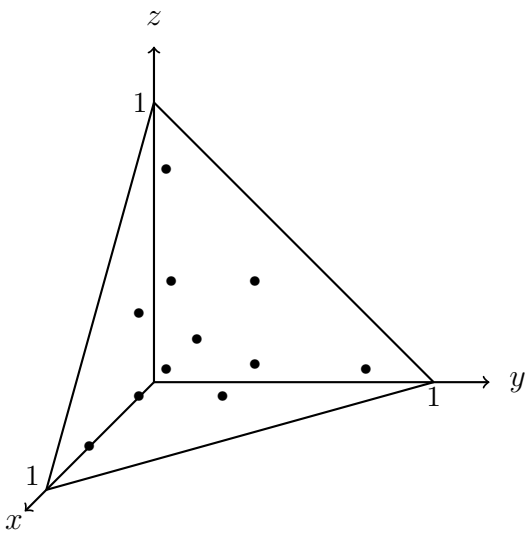


Figure 2.9: Location of Quadrature Points in Reference Tetrahedron.

2.6 Test Problem

Here we build an appropriate test problem for the fluid-structure problem of interest. The fluid domain $\mathcal{O}$ is given by $(x, y, z) \in (0, 1) \times (0, 1) \times (-1, 0)$. The plate Ω is the top boundary of the fluid domain, lying in the xy plane, namely $(x, y) \in (0, 1) \times (0, 1)$ The fluid velocity is given by $u = [u^{(1)}, u^{(2)}, u^{(3)}]$, with fluid pressure p and w_1 and w_2 are the plate displacement and velocity respectively. The functions w_1^*, w_2^*, and u^* are the given data from **H** and we desire u, w_1, w_2, and p satisfying:

$$\lambda w_1 - w_2 = w_1^* \qquad \text{in } \Omega, \tag{2.91}$$

$$\lambda w_2 + P_\rho^{-1}\Delta^2 w_1 - P_\rho^{-1} p\big|_\Omega = w_2^* \qquad \text{in } \Omega, (\text{where } p = G_1(w_1) + G_2(u)), \tag{2.92}$$

$$w_1 = \frac{\partial w_1}{\partial \nu} = 0 \qquad \text{on } \partial\Omega, \tag{2.93}$$

$$\lambda u - \Delta u + \nabla p = u^* \qquad \text{in } \mathcal{O}, \tag{2.94}$$

$$\text{div}(u) = 0 \qquad \text{in } \mathcal{O}, \tag{2.95}$$

$$u\big|_S = [0, 0, 0] \qquad \text{on } S, \tag{2.96}$$

$$u\big|_\Omega = [0, 0, w_2] \qquad \text{in } \Omega. \tag{2.97}$$

This test problem assumes that $\rho = 0$ and thus $P_\rho = I$. Then for any $\lambda > 0$ the functions

$$
\begin{aligned}
w_1 &= -x^4(x-1)^4(2x-1)y^4(y-1)^4 \ \in H_0^2(\Omega) \cap \widehat{L}^2(\Omega),\\
w_2 &= -\Delta w_1 = 12x^2(x-1)^2(2x-1)(6x^2-6x+1)y^4(y-1)^4\\
&\quad + x^4(x-1)^4(2x-1)4y^2(y-1)^2(14y^2-14y+3) \ \in H_0^2(\Omega) \cap \widehat{L}^2(\Omega),\\
u^{(1)} &= \big[2x^3(x-1)^3(9x^2-9x+2)y^4(y-1)^4 + (4/5)x^5(x-1)^5y^2(y-1)^2(14y^2-14y+3)\big]\times\\
&\quad \big[-30z^4-60z^3-30z^2\big],\\
u^{(2)} &= 0,\\
u^{(3)} &= -\big[12x^2(x-1)^2(2x-1)(6x^2-6x+1)y^4(y-1)^4 + 4x^4(x-1)^4(2x-1)\times\\
&\quad y^2(y-1)^2(14y^2-14y+3)\big] * \big[-6z^5-15z^4-10z^3-1\big],\\
p &= 0.
\end{aligned}
$$

solve (1.35) - (1.41) for data defined by

$$
\begin{aligned}
w_1^* &\equiv \lambda w_1 - w_2;\\
w_2^* &\equiv \lambda w_2 + \Delta^2 w_1;\\
u^* &\equiv \lambda u - \Delta u.
\end{aligned}
$$

Notice that $u = [u^{(1)}, u^{(2)}, u^{(3)}]$ is divergence free, $u = [0,0,0]$ on S, $\Delta u \cdot \nu = [0,0,0]$ on S, and $u = [0,0,w_2]$ on Ω. Moreover $p := G_{\rho,1}(w_1) + G_{\rho,2}(u) = 0$ because w_1 and u are chosen such that $\Delta^2 w_1 = -\Delta u \cdot \nu$ on Ω, and $\Delta u \cdot \nu = 0$ on S which causes the two terms to cancel. Finally we have $[w_1^*, w_2^*, u^*] \in \mathbf{H}_\rho$ (in the $\rho = 0$ case $w_2^* \in \widehat{L}^2(\Omega)$ only). The error in the numerical solution of the structure variable w_1 is summarized in the table below.

No. of elements	Characteristic Length	$\lvert w_1 - w_{1h}\rvert_{H^2}$	$\lvert w_1 - w_{1h}\rvert_{H^1}$	$\lVert w_1 - w_{1h}\rVert_{L^2}$
4	1	7.132×10^{-5}	4.993×10^{-6}	3.935×10^{-7}
16	.5	9.823×10^{-6}	3.450×10^{-7}	1.509×10^{-8}
64	.25	1.249×10^{-6}	2.761×10^{-8}	1.598×10^{-9}
256	.125	8.343×10^{-8}	1.253×10^{-9}	1.066×10^{-10}
1024	.0625	5.124×10^{-9}	6.771×10^{-11}	6.285×10^{-12}

Table 2.6: Errors of structure FEM approximations.

Since the mesh is refined by a factor of 2 at each step, we compute $\log\left(\frac{\text{Error}_i}{\text{Error}_{i+1}}\right)/\log(2)$. In the limit this ratio should approach the exponent of convergence, i.e. $\mathcal{O}(h^k)$. Now for smooth data (as we have for this test problem) the error estimates in (18) can be improved from those given in Theorem 18. The best possible convergence rate one could attain for Argyris elements (independent of the PDE) is $k = 4$ for the H^2-norm of w_1 which the numerical scheme does appear to attain (see Table 2.7). However, the H^1- and L^2-errors do not appear to improve to $k = 5$ and $k = 6$ respectively; this is possibly due to the (unavoidable) approximation of $\tilde{f}$ and $\tilde{\mu}$ described above or the fact that the fluid basis functions are of lower degree than those on the plate. (Note that the achieved rate is better than what we could prove in Theorem 18.)

	H^2	H^1	L^2
Mesh 1 / Mesh 2	2.86	3.86	4.71
Mesh 2 / Mesh 3	2.98	3.64	3.24
Mesh 3 / Mesh 4	3.90	4.46	3.91
Mesh 4 / Mesh 5	4.03	4.21	4.08

Table 2.7: Computed index k in $\mathcal{O}(h^k)$ for structure FEM approximations.

Similarly, for the fluid approximation we have:

No. of elements	Characteristic Length	$\|u-u_h\|_{\mathbf{L}^2}$	$\|u-u_h\|_{\mathbf{H}^1}$	$\|p-p_h\|_{L^2}$
24	1	5.26×10^{-4}	9.53×10^{-3}	1.40×10^{-4}
192	.5	1.42×10^{-5}	3.64×10^{-4}	5.85×10^{-5}
1536	.25	3.56×10^{-6}	1.75×10^{-4}	2.25×10^{-5}
12288	.125	4.98×10^{-7}	5.20×10^{-5}	3.56×10^{-6}
98304	.0625	6.33×10^{-8}	1.37×10^{-5}	6.67×10^{-7}

Table 2.8: Errors of fluid FEM approximations.

The log error ratios approach what is expected for a $\mathbb{P}^2/\mathbb{P}^1$ implementation, namely $k = 3, 2$, and 2 respectively as shown in Table 2.9.

	$\mathbf{L}^2$(fluid)	$\mathbf{H}^1$(fluid)	L^2(pressure)
Mesh 1 / Mesh 2	1.89	1.38	1.26
Mesh 2 / Mesh 3	1.99	1.05	1.38
Mesh 3 / Mesh 4	2.84	1.75	2.66
Mesh 4 / Mesh 5	2.97	1.93	2.41

Table 2.9: Computed index k in $\mathcal{O}(h^k)$ for fluid FEM approximations.

As the mesh is refined, the FEM approximation improves in quality. The figures below show that in a side by side comparison already at the Mesh 3 level that the approximations of w_1 are nearly indistinguishable from the true solution.

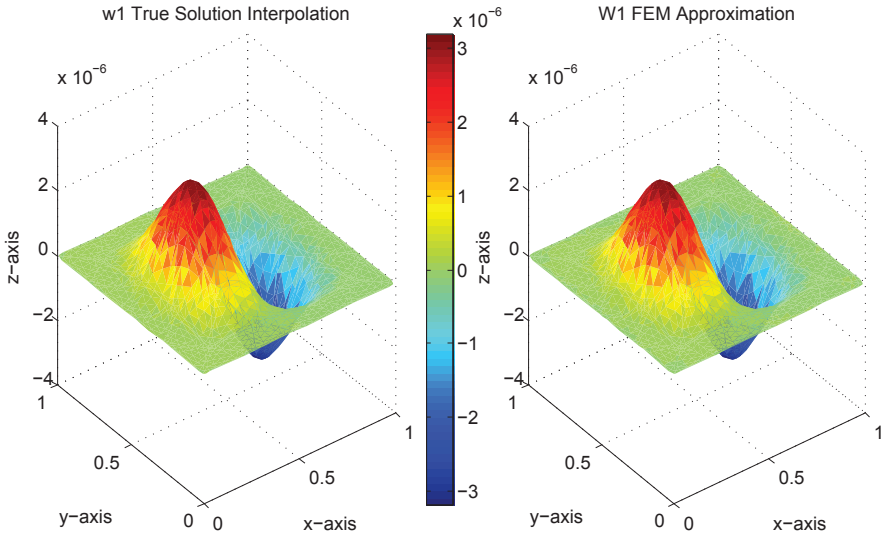


Figure 2.10: FEM approximation for w_1.

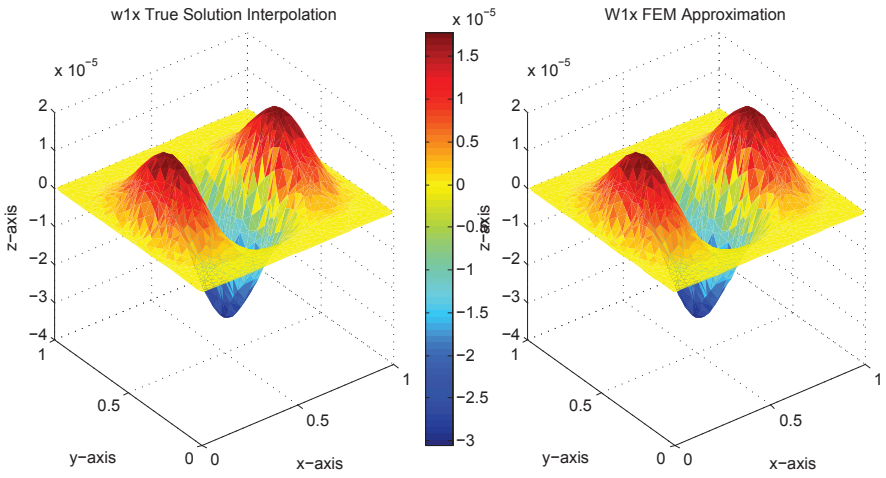


Figure 2.11: FEM approximation for w_{1x}.

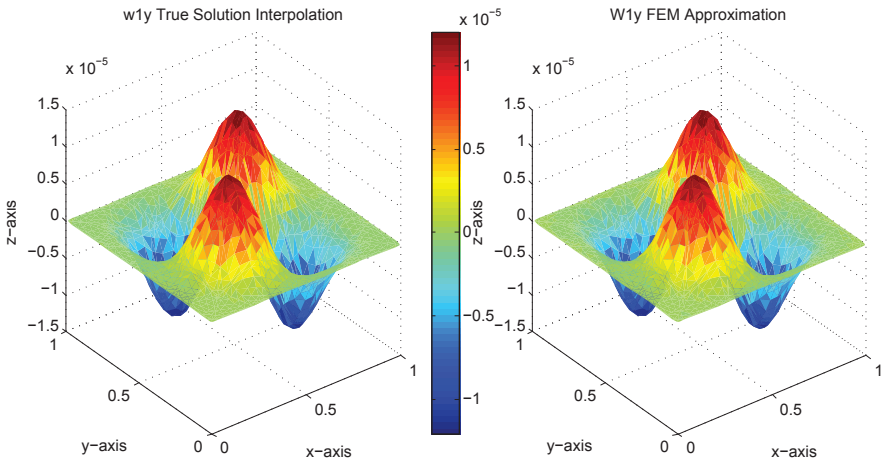


Figure 2.12: FEM approximation for w_{1y}.

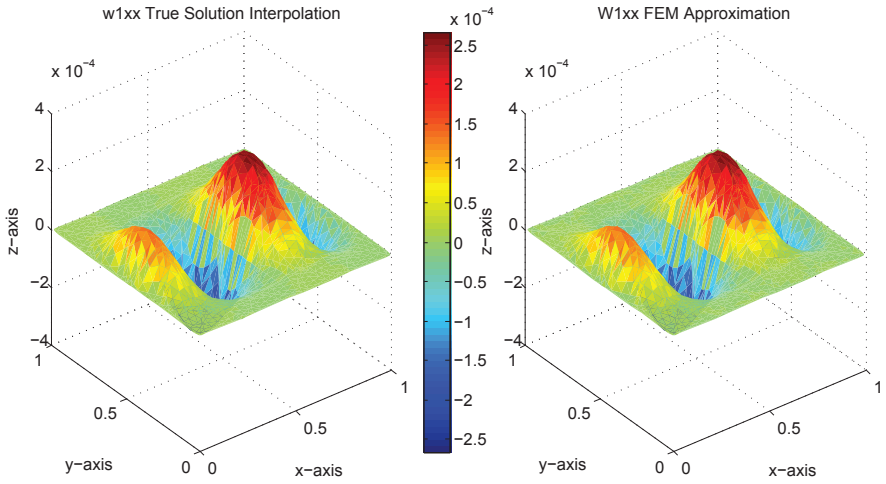


Figure 2.13: FEM approximation for w_{1xx}.

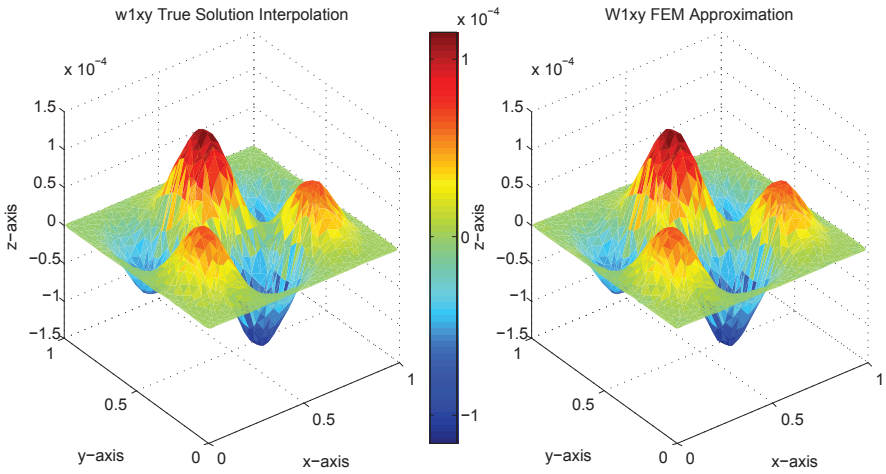


Figure 2.14: FEM approximation for w_{1xy}.

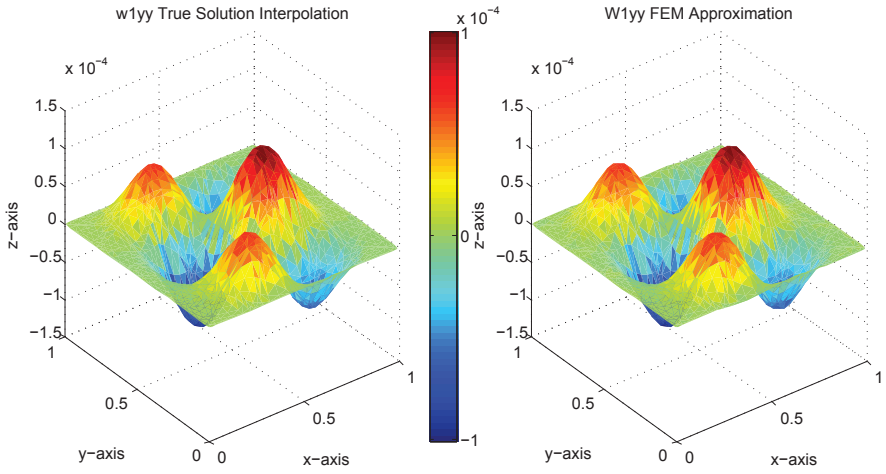


Figure 2.15: FEM approximation for w_{1yy}.

Below are plots of the fluid $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$ as well as the pressure solution p. The figures below display these plots on three dimensional domains using slicing. Note that

the order of the error on $u^{(2)}$ is 10^{-6} and on p is 10^{-4}.

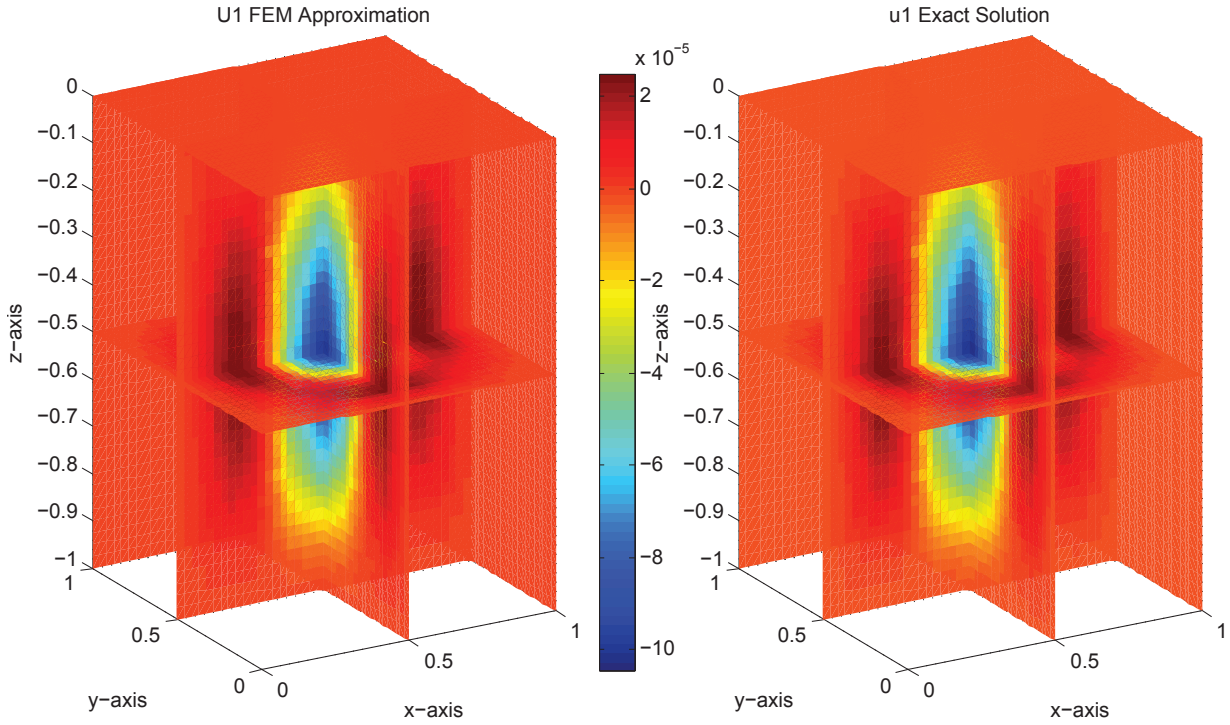


Figure 2.16: FEM approximation for $u^{(1)}$.

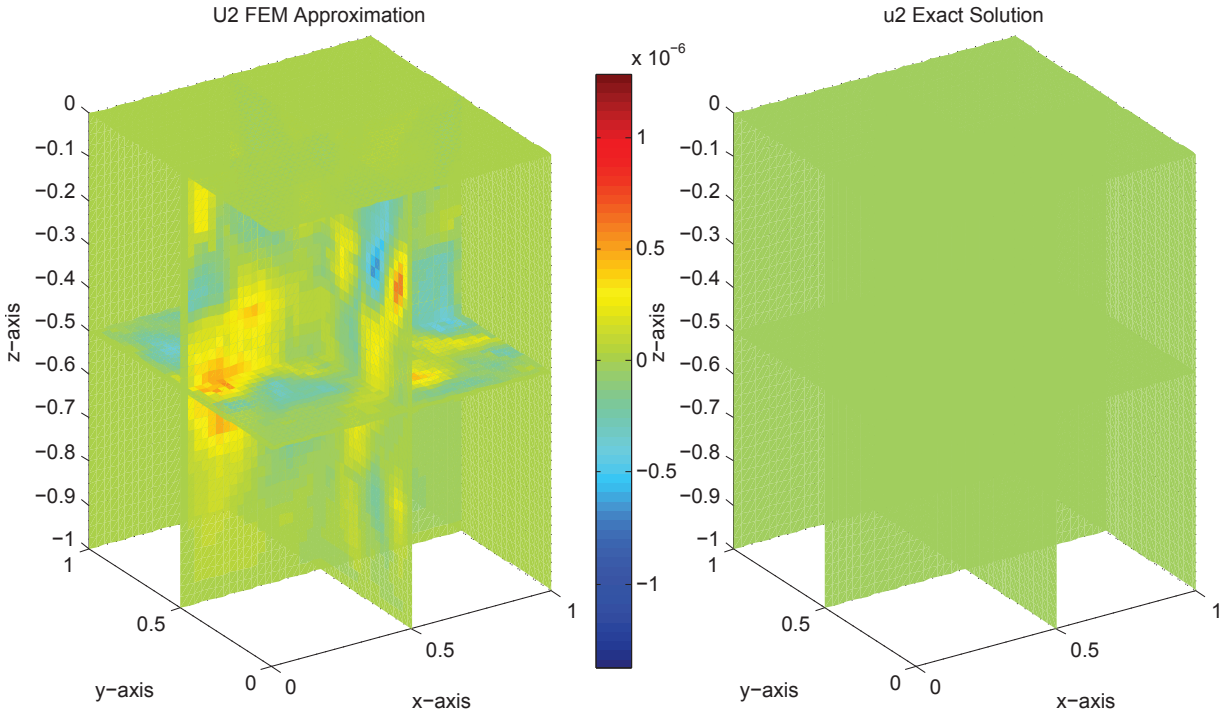


Figure 2.17: FEM approximation for $u^{(2)}$.

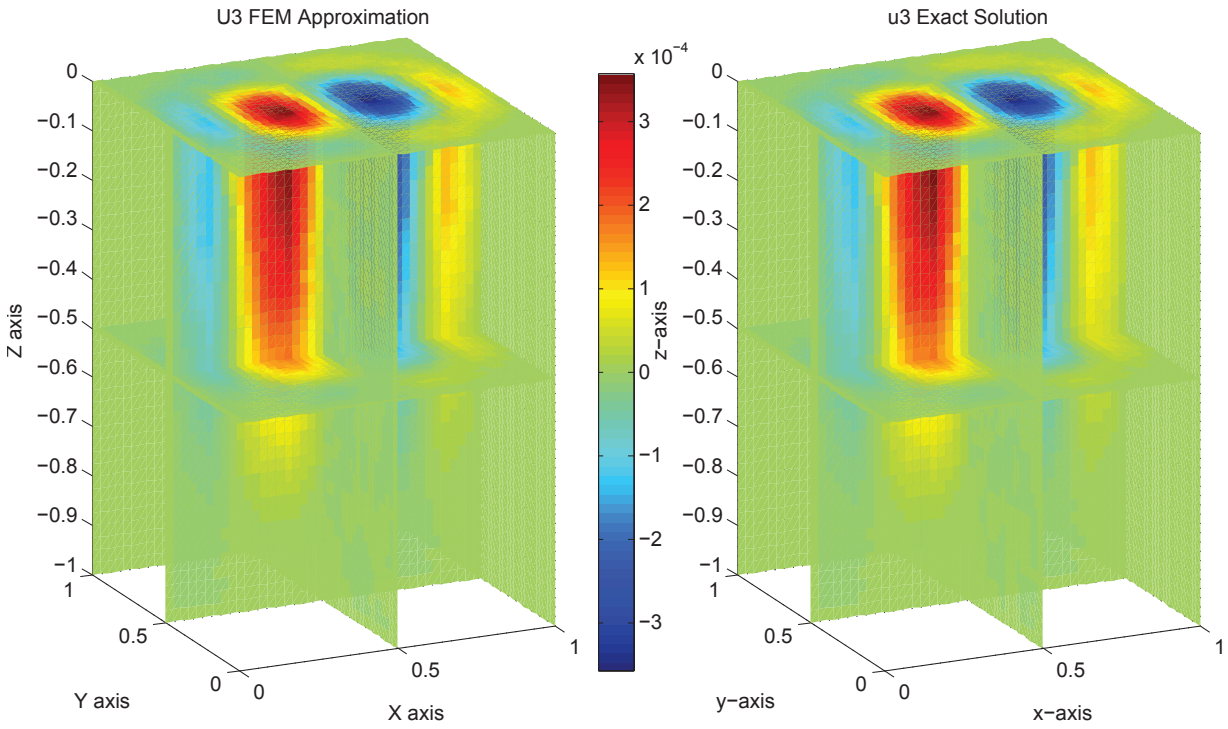


Figure 2.18: FEM approximation for $u^{(3)}$.

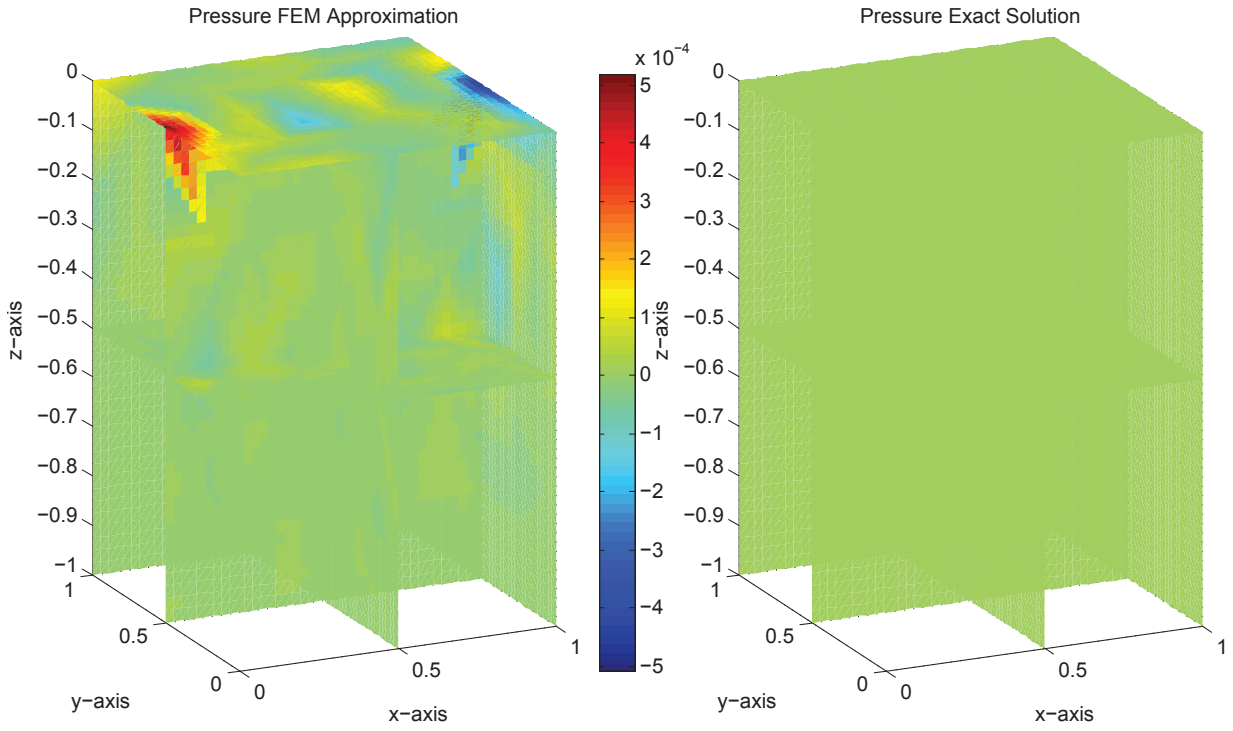


Figure 2.19: FEM approximation for p.

The mesh is made using the open source program GMSH. The programs takes a geometry of points, lines, planes, and volumes as inputs and creates the two and three dimensional meshing. It can also refine the mesh by splitting each triangle into four triangles and each tetrahedron into 8 tetrahedra. This means that if the original meshing is a regular mesh, all subsequent iterations will be regular as well, that is, none of the angles in the elements are too extreme.

It should also be noted that for a polyhedral domain, as is used here, the Taylor-Hood elements are stable if every tetrahedron has at least one internal vertex, see Theorem 8.2 in [8]. In view of this, the geometry and mesh are constructed in such a way to guarantee that this condition is satisfied so that the approximate solution to the fluid Stokes

problem doesn't exhibit "locking", i.e. the pressure space isn't overly rich in enforcing the divergence free condition on the fluid velocity. The figures below illustrate how the plate and fluid domains $\mathcal{O}$ is partitioned into triangles and tetrahedra respectively.

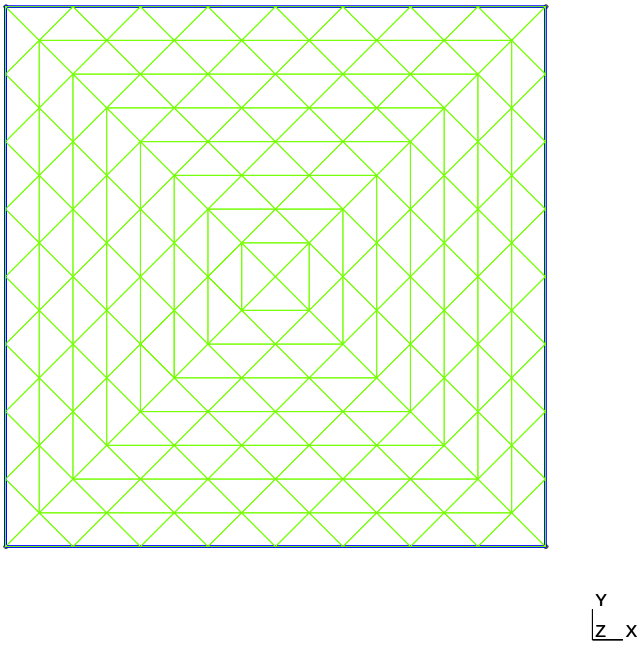


Figure 2.20: 2D plate mesh built with GMSH.

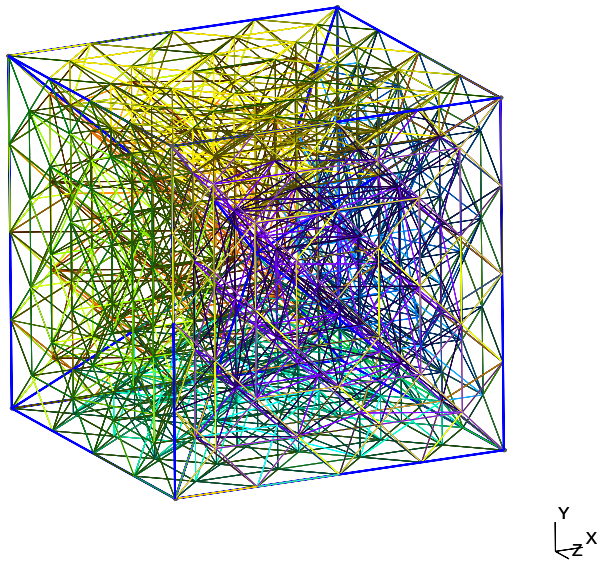


Figure 2.21: 3D fluid mesh built with GMSH.

Appendix A

Matlab Code

Here we include the matlab code that was written to solve implement the numerical scheme outlined in Chapter 2 to solve the fluid structure problem. The main program is **FluidStructure.m**. This program takes as inputs *Mesh* and *pMesh*, which are the fluid and plate geometry mesh data structures that must first be created using **gmsh2mesh3D.m** and **Mesh3DToPlate2D.m**. Note that these programs call **load_gmsh2.m** available in the GMSH source code file. These programs take the a geometry which is built in the open source mesh generating software GMSH and creates the coupled mesh data structures for the fluid and plate respectively. The mesh can be refined by splitting elements in GMSH and then solved iteratively on the successively finer meshes to see the convergence of the numerical scheme.

Once the mesh structures are built **FluidStructure.m** creates all the FEM basis functions for the fluid and plate geometries (the constants for the linear combinations of the Argyris shape functions are computed by **BasisConstant.m**) and then solves (2.91)-(2.97) numerically. The system can be solved for given data $[w_1^*, w_2^*, u^*] \in H_\rho$ for $\rho = 0$. The code as shown in the appendix is set up to solve the test problem in Section 2.6 for $\lambda = 1$.

To solve for w_1, the program builds the block matrix from (2.90) in pieces by calling the

intermediary programs **StiffnessA2D.m**, **StiffnessAL2D.m**, **StiffnessAFT2D.m**, **StiffnessQ3D.m**, **StiffnessQ3DB.m**, **fTilde.m**, and **BLFormA2D.m** and assembling it appropriately. The load vector from (2.90) is then constructed by **Load1A2D.m**, **Load2A2D.m**, **Load3A2D.m**, and **Load4A2D.m**.

Next, w_2 is given in terms of known quantities via (1.35). Finally (1.44) is solved to yield u and p. The program computes the errors from the known solutions and then plots them side by side for visual comparison. The programs which compute the various errors are: **L2ErrorQ3D.m** and **H1ErrorQ3D.m** for the fluid; **L2ErrorQ3DP.m** for the pressure; **DivErrorQ3D.m** for the divergence of the fluid; and **L2ErrorA2D.m**, **H1xErrorA2D.m**, **H1yErrorA2D.m**, **H2xxErrorA2D.m**, **H2xyErrorA2D.m** and **H2yyErrorA2D.m** for the plate displacement. For space reasons, we only include **L2ErrorQ3D.m** and **L2ErrorA2D.m** in the appendix because they are all quite similar.

Each of these programs contains internal documentation and comments to clarify the computations being made and their role in the overall scheme.

FluidStructure.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%  Full Fluid Structure Combined Problem
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% First load mesh into workspace
Mesh=Mesh3; % Mesh is built first with gmsh2mesh3D.m and Mesh3DToPlate2D.m
pMesh=pMesh3; % Save mesh refinements as Mesh1, Mesh2, etc.

Nf=length(Mesh.FNodePtrs);    % # of free fluid nodes
Np=length(Mesh.PNodePtrs);    % # of fluid pressure nodes
Nih=length(Mesh.IHNodePtrs); % # of fluid inhomogeneous nodes
Nc=length(Mesh.CNodePtrs);    % # of fluid constrained nodes
lambda=1;
```

```
%%% Test Problem
% Fluid
u1=@(x,y,z) (2*x^3*(x-1)^3*(9*x^2-9*x+2)*y^4*(y-1)^4 + (4/5)*x^5*(x-1)^5*y^2*(y-1)^2*(14*y...
    ^2-14*y+3))*30*(-z^4-2*z^3-z^2);
u2=@(x,y,z) 0;
u3=@(x,y,z) -1*(12*x^2*(x-1)^2*(2*x-1)*(6*x^2-6*x+1)*y^4*(y-1)^4 + 4*x^4*(x-1)^4*(2*x-1)*y^2*(...
    y-1)^2*(14*y^2-14*y+3))*(-6*z^5-15*z^4-10*z^3-1);
u1x=@(x,y,z) (12*x^2*(x-1)^2*(12*x^3-18*x^2+8*x-1)*y^4*(y-1)^4 + 4*x^4*(x-1)^4*(2*x-1)*y^2*(y...
    -1)^2*(14*y^2-14*y+3))*30*(-z^4-2*z^3-z^2);
u1y=@(x,y,z) (2*x^3*(x-1)^3*(9*x^2-9*x+2)*4*y^3*(y-1)^3*(2*y-1) + (24/5)*x^5*(x-1)^5*y*(y-1)...
    *(2*y-1)*(7*y^2-7*y+1))*30*(-z^4-2*z^3-z^2);
u1z=@(x,y,z) (2*x^3*(x-1)^3*(9*x^2-9*x+2)*y^4*(y-1)^4 + (4/5)*x^5*(x-1)^5*y^2*(y-1)^2*(14*y...
    ^2-14*y+3))*60*(-2*z^3-3*z^2-z);
u2x=@(x,y,z) 0;
u2y=@(x,y,z) 0;
u2z=@(x,y,z) 0;
u3x=@(x,y,z) -1*(24*x*(42*x^5-126*x^4+140*x^3-70*x^2+15*x-1)*y^4*(y-1)^4 + 8*x^3*(x-1)^3*(9*x...
    ^2-9*x+2)*y^2*(y-1)^2*(14*y^2-14*y+3))*(-6*z^5-15*z^4-10*z^3-1);
u3y=@(x,y,z) -1*(48*x^2*(x-1)^2*(2*x-1)*(6*x^2-6*x+1)*y^3*(y-1)^3*(2*y-1) + 24*x^4*(x-1)^4*(2*...
    x-1)*y*(y-1)*(2*y-1)*(7*y^2-7*y+1))*(-6*z^5-15*z^4-10*z^3-1);
u3z=@(x,y,z) -1*(12*x^2*(x-1)^2*(2*x-1)*(6*x^2-6*x+1)*y^4*(y-1)^4 + 4*x^4*(x-1)^4*(2*x-1)*y...
    ^2*(y-1)^2*(14*y^2-14*y+3))*30*(-z^4-2*z^3-z^2);
u1xx=@(x,y,z) (24*x*(x-1)*(42*x^4-84*x^3+56*x^2-14*x+1)*y^4*(y-1)^4 + 8*x^3*(x-1)^3*(9*x^2-9*x...
    +2)*y^2*(y-1)^2*(14*y^2-14*y+3))*30*(-z^4-2*z^3-z^2);
u1yy=@(x,y,z) (2*x^3*(x-1)^3*(9*x^2-9*x+2)*4*y^2*(y-1)^2*(14*y^2-14*y+3) + (24/5)*x^5*(x-1)...
    ^5*(70*y^4-140*y^3+90*y^2-20*y+1))*30*(-z^4-2*z^3-z^2);
u1zz=@(x,y,z) (2*x^3*(x-1)^3*(9*x^2-9*x+2)*y^4*(y-1)^4 + (4/5)*x^5*(x-1)^5*y^2*(y-1)^2*(14*y...
    ^2-14*y+3))*60*(-6*z^2-6*z-1);
u3xx=@(x,y,z) -1*(24*(252*x^5-630*x^4+560*x^3-210*x^2+30*x-1)*y^4*(y-1)^4 + 48*x^2*(x-1)^2*(2*...
    x-1)*(6*x^2-6*x+1)*y^2*(y-1)^2*(14*y^2-14*y+3))*(-6*z^5-15*z^4-10*z^3-1);
u3yy=@(x,y,z) -1*(48*x^2*(x-1)^2*(2*x-1)*(6*x^2-6*x+1)*y^2*(y-1)^2*(14*y^2-14*y+3) + 24*x^4*(x...
    -1)^4*(2*x-1)*(70*y^4-140*y^3+90*y^2-20*y+1))*(-6*z^5-15*z^4-10*z^3-1);
u3zz=@(x,y,z) -1*(12*x^2*(x-1)^2*(2*x-1)*(6*x^2-6*x+1)*y^4*(y-1)^4 + 4*x^4*(x-1)^4*(2*x-1)*y...
    ^2*(y-1)^2*(14*y^2-14*y+3))*60*(-2*z^3-3*z^2-z);
ustar1=@(x,y,z) lambda*u1(x,y,z) - u1xx(x,y,z) - u1yy(x,y,z) - u1zz(x,y,z);
ustar2=@(x,y,z) 0;
ustar3=@(x,y,z) lambda*u3(x,y,z) - u3xx(x,y,z) - u3yy(x,y,z) - u3zz(x,y,z);
p=@(x,y,z) 0;
%%% Group Together
u={u1,u2,u3};
gradu1={u1x,u1y,u1z};
```

```
gradu2={u2x,u2y,u2z};
gradu3={u3x,u3y,u3z};
ustar={ustar1,ustar2,ustar3};

%%% Plate (Note that w1, w2 are in L^2_0)
w1=@(x,y) -x^4*(x-1)^4*(2*x-1)*y^4*(y-1)^4;
w1x=@(x,y) -2*x^3*(x-1)^3*(9*x^2-9*x+2)*y^4*(y-1)^4;
w1y=@(x,y) -x^4*(x-1)^4*(2*x-1)*4*y^3*(y-1)^3*(2*y-1);
w1xx=@(x,y) -12*x^2*(x-1)^2*(12*x^3-18*x^2+8*x-1)*y^4*(y-1)^4;
w1xy=@(x,y) -8*x^3*(x-1)^3*(9*x^2-9*x+2)*y^3*(y-1)^3*(2*y-1);
w1yy=@(x,y) -x^4*(x-1)^4*(2*x-1)*4*y^2*(y-1)^2*(14*y^2-14*y+3);
BHw1=@(x,y)   -24*(252*x^5-630*x^4+560*x^3-210*x^2+30*x-1)*y^4*(y-1)^4 - 96*x^2*(x-1)^2*(12*x...
    ^3-18*x^2+8*x-1)*y^2*(y-1)^2*(14*y^2-14*y+3) - 24*x^4*(x-1)^4*(2*x-1)*(70*y^4-140*y^3+90*y...
    ^2-20*y+1);
w2=@(x,y) 12*x^2*(x-1)^2*(2*x-1)*(6*x^2-6*x+1)*y^4*(y-1)^4 + 4*x^4*(x-1)^4*(2*x-1)*y^2*(y-1)...
    ^2*(14*y^2-14*y+3);
w1star=@(x,y) lambda*w1(x,y) - w2(x,y);
w2star=@(x,y) lambda*w2(x,y) + BHw1(x,y);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialize Fluid Variables %%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scalar Basis Functions:
phi1=@(x,y,z) 2*(1-(x+y+z))*(1/2-(x+y+z));
phi2=@(x,y,z) 2*x*(x-1/2);
phi3=@(x,y,z) 2*y*(y-1/2);
phi4=@(x,y,z) 2*z*(z-1/2);
phi5=@(x,y,z) 4*x*(1-(x+y+z));
phi6=@(x,y,z) 4*x*y;
phi7=@(x,y,z) 4*y*(1-(x+y+z));
phi8=@(x,y,z) 4*z*(1-(x+y+z));
phi9=@(x,y,z) 4*y*z;
phi10=@(x,y,z) 4*x*z;
% Group together
phi={phi1,phi2,phi3,phi4,phi5,phi6,phi7,phi8,phi9,phi10};

% Gradients of Scalar Basis Functions:
phi1x=@(x,y,z) -3+4*x+4*y+4*z;
phi1y=@(x,y,z) -3+4*x+4*y+4*z;
phi1z=@(x,y,z) -3+4*x+4*y+4*z;
phi2x=@(x,y,z) 4*x-1;
phi2y=@(x,y,z) 0;
phi2z=@(x,y,z) 0;
```

```
phi3x=@(x,y,z) 0;
phi3y=@(x,y,z) 4*y-1;
phi3z=@(x,y,z) 0;
phi4x=@(x,y,z) 0;
phi4y=@(x,y,z) 0;
phi4z=@(x,y,z) 4*z-1;
phi5x=@(x,y,z) 4-8*x-4*y-4*z;
phi5y=@(x,y,z) -4*x;
phi5z=@(x,y,z) -4*x;
phi6x=@(x,y,z) 4*y;
phi6y=@(x,y,z) 4*x;
phi6z=@(x,y,z) 0;
phi7x=@(x,y,z) -4*y;
phi7y=@(x,y,z) 4-4*x-8*y-4*z;
phi7z=@(x,y,z) -4*y;
phi8x=@(x,y,z) -4*z;
phi8y=@(x,y,z) -4*z;
phi8z=@(x,y,z) 4-4*x-4*y-8*z;
phi9x=@(x,y,z) 0;
phi9y=@(x,y,z) 4*z;
phi9z=@(x,y,z) 4*y;
phi10x=@(x,y,z) 4*z;
phi10y=@(x,y,z) 0;
phi10z=@(x,y,z) 4*x;
% Group together
phix={phi1x,phi2x,phi3x,phi4x,phi5x,phi6x,phi7x,phi8x,phi9x,phi10x};
phiy={phi1y,phi2y,phi3y,phi4y,phi5y,phi6y,phi7y,phi8y,phi9y,phi10y};
phiz={phi1z,phi2z,phi3z,phi4z,phi5z,phi6z,phi7z,phi8z,phi9z,phi10z};

% Pressure Basis Functions
eta1=@(x,y,z) 1-(x+y+z);
eta2=@(x,y,z) x;
eta3=@(x,y,z) y;
eta4=@(x,y,z) z;
% Group Together
eta={eta1,eta2,eta3,eta4};

% Gradients of Pressure Basis Functions
eta1x=@(x,y,z) -1;
eta1y=@(x,y,z) -1;
eta1z=@(x,y,z) -1;
```

```
eta2x=@(x,y,z) 1;
eta2y=@(x,y,z) 0;
eta2z=@(x,y,z) 0;
eta3x=@(x,y,z) 0;
eta3y=@(x,y,z) 1;
eta3z=@(x,y,z) 0;
eta4x=@(x,y,z) 0;
eta4y=@(x,y,z) 0;
eta4z=@(x,y,z) 1;
% Group Together
etax={eta1x,eta2x,eta3x,eta4x};
etay={eta1y,eta2y,eta3y,eta4y};
etaz={eta1z,eta2z,eta3z,eta4z};

%%% 3D Fluid Quadrature Points & Weights (N = 11, D = 4) from
%%% http://www.mems.rice.edu/~akin/Elsevier/Chap_10.pdf
fqwt = [-74/5625; 343/45000; 343/45000; 343/45000; 343/45000;
    56/2250; 56/2250; 56/2250; 56/2250; 56/2250; 56/2250];
fqpt = zeros(3,11);
fqpt(:,1) = [1/4; 1/4; 1/4];
fqpt(:,2) = [11/14; 1/14; 1/14];
fqpt(:,3) = [1/14; 11/14; 1/14];
fqpt(:,4) = [1/14; 1/14; 11/14];
fqpt(:,5) = [1/14; 1/14; 1/14];
fqpt(:,6) = [0.100596423833201; 0.100596423833201; 0.399403576166799];
fqpt(:,7) = [0.100596423833201; 0.399403576166799; 0.100596423833201];
fqpt(:,8) = [0.100596423833201; 0.399403576166799; 0.399403576166799];
fqpt(:,9) = [0.399403576166799; 0.100596423833201; 0.100596423833201];
fqpt(:,10) = [0.399403576166799; 0.100596423833201; 0.399403576166799];
fqpt(:,11) = [0.399403576166799; 0.399403576166799; 0.100596423833201];

% Basis functions phiVal[i,j] = phi_j(r_i, s_i, t_i). The columns
% correspond to the different phi's and the rows to the values
% of phi_j at the quadrature points(qpt).
% The derivatives phixVal,phiyVal,phizVal are defined similarly.
fql=length(fqwt);
phiVal=zeros(fql,10);
phixVal=zeros(fql,10);
phiyVal=zeros(fql,10);
phizVal=zeros(fql,10);
for i=1:fql
```

```
    for j=1:10
        phiVal(i,j)=phi{j}(fqpt(1,i),fqpt(2,i),fqpt(3,i));
        phixVal(i,j)=phix{j}(fqpt(1,i),fqpt(2,i),fqpt(3,i));
        phiyVal(i,j)=phiy{j}(fqpt(1,i),fqpt(2,i),fqpt(3,i));
        phizVal(i,j)=phiz{j}(fqpt(1,i),fqpt(2,i),fqpt(3,i));
    end
end

% Pressure basis functions etaVal[i,j] = eta_j(r_i, s_i, t_i).  Columns
% correspond to the different etas and rows to the values of eta_j
% at the different quadrature points(qpt).
etaVal=zeros(fql,4);
for i=1:fql
    for j=1:4
        etaVal(i,j)=eta{j}(fqpt(1,i),fqpt(2,i),fqpt(3,i));
    end
end

%%% Generate C, the L2_0 corrector for each eta
C=L20Corrector(Mesh);

%%%%%%%%%%%%%%%%%%%%%%%% Initialize Plate Variables %%%%%%%%%%%%%%%%%%%%%%%%
%%% Initialize Plate Shape Functions (2D d=5 Argyris Shape Functions)
%%% for (0,0):
mu1=@(x,y) 1-10*x^3-10*y^3+15*x^4-30*x^2*y^2+15*y^4-6*x^5+30*x^3*y^2+30*x^2*y^3-6*y^5;% mu1...
    (0,0)=1
mu2=@(x,y) x-6*x^3-11*x*y^2+8*x^4+10*x^2*y^2+18*x*y^3-3*x^5+x^3*y^2-10*x^2*y^3-8*x*y^4; % ...
    mu2_x(0,0)=1
mu3=@(x,y) y-11*x^2*y-6*y^3+18*x^3*y+10*x^2*y^2+8*y^4-8*x^4*y-10*x^3*y^2+x^2*y^3-3*y^5;% mu3_y...
    (0,0) = 1
mu4=@(x,y) 0.5*x^2-1.5*x^3+1.5*x^4-1.5*x^2*y^2-0.5*x^5+1.5*x^3*y^2+x^2*y^3;% mu4_xx(0,0) = 1
mu5=@(x,y) x*y-4*x^2*y-4*x*y^2+5*x^3*y+10*x^2*y^2+5*x*y^3-2*x^4*y-6*x^3*y^2-6*x^2*y^3-2*x*y^4;...
    % mu5_xy(0,0) = 1
mu6=@(x,y) 0.5*y^2-1.5*y^3-1.5*x^2*y^2+1.5*y^4+x^3*y^2+1.5*x^2*y^3-0.5*y^5;% mu6_yy(0,0) = 1
%%% for (1,0):
mu7=@(x,y) 10*x^3-15*x^4+15*x^2*y^2+6*x^5-15*x^3*y^2-15*x^2*y^3;% mu7(1,0) = 1
mu8=@(x,y) -4*x^3+7*x^4-3.5*x^2*y^2-3*x^5+3.5*x^3*y^2+3.5*x^2*y^3;% mu8_x(1,0) = 1
mu9=@(x,y) -5*x^2*y+14*x^3*y+18.5*x^2*y^2-8*x^4*y-18.5*x^3*y^2-13.5*x^2*y^3;% mu9_y(1,0) = 1
mu10=@(x,y) 0.5*x^3-x^4+0.25*x^2*y^2+0.5*x^5-0.25*x^3*y^2-0.25*x^2*y^3;% mu10_xx(1,0) = 1
mu11=@(x,y) x^2*y-3*x^3*y-3.5*x^2*y^2+2*x^4*y+3.5*x^3*y^2+2.5*x^2*y^3;% mu11_xy(1,0) = 1
mu12=@(x,y) 1.25*x^2*y^2-0.75*x^3*y^2-1.25*x^2*y^3;% mu12_yy(1,0) = 1
```

```
%%% for (0,1):
mu13=@(x,y) 10*y^3+15*x^2*y^2-15*y^4-15*x^3*y^2-15*x^2*y^3+6*y^5;% mu13(0,1) = 1
mu14=@(x,y) -5*x*y^2+18.5*x^2*y^2+14*x*y^3-13.5*x^3*y^2-18.5*x^2*y^3-8*x*y^4;% mu14_x(0,1) = 1
mu15=@(x,y) -4*y^3-3.5*x^2*y^2+7*y^4+3.5*x^3*y^2+3.5*x^2*y^3-3*y^5;% mu15_y(0,0) = 1
mu16=@(x,y) 1.25*x^2*y^2-1.25*x^3*y^2-0.75*x^2*y^3;% mu16_xx(0,1) = 1
mu17=@(x,y) x*y^2-3.5*x^2*y^2-3*x*y^3+2.5*x^3*y^2+3.5*x^2*y^3+2*x*y^4;% mu17_xy(0,1) = 1
mu18=@(x,y) 0.5*y^3+0.25*x^2*y^2-y^4-0.25*x^3*y^2-0.25*x^2*y^3+0.5*y^5;% mu18_yy(0,1) = 1
%%% Edge normal derivative functions
mu19=@(x,y) -16*x^2*y+32*x^3*y+32*x^2*y^2-16*x^4*y-32*x^3*y^2-16*x^2*y^3;% (.5,0), -mu19_y(0...
    .5,0) = 1
mu20=@(x,y) sqrt(2)*(-8*x^2*y^2+8*x^3*y^2+8*x^2*y^3);% (.5,.5), sqrt(0.5)(mu20_x(0.5,0.5)+ ...
    mu20_y(0.5,0.5)) = 1
mu21=@(x,y) -16*x*y^2+32*x^2*y^2+32*x*y^3-16*x^3*y^2-32*x^2*y^3-16*x*y^4;% (0,.5), -mu21_x...
    (0,0.5) = 1
%%% Group Together
mu={mu1,mu2,mu3,mu4,mu5,mu6,mu7,mu8,mu9,mu10,mu11,mu12,mu13,mu14,mu15,mu16,mu17,mu18,mu19,mu20...
    ,mu21};

%%% d/dx Partial Derivative of Basis Functions:
%%% for (0,0):
mu1x=@(x,y) -30*x^2+60*x^3-60*x*y^2-30*x^4+90*x^2*y^2+60*x*y^3;% mu1(0,0)=1
mu2x=@(x,y) 1-18*x^2-11*y^2+32*x^3+20*x*y^2+18*y^3-15*x^4+3*x^2*y^2-20*x*y^3-8*y^4; % mu2_x...
    (0,0)=1
mu3x=@(x,y) -22*x*y+54*x^2*y+20*x*y^2-32*x^3*y-30*x^2*y^2+2*x*y^3;% mu3_y(0,0) = 1
mu4x=@(x,y) x-4.5*x^2+6*x^3-3*x*y^2-2.5*x^4+4.5*x^2*y^2+2*x*y^3;% mu4_xx(0,0) = 1
mu5x=@(x,y) y-8*x*y-4*y^2+15*x^2*y+20*x*y^2+5*y^3-8*x^3*y-18*x^2*y^2-12*x*y^3-2*y^4;% mu5_xy...
    (0,0) = 1
mu6x=@(x,y) -3*x*y^2+3*x^2*y^2+3*x*y^3;% mu6_yy(0,0) = 1
%%% for (1,0):
mu7x=@(x,y) 30*x^2-60*x^3+30*x*y^2+30*x^4-45*x^2*y^2-30*x*y^3;% mu7(1,0) = 1
mu8x=@(x,y) -12*x^2+28*x^3-7*x*y^2-15*x^4+10.5*x^2*y^2+7*x*y^3;% mu8_x(1,0) = 1
mu9x=@(x,y) -10*x*y+42*x^2*y+37*x*y^2-32*x^3*y-55.5*x^2*y^2-27*x*y^3;% mu9_y(1,0) = 1
mu10x=@(x,y) 1.5*x^2-4*x^3+0.5*x*y^2+2.5*x^4-0.75*x^2*y^2-0.5*x*y^3;% mu10_xx(1,0) = 1
mu11x=@(x,y) 2*x*y-9*x^2*y-7*x*y^2+8*x^3*y+10.5*x^2*y^2+5*x*y^3;% mu11_xy(1,0) = 1
mu12x=@(x,y) 2.5*x*y^2-2.25*x^2*y^2-2.5*x*y^3;% mu12_yy(1,0) = 1
%%% for (0,1):
mu13x=@(x,y) 30*x*y^2-45*x^2*y^2-30*x*y^3;% mu13(0,1) = 1
mu14x=@(x,y) -5*y^2+37*x*y^2+14*y^3-40.5*x^2*y^2-37*x*y^3-8*y^4;% mu14_x(0,1) = 1
mu15x=@(x,y) -7*x*y^2+10.5*x^2*y^2+7*x*y^3;% mu15_y(0,0) = 1
mu16x=@(x,y) 2.5*x*y^2-3.75*x^2*y^2-1.5*x*y^3;% mu16_xx(0,1) = 1
mu17x=@(x,y) y^2-7*x*y^2-3*y^3+7.5*x^2*y^2+7*x*y^3+2*y^4;% mu17_xy(0,1) = 1
```

```
mu18x=@(x,y) 0.5*x*y^2-0.75*x^2*y^2-0.5*x*y^3;% mu18_yy(0,1) = 1
%%% Edge normal derivative functions
mu19x=@(x,y) -32*x*y+96*x^2*y+64*x*y^2-64*x^3*y-96*x^2*y^2-32*x*y^3;% (.5,0), -mu19_y(0.5,0) ...
    = 1
mu20x=@(x,y) sqrt(2)*(-16*x*y^2+24*x^2*y^2+16*x*y^3);% (.5,.5), sqrt(0.5)(mu20_x(0.5,0.5)+...
    mu20_y(0.5,0.5)) = 1
mu21x=@(x,y) -16*y^2+64*x*y^2+32*y^3-48*x^2*y^2-64*x*y^3-16*y^4;% (0,.5), -mu21_x(0,0.5) = 1
%%% Group Together
mux={mu1x,mu2x,mu3x,mu4x,mu5x,mu6x,mu7x,mu8x,mu9x,mu10x,mu11x,mu12x,mu13x,mu14x,mu15x,mu16x,...
    mu17x,mu18x,mu19x,mu20x,mu21x};

%%% d/dy Partial Derivative of Basis Functions:
%%% for (0,0):
mu1y=@(x,y) -30*y^2-60*x^2*y+60*y^3+60*x^3*y+90*x^2*y^2-30*y^4;% mu1(0,0)=1
mu2y=@(x,y) -22*x*y+20*x^2*y+54*x*y^2+2*x^3*y-30*x^2*y^2-32*x*y^3; % mu2_x(0,0)=1
mu3y=@(x,y) 1-11*x^2-18*y^2+18*x^3+20*x^2*y+32*y^3-8*x^4-20*x^3*y+3*x^2*y^2-15*y^4;% mu3_y...
    (0,0) = 1
mu4y=@(x,y) -3*x^2*y+3*x^3*y+3*x^2*y^2;% mu4_xx(0,0) = 1
mu5y=@(x,y) x-4*x^2-8*x*y+5*x^3+20*x^2*y+15*x*y^2-2*x^4-12*x^3*y-18*x^2*y^2-8*x*y^3;% mu5_xy...
    (0,0) = 1
mu6y=@(x,y) y-4.5*y^2-3*x^2*y+6*y^3+2*x^3*y+4.5*x^2*y^2-2.5*y^4;% mu6_yy(0,0) = 1
%%% for (1,0):
mu7y=@(x,y) 30*x^2*y-30*x^3*y-45*x^2*y^2;% mu7(1,0) = 1
mu8y=@(x,y) -7*x^2*y+7*x^3*y+10.5*x^2*y^2;% mu8_x(1,0) = 1
mu9y=@(x,y) -5*x^2+14*x^3+37*x^2*y-8*x^4-37*x^3*y-40.5*x^2*y^2;% mu9_y(1,0) = 1
mu10y=@(x,y) 0.5*x^2*y-0.5*x^3*y-0.75*x^2*y^2;% mu10_xx(1,0) = 1
mu11y=@(x,y) x^2-3*x^3-7*x^2*y+2*x^4+7*x^3*y+7.5*x^2*y^2;% mu11_xy(1,0) = 1
mu12y=@(x,y) 2.5*x^2*y-1.5*x^3*y-3.75*x^2*y^2;% mu12_yy(1,0) = 1
%%% for (0,1):
mu13y=@(x,y) 30*y^2+30*x^2*y-60*y^3-30*x^3*y-45*x^2*y^2+30*y^4;% mu13(0,1) = 1
mu14y=@(x,y) -10*x*y+37*x^2*y+42*x*y^2-27*x^3*y-55.5*x^2*y^2-32*x*y^3;% mu14_x(0,1) = 1
mu15y=@(x,y) -12*y^2-7*x^2*y+28*y^3+7*x^3*y+10.5*x^2*y^2-15*y^4;% mu15_y(0,0) = 1
mu16y=@(x,y) 2.5*x^2*y-2.5*x^3*y-2.25*x^2*y^2;% mu16_xx(0,1) = 1
mu17y=@(x,y) 2*x*y-7*x^2*y-9*x*y^2+5*x^3*y+10.5*x^2*y^2+8*x*y^3;% mu17_xy(0,1) = 1
mu18y=@(x,y) 1.5*y^2+0.5*x^2*y-4*y^3-0.5*x^3*y-0.75*x^2*y^2+2.5*y^4;% mu18_yy(0,1) = 1
%%% Edge normal derivative functions
mu19y=@(x,y) -16*x^2+32*x^3+64*x^2*y-16*x^4-64*x^3*y-48*x^2*y^2;% (.5,0), -mu19_y(0.5,0) = 1
mu20y=@(x,y) sqrt(2)*(-16*x^2*y+16*x^3*y+24*x^2*y^2);% (.5,.5), sqrt(0.5)(mu20_x(0.5,0.5)+...
    mu20_y(0.5,0.5)) = 1
mu21y=@(x,y) -32*x*y+64*x^2*y+96*x*y^2-32*x^3*y-96*x^2*y^2-64*x*y^3;% (0,.5), -mu21_x(0,0.5) ...
    = 1
```

```
%%% Group Together
muy={mu1y,mu2y,mu3y,mu4y,mu5y,mu6y,mu7y,mu8y,mu9y,mu10y,mu11y,mu12y,mu13y,mu14y,mu15y,mu16y,...
    mu17y,mu18y,mu19y,mu20y,mu21y};

%%% d^2/dx^2 Partial Derivative of Basis Functions:
%%% for (0,0):
mu1xx=@(x,y) -60*x+180*x^2-60*y^2-120*x^3+180*x*y^2+60*y^3;% mu1(0,0)=1
mu2xx=@(x,y) -36*x+96*x^2+20*y^2-60*x^3+6*x*y^2-20*y^3; % mu2_x(0,0)=1
mu3xx=@(x,y) -22*y+108*x*y+20*y^2-96*x^2*y-60*x*y^2+2*y^3;% mu3_y(0,0) = 1
mu4xx=@(x,y) 1-9*x+18*x^2-3*y^2-10*x^3+9*x*y^2+2*y^3;% mu4_xx(0,0) = 1
mu5xx=@(x,y) -8*y+30*x*y+20*y^2-24*x^2*y-36*x*y^2-12*y^3;% mu5_xy(0,0) = 1
mu6xx=@(x,y) -3*y^2+6*x*y^2+3*y^3;% mu6_yy(0,0) = 1
%%% for (1,0):
mu7xx=@(x,y) 60*x-180*x^2+30*y^2+120*x^3-90*x*y^2-30*y^3;% mu7(1,0) = 1
mu8xx=@(x,y) -24*x+84*x^2-7*y^2-60*x^3+21*x*y^2+7*y^3;% mu8_x(1,0) = 1
mu9xx=@(x,y) -10*y+84*x*y+37*y^2-96*x^2*y-111*x*y^2-27*y^3;% mu9_y(1,0) = 1
mu10xx=@(x,y) 3*x-12*x^2+0.5*y^2+10*x^3-1.5*x*y^2-0.5*y^3;% mu10_xx(1,0) = 1
mu11xx=@(x,y) 2*y-18*x*y-7*y^2+24*x^2*y+21*x*y^2+5*y^3;% mu11_xy(1,0) = 1
mu12xx=@(x,y) 2.5*y^2-4.5*x*y^2-2.5*y^3;% mu12_yy(1,0) = 1
%%% for (0,1):
mu13xx=@(x,y) 30*y^2-90*x*y^2-30*y^3;% mu13(0,1) = 1
mu14xx=@(x,y) 37*y^2-81*x*y^2-37*y^3;% mu14_x(0,1) = 1
mu15xx=@(x,y) -7*y^2+21*x*y^2+7*y^3;% mu15_y(0,0) = 1
mu16xx=@(x,y) 2.5*y^2-7.5*x*y^2-1.5*y^3;% mu16_xx(0,1) = 1
mu17xx=@(x,y) -7*y^2+15*x*y^2+7*y^3;% mu17_xy(0,1) = 1
mu18xx=@(x,y) 0.5*y^2-1.5*x*y^2-0.5*y^3;% mu18_yy(0,1) = 1
%%% Edge normal derivative functions
mu19xx=@(x,y) -32*y+192*x*y+64*y^2-192*x^2*y-192*x*y^2-32*y^3;% (.5,0), -mu19_y(0.5,0) = 1
mu20xx=@(x,y) sqrt(2)*(-16*y^2+48*x*y^2+16*y^3);% (.5,.5), sqrt(0.5)(mu20_x(0.5,0.5)+mu20_y(0...
    .5,0.5)) = 1
mu21xx=@(x,y) 64*y^2-96*x*y^2-64*y^3;% (0,.5), -mu21_x(0,0.5) = 1
%%% Group Together
muxx={mu1xx,mu2xx,mu3xx,mu4xx,mu5xx,mu6xx,mu7xx,mu8xx,mu9xx,mu10xx,mu11xx,mu12xx,mu13xx,mu14xx...
    ,mu15xx,mu16xx,mu17xx,mu18xx,mu19xx,mu20xx,mu21xx};

%%% d^2/dxdy Partial Derivative of Basis Functions:
%%% for (0,0):
mu1xy=@(x,y) -120*x*y+180*x^2*y+180*x*y^2;% mu1(0,0)=1
mu2xy=@(x,y) -22*y+40*x*y+54*y^2+6*x^2*y-60*x*y^2-32*y^3; % mu2_x(0,0)=1
mu3xy=@(x,y) -22*x+54*x^2+40*x*y-32*x^3-60*x^2*y+6*x*y^2;% mu3_y(0,0) = 1
mu4xy=@(x,y) -6*x*y+9*x^2*y+6*x*y^2;% mu4_xx(0,0) = 1
```

```
mu5xy=@(x,y) 1-8*x-8*y+15*x^2+40*x*y+15*y^2-8*x^3-36*x^2*y-36*x*y^2-8*y^3;% mu5_xy(0,0) = 1
mu6xy=@(x,y) -6*x*y+6*x^2*y+9*x*y^2;% mu6_yy(0,0) = 1
%%% for (1,0):
mu7xy=@(x,y) 60*x*y-90*x^2*y-90*x*y^2;% mu7(1,0) = 1
mu8xy=@(x,y) -14*x*y+21*x^2*y+21*x*y^2;% mu8_x(1,0) = 1
mu9xy=@(x,y) -10*x+42*x^2+74*x*y-32*x^3-111*x^2*y-81*x*y^2;% mu9_y(1,0) = 1
mu10xy=@(x,y) x*y-1.5*x^2*y-1.5*x*y^2;% mu10_xx(1,0) = 1
mu11xy=@(x,y) 2*x-9*x^2-14*x*y+8*x^3+21*x^2*y+15*x*y^2;% mu11_xy(1,0) = 1
mu12xy=@(x,y) 5*x*y-4.5*x^2*y-7.5*x*y^2;% mu12_yy(1,0) = 1
%%% for (0,1):
mu13xy=@(x,y) 60*x*y-90*x^2*y-90*x*y^2;% mu13(0,1) = 1
mu14xy=@(x,y) -10*y+74*x*y+42*y^2-81*x^2*y-111*x*y^2-32*y^3;% mu14_x(0,1) = 1
mu15xy=@(x,y) -14*x*y+21*x^2*y+21*x*y^2;% mu15_y(0,0) = 1
mu16xy=@(x,y) 5*x*y-7.5*x^2*y-4.5*x*y^2;% mu16_xx(0,1) = 1
mu17xy=@(x,y) 2*y-14*x*y-9*y^2+15*x^2*y+21*x*y^2+8*y^3;% mu17_xy(0,1) = 1
mu18xy=@(x,y) x*y-1.5*x^2*y-1.5*x*y^2;% mu18_yy(0,1) = 1
%%% Edge normal derivative functions
mu19xy=@(x,y) -32*x+96*x^2+128*x*y-64*x^3-192*x^2*y-96*x*y^2;% (.5,0), -mu19_y(0.5,0) = 1
mu20xy=@(x,y) sqrt(2)*(-32*x*y+48*x^2*y+48*x*y^2);% (.5,.5), sqrt(0.5)(mu20_x(0.5,0.5)+mu20_y...
    (0.5,0.5)) = 1
mu21xy=@(x,y) -32*y+128*x*y+96*y^2-96*x^2*y-192*x*y^2-64*y^3;% (0,.5), -mu21_x(0,0.5) = 1
%%% Group Together
muxy={mu1xy,mu2xy,mu3xy,mu4xy,mu5xy,mu6xy,mu7xy,mu8xy,mu9xy,mu10xy,mu11xy,mu12xy,mu13xy,mu14xy...
    ,mu15xy,mu16xy,mu17xy,mu18xy,mu19xy,mu20xy,mu21xy};

%%% d^2/dy^2 Partial Derivative of Basis Functions:
%%% for (0,0):
mu1yy=@(x,y) -60*y-60*x^2+180*y^2+60*x^3+180*x^2*y-120*y^3;% mu1(0,0)=1
mu2yy=@(x,y) -22*x+20*x^2+108*x*y+2*x^3-60*x^2*y-96*x*y^2; % mu2_x(0,0)=1
mu3yy=@(x,y) -36*y+20*x^2+96*y^2-20*x^3+6*x^2*y-60*y^3;% mu3_y(0,0) = 1
mu4yy=@(x,y) -3*x^2+3*x^3+6*x^2*y;% mu4_xx(0,0) = 1
mu5yy=@(x,y) -8*x+20*x^2+30*x*y-12*x^3-36*x^2*y-24*x*y^2;% mu5_xy(0,0) = 1
mu6yy=@(x,y) 1-9*y-3*x^2+18*y^2+2*x^3+9*x^2*y-10*y^3;% mu6_yy(0,0) = 1
%%% for (1,0):
mu7yy=@(x,y) 30*x^2-30*x^3-90*x^2*y;% mu7(1,0) = 1
mu8yy=@(x,y) -7*x^2+7*x^3+21*x^2*y;% mu8_x(1,0) = 1
mu9yy=@(x,y) 37*x^2-37*x^3-81*x^2*y;% mu9_y(1,0) = 1
mu10yy=@(x,y) 0.5*x^2-0.5*x^3-1.5*x^2*y;% mu10_xx(1,0) = 1
mu11yy=@(x,y) -7*x^2+7*x^3+15*x^2*y;% mu11_xy(1,0) = 1
mu12yy=@(x,y) 2.5*x^2-1.5*x^3-7.5*x^2*y;% mu12_yy(1,0) = 1
%%% for (0,1):
```

```
mu13yy=@(x,y) 60*y+30*x^2-180*y^2-30*x^3-90*x^2*y+120*y^3;% mu13(0,1) = 1
mu14yy=@(x,y) -10*x+37*x^2+84*x*y-27*x^3-111*x^2*y-96*x*y^2;% mu14_x(0,1) = 1
mu15yy=@(x,y) -24*y-7*x^2+84*y^2+7*x^3+21*x^2*y-60*y^3;% mu15_y(0,0) = 1
mu16yy=@(x,y) 2.5*x^2-2.5*x^3-4.5*x^2*y;% mu16_xx(0,1) = 1
mu17yy=@(x,y) 2*x-7*x^2-18*x*y+5*x^3+21*x^2*y+24*x*y^2;% mu17_xy(0,1) = 1
mu18yy=@(x,y) 3*y+0.5*x^2-12*y^2-0.5*x^3-1.5*x^2*y+10*y^3;% mu18_yy(0,1) = 1
%%% Edge normal derivative functions
mu19yy=@(x,y) 64*x^2-64*x^3-96*x^2*y;% (.5,0), -mu19_y(0.5,0) = 1
mu20yy=@(x,y) sqrt(2)*(-16*x^2+16*x^3+48*x^2*y);% (.5,.5), sqrt(0.5)(mu20_x(0.5,0.5)+mu20_y(0...
    .5,0.5)) = 1
mu21yy=@(x,y) -32*x+64*x^2+192*x*y-32*x^3-192*x^2*y-192*x*y^2;% (0,.5), -mu21_x(0,0.5) = 1
%%% Group Together
muyy={mu1yy,mu2yy,mu3yy,mu4yy,mu5yy,mu6yy,mu7yy,mu8yy,mu9yy,mu10yy,mu11yy,mu12yy,mu13yy,mu14yy...
    ,mu15yy,mu16yy,mu17yy,mu18yy,mu19yy,mu20yy,mu21yy};

%%% 2D Plate Quadrature Points and Weights (N = 28, D = 11)
%%% http://people.sc.fsu.edu/~jburkardt/datasets/quadrature_rules_tri/toms612_28_w.txt
pqpt=zeros(2,28);
pqpt(:,1)=[0.33333333333333333; 0.333333333333333333];
pqpt(:,2)=[0.9480217181434233; 0.02598914092828833];
pqpt(:,3)=[0.02598914092828833; 0.9480217181434233];
pqpt(:,4)=[0.02598914092828833; 0.02598914092828833];
pqpt(:,5)=[0.8114249947041546; 0.09428750264792270];
pqpt(:,6)=[0.09428750264792270; 0.8114249947041546];
pqpt(:,7)=[0.09428750264792270; 0.09428750264792270];
pqpt(:,8)=[0.01072644996557060; 0.4946367750172147];
pqpt(:,9)=[0.4946367750172147; 0.01072644996557060];
pqpt(:,10)=[0.4946367750172147; 0.4946367750172147];
pqpt(:,11)=[0.5853132347709715; 0.2073433826145142];
pqpt(:,12)=[0.2073433826145142; 0.5853132347709715];
pqpt(:,13)=[0.2073433826145142; 0.2073433826145142];
pqpt(:,14)=[0.1221843885990187; 0.4389078057004907];
pqpt(:,15)=[0.4389078057004907; 0.1221843885990187];
pqpt(:,16)=[0.4389078057004907; 0.4389078057004907];
pqpt(:,17)=[0.6779376548825902; 0.04484167758913055];
pqpt(:,18)=[0.6779376548825902; 0.27722066752827925];
pqpt(:,19)=[0.04484167758913055; 0.6779376548825902];
pqpt(:,20)=[0.04484167758913055; 0.27722066752827925];
pqpt(:,21)=[0.27722066752827925; 0.6779376548825902];
pqpt(:,22)=[0.27722066752827925; 0.04484167758913055];
pqpt(:,23)=[0.8588702812826364; 0.00000000000000000];
```

```
pqpt(:,24)=[0.8588702812826364; 0.1411297187173636];
pqpt(:,25)=[0.0000000000000000; 0.8588702812826364];
pqpt(:,26)=[0.0000000000000000; 0.1411297187173636];
pqpt(:,27)=[0.1411297187173636; 0.8588702812826364];
pqpt(:,28)=[0.1411297187173636; 0.0000000000000000];

pqwt=(1/2)*[0.08797730116222190; 0.008744311553736190; 0.008744311553736190;
0.008744311553736190; 0.03808157199393533; 0.03808157199393533;
0.03808157199393533; 0.01885544805613125; 0.01885544805613125;
0.01885544805613125; 0.07215969754474100; 0.07215969754474100;
0.07215969754474100; 0.06932913870553720; 0.06932913870553720;
0.06932913870553720; 0.04105631542928860; 0.04105631542928860;
0.04105631542928860; 0.04105631542928860; 0.04105631542928860;
0.04105631542928860; 0.007362383783300573; 0.007362383783300573;
0.007362383783300573; 0.007362383783300573; 0.007362383783300573;
0.007362383783300573];

%%% Compute Plate Shape Function Values at Quadrature Points
%%% Shape function values muVal[i,j] = mu{j}(r_i, s_i).  The columns
%%% correspond to the shape functions mu1–mu21 and the rows to
%%% the values of mu{j} at the quadrature points(qpt), etc.
pql=length(pqwt(:,1));
muVal=zeros(pql,21);
muxVal=zeros(pql,21);
muyVal=zeros(pql,21);
muxxVal=zeros(pql,21);
muxyVal=zeros(pql,21);
muyyVal=zeros(pql,21);
for i=1:pql
    for j=1:21
        muVal(i,j)=mu{j}(pqpt(1,i),pqpt(2,i));

        muxVal(i,j)=mux{j}(pqpt(1,i),pqpt(2,i));
        muyVal(i,j)=muy{j}(pqpt(1,i),pqpt(2,i));

        muxxVal(i,j)=muxx{j}(pqpt(1,i),pqpt(2,i));
        muxyVal(i,j)=muxy{j}(pqpt(1,i),pqpt(2,i));
        muyyVal(i,j)=muyy{j}(pqpt(1,i),pqpt(2,i));
    end
end
```

```
%%% Save values of mu_j at 6 nodes of reference triangle
xn=[0,1,0,.5,.5,0];
yn=[0,0,1,0,.5,.5];
muNVal=zeros(6,21);
for i=1:6
    for j=1:21
        muNVal(i,j)=mu{j}(xn(i),yn(i));
    end
end

%%% Compute the Plate Basis Function Constants
bCnst=BasisConstant(pMesh,mux,muy,muxx,muxy,muyy);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Solve for w1 ...
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Create the Fluid Stiffness Matrix
[fK,fAs]=StiffnessQ3D(Mesh,phiVal,phixVal,phiyVal,phizVal,etaVal,C,fqwt,lambda);
%[fK,fAs]=StiffnessQ3DSparse(Mesh,phiVal,phixVal,phiyVal,phizVal,etaVal,C,fqwt,lambda);

[C1,C2]=StiffnessQ3DB(Mesh,phiVal,phixVal,phiyVal,phizVal,fqwt,lambda);
C1T=transpose(C1);

%%% Create the Plate Bilinear Form (in parts)
pK1=StiffnessA2D(pMesh,bCnst,muxxVal,muxyVal,muyyVal,pqwt);
pK2=StiffnessAL2D(pMesh,bCnst,muVal,pqwt,lambda);

%%% For every basis function mu in the plate, we solve for f~(mu)
% Program assumes that G1, G2 = 0!
[muSol,muSolP]=fTilde(pMesh,Mesh,fK,bCnst,muNVal,phiVal,phixVal,phiyVal,phizVal,etaVal,C,fqwt,...
    lambda);
Z1=sparse(Nf,Nih);
Z2=sparse(Nih,Nf);
Z3=sparse(Nf,Nf);
fTildeA=[fAs,Z3,Z3,Z1; Z3,fAs,Z3,Z1; Z3,Z3,fAs,C1; Z2,Z2,C1T,C2];
pK3=StiffnessAFT2D(pMesh,muSol,lambda*fTildeA);

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Solve for w1 and c~ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Assemble the Bilinear Form Matrix BL
pK=pK1+pK2+pK3;
pB=BLFormA2D(pMesh,bCnst,muVal,pqwt);
pBT=transpose(pB);
```

```
Z=zeros(1,1);
BL=[pK,pBT;pB,Z];

%%% Create the Load Vector (in parts)
ff=@(x,y) lambda*w1star(x,y) + w2star(x,y);
F1=Load1A2D(pMesh,bCnst,ff,muVal,pqpt,pqwt);

%%% Solve f~(w1*) to get F2, note that w1* is in L^2_0
w1starG=zeros(Nih,1);
for i=1:Nih
    x1=Mesh.POS(Mesh.IHNodePtrs(i,1),1);
    y1=Mesh.POS(Mesh.IHNodePtrs(i,1),2);
    w1starG(i,1)=w1star(x1,y1);
end
w1starF=LoadFT(Mesh,w1starG,phiVal,phixVal,phiyVal,phizVal,etaVal,C,fqwt,lambda,0);
w1starSol=symmlq(fK,w1starF,1e-12,10000);
w1starSolP=w1starSol(1+3*Nf:3*Nf+Np,1);
w1starSol=w1starSol(1:3*Nf,1);
w1starSol=[w1starSol;w1starG];
F2=Load2A2D(pMesh,w1starSol,muSol,fTildeA);

%%% Solve mu~(u*) to get F3
ustarF=LoadQ3D(Mesh,ustar,phiVal,fqpt,fqwt);
ustarSol=symmlq(fK,ustarF,1e-12,10000);
ustarSolP=ustarSol(1+3*Nf:3*Nf+Np,1);
ustarSol=ustarSol(1:3*Nf,1);
ustarSol=[ustarSol;zeros(Nih,1)];
F3=Load3A2D(pMesh,ustarSol,muSol,fTildeA);

%%% Derive F4
F4=Load4A2D(Mesh,pMesh,ustar,muSol,phiVal,fqpt,fqwt);
F=F1+F2-F3+F4; % note - sign on F3
F=[F;0];

W1=symmlq(BL,F,1e-11,10000);
W1l=length(W1);
ctilde=W1(W1l,1);
W1=W1(1:W1l-1,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% w2=lambda w1 - w1* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% w2 is given in terms of now known quantities and thus the error is the
```

```
%%% same as the error for w1, so we don't recompute it.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Solve for u and p %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fTw1=lambda*muSol*W1;
pTw1=lambda*muSolP*W1;

uSol=fTw1-w1starSol+ustarSol1;
pSol=pTw1-w1starSolP+ustarSolP;
%%% Note the constant c~ needs to be included with p

%%%%%%%%%%%%%%%%%%%%%% Find error of solution and plot w1 %%%%%%%%%%%%%%%%%%%%%
colormap (flipud(jet))

[L2Err,Zw,ZW]=L2ErrorA2D(pMesh,bCnst,w1,W1,muVal,pqpt,pqwt);
tri1=delaunay(Zw(:,1),Zw(:,2));
tri2=delaunay(ZW(:,1),ZW(:,2));
string=sprintf('L^2 Error of w1 is %d',L2Err);
disp(string);
figure(1)
clf
positionVector1 = [0.07, 0.08, 0.38, 0.8];
subplot('Position',positionVector1)
trisurf(tri1,Zw(:,1),Zw(:,2),Zw(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('w1 True Solution Interpolation')
shading flat
positionVector2 = [0.6, 0.08, 0.38, 0.8];
subplot('Position',positionVector2)
trisurf(tri2,ZW(:,1),ZW(:,2),ZW(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('W1 FEM Approximation')
shading flat
B=colorbar;
set(B, 'Position', [.485 .08 .03 .8])

[H1xErr,Zw1,ZW1]=H1xErrorA2D(pMesh,bCnst,w1x,W1,muxVal,muyVal,pqpt,pqwt);
tri1=delaunay(Zw1(:,1),Zw1(:,2));
```

```
tri2=delaunay(ZW1(:,1),ZW1(:,2));
string=sprintf('H^1 Error of w1x is %d',H1xErr);
disp(string);
figure(2)
clf
positionVector1 = [0.07, 0.08, 0.38, 0.8];
subplot('Position',positionVector1)
trisurf(tri1,Zw1(:,1),Zw1(:,2),Zw1(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('w1x True Solution Interpolation')
shading flat
positionVector2 = [0.6, 0.08, 0.38, 0.8];
subplot('Position',positionVector2)
trisurf(tri2,ZW1(:,1),ZW1(:,2),ZW1(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('W1x FEM Approximation')
shading flat
B=colorbar;
set(B, 'Position', [.485 .08 .03 .8])

[H1yErr,Zw1,ZW1]=H1yErrorA2D(pMesh,bCnst,w1y,W1,muxVal,muyVal,pqpt,pqwt);
tri1=delaunay(Zw1(:,1),Zw1(:,2));
tri2=delaunay(ZW1(:,1),ZW1(:,2));
string=sprintf('H^1 Error of w1y is %d',H1yErr);
disp(string);
figure(3)
clf
positionVector1 = [0.07, 0.08, 0.38, 0.8];
subplot('Position',positionVector1)
trisurf(tri1,Zw1(:,1),Zw1(:,2),Zw1(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('w1y True Solution Interpolation')
shading flat
positionVector2 = [0.6, 0.08, 0.38, 0.8];
subplot('Position',positionVector2)
```

```
trisurf(tri2,ZW1(:,1),ZW1(:,2),ZW1(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('W1y FEM Approximation')
shading flat
B=colorbar;
set(B, 'Position', [.485 .08 .03 .8])

[H2xxErr,Zw2,ZW2]=H2xxErrorA2D(pMesh,bCnst,w1xx,W1,muxxVal,muxyVal,muyyVal,pqpt,pqwt);
tri1=delaunay(Zw2(:,1),Zw2(:,2));
tri2=delaunay(ZW2(:,1),ZW2(:,2));
string=sprintf('H^2 Error of w1xx is %d',H2xxErr);
disp(string);
figure(4)
clf
positionVector1 = [0.07, 0.08, 0.38, 0.8];
subplot('Position',positionVector1)
trisurf(tri1,Zw2(:,1),Zw2(:,2),Zw2(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('w1xx True Solution Interpolation')
shading flat
positionVector2 = [0.6, 0.08, 0.38, 0.8];
subplot('Position',positionVector2)
trisurf(tri2,ZW2(:,1),ZW2(:,2),ZW2(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('W1xx FEM Approximation')
shading flat
B=colorbar;
set(B, 'Position', [.485 .08 .03 .8])

[H2xyErr,Zw2,ZW2]=H2xyErrorA2D(pMesh,bCnst,w1xy,W1,muxxVal,muxyVal,muyyVal,pqpt,pqwt);
tri1=delaunay(Zw2(:,1),Zw2(:,2));
tri2=delaunay(ZW2(:,1),ZW2(:,2));
string=sprintf('H^2 Error of w1xy is %d',H2xyErr);
disp(string);
figure(5)
```

```
clf
positionVector1 = [0.07, 0.08, 0.38, 0.8];
subplot('Position',positionVector1)
trisurf(tri1,Zw2(:,1),Zw2(:,2),Zw2(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('w1xy True Solution Interpolation')
shading flat
positionVector2 = [0.6, 0.08, 0.38, 0.8];
subplot('Position',positionVector2)
trisurf(tri2,ZW2(:,1),ZW2(:,2),ZW2(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('W1xy FEM Approximation')
shading flat
B=colorbar;
set(B, 'Position', [.485 .08 .03 .8])

[H2yyErr,Zw2,ZW2]=H2yyErrorA2D(pMesh,bCnst,w1yy,W1,muxxVal,muxyVal,muyyVal,pqpt,pqwt);
tri1=delaunay(Zw2(:,1),Zw2(:,2));
tri2=delaunay(ZW2(:,1),ZW2(:,2));
string=sprintf('H^2 Error of w1yy is %d',H2yyErr);
disp(string);
figure(6)
clf
positionVector1 = [0.07, 0.08, 0.38, 0.8];
subplot('Position',positionVector1)
trisurf(tri1,Zw2(:,1),Zw2(:,2),Zw2(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('w1yy True Solution Interpolation')
shading flat
positionVector2 = [0.6, 0.08, 0.38, 0.8];
subplot('Position',positionVector2)
trisurf(tri2,ZW2(:,1),ZW2(:,2),ZW2(:,3))
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

```
title('Wlyy FEM Approximation')
shading flat
B=colorbar;
set(B, 'Position', [.485 .08 .03 .8])

%%%%%%%%%%%%%%%%%%% Find error of solution and plot u and p %%%%%%%%%%%%%%%%%%%
U1 = uSol(1:Nf,1);
G1 = zeros(Nc,1);
U2 = uSol(Nf+1:2*Nf,1);
G2 = zeros(Nc,1);
U3 = uSol(2*Nf+1:3*Nf,1);
G3 = zeros(Nc,1);
for i=1:Nih
    gni=Mesh.IHNodePtrs(i,1);
    cni=-Mesh.NodePtrs(gni,1);
    G3(cni,1)=uSol(3*Nf+i,1);
end
%%% Compute the L2 error of u:
e1=L2ErrorQ3D(Mesh,u1,U1,G1,phi,fqpt,fqwt);
e2=L2ErrorQ3D(Mesh,u2,U2,G2,phi,fqpt,fqwt);
e3=L2ErrorQ3D(Mesh,u3,U3,G3,phi,fqpt,fqwt);
l2err=sqrt(e1^2+e2^2+e3^2);
string=sprintf('L^2 Error of u is %d',l2err);
disp(string);
%%% Compute the H1 error of u:
e4=H1ErrorQ3D(Mesh,gradu1,U1,G1,phix,phiy,phiz,fqpt,fqwt);
e5=H1ErrorQ3D(Mesh,gradu2,U2,G2,phix,phiy,phiz,fqpt,fqwt);
e6=H1ErrorQ3D(Mesh,gradu3,U3,G3,phix,phiy,phiz,fqpt,fqwt);
h1err=sqrt(e4^2+e5^2+e6^2);
string=sprintf('H^1 Error of u is %d',h1err);
disp(string);
%%% Compute the L2 error of p:
pctilde=@(x,y,z) ctilde;
perr=@(x,y,z) p(x,y,z)-pctilde(x,y,z);
perr=L2ErrorQ3DP(Mesh,perr,pSol,eta,C,fqpt,fqwt);
string=sprintf('L^2 Error of p is %d',perr);
disp(string);
%%% Compute the integral of the divergence:
div=DivIntQ3D(Mesh,uSol,G3,phix,phiy,phiz,fqpt,fqwt);
string=sprintf('The integral of the fluid divergence is %d',div);
disp(string);
```

```
%%% Compute the L2 error of the divergence:
divL2=DivErrorQ3D(Mesh,uSol,G3,phix,phiy,phiz,fqpt,fqwt);
string=sprintf('L^2 Error of the fluid divergence is %d',divL2);
disp(string);

%%% "Plot" the solution
Nn=Mesh.nbNod;
U1plot=zeros(Nn,1);
U1plot(Mesh.FNodePtrs)=U1;
U1plot(Mesh.CNodePtrs)=G1;
U2plot=zeros(Nn,1);
U2plot(Mesh.FNodePtrs)=U2;
U2plot(Mesh.CNodePtrs)=G2;
U3plot=zeros(Nn,1);
U3plot(Mesh.FNodePtrs)=U3;
U3plot(Mesh.CNodePtrs)=G3;

%%% Create interpolation U
U1int=zeros(Nn,1);
U2int=zeros(Nn,1);
U3int=zeros(Nn,1);
for i=1:Mesh.nbNod
    xc=Mesh.POS(i,1);
    yc=Mesh.POS(i,2);
    zc=Mesh.POS(i,3);
    U1int(i,1)=u1(xc,yc,zc);
    U2int(i,1)=u2(xc,yc,zc);
    U3int(i,1)=u3(xc,yc,zc);
end
%%% Create interpolation p
Pint=zeros(Np,1);
PX=Mesh.POS(Mesh.PNodePtrs,1);
PY=Mesh.POS(Mesh.PNodePtrs,2);
PZ=Mesh.POS(Mesh.PNodePtrs,3);
for i=1:Np
    Pint(i,1)=p(PX(i,1),PY(i,1),PY(i,1));
end

%%% Find nodal values for pSol
Pcnst=ctilde - sum(pSol.*C);
Pplot=pSol+Pcnst*ones(Np,1);
```

```
[fluidX,fluidY,fluidZ] = meshgrid(0:.025:1,0:.025:1,-1:.025:0);
U1plotF = scatteredInterpolant(Mesh.POS(:,1),Mesh.POS(:,2),Mesh.POS(:,3),U1plot);
U1intF = scatteredInterpolant(Mesh.POS(:,1),Mesh.POS(:,2),Mesh.POS(:,3),U1int);
U1plotV=U1plotF(fluidX,fluidY,fluidZ);
U1intV=U1intF(fluidX,fluidY,fluidZ);
U2plotF = scatteredInterpolant(Mesh.POS(:,1),Mesh.POS(:,2),Mesh.POS(:,3),U2plot);
U2intF = scatteredInterpolant(Mesh.POS(:,1),Mesh.POS(:,2),Mesh.POS(:,3),U2int);
U2plotV=U2plotF(fluidX,fluidY,fluidZ);
U2intV=U2intF(fluidX,fluidY,fluidZ);
U3plotF = scatteredInterpolant(Mesh.POS(:,1),Mesh.POS(:,2),Mesh.POS(:,3),U3plot);
U3intF = scatteredInterpolant(Mesh.POS(:,1),Mesh.POS(:,2),Mesh.POS(:,3),U3int);
U3plotV=U3plotF(fluidX,fluidY,fluidZ);
U3intV=U3intF(fluidX,fluidY,fluidZ);
PplotF = scatteredInterpolant(PX,PY,PZ,Pplot);
PintF = scatteredInterpolant(PX,PY,PZ,Pint);
PplotV=PplotF(fluidX,fluidY,fluidZ);
PintV=PintF(fluidX,fluidY,fluidZ);

figure(7) %%% Fluid U1 plot
clf
xslice = [.5,1];
yslice = [.5,1];
zslice = [0,-.5];
positionVector1 = [0.07, 0.08, 0.38, 0.85];
subplot('Position',positionVector1)
slice(fluidX,fluidY,fluidZ,U1plotV,xslice,yslice,zslice);
shading flat;
view([-30 14])
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('U1 FEM Approximation')
B=colorbar;
positionVector2 = [0.6, 0.08, 0.38, 0.85];
subplot('Position',positionVector2)
slice(fluidX,fluidY,fluidZ,U1intV,xslice,yslice,zslice);
shading flat;
view([-30 14])
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

```
title('u1 Exact Solution')
set(B, 'Position', [.485 .08 .03 .8])

figure(8) %%% Fluid U2 plot
clf
xslice = [.5,1];
yslice = [.5,1];
zslice = [0,-.5];
positionVector1 = [0.07, 0.08, 0.38, 0.85];
subplot('Position',positionVector1)
slice(fluidX,fluidY,fluidZ,U2plotV,xslice,yslice,zslice);
shading flat;
view([-33 18])
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('U2 FEM Approximation')
B=colorbar;
positionVector2 = [0.6, 0.08, 0.38, 0.85];
subplot('Position',positionVector2)
slice(fluidX,fluidY,fluidZ,U2intV,xslice,yslice,zslice);
shading flat;
view([-33 18])
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('u2 Exact Solution')
set(B, 'Position', [.485 .08 .03 .8])

figure(9) %%% Fluid U3 plot
clf
xslice = [.5,1];
yslice = [.5,1];
zslice = [0,-.5];
positionVector1 = [0.07, 0.08, 0.38, 0.85];
subplot('Position',positionVector1)
slice(fluidX,fluidY,fluidZ,U3plotV,xslice,yslice,zslice);
shading flat;
view([-25 12])
xlabel('X axis')
ylabel('Y axis')
```

```
zlabel('Z axis')
title('U3 FEM Approximation')
B=colorbar;
positionVector2 = [0.6, 0.08, 0.38, 0.85];
subplot('Position',positionVector2)
slice(fluidX,fluidY,fluidZ,U3intV,xslice,yslice,zslice);
shading flat;
view([-25 12])
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('u3 Exact Solution')
set(B, 'Position', [.485 .08 .03 .8])

figure(10) %%% Fluid Pressure plot
clf
xslice = [.5,1];
yslice = [.5,1];
zslice = [0,-.5];
positionVector1 = [0.07, 0.08, 0.38, 0.85];
subplot('Position',positionVector1)
slice(fluidX,fluidY,fluidZ,PplotV,xslice,yslice,zslice);
shading flat;
view([-25 12])
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('Pressure FEM Approximation')
B=colorbar;
positionVector2 = [0.6, 0.08, 0.38, 0.85];
subplot('Position',positionVector2)
slice(fluidX,fluidY,fluidZ,PintV,xslice,yslice,zslice);
shading flat;
view([-25 12])
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
title('Pressure Exact Solution')
set(B, 'Position', [.485 .08 .03 .8])
```

gmsh2mesh3D.m

```
function Mesh=gmsh2mesh3D(filename,pn)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The program takes a mesh file from GMSH and converts it to the desired
% FEM data structure.  filename is the .msh file from gmsh
% pn is the physical surface number of the plate Omega from GMSH
% Mesh is the output file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Mesh=load_gmsh2(filename,[8 9 11 15]);  % Create the mesh file
%%% Encode in the fourth column of the nodes file Mesh.POS whether that
%%% node is free=1 or constrained=-1;  only nodes on a tetrahedron
%%% along the boundary surface can be constrained.
Mesh.POS(:,4)=1;
nbTriangles6=Mesh.nbTriangles6;
for i=1:nbTriangles6
    if Mesh.TRIANGLES6(i,7)==pn  %%% a plate triangle -> inhomogeneous
        for j=1:6
            pnd=Mesh.TRIANGLES6(i,j);
            Mesh.POS(pnd,4)=-10;
        end
    else % a no-slip triangle -> constrained node
        for j=1:6
            cnd=Mesh.TRIANGLES6(i,j);
            Mesh.POS(cnd,4)=-1;
        end
    end
end
%%% Create the pointers to and from Mesh.POS for free boundary nodes and
%%% constrained boundary nodes.
nbNod=Mesh.nbNod;
Z=Mesh.POS(:,4);
ZZ=find(Z>0);
nn=length(Z)-length(ZZ);
Mesh.NodePtrs=zeros(nbNod,3);
Mesh.FNodePtrs=zeros(length(ZZ),1);
Mesh.CNodePtrs=zeros(nn,1);
fbindex=1;
cbindex=1;
ihindex=1;
for i=1:nbNod
```

```
    if Mesh.POS(i,4)>0
        Mesh.FNodePtrs(fbindex,1)=i;
        Mesh.NodePtrs(i,1)=fbindex;    %+ for free
        fbindex=fbindex+1;
    else
        Mesh.CNodePtrs(cbindex,1)=i;
        Mesh.NodePtrs(i,1)=-cbindex;   %- for constrained
        cbindex=cbindex+1;

        if Mesh.POS(i,3)==0
            if Mesh.POS(i,1)>0 && Mesh.POS(i,1)<1
                if Mesh.POS(i,2)>0 && Mesh.POS(i,2)<1
                    Mesh.IHNodePtrs(ihindex,1)=i;
                    Mesh.NodePtrs(i,3)=ihindex;
                    ihindex=ihindex+1;
                end
            end
        end
    end
end
%%% Encode in the fifth column of the nodes file Mesh.POS whether that
%%% node is a pressure node or not. Because the pressure basis functions
%%% are linear, only the first four nodes of a tetraheral element are
%%% pressure nodes.  This is indicated by a 2 for a pressure node
%%% and a 0 for a non-pressure node.  There are no boundary conditions
%%% imposed on the pressure nodes.
Mesh.POS(:,5)=0;
for i=1:Mesh.nbTets10
    for j=1:4
        pnode=Mesh.TETS10(i,j);
        Mesh.POS(pnode,5)=2;
    end
end
%%% Create the pointers to and from Mesh.POS for the pressure nodes and
%%% non-pressure nodes.
pnindex=1;  % The pressure node index
npnindex=1; % The non-pressure node index
for i=1:nbNod
    if Mesh.POS(i,5)>1
        Mesh.PNodePtrs(pnindex,1)=i;
        Mesh.NodePtrs(i,2)=pnindex;    %+ for pressure nodes
```

```
            pnindex=pnindex+1;
        else
            Mesh.NPNodePtrs(npnindex,1)=i;
            Mesh.NodePtrs(i,2)=-npnindex;  %- for non-pressure nodes
            npnindex=npnindex+1;
        end
end
Mesh.TETS10(:,12)=ones(Mesh.nbTets10,1);
%%% Note which tetrahedra have at least one vertex on Omega
for i=1:Mesh.nbTets10
    for j=1:4
        ndnum=Mesh.TETS10(i,j);
        if Mesh.POS(ndnum,3)==0
            Mesh.TETS10(i,12)=-2;
        end
    end
end
%%% Note which tetrahedra have one face on the plate Omega
for i=1:Mesh.nbTriangles6
    %%% for any plate triangle, find the corresponding tetrahedron
    if Mesh.TRIANGLES6(i,7)==pn
       j=1;
       while j < Mesh.nbTets10+1
           Tri=Mesh.TRIANGLES6(i,1:3);
           Tet=Mesh.TETS10(j,1:4);
           Cmn=ismember(Tri,Tet);
           if sum(Cmn)==3
              Mesh.TETS10(j,12)=-4;  %%% This tetrahedron borders Omega
              j=Mesh.nbTets10+2;
           end
           j=j+1;
       end
    end
end
```

Mesh3dToPlate2D.m

```
function [pMesh] = Mesh3DToPlate2D(Mesh,pn)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function takes the 3D mesh file Mesh and creates the 2D plate mesh
```

```
%%% file pMesh as well as pointers between the two data structures.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pMesh.Types = Mesh.Types;
pMesh.nbType = Mesh.nbType;
pMesh.MAX = [1 1 0];
pMesh.MIN = [0 0 0];
%%% Nodes on Omega
Z=Mesh.POS(:,3);
ZZ=find(Z<0);
nn=length(Z)-length(ZZ);
pMesh.PtrsTo2DNodes=zeros(Mesh.nbNod,1);
pMesh.PtrsTo3DNodes=zeros(nn,1);
j=1;
for i = 1:Mesh.nbNod
    if Mesh.POS(i,3)==0
        pMesh.POS(j,1:5)=Mesh.POS(i,1:5);
        pMesh.PtrsTo3DNodes(j,1)=i;
        pMesh.PtrsTo2DNodes(i,1)=j;
        j=j+1;
    end
end
pMesh.nbNod=length(pMesh.POS(:,1));
%%% Triangles on Omega
j=1;
for i = 1:Mesh.nbTriangles6
    if Mesh.TRIANGLES6(i,7)==pn
        pMesh.TRIANGLES6(j,1:7)=Mesh.TRIANGLES6(i,1:7);
        j=j+1;
    end
end
pMesh.nbTriangles6=length(pMesh.TRIANGLES6(:,1));
%%% Boundary Lines of Omega
pMesh.nbLines3=Mesh.nbLines3;
pMesh.LINES3=Mesh.LINES3;
Nbn=2*pMesh.nbLines3;
Nfn=pMesh.nbNod-Nbn;
Nmn=(pMesh.nbTriangles6*3+pMesh.nbLines3)/2;
Nfmn=Nmn-(Nbn/2);
Nfvn=Nfn-Nfmn;
Nbxxn=(Nbn/4)-2;
```

```
%%% Building Node Pointers for Boundary and Free Nodes
pMesh.NodePtrs=zeros(pMesh.nbNod,3); % 3rd column is Bxx and Byy node numbers
pMesh.BVNodePtrs=zeros(Nbn/2,2);
pMesh.BMNodePtrs=zeros(Nbn/2,2);
pMesh.FVNodePtrs=zeros(Nfvn,2);
pMesh.FMNodePtrs=zeros(Nfmn,2);
pMesh.BxxNodePtrs=zeros(Nbxxn,1);
pMesh.ByyNodePtrs=zeros(Nbxxn,1);
BVindex=1;
BMindex=1;
FVindex=1;
FMindex=1;
Bxxindex=1;
Byyindex=1;
for i=1:pMesh.nbNod
    if pMesh.POS(i,4)==-1  %Boundary Node of Omega
        if pMesh.POS(i,5)==2 %Boundary Vertex Node
            pMesh.BVNodePtrs(BVindex,1)=i;
            pMesh.NodePtrs(i,1)=BVindex;
            pMesh.NodePtrs(i,2)=3;  %3 indicates Boundary Vertex Node
            BVindex=BVindex+1;
            if pMesh.POS(i,1)>0 && pMesh.POS(i,1)<1 %0<x<1 is Byy Node
                pMesh.POS(i,4)=-4; % -4 indicates Byy
                pMesh.ByyNodePtrs(Byyindex,1)=i;
                pMesh.NodePtrs(i,3)=-Byyindex;
                Byyindex=Byyindex+1;
            elseif pMesh.POS(i,2)>0 && pMesh.POS(i,2)<1 %0<y<1 is Bxx Node
                pMesh.POS(i,4)=-3; % -3 indicates Bxx
                pMesh.BxxNodePtrs(Bxxindex,1)=i;
                pMesh.NodePtrs(i,3)=Bxxindex;
                Bxxindex=Bxxindex+1;
            else % Must be a corner node
                pMesh.POS(i,4)=-2; % -2 indicates corner node
            end
        %%% Note: 0 in NodePtrs(i,3) indicates unused boundary node either
        %%% a corner node or a midpoint node.
        else %It must be a Boundary Midpoint Node of Omega
            pMesh.BMNodePtrs(BMindex,1)=i;
            pMesh.NodePtrs(i,1)=BMindex;
            pMesh.NodePtrs(i,2)=4;  %4 indicates Boundary Midpoint Node
            BMindex=BMindex+1;
```

```
        end
    else %It must be a Free Node of Omega
        if pMesh.POS(i,5)==2 %Free Vertex Node
            pMesh.FVNodePtrs(FVindex,1)=i;
            pMesh.NodePtrs(i,1)=FVindex;
            pMesh.NodePtrs(i,2)=1;  %1 indicates Free Vertex Node
            FVindex=FVindex+1;
        else %It must be a Free Midpoint Node of Omega
            pMesh.FMNodePtrs(FMindex,1)=i;
            pMesh.NodePtrs(i,1)=FMindex;
            pMesh.NodePtrs(i,2)=2;  %2 indicates Free Midpoint Node
            FMindex=FMindex+1;
        end
    end
end
%%% Fix the node numbers on Triangles
for i=1:pMesh.nbTriangles6
    for j=1:6
        Nnum3D=pMesh.TRIANGLES6(i,j);
        Nnum2D=pMesh.PtrsTo2DNodes(Nnum3D);
        pMesh.TRIANGLES6(i,j)=Nnum2D;
    end
end
%%% Fix the node numbers on Boundary Edges (LINES3)
for i=1:pMesh.nbLines3
    for j=1:3
        Nnum3D=pMesh.LINES3(i,j);
        Nnum2D=pMesh.PtrsTo2DNodes(Nnum3D);
        pMesh.LINES3(i,j)=Nnum2D;
    end
end
%%% Create the edge data structure
pMesh.nbEdges3=(pMesh.nbTriangles6*3+pMesh.nbLines3)/2;
pMesh.EDGES3=zeros(pMesh.nbEdges3,4);
pMesh.T2EPtrs=zeros(pMesh.nbTriangles6,3);
pMesh.E2TPtrs=zeros(pMesh.nbEdges3,2);
tempflags=zeros(pMesh.nbNod,1);
eIndex=1;

for k=1:pMesh.nbTriangles6
    E1=pMesh.TRIANGLES6(k,[1,2,4]);
```

```
        E2=pMesh.TRIANGLES6(k,[2,3,5]);
        E3=pMesh.TRIANGLES6(k,[3,1,6]);
        %%% Edge 1
        if tempflags(E1(1,3),1)==0
            pMesh.EDGES3(eIndex,1:3)=E1;
            tempflags(E1(1,3),1)=eIndex;
            pMesh.T2EPtrs(k,1)=eIndex;
            pMesh.E2TPtrs(eIndex,1)=k;
            eIndex=eIndex+1;
        else
            tfIndex=tempflags(E1(1,3),1);
            pMesh.T2EPtrs(k,1)=tfIndex;
            pMesh.E2TPtrs(tfIndex,2)=k;
        end
        %%% Edge 2
        if tempflags(E2(1,3),1)==0
            pMesh.EDGES3(eIndex,1:3)=E2;
            tempflags(E2(1,3),1)=eIndex;
            pMesh.T2EPtrs(k,2)=eIndex;
            pMesh.E2TPtrs(eIndex,1)=k;
            eIndex=eIndex+1;
        else
            tfIndex=tempflags(E2(1,3),1);
            pMesh.T2EPtrs(k,2)=tfIndex;
            pMesh.E2TPtrs(tfIndex,2)=k;
        end
        %%% Edge 3
        if tempflags(E3(1,3),1)==0
            pMesh.EDGES3(eIndex,1:3)=E3;
            tempflags(E3(1,3),1)=eIndex;
            pMesh.T2EPtrs(k,3)=eIndex;
            pMesh.E2TPtrs(eIndex,1)=k;
            eIndex=eIndex+1;
        else
            tfIndex=tempflags(E3(1,3),1);
            pMesh.T2EPtrs(k,3)=tfIndex;
            pMesh.E2TPtrs(tfIndex,2)=k;
        end
    end
    %%% Determine which edges are boundary edges and store the
    %%% boundary number (1-4) in column 4.
```

```
for e=1:pMesh.nbEdges3
    emn=pMesh.EDGES3(e,3);
    e3x=pMesh.POS(emn,1);
    e3y=pMesh.POS(emn,2);
    if e3y==0
        pMesh.EDGES3(e,4)=1;
    end
    if e3x==1
        pMesh.EDGES3(e,4)=2;
    end
    if e3y==1
        pMesh.EDGES3(e,4)=3;
    end
    if e3x==0
        pMesh.EDGES3(e,4)=4;
    end
    % Else it is an interior edge
end
%%% Create Unit Normal Vectors for each edge
%%% Note that 3rd column is midpoint node#
pMesh.NORMALS=pMesh.EDGES3;
for e=1:pMesh.nbEdges3
    en1=pMesh.EDGES3(e,1);
    en2=pMesh.EDGES3(e,2);
    x1=pMesh.POS(en1,1);
    y1=pMesh.POS(en1,2);
    x2=pMesh.POS(en2,1);
    y2=pMesh.POS(en2,2);
    PerpEdge=[y2-y1,x1-x2];
    unitperp=PerpEdge/norm(PerpEdge);
    pMesh.NORMALS(e,1:2)=unitperp;
end
```

BasisConstant.m

```
function bCnst=BasisConstant(pMesh,mux,muy,muxx,muxy,muyy)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% This function creates the basis function correction constants so that
%%% the edge normal derivatives are zero for each basis function.
```

```
%%%
%%% pMesh is the Plate Mesh Structure file
%%%
%%% bCor1(nT,n,i) is an 3D array where nT is the triangle number, n is
%%% the local node number, and i corresponds to the correction constants
%%% alpha, beta (see p261-265 of (Solin's PDE and the FEM) for details.)
%%% bCor2-bCor7 are defined similarly and combined into bCnst{1-7}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

bCor1=zeros(pMesh.nbTriangles6,3,3);%% mu1, mu7,  mu13
bCor2=zeros(pMesh.nbTriangles6,3,5);%% mu2, mu8,  mu14
bCor3=zeros(pMesh.nbTriangles6,3,5);%% mu3, mu9,  mu15
bCor4=zeros(pMesh.nbTriangles6,3,6);%% mu4, mu10, mu16
bCor5=zeros(pMesh.nbTriangles6,3,6);%% mu5, mu11, mu17
bCor6=zeros(pMesh.nbTriangles6,3,6);%% mu6, mu12, mu18
bCor7=zeros(pMesh.nbTriangles6,3,3);%% mu19, mu20, mu21
%%% Compute Gradients of shape functions on Reference Triangle
%%% row i is midpoints, column is shape function j
muxVal=zeros(6,21); %%%Nodes 1-3, Midpoints 4-6
muyVal=zeros(6,21);
for j=1:21
    %%% Evaluate at Nodes
    muxVal(1,j)=mux{j}(0,0);
    muxVal(2,j)=mux{j}(1,0);
    muxVal(3,j)=mux{j}(0,1);
    muyVal(1,j)=muy{j}(0,0);
    muyVal(2,j)=muy{j}(1,0);
    muyVal(3,j)=muy{j}(0,1);
    %%% Evaluate at Midpoints
    muxVal(4,j)=mux{j}(.5,0);
    muxVal(5,j)=mux{j}(.5,.5);
    muxVal(6,j)=mux{j}(0,.5);
    muyVal(4,j)=muy{j}(.5,0);
    muyVal(5,j)=muy{j}(.5,.5);
    muyVal(6,j)=muy{j}(0,.5);
end
%%% Compute 2nd Partials of shape functions on Reference Triangle
%%% row i is midpoints, column is shape function j
muxxVal=zeros(3,21); %%%Nodes 1-3
muxyVal=zeros(3,21);
muyyVal=zeros(3,21);
```

```
for j=1:21
    %%% Evaluate at Nodes
    muxxVal(1,j)=muxx{j}(0,0);
    muxxVal(2,j)=muxx{j}(1,0);
    muxxVal(3,j)=muxx{j}(0,1);
    muxyVal(1,j)=muxy{j}(0,0);
    muxyVal(2,j)=muxy{j}(1,0);
    muxyVal(3,j)=muxy{j}(0,1);
    muyyVal(1,j)=muyy{j}(0,0);
    muyyVal(2,j)=muyy{j}(1,0);
    muyyVal(3,j)=muyy{j}(0,1);
end
%%% Loop over Triangles
for k=1:pMesh.nbTriangles6
    nodes=pMesh.TRIANGLES6(k,1:3);
%%%(x,y) coordinates of (local) nodes 1, 2, 3(Vertices of T_k)
    x1=pMesh.POS(nodes(1),1);
    y1=pMesh.POS(nodes(1),2);
    x2=pMesh.POS(nodes(2),1);
    y2=pMesh.POS(nodes(2),2);
    x3=pMesh.POS(nodes(3),1);
    y3=pMesh.POS(nodes(3),2);
%%% Compute transormation J: [x,y] = [x1, y1] + Ju for u in reference
%%% triangle, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1; y2-y1,y3-y1];
    JkIT=InvTran1(Jk);
    detJk=det(Jk);
    %%% Ak gives the transformed 2nd partial derivatives when applied to
    %%% the vector [muxx; muxy; muyy]
    Ak=((1/detJk)^2)*[Jk(2,2)^2, -2*Jk(2,1)*Jk(2,2), Jk(2,1)^2;
        -Jk(1,2)*Jk(2,2),(Jk(1,1)*Jk(2,2)+Jk(1,2)*Jk(2,1)), -Jk(1,1)*Jk(2,1);
        Jk(1,2)^2, -2*Jk(1,1)*Jk(1,2), Jk(1,1)^2];
    e1=pMesh.T2EPtrs(k,1);
    e2=pMesh.T2EPtrs(k,2);
    e3=pMesh.T2EPtrs(k,3);
    nu1=[pMesh.NORMALS(e1,1);pMesh.NORMALS(e1,2)];
    nu2=[pMesh.NORMALS(e2,1);pMesh.NORMALS(e2,2)];
    nu3=[pMesh.NORMALS(e3,1);pMesh.NORMALS(e3,2)];
    db19nu1=sum(nu1.*(JkIT*[muxVal(4,19);muyVal(4,19)]));
    db20nu2=sum(nu2.*(JkIT*[muxVal(5,20);muyVal(5,20)]));
    db21nu3=sum(nu3.*(JkIT*[muxVal(6,21);muyVal(6,21)]));
```

```
%%% Compute bCor1 for mu1, mu7, mu13
    db1nu1=sum(nu1.*(JkIT*[muxVal(4,1);muyVal(4,1)]));
    db1nu2=sum(nu2.*(JkIT*[muxVal(5,1);muyVal(5,1)]));
    db1nu3=sum(nu3.*(JkIT*[muxVal(6,1);muyVal(6,1)]));
    db7nu1=sum(nu1.*(JkIT*[muxVal(4,7);muyVal(4,7)]));
    db7nu2=sum(nu2.*(JkIT*[muxVal(5,7);muyVal(5,7)]));
    db7nu3=sum(nu3.*(JkIT*[muxVal(6,7);muyVal(6,7)]));
    db13nu1=sum(nu1.*(JkIT*[muxVal(4,13);muyVal(4,13)]));
    db13nu2=sum(nu2.*(JkIT*[muxVal(5,13);muyVal(5,13)]));
    db13nu3=sum(nu3.*(JkIT*[muxVal(6,13);muyVal(6,13)]));
    bCor1(k,1,:)=[-db1nu1/db19nu1;-db1nu2/db20nu2;-db1nu3/db21nu3];
    bCor1(k,2,:)=[-db7nu1/db19nu1;-db7nu2/db20nu2;-db7nu3/db21nu3];
    bCor1(k,3,:)=[-db13nu1/db19nu1;-db13nu2/db20nu2;-db13nu3/db21nu3];
%%% Compute bCor2 and bCor3 for mu2-3 (i.e. mu2x=1 at (0,0), etc)
    db2nu1=sum(nu1.*(JkIT*[muxVal(4,2);muyVal(4,2)]));
    db2nu2=sum(nu2.*(JkIT*[muxVal(5,2);muyVal(5,2)]));
    db2nu3=sum(nu3.*(JkIT*[muxVal(6,2);muyVal(6,2)]));
    db3nu1=sum(nu1.*(JkIT*[muxVal(4,3);muyVal(4,3)]));
    db3nu2=sum(nu2.*(JkIT*[muxVal(5,3);muyVal(5,3)]));
    db3nu3=sum(nu3.*(JkIT*[muxVal(6,3);muyVal(6,3)]));
    db2x=JkIT(1,:)*[muxVal(1,2);muyVal(1,2)];
    db2y=JkIT(2,:)*[muxVal(1,2);muyVal(1,2)];
    db3x=JkIT(1,:)*[muxVal(1,3);muyVal(1,3)];
    db3y=JkIT(2,:)*[muxVal(1,3);muyVal(1,3)];
    bCor2(k,1,:)=[db2x,db3x,0,0,0; db2y,db3y,0,0,0; db2nu1,db3nu1,db19nu1,0,0;
        db2nu2,db3nu2,0,db20nu2,0; db2nu3,db3nu3,0,0,db21nu3]\[1;0;0;0;0];
    bCor3(k,1,:)=[db2x,db3x,0,0,0; db2y,db3y,0,0,0; db2nu1,db3nu1,db19nu1,0,0;
        db2nu2,db3nu2,0,db20nu2,0; db2nu3,db3nu3,0,0,db21nu3]\[0;1;0;0;0];
%%% Compute bCor2 and bCor3 for mu8-9 (i.e. mu8x=1 at (1,0), etc)
    db8nu1=sum(nu1.*(JkIT*[muxVal(4,8);muyVal(4,8)]));
    db8nu2=sum(nu2.*(JkIT*[muxVal(5,8);muyVal(5,8)]));
    db8nu3=sum(nu3.*(JkIT*[muxVal(6,8);muyVal(6,8)]));
    db9nu1=sum(nu1.*(JkIT*[muxVal(4,9);muyVal(4,9)]));
    db9nu2=sum(nu2.*(JkIT*[muxVal(5,9);muyVal(5,9)]));
    db9nu3=sum(nu3.*(JkIT*[muxVal(6,9);muyVal(6,9)]));
    db8x=JkIT(1,:)*[muxVal(2,8);muyVal(2,8)];
    db8y=JkIT(2,:)*[muxVal(2,8);muyVal(2,8)];
    db9x=JkIT(1,:)*[muxVal(2,9);muyVal(2,9)];
    db9y=JkIT(2,:)*[muxVal(2,9);muyVal(2,9)];
    bCor2(k,2,:)=[db8x,db9x,0,0,0; db8y,db9y,0,0,0; db8nu1,db9nu1,db19nu1,0,0;
        db8nu2,db9nu2,0,db20nu2,0; db8nu3,db9nu3,0,0,db21nu3]\[1;0;0;0;0];
```

```
    bCor3(k,2,:)=[db8x,db9x,0,0,0; db8y,db9y,0,0,0; db8nu1,db9nu1,db19nu1,0,0;
        db8nu2,db9nu2,0,db20nu2,0; db8nu3,db9nu3,0,0,db21nu3]\[0;1;0;0;0];
%%% Compute bCor2 and bCor3 for mu14-15 (i.e. mu14x=1 at (0,1), etc)
    db14nu1=sum(nu1.*(JkIT*[muxVal(4,14);muyVal(4,14)]));
    db14nu2=sum(nu2.*(JkIT*[muxVal(5,14);muyVal(5,14)]));
    db14nu3=sum(nu3.*(JkIT*[muxVal(6,14);muyVal(6,14)]));
    db15nu1=sum(nu1.*(JkIT*[muxVal(4,15);muyVal(4,15)]));
    db15nu2=sum(nu2.*(JkIT*[muxVal(5,15);muyVal(5,15)]));
    db15nu3=sum(nu3.*(JkIT*[muxVal(6,15);muyVal(6,15)]));
    db14x=JkIT(1,:)*[muxVal(3,14);muyVal(3,14)];
    db14y=JkIT(2,:)*[muxVal(3,14);muyVal(3,14)];
    db15x=JkIT(1,:)*[muxVal(3,15);muyVal(3,15)];
    db15y=JkIT(2,:)*[muxVal(3,15);muyVal(3,15)];
    bCor2(k,3,:)=[db14x,db15x,0,0,0; db14y,db15y,0,0,0; db14nu1,db15nu1,db19nu1,0,0;
        db14nu2,db15nu2,0,db20nu2,0; db14nu3,db15nu3,0,0,db21nu3]\[1;0;0;0;0];
    bCor3(k,3,:)=[db14x,db15x,0,0,0; db14y,db15y,0,0,0; db14nu1,db15nu1,db19nu1,0,0;
        db14nu2,db15nu2,0,db20nu2,0; db14nu3,db15nu3,0,0,db21nu3]\[0;1;0;0;0];
%%% Compute bCor4, bCor5, and bCor6 for mu4-6 (i.e. mu4xx=1 at (0,0), etc)
    db4nu1=sum(nu1.*(JkIT*[muxVal(4,4);muyVal(4,4)]));
    db4nu2=sum(nu2.*(JkIT*[muxVal(5,4);muyVal(5,4)]));
    db4nu3=sum(nu3.*(JkIT*[muxVal(6,4);muyVal(6,4)]));
    db5nu1=sum(nu1.*(JkIT*[muxVal(4,5);muyVal(4,5)]));
    db5nu2=sum(nu2.*(JkIT*[muxVal(5,5);muyVal(5,5)]));
    db5nu3=sum(nu3.*(JkIT*[muxVal(6,5);muyVal(6,5)]));
    db6nu1=sum(nu1.*(JkIT*[muxVal(4,6);muyVal(4,6)]));
    db6nu2=sum(nu2.*(JkIT*[muxVal(5,6);muyVal(5,6)]));
    db6nu3=sum(nu3.*(JkIT*[muxVal(6,6);muyVal(6,6)]));

    db4xx=Ak(1,:)*[muxxVal(1,4);muxyVal(1,4);muyyVal(1,4)];
    db4xy=Ak(2,:)*[muxxVal(1,4);muxyVal(1,4);muyyVal(1,4)];
    db4yy=Ak(3,:)*[muxxVal(1,4);muxyVal(1,4);muyyVal(1,4)];
    db5xx=Ak(1,:)*[muxxVal(1,5);muxyVal(1,5);muyyVal(1,5)];
    db5xy=Ak(2,:)*[muxxVal(1,5);muxyVal(1,5);muyyVal(1,5)];
    db5yy=Ak(3,:)*[muxxVal(1,5);muxyVal(1,5);muyyVal(1,5)];
    db6xx=Ak(1,:)*[muxxVal(1,6);muxyVal(1,6);muyyVal(1,6)];
    db6xy=Ak(2,:)*[muxxVal(1,6);muxyVal(1,6);muyyVal(1,6)];
    db6yy=Ak(3,:)*[muxxVal(1,6);muxyVal(1,6);muyyVal(1,6)];

    bCor4(k,1,:)=[db4xx,db5xx,db6xx,0,0,0; db4xy,db5xy,db6xy,0,0,0; db4yy,db5yy,db6yy,0,0,0; ...
        db4nu1,db5nu1,db6nu1,db19nu1,0,0;
        db4nu2,db5nu2,db6nu2,0,db20nu2,0; db4nu3,db5nu3,db6nu3,0,0,db21nu3]\[1;0;0;0;0;0];
```

```
bCor5(k,1,:)=[db4xx,db5xx,db6xx,0,0,0; db4xy,db5xy,db6xy,0,0,0; db4yy,db5yy,db6yy,0,0,0; ...
    db4nu1,db5nu1,db6nu1,db19nu1,0,0;
    db4nu2,db5nu2,db6nu2,0,db20nu2,0; db4nu3,db5nu3,db6nu3,0,0,db21nu3]\[0;1;0;0;0;0];
bCor6(k,1,:)=[db4xx,db5xx,db6xx,0,0,0; db4xy,db5xy,db6xy,0,0,0; db4yy,db5yy,db6yy,0,0,0; ...
    db4nu1,db5nu1,db6nu1,db19nu1,0,0;
    db4nu2,db5nu2,db6nu2,0,db20nu2,0; db4nu3,db5nu3,db6nu3,0,0,db21nu3]\[0;0;1;0;0;0];

%%% Compute bCor4, bCor5, and bCor6 for mu10-12 (i.e. mu10xx=1 at (1,0), etc)
db10nu1=sum(nu1.*(JkIT*[muxVal(4,10);muyVal(4,10)]));
db10nu2=sum(nu2.*(JkIT*[muxVal(5,10);muyVal(5,10)]));
db10nu3=sum(nu3.*(JkIT*[muxVal(6,10);muyVal(6,10)]));
db11nu1=sum(nu1.*(JkIT*[muxVal(4,11);muyVal(4,11)]));
db11nu2=sum(nu2.*(JkIT*[muxVal(5,11);muyVal(5,11)]));
db11nu3=sum(nu3.*(JkIT*[muxVal(6,11);muyVal(6,11)]));
db12nu1=sum(nu1.*(JkIT*[muxVal(4,12);muyVal(4,12)]));
db12nu2=sum(nu2.*(JkIT*[muxVal(5,12);muyVal(5,12)]));
db12nu3=sum(nu3.*(JkIT*[muxVal(6,12);muyVal(6,12)]));

db10xx=Ak(1,:)*[muxxVal(2,10);muxyVal(2,10);muyyVal(2,10)];
db10xy=Ak(2,:)*[muxxVal(2,10);muxyVal(2,10);muyyVal(2,10)];
db10yy=Ak(3,:)*[muxxVal(2,10);muxyVal(2,10);muyyVal(2,10)];
db11xx=Ak(1,:)*[muxxVal(2,11);muxyVal(2,11);muyyVal(2,11)];
db11xy=Ak(2,:)*[muxxVal(2,11);muxyVal(2,11);muyyVal(2,11)];
db11yy=Ak(3,:)*[muxxVal(2,11);muxyVal(2,11);muyyVal(2,11)];
db12xx=Ak(1,:)*[muxxVal(2,12);muxyVal(2,12);muyyVal(2,12)];
db12xy=Ak(2,:)*[muxxVal(2,12);muxyVal(2,12);muyyVal(2,12)];
db12yy=Ak(3,:)*[muxxVal(2,12);muxyVal(2,12);muyyVal(2,12)];

bCor4(k,2,:)=[db10xx,db11xx,db12xx,0,0,0; db10xy,db11xy,db12xy,0,0,0; db10yy,db11yy,db12yy...
    ,0,0,0; db10nu1,db11nu1,db12nu1,db19nu1,0,0;
    db10nu2,db11nu2,db12nu2,0,db20nu2,0; db10nu3,db11nu3,db12nu3,0,0,db21nu3...
        ]\[1;0;0;0;0;0];
bCor5(k,2,:)=[db10xx,db11xx,db12xx,0,0,0; db10xy,db11xy,db12xy,0,0,0; db10yy,db11yy,db12yy...
    ,0,0,0; db10nu1,db11nu1,db12nu1,db19nu1,0,0;
    db10nu2,db11nu2,db12nu2,0,db20nu2,0; db10nu3,db11nu3,db12nu3,0,0,db21nu3...
        ]\[0;1;0;0;0;0];
bCor6(k,2,:)=[db10xx,db11xx,db12xx,0,0,0; db10xy,db11xy,db12xy,0,0,0; db10yy,db11yy,db12yy...
    ,0,0,0; db10nu1,db11nu1,db12nu1,db19nu1,0,0;
    db10nu2,db11nu2,db12nu2,0,db20nu2,0; db10nu3,db11nu3,db12nu3,0,0,db21nu3...
        ]\[0;0;1;0;0;0];
```

```
%%% Compute bCor4, bCor5, and bCor6 for mu16-18 (i.e. mu16xx=1 at (0,1), etc)
    db16nu1=sum(nu1.*(JkIT*[muxVal(4,16);muyVal(4,16)]));
    db16nu2=sum(nu2.*(JkIT*[muxVal(5,16);muyVal(5,16)]));
    db16nu3=sum(nu3.*(JkIT*[muxVal(6,16);muyVal(6,16)]));
    db17nu1=sum(nu1.*(JkIT*[muxVal(4,17);muyVal(4,17)]));
    db17nu2=sum(nu2.*(JkIT*[muxVal(5,17);muyVal(5,17)]));
    db17nu3=sum(nu3.*(JkIT*[muxVal(6,17);muyVal(6,17)]));
    db18nu1=sum(nu1.*(JkIT*[muxVal(4,18);muyVal(4,18)]));
    db18nu2=sum(nu2.*(JkIT*[muxVal(5,18);muyVal(5,18)]));
    db18nu3=sum(nu3.*(JkIT*[muxVal(6,18);muyVal(6,18)]));

    db16xx=Ak(1,:)*[muxxVal(3,16);muxyVal(3,16);muyyVal(3,16)];
    db16xy=Ak(2,:)*[muxxVal(3,16);muxyVal(3,16);muyyVal(3,16)];
    db16yy=Ak(3,:)*[muxxVal(3,16);muxyVal(3,16);muyyVal(3,16)];
    db17xx=Ak(1,:)*[muxxVal(3,17);muxyVal(3,17);muyyVal(3,17)];
    db17xy=Ak(2,:)*[muxxVal(3,17);muxyVal(3,17);muyyVal(3,17)];
    db17yy=Ak(3,:)*[muxxVal(3,17);muxyVal(3,17);muyyVal(3,17)];
    db18xx=Ak(1,:)*[muxxVal(3,18);muxyVal(3,18);muyyVal(3,18)];
    db18xy=Ak(2,:)*[muxxVal(3,18);muxyVal(3,18);muyyVal(3,18)];
    db18yy=Ak(3,:)*[muxxVal(3,18);muxyVal(3,18);muyyVal(3,18)];

    bCor4(k,3,:)=[db16xx,db17xx,db18xx,0,0,0; db16xy,db17xy,db18xy,0,0,0;
        db16yy,db17yy,db18yy,0,0,0; db16nu1,db17nu1,db18nu1,db19nu1,0,0;
        db16nu2,db17nu2,db18nu2,0,db20nu2,0; db16nu3,db17nu3,db18nu3,0,0,db21nu3...
            ]\[1;0;0;0;0;0];
    bCor5(k,3,:)=[db16xx,db17xx,db18xx,0,0,0; db16xy,db17xy,db18xy,0,0,0;
        db16yy,db17yy,db18yy,0,0,0; db16nu1,db17nu1,db18nu1,db19nu1,0,0;
        db16nu2,db17nu2,db18nu2,0,db20nu2,0; db16nu3,db17nu3,db18nu3,0,0,db21nu3...
            ]\[0;1;0;0;0;0];
    bCor6(k,3,:)=[db16xx,db17xx,db18xx,0,0,0; db16xy,db17xy,db18xy,0,0,0;
        db16yy,db17yy,db18yy,0,0,0; db16nu1,db17nu1,db18nu1,db19nu1,0,0;
        db16nu2,db17nu2,db18nu2,0,db20nu2,0; db16nu3,db17nu3,db18nu3,0,0,db21nu3...
            ]\[0;0;1;0;0;0];

    bCor7(k,1,1)=1/db19nu1;
    bCor7(k,2,2)=1/db20nu2;
    bCor7(k,3,3)=1/db21nu3;
end  %%% Ends loop over triangles
%%% Combine basis correction constants into one structure
bCnst={bCor1,bCor2,bCor3,bCor4,bCor5,bCor6,bCor7};
```

StiffnessA2D.m

```
function K=StiffnessA2D(pMesh,bCnst,muxxVal,muxyVal,muyyVal,qwt)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function creates the stiffness matrix for the free nodes for the
%%% term (Delta mu_1, Delta mu_2)_\Omega
%%% pMesh is the plate mesh data structure
%%% qpt and qwt are the quadrature points and weights respectively.
%%% The stiffness matrix is K.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Initialize Stiffness Block Matrices
Nfvn=length(pMesh.FVNodePtrs(:,1));
Nfmn=length(pMesh.FMNodePtrs(:,1));
Nbxxn=length(pMesh.BxxNodePtrs(:,1));
Nbyyn=length(pMesh.ByyNodePtrs(:,1));
K11=zeros(Nfvn,Nfvn);
K12=zeros(Nfvn,Nfvn);
K13=zeros(Nfvn,Nfvn);
K14=zeros(Nfvn,Nfvn);
K15=zeros(Nfvn,Nfvn);
K16=zeros(Nfvn,Nfvn);
K17=zeros(Nfvn,Nfmn);
K18=zeros(Nfvn,Nbxxn);
K19=zeros(Nfvn,Nbyyn);
K22=zeros(Nfvn,Nfvn);
K23=zeros(Nfvn,Nfvn);
K24=zeros(Nfvn,Nfvn);
K25=zeros(Nfvn,Nfvn);
K26=zeros(Nfvn,Nfvn);
K27=zeros(Nfvn,Nfmn);
K28=zeros(Nfvn,Nbxxn);
K29=zeros(Nfvn,Nbyyn);
K33=zeros(Nfvn,Nfvn);
K34=zeros(Nfvn,Nfvn);
K35=zeros(Nfvn,Nfvn);
K36=zeros(Nfvn,Nfvn);
K37=zeros(Nfvn,Nfmn);
K38=zeros(Nfvn,Nbxxn);
K39=zeros(Nfvn,Nbyyn);
```

```
K44=zeros(Nfvn,Nfvn);
K45=zeros(Nfvn,Nfvn);
K46=zeros(Nfvn,Nfvn);
K47=zeros(Nfvn,Nfmn);
K48=zeros(Nfvn,Nbxxn);
K49=zeros(Nfvn,Nbyyn);
K55=zeros(Nfvn,Nfvn);
K56=zeros(Nfvn,Nfvn);
K57=zeros(Nfvn,Nfmn);
K58=zeros(Nfvn,Nbxxn);
K59=zeros(Nfvn,Nbyyn);
K66=zeros(Nfvn,Nfvn);
K67=zeros(Nfvn,Nfmn);
K68=zeros(Nfvn,Nbxxn);
K69=zeros(Nfvn,Nbyyn);
K77=zeros(Nfmn,Nfmn);
K78=zeros(Nfmn,Nbxxn);
K79=zeros(Nfmn,Nbyyn);
K88=zeros(Nbxxn,Nbxxn);
K89=zeros(Nbxxn,Nbyyn);
K99=zeros(Nbyyn,Nbyyn);
%%% Loop over triangles in the mesh
for k=1:pMesh.nbTriangles6
%%% Get global node numbers for the nodes of Element k
    nodes=pMesh.TRIANGLES6(k,1:6);
%%% (x,y) coordinates of (local) nodes 1, 2, 3 (Vertices of T_k)
    x1=pMesh.POS(nodes(1),1);
    y1=pMesh.POS(nodes(1),2);
    x2=pMesh.POS(nodes(2),1);
    y2=pMesh.POS(nodes(2),2);
    x3=pMesh.POS(nodes(3),1);
    y3=pMesh.POS(nodes(3),2);
%%% Compute transormation J: [x,y] = [x1, y1] + Ju for u in reference
%%% triangle, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1; y2-y1,y3-y1];
    detJk=det(Jk);
    %%% Ak gives the transformed 2nd partial derivatives when applied to
    %%% the vector [muxx; muxy; muyy]
    Ak=(1/detJk)^2*[Jk(2,2)^2, -2*Jk(2,1)*Jk(2,2), Jk(2,1)^2;
                    Jk(1,2)^2, -2*Jk(1,1)*Jk(1,2), Jk(1,1)^2];
    %%% Only want muxx and muyy for Laplacian so middle row is removed
```

```
for n1=1:3 % Loop over 1st vertex nodes
    if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
        gn1=nodes(n1); % Global Node Number
        vfn1=pMesh.NodePtrs(gn1); % 1st Vertex Free Node Number
    %%% Compute Delta mu_1 at quadrature points(stored in a row vector)
        deltaMu1=sum(Ak*[muxxVal(:,1+6*(n1-1))';muxyVal(:,1+6*(n1-1))';muyyVal(:,1+6*(n1...
            -1))']);
        deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
        deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
        deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
        b1=bCnst{1}(k,n1,1);
        b2=bCnst{1}(k,n1,2);
        b3=bCnst{1}(k,n1,3);
        deltaMu_1=deltaMu1+b1*deltaMu19+b2*deltaMu20+b3*deltaMu21;
    %%% Compute Delta mu_2 at quadrature points(stored in a row vector)
        deltaMu2=sum(Ak*[muxxVal(:,2+6*(n1-1))';muxyVal(:,2+6*(n1-1))';muyyVal(:,2+6*(n1...
            -1))']);
        deltaMu3=sum(Ak*[muxxVal(:,3+6*(n1-1))';muxyVal(:,3+6*(n1-1))';muyyVal(:,3+6*(n1...
            -1))']);
        deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
        deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
        deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
        a1=bCnst{2}(k,n1,1);
        a2=bCnst{2}(k,n1,2);
        b1=bCnst{2}(k,n1,3);
        b2=bCnst{2}(k,n1,4);
        b3=bCnst{2}(k,n1,5);
        deltaMu_2=a1*deltaMu2+a2*deltaMu3+b1*deltaMu19+b2*deltaMu20+b3*deltaMu21;
    %%% Compute Delta mu_3 at quadrature points(stored in a row vector)
        deltaMu2=sum(Ak*[muxxVal(:,2+6*(n1-1))';muxyVal(:,2+6*(n1-1))';muyyVal(:,2+6*(n1...
            -1))']);
        deltaMu3=sum(Ak*[muxxVal(:,3+6*(n1-1))';muxyVal(:,3+6*(n1-1))';muyyVal(:,3+6*(n1...
            -1))']);
        deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
        deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
        deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
        a1=bCnst{3}(k,n1,1);
        a2=bCnst{3}(k,n1,2);
        b1=bCnst{3}(k,n1,3);
        b2=bCnst{3}(k,n1,4);
        b3=bCnst{3}(k,n1,5);
```

```
    deltaMu_3=a1*deltaMu2+a2*deltaMu3+b1*deltaMu19+b2*deltaMu20+b3*deltaMu21;
%%% Compute Delta mu_4 at quadrature points(stored in a row vector)
    deltaMu4=sum(Ak*[muxxVal(:,4+6*(n1-1))';muxyVal(:,4+6*(n1-1))';muyyVal(:,4+6*(n1...
        -1))']);
    deltaMu5=sum(Ak*[muxxVal(:,5+6*(n1-1))';muxyVal(:,5+6*(n1-1))';muyyVal(:,5+6*(n1...
        -1))']);
    deltaMu6=sum(Ak*[muxxVal(:,6+6*(n1-1))';muxyVal(:,6+6*(n1-1))';muyyVal(:,6+6*(n1...
        -1))']);
    deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
    deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
    deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
    a1=bCnst{4}(k,n1,1);
    a2=bCnst{4}(k,n1,2);
    a3=bCnst{4}(k,n1,3);
    b1=bCnst{4}(k,n1,4);
    b2=bCnst{4}(k,n1,5);
    b3=bCnst{4}(k,n1,6);
    deltaMu_4=a1*deltaMu4+a2*deltaMu5+a3*deltaMu6+b1*deltaMu19+b2*deltaMu20+b3*...
        deltaMu21;
%%% Compute Delta mu_5 at quadrature points(stored in a row vector)
    deltaMu4=sum(Ak*[muxxVal(:,4+6*(n1-1))';muxyVal(:,4+6*(n1-1))';muyyVal(:,4+6*(n1...
        -1))']);
    deltaMu5=sum(Ak*[muxxVal(:,5+6*(n1-1))';muxyVal(:,5+6*(n1-1))';muyyVal(:,5+6*(n1...
        -1))']);
    deltaMu6=sum(Ak*[muxxVal(:,6+6*(n1-1))';muxyVal(:,6+6*(n1-1))';muyyVal(:,6+6*(n1...
        -1))']);
    deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
    deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
    deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
    a1=bCnst{5}(k,n1,1);
    a2=bCnst{5}(k,n1,2);
    a3=bCnst{5}(k,n1,3);
    b1=bCnst{5}(k,n1,4);
    b2=bCnst{5}(k,n1,5);
    b3=bCnst{5}(k,n1,6);
    deltaMu_5=a1*deltaMu4+a2*deltaMu5+a3*deltaMu6+b1*deltaMu19+b2*deltaMu20+b3*...
        deltaMu21;
%%% Compute Delta mu_6 at quadrature points(stored in a row vector)
    deltaMu4=sum(Ak*[muxxVal(:,4+6*(n1-1))';muxyVal(:,4+6*(n1-1))';muyyVal(:,4+6*(n1...
        -1))']);
    deltaMu5=sum(Ak*[muxxVal(:,5+6*(n1-1))';muxyVal(:,5+6*(n1-1))';muyyVal(:,5+6*(n1...
```

```
    -1)) ']);
deltaMu6=sum(Ak*[muxxVal(:,6+6*(n1-1)) ';muxyVal(:,6+6*(n1-1)) ';muyyVal(:,6+6*(n1...
    -1)) ']);
deltaMu19=sum(Ak*[muxxVal(:,19) ';muxyVal(:,19) ';muyyVal(:,19) ']);
deltaMu20=sum(Ak*[muxxVal(:,20) ';muxyVal(:,20) ';muyyVal(:,20) ']);
deltaMu21=sum(Ak*[muxxVal(:,21) ';muxyVal(:,21) ';muyyVal(:,21) ']);
a1=bCnst{6}(k,n1,1);
a2=bCnst{6}(k,n1,2);
a3=bCnst{6}(k,n1,3);
b1=bCnst{6}(k,n1,4);
b2=bCnst{6}(k,n1,5);
b3=bCnst{6}(k,n1,6);
deltaMu_6=a1*deltaMu4+a2*deltaMu5+a3*deltaMu6+b1*deltaMu19+b2*deltaMu20+b3*...
    deltaMu21;

for n2=1:3 % Loop over 2nd vertex nodes
    if pMesh.POS(nodes(n2),4)==-10 % Interior Plate Node
        gn2=nodes(n2); % Global Node Number
        vfn2=pMesh.NodePtrs(gn2); % 2nd Vertex Free Node Number
    %%% Compute Delta Mu_1 at quadrature points(stored in a row vector)
        DeltaMu1=sum(Ak*[muxxVal(:,1+6*(n2-1)) ';muxyVal(:,1+6*(n2-1)) ';muyyVal...
            (:,1+6*(n2-1)) ']);
        DeltaMu19=sum(Ak*[muxxVal(:,19) ';muxyVal(:,19) ';muyyVal(:,19) ']);
        DeltaMu20=sum(Ak*[muxxVal(:,20) ';muxyVal(:,20) ';muyyVal(:,20) ']);
        DeltaMu21=sum(Ak*[muxxVal(:,21) ';muxyVal(:,21) ';muyyVal(:,21) ']);
        B1=bCnst{1}(k,n2,1);
        B2=bCnst{1}(k,n2,2);
        B3=bCnst{1}(k,n2,3);
        DeltaMu_1=DeltaMu1+B1*DeltaMu19+B2*DeltaMu20+B3*DeltaMu21;
    %%% Compute Delta Mu_2 at quadrature points(stored in a row vector)
        DeltaMu2=sum(Ak*[muxxVal(:,2+6*(n2-1)) ';muxyVal(:,2+6*(n2-1)) ';muyyVal...
            (:,2+6*(n2-1)) ']);
        DeltaMu3=sum(Ak*[muxxVal(:,3+6*(n2-1)) ';muxyVal(:,3+6*(n2-1)) ';muyyVal...
            (:,3+6*(n2-1)) ']);
        DeltaMu19=sum(Ak*[muxxVal(:,19) ';muxyVal(:,19) ';muyyVal(:,19) ']);
        DeltaMu20=sum(Ak*[muxxVal(:,20) ';muxyVal(:,20) ';muyyVal(:,20) ']);
        DeltaMu21=sum(Ak*[muxxVal(:,21) ';muxyVal(:,21) ';muyyVal(:,21) ']);
        A1=bCnst{2}(k,n2,1);
        A2=bCnst{2}(k,n2,2);
        B1=bCnst{2}(k,n2,3);
        B2=bCnst{2}(k,n2,4);
```

```
    B3=bCnst{2}(k,n2,5);
    DeltaMu_2=A1*DeltaMu2+A2*DeltaMu3+B1*DeltaMu19+B2*DeltaMu20+B3*DeltaMu21;
%%% Compute Delta Mu_3 at quadrature points(stored in a row vector)
    DeltaMu2=sum(Ak*[muxxVal(:,2+6*(n2-1))';muxyVal(:,2+6*(n2-1))';muyyVal...
        (:,2+6*(n2-1))']);
    DeltaMu3=sum(Ak*[muxxVal(:,3+6*(n2-1))';muxyVal(:,3+6*(n2-1))';muyyVal...
        (:,3+6*(n2-1))']);
    DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
    DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
    DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
    A1=bCnst{3}(k,n2,1);
    A2=bCnst{3}(k,n2,2);
    B1=bCnst{3}(k,n2,3);
    B2=bCnst{3}(k,n2,4);
    B3=bCnst{3}(k,n2,5);
    DeltaMu_3=A1*DeltaMu2+A2*DeltaMu3+B1*DeltaMu19+B2*DeltaMu20+B3*DeltaMu21;
%%% Compute Delta Mu_4 at quadrature points(stored in a row vector)
    DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
        (:,4+6*(n2-1))']);
    DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
        (:,5+6*(n2-1))']);
    DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
        (:,6+6*(n2-1))']);
    DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
    DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
    DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
    A1=bCnst{4}(k,n2,1);
    A2=bCnst{4}(k,n2,2);
    A3=bCnst{4}(k,n2,3);
    B1=bCnst{4}(k,n2,4);
    B2=bCnst{4}(k,n2,5);
    B3=bCnst{4}(k,n2,6);
    DeltaMu_4=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
        *DeltaMu21;
%%% Compute Delta Mu_5 at quadrature points(stored in a row vector)
    DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
        (:,4+6*(n2-1))']);
    DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
        (:,5+6*(n2-1))']);
    DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
        (:,6+6*(n2-1))']);
```

```
DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
A1=bCnst{5}(k,n2,1);
A2=bCnst{5}(k,n2,2);
A3=bCnst{5}(k,n2,3);
B1=bCnst{5}(k,n2,4);
B2=bCnst{5}(k,n2,5);
B3=bCnst{5}(k,n2,6);
DeltaMu_5=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
    *DeltaMu21;
%%% Compute Delta Mu_6 at quadrature points(stored in a row vector)
DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
    (:,4+6*(n2-1))']);
DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
    (:,5+6*(n2-1))']);
DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
    (:,6+6*(n2-1))']);
DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
A1=bCnst{6}(k,n2,1);
A2=bCnst{6}(k,n2,2);
A3=bCnst{6}(k,n2,3);
B1=bCnst{6}(k,n2,4);
B2=bCnst{6}(k,n2,5);
B3=bCnst{6}(k,n2,6);
DeltaMu_6=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
    *DeltaMu21;


%%% Int_Tk (Delta mu_1)*(Delta Mu_1) Using Quadrature
qVal=detJk*((deltaMu_1.*DeltaMu_1)*qwt);
K11(vfn1,vfn2)=K11(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_1)*(Delta Mu_2) Using Quadrature
qVal=detJk*((deltaMu_1.*DeltaMu_2)*qwt);
K12(vfn1,vfn2)=K12(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_1)*(Delta Mu_3) Using Quadrature
qVal=detJk*((deltaMu_1.*DeltaMu_3)*qwt);
K13(vfn1,vfn2)=K13(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_1)*(Delta Mu_4) Using Quadrature
qVal=detJk*((deltaMu_1.*DeltaMu_4)*qwt);
```

```
        K14(vfn1,vfn2)=K14(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_1)*(Delta Mu_5) Using Quadrature
        qVal=detJk*((deltaMu_1.*DeltaMu_5)*qwt);
        K15(vfn1,vfn2)=K15(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_1)*(Delta Mu_6) Using Quadrature
        qVal=detJk*((deltaMu_1.*DeltaMu_6)*qwt);
        K16(vfn1,vfn2)=K16(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_2)*(Delta Mu_2) Using Quadrature
        qVal=detJk*((deltaMu_2.*DeltaMu_2)*qwt);
        K22(vfn1,vfn2)=K22(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_2)*(Delta Mu_3) Using Quadrature
        qVal=detJk*((deltaMu_2.*DeltaMu_3)*qwt);
        K23(vfn1,vfn2)=K23(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_2)*(Delta Mu_4) Using Quadrature
        qVal=detJk*((deltaMu_2.*DeltaMu_4)*qwt);
        K24(vfn1,vfn2)=K24(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_2)*(Delta Mu_5) Using Quadrature
        qVal=detJk*((deltaMu_2.*DeltaMu_5)*qwt);
        K25(vfn1,vfn2)=K25(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_2)*(Delta Mu_6) Using Quadrature
        qVal=detJk*((deltaMu_2.*DeltaMu_6)*qwt);
        K26(vfn1,vfn2)=K26(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_3)*(Delta Mu_3) Using Quadrature
        qVal=detJk*((deltaMu_3.*DeltaMu_3)*qwt);
        K33(vfn1,vfn2)=K33(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_3)*(Delta Mu_4) Using Quadrature
        qVal=detJk*((deltaMu_3.*DeltaMu_4)*qwt);
        K34(vfn1,vfn2)=K34(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_3)*(Delta Mu_5) Using Quadrature
        qVal=detJk*((deltaMu_3.*DeltaMu_5)*qwt);
        K35(vfn1,vfn2)=K35(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_3)*(Delta Mu_6) Using Quadrature
        qVal=detJk*((deltaMu_3.*DeltaMu_6)*qwt);
        K36(vfn1,vfn2)=K36(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_4)*(Delta Mu_4) Using Quadrature
        qVal=detJk*((deltaMu_4.*DeltaMu_4)*qwt);
        K44(vfn1,vfn2)=K44(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_4)*(Delta Mu_5) Using Quadrature
        qVal=detJk*((deltaMu_4.*DeltaMu_5)*qwt);
        K45(vfn1,vfn2)=K45(vfn1,vfn2)+qVal;
%%% Int_Tk (Delta mu_4)*(Delta Mu_6) Using Quadrature
```

```
        qVal=detJk*((deltaMu_4.*DeltaMu_6)*qwt);
        K46(vfn1,vfn2)=K46(vfn1,vfn2)+qVal;
    %%% Int_Tk (Delta mu_5)*(Delta Mu_5) Using Quadrature
        qVal=detJk*((deltaMu_5.*DeltaMu_5)*qwt);
        K55(vfn1,vfn2)=K55(vfn1,vfn2)+qVal;
    %%% Int_Tk (Delta mu_5)*(Delta Mu_6) Using Quadrature
        qVal=detJk*((deltaMu_5.*DeltaMu_6)*qwt);
        K56(vfn1,vfn2)=K56(vfn1,vfn2)+qVal;
    %%% Int_Tk (Delta mu_6)*(Delta Mu_6) Using Quadrature
        qVal=detJk*((deltaMu_6.*DeltaMu_6)*qwt);
        K66(vfn1,vfn2)=K66(vfn1,vfn2)+qVal;

    elseif pMesh.NodePtrs(nodes(n2),3)>0 % Boundary d^2/dx^2 Node
        gn2=nodes(n2); % Global Node Number
        vbxxn2=pMesh.NodePtrs(gn2,3); % Boundary d^2/dx^2 Node Number
        %%% Compute Delta Mu_8 at quadrature points(stored in a row vector)
        DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
            (:,4+6*(n2-1))']);
        DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
            (:,5+6*(n2-1))']);
        DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
            (:,6+6*(n2-1))']);
        DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
        DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
        DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
        A1=bCnst{4}(k,n2,1);
        A2=bCnst{4}(k,n2,2);
        A3=bCnst{4}(k,n2,3);
        B1=bCnst{4}(k,n2,4);
        B2=bCnst{4}(k,n2,5);
        B3=bCnst{4}(k,n2,6);
        DeltaMu_8=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
            *DeltaMu21;

    %%% Int_Tk (Delta mu_1)*(Delta Mu_8) Using Quadrature
        qVal=detJk*((deltaMu_1.*DeltaMu_8)*qwt);
        K18(vfn1,vbxxn2)=K18(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (Delta mu_2)*(Delta Mu_8) Using Quadrature
        qVal=detJk*((deltaMu_2.*DeltaMu_8)*qwt);
        K28(vfn1,vbxxn2)=K28(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (Delta mu_3)*(Delta Mu_8) Using Quadrature
```

```
        qVal=detJk*((deltaMu_3.*DeltaMu_8)*qwt);
        K38(vfn1,vbxxn2)=K38(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (Delta mu_4)*(Delta Mu_8) Using Quadrature
        qVal=detJk*((deltaMu_4.*DeltaMu_8)*qwt);
        K48(vfn1,vbxxn2)=K48(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (Delta mu_5)*(Delta Mu_8) Using Quadrature
        qVal=detJk*((deltaMu_5.*DeltaMu_8)*qwt);
        K58(vfn1,vbxxn2)=K58(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (Delta mu_6)*(Delta Mu_8) Using Quadrature
        qVal=detJk*((deltaMu_6.*DeltaMu_8)*qwt);
        K68(vfn1,vbxxn2)=K68(vfn1,vbxxn2)+qVal;

    elseif pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
        gn2=nodes(n2); % Global Node Number
        vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
        %%% Compute Delta Mu_9 at quadrature points(stored in a row vector)
        DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
            (:,4+6*(n2-1))']);
        DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
            (:,5+6*(n2-1))']);
        DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
            (:,6+6*(n2-1))']);
        DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
        DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
        DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
        A1=bCnst{6}(k,n2,1);
        A2=bCnst{6}(k,n2,2);
        A3=bCnst{6}(k,n2,3);
        B1=bCnst{6}(k,n2,4);
        B2=bCnst{6}(k,n2,5);
        B3=bCnst{6}(k,n2,6);
        DeltaMu_9=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
            *DeltaMu21;

    %%% Int_Tk (Delta mu_1)*(Delta Mu_9) Using Quadrature
        qVal=detJk*((deltaMu_1.*DeltaMu_9)*qwt);
        K19(vfn1,vbyyn2)=K19(vfn1,vbyyn2)+qVal;
    %%% Int_Tk (Delta mu_2)*(Delta Mu_9) Using Quadrature
        qVal=detJk*((deltaMu_2.*DeltaMu_9)*qwt);
        K29(vfn1,vbyyn2)=K29(vfn1,vbyyn2)+qVal;
    %%% Int_Tk (Delta mu_3)*(Delta Mu_9) Using Quadrature
```

```
            qVal=detJk*((deltaMu_3.*DeltaMu_9)*qwt);
            K39(vfn1,vbyyn2)=K39(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (Delta mu_4)*(Delta Mu_9) Using Quadrature
            qVal=detJk*((deltaMu_4.*DeltaMu_9)*qwt);
            K49(vfn1,vbyyn2)=K49(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (Delta mu_5)*(Delta Mu_9) Using Quadrature
            qVal=detJk*((deltaMu_5.*DeltaMu_9)*qwt);
            K59(vfn1,vbyyn2)=K59(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (Delta mu_6)*(Delta Mu_9) Using Quadrature
            qVal=detJk*((deltaMu_6.*DeltaMu_9)*qwt);
            K69(vfn1,vbyyn2)=K69(vfn1,vbyyn2)+qVal;
        end %end if loop for vertex 2
    end %end loop over vertex 2

    for n2=4:6 % Loop over midpoint nodes
        if pMesh.POS(nodes(n2),4)==-10 % Interior Plate Node
            gn2=nodes(n2); % Global Node Number
            mfn2=pMesh.NodePtrs(gn2); % Midpoint Free Node Number
            %%% Compute Delta Mu_7 at quadrature points(stored in a row vector)
            %%% Note that only one of B1-B3 should be non-zero
            DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
            DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
            DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
            B1=bCnst{7}(k,n2-3,1); % Note n2-3 to get 1,2,3 from 4,5,6
            B2=bCnst{7}(k,n2-3,2);
            B3=bCnst{7}(k,n2-3,3);
            DeltaMu_7=B1*DeltaMu19+B2*DeltaMu20+B3*DeltaMu21;

        %%% Int_Tk (Delta mu_1)*(Delta Mu_7) Using Quadrature
            qVal=detJk*((deltaMu_1.*DeltaMu_7)*qwt);
            K17(vfn1,mfn2)=K17(vfn1,mfn2)+qVal;
        %%% Int_Tk (Delta mu_2)*(Delta Mu_7) Using Quadrature
            qVal=detJk*((deltaMu_2.*DeltaMu_7)*qwt);
            K27(vfn1,mfn2)=K27(vfn1,mfn2)+qVal;
        %%% Int_Tk (Delta mu_3)*(Delta Mu_7) Using Quadrature
            qVal=detJk*((deltaMu_3.*DeltaMu_7)*qwt);
            K37(vfn1,mfn2)=K37(vfn1,mfn2)+qVal;
        %%% Int_Tk (Delta mu_4)*(Delta Mu_7) Using Quadrature
            qVal=detJk*((deltaMu_4.*DeltaMu_7)*qwt);
            K47(vfn1,mfn2)=K47(vfn1,mfn2)+qVal;
        %%% Int_Tk (Delta mu_5)*(Delta Mu_7) Using Quadrature
```

```
                qVal=detJk*((deltaMu_5.*DeltaMu_7)*qwt);
                K57(vfn1,mfn2)=K57(vfn1,mfn2)+qVal;
            %%% Int_Tk (Delta mu_6)*(Delta Mu_7) Using Quadrature
                qVal=detJk*((deltaMu_6.*DeltaMu_7)*qwt);
                K67(vfn1,mfn2)=K67(vfn1,mfn2)+qVal;
            end % end if loop for midpoint node 2
        end % end loop over midpoint node 2

    elseif pMesh.NodePtrs(nodes(n1),3)>0 % Boundary d^2/dx^2 Node
        gn1=nodes(n1); % Global Node Number
        vbxxn1=pMesh.NodePtrs(gn1,3); % Boundary d^2/dx^2 Node Number
        %%% Compute Delta mu_8 at quadrature points(stored in a row vector)
        deltaMu4=sum(Ak*[muxxVal(:,4+6*(n1-1))';muxyVal(:,4+6*(n1-1))';muyyVal(:,4+6*(n1...
            -1))']);
        deltaMu5=sum(Ak*[muxxVal(:,5+6*(n1-1))';muxyVal(:,5+6*(n1-1))';muyyVal(:,5+6*(n1...
            -1))']);
        deltaMu6=sum(Ak*[muxxVal(:,6+6*(n1-1))';muxyVal(:,6+6*(n1-1))';muyyVal(:,6+6*(n1...
            -1))']);
        deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
        deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
        deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
        a1=bCnst{4}(k,n1,1);
        a2=bCnst{4}(k,n1,2);
        a3=bCnst{4}(k,n1,3);
        b1=bCnst{4}(k,n1,4);
        b2=bCnst{4}(k,n1,5);
        b3=bCnst{4}(k,n1,6);
        deltaMu_8=a1*deltaMu4+a2*deltaMu5+a3*deltaMu6+b1*deltaMu19+b2*deltaMu20+b3*...
            deltaMu21;

        for n2=1:3 % Loop over 2nd vertex nodes
            if pMesh.NodePtrs(nodes(n2),3)>0 % Boundary d^2/dx^2 Node
                gn2=nodes(n2); % Global Node Number
                vbxxn2=pMesh.NodePtrs(gn2,3); % Boundary d^2/dx^2 Node Number
            %%% Compute Delta Mu_8 at quadrature points(stored in a row vector)
                DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
                    (:,4+6*(n2-1))']);
                DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
                    (:,5+6*(n2-1))']);
                DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
                    (:,6+6*(n2-1))']);
```

```
            DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
            DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
            DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
            A1=bCnst{4}(k,n2,1);
            A2=bCnst{4}(k,n2,2);
            A3=bCnst{4}(k,n2,3);
            B1=bCnst{4}(k,n2,4);
            B2=bCnst{4}(k,n2,5);
            B3=bCnst{4}(k,n2,6);
            DeltaMu_8=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
                *DeltaMu21;
        %%% Int_Tk (Delta mu_8)*(Delta Mu_8) Using Quadrature
            qVal=detJk*((deltaMu_8.*DeltaMu_8)*qwt);
            K88(vbxxn1,vbxxn2)=K88(vbxxn1,vbxxn2)+qVal;

        elseif pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
            gn2=nodes(n2); % Global Node Number
            vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
        %%% Compute Delta Mu_9 at quadrature points(stored in a row vector)
            DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
                (:,4+6*(n2-1))']);
            DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
                (:,5+6*(n2-1))']);
            DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
                (:,6+6*(n2-1))']);
            DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
            DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
            DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
            A1=bCnst{6}(k,n2,1);
            A2=bCnst{6}(k,n2,2);
            A3=bCnst{6}(k,n2,3);
            B1=bCnst{6}(k,n2,4);
            B2=bCnst{6}(k,n2,5);
            B3=bCnst{6}(k,n2,6);
            DeltaMu_9=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
                *DeltaMu21;
        %%% Int_Tk (Delta mu_8)*(Delta Mu_9) Using Quadrature
            qVal=detJk*((deltaMu_8.*DeltaMu_9)*qwt);
            K89(vbxxn1,vbyyn2)=K89(vbxxn1,vbyyn2)+qVal;
        end % ends if loop over vertex 2
    end % ends loop over vertex 2
```

```
elseif pMesh.NodePtrs(nodes(n1),3)<0 % Boundary d^2/dy^2 Node
    gn1=nodes(n1); % Global Node Number
    vbyyn1=-pMesh.NodePtrs(gn1,3); % Boundary d^2/dy^2 Node Number
%%% Compute Delta mu_9 at quadrature points(stored in a row vector)
    deltaMu4=sum(Ak*[muxxVal(:,4+6*(n1-1))';muxyVal(:,4+6*(n1-1))';muyyVal(:,4+6*(n1...
        -1))']);
    deltaMu5=sum(Ak*[muxxVal(:,5+6*(n1-1))';muxyVal(:,5+6*(n1-1))';muyyVal(:,5+6*(n1...
        -1))']);
    deltaMu6=sum(Ak*[muxxVal(:,6+6*(n1-1))';muxyVal(:,6+6*(n1-1))';muyyVal(:,6+6*(n1...
        -1))']);
    deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
    deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
    deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
    a1=bCnst{6}(k,n1,1);
    a2=bCnst{6}(k,n1,2);
    a3=bCnst{6}(k,n1,3);
    b1=bCnst{6}(k,n1,4);
    b2=bCnst{6}(k,n1,5);
    b3=bCnst{6}(k,n1,6);
    deltaMu_9=a1*deltaMu4+a2*deltaMu5+a3*deltaMu6+b1*deltaMu19+b2*deltaMu20+b3*...
        deltaMu21;

    for n2=1:3 % Loop over 2nd vertex nodes
        if pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
            gn2=nodes(n2); % Global Node Number
            vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
        %%% Compute Delta Mu_9 at quadrature points(stored in a row vector)
            DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
                (:,4+6*(n2-1))']);
            DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
                (:,5+6*(n2-1))']);
            DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
                (:,6+6*(n2-1))']);
            DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
            DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
            DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
            A1=bCnst{6}(k,n2,1);
            A2=bCnst{6}(k,n2,2);
            A3=bCnst{6}(k,n2,3);
            B1=bCnst{6}(k,n2,4);
```

```
                B2=bCnst{6}(k,n2,5);
                B3=bCnst{6}(k,n2,6);
                DeltaMu_9=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
                    *DeltaMu21;
            %%% Int_Tk (Delta mu_9)*(Delta Mu_9) Using Quadrature
                qVal=detJk*((deltaMu_9.*DeltaMu_9)*qwt);
                K99(vbyyn1,vbyyn2)=K99(vbyyn1,vbyyn2)+qVal;
            end
        end
    end %end if loop for vertex 1
end %end loop over vertex 1

for n1=4:6 % Loop over midpoint nodes
    if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
        gn1=nodes(n1); % Global Node Number
        mfn1=pMesh.NodePtrs(gn1); % 1st Midpoint Free Node Number
    %%% Compute Delta mu_7 at quadrature points(stored in a row vector)
    %%% Note that only one of b1-b3 should be non-zero
        deltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
        deltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
        deltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
        b1=bCnst{7}(k,n1-3,1); % Note n1-3 to get 1,2,3 from 4,5,6
        b2=bCnst{7}(k,n1-3,2);
        b3=bCnst{7}(k,n1-3,3);
        deltaMu_7=b1*deltaMu19+b2*deltaMu20+b3*deltaMu21;

        for n2=4:6 % Loop over midpoint nodes
            if pMesh.POS(nodes(n2),4)==-10 % Interior Plate Node
                gn2=nodes(n2); % Global Node Number
                mfn2=pMesh.NodePtrs(gn2); % 2nd Midpoint Free Node Number
            %%% Compute Delta Mu_7 at quadrature points(stored in a row vector)
            %%% Note that only one of B1-B3 should be non-zero
                DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
                DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
                DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
                B1=bCnst{7}(k,n2-3,1); % Note n2-3 to get 1,2,3 from 4,5,6
                B2=bCnst{7}(k,n2-3,2);
                B3=bCnst{7}(k,n2-3,3);
                DeltaMu_7=B1*DeltaMu19+B2*DeltaMu20+B3*DeltaMu21;

            %%% Int_Tk (Delta mu_7)*(Delta Mu_7) Using Quadrature
```

```
            qVal=detJk*((deltaMu_7.*DeltaMu_7)*qwt);
            K77(mfn1,mfn2)=K77(mfn1,mfn2)+qVal;
        end
    end

    for n2=1:3 % Loop over 2nd vertex nodes
        if pMesh.NodePtrs(nodes(n2),3)>0 % Boundary d^2/dx^2 Node
            gn2=nodes(n2); % Global Node Number
            vbxxn2=pMesh.NodePtrs(gn2,3); % Boundary d^2/dx^2 Node Number
        %%% Compute Delta Mu_8 at quadrature points(stored in a row vector)
            DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
                (:,4+6*(n2-1))']);
            DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
                (:,5+6*(n2-1))']);
            DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
                (:,6+6*(n2-1))']);
            DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
            DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
            DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
            A1=bCnst{4}(k,n2,1);
            A2=bCnst{4}(k,n2,2);
            A3=bCnst{4}(k,n2,3);
            B1=bCnst{4}(k,n2,4);
            B2=bCnst{4}(k,n2,5);
            B3=bCnst{4}(k,n2,6);
            DeltaMu_8=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
                *DeltaMu21;
        %%% Int_Tk (Delta mu_7)*(Delta Mu_8) Using Quadrature
            qVal=detJk*((deltaMu_7.*DeltaMu_8)*qwt);
            K78(mfn1,vbxxn2)=K78(mfn1,vbxxn2)+qVal;
        elseif pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
            gn2=nodes(n2); % Global Node Number
            vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
        %%% Compute Delta Mu_9 at quadrature points(stored in a row vector)
            DeltaMu4=sum(Ak*[muxxVal(:,4+6*(n2-1))';muxyVal(:,4+6*(n2-1))';muyyVal...
                (:,4+6*(n2-1))']);
            DeltaMu5=sum(Ak*[muxxVal(:,5+6*(n2-1))';muxyVal(:,5+6*(n2-1))';muyyVal...
                (:,5+6*(n2-1))']);
            DeltaMu6=sum(Ak*[muxxVal(:,6+6*(n2-1))';muxyVal(:,6+6*(n2-1))';muyyVal...
                (:,6+6*(n2-1))']);
            DeltaMu19=sum(Ak*[muxxVal(:,19)';muxyVal(:,19)';muyyVal(:,19)']);
```

```
                    DeltaMu20=sum(Ak*[muxxVal(:,20)';muxyVal(:,20)';muyyVal(:,20)']);
                    DeltaMu21=sum(Ak*[muxxVal(:,21)';muxyVal(:,21)';muyyVal(:,21)']);
                    A1=bCnst{6}(k,n2,1);
                    A2=bCnst{6}(k,n2,2);
                    A3=bCnst{6}(k,n2,3);
                    B1=bCnst{6}(k,n2,4);
                    B2=bCnst{6}(k,n2,5);
                    B3=bCnst{6}(k,n2,6);
                    DeltaMu_9=A1*DeltaMu4+A2*DeltaMu5+A3*DeltaMu6+B1*DeltaMu19+B2*DeltaMu20+B3...
                        *DeltaMu21;
                %%% Int_Tk (Delta mu_7)*(Delta Mu_9) Using Quadrature
                    qVal=detJk*((deltaMu_7.*DeltaMu_9)*qwt);
                    K79(mfn1,vbyyn2)=K79(mfn1,vbyyn2)+qVal;
                end
            end
        end
    end
end %end loop over triangles
K21=transpose(K12);  % Copy to Lower Triangular blocks
K31=transpose(K13);
K32=transpose(K23);
K41=transpose(K14);
K42=transpose(K24);
K43=transpose(K34);
K51=transpose(K15);
K52=transpose(K25);
K53=transpose(K35);
K54=transpose(K45);
K61=transpose(K16);
K62=transpose(K26);
K63=transpose(K36);
K64=transpose(K46);
K65=transpose(K56);
K71=transpose(K17);
K72=transpose(K27);
K73=transpose(K37);
K74=transpose(K47);
K75=transpose(K57);
K76=transpose(K67);
K81=transpose(K18);
K82=transpose(K28);
```

```
K83=transpose(K38);
K84=transpose(K48);
K85=transpose(K58);
K86=transpose(K68);
K87=transpose(K78);
K91=transpose(K19);
K92=transpose(K29);
K93=transpose(K39);
K94=transpose(K49);
K95=transpose(K59);
K96=transpose(K69);
K97=transpose(K79);
K98=transpose(K89);
%%% Assemble the Stiffness Matrix K
K=[K11,K12,K13,K14,K15,K16,K17,K18,K19; K21,K22,K23,K24,K25,K26,K27,K28,K29;
   K31,K32,K33,K34,K35,K36,K37,K38,K39; K41,K42,K43,K44,K45,K46,K47,K48,K49;
   K51,K52,K53,K54,K55,K56,K57,K58,K59; K61,K62,K63,K64,K65,K66,K67,K68,K69;
   K71,K72,K73,K74,K75,K76,K77,K78,K79; K81,K82,K83,K84,K85,K86,K87,K88,K89;
   K91,K92,K93,K94,K95,K96,K97,K98,K99];
K=sparse(K);
```

StiffnessAL2D.m

```
function K=StiffnessAL2D(pMesh,bCnst,muVal,qwt,lambda)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function creates the part of the plate bilinear form that
%%% corresponds to the term: lambda^2(mu1, mu2)_\Omega
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Initialize Stiffness Block Matrices
Nfvn=length(pMesh.FVNodePtrs(:,1));
Nfmn=length(pMesh.FMNodePtrs(:,1));
Nbxxn=length(pMesh.BxxNodePtrs(:,1));
Nbyyn=length(pMesh.ByyNodePtrs(:,1));
K11=zeros(Nfvn,Nfvn);
K12=zeros(Nfvn,Nfvn);
K13=zeros(Nfvn,Nfvn);
K14=zeros(Nfvn,Nfvn);
K15=zeros(Nfvn,Nfvn);
K16=zeros(Nfvn,Nfvn);
```

```
K17=zeros(Nfvn,Nfmn);
K18=zeros(Nfvn,Nbxxn);
K19=zeros(Nfvn,Nbyyn);
K22=zeros(Nfvn,Nfvn);
K23=zeros(Nfvn,Nfvn);
K24=zeros(Nfvn,Nfvn);
K25=zeros(Nfvn,Nfvn);
K26=zeros(Nfvn,Nfvn);
K27=zeros(Nfvn,Nfmn);
K28=zeros(Nfvn,Nbxxn);
K29=zeros(Nfvn,Nbyyn);
K33=zeros(Nfvn,Nfvn);
K34=zeros(Nfvn,Nfvn);
K35=zeros(Nfvn,Nfvn);
K36=zeros(Nfvn,Nfvn);
K37=zeros(Nfvn,Nfmn);
K38=zeros(Nfvn,Nbxxn);
K39=zeros(Nfvn,Nbyyn);
K44=zeros(Nfvn,Nfvn);
K45=zeros(Nfvn,Nfvn);
K46=zeros(Nfvn,Nfvn);
K47=zeros(Nfvn,Nfmn);
K48=zeros(Nfvn,Nbxxn);
K49=zeros(Nfvn,Nbyyn);
K55=zeros(Nfvn,Nfvn);
K56=zeros(Nfvn,Nfvn);
K57=zeros(Nfvn,Nfmn);
K58=zeros(Nfvn,Nbxxn);
K59=zeros(Nfvn,Nbyyn);
K66=zeros(Nfvn,Nfvn);
K67=zeros(Nfvn,Nfmn);
K68=zeros(Nfvn,Nbxxn);
K69=zeros(Nfvn,Nbyyn);
K77=zeros(Nfmn,Nfmn);
K78=zeros(Nfmn,Nbxxn);
K79=zeros(Nfmn,Nbyyn);
K88=zeros(Nbxxn,Nbxxn);
K89=zeros(Nbxxn,Nbyyn);
K99=zeros(Nbyyn,Nbyyn);

%%% Loop over triangles in the mesh
```

```
for k=1:pMesh.nbTriangles6
%%% Get global node numbers for the nodes of Element k
    nodes=pMesh.TRIANGLES6(k,1:6);
%%% (x,y) coordinates of (local) nodes 1, 2, 3 (Vertices of T_k)
    x1=pMesh.POS(nodes(1),1);
    y1=pMesh.POS(nodes(1),2);
    x2=pMesh.POS(nodes(2),1);
    y2=pMesh.POS(nodes(2),2);
    x3=pMesh.POS(nodes(3),1);
    y3=pMesh.POS(nodes(3),2);
%%% Compute transormation J: [x,y] = [x1, y1] + Ju for u in reference
%%% triangle, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1; y2-y1,y3-y1];
    detJk=det(Jk);

    for n1=1:3 % Loop over 1st vertex nodes
        if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
            gn1=nodes(n1); % Global Node Number
            vfn1=pMesh.NodePtrs(gn1); % 1st Vertex Free Node Number
        %%% Compute mu_1 at quadrature points(stored in a row vector)
            mu1=muVal(:,1+6*(n1-1))';
            mu19=muVal(:,19)';
            mu20=muVal(:,20)';
            mu21=muVal(:,21)';
            b1=bCnst{1}(k,n1,1);
            b2=bCnst{1}(k,n1,2);
            b3=bCnst{1}(k,n1,3);
            mu_1=mu1+b1*mu19+b2*mu20+b3*mu21;
        %%% Compute mu_2 at quadrature points(stored in a row vector)
            mu2=muVal(:,2+6*(n1-1))';
            mu3=muVal(:,3+6*(n1-1))';
            mu19=muVal(:,19)';
            mu20=muVal(:,20)';
            mu21=muVal(:,21)';
            a1=bCnst{2}(k,n1,1);
            a2=bCnst{2}(k,n1,2);
            b1=bCnst{2}(k,n1,3);
            b2=bCnst{2}(k,n1,4);
            b3=bCnst{2}(k,n1,5);
            mu_2=a1*mu2+a2*mu3+b1*mu19+b2*mu20+b3*mu21;
        %%% Compute mu_3 at quadrature points(stored in a row vector)
```

```
        mu2=muVal(:,2+6*(n1-1))';
        mu3=muVal(:,3+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{3}(k,n1,1);
        a2=bCnst{3}(k,n1,2);
        b1=bCnst{3}(k,n1,3);
        b2=bCnst{3}(k,n1,4);
        b3=bCnst{3}(k,n1,5);
        mu_3=a1*mu2+a2*mu3+b1*mu19+b2*mu20+b3*mu21;
    %%% Compute mu_4 at quadrature points(stored in a row vector)
        mu4=muVal(:,4+6*(n1-1))';
        mu5=muVal(:,5+6*(n1-1))';
        mu6=muVal(:,6+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{4}(k,n1,1);
        a2=bCnst{4}(k,n1,2);
        a3=bCnst{4}(k,n1,3);
        b1=bCnst{4}(k,n1,4);
        b2=bCnst{4}(k,n1,5);
        b3=bCnst{4}(k,n1,6);
        mu_4=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;
    %%% Compute mu_5 at quadrature points(stored in a row vector)
        mu4=muVal(:,4+6*(n1-1))';
        mu5=muVal(:,5+6*(n1-1))';
        mu6=muVal(:,6+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{5}(k,n1,1);
        a2=bCnst{5}(k,n1,2);
        a3=bCnst{5}(k,n1,3);
        b1=bCnst{5}(k,n1,4);
        b2=bCnst{5}(k,n1,5);
        b3=bCnst{5}(k,n1,6);
        mu_5=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;
    %%% Compute mu_6 at quadrature points(stored in a row vector)
        mu4=muVal(:,4+6*(n1-1))';
```

```
mu5=muVal(:,5+6*(n1-1))';
mu6=muVal(:,6+6*(n1-1))';
mu19=muVal(:,19)';
mu20=muVal(:,20)';
mu21=muVal(:,21)';
a1=bCnst{6}(k,n1,1);
a2=bCnst{6}(k,n1,2);
a3=bCnst{6}(k,n1,3);
b1=bCnst{6}(k,n1,4);
b2=bCnst{6}(k,n1,5);
b3=bCnst{6}(k,n1,6);
mu_6=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;

for n2=1:3 % Loop over 2nd vertex nodes
    if pMesh.POS(nodes(n2),4)==-10 % Interior Plate Node
        gn2=nodes(n2); % Global Node Number
        vfn2=pMesh.NodePtrs(gn2); % 2nd Vertex Free Node Number
    %%% Compute Mu_1 at quadrature points(stored in a row vector)
        Mu1=muVal(:,1+6*(n2-1))';
        Mu19=muVal(:,19)';
        Mu20=muVal(:,20)';
        Mu21=muVal(:,21)';
        B1=bCnst{1}(k,n2,1);
        B2=bCnst{1}(k,n2,2);
        B3=bCnst{1}(k,n2,3);
        Mu_1=Mu1+B1*Mu19+B2*Mu20+B3*Mu21;
    %%% Compute Mu_2 at quadrature points(stored in a row vector)
        Mu2=muVal(:,2+6*(n2-1))';
        Mu3=muVal(:,3+6*(n2-1))';
        Mu19=muVal(:,19)';
        Mu20=muVal(:,20)';
        Mu21=muVal(:,21)';
        A1=bCnst{2}(k,n2,1);
        A2=bCnst{2}(k,n2,2);
        B1=bCnst{2}(k,n2,3);
        B2=bCnst{2}(k,n2,4);
        B3=bCnst{2}(k,n2,5);
        Mu_2=A1*Mu2+A2*Mu3+B1*Mu19+B2*Mu20+B3*Mu21;
    %%% Compute Mu_3 at quadrature points(stored in a row vector)
        Mu2=muVal(:,2+6*(n2-1))';
        Mu3=muVal(:,3+6*(n2-1))';
```

```
Mu19=muVal(:,19)';
Mu20=muVal(:,20)';
Mu21=muVal(:,21)';
A1=bCnst{3}(k,n2,1);
A2=bCnst{3}(k,n2,2);
B1=bCnst{3}(k,n2,3);
B2=bCnst{3}(k,n2,4);
B3=bCnst{3}(k,n2,5);
Mu_3=A1*Mu2+A2*Mu3+B1*Mu19+B2*Mu20+B3*Mu21;
%%% Compute Mu_4 at quadrature points(stored in a row vector)
Mu4=muVal(:,4+6*(n2-1))';
Mu5=muVal(:,5+6*(n2-1))';
Mu6=muVal(:,6+6*(n2-1))';
Mu19=muVal(:,19)';
Mu20=muVal(:,20)';
Mu21=muVal(:,21)';
A1=bCnst{4}(k,n2,1);
A2=bCnst{4}(k,n2,2);
A3=bCnst{4}(k,n2,3);
B1=bCnst{4}(k,n2,4);
B2=bCnst{4}(k,n2,5);
B3=bCnst{4}(k,n2,6);
Mu_4=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;
%%% Compute Mu_5 at quadrature points(stored in a row vector)
Mu4=muVal(:,4+6*(n2-1))';
Mu5=muVal(:,5+6*(n2-1))';
Mu6=muVal(:,6+6*(n2-1))';
Mu19=muVal(:,19)';
Mu20=muVal(:,20)';
Mu21=muVal(:,21)';
A1=bCnst{5}(k,n2,1);
A2=bCnst{5}(k,n2,2);
A3=bCnst{5}(k,n2,3);
B1=bCnst{5}(k,n2,4);
B2=bCnst{5}(k,n2,5);
B3=bCnst{5}(k,n2,6);
Mu_5=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;
%%% Compute Mu_6 at quadrature points(stored in a row vector)
Mu4=muVal(:,4+6*(n2-1))';
Mu5=muVal(:,5+6*(n2-1))';
Mu6=muVal(:,6+6*(n2-1))';
```

```
Mu19=muVal(:,19)';
Mu20=muVal(:,20)';
Mu21=muVal(:,21)';
A1=bCnst{6}(k,n2,1);
A2=bCnst{6}(k,n2,2);
A3=bCnst{6}(k,n2,3);
B1=bCnst{6}(k,n2,4);
B2=bCnst{6}(k,n2,5);
B3=bCnst{6}(k,n2,6);
Mu_6=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;

%%% Int_Tk (mu_1)*(Mu_1) Using Quadrature
qVal=detJk*((mu_1.*Mu_1)*qwt);
K11(vfn1,vfn2)=K11(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_1)*(Mu_2) Using Quadrature
qVal=detJk*((mu_1.*Mu_2)*qwt);
K12(vfn1,vfn2)=K12(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_1)*(Mu_3) Using Quadrature
qVal=detJk*((mu_1.*Mu_3)*qwt);
K13(vfn1,vfn2)=K13(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_1)*(Mu_4) Using Quadrature
qVal=detJk*((mu_1.*Mu_4)*qwt);
K14(vfn1,vfn2)=K14(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_1)*(Mu_5) Using Quadrature
qVal=detJk*((mu_1.*Mu_5)*qwt);
K15(vfn1,vfn2)=K15(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_1)*(Mu_6) Using Quadrature
qVal=detJk*((mu_1.*Mu_6)*qwt);
K16(vfn1,vfn2)=K16(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_2)*(Mu_2) Using Quadrature
qVal=detJk*((mu_2.*Mu_2)*qwt);
K22(vfn1,vfn2)=K22(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_2)*(Mu_3) Using Quadrature
qVal=detJk*((mu_2.*Mu_3)*qwt);
K23(vfn1,vfn2)=K23(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_2)*(Mu_4) Using Quadrature
qVal=detJk*((mu_2.*Mu_4)*qwt);
K24(vfn1,vfn2)=K24(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_2)*(Mu_5) Using Quadrature
qVal=detJk*((mu_2.*Mu_5)*qwt);
K25(vfn1,vfn2)=K25(vfn1,vfn2)+qVal;
```

```
%%% Int_Tk (mu_2)*(Mu_6) Using Quadrature
    qVal=detJk*((mu_2.*Mu_6)*qwt);
    K26(vfn1,vfn2)=K26(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_3)*(Mu_3) Using Quadrature
    qVal=detJk*((mu_3.*Mu_3)*qwt);
    K33(vfn1,vfn2)=K33(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_3)*(Mu_4) Using Quadrature
    qVal=detJk*((mu_3.*Mu_4)*qwt);
    K34(vfn1,vfn2)=K34(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_3)*(Mu_5) Using Quadrature
    qVal=detJk*((mu_3.*Mu_5)*qwt);
    K35(vfn1,vfn2)=K35(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_3)*(Mu_6) Using Quadrature
    qVal=detJk*((mu_3.*Mu_6)*qwt);
    K36(vfn1,vfn2)=K36(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_4)*(Mu_4) Using Quadrature
    qVal=detJk*((mu_4.*Mu_4)*qwt);
    K44(vfn1,vfn2)=K44(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_4)*(Mu_5) Using Quadrature
    qVal=detJk*((mu_4.*Mu_5)*qwt);
    K45(vfn1,vfn2)=K45(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_4)*(Mu_6) Using Quadrature
    qVal=detJk*((mu_4.*Mu_6)*qwt);
    K46(vfn1,vfn2)=K46(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_5)*(Mu_5) Using Quadrature
    qVal=detJk*((mu_5.*Mu_5)*qwt);
    K55(vfn1,vfn2)=K55(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_5)*(Mu_6) Using Quadrature
    qVal=detJk*((mu_5.*Mu_6)*qwt);
    K56(vfn1,vfn2)=K56(vfn1,vfn2)+qVal;
%%% Int_Tk (mu_6)*(Mu_6) Using Quadrature
    qVal=detJk*((mu_6.*Mu_6)*qwt);
    K66(vfn1,vfn2)=K66(vfn1,vfn2)+qVal;

elseif pMesh.NodePtrs(nodes(n2),3)>0 % Boundary d^2/dx^2 Node
    gn2=nodes(n2); % Global Node Number
    vbxxn2=pMesh.NodePtrs(gn2,3); % Boundary d^2/dx^2 Node Number
%%% Compute Mu_8 at quadrature points(stored in a row vector)
    Mu4=muVal(:,4+6*(n2-1))';
    Mu5=muVal(:,5+6*(n2-1))';
    Mu6=muVal(:,6+6*(n2-1))';
```

```
        Mu19=muVal(:,19)';
        Mu20=muVal(:,20)';
        Mu21=muVal(:,21)';
        A1=bCnst{4}(k,n2,1);
        A2=bCnst{4}(k,n2,2);
        A3=bCnst{4}(k,n2,3);
        B1=bCnst{4}(k,n2,4);
        B2=bCnst{4}(k,n2,5);
        B3=bCnst{4}(k,n2,6);
        Mu_8=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;

    %%% Int_Tk (mu_1)*(Mu_8) Using Quadrature
        qVal=detJk*((mu_1.*Mu_8)*qwt);
        K18(vfn1,vbxxn2)=K18(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (mu_2)*(Mu_8) Using Quadrature
        qVal=detJk*((mu_2.*Mu_8)*qwt);
        K28(vfn1,vbxxn2)=K28(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (mu_3)*(Mu_8) Using Quadrature
        qVal=detJk*((mu_3.*Mu_8)*qwt);
        K38(vfn1,vbxxn2)=K38(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (mu_4)*(Mu_8) Using Quadrature
        qVal=detJk*((mu_4.*Mu_8)*qwt);
        K48(vfn1,vbxxn2)=K48(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (mu_5)*(Mu_8) Using Quadrature
        qVal=detJk*((mu_5.*Mu_8)*qwt);
        K58(vfn1,vbxxn2)=K58(vfn1,vbxxn2)+qVal;
    %%% Int_Tk (mu_6)*(Mu_8) Using Quadrature
        qVal=detJk*((mu_6.*Mu_8)*qwt);
        K68(vfn1,vbxxn2)=K68(vfn1,vbxxn2)+qVal;

    elseif pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
        gn2=nodes(n2); % Global Node Number
        vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
    %%% Compute Mu_9 at quadrature points(stored in a row vector)
        Mu4=muVal(:,4+6*(n2-1))';
        Mu5=muVal(:,5+6*(n2-1))';
        Mu6=muVal(:,6+6*(n2-1))';
        Mu19=muVal(:,19)';
        Mu20=muVal(:,20)';
        Mu21=muVal(:,21)';
        A1=bCnst{6}(k,n2,1);
```

```
            A2=bCnst{6}(k,n2,2);
            A3=bCnst{6}(k,n2,3);
            B1=bCnst{6}(k,n2,4);
            B2=bCnst{6}(k,n2,5);
            B3=bCnst{6}(k,n2,6);
            Mu_9=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;

        %%% Int_Tk (mu_1)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_1.*Mu_9)*qwt);
            K19(vfn1,vbyyn2)=K19(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (mu_2)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_2.*Mu_9)*qwt);
            K29(vfn1,vbyyn2)=K29(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (mu_3)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_3.*Mu_9)*qwt);
            K39(vfn1,vbyyn2)=K39(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (mu_4)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_4.*Mu_9)*qwt);
            K49(vfn1,vbyyn2)=K49(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (mu_5)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_5.*Mu_9)*qwt);
            K59(vfn1,vbyyn2)=K59(vfn1,vbyyn2)+qVal;
        %%% Int_Tk (mu_6)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_6.*Mu_9)*qwt);
            K69(vfn1,vbyyn2)=K69(vfn1,vbyyn2)+qVal;
        end %end if loop for vertex 2
    end %end loop over vertex 2

    for n2=4:6 % Loop over midpoint nodes
        if pMesh.POS(nodes(n2),4)==-10 % Interior Plate Node
            gn2=nodes(n2); % Global Node Number
            mfn2=pMesh.NodePtrs(gn2); % Midpoint Free Node Number
        %%% Compute Mu_7 at quadrature points(stored in a row vector)
            %%% Note that only one of B1-B3 should be non-zero
            Mu19=muVal(:,19)';
            Mu20=muVal(:,20)';
            Mu21=muVal(:,21)';
            B1=bCnst{7}(k,n2-3,1); % Note n2-3 to get 1,2,3 from 4,5,6
            B2=bCnst{7}(k,n2-3,2);
            B3=bCnst{7}(k,n2-3,3);
            Mu_7=B1*Mu19+B2*Mu20+B3*Mu21;
```

```
            %%% Int_Tk (mu_1)*(Mu_7) Using Quadrature
                qVal=detJk*((mu_1.*Mu_7)*qwt);
                K17(vfn1,mfn2)=K17(vfn1,mfn2)+qVal;
            %%% Int_Tk (mu_2)*(Mu_7) Using Quadrature
                qVal=detJk*((mu_2.*Mu_7)*qwt);
                K27(vfn1,mfn2)=K27(vfn1,mfn2)+qVal;
            %%% Int_Tk (mu_3)*(Mu_7) Using Quadrature
                qVal=detJk*((mu_3.*Mu_7)*qwt);
                K37(vfn1,mfn2)=K37(vfn1,mfn2)+qVal;
            %%% Int_Tk (mu_4)*(Mu_7) Using Quadrature
                qVal=detJk*((mu_4.*Mu_7)*qwt);
                K47(vfn1,mfn2)=K47(vfn1,mfn2)+qVal;
            %%% Int_Tk (mu_5)*(Mu_7) Using Quadrature
                qVal=detJk*((mu_5.*Mu_7)*qwt);
                K57(vfn1,mfn2)=K57(vfn1,mfn2)+qVal;
            %%% Int_Tk (mu_6)*(Mu_7) Using Quadrature
                qVal=detJk*((mu_6.*Mu_7)*qwt);
                K67(vfn1,mfn2)=K67(vfn1,mfn2)+qVal;
            end % end if loop for midpoint node 2
        end % end loop over midpoint node 2

    elseif pMesh.NodePtrs(nodes(n1),3)>0 % Boundary d^2/dx^2 Node
        gn1=nodes(n1); % Global Node Number
        vbxxn1=pMesh.NodePtrs(gn1,3); % Boundary d^2/dx^2 Node Number
    %%% Compute mu_8 at quadrature points(stored in a row vector)
        mu4=muVal(:,4+6*(n1-1))';
        mu5=muVal(:,5+6*(n1-1))';
        mu6=muVal(:,6+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{4}(k,n1,1);
        a2=bCnst{4}(k,n1,2);
        a3=bCnst{4}(k,n1,3);
        b1=bCnst{4}(k,n1,4);
        b2=bCnst{4}(k,n1,5);
        b3=bCnst{4}(k,n1,6);
        mu_8=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;

        for n2=1:3 % Loop over 2nd vertex nodes
```

```
if pMesh.NodePtrs(nodes(n2),3)>0 % Boundary d^2/dx^2 Node
    gn2=nodes(n2); % Global Node Number
    vbxxn2=pMesh.NodePtrs(gn2,3); % Boundary d^2/dx^2 Node Number
%%% Compute Mu_8 at quadrature points(stored in a row vector)
    Mu4=muVal(:,4+6*(n2-1))';
    Mu5=muVal(:,5+6*(n2-1))';
    Mu6=muVal(:,6+6*(n2-1))';
    Mu19=muVal(:,19)';
    Mu20=muVal(:,20)';
    Mu21=muVal(:,21)';
    A1=bCnst{4}(k,n2,1);
    A2=bCnst{4}(k,n2,2);
    A3=bCnst{4}(k,n2,3);
    B1=bCnst{4}(k,n2,4);
    B2=bCnst{4}(k,n2,5);
    B3=bCnst{4}(k,n2,6);
    Mu_8=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;
%%% Int_Tk (mu_8)*(Mu_8) Using Quadrature
    qVal=detJk*((mu_8.*Mu_8)*qwt);
    K88(vbxxn1,vbxxn2)=K88(vbxxn1,vbxxn2)+qVal;

elseif pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
    gn2=nodes(n2); % Global Node Number
    vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
%%% Compute Mu_9 at quadrature points(stored in a row vector)
    Mu4=muVal(:,4+6*(n2-1))';
    Mu5=muVal(:,5+6*(n2-1))';
    Mu6=muVal(:,6+6*(n2-1))';
    Mu19=muVal(:,19)';
    Mu20=muVal(:,20)';
    Mu21=muVal(:,21)';
    A1=bCnst{6}(k,n2,1);
    A2=bCnst{6}(k,n2,2);
    A3=bCnst{6}(k,n2,3);
    B1=bCnst{6}(k,n2,4);
    B2=bCnst{6}(k,n2,5);
    B3=bCnst{6}(k,n2,6);
    Mu_9=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;
%%% Int_Tk (mu_8)*(Mu_9) Using Quadrature
    qVal=detJk*((mu_8.*Mu_9)*qwt);
    K89(vbxxn1,vbyyn2)=K89(vbxxn1,vbyyn2)+qVal;
```

```
        end % ends if loop over vertex 2
    end % ends loop over vertex 2

elseif pMesh.NodePtrs(nodes(n1),3)<0 % Boundary d^2/dy^2 Node
    gn1=nodes(n1); % Global Node Number
    vbyyn1=-pMesh.NodePtrs(gn1,3); % Boundary d^2/dy^2 Node Number
%%% Compute mu_9 at quadrature points(stored in a row vector)
    mu4=muVal(:,4+6*(n1-1))';
    mu5=muVal(:,5+6*(n1-1))';
    mu6=muVal(:,6+6*(n1-1))';
    mu19=muVal(:,19)';
    mu20=muVal(:,20)';
    mu21=muVal(:,21)';
    a1=bCnst{6}(k,n1,1);
    a2=bCnst{6}(k,n1,2);
    a3=bCnst{6}(k,n1,3);
    b1=bCnst{6}(k,n1,4);
    b2=bCnst{6}(k,n1,5);
    b3=bCnst{6}(k,n1,6);
    mu_9=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;

    for n2=1:3 % Loop over 2nd vertex nodes
        if pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
            gn2=nodes(n2); % Global Node Number
            vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
        %%% Compute Mu_9 at quadrature points(stored in a row vector)
            Mu4=muVal(:,4+6*(n2-1))';
            Mu5=muVal(:,5+6*(n2-1))';
            Mu6=muVal(:,6+6*(n2-1))';
            Mu19=muVal(:,19)';
            Mu20=muVal(:,20)';
            Mu21=muVal(:,21)';
            A1=bCnst{6}(k,n2,1);
            A2=bCnst{6}(k,n2,2);
            A3=bCnst{6}(k,n2,3);
            B1=bCnst{6}(k,n2,4);
            B2=bCnst{6}(k,n2,5);
            B3=bCnst{6}(k,n2,6);
            Mu_9=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;
        %%% Int_Tk (mu_9)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_9.*Mu_9)*qwt);
```

```
                K99(vbyyn1,vbyyn2)=K99(vbyyn1,vbyyn2)+qVal;
            end
        end
    end %end if loop for vertex 1
end %end loop over vertex 1

for n1=4:6 % Loop over midpoint nodes
    if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
        gn1=nodes(n1); % Global Node Number
        mfn1=pMesh.NodePtrs(gn1); % 1st Midpoint Free Node Number
    %%% Compute mu_7 at quadrature points(stored in a row vector)
    %%% Note that only one of b1-b3 should be non-zero
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        b1=bCnst{7}(k,n1-3,1); % Note n1-3 to get 1,2,3 from 4,5,6
        b2=bCnst{7}(k,n1-3,2);
        b3=bCnst{7}(k,n1-3,3);
        mu_7=b1*mu19+b2*mu20+b3*mu21;

        for n2=4:6 % Loop over midpoint nodes
            if pMesh.POS(nodes(n2),4)==-10 % Interior Plate Node
                gn2=nodes(n2); % Global Node Number
                mfn2=pMesh.NodePtrs(gn2); % 2nd Midpoint Free Node Number
            %%% Compute Mu_7 at quadrature points(stored in a row vector)
            %%% Note that only one of B1-B3 should be non-zero
                Mu19=muVal(:,19)';
                Mu20=muVal(:,20)';
                Mu21=muVal(:,21)';
                B1=bCnst{7}(k,n2-3,1); % Note n2-3 to get 1,2,3 from 4,5,6
                B2=bCnst{7}(k,n2-3,2);
                B3=bCnst{7}(k,n2-3,3);
                Mu_7=B1*Mu19+B2*Mu20+B3*Mu21;
            %%% Int_Tk (mu_7)*(Mu_7) Using Quadrature
                qVal=detJk*((mu_7.*Mu_7)*qwt);
                K77(mfn1,mfn2)=K77(mfn1,mfn2)+qVal;
            end
        end

        for n2=1:3 % Loop over 2nd vertex nodes
            if pMesh.NodePtrs(nodes(n2),3)>0 % Boundary d^2/dx^2 Node
```

```
            gn2=nodes(n2); % Global Node Number
            vbxxn2=pMesh.NodePtrs(gn2,3); % Boundary d^2/dx^2 Node Number
        %%% Compute Mu_8 at quadrature points(stored in a row vector)
            Mu4=muVal(:,4+6*(n2-1))';
            Mu5=muVal(:,5+6*(n2-1))';
            Mu6=muVal(:,6+6*(n2-1))';
            Mu19=muVal(:,19)';
            Mu20=muVal(:,20)';
            Mu21=muVal(:,21)';
            A1=bCnst{4}(k,n2,1);
            A2=bCnst{4}(k,n2,2);
            A3=bCnst{4}(k,n2,3);
            B1=bCnst{4}(k,n2,4);
            B2=bCnst{4}(k,n2,5);
            B3=bCnst{4}(k,n2,6);
            Mu_8=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;
        %%% Int_Tk (mu_7)*(Mu_8) Using Quadrature
            qVal=detJk*((mu_7.*Mu_8)*qwt);
            K78(mfn1,vbxxn2)=K78(mfn1,vbxxn2)+qVal;
        elseif pMesh.NodePtrs(nodes(n2),3)<0 % Boundary d^2/dy^2 Node
            gn2=nodes(n2); % Global Node Number
            vbyyn2=-pMesh.NodePtrs(gn2,3); % Boundary d^2/dy^2 Node Number
        %%% Compute Mu_9 at quadrature points(stored in a row vector)
            Mu4=muVal(:,4+6*(n2-1))';
            Mu5=muVal(:,5+6*(n2-1))';
            Mu6=muVal(:,6+6*(n2-1))';
            Mu19=muVal(:,19)';
            Mu20=muVal(:,20)';
            Mu21=muVal(:,21)';
            A1=bCnst{6}(k,n2,1);
            A2=bCnst{6}(k,n2,2);
            A3=bCnst{6}(k,n2,3);
            B1=bCnst{6}(k,n2,4);
            B2=bCnst{6}(k,n2,5);
            B3=bCnst{6}(k,n2,6);
            Mu_9=A1*Mu4+A2*Mu5+A3*Mu6+B1*Mu19+B2*Mu20+B3*Mu21;
        %%% Int_Tk (mu_7)*(Mu_9) Using Quadrature
            qVal=detJk*((mu_7.*Mu_9)*qwt);
            K79(mfn1,vbyyn2)=K79(mfn1,vbyyn2)+qVal;
        end
    end
```

```
        end
    end
end %end loop over triangles
K21=transpose(K12);  % Copy to Lower Triangular blocks
K31=transpose(K13);
K32=transpose(K23);
K41=transpose(K14);
K42=transpose(K24);
K43=transpose(K34);
K51=transpose(K15);
K52=transpose(K25);
K53=transpose(K35);
K54=transpose(K45);
K61=transpose(K16);
K62=transpose(K26);
K63=transpose(K36);
K64=transpose(K46);
K65=transpose(K56);
K71=transpose(K17);
K72=transpose(K27);
K73=transpose(K37);
K74=transpose(K47);
K75=transpose(K57);
K76=transpose(K67);
K81=transpose(K18);
K82=transpose(K28);
K83=transpose(K38);
K84=transpose(K48);
K85=transpose(K58);
K86=transpose(K68);
K87=transpose(K78);
K91=transpose(K19);
K92=transpose(K29);
K93=transpose(K39);
K94=transpose(K49);
K95=transpose(K59);
K96=transpose(K69);
K97=transpose(K79);
K98=transpose(K89);
%%% Assemble the Stiffness Matrix K
K=(lambda^2)*[K11,K12,K13,K14,K15,K16,K17,K18,K19; K21,K22,K23,K24,K25,K26,K27,K28,K29;
```

```
    K31,K32,K33,K34,K35,K36,K37,K38,K39;  K41,K42,K43,K44,K45,K46,K47,K48,K49;
    K51,K52,K53,K54,K55,K56,K57,K58,K59;  K61,K62,K63,K64,K65,K66,K67,K68,K69;
    K71,K72,K73,K74,K75,K76,K77,K78,K79;  K81,K82,K83,K84,K85,K86,K87,K88,K89;
    K91,K92,K93,K94,K95,K96,K97,K98,K99];
K=sparse(K);
```

StiffnessAFT2D.m

```
function K=StiffnessAFT2D(pMesh,muSoln,fTildeA)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  This function creates the part of the plate bilinear form that
%%%  corresponds to the term:
%%%  lambda(grad f~(mu1), grad f~(mu2))_O + lambda^2(f~(mu1), f~(mu2))_O
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Initialize Stiffness Block Matrices
Nfvn=length(pMesh.FVNodePtrs(:,1));
Nfmn=length(pMesh.FMNodePtrs(:,1));
Nbxxn=length(pMesh.BxxNodePtrs(:,1));
Nbyyn=length(pMesh.ByyNodePtrs(:,1));
K11=zeros(Nfvn,Nfvn);
K12=zeros(Nfvn,Nfvn);
K13=zeros(Nfvn,Nfvn);
K14=zeros(Nfvn,Nfvn);
K15=zeros(Nfvn,Nfvn);
K16=zeros(Nfvn,Nfvn);
K17=zeros(Nfvn,Nfmn);
K18=zeros(Nfvn,Nbxxn);
K19=zeros(Nfvn,Nbyyn);
K22=zeros(Nfvn,Nfvn);
K23=zeros(Nfvn,Nfvn);
K24=zeros(Nfvn,Nfvn);
K25=zeros(Nfvn,Nfvn);
K26=zeros(Nfvn,Nfvn);
K27=zeros(Nfvn,Nfmn);
K28=zeros(Nfvn,Nbxxn);
K29=zeros(Nfvn,Nbyyn);
K33=zeros(Nfvn,Nfvn);
K34=zeros(Nfvn,Nfvn);
K35=zeros(Nfvn,Nfvn);
```

```
K36=zeros(Nfvn,Nfvn);
K37=zeros(Nfvn,Nfmn);
K38=zeros(Nfvn,Nbxxn);
K39=zeros(Nfvn,Nbyyn);
K44=zeros(Nfvn,Nfvn);
K45=zeros(Nfvn,Nfvn);
K46=zeros(Nfvn,Nfvn);
K47=zeros(Nfvn,Nfmn);
K48=zeros(Nfvn,Nbxxn);
K49=zeros(Nfvn,Nbyyn);
K55=zeros(Nfvn,Nfvn);
K56=zeros(Nfvn,Nfvn);
K57=zeros(Nfvn,Nfmn);
K58=zeros(Nfvn,Nbxxn);
K59=zeros(Nfvn,Nbyyn);
K66=zeros(Nfvn,Nfvn);
K67=zeros(Nfvn,Nfmn);
K68=zeros(Nfvn,Nbxxn);
K69=zeros(Nfvn,Nbyyn);
K77=zeros(Nfmn,Nfmn);
K78=zeros(Nfmn,Nbxxn);
K79=zeros(Nfmn,Nbyyn);
K88=zeros(Nbxxn,Nbxxn);
K89=zeros(Nbxxn,Nbyyn);
K99=zeros(Nbyyn,Nbyyn);

for i=1:Nfvn
    for j=1:Nfvn
        K11(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,j));
        K12(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,Nfvn+j));
        K13(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,2*Nfvn+j));
        K14(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,3*Nfvn+j));
        K15(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,4*Nfvn+j));
        K16(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,5*Nfvn+j));
        K22(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,Nfvn+j));
        K23(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,2*Nfvn+j));
        K24(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,3*Nfvn+j));
        K25(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,4*Nfvn+j));
        K26(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,5*Nfvn+j));
        K33(i,j)=transpose(muSoln(:,2*Nfvn+i))*(fTildeA*muSoln(:,2*Nfvn+j));
        K34(i,j)=transpose(muSoln(:,2*Nfvn+i))*(fTildeA*muSoln(:,3*Nfvn+j));
```

```
            K35(i,j)=transpose(muSoln(:,2*Nfvn+i))*(fTildeA*muSoln(:,4*Nfvn+j));
            K36(i,j)=transpose(muSoln(:,2*Nfvn+i))*(fTildeA*muSoln(:,5*Nfvn+j));
            K44(i,j)=transpose(muSoln(:,3*Nfvn+i))*(fTildeA*muSoln(:,3*Nfvn+j));
            K45(i,j)=transpose(muSoln(:,3*Nfvn+i))*(fTildeA*muSoln(:,4*Nfvn+j));
            K46(i,j)=transpose(muSoln(:,3*Nfvn+i))*(fTildeA*muSoln(:,5*Nfvn+j));
            K55(i,j)=transpose(muSoln(:,4*Nfvn+i))*(fTildeA*muSoln(:,4*Nfvn+j));
            K56(i,j)=transpose(muSoln(:,4*Nfvn+i))*(fTildeA*muSoln(:,5*Nfvn+j));
            K66(i,j)=transpose(muSoln(:,5*Nfvn+i))*(fTildeA*muSoln(:,5*Nfvn+j));
        end
        for j=1:Nfmn
            K17(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,6*Nfvn+j));
            K27(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+j));
            K37(i,j)=transpose(muSoln(:,2*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+j));
            K47(i,j)=transpose(muSoln(:,3*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+j));
            K57(i,j)=transpose(muSoln(:,4*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+j));
            K67(i,j)=transpose(muSoln(:,5*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+j));
        end
        for j=1:Nbxxn
            K18(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
            K28(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
            K38(i,j)=transpose(muSoln(:,2*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
            K48(i,j)=transpose(muSoln(:,3*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
            K58(i,j)=transpose(muSoln(:,4*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
            K68(i,j)=transpose(muSoln(:,5*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
        end
        for j=1:Nbyyn
            K19(i,j)=transpose(muSoln(:,i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
            K29(i,j)=transpose(muSoln(:,Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
            K39(i,j)=transpose(muSoln(:,2*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
            K49(i,j)=transpose(muSoln(:,3*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
            K59(i,j)=transpose(muSoln(:,4*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
            K69(i,j)=transpose(muSoln(:,5*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
        end
    end
    for i=1:Nfmn
        for j=1:Nfmn
            K77(i,j)=transpose(muSoln(:,6*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+j));
        end
        for j=1:Nbxxn
            K78(i,j)=transpose(muSoln(:,6*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
        end
```

```
    for j=1:Nbyyn
        K79(i,j)=transpose(muSoln(:,6*Nfvn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
    end
end
for i=1:Nbxxn
    for j=1:Nbxxn
        K88(i,j)=transpose(muSoln(:,6*Nfvn+Nfmn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+j));
    end
    for j=1:Nbyyn
        K89(i,j)=transpose(muSoln(:,6*Nfvn+Nfmn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+j));
    end
end
for i=1:Nbyyn
    for j=1:Nbyyn
        K99(i,j)=transpose(muSoln(:,6*Nfvn+Nfmn+Nbxxn+i))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+...
            j));
    end
end
K21=transpose(K12);  % Copy to Lower Triangular blocks
K31=transpose(K13);
K32=transpose(K23);
K41=transpose(K14);
K42=transpose(K24);
K43=transpose(K34);
K51=transpose(K15);
K52=transpose(K25);
K53=transpose(K35);
K54=transpose(K45);
K61=transpose(K16);
K62=transpose(K26);
K63=transpose(K36);
K64=transpose(K46);
K65=transpose(K56);
K71=transpose(K17);
K72=transpose(K27);
K73=transpose(K37);
K74=transpose(K47);
K75=transpose(K57);
K76=transpose(K67);
K81=transpose(K18);
K82=transpose(K28);
```

```
K83=transpose(K38);
K84=transpose(K48);
K85=transpose(K58);
K86=transpose(K68);
K87=transpose(K78);
K91=transpose(K19);
K92=transpose(K29);
K93=transpose(K39);
K94=transpose(K49);
K95=transpose(K59);
K96=transpose(K69);
K97=transpose(K79);
K98=transpose(K89);
%%% Assemble the Stiffness Matrix K
K=[K11,K12,K13,K14,K15,K16,K17,K18,K19; K21,K22,K23,K24,K25,K26,K27,K28,K29;
   K31,K32,K33,K34,K35,K36,K37,K38,K39; K41,K42,K43,K44,K45,K46,K47,K48,K49;
   K51,K52,K53,K54,K55,K56,K57,K58,K59; K61,K62,K63,K64,K65,K66,K67,K68,K69;
   K71,K72,K73,K74,K75,K76,K77,K78,K79; K81,K82,K83,K84,K85,K86,K87,K88,K89;
   K91,K92,K93,K94,K95,K96,K97,K98,K99];
K=sparse(K);
```

StiffnessQ3D.m

```
function [K,As]=StiffnessQ3D(Mesh,phiVal,phixVal,phiyVal,phizVal,etaVal,C,qwt,lambda)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This function creates the fluid stiffness matrix for the free nodes
%
% C is a vector giving the values of \int_O eta_j, note that this
% calculation assumes that M(O)=1(which it is in this geometry)
%
% The stiffness matrix K will have the form:
%        A    0    0   B1
%        0    A    0   B2
%   K =  0    0    A   B3
%       B1T  B2T  B3T  0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nf=length(Mesh.FNodePtrs);  % The number of free nodes
Np=length(Mesh.PNodePtrs);  % The number of pressure nodes
Nt=Mesh.nbTets10;
```

```
%%% Number of quadrature points
ql=length(qwt);
%%% Initialize loop variables
B1Val=zeros(ql,1);
B2Val=zeros(ql,1);
B3Val=zeros(ql,1);

% Need to define K separately from its parts: A, B1, B2, B3, etc.
A=zeros(Nf,Nf);
B1=zeros(Nf,Np);
B2=zeros(Nf,Np);
B3=zeros(Nf,Np);

%%% Loop over all tetrahedra and compute element by element the
%%% values of A, B1, B2 and B3.
%%% A is symmetric, so we only compute the upper triangle.
for k = 1:Nt;
% Get global node numbers for the nodes of Element k
    nodes=Mesh.TETS10(k,1:10);
%%% (x,y,z) coordinates of (local) nodes 1, 2, 3, 4 (Vertices of T_k)
    x1=Mesh.POS(nodes(1),1);
    y1=Mesh.POS(nodes(1),2);
    z1=Mesh.POS(nodes(1),3);
    x2=Mesh.POS(nodes(2),1);
    y2=Mesh.POS(nodes(2),2);
    z2=Mesh.POS(nodes(2),3);
    x3=Mesh.POS(nodes(3),1);
    y3=Mesh.POS(nodes(3),2);
    z3=Mesh.POS(nodes(3),3);
    x4=Mesh.POS(nodes(4),1);
    y4=Mesh.POS(nodes(4),2);
    z4=Mesh.POS(nodes(4),3);
% Compute transormation J: [x,y,z] = [x1, y1, z1] + Ju for u in reference
% tetrahedron, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1,x4-x1; y2-y1,y3-y1,y4-y1; z2-z1,z3-z1,z4-z1];
    JkIT=transpose(inv(Jk));
    detJk=abs(det(Jk));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% The Matrix A i.e. the contribution from  grad phi_i dot grad phi_j
    %%%
    %%% Compute \int_{T_k} \nabla phi_s \cdot \nabla phi_r dV using
```

```
%%% quadrature and the reference tetrahedron, if both are free nodes.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for r=1:10;  % r is the 1st local node number
    nkr=nodes(r);
    if Mesh.NodePtrs(nkr,1)>0; % If node is free then...
        i_r=Mesh.NodePtrs(nkr,1); % Free node number for A_ij
        for s=r:10;  % s is the 2nd local node number(upper half)
            nks=nodes(s);
            if Mesh.NodePtrs(nks,1)>0;   % Is it a free node?
                j_s=Mesh.NodePtrs(nks,1); % Free node number for A_ij
            %%% Contribution from -Delta phi
                gradphi_rf=JkIT*transpose([phixVal(:,r),phiyVal(:,r),phizVal(:,r)]);
                gradphi_sf=JkIT*transpose([phixVal(:,s),phiyVal(:,s),phizVal(:,s)]);
                sum1=sum(gradphi_rf.*gradphi_sf)*qwt;
            %%% Contribution from lambda phi
                phi_rf=phiVal(:,r);
                phi_sf=phiVal(:,s);
                sum2=sum(phi_rf.*phi_sf.*qwt);

                A(i_r,j_s)=A(i_r,j_s) + detJk*sum1 + detJk*lambda*sum2;
                if r<s;
                    A(j_s,i_r)=A(i_r,j_s);
                end
            end
        end
    end
end % ends loop for A

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% The Matrices B1, B2, and B3 i.e. the contribution from
%%% eta_j * div phi_i
%%%
%%% Compute \int_{T_k} eta_r * div phi_s dV using
%%% quadrature and the reference tetrahedron, if nodes "s" is free, and
%%% add it to B1("s","r"), etc.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% B1, B2, B3
for r=1:4;  % r is the 1st local node number (pressure nodes)
    nkr=nodes(r);
    i_r=Mesh.NodePtrs(nkr,2); % Pressure node number for B_ij
```

```
        for s=1:10;  % s is the 2nd local node number
            nks=nodes(s);
            if Mesh.NodePtrs(nks,1)>0;   % Is it a free node?
               j_s=Mesh.NodePtrs(nks,1); % Free node number for B_ij

               for i=1:ql
                   eta_r=etaVal(i,r);
                   gradphi_s=[phixVal(i,s);phiyVal(i,s);phizVal(i,s)];

                   B1divphi_s=JkIT(1,:)*gradphi_s;
                   B1Val(i,1)=qwt(i)*eta_r*B1divphi_s;

                   B2divphi_s=JkIT(2,:)*gradphi_s;
                   B2Val(i,1)=qwt(i)*eta_r*B2divphi_s;

                   B3divphi_s=JkIT(3,:)*gradphi_s;
                   B3Val(i,1)=qwt(i)*eta_r*B3divphi_s;
               end

               sum2=sum(B1Val);
               B1(j_s,i_r)=B1(j_s,i_r) - detJk*sum2; % Note - sign
               sum3=sum(B2Val);
               B2(j_s,i_r)=B2(j_s,i_r) - detJk*sum3;
               sum4=sum(B3Val);
               B3(j_s,i_r)=B3(j_s,i_r) - detJk*sum4;
            end
        end
    end % ends first loop for B1, B2, B3

    %%% B1, B2, B3 %%% Including these values only changes the matrix on
    %%% the order of 10^(-18) or less.  Thus we have written the code to
    %%% compute it, but it is effectively meaningless and only wastes
    %%% computation power and destroys the sparsity of B1, B2, and B3.
%     for s=1:10  % Need to factor in L^2_0 corrector for eta
%         nks=nodes(s);
%         if Mesh.NodePtrs(nks,1)>0; % Is it a free node?
%            j_s=Mesh.NodePtrs(nks,1); % Free node number for B_ij
%        %%% Compute value of \int_Tk div(phi_j)
%            for i=1:ql
%                gradphi_s=[phixVal(i,s);phiyVal(i,s);phizVal(i,s)];
%                B1divphi_s=JkIT(1,:)*gradphi_s;
```

```
%                   B1Val(i,1)=qwt(i)*B1divphi_s;
%
%                   B2divphi_s=JkIT(2,:)*gradphi_s;
%                   B2Val(i,1)=qwt(i)*B2divphi_s;
%
%                   B3divphi_s=JkIT(3,:)*gradphi_s;
%                   B3Val(i,1)=qwt(i)*B3divphi_s;
%               end
%               sum5=detJk*sum(B1Val);
%               sum6=detJk*sum(B2Val);
%               sum7=detJk*sum(B3Val);
%               for pn=1:Np
%                   B1(j_s,pn)=B1(j_s,pn) + C(pn,1)*sum5; % Note + sign
%                   B2(j_s,pn)=B2(j_s,pn) + C(pn,1)*sum6;
%                   B3(j_s,pn)=B3(j_s,pn) + C(pn,1)*sum7;
%               end
%           end
%       end % Ends second loop for B1, B2, B3
end
As=sparse(A);
B1s=sparse(B1);
B2s=sparse(B2);
B3s=sparse(B3);

% Include the matrices B1^T, B2^T and B3^T into the matrix K.
B1Ts = transpose(B1s);
B2Ts = transpose(B2s);
B3Ts = transpose(B3s);
Z1s=sparse(Nf,Nf);
Z2s=sparse(Np,Np);
K=[As, Z1s, Z1s, B1s; Z1s, As, Z1s, B2s; Z1s, Z1s, As, B3s; B1Ts, B2Ts, B3Ts, Z2s];
```

StiffnessQ3DB.m

```
function [C1,C2]=StiffnessQ3DB(Mesh,phiVal,phixVal,phiyVal,phizVal,qwt,lambda)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This function creates part of the stiffness matrix corresponding to the
%  nodes boundary inhomogeneous nodes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Nf=length(Mesh.FNodePtrs);   % # of free nodes
Nih=length(Mesh.IHNodePtrs); % # of inhomogeneous constrained nodes
Nt=Mesh.nbTets10;
%%% Initialize loop variables
C1=zeros(Nf,Nih); % free row, IH column
C2=zeros(Nih,Nih); % IH row, IH column
%%% Loop over all tetrahedra and compute element by element the
%%% values of of C1 and C2
for k = 1:Nt;
% Get global node numbers for the nodes of Element k
    nodes=Mesh.TETS10(k,1:10);
%%% (x,y,z) coordinates of (local) nodes 1, 2, 3, 4 (Vertices of T_k)
    x1=Mesh.POS(nodes(1),1);
    y1=Mesh.POS(nodes(1),2);
    z1=Mesh.POS(nodes(1),3);
    x2=Mesh.POS(nodes(2),1);
    y2=Mesh.POS(nodes(2),2);
    z2=Mesh.POS(nodes(2),3);
    x3=Mesh.POS(nodes(3),1);
    y3=Mesh.POS(nodes(3),2);
    z3=Mesh.POS(nodes(3),3);
    x4=Mesh.POS(nodes(4),1);
    y4=Mesh.POS(nodes(4),2);
    z4=Mesh.POS(nodes(4),3);
% Compute transormation J: [x,y,z] = [x1, y1, z1] + Ju for u in reference
% tetrahedron, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1,x4-x1; y2-y1,y3-y1,y4-y1; z2-z1,z3-z1,z4-z1];
    JkIT=transpose(inv(Jk));
    detJk=abs(det(Jk));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% The Matrices C1, C2: contribution from  grad phi_i dot grad phi_j
    %%% and lambda phi_j *phi_j
    %%%
    %%% Compute \int_{T_k} grad phi_s dot grad phi_r + lambda phi_s*phi_r dV
    %%% using quadrature and the reference tetrahedron.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for s=1:10;  % s is the 1st local node number
        nks=nodes(s);
        if Mesh.NodePtrs(nks,3)>0; % If node is IH then...
            j_s=Mesh.NodePtrs(nks,3); % IH Node number for C_ij
            for r=1:10;  % r is the 2nd local node number
```

```
                nkr=nodes(r);
                if Mesh.NodePtrs(nkr,3)>0;    % Is it an IH Node?
                    i_r=Mesh.NodePtrs(nkr,3); % IH Node number for C2_ij
                %%% Contribution from -Delta phi
                    gradphi_rf=JkIT*transpose([phixVal(:,r),phiyVal(:,r),phizVal(:,r)]);
                    gradphi_sf=JkIT*transpose([phixVal(:,s),phiyVal(:,s),phizVal(:,s)]);
                    sum1=sum(gradphi_rf.*gradphi_sf)*qwt;
                %%% Contribution from lambda phi
                    phi_rf=phiVal(:,r);
                    phi_sf=phiVal(:,s);
                    sum2=sum(phi_rf.*phi_sf.*qwt);
                    C2(i_r,j_s)=C2(i_r,j_s) + detJk*sum1 + detJk*lambda*sum2;

                elseif Mesh.NodePtrs(nkr,1)>0 % If node is free then...
                    i_r=Mesh.NodePtrs(nkr,1); % Free node number for C1_ij
                %%% Contribution from -Delta phi
                    gradphi_rf=JkIT*transpose([phixVal(:,r),phiyVal(:,r),phizVal(:,r)]);
                    gradphi_sf=JkIT*transpose([phixVal(:,s),phiyVal(:,s),phizVal(:,s)]);
                    sum1=sum(gradphi_rf.*gradphi_sf)*qwt;
                %%% Contribution from lambda phi
                    phi_rf=phiVal(:,r);
                    phi_sf=phiVal(:,s);
                    sum2=sum(phi_rf.*phi_sf.*qwt);
                    C1(i_r,j_s)=C1(i_r,j_s) + detJk*sum1 + detJk*lambda*sum2;
                end
            end
        end
    end
end
C1=sparse(C1);
C2=sparse(C2);
```

fTilde.m

```
function [muSol,muSolP]=fTilde(pMesh,Mesh,fK,bCnst,muNVal,phiVal,phixVal,phiyVal,phizVal,...
    etaVal,C,fqwt,lambda)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% For every basis function mu on the plate, f~(mu) solves:
%%%
%%% lambda f~ - Delta f~ + grad pi~ = 0                in O
```

```
%%%                    -div(f~) = -int_Omega mu   in O
%%%                         f~ = (0,0,0)          on S
%%%                         f~ = (0,0,mu)         on Omega
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nf=length(Mesh.FNodePtrs);  % The number of free nodes
Nih=length(Mesh.IHNodePtrs);  % The number of inhomogeneous constrained nodes
Np=length(Mesh.PNodePtrs); % The number of pressure nodes
Nfvn=length(pMesh.FVNodePtrs(:,1));
Nfmn=length(pMesh.FMNodePtrs(:,1));
Nbxxn=length(pMesh.BxxNodePtrs(:,1));
Nbyyn=length(pMesh.ByyNodePtrs(:,1));
muSol=zeros(3*Nf+Np,6*Nfvn+Nfmn+Nbxxn+Nbyyn);
muBC=zeros(Nih,6*Nfvn+Nfmn+Nbxxn+Nbyyn);
phiSoln=zeros(3*Nf+Np,Nih);

%%% Each Midpoint phi is on 2 triangles, and integrates to 1/6 on reference
%%% triangle and the ratio of areas is (6/Mesh.nbTri)/(1/2)
phiDiv=zeros(Nih,1); % integral of phi on boundary
mdptdivcnst=4/Mesh.nbTriangles6;
for i = 1:Nih
    gn=Mesh.IHNodePtrs(i,1);
    if Mesh.POS(gn,5)==0 % Midpoint Node
        phiDiv(i,1)=-mdptdivcnst;  % Note that - sign is included
    end
end
parfor i = 1:Nih
    G=zeros(Nih,1);
    G(i,1)=1;
    FT=LoadFT(Mesh,G,phiVal,phixVal,phiyVal,phizVal,etaVal,C,fqwt,lambda,phiDiv(i,1));
    phiSoln(:,i)=symmlq(fK,FT,1e-12,10000);
    %phiSoln(:,i)=fK\FT; %%% faster but less accurate
end
%%%% mu %%%%
for i = 1:Nfvn
%%% Create the interpolation function G for mu1, G=(G1,G2,G3)
    G31=zeros(Nih,1);
    G32=zeros(Nih,1);
    G33=zeros(Nih,1);
    G34=zeros(Nih,1);
    G35=zeros(Nih,1);
```

```
G36=zeros(Nih,1);
pni=pMesh.FVNodePtrs(i,1);
fni=pMesh.PtrsTo3DNodes(pni,1);
fcni=Mesh.NodePtrs(fni,3);
G31(fcni,1)=1;
for k=1:pMesh.nbTriangles6
    if pMesh.TRIANGLES6(k,1)==pni
    %%% Non-zero on nodes 1,4,6
        pni4=pMesh.TRIANGLES6(k,4);
        fni4=pMesh.PtrsTo3DNodes(pni4,1);
        fcni4=Mesh.NodePtrs(fni4,3);
        G31(fcni4,1)=1/2;
        G32(fcni4,1)=bCnst{2}(k,1,1)*muNVal(4,2)+bCnst{2}(k,1,2)*muNVal(4,3);
        G33(fcni4,1)=bCnst{3}(k,1,1)*muNVal(4,2)+bCnst{3}(k,1,2)*muNVal(4,3);
        G34(fcni4,1)=bCnst{4}(k,1,1)*muNVal(4,4)+bCnst{4}(k,1,2)*muNVal(4,5)+bCnst{4}(k...
            ,1,3)*muNVal(4,6);
        G35(fcni4,1)=bCnst{5}(k,1,1)*muNVal(4,4)+bCnst{5}(k,1,2)*muNVal(4,5)+bCnst{5}(k...
            ,1,3)*muNVal(4,6);
        G36(fcni4,1)=bCnst{6}(k,1,1)*muNVal(4,4)+bCnst{6}(k,1,2)*muNVal(4,5)+bCnst{6}(k...
            ,1,3)*muNVal(4,6);

        pni6=pMesh.TRIANGLES6(k,6);
        fni6=pMesh.PtrsTo3DNodes(pni6,1);
        fcni6=Mesh.NodePtrs(fni6,3);
        G31(fcni6,1)=1/2;
        G32(fcni6,1)=bCnst{2}(k,1,1)*muNVal(6,2)+bCnst{2}(k,1,2)*muNVal(6,3);
        G33(fcni6,1)=bCnst{3}(k,1,1)*muNVal(6,2)+bCnst{3}(k,1,2)*muNVal(6,3);
        G34(fcni6,1)=bCnst{4}(k,1,1)*muNVal(6,4)+bCnst{4}(k,1,2)*muNVal(6,5)+bCnst{4}(k...
            ,1,3)*muNVal(6,6);
        G35(fcni6,1)=bCnst{5}(k,1,1)*muNVal(6,4)+bCnst{5}(k,1,2)*muNVal(6,5)+bCnst{5}(k...
            ,1,3)*muNVal(6,6);
        G36(fcni6,1)=bCnst{6}(k,1,1)*muNVal(6,4)+bCnst{6}(k,1,2)*muNVal(6,5)+bCnst{6}(k...
            ,1,3)*muNVal(6,6);
    elseif pMesh.TRIANGLES6(k,2)==pni
    %%% Non-zero on nodes 2,5,4
        pni5=pMesh.TRIANGLES6(k,5);
        fni5=pMesh.PtrsTo3DNodes(pni5,1);
        fcni5=Mesh.NodePtrs(fni5,3);
        G31(fcni5,1)=1/2;
        G32(fcni5,1)=bCnst{2}(k,2,1)*muNVal(5,8)+bCnst{2}(k,2,2)*muNVal(5,9);
        G33(fcni5,1)=bCnst{3}(k,2,1)*muNVal(5,8)+bCnst{3}(k,2,2)*muNVal(5,9);
```

```
        G34(fcni5,1)=bCnst{4}(k,2,1)*muNVal(5,10)+bCnst{4}(k,2,2)*muNVal(5,11)+bCnst{4}(k...
            ,2,3)*muNVal(5,12);
        G35(fcni5,1)=bCnst{5}(k,2,1)*muNVal(5,10)+bCnst{5}(k,2,2)*muNVal(5,11)+bCnst{5}(k...
            ,2,3)*muNVal(5,12);
        G36(fcni5,1)=bCnst{6}(k,2,1)*muNVal(5,10)+bCnst{6}(k,2,2)*muNVal(5,11)+bCnst{6}(k...
            ,2,3)*muNVal(5,12);

        pni4=pMesh.TRIANGLES6(k,4);
        fni4=pMesh.PtrsTo3DNodes(pni4,1);
        fcni4=Mesh.NodePtrs(fni4,3);
        G31(fcni4,1)=1/2;
        G32(fcni4,1)=bCnst{2}(k,2,1)*muNVal(4,8)+bCnst{2}(k,2,2)*muNVal(4,9);
        G33(fcni4,1)=bCnst{3}(k,2,1)*muNVal(4,8)+bCnst{3}(k,2,2)*muNVal(4,9);
        G34(fcni4,1)=bCnst{4}(k,2,1)*muNVal(4,10)+bCnst{4}(k,2,2)*muNVal(4,11)+bCnst{4}(k...
            ,2,3)*muNVal(4,12);
        G35(fcni4,1)=bCnst{5}(k,2,1)*muNVal(4,10)+bCnst{5}(k,2,2)*muNVal(4,11)+bCnst{5}(k...
            ,2,3)*muNVal(4,12);
        G36(fcni4,1)=bCnst{6}(k,2,1)*muNVal(4,10)+bCnst{6}(k,2,2)*muNVal(4,11)+bCnst{6}(k...
            ,2,3)*muNVal(4,12);
    elseif pMesh.TRIANGLES6(k,3)==pni
    %%% Non-zero on nodes 3,6,5
        pni6=pMesh.TRIANGLES6(k,6);
        fni6=pMesh.PtrsTo3DNodes(pni6,1);
        fcni6=Mesh.NodePtrs(fni6,3);
        G31(fcni6,1)=1/2;
        G32(fcni6,1)=bCnst{2}(k,3,1)*muNVal(6,14)+bCnst{2}(k,3,2)*muNVal(6,15);
        G33(fcni6,1)=bCnst{3}(k,3,1)*muNVal(6,14)+bCnst{3}(k,3,2)*muNVal(6,15);
        G34(fcni6,1)=bCnst{4}(k,3,1)*muNVal(6,16)+bCnst{4}(k,3,2)*muNVal(6,17)+bCnst{4}(k...
            ,3,3)*muNVal(6,18);
        G35(fcni6,1)=bCnst{5}(k,3,1)*muNVal(6,16)+bCnst{5}(k,3,2)*muNVal(6,17)+bCnst{5}(k...
            ,3,3)*muNVal(6,18);
        G36(fcni6,1)=bCnst{6}(k,3,1)*muNVal(6,16)+bCnst{6}(k,3,2)*muNVal(6,17)+bCnst{6}(k...
            ,3,3)*muNVal(6,18);

        pni5=pMesh.TRIANGLES6(k,5);
        fni5=pMesh.PtrsTo3DNodes(pni5,1);
        fcni5=Mesh.NodePtrs(fni5,3);
        G31(fcni5,1)=1/2;
        G32(fcni5,1)=bCnst{2}(k,3,1)*muNVal(5,14)+bCnst{2}(k,3,2)*muNVal(5,15);
        G33(fcni5,1)=bCnst{3}(k,3,1)*muNVal(5,14)+bCnst{3}(k,3,2)*muNVal(5,15);
        G34(fcni5,1)=bCnst{4}(k,3,1)*muNVal(5,16)+bCnst{4}(k,3,2)*muNVal(5,17)+bCnst{4}(k...
```

```
                    ,3,3)*muNVal(5,18);
                G35(fcni5,1)=bCnst{5}(k,3,1)*muNVal(5,16)+bCnst{5}(k,3,2)*muNVal(5,17)+bCnst{5}(k...
                    ,3,3)*muNVal(5,18);
                G36(fcni5,1)=bCnst{6}(k,3,1)*muNVal(5,16)+bCnst{6}(k,3,2)*muNVal(5,17)+bCnst{6}(k...
                    ,3,3)*muNVal(5,18);
            end
        end % ends loop over triangles
    %%% Superimpose solutions:
        %%% mu1 %%%
        muSol(:,i)=phiSoln*G31;
        muBC(:,i)=G31;
        %%% mu2 %%%
        muSol(:,i+Nfvn)=phiSoln*G32;
        muBC(:,i+Nfvn)=G32;
        %%% mu3 %%%
        muSol(:,i+2*Nfvn)=phiSoln*G33;
        muBC(:,i+2*Nfvn)=G33;
        %%% mu4 %%%
        muSol(:,i+3*Nfvn)=phiSoln*G34;
        muBC(:,i+3*Nfvn)=G34;
        %%% mu5 %%%
        muSol(:,i+4*Nfvn)=phiSoln*G35;
        muBC(:,i+4*Nfvn)=G35;
        %%% mu6 %%%
        muSol(:,i+5*Nfvn)=phiSoln*G36;
        muBC(:,i+5*Nfvn)=G36;
    end
        %%% mu7 %%%
        % Note that mu7=0 at all 6 nodes, so this solution is all zeros
    for i=1:Nbxxn
        G38=zeros(Nih,1);
        pni=pMesh.BxxNodePtrs(i,1);
        for k=1:pMesh.nbTriangles6
            if pMesh.TRIANGLES6(k,1)==pni
                %%% Non-zero on nodes 1,4,6
                pni4=pMesh.TRIANGLES6(k,4);
                fni4=pMesh.PtrsTo3DNodes(pni4,1);
                fcni4=Mesh.NodePtrs(fni4,3);
                if fcni4>0
                    G38(fcni4,1)=bCnst{4}(k,1,1)*muNVal(4,4)+bCnst{4}(k,1,2)*muNVal(4,5)+bCnst{4}(...
                        k,1,3)*muNVal(4,6);
```

```
        end
        pni6=pMesh.TRIANGLES6(k,6);
        fni6=pMesh.PtrsTo3DNodes(pni6,1);
        fcni6=Mesh.NodePtrs(fni6,3);
        if fcni6>0
            G38(fcni6,1)=bCnst{4}(k,1,1)*muNVal(6,4)+bCnst{4}(k,1,2)*muNVal(6,5)+bCnst{4}(...
                k,1,3)*muNVal(6,6);
        end
    elseif pMesh.TRIANGLES6(k,2)==pni
        %%% Non-zero on nodes 2,5,4
        pni5=pMesh.TRIANGLES6(k,5);
        fni5=pMesh.PtrsTo3DNodes(pni5,1);
        fcni5=Mesh.NodePtrs(fni5,3);
        if fcni5>0
            G38(fcni5,1)=bCnst{4}(k,2,1)*muNVal(5,10)+bCnst{4}(k,2,2)*muNVal(5,11)+bCnst...
                {4}(k,2,3)*muNVal(5,12);
        end
        pni4=pMesh.TRIANGLES6(k,4);
        fni4=pMesh.PtrsTo3DNodes(pni4,1);
        fcni4=Mesh.NodePtrs(fni4,3);
        if fcni4>0
            G38(fcni4,1)=bCnst{4}(k,2,1)*muNVal(4,10)+bCnst{4}(k,2,2)*muNVal(4,11)+bCnst...
                {4}(k,2,3)*muNVal(4,12);
        end
    elseif pMesh.TRIANGLES6(k,3)==pni
        %%% Non-zero on nodes 3,6,5
        pni6=pMesh.TRIANGLES6(k,6);
        fni6=pMesh.PtrsTo3DNodes(pni6,1);
        fcni6=Mesh.NodePtrs(fni6,3);
        if fcni6>0
            G38(fcni6,1)=bCnst{4}(k,3,1)*muNVal(6,16)+bCnst{4}(k,3,2)*muNVal(6,17)+bCnst...
                {4}(k,3,3)*muNVal(6,18);
        end
        pni5=pMesh.TRIANGLES6(k,5);
        fni5=pMesh.PtrsTo3DNodes(pni5,1);
        fcni5=Mesh.NodePtrs(fni5,3);
        if fcni5>0
            G38(fcni5,1)=bCnst{4}(k,3,1)*muNVal(5,16)+bCnst{4}(k,3,2)*muNVal(5,17)+bCnst...
                {4}(k,3,3)*muNVal(5,18);
        end
    end
```

```
    end % ends loop over triangles
    %%% mu8 %%%
    muSol(:,i+6*Nfvn+Nfmn)=phiSoln*G38;
    muBC(:,i+6*Nfvn+Nfmn)=G38;
end

for i=1:Nbyyn
    G39=zeros(Nih,1);
    pni=pMesh.ByyNodePtrs(i,1);
    for k=1:pMesh.nbTriangles6
        if pMesh.TRIANGLES6(k,1)==pni
            %%% Non-zero on nodes 1,4,6
            pni4=pMesh.TRIANGLES6(k,4);
            fni4=pMesh.PtrsTo3DNodes(pni4,1);
            fcni4=Mesh.NodePtrs(fni4,3);
            if fcni4>0
                G39(fcni4,1)=bCnst{6}(k,1,1)*muNVal(4,4)+bCnst{6}(k,1,2)*muNVal(4,5)+bCnst{6}(...
                    k,1,3)*muNVal(4,6);
            end
            pni6=pMesh.TRIANGLES6(k,6);
            fni6=pMesh.PtrsTo3DNodes(pni6,1);
            fcni6=Mesh.NodePtrs(fni6,3);
            if fcni6>0
                G39(fcni6,1)=bCnst{6}(k,1,1)*muNVal(6,4)+bCnst{6}(k,1,2)*muNVal(6,5)+bCnst{6}(...
                    k,1,3)*muNVal(6,6);
            end
        elseif pMesh.TRIANGLES6(k,2)==pni
            %%% Non-zero on nodes 2,5,4
            pni5=pMesh.TRIANGLES6(k,5);
            fni5=pMesh.PtrsTo3DNodes(pni5,1);
            fcni5=Mesh.NodePtrs(fni5,3);
            if fcni5>0
                G39(fcni5,1)=bCnst{6}(k,2,1)*muNVal(5,10)+bCnst{6}(k,2,2)*muNVal(5,11)+bCnst...
                    {6}(k,2,3)*muNVal(5,12);
            end
            pni4=pMesh.TRIANGLES6(k,4);
            fni4=pMesh.PtrsTo3DNodes(pni4,1);
            fcni4=Mesh.NodePtrs(fni4,3);
            if fcni4>0
                G39(fcni4,1)=bCnst{6}(k,2,1)*muNVal(4,10)+bCnst{6}(k,2,2)*muNVal(4,11)+bCnst...
                    {6}(k,2,3)*muNVal(4,12);
```

```
            end
        elseif pMesh.TRIANGLES6(k,3)==pni
            %%% Non-zero on nodes 3,6,5
            pni6=pMesh.TRIANGLES6(k,6);
            fni6=pMesh.PtrsTo3DNodes(pni6,1);
            fcni6=Mesh.NodePtrs(fni6,3);
            if fcni6>0
                G39(fcni6,1)=bCnst{6}(k,3,1)*muNVal(6,16)+bCnst{6}(k,3,2)*muNVal(6,17)+bCnst...
                    {6}(k,3,3)*muNVal(6,18);
            end
            pni5=pMesh.TRIANGLES6(k,5);
            fni5=pMesh.PtrsTo3DNodes(pni5,1);
            fcni5=Mesh.NodePtrs(fni5,3);
            if fcni5>0
                G39(fcni5,1)=bCnst{6}(k,3,1)*muNVal(5,16)+bCnst{6}(k,3,2)*muNVal(5,17)+bCnst...
                    {6}(k,3,3)*muNVal(5,18);
            end
        end
    end % ends loop over triangles
    %%% mu9 %%%
    muSol(:,i+6*Nfvn+Nfmn+Nbxxn)=phiSoln*G39;
    muBC(:,i+6*Nfvn+Nfmn+Nbxxn)=G39;
end
%%% Pressure Solution
muSolP=muSol(3*Nf+1:3*Nf+Np,:);
%%% Fluid Solution
muSol=muSol(1:3*Nf,:);
muSol=[muSol;muBC];
```

BLFormA2D.m

```
function B=BLFormA2D(pMesh,bCnst,muVal,qwt)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  This function creates the row of the bilinear form
%%%  corresponding to the term:
%%%  b(phi,r) = -r \int_\Omega phi dOmega
%%%  The row vector is B
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Initialize Stiffness Block Matrices
```

```
Nfvn=length(pMesh.FVNodePtrs(:,1));
Nfmn=length(pMesh.FMNodePtrs(:,1));
Nbxxn=length(pMesh.BxxNodePtrs(:,1));
Nbyyn=length(pMesh.ByyNodePtrs(:,1));
B1=zeros(1,Nfvn);
B2=zeros(1,Nfvn);
B3=zeros(1,Nfvn);
B4=zeros(1,Nfvn);
B5=zeros(1,Nfvn);
B6=zeros(1,Nfvn);
B7=zeros(1,Nfmn);
B8=zeros(1,Nbxxn);
B9=zeros(1,Nbyyn);
%%% Loop over triangles in the mesh
for k=1:pMesh.nbTriangles6
%%% Get global node numbers for the nodes of Element k
    nodes=pMesh.TRIANGLES6(k,1:6);
%%% (x,y) coordinates of (local) nodes 1, 2, 3 (Vertices of T_k)
    x1=pMesh.POS(nodes(1),1);
    y1=pMesh.POS(nodes(1),2);
    x2=pMesh.POS(nodes(2),1);
    y2=pMesh.POS(nodes(2),2);
    x3=pMesh.POS(nodes(3),1);
    y3=pMesh.POS(nodes(3),2);
%%% Compute transormation J: [x,y] = [x1, y1] + Ju for u in reference
%%% triangle, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1; y2-y1,y3-y1];
    detJk=det(Jk);
    for n1=1:3 % Loop over 1st vertex nodes
        if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
            gn1=nodes(n1); % Global Node Number
            vfn1=pMesh.NodePtrs(gn1); % 1st Vertex Free Node Number
        %%% Compute mu_1 at quadrature points(stored in a row vector)
            mu1=muVal(:,1+6*(n1-1))';
            mu19=muVal(:,19)';
            mu20=muVal(:,20)';
            mu21=muVal(:,21)';
            b1=bCnst{1}(k,n1,1);
            b2=bCnst{1}(k,n1,2);
            b3=bCnst{1}(k,n1,3);
            mu_1=mu1+b1*mu19+b2*mu20+b3*mu21;
```

```
%%% Compute mu_2 at quadrature points(stored in a row vector)
    mu2=muVal(:,2+6*(n1-1))';
    mu3=muVal(:,3+6*(n1-1))';
    mu19=muVal(:,19)';
    mu20=muVal(:,20)';
    mu21=muVal(:,21)';
    a1=bCnst{2}(k,n1,1);
    a2=bCnst{2}(k,n1,2);
    b1=bCnst{2}(k,n1,3);
    b2=bCnst{2}(k,n1,4);
    b3=bCnst{2}(k,n1,5);
    mu_2=a1*mu2+a2*mu3+b1*mu19+b2*mu20+b3*mu21;
%%% Compute mu_3 at quadrature points(stored in a row vector)
    mu2=muVal(:,2+6*(n1-1))';
    mu3=muVal(:,3+6*(n1-1))';
    mu19=muVal(:,19)';
    mu20=muVal(:,20)';
    mu21=muVal(:,21)';
    a1=bCnst{3}(k,n1,1);
    a2=bCnst{3}(k,n1,2);
    b1=bCnst{3}(k,n1,3);
    b2=bCnst{3}(k,n1,4);
    b3=bCnst{3}(k,n1,5);
    mu_3=a1*mu2+a2*mu3+b1*mu19+b2*mu20+b3*mu21;
%%% Compute mu_4 at quadrature points(stored in a row vector)
    mu4=muVal(:,4+6*(n1-1))';
    mu5=muVal(:,5+6*(n1-1))';
    mu6=muVal(:,6+6*(n1-1))';
    mu19=muVal(:,19)';
    mu20=muVal(:,20)';
    mu21=muVal(:,21)';
    a1=bCnst{4}(k,n1,1);
    a2=bCnst{4}(k,n1,2);
    a3=bCnst{4}(k,n1,3);
    b1=bCnst{4}(k,n1,4);
    b2=bCnst{4}(k,n1,5);
    b3=bCnst{4}(k,n1,6);
    mu_4=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;
%%% Compute mu_5 at quadrature points(stored in a row vector)
    mu4=muVal(:,4+6*(n1-1))';
    mu5=muVal(:,5+6*(n1-1))';
```

```
        mu6=muVal(:,6+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{5}(k,n1,1);
        a2=bCnst{5}(k,n1,2);
        a3=bCnst{5}(k,n1,3);
        b1=bCnst{5}(k,n1,4);
        b2=bCnst{5}(k,n1,5);
        b3=bCnst{5}(k,n1,6);
        mu_5=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;
    %%% Compute mu_6 at quadrature points(stored in a row vector)
        mu4=muVal(:,4+6*(n1-1))';
        mu5=muVal(:,5+6*(n1-1))';
        mu6=muVal(:,6+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{6}(k,n1,1);
        a2=bCnst{6}(k,n1,2);
        a3=bCnst{6}(k,n1,3);
        b1=bCnst{6}(k,n1,4);
        b2=bCnst{6}(k,n1,5);
        b3=bCnst{6}(k,n1,6);
        mu_6=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;
    %%% Int_Tk (mu_1)*(1) Using Quadrature
        qVal=detJk*((mu_1)*qwt);
        B1(1,vfn1)=B1(1,vfn1)-qVal;
    %%% Int_Tk (mu_1)*(1) Using Quadrature
        qVal=detJk*((mu_2)*qwt);
        B2(1,vfn1)=B2(1,vfn1)-qVal;
    %%% Int_Tk (mu_1)*(1) Using Quadrature
        qVal=detJk*((mu_3)*qwt);
        B3(1,vfn1)=B3(1,vfn1)-qVal;
    %%% Int_Tk (mu_1)*(1) Using Quadrature
        qVal=detJk*((mu_4)*qwt);
        B4(1,vfn1)=B4(1,vfn1)-qVal;
    %%% Int_Tk (mu_1)*(1) Using Quadrature
        qVal=detJk*((mu_5)*qwt);
        B5(1,vfn1)=B5(1,vfn1)-qVal;
    %%% Int_Tk (mu_1)*(1) Using Quadrature
```

```
        qVal=detJk*((mu_6)*qwt);
        B6(1,vfn1)=B6(1,vfn1)-qVal;
    elseif pMesh.NodePtrs(nodes(n1),3)>0 % Boundary d^2/dx^2 Node
        gn1=nodes(n1); % Global Node Number
        vbxxn1=pMesh.NodePtrs(gn1,3); % Boundary d^2/dx^2 Node Number
        %%% Compute mu_8 at quadrature points(stored in a row vector)
        mu4=muVal(:,4+6*(n1-1))';
        mu5=muVal(:,5+6*(n1-1))';
        mu6=muVal(:,6+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{4}(k,n1,1);
        a2=bCnst{4}(k,n1,2);
        a3=bCnst{4}(k,n1,3);
        b1=bCnst{4}(k,n1,4);
        b2=bCnst{4}(k,n1,5);
        b3=bCnst{4}(k,n1,6);
        mu_8=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;
    %%% Int_Tk (mu_8)*(1) Using Quadrature
        qVal=detJk*((mu_8)*qwt);
        B8(1,vbxxn1)=B8(1,vbxxn1)-qVal;
    elseif pMesh.NodePtrs(nodes(n1),3)<0 % Boundary d^2/dy^2 Node
        gn1=nodes(n1); % Global Node Number
        vbyyn1=-pMesh.NodePtrs(gn1,3); % Boundary d^2/dy^2 Node Number
        %%% Compute mu_9 at quadrature points(stored in a row vector)
        mu4=muVal(:,4+6*(n1-1))';
        mu5=muVal(:,5+6*(n1-1))';
        mu6=muVal(:,6+6*(n1-1))';
        mu19=muVal(:,19)';
        mu20=muVal(:,20)';
        mu21=muVal(:,21)';
        a1=bCnst{6}(k,n1,1);
        a2=bCnst{6}(k,n1,2);
        a3=bCnst{6}(k,n1,3);
        b1=bCnst{6}(k,n1,4);
        b2=bCnst{6}(k,n1,5);
        b3=bCnst{6}(k,n1,6);
        mu_9=a1*mu4+a2*mu5+a3*mu6+b1*mu19+b2*mu20+b3*mu21;
    %%% Int_Tk (mu_9)*(1) Using Quadrature
        qVal=detJk*((mu_9)*qwt);
```

```
            B9(1,vbyyn1)=B9(1,vbyyn1)-qVal;
        end %end if loop for vertex 1
    end %end loop over vertex 1

    for n1=4:6 % Loop over midpoint nodes
        if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
            gn1=nodes(n1); % Global Node Number
            mfn1=pMesh.NodePtrs(gn1); % 1st Midpoint Free Node Number
        %%% Compute mu_7 at quadrature points(stored in a row vector)
        %%% Note that only one of b1-b3 should be non-zero
            mu19=muVal(:,19)';
            mu20=muVal(:,20)';
            mu21=muVal(:,21)';
            b1=bCnst{7}(k,n1-3,1); % Note n1-3 to get 1,2,3 from 4,5,6
            b2=bCnst{7}(k,n1-3,2);
            b3=bCnst{7}(k,n1-3,3);
            mu_7=b1*mu19+b2*mu20+b3*mu21;
        %%% Int_Tk (mu_7)*(1) Using Quadrature
            qVal=detJk*((mu_7)*qwt);
            B7(1,mfn1)=B7(1,mfn1)-qVal;
        end
    end
end %end loop over triangles
B=[B1,B2,B3,B4,B5,B6,B7,B8,B9];
```

Load1A2D.m

```
function F=Load1A2D(pMesh,bCnst,f,muVal,qpt,qwt)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  This function creates the load vector for the plate free nodes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nfvn=length(pMesh.FVNodePtrs(:,1));  % The number of free vertex nodes
Nfmn=length(pMesh.FMNodePtrs(:,1));  % The number of free midpoint nodes
Nbxxn=length(pMesh.BxxNodePtrs(:,1));
Nbyyn=length(pMesh.ByyNodePtrs(:,1));
F1=zeros(Nfvn,1); % mu1,7,13
F2=zeros(Nfvn,1); % mu2,8,14
F3=zeros(Nfvn,1); % mu3,9,15
F4=zeros(Nfvn,1); % mu4,10,16
```

```
F5=zeros(Nfvn,1); % mu5,11,17
F6=zeros(Nfvn,1); % mu6,12,18
F7=zeros(Nfmn,1); % mu19,20,21
F8=zeros(Nbxxn,1); % mu4,10,16
F9=zeros(Nbyyn,1); % mu6,12,18
ql=length(qwt(:,1));
%%% Loop over triangular elements
for k=1:pMesh.nbTriangles6
%%% Get global node numbers for the nodes of Element k
    nodes=pMesh.TRIANGLES6(k,1:6);
%%% (x,y) coordinates of (local) nodes 1, 2, 3 (Vertices of T_k)
    x1=pMesh.POS(nodes(1),1);
    y1=pMesh.POS(nodes(1),2);
    x2=pMesh.POS(nodes(2),1);
    y2=pMesh.POS(nodes(2),2);
    x3=pMesh.POS(nodes(3),1);
    y3=pMesh.POS(nodes(3),2);
%%% Compute transormation J: [x,y] = [x1, y1] + Ju for u in ref. triangle
    Jk=[x2-x1,x3-x1; y2-y1,y3-y1];
    detJk=det(Jk);
%%% Get the (x,y) coordinates of the transformed quadrature points
    Tqpt=zeros(2,ql);
    for j=1:ql
        Tqpt(1,j)=x1+Jk(1,:)*qpt(:,j); % transformed x
        Tqpt(2,j)=y1+Jk(2,:)*qpt(:,j); % transformed y
    end
%%% Evaluate the function f at the quadrature points:
    fVal=zeros(ql,1);
    for i=1:ql
        fVal(i,1)=f(Tqpt(1,i),Tqpt(2,i));
    end
%%% Create loop for mu1,mu7,mu13
    for n1=1:3
        if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
            gn1=nodes(n1); % Global Node Number
            vfn1=pMesh.NodePtrs(gn1); % Vertex Free Node Number
        %%% Compute mu_1 at quadrature points(stored in a column vector)
            Mu1=muVal(:,1+(n1-1)*6); % values of mu1, mu7 or mu13
            Mu19=muVal(:,19);
            Mu20=muVal(:,20);
            Mu21=muVal(:,21);
```

```
        b1=bCnst{1}(k,n1,1);
        b2=bCnst{1}(k,n1,2);
        b3=bCnst{1}(k,n1,3);
        Mu_1=Mu1+b1*Mu19+b2*Mu20+b3*Mu21;
%%% Compute Int_Tk mu_1*f and add the contribution to F1(vfn1,1)
        qVal=detJk*sum(Mu_1.*fVal.*qwt);
        F1(vfn1,1)=F1(vfn1,1)+qVal;
%%% Compute mu_2 at quadrature points(stored in a column vector)
        Mu2=muVal(:,2+(n1-1)*6); % values of mu2, mu8 or mu14
        Mu3=muVal(:,3+(n1-1)*6); % values of mu3, mu9 or mu15
        Mu19=muVal(:,19);
        Mu20=muVal(:,20);
        Mu21=muVal(:,21);
        a1=bCnst{2}(k,n1,1);
        a2=bCnst{2}(k,n1,2);
        b1=bCnst{2}(k,n1,3);
        b2=bCnst{2}(k,n1,4);
        b3=bCnst{2}(k,n1,5);
        Mu_2=a1*Mu2+a2*Mu3+b1*Mu19+b2*Mu20+b3*Mu21;
%%% Compute Int_Tk mu_2*f and add the contribution to F2(vfn1,1)
        qVal=detJk*sum(Mu_2.*fVal.*qwt);
        F2(vfn1,1)=F2(vfn1,1)+qVal;
%%% Compute mu_3 at quadrature points(stored in a column vector)
        Mu2=muVal(:,2+(n1-1)*6); % values of mu2, mu8 or mu14
        Mu3=muVal(:,3+(n1-1)*6); % values of mu3, mu9 or mu15
        Mu19=muVal(:,19);
        Mu20=muVal(:,20);
        Mu21=muVal(:,21);
        a1=bCnst{3}(k,n1,1);
        a2=bCnst{3}(k,n1,2);
        b1=bCnst{3}(k,n1,3);
        b2=bCnst{3}(k,n1,4);
        b3=bCnst{3}(k,n1,5);
        Mu_3=a1*Mu2+a2*Mu3+b1*Mu19+b2*Mu20+b3*Mu21;
%%% Compute Int_Tk mu_3*f and add the contribution to F3(vfn1,1)
        qVal=detJk*sum(Mu_3.*fVal.*qwt);
        F3(vfn1,1)=F3(vfn1,1)+qVal;
%%% Compute mu_4 at quadrature points(stored in a column vector)
        Mu4=muVal(:,4+(n1-1)*6); % values of mu4, mu10 or mu16
        Mu5=muVal(:,5+(n1-1)*6); % values of mu5, mu11 or mu17
        Mu6=muVal(:,6+(n1-1)*6); % values of mu6, mu12 or mu18
```

```
Mu19=muVal(:,19);
Mu20=muVal(:,20);
Mu21=muVal(:,21);
a1=bCnst{4}(k,n1,1);
a2=bCnst{4}(k,n1,2);
a3=bCnst{4}(k,n1,3);
b1=bCnst{4}(k,n1,4);
b2=bCnst{4}(k,n1,5);
b3=bCnst{4}(k,n1,6);
Mu_4=a1*Mu4+a2*Mu5+a3*Mu6+b1*Mu19+b2*Mu20+b3*Mu21;
%%% Compute Int_Tk mu_4*f and add the contribution to F4(vfn1,1)
qVal=detJk*sum(Mu_4.*fVal.*qwt);
F4(vfn1,1)=F4(vfn1,1)+qVal;
%%% Compute mu_5 at quadrature points(stored in a column vector)
Mu4=muVal(:,4+(n1-1)*6); % values of mu4, mu10 or mu16
Mu5=muVal(:,5+(n1-1)*6); % values of mu5, mu11 or mu17
Mu6=muVal(:,6+(n1-1)*6); % values of mu6, mu12 or mu18
Mu19=muVal(:,19);
Mu20=muVal(:,20);
Mu21=muVal(:,21);
a1=bCnst{5}(k,n1,1);
a2=bCnst{5}(k,n1,2);
a3=bCnst{5}(k,n1,3);
b1=bCnst{5}(k,n1,4);
b2=bCnst{5}(k,n1,5);
b3=bCnst{5}(k,n1,6);
Mu_5=a1*Mu4+a2*Mu5+a3*Mu6+b1*Mu19+b2*Mu20+b3*Mu21;
%%% Compute Int_Tk mu_5*f and add the contribution to F5(vfn1,1)
qVal=detJk*sum(Mu_5.*fVal.*qwt);
F5(vfn1,1)=F5(vfn1,1)+qVal;
%%% Compute mu_6 at quadrature points(stored in a column vector)
Mu4=muVal(:,4+(n1-1)*6); % values of mu4, mu10 or mu16
Mu5=muVal(:,5+(n1-1)*6); % values of mu5, mu11 or mu17
Mu6=muVal(:,6+(n1-1)*6); % values of mu6, mu12 or mu18
Mu19=muVal(:,19);
Mu20=muVal(:,20);
Mu21=muVal(:,21);
a1=bCnst{6}(k,n1,1);
a2=bCnst{6}(k,n1,2);
a3=bCnst{6}(k,n1,3);
b1=bCnst{6}(k,n1,4);
```

```
            b2=bCnst{6}(k,n1,5);
            b3=bCnst{6}(k,n1,6);
            Mu_6=a1*Mu4+a2*Mu5+a3*Mu6+b1*Mu19+b2*Mu20+b3*Mu21;
        %%% Compute Int_Tk mu_6*f and add the contribution to F6(vfn1,1)
            qVal=detJk*sum(Mu_6.*fVal.*qwt);
            F6(vfn1,1)=F6(vfn1,1)+qVal;
        end
    end
%%% Create loop for mu19,mu20,mu21
    for n1=4:6
        if pMesh.POS(nodes(n1),4)==-10  % Interior Plate Node
            gn1=nodes(n1); % Global Node Number
            mfn1=pMesh.NodePtrs(gn1); % Midpoint Free Node Number
        %%% Compute mu_19,20,21 at quadrature points(stored in a column vector)
            Mu19=muVal(:,19);
            Mu20=muVal(:,20);
            Mu21=muVal(:,21);
            b1=bCnst{7}(k,n1-3,1); % Note n1-3 to get 1,2,3 from 4,5,6
            b2=bCnst{7}(k,n1-3,2);
            b3=bCnst{7}(k,n1-3,3);
            Mu_19=b1*Mu19+b2*Mu20+b3*Mu21;
        %%% Compute Int_Tk mu_19*f and add the contribution to F7(mfn1,1)
            qVal=detJk*sum(Mu_19.*fVal.*qwt);
            F7(mfn1,1)=F7(mfn1,1)+qVal;
        end
    end
%%% Loops for Boundary Nodes
%%% Create loop for mu4,mu10,mu16
    for n1=1:3
        if pMesh.NodePtrs(nodes(n1),3)>0 % Boundary d^2/dx^2 Node
            gn1=nodes(n1); % Global Node Number
            vbxxn1=pMesh.NodePtrs(gn1,3); % Boundary d^2/dx^2 Node Number
        %%% Compute mu_4 at quadrature points(stored in a column vector)
            Mu4=muVal(:,4+(n1-1)*6); % values of mu4, mu10 or mu16
            Mu5=muVal(:,5+(n1-1)*6); % values of mu5, mu11 or mu17
            Mu6=muVal(:,6+(n1-1)*6); % values of mu6, mu12 or mu18
            Mu19=muVal(:,19);
            Mu20=muVal(:,20);
            Mu21=muVal(:,21);
            a1=bCnst{4}(k,n1,1);
            a2=bCnst{4}(k,n1,2);
```

```
            a3=bCnst{4}(k,n1,3);
            b1=bCnst{4}(k,n1,4);
            b2=bCnst{4}(k,n1,5);
            b3=bCnst{4}(k,n1,6);
            Mu_4=a1*Mu4+a2*Mu5+a3*Mu6+b1*Mu19+b2*Mu20+b3*Mu21;
        %%% Compute Int_Tk mu_4*f and add the contribution to F4(vfn1,1)
            qVal=detJk*sum(Mu_4.*fVal.*qwt);
            F8(vbxxn1,1)=F8(vbxxn1,1)+qVal;
        end
    end
%%% Create loop for mu6,mu12,mu18
    for n1=1:3
        if pMesh.NodePtrs(nodes(n1),3)<0 % Boundary d^2/dy^2 Node
            gn1=nodes(n1); % Global Node Number
            vbyyn1=-pMesh.NodePtrs(gn1,3); % Boundary d^2/dy^2 Node Number
        %%% Compute mu_6 at quadrature points(stored in a column vector)
            Mu4=muVal(:,4+(n1-1)*6); % values of mu4, mu10 or mu16
            Mu5=muVal(:,5+(n1-1)*6); % values of mu5, mu11 or mu17
            Mu6=muVal(:,6+(n1-1)*6); % values of mu6, mu12 or mu18
            Mu19=muVal(:,19);
            Mu20=muVal(:,20);
            Mu21=muVal(:,21);
            a1=bCnst{6}(k,n1,1);
            a2=bCnst{6}(k,n1,2);
            a3=bCnst{6}(k,n1,3);
            b1=bCnst{6}(k,n1,4);
            b2=bCnst{6}(k,n1,5);
            b3=bCnst{6}(k,n1,6);
            Mu_6=a1*Mu4+a2*Mu5+a3*Mu6+b1*Mu19+b2*Mu20+b3*Mu21;
        %%% Compute Int_Tk mu_6*f and add the contribution to F6(vfn1,1)
            qVal=detJk*sum(Mu_6.*fVal.*qwt);
            F9(vbyyn1,1)=F9(vbyyn1,1)+qVal;
        end
    end
end % Ends loop over triangles
%%% Create load vector from its components
F=[F1;F2;F3;F4;F5;F6;F7;F8;F9];
```

Load2A2D.m

```
function F=Load2A2D(pMesh,w1Soln,muSoln,fTildeA)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  This function creates the part of the load vector corresponding to the
%%%  terms:
%%%  (grad f~(w1*), grad f~(mu))_O + lambda(f~(w1*), f~(mu))_O
%%%
%%%  pMesh is the plate mesh data structure, w1Soln is the fluid
%%%  coefficients for f~(w1*), muSoln is the fluid coefficients for f~(mu),
%%%  and the load vector is F.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nfvn=length(pMesh.FVNodePtrs(:,1));  % The number of free vertex nodes
Nfmn=length(pMesh.FMNodePtrs(:,1));  % The number of free midpoint nodes
Nbxxn=length(pMesh.BxxNodePtrs(:,1));
Nbyyn=length(pMesh.ByyNodePtrs(:,1));
F1=zeros(Nfvn,1); % mu1,7,13
F2=zeros(Nfvn,1); % mu2,8,14
F3=zeros(Nfvn,1); % mu3,9,15
F4=zeros(Nfvn,1); % mu4,10,16
F5=zeros(Nfvn,1); % mu5,11,17
F6=zeros(Nfvn,1); % mu6,12,18
F7=zeros(Nfmn,1); % mu19,20,21
F8=zeros(Nbxxn,1); % mu4,10,16
F9=zeros(Nbyyn,1); % mu6,12,18
for i=1:Nfvn
    F1(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,i));
    F2(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,Nfvn+i));
    F3(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,2*Nfvn+i));
    F4(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,3*Nfvn+i));
    F5(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,4*Nfvn+i));
    F6(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,5*Nfvn+i));
end
for i=1:Nfmn
    F7(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,6*Nfvn+i));
end
for i=1:Nbxxn
    F8(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+i));
end
for i=1:Nbyyn
    F9(i,1)=transpose(w1Soln(:,1))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+i));
end
```

```
%%% Create load vector from its components
F=[F1;F2;F3;F4;F5;F6;F7;F8;F9];
```

Load3A2D.m

```
function F=Load3A2D(pMesh,uSoln,muSoln,fTildeA)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  This function creates the part of the load vector corresponding to the
%%%  terms:
%%%  (grad mu~(u*), grad f~(mu))_O + lambda(mu~(u*), f~(mu))_O
%%%
%%%  pMesh is the plate mesh data structure, uSoln is the fluid
%%%  coefficients for mu~(u*), muSoln is the fluid coefficients for f~(mu),
%%%  and the load vector is F.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nfvn=length(pMesh.FVNodePtrs(:,1));  % The number of free vertex nodes
Nfmn=length(pMesh.FMNodePtrs(:,1));  % The number of free midpoint nodes
Nbxxn=length(pMesh.BxxNodePtrs(:,1));% The number of boundary dxx nodes
Nbyyn=length(pMesh.ByyNodePtrs(:,1));% The number of boundary dyy nodes
F1=zeros(Nfvn,1); % mu1,7,13
F2=zeros(Nfvn,1); % mu2,8,14
F3=zeros(Nfvn,1); % mu3,9,15
F4=zeros(Nfvn,1); % mu4,10,16
F5=zeros(Nfvn,1); % mu5,11,17
F6=zeros(Nfvn,1); % mu6,12,18
F7=zeros(Nfmn,1); % mu19,20,21
F8=zeros(Nbxxn,1); % mu4,10,16
F9=zeros(Nbyyn,1); % mu6,12,18
for i=1:Nfvn
    F1(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,i));
    F2(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,Nfvn+i));
    F3(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,2*Nfvn+i));
    F4(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,3*Nfvn+i));
    F5(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,4*Nfvn+i));
    F6(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,5*Nfvn+i));
end
for i=1:Nfmn
    F7(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,6*Nfvn+i));
end
```

```
for i=1:Nbxxn
    F8(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+i));
end
for i=1:Nbyyn
    F9(i,1)=transpose(uSoln(:,1))*(fTildeA*muSoln(:,6*Nfvn+Nfmn+Nbxxn+i));
end
%%% Create load vector from its components
F=[F1;F2;F3;F4;F5;F6;F7;F8;F9];
```

Load4A2D.m

```
function F=Load4A2D(Mesh,pMesh,ustar,muSoln,phiVal,qpt,qwt)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  This function creates the part of the load vector corresponding to the
%%%  term:
%%%  (u*, f~(mu))_O
%%%
%%%  pMesh is the plate mesh data structure, ustar is the fluid forcing
%%%  function u*, muSoln is the fluid coefficients for f~(mu),
%%%  and the load vector is F.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nf=length(Mesh.FNodePtrs(:,1)); % The number of free fluid nodes
Nih=length(Mesh.IHNodePtrs(:,1)); % The number of inhomogeneous fluid nodes
Nfvn=length(pMesh.FVNodePtrs(:,1)); % The number of free vertex nodes
Nfmn=length(pMesh.FMNodePtrs(:,1)); % The number of free midpoint nodes
Nbxxn=length(pMesh.BxxNodePtrs(:,1)); % The number of boundary dxx nodes
Nbyyn=length(pMesh.ByyNodePtrs(:,1)); % The number of boundary dyy nodes
F1=zeros(Nfvn,1); % mu1,7,13
F2=zeros(Nfvn,1); % mu2,8,14
F3=zeros(Nfvn,1); % mu3,9,15
F4=zeros(Nfvn,1); % mu4,10,16
F5=zeros(Nfvn,1); % mu5,11,17
F6=zeros(Nfvn,1); % mu6,12,18
F7=zeros(Nfmn,1); % mu19,20,21
F8=zeros(Nbxxn,1); % mu4,10,16
F9=zeros(Nbyyn,1); % mu6,12,18
ql=length(qwt);
%%% Compute the values of the muSoln at the global nodes
ll=Mesh.nbNod;
```

```
muSol1=zeros(ll,6*Nfvn+Nfmn+Nbxxn+Nbyyn);
muSol2=zeros(ll,6*Nfvn+Nfmn+Nbxxn+Nbyyn);
muSol3=zeros(ll,6*Nfvn+Nfmn+Nbxxn+Nbyyn);
for j=1:6*Nfvn+Nfmn+Nbxxn+Nbyyn
    muSol1(Mesh.FNodePtrs,j)=muSoln(1:Nf,j);
    muSol2(Mesh.FNodePtrs,j)=muSoln(1+Nf:2*Nf,j);
    muSol3(Mesh.FNodePtrs,j)=muSoln(1+2*Nf:3*Nf,j);
    muSol3(Mesh.IHNodePtrs,j)=muSoln(1+3*Nf:3*Nf+Nih,j);
end
u1Val=zeros(ql,1);
u2Val=zeros(ql,1);
u3Val=zeros(ql,1);
for k = 1:Mesh.nbTets10;  % Loop and sum over all elements
%%% Get Global Node numbers for nodes of Element k
    nodes=Mesh.TETS10(k,1:10);
%%% (x,y,z) coordinates of (local) nodes 1, 2, 3, 4 (Vertices of T_k)
    x1=Mesh.POS(nodes(1),1);
    y1=Mesh.POS(nodes(1),2);
    z1=Mesh.POS(nodes(1),3);
    x2=Mesh.POS(nodes(2),1);
    y2=Mesh.POS(nodes(2),2);
    z2=Mesh.POS(nodes(2),3);
    x3=Mesh.POS(nodes(3),1);
    y3=Mesh.POS(nodes(3),2);
    z3=Mesh.POS(nodes(3),3);
    x4=Mesh.POS(nodes(4),1);
    y4=Mesh.POS(nodes(4),2);
    z4=Mesh.POS(nodes(4),3);
% Compute transormation J: [x,y,z] = [x1, y1, z1] + Ju for u in reference
% tetrahedron, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1,x4-x1;y2-y1,y3-y1,y4-y1;z2-z1,z3-z1,z4-z1];
    detJk=abs(det(Jk));
%%% Get the (x,y,z) coordinates of the transformed quadrature points
    Tqpt=[x1,0,0; 0,y1,0; 0,0,z1]*ones(3,ql)+Jk*qpt;
%%% Compute the values of u* at the quadrature points
    for j=1:ql
        u1Val(j,1)=ustar{1}(Tqpt(1,j),Tqpt(2,j),Tqpt(3,j));
        u2Val(j,1)=ustar{2}(Tqpt(1,j),Tqpt(2,j),Tqpt(3,j));
        u3Val(j,1)=ustar{3}(Tqpt(1,j),Tqpt(2,j),Tqpt(3,j));
    end
    for i=1:Nfvn
```

```
            F1(i,1)=F1(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i)));
            F1(i,1)=F1(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i)));
            F1(i,1)=F1(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i)));
            F2(i,1)=F2(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+Nfvn)));
            F2(i,1)=F2(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+Nfvn)));
            F2(i,1)=F2(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+Nfvn)));
            F3(i,1)=F3(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+2*Nfvn)));
            F3(i,1)=F3(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+2*Nfvn)));
            F3(i,1)=F3(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+2*Nfvn)));
            F4(i,1)=F4(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+3*Nfvn)));
            F4(i,1)=F4(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+3*Nfvn)));
            F4(i,1)=F4(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+3*Nfvn)));
            F5(i,1)=F5(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+4*Nfvn)));
            F5(i,1)=F5(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+4*Nfvn)));
            F5(i,1)=F5(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+4*Nfvn)));
            F6(i,1)=F6(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+5*Nfvn)));
            F6(i,1)=F6(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+5*Nfvn)));
            F6(i,1)=F6(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+5*Nfvn)));
        end
        for i=1:Nfmn
            F7(i,1)=F7(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+6*Nfvn)));
            F7(i,1)=F7(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+6*Nfvn)));
            F7(i,1)=F7(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+6*Nfvn)));
        end
        for i=1:Nbxxn
            F8(i,1)=F8(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+6*Nfvn+Nfmn)));
            F8(i,1)=F8(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+6*Nfvn+Nfmn)));
            F8(i,1)=F8(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+6*Nfvn+Nfmn)));
        end
        for i=1:Nbyyn
            F9(i,1)=F9(i,1)+detJk*sum(qwt.*u1Val.*(phiVal*muSol1(nodes,i+6*Nfvn+Nfmn+Nbxxn)));
            F9(i,1)=F9(i,1)+detJk*sum(qwt.*u2Val.*(phiVal*muSol2(nodes,i+6*Nfvn+Nfmn+Nbxxn)));
            F9(i,1)=F9(i,1)+detJk*sum(qwt.*u3Val.*(phiVal*muSol3(nodes,i+6*Nfvn+Nfmn+Nbxxn)));
        end
    end
%%% Create load vector from its components
F=[F1;F2;F3;F4;F5;F6;F7;F8;F9];
```

L2ErrorQ3D.m

```
function [error,Uapr]=L2ErrorQ3D(Mesh,usol,U,G,phi,qpt,qwt)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mesh is the mesh file.
% usol = u1 or u2 or u3 is a component of the true solution to the PDE.
% U is a vector of the free nodal values of the FEM approximate solution
% which correspond to that component.
% G is a vector of the constrained nodal values, interpolated from (0,0,g).
% phi is the local basis functions on T_R.
% qpt is the set of quadrature points and qwt is the corresponding weights.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Basis functions phiVal[i,j] = phi_j(s_i, t_i), that is the columns
% correspond to the different phi's and the rows to the the values
% of phi_j at the quadrature points.
ql=length(qwt);
phiVal=zeros(ql,10);
for i=1:ql
    for j=1:10
        phiVal(i,j)=phi{j}(qpt(1,i),qpt(2,i),qpt(3,i));
    end
end
%%% Get the values of the piecewise quadratic solution Vaprx at global nodes
ll=Mesh.nbNod;
Uapr=zeros(ll,1);
Uapr(Mesh.FNodePtrs)=U;
Uapr(Mesh.CNodePtrs)=G;
Nt=Mesh.nbTets10;  % Number of tetrahedral elements
%%% Initialize the L2 Error to be zero
L2Err = 0;
for k = 1:Nt  %Loop and sum over all elements
    SumQ=0;
    U_Val=0;
%%% Get Global Node #s for nodes of Element k
    nodes=Mesh.TETS10(k,1:10);
%%% (x,y,z) coordinates of (local) nodes 1, 2, 3, 4 (Vertices of T_k)
    x1=Mesh.POS(nodes(1),1);
    y1=Mesh.POS(nodes(1),2);
    z1=Mesh.POS(nodes(1),3);
    x2=Mesh.POS(nodes(2),1);
    y2=Mesh.POS(nodes(2),2);
    z2=Mesh.POS(nodes(2),3);
```

```
    x3=Mesh.POS(nodes(3),1);
    y3=Mesh.POS(nodes(3),2);
    z3=Mesh.POS(nodes(3),3);
    x4=Mesh.POS(nodes(4),1);
    y4=Mesh.POS(nodes(4),2);
    z4=Mesh.POS(nodes(4),3);
%%% Compute transormation J: [x,y,z] = [x1, y1, z1] + Ju for u in reference
%%% tetrahedron, as well as its inverse transpose.
    Jk=[x2-x1,x3-x1,x4-x1;y2-y1,y3-y1,y4-y1;z2-z1,z3-z1,z4-z1];
%%% Get the (x,y,z) coordinates of the transformed quadrature points
    Tqpt=zeros(3,ql);
    for j=1:ql
            Tqpt(1,j)=x1+Jk(1,:)*qpt(:,j); %transformed x
            Tqpt(2,j)=y1+Jk(2,:)*qpt(:,j); %transformed y
            Tqpt(3,j)=z1+Jk(3,:)*qpt(:,j); %transformed z
    end
    for q=1:ql  % Evaluate v at each of the quadrature points
        uVal=usol(Tqpt(1,q),Tqpt(2,q),Tqpt(3,q));
    %%% Sum over the basis functions phi_j at the quadrature point q
        for j=1:10
        U_Val=U_Val + phiVal(q,j)*Uapr(nodes(j));
        end
        SumQ=SumQ+qwt(q)*(uVal-U_Val)^2;  % Sum over quadrature points
        U_Val=0;
    end
    L2Err=L2Err + abs(det(Jk))*abs(SumQ);
end
error=L2Err^(.5);
```

L2ErrorA2D.m

```
function [error,Zw,ZW]=L2ErrorA2D(pMesh,bCnst,w,W,muVal,qpt,qwt)

Nt=pMesh.nbTriangles6;  % Number of triangular elements
ql=length(qwt(:,1));
Nvfn=length(pMesh.FVNodePtrs(:,1));
Nmfn=length(pMesh.FMNodePtrs(:,1));
Nbxxn=length(pMesh.BxxNodePtrs(:,1));

%%% Initialize the L2 Error to be zero
L2Err = 0;
```

```
wVal=zeros(ql,1);
zindex=1;
Zw=zeros(Nt*ql,3);
ZW=zeros(Nt*ql,3);

for k = 1:Nt  %Loop and sum over all elements
    WVal=zeros(ql,1);
%%% Get GlobalNode #s for nodes of Element k
    nodes=pMesh.TRIANGLES6(k,1:6);
%%% (x,y) coordinates of (local) nodes 1,2,3.
    x1=pMesh.POS(nodes(1),1);
    y1=pMesh.POS(nodes(1),2);
    x2=pMesh.POS(nodes(2),1);
    y2=pMesh.POS(nodes(2),2);
    x3=pMesh.POS(nodes(3),1);
    y3=pMesh.POS(nodes(3),2);
%%% Matrix Transformation from the reference element to the kth element
    Jk=[x2-x1,x3-x1;y2-y1,y3-y1];
    detJk=det(Jk);
%%% Vertex Node 1 %%%
    gn1=nodes(1); % Global Node Number
    if pMesh.POS(gn1,4)==-10 % Free Vertex Node
        vn1=pMesh.NodePtrs(gn1); % 1st Vertex Node Number
        W1=W(vn1);
        W2=W(vn1+Nvfn);
        W3=W(vn1+2*Nvfn);
        W4=W(vn1+3*Nvfn);
        W5=W(vn1+4*Nvfn);
        W6=W(vn1+5*Nvfn);
    elseif pMesh.NodePtrs(gn1,3)>0 % d^2/dx^2 Boundary Node
        vbxxn1=pMesh.NodePtrs(gn1,3);
        W1=0;
        W2=0;
        W3=0;
        W4=W(vbxxn1+6*Nvfn+Nmfn);
        W5=0;
        W6=0;
    elseif pMesh.NodePtrs(gn1,3)<0 % d^2/dy^2 Boundary Node
        vbyyn1=-pMesh.NodePtrs(gn1,3);
        W1=0;
        W2=0;
```

```
        W3=0;
        W4=0;
        W5=0;
        W6=W(vbyyn1+6*Nvfn+Nmfn+Nbxxn);
    else % Corner node
        W1=0;
        W2=0;
        W3=0;
        W4=0;
        W5=0;
        W6=0;
    end

%%% Vertex Node 2 %%%
    gn2=nodes(2); % Global Node Number
    if pMesh.POS(gn2,4)==-10 % Free Vertex Node
        vn2=pMesh.NodePtrs(gn2); % 2nd Vertex Node Number
        W7=W(vn2);
        W8=W(vn2+Nvfn);
        W9=W(vn2+2*Nvfn);
        W10=W(vn2+3*Nvfn);
        W11=W(vn2+4*Nvfn);
        W12=W(vn2+5*Nvfn);
    elseif pMesh.NodePtrs(gn2,3)>0 % d^2/dx^2 Boundary Node
        vbxxn2=pMesh.NodePtrs(gn2,3);
        W7=0;
        W8=0;
        W9=0;
        W10=W(vbxxn2+6*Nvfn+Nmfn);
        W11=0;
        W12=0;
    elseif pMesh.NodePtrs(gn2,3)<0 % d^2/dy^2 Boundary Node
        vbyyn2=-pMesh.NodePtrs(gn2,3);
        W7=0;
        W8=0;
        W9=0;
        W10=0;
        W11=0;
        W12=W(vbyyn2+6*Nvfn+Nmfn+Nbxxn);
    else % Corner node
        W7=0;
```

```
        W8=0;
        W9=0;
        W10=0;
        W11=0;
        W12=0;
    end

%%% Vertex Node 3 %%%
    gn3=nodes(3); % Global Node Number
    if pMesh.POS(gn3,4)==-10 % Free Vertex Node
        vn3=pMesh.NodePtrs(gn3); % 3rd Vertex Node Number
        W13=W(vn3);
        W14=W(vn3+Nvfn);
        W15=W(vn3+2*Nvfn);
        W16=W(vn3+3*Nvfn);
        W17=W(vn3+4*Nvfn);
        W18=W(vn3+5*Nvfn);
    elseif pMesh.NodePtrs(gn3,3)>0 % d^2/dx^2 Boundary Node
        vbxxn3=pMesh.NodePtrs(gn3,3);
        W13=0;
        W14=0;
        W15=0;
        W16=W(vbxxn3+6*Nvfn+Nmfn);
        W17=0;
        W18=0;
    elseif pMesh.NodePtrs(gn3,3)<0 % d^2/dy^2 Boundary Node
        vbyyn3=-pMesh.NodePtrs(gn3,3);
        W13=0;
        W14=0;
        W15=0;
        W16=0;
        W17=0;
        W18=W(vbyyn3+6*Nvfn+Nmfn+Nbxxn);
    else % Corner node
        W13=0;
        W14=0;
        W15=0;
        W16=0;
        W17=0;
        W18=0;
    end
```

```
%%% Midpoint Node 1 %%%
    gmn1=nodes(4); % Global Midpoint Node Number
    mn1=pMesh.NodePtrs(gmn1); % 1st Midpoint Node Number
    if pMesh.POS(gmn1,4)==-10 % Free Midpoint Node
        W19=W(mn1+6*Nvfn);
    else % Boundary Midpoint Node
        W19=0;
    end
%%% Midpoint Node 2 %%%
    gmn2=nodes(5); % Global Midpoint Node Number
    mn2=pMesh.NodePtrs(gmn2); % 2nd Midpoint Node Number
    if pMesh.POS(gmn2,4)==-10 % Free Midpoint Node
        W20=W(mn2+6*Nvfn);
    else % Boundary Midpoint Node
        W20=0;
    end
%%% Midpoint Node 3 %%%
    gmn3=nodes(6); % Global Midpoint Node Number
    mn3=pMesh.NodePtrs(gmn3); % 3rd Midpoint Node Number
    if pMesh.POS(gmn3,4)==-10 % Free Midpoint Node
        W21=W(mn3+6*Nvfn);
    else % Boundary Midpoint Node
        W21=0;
    end

    for q=1:ql  %Evaluate w at each of the quadrature points
        Tqpt=[x1+Jk(1,:)*qpt(:,q),y1+Jk(2,:)*qpt(:,q)];
        Zw(zindex,1:2)=Tqpt;
        ZW(zindex,1:2)=Tqpt;
        wVal(q,1)=w(x1+Jk(1,:)*qpt(:,q),y1+Jk(2,:)*qpt(:,q));
        Zw(zindex,3)=wVal(q,1);

    %%% Node 1
        b1=bCnst{1}(k,1,1);
        b2=bCnst{1}(k,1,2);
        b3=bCnst{1}(k,1,3);
        Mu1=muVal(q,1)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

        a1=bCnst{2}(k,1,1);
        a2=bCnst{2}(k,1,2);
```

```
b1=bCnst{2}(k,1,3);
b2=bCnst{2}(k,1,4);
b3=bCnst{2}(k,1,5);
Mu2=a1*muVal(q,2)+a2*muVal(q,3)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{3}(k,1,1);
a2=bCnst{3}(k,1,2);
b1=bCnst{3}(k,1,3);
b2=bCnst{3}(k,1,4);
b3=bCnst{3}(k,1,5);
Mu3=a1*muVal(q,2)+a2*muVal(q,3)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{4}(k,1,1);
a2=bCnst{4}(k,1,2);
a3=bCnst{4}(k,1,3);
b1=bCnst{4}(k,1,4);
b2=bCnst{4}(k,1,5);
b3=bCnst{4}(k,1,6);
Mu4=a1*muVal(q,4)+a2*muVal(q,5)+a3*muVal(q,6)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q...
    ,21);

a1=bCnst{5}(k,1,1);
a2=bCnst{5}(k,1,2);
a3=bCnst{5}(k,1,3);
b1=bCnst{5}(k,1,4);
b2=bCnst{5}(k,1,5);
b3=bCnst{5}(k,1,6);
Mu5=a1*muVal(q,4)+a2*muVal(q,5)+a3*muVal(q,6)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q...
    ,21);

a1=bCnst{6}(k,1,1);
a2=bCnst{6}(k,1,2);
a3=bCnst{6}(k,1,3);
b1=bCnst{6}(k,1,4);
b2=bCnst{6}(k,1,5);
b3=bCnst{6}(k,1,6);
Mu6=a1*muVal(q,4)+a2*muVal(q,5)+a3*muVal(q,6)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q...
    ,21);

%%% Node 2
b1=bCnst{1}(k,2,1);
```

```
b2=bCnst{1}(k,2,2);
b3=bCnst{1}(k,2,3);
Mu7=muVal(q,7)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{2}(k,2,1);
a2=bCnst{2}(k,2,2);
b1=bCnst{2}(k,2,3);
b2=bCnst{2}(k,2,4);
b3=bCnst{2}(k,2,5);
Mu8=a1*muVal(q,8)+a2*muVal(q,9)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{3}(k,2,1);
a2=bCnst{3}(k,2,2);
b1=bCnst{3}(k,2,3);
b2=bCnst{3}(k,2,4);
b3=bCnst{3}(k,2,5);
Mu9=a1*muVal(q,8)+a2*muVal(q,9)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{4}(k,2,1);
a2=bCnst{4}(k,2,2);
a3=bCnst{4}(k,2,3);
b1=bCnst{4}(k,2,4);
b2=bCnst{4}(k,2,5);
b3=bCnst{4}(k,2,6);
Mu10=a1*muVal(q,10)+a2*muVal(q,11)+a3*muVal(q,12)+b1*muVal(q,19)+b2*muVal(q,20)+b3*...
    muVal(q,21);

a1=bCnst{5}(k,2,1);
a2=bCnst{5}(k,2,2);
a3=bCnst{5}(k,2,3);
b1=bCnst{5}(k,2,4);
b2=bCnst{5}(k,2,5);
b3=bCnst{5}(k,2,6);
Mu11=a1*muVal(q,10)+a2*muVal(q,11)+a3*muVal(q,12)+b1*muVal(q,19)+b2*muVal(q,20)+b3*...
    muVal(q,21);

a1=bCnst{6}(k,2,1);
a2=bCnst{6}(k,2,2);
a3=bCnst{6}(k,2,3);
b1=bCnst{6}(k,2,4);
b2=bCnst{6}(k,2,5);
```

```
b3=bCnst{6}(k,2,6);
Mu12=a1*muVal(q,10)+a2*muVal(q,11)+a3*muVal(q,12)+b1*muVal(q,19)+b2*muVal(q,20)+b3*...
    muVal(q,21);

%%% Node 3
b1=bCnst{1}(k,3,1);
b2=bCnst{1}(k,3,2);
b3=bCnst{1}(k,3,3);
Mu13=muVal(q,13)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{2}(k,3,1);
a2=bCnst{2}(k,3,2);
b1=bCnst{2}(k,3,3);
b2=bCnst{2}(k,3,4);
b3=bCnst{2}(k,3,5);
Mu14=a1*muVal(q,14)+a2*muVal(q,15)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{3}(k,3,1);
a2=bCnst{3}(k,3,2);
b1=bCnst{3}(k,3,3);
b2=bCnst{3}(k,3,4);
b3=bCnst{3}(k,3,5);
Mu15=a1*muVal(q,14)+a2*muVal(q,15)+b1*muVal(q,19)+b2*muVal(q,20)+b3*muVal(q,21);

a1=bCnst{4}(k,3,1);
a2=bCnst{4}(k,3,2);
a3=bCnst{4}(k,3,3);
b1=bCnst{4}(k,3,4);
b2=bCnst{4}(k,3,5);
b3=bCnst{4}(k,3,6);
Mu16=a1*muVal(q,16)+a2*muVal(q,17)+a3*muVal(q,18)+b1*muVal(q,19)+b2*muVal(q,20)+b3*...
    muVal(q,21);

a1=bCnst{5}(k,3,1);
a2=bCnst{5}(k,3,2);
a3=bCnst{5}(k,3,3);
b1=bCnst{5}(k,3,4);
b2=bCnst{5}(k,3,5);
b3=bCnst{5}(k,3,6);
Mu17=a1*muVal(q,16)+a2*muVal(q,17)+a3*muVal(q,18)+b1*muVal(q,19)+b2*muVal(q,20)+b3*...
    muVal(q,21);
```

```
            a1=bCnst{6}(k,3,1);
            a2=bCnst{6}(k,3,2);
            a3=bCnst{6}(k,3,3);
            b1=bCnst{6}(k,3,4);
            b2=bCnst{6}(k,3,5);
            b3=bCnst{6}(k,3,6);
            Mu18=a1*muVal(q,16)+a2*muVal(q,17)+a3*muVal(q,18)+b1*muVal(q,19)+b2*muVal(q,20)+b3*...
                muVal(q,21);

        %%% Nodes 4-6
            Mu19=bCnst{7}(k,1,1)*muVal(q,19);
            Mu20=bCnst{7}(k,2,2)*muVal(q,20);
            Mu21=bCnst{7}(k,3,3)*muVal(q,21);
            WVal(q,1)=W1*Mu1+W2*Mu2+W3*Mu3+W4*Mu4+W5*Mu5+W6*Mu6+W7*Mu7+W8*Mu8+W9*Mu9+W10*Mu10+W11*...
                Mu11+W12*Mu12+W13*Mu13+W14*Mu14+W15*Mu15+W16*Mu16+W17*Mu17+W18*Mu18+W19*Mu19+W20*...
                Mu20+W21*Mu21;
            ZW(zindex,3)=WVal(q,1);
            zindex=zindex+1;
        end
        SumQ=sum(qwt.*((wVal-WVal).^2));  %Sum over quadrature points
        L2Err=L2Err + detJk*abs(SumQ);
    end
    error=L2Err^(.5);
```

Appendix B

GMSH Geometry File

This is the GMSH (.geo) file that builds the fluid-structure geometry. GMSH creates and iterates the mesh from this core file. Note that some of the "Physical Numbers" are referenced in the MATLAB code, namely in the program that converts the GMSH (.msh) file into a MATLAB data structure.

```
// Gmsh project created on Fri Feb 08 16:16:01 2013

lc = 5;
Point(1) = {0, 0, 0, lc};
Point(2) = {1, 0, 0, lc};
Point(3) = {1, 1, 0, lc};
Point(4) = {0, 1, 0, lc};
Point(5) = {0, 0, -1, lc};
Point(6) = {1, 0, -1, lc};
Point(7) = {1, 1, -1, lc};
Point(8) = {0, 1, -1, lc};
Point(9) = {.5, .5, -.5, lc};

Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 1};
Line(5) = {5, 6};
Line(6) = {6, 7};
Line(7) = {7, 8};
Line(8) = {8, 5};
```

```
Line(9) = {1, 5};
Line(10) = {2, 6};
Line(11) = {3, 7};
Line(12) = {4, 8};
Line(13) = {1, 9};
Line(14) = {2, 9};
Line(15) = {3, 9};

Line(16) ={4, 9};
Line(17) = {5, 9};
Line(18) = {6, 9};
Line(19) = {7, 9};
Line(20) = {8, 9};

Line Loop(27) = {1, 2, 3, 4};
Plane Surface(28) = {27};
Line Loop(29) = {5, 6, 7, 8};
Plane Surface(30) = {29};
Line Loop(31) = {9, 5, -10, -1};
Plane Surface(32) = {31};
Line Loop(33) = {10, 6, -11, -2};
Plane Surface(34) = {33};
Line Loop(35) = {3, 12, -7, -11};
Plane Surface(36) ={35};
Line Loop(37) = {4, 9, -8, -12};
Plane Surface(38) = {37};

Line Loop(39) = {1, 14, -13};
Plane Surface(40) = {39};
Line Loop(41) = {2, 15, -14};
Plane Surface(42) = {41};
Line Loop(43) = {3, 16, -15};
Plane Surface(44) = {43};
Line Loop(45) = {4, 13, -16};
Plane Surface(46) = {45};
Line Loop(47) = {5, 18, -17};
Plane Surface(48) = {47};

Line Loop(49) = {6, 19, -18};
```

```
Plane Surface(50) = {49};
Line Loop(51) = {7, 20, -19};
Plane Surface(52) = {51};
Line Loop(53) = {8, 17, -20};
Plane Surface(54) = {53};
Line Loop(55) = {9, 17, -13};
Plane Surface(56) = {55};
Line Loop(57) = {10, 18, -14};
Plane Surface(58) = {57};

Line Loop(59) = {11, 19, -15};
Plane Surface(60) = {59};
Line Loop(61) = {12, 20, -16};
Plane Surface(62) = {61};

//////////////////Top and No Slip sides
Physical Surface(90) = {32, 34, 36, 38, 30};
Physical Surface(91) = {28};

//Top
///////Using Square Top
Surface Loop(96) = {28, 40, 42, 44, 46};
Volume(97) = {96};
Physical Volume(98) = {97};

//Bottom
////////Using Square Bottom
Surface Loop(99) = {30, 48, 50, 52, 54};
Volume(100) = {99};
Physical Volume(101) = {100};

//Left
////////Using Square Left Side
Surface Loop(102) = {32, 40, 58, 48, 56};
Volume(103) = {102};
Physical Volume(104) = {103};

//Right
////////Using Square Right Side
```

```
Surface Loop(105) = {36, 44, 60, 52, 62};
Volume(106) = {105};
Physical Volume(107) = {106};

//Front
////////Using Square Front Side
Surface Loop(108) = {34, 42, 58, 50, 60};
Volume(109) = {108};
Physical Volume(110) = {109};

//Back
////////Using Square Back Side
Surface Loop(111) = {38, 46, 56, 54, 62};
Volume(112) = {111};
Physical Volume(113) = {112};
```

Bibliography

[1] J. Akin, "Finite Element Analysis with Error Estimators", Elsevier (2005).

[2] G. Avalos and F. Bucci, "Spectral analysis and rational decay rates of strong solutions to a fluid-structure PDE system ", e-Print arXiv:1312.4812v1 [Math.AP], 2013.

[3] G. Avalos and M. Dvorak, "A new maximality argument for a coupled fluid-structure interaction, with implications for a divergence-free finite element method", *Applicationes Mathematicae*, Vol. 35, No. 3 (2008), pp. 259-280.

[4] G. Avalos and R. Triggiani, "Backward uniqueness of the s.c. semigroup arising in parabolic-hyperbolic fluid-structure interaction", J. Differential Equations, 245 (2008), p. 737-761.

[5] G. Avalos and R. Triggiani, "Backwards-Uniqueness of the C_0 Semigroup Associated with a Parabolic-Hyperbolic Stokes-Lamé Partial Differential Equation System", *Transactions of the American Mathematical Society*, Volume 362, Number 7 (July 2010), pp. 3535-3561.

[6] O. Axelsson and V.A. Barker, "Finite Element Solution of Boundary Value Problems: Theory and Computation", Academic Press (1984).

[7] H. Blum and R. Rannacher, "On the Boundary Value Problem of the Biharmonic Operator on Domains with Angular Corners."Math. Meth. in the Appl. Sci. **2** (1980) 556-581.

[8] D. Boffi, F. Brezzi, and M. Fortin. Finite elements for the Stokes problem. In Mixed finite elements, compatibility conditions, and applications. Lectures given at the C.I.M.E. Summer School held in Cetraro, Italy, June 26-July 1, 2006. Lecture Notes in Mathematics. Springer Verlag. Vol. 1939 (2008), pp. 45-100. D. Boffi, L. Gastaldi editors.

[9] J. Bramble, "The Lagrange multiplier method for Dirichlet's mroblem", Math. Comput. 37 (1981) 1-11.

[10] S. Brenner and L. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York (1994).

[11] F. Brezzi and M. Fortin, "Mixed and Hybrid Finite Element Methods", Springer-Verlag (1991).

[12] J. Burkardt, "Quadrature Rules for Triangles."Website, 2010. http://people.sc.fsu.edu/~jburkardt/datasets/quadrature_rules_tri/quadrature_rules_tri.html.

[13] A. Chambolle, B. Desjardins, M. Esteban, C. Grandmont,"Existence of weak solutions for the unsteady interaction of a viscous fluid with an elastic plate."J. Math. Fluid Mech. 7 (2005), 368404.

[14] I. Chueshov and I. Ryzhkova, "A global attractor for a fluid-plate interaction model", *Communications on Pure and Applied Analysis*, Volume 12, Number 4 (July 2013), pp. 1635-1656.

[15] I. Chueshov, "A global attractor for a fluid-plate interaction model accounting only for longitudinal deformations of the plate, "Math. Methods Appl. Sci. 34, 1801-1812.

[16] P. Ciarlet, "The Finite Element Method for Elliptic Problems", North-Holland (1978).

[17] M. Dauge, "Stationary Stokes and Navier-Stokes systems on Two- or Three-Dimensional Domains with Corners. Part 1: Linearized Equations", *Siam J. Math. Anal.* Vol. 20, No.1, January 1989.

[18] V. Domínguez and F.J. Sayas, "Algorithm 884: A simple MATLAB implementation of the Argyris element ", *ACM Trans. Math. Software*, Volume 35, Issue 2 (2008).

[19] E. Dowell and K. Hall, "Modeling of Fluid-Structure Interaction", *Annu. Rev. Fluid Mech.* **33** (2001), pp. 445-490.

[20] M. Dvorak, *Qualitative and Quantitative Analysis of a Fluid-Structure Interactive Partial Differential Equation Model* Ph.D. Thesis. University of Nebraska-Lincoln (2008).

[21] A. Ern and J. Guermond, "Theory and Practice of Finite Elements", Springer-Verlag (2004).

[22] E. Foster, *Finite Elements for the Quasi-Geostrophic Equations of the Ocean* Ph.D. Thesis. Virginia Polytechnic Institute and State University (2013).

[23] G.P. Galdi, *An Introduction to the Mathematical Theory of the Navier-Stokes Equations*, Springer Tracts in Natural Philosophy 38, Springer, New York (1994).

[24] M. Gockenbach, "Understanding and Implementing the Finite Element Method", Siam (2006).

[25] P. Grisvard, "Caracterization de quelques espaces d'interpolation", *Arch. Rational Mech. Anal.* **25** (1967), pp. 40-63.

[26] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman Advanced Pub. Program (1985).

[27] B. Kellogg, "Properties of solutions of elliptic boundary value problems", in *The Mathematical Foundations of the Finite Element Method with Applications to Partial*

Differential Equations, Edited by A. K. Aziz, Academic Press, New York (1972), pp. 47-81.

[28] S. Kesavan, *Topics in Functional Analysis and Applications*, Wiley, New York (1989).

[29] S.G. Krein, *Linear Differential Equations in Banach Space*, Amer. Math. Soc., Providence, RI (1971).

[30] I. Lasiecka, M. Renardy, R. Triggiani, "Backward uniqueness for thermoelastic plates", Semigroup Forum 62 (2001), p. 217-242.

[31] J.L. Lions and E. Magenes, *Non-homogeneous boundary value problems and applications*, Vol. I, Springer-Verlag (1972).

[32] A. Pazy, *Semigroups of Linear Operators and Applications to Partial Differential Equations*, Springer-Verlag, New York (1983).

[33] P. Šolin "Partial Differential Equations and the Finite Element Method ", Wiley (2006).

[34] R. Temam, "Navier-Stokes Equations, Theory and Numerical Analysis", AMS Chelsea Publishing, Providence, Rhode Island (2001).

[35] R. Triggiani, "Backward uniqueness of semigroups arising in coupled PDE systems of structural acoustics", Adv. Differential Equations 8 (1–2) (Jan.–Feb. 2004) 53-84. Preliminary announcement in: C. Kubrusly, N. Levan, M. da Silveira (Eds.), Semigroups of Operators: Theory and Applications, 2002, pp. 285–300.

[36] R. Wait and A.R. Mitchell, "Finite Element Analysis and Applications", Wiley (1985).