UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE

MÉTODOS DE ELEMENTOS FINITOS PARA PROBLEMAS DE FLUIDOS: ASPECTOS COMPUTACIONALES.

GUSTAVO GARCÍA GUERRERO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESORES GUÍA: ALEJANDRO ALLENDES, PhD. - Departamento de Matemática.
ENRIQUE OTÁROLA, PhD. - Departamento de Matemática.

PROFESOR CORREFERENTE: CLAUDIO TORRES, PhD. - Departamento de Informática.

Índice

1. Introducción 4
2. Resolución de sistemas lineales 7
3. Marco funcional 10
4. El Método de elementos finitos para adveción–reacción–difusión 14
4.1. Formulación débil 14
4.2. Formulación de Galerkin 18
4.3. Convergencia del método de elementos finitos 25
4.4. Un código de elementos finitos en C/C++ 28
4.5. Ejemplos numéricos 37
5. Un Método de elementos finitos estabilizado para adveción–reacción–difusión 44
5.1. Formulación de Galerkin estabilizada 44
5.2. Convergencia del método estabilizado 48
5.3. Código del Esquema Estabilizado 53
5.4. Ejemplos numéricos 56
6. Un método de elementos finitos estabilizados para un problema de Stokes 61
6.1. Un método de elementos finitos estabilizado 64
6.2. Convergencia del método estabilizado 67
6.3. Código de Stokes 70
6.4. Ejemplos numéricos 72
7. Un método de elementos finitos estabilizados para un problema de Navier–Stokes 75
7.1. Un método de punto fijo 76
7.2. Un método de Newton 77
7.3. Código del método estabilizado para Navier Stokes 81
7.4. Ejemplos numéricos 88
8. Conclusiones 93
Referencias 95

1. Introducción

La motivación de este trabajo es estudiar y resolver una serie de ecuaciones diferenciales parciales de segundo orden asociadas a la discretización de una de las ecuaciones más fundamentales en la dinámica de fluidos, las cuales son las ecuaciones de Navier–Stokes:

Hallar $(\mathbf{u}, p)$ ***tal que,***

$$\text{(1.1)} \qquad \begin{cases} -\varepsilon\Delta\mathbf{u}(\boldsymbol{x}) + \mathbf{u}(\boldsymbol{x})\cdot\nabla\mathbf{u}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) &= \mathbf{f}(\boldsymbol{x}) \quad \forall\, \boldsymbol{x}\in\Omega, \\ \text{div } \mathbf{u}(\boldsymbol{x}) &= 0 \quad \forall\, \boldsymbol{x}\in\partial\Omega, \\ \mathbf{u}(\boldsymbol{x}) &= \mathbf{0} \quad \forall\, \boldsymbol{x}\in\partial\Omega, \end{cases}$$

Donde $\mathbf{u}$ es la velocidad de un fluido y p es la presión del fluido, el cual está presente en un dominio Ω en $\mathbb{R}^2$.

La discretización de dichas ecuaciones implica la separación en dos fenómenos distintos, los cuales tienen que ser estudiados de forma diferente.

El primer fenómeno corresponde al transporte, lo que se modela mediante la llamada ecuación de advección–reacción–difusión:

Hallar u ***tal que,***

$$\text{(1.2)} \qquad \begin{cases} -\varepsilon\Delta\mathsf{u}(\boldsymbol{x}) + \boldsymbol{b}(\boldsymbol{x})\cdot\nabla\mathsf{u}(\boldsymbol{x}) + \mathsf{u}(\boldsymbol{x}) &= \mathsf{f}(\boldsymbol{x}) \quad \forall\, \boldsymbol{x}\in\Omega, \\ \mathsf{u}(\boldsymbol{x}) &= 0 \quad \forall\, \boldsymbol{x}\in\partial\Omega, \end{cases}$$

donde se tiene que:

- Ω es un dominio en $\mathbb{R}^2$, con frontera $\partial\Omega$;
- $\varepsilon > 0$ es el parámetro de difusión.
- el operador diferencial Laplaciano Δ está dado por:

$$\Delta\mathsf{u}(x,y) = \frac{\partial^2\mathsf{u}}{\partial x^2} + \frac{\partial^2\mathsf{u}}{\partial y^2};$$

- $\boldsymbol{b}(\boldsymbol{x}) = (b_1(\boldsymbol{x}), b_2(\boldsymbol{x}))$ es un campo vectorial lo suficientemente suave, llamado el campo de advección;
- el operador diferencial divergencia ∇ está dado por:

$$\nabla\mathsf{u}(x,y) = \left(\frac{\partial\mathsf{u}}{\partial x}, \frac{\partial\mathsf{u}}{\partial y}\right);$$

El problema (1.1) modela distintos fenómenos físicos de transporte, donde la variable u corresponde a la cantidad transferida al medio Ω, por ejemplo contaminante, temperatura, etc (ver [15]).

El segundo fenómeno que se estudiará corresponde a las ecuaciones de un fluido, para lo cual nos vamos a restringir a un fluido lineal, para lo cual, consideraremos un fluido modelado mediante las ecuaciones de Stokes:

Hallar $(\mathbf{u}, p)$ ***tal que,***

$$\left\{\begin{array}{rcl} -\varepsilon\Delta\mathbf{u}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) & = & \mathbf{f}(\boldsymbol{x}) \quad \forall\, \boldsymbol{x} \in \Omega, \\ \text{div } \mathbf{u}(\boldsymbol{x}) & = & 0 \quad \forall\, \boldsymbol{x} \in \partial\Omega, \\ \mathbf{u}(\boldsymbol{x}) & = & \mathbf{0} \quad \forall\, \boldsymbol{x} \in \partial\Omega. \end{array}\right. \tag{1.3}$$

La dificultad que emerge de estos sistemas de ecuaciones proviene del hecho de que son ecuaciones acopladas, entonces se requiere la elección apropiada de la forma de discretización del dominio presentado, ya que dicha elección influirá ampliamente en los espacios de elementos finitos a generar.

Claramente el único tipo de solución que puede obtenerse para estos problemas es la del tipo analítica, entonces es fundamental recurrir a esquemas numéricos que puedan aproximar dicha solución, y dentro de la gran variedad de esquemas existentes [3, 14] este trabajo se basará en el Método de Elementos Finitos.

El Método de Elementos Finitos es un esquema numérico que sirve para aproximar la solución de una ecuación diferencial parcial u ordinaria, cuya solución vive en un espacio de dimensión infinita, por una que viva en un espacio de dimensión finita. Aún considerando que en la actualidad existe una gran variedad de métodos de elementos finitos para resolver dichos problemas (ver [15]), el Método de Elementos Finitos es el preferido en la comunidad ingenieril dada su gran robustez y amplio desarrollo matemático (ver [5, 8, 4, 9]).

Dentro de los fundamentos del esquema de elementos finitos está la necesidad de construir mallas sobre el dominio en cuestión, para después aproximar mediante espacios polinomiales, construidos en base a dicha partición del dominio físico (ver Figura 1).

En general, la teoría de Elementos Finitos se apoya en el uso de formulaciones débiles, para ecuaciones diferenciales (ver [10]), las cuales son discretizadas y que requieren de la resolución de sistemas de ecuaciones lineales de gran tamaño.

En base a lo anterior, el objetivo general de este trabajo es:

- Estudiar e implementar Métodos de Elementos Finitos para resolver las ecuaciones de Navier–Stokes (1.1).

Y los objetivos específicos son:

- Estudiar distintos métodos de elementos finitos adecuados para una apropiada aproximación de la solución de los problemas de Advección-Reacción-Difusión (1.2) y de Stokes (1.3).
- Implementar computacionalmente dichos esquemas para distintos escenarios, en dos dimensiones.
- Considerar en dichas implementaciones aplicaciones con geometrías que simulen algún fenómeno físico.
- En base a las técnicas de métodos de elementos finitos logradas en los problemas anteriores, resolver el problema (1.1) usando el Método de Elementos Finitos.

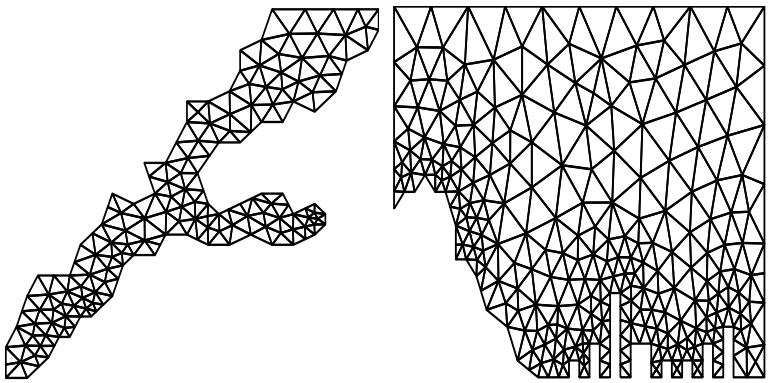

FIGURA 1. Partición de dominios en dos y tres dimensiones, basados en particiones usando elementos triangulares y tetrahédricos.

2. Resolución de sistemas lineales

Al momento de discretizar una ecuación diferencial parcial mediante el método de elementos finitos, es que se tiene que resolver un sistema de ecuaciones lineales de la forma

$$\boldsymbol{B\alpha} = \boldsymbol{f},$$

donde la matriz $\boldsymbol{B}$ es la llamada, en términos generales, *matriz de rigidez*, $\boldsymbol{f}$ es llamado el vector fuente y $\boldsymbol{\alpha}$ el conjunto de incógnitas. Existen en la actualidad distintos métodos para resolver directamente este sistema de ecuaciones. Una forma simple y directa de resolver el problema es invertir la matriz $\boldsymbol{B}$ y multiplicarla por el vector $\boldsymbol{f}$, dando por resultado un vector con las soluciones a las incógnitas. Sin embargo, nunca se realiza el cálculo de la inversa de $\boldsymbol{B}$ ya que no es recomendado en ningún caso. Para calcular la solución directamente se emplean otros métodos comunes tales como la eliminación gaussiana y las descomposiciones LU, Cholesky o QR.

Dada la particular estructura del método de elementos finitos, es que dicha construcción hace, en general, que la matriz de rigidez tenga una estructura *sparse*, al discretizar un problema continuo. Claramente, dada la gran cantidad de ceros presente en la matriz, es que cualquier tipo de operación tiene que ser realizada de forma eficiente. Para resolver este problema existen dos clases de métodos de resolución: los *métodos directos* e *iterativos*.

Los *métodos directos* descomponen el sistema de ecuaciones, por ejemplo, utilizando una descomposición del tipo LU, es decir:

$$\boldsymbol{B\alpha} = \boldsymbol{f} \quad \Leftrightarrow \quad \boldsymbol{Ly} = \boldsymbol{f} \quad \text{y} \quad \boldsymbol{U\alpha} = \boldsymbol{y},$$

y posteriormente se realizan sustituciones hacia adelante y hacia atras.

Los *métodos iterativos* asumen una solución aproximada inicial del sistema para el vector de incógnitas x, digamos un $\boldsymbol{\alpha}_0$. Luego, los siguientes $\boldsymbol{\alpha}_i$, con $i = 1, 2, ...$, son calculados de tal manera de minimizar el error residual

$$\boldsymbol{B\alpha}_i - \boldsymbol{f}.$$

Adicionalmente, un solver iterativo realiza un proceso llamado *precondicionamiento*. El proceso de precondicionamiento se realiza antes de resolver el sistema, mejorando el número de condición de la matriz. A partir del sistema $\boldsymbol{B\alpha} = \boldsymbol{f}$, se multiplica por una matriz adecuada $\boldsymbol{M}$, para luego resolver el sistema

$$\boldsymbol{MB\alpha} = \boldsymbol{Mf}.$$

Este proceso claramente no cambia la solución $\boldsymbol{\alpha}$, sin embargo cambia el comportamiento del sistema, mejorar el número de condición de la matriz $\boldsymbol{MB}$.

Para resolver el sistema de ecuaciones se utiliza el solver MUMPS (MUltifrontal Massively Parallel sparse direct Solver) [1, 2], especializado en la resolución de sistemas sparse mediante un método directo basado en una aproximación multifrontal que realiza una descomposición de la forma

$$\boldsymbol{A} = \boldsymbol{LU}.$$

En el caso particular de la resolución del método de elementos finitos para la ecuación de Laplace, el sistema se resuelve de la forma

$$\boldsymbol{A} = \boldsymbol{LDL}^T.$$

Este software se distribuye bajo la licencia CeCILL (CEA CNRS INRIA Logiciel Libre), la cual es una adaptación legal francesa a la licencia GPL (GNU General Public License), garantizando el derecho al usuario de modificación y libre distribución de éste.

A partir del análisis realizado en [16] el cual considera distintos solvers existentes se encuentran los siguientes resultados enfocados en el rendimiento del proceso completo de resolución del sistema y la capacidad de resolver correctamente una amplia variedad de problemas. Si bien MUMPS no sobresale en ninguna de las características, se encuentra en un rango intermedio de rendimiento y estabilidad lo cual apoya la elección de MUMPS como el solver a utilizar en este código.

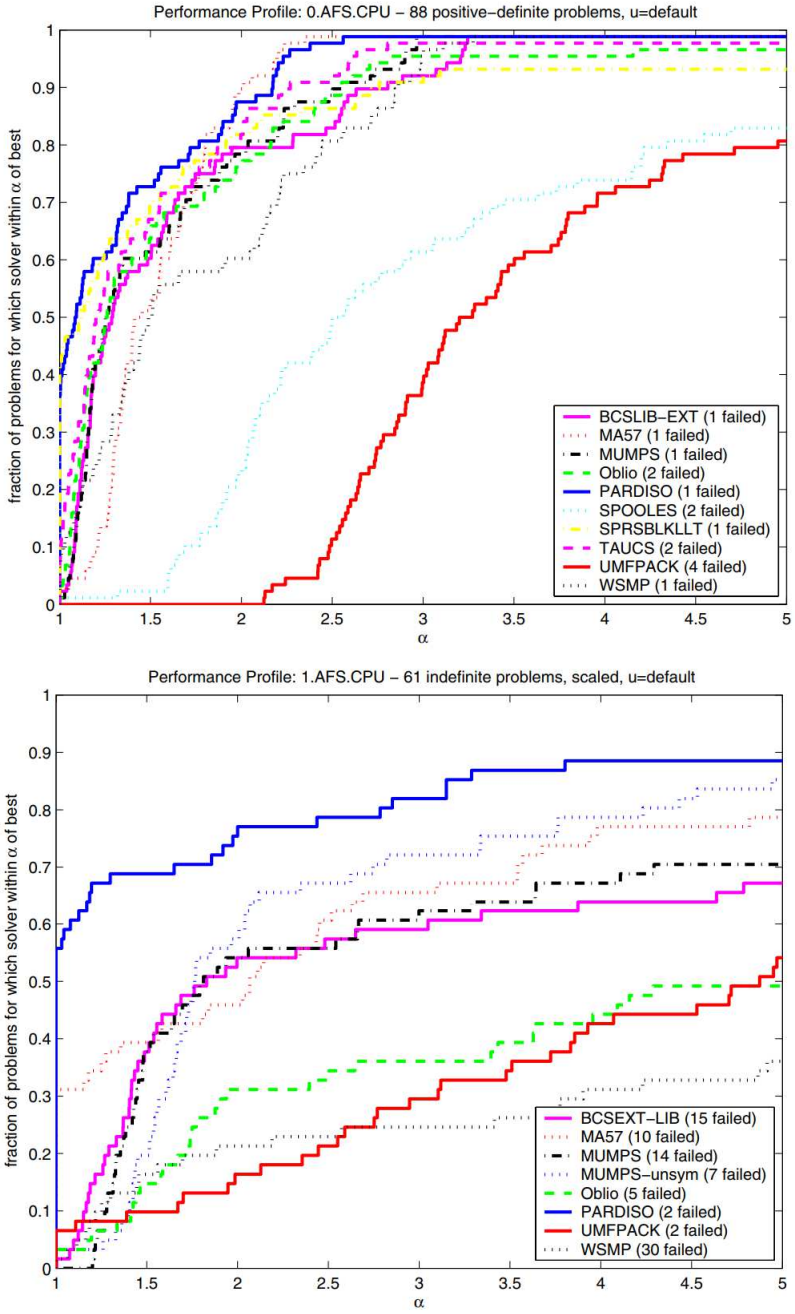

FIGURA 2. Perfil de desempeño entre distintos solvers para problemas con matrices definidas positivamente y no definidas positivamente (ver [16]).

3. Marco funcional

En esta sección, nos concentraremos en mostrar el marco funcional matemático, en el cual nos basaremos para plantear resultados de existencia de soluciones, unicidad de soluciones y la convergencia de los métodos de elementos finitos. Las propiedades de los espacios de funciones que se presentan a continuación en conjunto con distintos resultados que son válidos en estos espacios se pueden encontrar con más detalle en [6].

Primero, recordemos que un espacio de Hilbert, corresponde a un espacio vectorial, el cual es completo, es decir, toda sucesión de Cauchy converge a un elemento del espacio, y en donde el producto interno induce una norma.

Ahora, introducimos el siguiente espacio de Hilbert, de funciones de cuadrado integrable en un dominio Ω que es un subconjunto cerrado y acotado de $\mathbb{R}^d$, con frontera $\partial\Omega$:

$$L^2(\Omega) := \left\{ v(\boldsymbol{x}) : \int_\Omega v^2(\boldsymbol{x})\, d\boldsymbol{x} < \infty \right\}, \tag{3.1}$$

en el cual, se puede introducir el siguiente producto interno

$$(v, w)_{L^2(\Omega)} := \int_\Omega v(\boldsymbol{x}) w(\boldsymbol{x})\, d\boldsymbol{x}. \tag{3.2}$$

Claramente, el producto interno anterior, induce la siguiente norma

$$\|t\|_{L^2(\Omega)} := (t,t)^{1/2}_{L^2(\Omega)} = \left(\int_\Omega t^2\, d\boldsymbol{x} \right)^{1/2}. \tag{3.3}$$

Entonces, se tiene que el espacio anterior con la norma mencionada $\left(L^2(\Omega), \|\cdot\|_{L^2(\Omega)}\right)$ es un espacio de Hilbert. Dentro del análisis de elementos finitos es que es necesario introducir el siguiente espacio, definido sobre un dominio $\Omega \subset \mathbb{R}^d$, con frontera $\partial\Omega$, llamado espacio de Sobolev:

$$H^1(\Omega) := \left\{ v(\boldsymbol{x}) \in L^2(\Omega) : \frac{\partial v}{\partial x_i} \in L^2(\Omega),\ \forall\ i = 1, ..., d \right\}. \tag{3.4}$$

El espacio anterior es un espacio de Hilbert con norma inducida

$$\|v\|_{H^1(\Omega)} := \left(\|v\|^2_{L^2(\Omega)} + \|\nabla v\|^2_{L^2(\Omega)} \right)^{1/2} = \left(\|v\|^2_{L^2(\Omega)} + \sum_{i=1}^{d} \left\| \frac{\partial v}{\partial x_i} \right\|^2_{L^2(\Omega)} \right)^{1/2}. \tag{3.5}$$

De igual forma, podemos definir el espacio de Sobolev

$$H^2(\Omega) := \left\{ v(\boldsymbol{x}) \in L^2(\Omega) : \frac{\partial v}{\partial x_i} \in L^2(\Omega),\ \frac{\partial^2 v}{\partial x_i \partial x_j} \in L^2(\Omega),\ \forall\ i, j = 1, ..., d \right\}, \tag{3.6}$$

con norma

$$\|v\|_{H^2(\Omega)} = \left(\|v\|^2_{L^2(\Omega)} + \sum_{i=1}^{d} \left\| \frac{\partial v}{\partial x_i} \right\|^2_{L^2(\Omega)} + \sum_{i=1}^{d} \left\| \frac{\partial^2 v}{\partial x_i \partial x_j} \right\|^2_{L^2(\Omega)} \right)^{1/2}.$$

También se introduce el siguiente subespacio de $H^1(\Omega)$, el cual incorpora condiciones de frontera sobre la funciones, que es

$$H^1_0(\Omega) := \left\{ v(\boldsymbol{x}) \in H^1(\Omega) : \ v_{|\partial\Omega} = 0 \right\}, \tag{3.7}$$

donde $v_{|\partial\Omega}$ corresponde a la restricción de v a la frontera de Ω, pero en un sentido distribucional.

Dentro de todo el análisis de elementos finitos es que ocuparemos reiteradamente, lo que se conoce como integración por partes, que corresponde a una serie de corolarios que se obtienen a partir del conocido teorema de Gauss, el cual dice que para cualquier dominio Ω lo suficientemente suave y con vector normal unitario $\boldsymbol{n}$ (ver Figura 3), dado cualquier campo vectorial $\boldsymbol{G}$, lo suficientemente suave, se cumple que

$$\int_\Omega \text{div } \boldsymbol{G} \, d\boldsymbol{x} = \int_{\partial\Omega} \boldsymbol{G} \cdot \boldsymbol{n} \, ds. \tag{3.8}$$

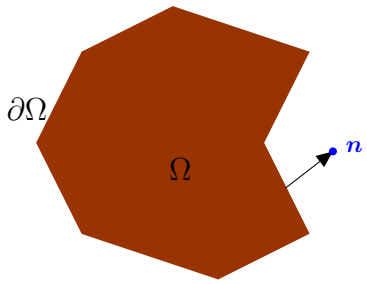

FIGURA 3. Dominio Ω, con frontera $\partial\Omega$ y vector normal unitario $\boldsymbol{n}$.

Con respecto al resultado anterior, es que se pueden deducir una serie de resultados con respecto a integración. Lo primero es considerar el siguiente campo vectorial,

$$\boldsymbol{G} = \left(0, ..., \underbrace{u(\boldsymbol{x})v(\boldsymbol{x})}_{\text{i-ésima}}, .., 0 \right)$$

que contiene solo ceros, menos en su i-ésima entrada, la cual contiene la multiplicación de dos funciones lo suficientemente suaves u y v. Luego, si el vector normal está dado

por $\boldsymbol{n} = (n_1, ..., n_d)$, luego aplicando el teorema de Gauss, es decir, reemplazando dicha elección en (3.8), se tiene que

$$\int_\Omega \frac{\partial u}{\partial x_i} v \; d\boldsymbol{x} = -\int_\Omega u \frac{\partial v}{\partial x_i} \; d\boldsymbol{x} + \int_{\partial\Omega} u \, v \, n_i \; ds. \tag{3.9}$$

Ahora, si consideramos que la función u es talque $u = \frac{\partial w}{\partial x_i}$, donde la función $w(\boldsymbol{x})$ es de igual forma lo suficientemente suave, luego reemplazando dicha elección en (3.9), se tiene que se cumple

$$\int_\Omega \frac{\partial^2 w}{\partial x_i} v \; d\boldsymbol{x} = -\int_\Omega \frac{\partial w}{\partial x_i} \frac{\partial v}{\partial x_i} \; d\boldsymbol{x} + \int_{\partial\Omega} \frac{\partial w}{\partial x_i} n_i \, v \; ds. \tag{3.10}$$

Finalmente, notando que dicha elección fue arbitraria, luego variando la i-ésima entrada y sumando sobre todos los índices, es que se puede concluir que para funciones lo suficientemente suaves, se cumple que

$$\int_\Omega \Delta u \, v \; d\boldsymbol{x} = -\int_\Omega \nabla u \cdot \nabla v \; d\boldsymbol{x} + \int_{\partial\Omega} \nabla u \cdot \boldsymbol{n} \, v \; ds. \tag{3.11}$$

Ahora recordaremos una serie de desigualdades que serán útiles dentro de todo el análisis de elementos finitos junto con el análisis de error. La primera es la conocida desigualdad de Cauchy–Schwarz, que dice que si dos funciones $f, g \in L^2(\Omega)$, luego se cumple que

$$\int_\Omega fg \; d\boldsymbol{x} \leq \|f\|_{L^2(\Omega)} \|g\|_{L^2(\Omega)}. \tag{3.12}$$

Una generalización del resultado anterior, es la conocida desigualdad de Hölder, que dice que si $f \in L^p(\Omega)$ y $g \in L^q(\Omega)$, donde $\frac{1}{p} + \frac{1}{q} = 1$, luego se cumple que

$$\int_\Omega fg \; d\boldsymbol{x} \leq \|f\|_{L^p(\Omega)} \|g\|_{L^q(\Omega)}, \tag{3.13}$$

donde el espacio $L^t(\Omega)$, para un natural t, se define como

$$L^t(\Omega) := \left\{ v : \; \int_\Omega v^t \; d\boldsymbol{x} < \infty \right\},$$

y para el caso cuando $t = \infty$, el espacio se define como

$$L^\infty(\Omega) := \left\{ v : \; \|v\|_{L^\infty(\Omega)} < \infty \right\},$$

donde

$$\|v\|_{L^\infty(\Omega)} = \text{ínf}\{M \geq 0 : \; |v(\boldsymbol{x})| \leq M \text{ en } \Omega\}.$$

Finalmente, una generalización del resultado anterior es que si una serie de funciones $f_1, f_2, ..., f_k$ para las cuales se cumple que

$$f_i \in L^{p_i}, \; 1 \leq i \leq k \text{ con } \frac{1}{p} = \frac{1}{p_1} + \frac{1}{p_2} + ... + \frac{1}{p_k} \leq 1,$$

luego

$$\left(\int_\Omega (f_1 \cdots f_k)^p \, d\boldsymbol{x}\right)^{1/p} \leq \|f_1\|_{L^{p_1}(\Omega)} \|f_2\|_{L^{p_2}(\Omega)} \cdots \|f_k\|_{L^{p_k}(\Omega)}. \tag{3.14}$$

La última desigualdad que será útil dentro de nuestro análisis es la desigualdad de Poincaré, que dice que

$$\|v\|_{L^2(\Omega)} \leq \mathtt{C}_{P,\Omega} \|\nabla v\|_{L^2(\Omega)} \quad \forall\ v \in H_0^1(\Omega), \tag{3.15}$$

donde la constante $\mathtt{C}_{P,\Omega}$ solamente depende del dominio.

Por último, recordaremos un conocido resultado del análisis funcional que nos entregará existencia y unicidad tanto para el problema débil como también para la aproximación de elementos finitos (ver [12, Capítulo 1]), para lo cual usaremos las siguientes definiciones:

Dada una forma bilineal $\mathcal{B} : V \times V \to \mathbb{R}$, es decir, que es lineal en cada componente, diremos que:

- $\mathcal{B}$ es continua, si
$$\mathcal{B}(\mathsf{w}, \mathsf{v}) \leq C_{up} \|\mathsf{w}\|_V \|\mathsf{v}\|_V, \quad \forall\ \mathsf{w}, \mathsf{v} \in V;$$
- $\mathcal{B}$ es coerciva, si
$$\mathcal{B}(\mathsf{v}, \mathsf{v}) \leq C_{lw} \|\mathsf{v}\|_V^2 \quad \forall\ \mathsf{v} \in V,$$
donde V es un espacio de Hilbert con norma $\|\cdot\|_V$. De forma similar, dado un funcional lineal $\mathcal{F} : V \to V$, diremos que el funcional es continua si satisface que
$$\mathcal{F}(\mathsf{v}) \leq C \|\mathsf{v}\|_V \quad \forall\ \mathsf{v} \in V.$$

Teorema 1 (Lax-Milgram)**.** *Sea $(V, \|\cdot\|_V)$ un espacio de Hilbert, $\mathcal{B} : V \times V \to \mathbb{R}$ una forma bilineal continua y coerciva, y $\mathcal{F} : V \to \mathbb{R}$ un funcional lineal y continuo. Luego, existe solución única para el problema:*

Hallar $\mathsf{u} \in V$ tal que cumpla con

$$\mathcal{B}(\mathsf{u}, \mathsf{v}) = \mathcal{F}(\mathsf{v}) \quad \forall\ \mathsf{v} \in V.$$

Cada requerimiento del teorema anterior, será especificado en lo que sigue de las secciones y demostrado cuando se requiera.

4. El Método de elementos finitos para adveción–reacción–difusión

En esta sección, aplicaremos la teoría de elementos finitos a nuestra ecuación (1.2), para lo cual nos basaremos en la teoría expuesta en [5].

4.1. Formulación débil. El primer paso para el planteo de un esquema de elementos finitos, corresponde a obtener la formulación débil asociada a la ecuación diferencial parcial (1.1). El tratamiento usual es como sigue. Consideramos primero cualquier función lo suficientemente suave $v(\boldsymbol{x})$, la cual multiplicará toda la ecuación diferencial, es decir,

$$-\varepsilon\Delta\mathsf{u}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x}) + \boldsymbol{b}\cdot\nabla\mathsf{u}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x}) + \mathsf{u}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x}) = \mathsf{f}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x}).$$

Ahora, integramos la ecuación anterior en todo el dominio, es decir,

$$\int_\Omega \left(-\varepsilon\Delta\mathsf{u}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x}) + \boldsymbol{b}\cdot\nabla\mathsf{u}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x}) + \mathsf{u}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x})\right)\, d\boldsymbol{x} = \int_\Omega \mathsf{f}(\boldsymbol{x})\mathsf{v}(\boldsymbol{x})\, d\boldsymbol{x}.$$

Usando la integración por partes (3.11), podemos reescribir la ecuación anterior como

$$\int_\Omega \left(\varepsilon\nabla\mathsf{u}\cdot\nabla\mathsf{v} + \boldsymbol{b}\cdot\nabla\mathsf{uv} + \mathsf{uv}\right)\, d\boldsymbol{x} - \int_{\partial\Omega}(\nabla\mathsf{u}\cdot\boldsymbol{n})\mathsf{v}\, ds = \int_\Omega \mathsf{fv}\, d\boldsymbol{x}.$$

Como nuestro problema contempla condiciones del tipo Dirichlet en la frontera, es decir, $u = 0$ en $\partial\Omega$, luego la derivada normal que aparece de forma natural, pero que no es dato dentro de nuestro problema, es que tenemos que eliminarla de la ecuación anterior. Entonces, a la función arbitraria v le pediremos que se anule en la frontera del dominio, luego con dicho requerimiento es que podemos reescribir la ecuación anterior como

$$\int_\Omega \left(\varepsilon\nabla\mathsf{u}\cdot\nabla\mathsf{v} + \boldsymbol{b}\cdot\nabla\mathsf{uv} + \mathsf{uv}\right)\, d\boldsymbol{x} = \int_\Omega \mathsf{fv}\, d\boldsymbol{x}.$$

La ecuación anterior es la conocida formulación débil del problema diferencial ya que transformamos los requerimientos de continuidad a requerimientos de integrabilidad. Ahora, basados en la Sección 3, es que podemos reescribir finalmente nuestro problema como sigue:

Encontrar $\mathsf{u} \in H_0^1(\Omega)$ tal que

$$\mathcal{B}(\mathsf{u},\mathsf{v}) = \mathcal{F}(\mathsf{v}), \quad \forall\ \mathsf{v} \in H_0^1(\Omega), \tag{4.1}$$

donde

$$\begin{aligned} \mathcal{B}(\mathsf{u},\mathsf{v}) &:= \int_\Omega \left(\varepsilon\nabla\mathsf{u}\cdot\nabla\mathsf{v} + \boldsymbol{b}\cdot\nabla\mathsf{uv} + \mathsf{uv}\right)\, d\boldsymbol{x}, \\ \mathcal{F}(\mathsf{v}) &:= \int_\Omega \mathsf{fv}\, d\boldsymbol{x}. \end{aligned} \tag{4.2}$$

Para poder demostrar existencia y unicidad de una solución para el problema anterior es que necesitamos demostrar todos los requerimientos presentados en el

Teorema (1), para el cual consideraremos que $V = H_0^1(\Omega)$ dotado con la siguiente norma:

$$\|\mathsf{v}\|_{\Omega} := \left(\varepsilon\|\nabla\mathsf{v}\|_{L^2(\Omega)} + \|\mathsf{v}\|_{L^2(\Omega)}\right)^{1/2} \quad \forall\ \mathsf{v} \in H_0^1(\Omega). \tag{4.3}$$

Continuidad de $\mathcal{B}$: En la Definición 1.2 en [12], se tiene que la continuidad de la forma $\mathcal{B}$ corresponde a demostrar que, dado un espacio de Hilbert V con norma $\|\cdot\|_V$, se cumple la siguiente desigualdad

$$|\mathcal{B}(\mathsf{w},\mathsf{v})| \leq \mathtt{C}_{up}\|\mathsf{w}\|_V\|\mathsf{v}\|_V \quad \forall\ \mathsf{w},\mathsf{v} \in V.$$

Ahora, tomando cualquier par de funciones $\mathsf{w},\mathsf{v} \in H_0^1(\Omega)$, luego por definición, tanto las funciones como sus derivadas viven en el espacio $L^2(\Omega)$, entonces podemos hacer uso de la desigualdad de Cauchy–Schwarz (3.12), de la siguiente manera

$$\int_\Omega \varepsilon\nabla\mathsf{u}\cdot\nabla\mathsf{v}\ d\boldsymbol{x} = \sum_{i=1}^{d}\varepsilon\int_\Omega \frac{\partial\mathsf{u}}{\partial x_i}\frac{\partial\mathsf{v}}{\partial x_i}\ d\boldsymbol{x} \leq \sum_{i=1}^{d}\sqrt{\varepsilon}\left\|\frac{\partial\mathsf{u}}{\partial x_i}\right\|_{L^2(\Omega)}\sqrt{\varepsilon}\left\|\frac{\partial\mathsf{v}}{\partial x_i}\right\|_{L^2(\Omega)}.$$

Ahora, podemos utilizar la desigualdad de Hölder (3.14), con parámetros $p_1 = \infty$, $p_2 = p_3 = 2$, de la siguiente manera

$$\int_\Omega \boldsymbol{b}\cdot\nabla\mathsf{u}\mathsf{v}\ d\boldsymbol{x} = \sum_{i=1}^{d}\int_\Omega b_i\frac{\partial\mathsf{u}}{\partial x_i}\mathsf{v}\ d\boldsymbol{x} \leq \sum_{i=1}^{d}\frac{\|b_i\|_{L^\infty(\Omega)}}{\sqrt{\varepsilon}}\sqrt{\varepsilon}\left\|\frac{\partial\mathsf{u}}{\partial x_i}\right\|_{L^2(\Omega)}\|\mathsf{v}\|_{L^2(\Omega)}.$$

Por último, nuevamente podemos hacer uso de (3.12) como

$$\int_\Omega \mathsf{u}\mathsf{v}\ d\boldsymbol{x} \leq \|\mathsf{u}\|_{L^2(\Omega)}\|\mathsf{v}\|_{L^2(\Omega)}.$$

Ahora, sumando todas las desigualdades anteriores, permite concluir que

$$
\begin{aligned}
\mathcal{B}(\mathsf{u},\mathsf{v}) &= \int_\Omega (\varepsilon \nabla \mathsf{u}\cdot\nabla \mathsf{v} + \boldsymbol{b}\cdot\nabla \mathsf{u}\mathsf{v} + \mathsf{u}\mathsf{v})\ d\boldsymbol{x} \\
&= \varepsilon \sum_{i=1}^{d} \int_\Omega \frac{\partial \mathsf{u}}{\partial x_i}\frac{\partial \mathsf{v}}{\partial x_i}\ d\boldsymbol{x} + \sum_{i=1}^{d}\int_\Omega b_i \frac{\partial \mathsf{u}}{\partial x_i}\mathsf{v}\ d\boldsymbol{x} + \int_\Omega \mathsf{u}\mathsf{v}\ d\boldsymbol{x} \\
&\leq \sum_{i=1}^{d} \sqrt{\varepsilon}\left\|\frac{\partial \mathsf{u}}{\partial x_i}\right\|_{L^2(\Omega)} \sqrt{\varepsilon}\left\|\frac{\partial \mathsf{v}}{\partial x_i}\right\|_{L^2(\Omega)} + \sum_{i=1}^{d}\frac{\|b_i\|_{L^\infty(\Omega)}}{\sqrt{\varepsilon}}\sqrt{\varepsilon}\left\|\frac{\partial \mathsf{u}}{\partial x_i}\right\|_{L^2(\Omega)}\|\mathsf{v}\|_{L^2(\Omega)} \\
&\qquad + \|\mathsf{u}\|_{L^2(\Omega)}\|\mathsf{v}\|_{L^2(\Omega)} \\
&\leq \sum_{i=1}^{d} \sqrt{\varepsilon}\left\|\frac{\partial \mathsf{u}}{\partial x_i}\right\|_{L^2(\Omega)} \sqrt{\varepsilon}\left\|\frac{\partial \mathsf{v}}{\partial x_i}\right\|_{L^2(\Omega)} + \max_{i=1,\ldots,d}\frac{\|b_i\|_{L^\infty(\Omega)}}{\sqrt{\varepsilon}}\sum_{i=1}^{d}\sqrt{\varepsilon}\left\|\frac{\partial \mathsf{u}}{\partial x_i}\right\|_{L^2(\Omega)}\|\mathsf{v}\|_{L^2(\Omega)} \\
&\qquad + \|\mathsf{u}\|_{L^2(\Omega)}\|\mathsf{v}\|_{L^2(\Omega)} \\
&\leq \max\left\{1, \max_{i=1,\ldots,d}\frac{\|b_i\|_{L^\infty(\Omega)}}{\sqrt{\varepsilon}}\right\} \\
&\qquad \left(\|\mathsf{u}\|_{L^2(\Omega)} + \sum_{i=1}^{d}\sqrt{\varepsilon}\left\|\frac{\partial \mathsf{u}}{\partial x_i}\right\|_{L^2(\Omega)}\right)\left(\|\mathsf{v}\|_{L^2(\Omega)} + \sum_{i=1}^{d}\sqrt{\varepsilon}\left\|\frac{\partial \mathsf{v}}{\partial x_i}\right\|_{L^2(\Omega)}\right) \\
&\leq (d+1)\max\left\{1, \max_{i=1,\ldots,d}\frac{\|b_i\|_{L^\infty(\Omega)}}{\sqrt{\varepsilon}}\right\} \\
&\qquad \left(\|\mathsf{u}\|^2_{L^2(\Omega)} + \sum_{i=1}^{d}\varepsilon\left\|\frac{\partial \mathsf{u}}{\partial x_i}\right\|^2_{L^2(\Omega)}\right)^{1/2}\left(\|\mathsf{v}\|^2_{L^2(\Omega)} + \sum_{i=1}^{d}\varepsilon\left\|\frac{\partial \mathsf{v}}{\partial x_i}\right\|^2_{L^2(\Omega)}\right)^{1/2} \\
&= (d+1)\max\left\{1, \max_{i=1,\ldots,d}\frac{\|b_i\|_{L^\infty(\Omega)}}{\sqrt{\varepsilon}}\right\}|||\mathsf{u}|||_\Omega |||\mathsf{v}|||_\Omega,
\end{aligned}
$$

donde en la última desigualdad usamos el hecho que $\left(\sum_{i=1}^{d+1} a_i\right)^2 \leq (d+1)\sum_{i=1}^{d} a_i^2$. Luego, la continuidad se obtiene tomando la constante $\mathtt{C}_{up}$ como

$$
\mathtt{C}_{up} := (d+1)\max\left\{1, \max_{i=1,\ldots,d}\frac{\|b_i\|_{L^\infty(\Omega)}}{\sqrt{\varepsilon}}\right\}.
$$

Coercividad de $\mathcal{B}$: De la Definición 1.3 en [12], se tiene que la continuidad de la forma $\mathcal{B}$ corresponde a demostrar que, dado un espacio de Hilbert $(V, \|\cdot\|_V)$, se cumple la siguiente desigualdad

$$\mathcal{B}(\mathsf{v},\mathsf{v}) \geq \mathtt{C}_{lw}\|\mathsf{v}\|_V^2 \quad \forall\ v \in V.$$

Ahora, con respecto a nuestro problema se tiene que

$$\begin{aligned}\mathcal{B}(\mathsf{v},\mathsf{v}) &= \int_\Omega (\varepsilon\nabla\mathsf{v}\cdot\nabla\mathsf{v} + \boldsymbol{b}\cdot\nabla\mathsf{v}\ \mathsf{v} + \mathsf{v}\ \mathsf{v})\ d\boldsymbol{x}\\ &= \varepsilon\sum_{i=1}^{d}\int_\Omega\left(\frac{\partial\mathsf{v}_i}{\partial x_i}\right)^2\ d\boldsymbol{x} + \int_\Omega \boldsymbol{b}\cdot\nabla\mathsf{v}\ \mathsf{v}\ d\boldsymbol{x} + \int_\Omega \mathsf{v}^2\ d\boldsymbol{x}\end{aligned}$$

Usando la integración por partes de (3.9) para cualquier función $\mathsf{v}\in H_0^1(\Omega)$, se tiene que

$$\begin{aligned}\int_\Omega \boldsymbol{b}\cdot\nabla\mathsf{v}\ \mathsf{v}\ d\boldsymbol{x} &= \sum_{i=1}^{d}\int_\Omega \frac{\partial\mathsf{v}}{\partial x_i}b_i\mathsf{v}\ d\boldsymbol{x}\\ &= \sum_{i=1}^{d}\left(-\int_\Omega \mathsf{v}\frac{\partial}{\partial x_i}(b_i\mathsf{v})\ d\boldsymbol{x} + \underbrace{\int_{\partial\Omega}\mathsf{v}^2 b_i n_i\ ds}_{=0}\right)\\ &= \sum_{i=1}^{d} -\int_\Omega \mathsf{v}\left(\frac{\partial b_i}{\partial x_i}\mathsf{v} + b_i\frac{\partial\mathsf{v}}{\partial x_i}\right)\ d\boldsymbol{x}\\ &= -\int_\Omega \mathsf{v}^2\mathrm{div}\ (\boldsymbol{b})\ d\boldsymbol{x} - \int_\Omega \boldsymbol{b}\cdot\nabla\mathsf{v}\ \mathsf{v}\ d\boldsymbol{x},\end{aligned}$$

lo cual permite concluir que

$$\int_\Omega \boldsymbol{b}\cdot\nabla\mathsf{v}\ \mathsf{v}\ d\boldsymbol{x} = -\frac{1}{2}\int_\Omega \mathsf{v}^2\mathrm{div}\ (\boldsymbol{b})\ d\boldsymbol{x}.$$

La igualdad anterior, se puede insertar en la propiedad de coercividad, y permite reescribirla como

$$\begin{aligned}\mathcal{B}(\mathsf{v},\mathsf{v}) &= \int_\Omega (\varepsilon\nabla\mathsf{v}\cdot\nabla\mathsf{v} + \boldsymbol{b}\cdot\nabla\mathsf{v}\ \mathsf{v} + \mathsf{v}\ \mathsf{v})\ d\boldsymbol{x}\\ &= \varepsilon\sum_{i=1}^{d}\int_\Omega\left(\frac{\partial\mathsf{v}}{\partial x_i}\right)^2\ d\boldsymbol{x} + \int_\Omega \boldsymbol{b}\cdot\nabla\mathsf{v}\ \mathsf{v}\ d\boldsymbol{x} + \int_\Omega \mathsf{v}^2\ d\boldsymbol{x}\\ &= \varepsilon\sum_{i=1}^{d}\int_\Omega\left(\frac{\partial\mathsf{v}}{\partial x_i}\right)^2\ d\boldsymbol{x} + \int_\Omega\left(1-\frac{1}{2}\mathrm{div}\ \boldsymbol{b}\right)\mathsf{v}^2\ d\boldsymbol{x}.\end{aligned}$$

Ahora, haciendo el siguiente supuesto clásico (ver Capítulo 2 en [9]), de que

$$1-\frac{1}{2}\mathrm{div}\ \boldsymbol{b} \geq \mathtt{C}_b > 0, \tag{4.4}$$

entonces, usando el análisis anterior y (3.5), podemos concluir que

$$\mathcal{B}(\mathsf{v},\mathsf{v}) \geq \min\{1,\mathtt{C}_b\}\left(\varepsilon\|\nabla\mathsf{v}\|^2_{L^2(\Omega)} + \|\mathsf{v}\|^2_{L^2(\Omega)}\right) = \min\{1,\mathtt{C}_b\}|||\mathsf{v}|||^2. \tag{4.5}$$

Luego, la coercividad se obtiene tomando

$$\mathtt{C}_{lw} = \min\{1,\mathtt{C}_b\}.$$

Continuidad del funcional $\mathcal{F}$: Por último, de la Proposición 2.2.3. en [3], basta demostrar que el funcional lineal es continuo, para lo cual basta con demostrar que

$$\mathcal{F}(v) \leq \mathtt{C}\|v\|_V \quad \forall\ v \in V.$$

Ahora, en nuestro caso, se tiene que

$$\mathcal{F}(v) = \int_\Omega \mathsf{f}\mathsf{v}\ d\boldsymbol{x},$$

luego usando el hecho que $\mathsf{f} \in L^2(\Omega)$, $\mathsf{v} \in H^1_0(\Omega)$, la desigualdad (3.12) y la definición de la norma (4.3), se sigue que

$$\mathcal{F}(v) = \int_\Omega \mathsf{f}\mathsf{v}\ d\boldsymbol{x} \leq \|\mathsf{f}\|_{L^2(\Omega)}\|\mathsf{v}\|_{L^2(\Omega)} \leq \|\mathsf{f}\|_{L^2(\Omega)}|||\mathsf{v}|||_\Omega.$$

Entonces, gracias al Teorema 1 se puede concluir la existencia y unicidad de una solución débil para nuestro problema.

4.2. Formulación de Galerkin. El problema débil de la sección anterior aún se encuentra puesto en el espacio de funciones $H^1_0(\Omega)$, el cual es un espacio de dimensión infinita. Ahora, para poder aproximar dicho problema, es que utilizaremos el llamado esquema de Galerkin. Para dicho efecto, primero consideraremos una partición regular del dominio Ω, en una partición finita $\mathscr{T}$, compuesta por de elementos triangulares T, para la cual se cumple que

$$\overline{\Omega} = \bigcup_{i=1}^{\mathsf{Ne}} T_i,$$

en donde, Ne es el número total de elementos y todos ellos, son conjuntos cerrados con interior $\mathsf{Int}(T_i)$ no vacío, y además se cumple que $\mathsf{Int}(T_i) \cap \mathsf{Int}(T_j) = \emptyset$, cuando $i \neq j$ (ver Figura 4, para un ejemplo de partición para un dominio dado).

$$\overline{\Omega} = \bigcup_{i=1}^{10} T_i,$$
$$\mathcal{V}(\mathscr{T}) = \{\boldsymbol{x}_i\}_{i=1}^{10}.$$

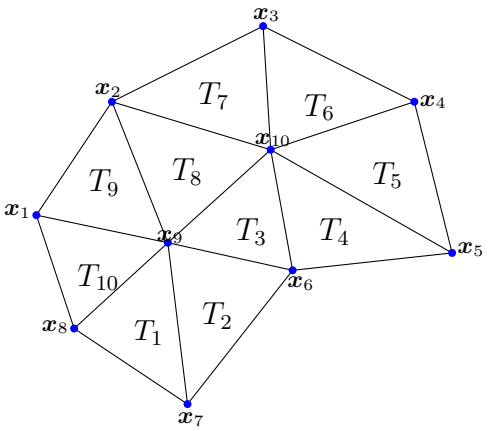

FIGURA 4. Dominio Ω, con una partición $\mathscr{T} = \bigcup_{i=1}^{10} T_i$ basada en triángulos T_i, y con un conjunto de vértices $\mathcal{V}(\mathscr{T}) = \{\boldsymbol{x}_i\}_{i=1}^{10}$.

Basados en la partición anterior es que ahora se introduce el siguiente espacio vectorial polinomial, usualmente llamado como espacio de elementos finitos Lagrangeano:

$$\mathbb{V}(\mathscr{T}) := \left\{ v \in \mathcal{C}(\overline{\Omega}) : \ v_{|T} \in \mathbb{P}_1(T), \ \forall\, T \in \mathscr{T}, \ v_{|\partial\Omega} = 0 \right\},$$

donde $\mathcal{C}(\overline{\Omega})$ corresponde al espacio de funciones continuas hasta el borde del dominio, $\mathbb{P}_1(T)$ corresponde al espacio de polinomios de grado menor o igual a uno definido sobre cada elemento T de la partición $\mathscr{T}$, y $v_{|T}$ corresponde a la restricción de la función v al elemento (triángulo T), es decir, $\mathbb{V}(\mathscr{T})$ corresponde a un espacio vectorial de todas las funciones polinomiales continuas a trozos. Un ejemplo de una función que viva es un espacio de elementos finitos como el descrito, se muestra en la Figura 5.

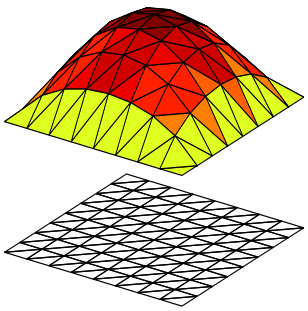

FIGURA 5. Prototipo de función de elementos finitos Lagrangeana, que corresponde a una función continua a trozos y que en cada elemento triángular es un polinomio de grado uno.

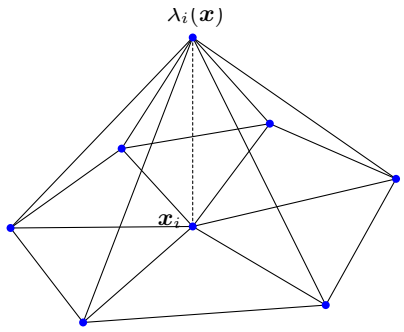

FIGURA 6. Función de base Lagrangeana $\lambda_i(\boldsymbol{x})$, asociada al nodo $\boldsymbol{x}_i \in \mathcal{V}(\mathscr{T})$.

Ahora, como el espacio de elementos finitos $\mathbb{V}(\mathscr{T})$ es un espacio vectorial, podemos encontrar una base. Para poder definir una base adecuada para dicho espacio, es que consideraremos la siguiente función polinomial de grado uno $\lambda_i(\boldsymbol{x})$, la cual se asocia a cada vértice de la partición $\boldsymbol{x}_i \in \mathcal{V}(\mathscr{T})$ y que cumple con

$$\lambda_i(\boldsymbol{x}_j) = \begin{cases} 1 & \text{si } \boldsymbol{x}_j = \boldsymbol{x}_i, \\ 0 & \text{otro caso.} \end{cases}$$

La función anterior es una función que vale uno en el nodo asociado y decae de forma lineal a todos los nodos vecinos (ver Figura 6).

Ahora, dadas dichas funciones y tomando como $\mathsf{Nv} = \#\mathcal{V}(\mathscr{T})$ es que se tiene que

$$\mathbb{V}(\mathscr{T}) = \mathsf{span}\{\lambda_1, ..., \lambda_{\mathsf{Nv}}\},$$

es decir, cualquier función que viva en $\mathbb{V}(\mathscr{T})$, puede ser escrita en términos de una combinación lineal con respecto a los elementos de la base, es decir, si $v \in \mathbb{V}(\mathscr{T})$, luego

$$v(\boldsymbol{x}) = \sum_{i=1}^{\mathsf{Nv}} \alpha_i \lambda_i(\boldsymbol{x}), \quad \text{donde } \alpha_i \in \mathbb{R},\ i = 1, ..., \mathsf{Nv}.$$

Claramente, de lo anterior se tiene que la dimensión del espacio vectorial es

$$\mathsf{dim}(\mathbb{V}(\mathscr{T})) = \#(\mathcal{V}(\mathscr{T})) = \mathsf{Nv}.$$

Ahora que se tiene un espacio de elementos finitos es que utilizamos la idea de aproximaciones de Galerkin, es decir, vamos a aproximar la solución u de nuestro problema débil (4.1), mediante un elemento $\mathsf{u}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T})$, de la forma

$$H_0^1(\Omega) \ni \mathsf{u}(\boldsymbol{x}) \approx \mathsf{u}_{\mathscr{T}}(\boldsymbol{x}) = \sum_{i=1}^{\mathsf{Nv}} \alpha_i \lambda_i(\boldsymbol{x}) \in \mathbb{V}(\mathscr{T}).$$

Luego el esquema de Galerkin corresponde a poner el problema débil, pero ahora en un espacio de dimensión finita, es decir, aproximaremos nuestro problema como sigue:

Encontrar $\mathsf{u}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T})$ tal que

$$\mathcal{B}(\mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = \mathcal{F}(\mathsf{v}_{\mathscr{T}}), \quad \text{para toda } \mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T}), \tag{4.6}$$

donde $\mathcal{B}$ y $\mathcal{F}$ están dadas en (4.29).

Notemos que hasta ahora, solamente cambiamos a un espacio discreto el cual sigue siendo un espacio de Hilbert dotado de la misma norma anterior, es que todas las propiedades de la forma bilineal y el funcional lineal siguen siendo válidas, lo que se traduce en que se tiene existencia y unicidad de una solución discreta $\mathsf{u}_{\mathscr{T}}$ para (4.6) usando los resultados de la sección anterior.

Ahora, nos centraremos en reescribir el problema anterior de una forma más adecuada. Primero, usaremos la definición de la forma bilineal y el hecho que la función de aproximación se puede escribir como una combinación lineal de los elementos de la base de $\mathbb{V}(\mathscr{T})$, es decir;

$$\begin{aligned}
\mathcal{B}(\mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) &= \mathcal{B}\left(\sum_{i=1}^{\mathsf{Nv}} \alpha_i \lambda_i \, , \, \mathsf{v}_{\mathscr{T}}\right) \\
&= \int_\Omega \varepsilon \nabla \left(\sum_{i=1}^{\mathsf{Nv}} \alpha_i \lambda_i\right) \cdot \nabla \mathsf{v} + \boldsymbol{b} \cdot \nabla \left(\sum_{i=1}^{\mathsf{Nv}} \alpha_i \lambda_i\right) + \sum_{i=1}^{\mathsf{Nv}} \alpha_i \lambda_i \mathsf{v} \; d\boldsymbol{x} \\
&= \sum_{i=1}^{\mathsf{Nv}} \alpha_i \int_\Omega \varepsilon \nabla \lambda_i \cdot \nabla \mathsf{v} + \boldsymbol{b} \cdot \nabla \lambda_i \mathsf{v} + \lambda_i \mathsf{v} \; d\boldsymbol{x} \\
&= \sum_{i=1}^{\mathsf{Nv}} \alpha_i \int_\Omega \varepsilon \nabla \lambda_i \cdot \nabla \mathsf{v} + (\boldsymbol{b} \cdot \nabla \lambda_i + \lambda_i) \mathsf{v} \; d\boldsymbol{x}.
\end{aligned}$$

Notemos ahora que el problema (4.6), se tiene que cumplir para cada $\mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T})$, que es equivalente a pedir que se cumpla para cada elemento de la base que genera al espacio $\mathbb{V}(\mathscr{T})$, es decir, utilizando la reescritura anterior, tenemos que se tiene que

cumplir el siguiente sistema de ecuaciones:

$$
\left\{
\begin{array}{rcl}
\displaystyle\sum_{i=1}^{\mathsf{Nv}} \alpha_i \int_\Omega \varepsilon \nabla\lambda_i \cdot \nabla\lambda_1 + (\boldsymbol{b}\cdot\nabla\lambda_i + \lambda_i)\lambda_1 \, d\boldsymbol{x} &=& \displaystyle\int_\Omega \mathsf{f}\lambda_1 \, d\boldsymbol{x}, \\
\displaystyle\sum_{i=1}^{\mathsf{Nv}} \alpha_i \int_\Omega \varepsilon \nabla\lambda_i \cdot \nabla\lambda_2 + (\boldsymbol{b}\cdot\nabla\lambda_i + \lambda_i)\lambda_2 \, d\boldsymbol{x} &=& \displaystyle\int_\Omega \mathsf{f}\lambda_2 \, d\boldsymbol{x}, \\
& \vdots & \\
\displaystyle\sum_{i=1}^{\mathsf{Nv}} \alpha_i \int_\Omega \varepsilon \nabla\lambda_i \cdot \nabla\lambda_{\mathsf{Nv}} + (\boldsymbol{b}\cdot\nabla\lambda_i + \lambda_i)\lambda_{\mathsf{Nv}} \, d\boldsymbol{x} &=& \displaystyle\int_\Omega \mathsf{f}\lambda_{\mathsf{Nv}} \, d\boldsymbol{x}.
\end{array}
\right. \tag{4.7}
$$

Entonces, definiendo una matriz $\mathbf{B}$ de tamaño $\mathsf{Nv} \times \mathsf{Nv}$, la cual tiene como entradas

$$
(\mathbf{B})_{ij} = \int_\Omega \varepsilon \nabla\lambda_j \cdot \nabla\lambda_i + (\boldsymbol{b}\cdot\nabla\lambda_j + \lambda_j)\lambda_i \, d\boldsymbol{x}, \quad \forall\ i,j = 1,...,\mathsf{Nv}, \tag{4.8}
$$

y de igual forma definiendo los vectores $\boldsymbol{\alpha}$ y $\mathbf{f}$, de tamaño $\mathsf{Nv} \times 1$, como

$$
\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{\mathsf{Nv}} \end{bmatrix} \quad \text{y} \quad \mathbf{f} = \begin{bmatrix} \displaystyle\int_\Omega \mathsf{f}\lambda_1 \, d\boldsymbol{x} \\ \vdots \\ \displaystyle\int_\Omega \mathsf{f}\lambda_{\mathsf{Nv}} \, d\boldsymbol{x} \end{bmatrix}, \tag{4.9}
$$

se puede concluir que el problema (4.6), es equivalente a resolver el siguiente problema matricial:

Encontrar $\alpha_1, ..., \alpha_{\mathsf{Nv}}$ tal que

$$
\mathbf{B}\boldsymbol{\alpha} = \mathbf{f}. \tag{4.10}
$$

Notemos que, de la propiedad de coercividad de la forma $\mathcal{B}$, se tiene que para cualquier elemento

$$
\mathbb{V}(\mathscr{T}) \ni \mathsf{v}_{\mathscr{T}} = \sum_{i=1}^{\mathsf{Nv}} \tilde{\alpha}_i \lambda_i(\boldsymbol{x}),
$$

se cumple que, definiendo como $\tilde{\boldsymbol{\alpha}} = (\tilde{\alpha}_1, ..., \tilde{\alpha}_{\mathsf{Nv}})$,

$$
0 < \mathsf{C}_{lw} \|\mathsf{v}_{\mathscr{T}}\|_\Omega^2 \leq \mathcal{B}(\mathsf{v}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = \tilde{\boldsymbol{\alpha}}\mathbf{B}\tilde{\boldsymbol{\alpha}}^T,
$$

es decir, lo anterior nos dice que la matriz de elementos finitos es semidefinida positiva, que es equivalente a decir que *es invertible (su determinante es positivo), y su inversa es definida positiva.*

Para poder resolver el problema lineal anterior, tenemos que ensamblar la matriz $\mathbf{B}$ y el vector del lado derecho del sistema $\mathbf{f}$, usando (4.8) y (4.9), respectivamente.

Ahora, para efectuar dicho cálculo es que nos basaremos en el siguiente simple hecho que para todo $i, j = 1, ..., \mathsf{Nv}$ se cumple que

$$\begin{aligned}&\int_\Omega \varepsilon \nabla\lambda_j \cdot \nabla\lambda_i + (\boldsymbol{b}\cdot\nabla\lambda_j + \lambda_j)\lambda_i \; d\boldsymbol{x} \\ &= \sum_{T\in\mathscr{T}} \int_T \varepsilon \nabla\lambda_{j|T} \cdot \nabla\lambda_{i|T} + (\boldsymbol{b}\cdot\nabla\lambda_{j|T} + \lambda_{j|T})\lambda_{i|T} \; d\boldsymbol{x},\end{aligned}$$

donde la integral en todo el domino se transformó en una suma de integrales por todos los elementos de la partición $\mathscr{T}$. Luego, podemos hacer un cálculo local por cada elemento T. Para ésto, utilizaremos formulas para las tres funciones de base asociadas a cada vértice de un elemento (ver Figura (7)). Ahora, basados en la

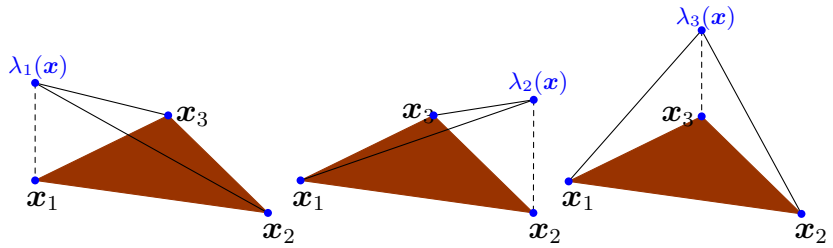

FIGURA 7. Prototipo de elemento tringular T, compuesto por sus tres vértices $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ y $\boldsymbol{x}_1$, con sus respectivas tres funciones de base asociadas a cada vértice $\lambda_1(\boldsymbol{x})$, $\lambda_2(\boldsymbol{x})$ y $\lambda_3(\boldsymbol{x})$.

Figura 8, que da un ordenamiento usual anti horario de elementos finitos, se tiene que cada función de base restringida al elemento T, está dada por

$$\lambda_i(x,y)_{|T} = 1 - \frac{(x-x_i, y-y_i)\cdot \boldsymbol{n}_i}{2|T|} \quad \text{para todo } i = 1,2,3, \tag{4.11}$$

donde $|T|$ denota el área del triángulo $T \in \mathscr{T}$. Claramente, para el cálculo de la primera integral es necesario tener los gradientes de dichas funciones, el cual, usando la expresión anterior, es claramente

$$\nabla\lambda_i(x,y)_{|T} = -\frac{1}{2|T|}\boldsymbol{n}_i \quad \text{para todo } i = 1,2,3. \tag{4.12}$$

Finalmente, asumiendo que el campo convectivo $\boldsymbol{b} \in \mathbb{R}^d$, podemos usar una formula de integración que es exacta para polinomios de grado dos en triángulos, que corresponde a una extensión de la conocida regla de Simpson en una dimensión

$$\int_T \mathsf{p}(\boldsymbol{x}) \; d\boldsymbol{x} = \frac{|T|}{3}\sum_{i=1}^{3} \mathsf{p}(\boldsymbol{x}_{M_i}) \quad \forall\; \mathsf{p}(\boldsymbol{x}) \in \mathbb{P}_2(T),$$

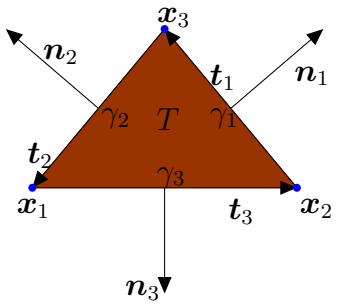

FIGURA 8. Prototipo de elemento triangular T, compuesto por: sus tres vértices $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ y $\boldsymbol{x}_1$; tres lados γ_1, γ_2 y γ_3, opuestos a sus respetivos vértices; vectores tangentes $\boldsymbol{t}_1 = \boldsymbol{x}_3 - \boldsymbol{x}_2$, $\boldsymbol{t}_2 = \boldsymbol{x}_1 - \boldsymbol{x}_2$ y $\boldsymbol{t}_3 = \boldsymbol{x}_2 - \boldsymbol{x}_1$; y tres vectores normales $\boldsymbol{n}_1$, $\boldsymbol{n}_2$ y $\boldsymbol{n}_3$ que se obtiene al rotar los tangentes en noventa grados en sentido horario.

donde $\boldsymbol{x}_{M_i}$ corresponde al punto medio del lado γ_i, para $i = 1, 2, 3$ y $\mathbb{P}_2(T)$ es el conjunto de todos los polinomios de grado menor o igual a dos. Finalmente, usando el hecho que cada función de base es lineal y en cada uno de sus nodos asociados vale uno, luego $\lambda_i(\boldsymbol{x}_{M_j}) = 1/2$ cuando $i, j = 1, 2, 3$ y $i \neq j$, y vale cero en caso contrario, y además asumiendo que el campo vectorial $\boldsymbol{b}$ es constante, es decir, $\boldsymbol{b} \in \mathbb{R}^2$, podemos concluir que para todo $T \in \mathscr{T}$,

(4.13)

$$\int_T \varepsilon \nabla \lambda_{j|T} \cdot \nabla \lambda_{i|T} + (\boldsymbol{b} \cdot \nabla \lambda_{j|T} + \lambda_{j|T}) \lambda_{i|T} \, d\boldsymbol{x} = \begin{cases} \varepsilon \dfrac{\boldsymbol{n}_j \cdot \boldsymbol{n}_i}{4|T|} - \dfrac{\boldsymbol{b} \cdot \boldsymbol{n}_j}{6} + \dfrac{|T|}{6} & \text{si } i = j, \\ \varepsilon \dfrac{\boldsymbol{n}_j \cdot \boldsymbol{n}_i}{4|T|} - \dfrac{\boldsymbol{b} \cdot \boldsymbol{n}_j}{6} + \dfrac{|T|}{12} & \text{si } i \neq j. \end{cases}$$

En un caso más general, es que podemos hacer uso de las siguentes formulas:

$$\int_T \lambda_i^l \lambda_j^m \lambda_k^n \, d\boldsymbol{x} = \frac{2(l!m!n!)}{(l+m+n+2)} |T|, \tag{4.14}$$

donde $l, m, n \geq 0$ con enteros positivos y los vértices del elemento T corresponden a $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k$. De forma similar, en caso que algún problema requiera de una integración, en un lado que sea parte del borde de un elemento o de la frontera del dominio, es que se cumple que

$$\int_\gamma \lambda_i^l \lambda_j^m \, ds = \frac{l!m!}{(l+m+1)} |\gamma|, \tag{4.15}$$

con l, m enteros positivos.

Finalmente tenemos que calcular el lado derecho, el cual puede ser calculado con la ayuda de alguna cuadratura Gaussiana, de la forma

$$\int_T \mathsf{f}(\boldsymbol{x})\lambda_i(\boldsymbol{x})\, d\boldsymbol{x} \approx \sum_{i=1}^{\mathsf{N}_G} \omega_i \mathsf{f}(\boldsymbol{x}_i)\lambda_i(\boldsymbol{x}_i), \tag{4.16}$$

donde ω_i y $\boldsymbol{x}_i$ para $i = 1, ..., \mathsf{N}_G$, son pesos y puntos de integración de Gauss, respectivamente (ver Sección 8.1 en [9]).

4.3. Convergencia del método de elementos finitos. Para obtener un resultado de convergencia, necesitamos medir el error en alguna norma, entre la diferencia entre u que resuelve

$$\mathsf{u} \in H_0^1(\Omega): \quad \mathcal{B}(\mathsf{u}, \mathsf{v}) = \mathcal{F}(\mathsf{v}) \quad \forall\, \mathsf{v} \in H_0^1(\Omega), \tag{4.17}$$

y su aproximación discreta de elementos finitos $\mathsf{u}_{\mathscr{T}}$ que resuelve

$$\mathsf{u}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T}): \quad \mathcal{B}(\mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = \mathcal{F}(\mathsf{v}_{\mathscr{T}}) \quad \forall\, \mathsf{v} \in \mathbb{V}(\mathscr{T}). \tag{4.18}$$

Como $\mathbb{V}(\mathscr{T}) \subset H_0^1(\Omega)$, podemos tomar en (4.17) la función $\mathsf{v} = \mathsf{v}_{\mathscr{T}}$, y después podemos restar (4.17) de (4.18), para así obtener, usando la bilinealidad de la forma $\mathcal{B}$,

$$\mathcal{B}(\mathsf{u} - \mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = \mathcal{B}(\mathsf{u}, \mathsf{v}_{\mathscr{T}}) - \mathcal{B}(\mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = \mathcal{F}(\mathsf{v}_{\mathscr{T}}) - \mathcal{F}(\mathsf{v}_{\mathscr{T}}) = 0,$$

y como la función discreta es cualquiera, entonces se obtiene lo que usualmente se denomina como ortogonalidad de Galerkin:

$$\mathcal{B}(\mathsf{u} - \mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = 0 \quad \forall\, \mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T}). \tag{4.19}$$

Para medir error, ya que nos encontramos en un marco funcional, es que se tendrá que medir el error en la norma que se definió en (4.3), es decir, queremos medir el error en

$$\|\!|\mathsf{u} - \mathsf{u}_{\mathscr{T}}|\!\|_{\Omega}.$$

Recordemos ahora, un resultado que se utilizó para la demostración de existencia y unicidad, que correspondía a la propiedad de coercividad de la forma $\mathcal{B}$:

$$\mathsf{C}_{lw}\|\!|\mathsf{v}|\!\|_{\Omega}^2 \leq \mathcal{B}(\mathsf{v}, \mathsf{v}) \quad \forall\, \mathsf{v} \in H_0^1(\Omega), \tag{4.20}$$

y de igual forma se utilizó lo que se denominó como continuidad de la forma, es decir,

$$\mathcal{B}(\mathsf{v}, \mathsf{w}) \leq \mathsf{C}_{up}\|\!|\mathsf{v}|\!\|_{\Omega}\|\!|\mathsf{w}|\!\|_{\Omega} \quad \forall\, \mathsf{v}, \mathsf{w} \in H_0^1(\Omega). \tag{4.21}$$

Ahora, tenemos los ingredientes para obtener un primer resultado de convergencia, denominado como Lema de Cea. Primero, tomamos $\mathsf{v} = \mathsf{u} - \mathsf{u}_{\mathscr{T}}$ en (4.20) y hacemos

uso de la ortogonalidad de Galerkin dos veces con cualquier función $\mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T})$ como sigue

$$\begin{aligned}
\mathsf{C}_{lw}\|\mathsf{u}-\mathsf{u}_{\mathscr{T}}\|_{\Omega}^2 &\leq \mathcal{B}(\mathsf{u}-\mathsf{u}_{\mathscr{T}},\mathsf{u}-\mathsf{u}_{\mathscr{T}}) \quad [\text{Coercividad}]\\
&= \mathcal{B}(\mathsf{u}-\mathsf{u}_{\mathscr{T}},\mathsf{u}) - \underbrace{\mathcal{B}(\mathsf{u}-\mathsf{u}_{\mathscr{T}},\mathsf{u}_{\mathscr{T}})}_{=0} \quad [\text{Ortogonalidad}]\\
&= \mathcal{B}(\mathsf{u}-\mathsf{u}_{\mathscr{T}},\mathsf{u}) - \underbrace{\mathcal{B}(\mathsf{u}-\mathsf{u}_{\mathscr{T}},\mathsf{v}_{\mathscr{T}})}_{=0} \quad [\text{Ortogonalidad}]\\
&= \mathcal{B}(\mathsf{u}-\mathsf{u}_{\mathscr{T}},\mathsf{u}-\mathsf{v}_{\mathscr{T}})\\
&\leq \mathsf{C}_{up}\|\mathsf{u}-\mathsf{u}_{\mathscr{T}}\|_{\Omega}\|\mathsf{u}-\mathsf{v}_{\mathscr{T}}\|_{\Omega}, \quad [\text{Continuidad}]
\end{aligned}$$

lo que permite concluir un primer resultado de cuasi mejor aproximabilidad

$$\|\mathsf{u}-\mathsf{u}_{\mathscr{T}}\|_{\Omega} \leq \frac{\mathsf{C}_{up}}{\mathsf{C}_{lw}} \inf_{\mathsf{v}_{\mathscr{T}}\in\mathbb{V}(\mathscr{T})} \|\mathsf{u}-\mathsf{v}_{\mathscr{T}}\|_{\Omega}. \tag{4.22}$$

Ahora, para poder obtener el resultado final de convergencia, es que usaremos el siguiente resultado de interpolación (ver ecuación (1.102) en [9]).

Teorema. Dado un dominio lo suficientemente suave Ω y una partición regular $\mathscr{T}$ del dominio, luego existe un operador de interpolación $I_L : H_0^1(\Omega) \to \mathbb{V}(\mathscr{T})$ tal que

$$\|v - I_L(v)\|_{L^2(\Omega)} + h\|\nabla(v - I_L(v))\|_{L^2(\Omega)} \leq \mathsf{C}_L h^2 |v|_{H^2(\Omega)} \ \forall\ v \in H^2(\Omega), \tag{4.23}$$

donde el espacio $H^2(\Omega)$ está definido en (3.6), y la seminorma es

$$|v|_{H^2(\Omega)} := \left(\sum_{i,j=1}^{d} \left\| \frac{\partial^2 v}{\partial x_i \partial x_j} \right\|_{L^2(\Omega)}^2 \right)^{1/2},$$

y definiendo, basados en la Figura 8, el diámetro de cada elemento $T \in \mathscr{T}$, como

$$h_T = \max_{i\in\{1,2,3\}} \{|\gamma_i|\}, \tag{4.24}$$

luego el largo característico de la partición $\mathscr{T}$ se define como

$$h := \max_{T\in\mathscr{T}} \{h_T\}. \tag{4.25}$$

Finalmente, tomando como $v_{\mathscr{T}} = I_L(\mathsf{u})$ en (4.22), y usando el Teorema anterior, podemos concluir que

$$\begin{aligned}
\|\mathsf{u} - \mathsf{u}_{\mathscr{T}}\|_\Omega &\leq \frac{\mathtt{C}_{up}}{\mathtt{C}_{lw}} \inf_{\mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T})} \|\mathsf{u} - \mathsf{v}_{\mathscr{T}}\|_\Omega \\
&\leq \frac{\mathtt{C}_{up}}{\mathtt{C}_{lw}} \|\mathsf{u} - I_L(\mathsf{u})\|_\Omega \\
&= \frac{\mathtt{C}_{up}}{\mathtt{C}_{lw}} \left(\varepsilon \|\mathsf{u} - I_L(\mathsf{u})\|^2_{L^2(\Omega)} + \|\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(\Omega)} \right)^{1/2} \\
&\leq \frac{\mathtt{C}_{up}}{\mathtt{C}_{lw}} \left(\varepsilon \mathtt{C}_L^2 h^4 |v|^2_{H^2(\Omega)} + \mathtt{C}_L^2 h^2 |v|^2_{H^2(\Omega)} \right)^{1/2} \\
&= \frac{\mathtt{C}_{up}}{\mathtt{C}_{lw}} \mathtt{C}_L h |v|_{H^2(\Omega)} (\varepsilon h^2 + 1)^{1/2}.
\end{aligned}$$

Entonces, se tiene el siguiente resultado. Para un dominio lo suficientemente suave para el cual la solución débil $\mathsf{u} \in H^2(\Omega)$, luego su aproximación de elementos finitos satisface la siguiente relación de convergencia:

$$\|\mathsf{u} - \mathsf{u}_{\mathscr{T}}\|_\Omega \leq \mathtt{C}_{con}\, h. \tag{4.26}$$

Notemos que el resultado anterior nos dice que si hacemos el largo característico de la partición cada vez más pequeño, la solución de elementos finitos converge con orden lineal a la solución del problema débil.

4.4. Un código de elementos finitos en C/C++. A continuación se presenta la descripción en pseudocódigo del algoritmo utilizado en la resolución de un problema de elementos finitos realizado en C/C++ para aproximar la solución del problema (3.1), mediante el esquema de Galerkin descrito en (3.6).

Algoritmo 1: Función main()

```
ndof = 0
maxndof = 50 (arbitrario)
while ndof ≤ maxndof do
    ndof = run_prog()
end
return 0
```

Algoritmo 2: Función run_prog()

```
Lectura de malla
Cálculo de matriz y lado derecho para resolver ecuación B α = f
Cálculo de error
Guardado de resultados
Generar visualización de resultados en ParaView
Refinar Malla
```

Algoritmo 3: Resolución del sistema de ecuaciones

```
begin
    Cálculo de áreas de triángulos: get_normals_areas()
    Armado de lado derecho: get_RHS()
    Ensamble de matriz: get_stiffnessmatrix()
    Resolución del sistema
end
```

Todo esto se corresponde con el siguiente código de las principales funciones del programa:

```
int main(){
    std::ofstream outfile;
    outfile.open("results.txt", std::ios::app);
    outfile.precision(16);
    outfile.close();
    int ndof=0;
    int maxndof=50;
    while (ndof<maxndof){ ndof=run_prog(maxndof,ndof);}
    return 0;
}
```

LISTING 1. Función main

```
int run_prog(int maxndof,int ndofant){
    int nv,nt,nbf;
    MatrixXd vertices;
    VectorXi previous, subdomain, boundary_subdomain;

    MatrixXi elements, boundary;
    lee_mall(nv,nt,nbf,vertices,previous,subdomain,elements,boundary_subdomain,boundary);

    MatrixXi edgeelement(nt,3); MatrixXi edgeedge(nt,3);
    get_neighbours(nv,nt,elements,nbf,boundary_subdomain,boundary,edgeelement,edgeedge);

    VectorXi Dnodes(nv), Nnodes(nv);
    boundary_nodes(nv,nt,nbf,boundary_subdomain,boundary,Dnodes,Nnodes);

    Matrix<double, 3, 2>* normals= new Matrix<double, 3, 2>[nt];
    VectorXd areas(nt);
    get_normals_areas(elements, vertices, normals,areas,nt);

    std::vector<double> RHS(nv);
    get_RHS(nt,subdomain,elements,vertices,nbf,boundary_subdomain,boundary,RHS,normals,areas)
        ;

    VectorXi nnzpc(get_ss(nv,nt));
    get_nnzpc(nv,nt,elements,nbf,boundary,nnzpc,edgeelement,boundary_subdomain,Dnodes);

    SparseMatrix<double> K(get_ss(nv,nt),get_ss(nv,nt));
    K.reserve(nnzpc);
    get_stiffnessmatrix(nv,nbf,boundary_subdomain,boundary,nt,elements,vertices,RHS,K,
        edgeelement,edgeedge,normals,areas,Dnodes);
    MUMPS(K,RHS);

    std::ofstream outfile;
    outfile.open("results.txt", std::ios::app);
    outfile.precision(16);
    outfile << nv << " " << nt;
    outfile << std::endl;
    outfile.close();
    int ndof=nv;
    if (ndof>=maxndof){ write_solution_paraview(nv, nt, RHS,vertices,elements); }
    else { Rmeshrefine(nv,vertices,nt,previous,subdomain,elements,nbf,boundary_subdomain,
        boundary,refine,edgeelement,edgeedge); }
    delete [] normals;
    return ndof;
}
```

LISTING 2. Función run_prog

Variable	Descripción
nn	Número de nodos de la malla.
vertices(i,j)	Coordenadas x e y de cada nodo i.
nt	Número de elementos de la malla.
previous(i)	Identificador de cada triángulo i en la iteración previa.
subdomain(i)	Identificador de subdominio para cada elemento i, para efectos de marcado (Ej: un punto fuente).
elements(i,0)(i,1)(i,2)	Identificador de cada vértice del triángulo i, leídos en sentido antihorario.
nbf	Número de caras que se encuentran en el borde de la malla.
boundary_subdomain(i)	Identificador de subdominio para cada cara en el borde i.
boundary(i,0)(i,1)	Identificador de los vértices de cada cara en el borde i leídos en sentido antihorario.

CUADRO 1. Listado de variables obtenidas de la lectura de la malla y su descripción.

Puede notarse que el propósito de la función `main` es de establecer el máximo de grados de libertad a obtener y de llamar a la función `run_prog` mientras ese límite no se alcance.

Por otro lado, la función `run_prog` se encarga de realizar en cada iteración las operaciones necesarias para resolver el problema:

- En las líneas 2-4 se realiza la declaración de variables mencionada en el próximo apartado, en la tabla 1
- En la línea 6 se declaran dos matrices que servirán para posterior identificación de los elementos a calcular. En la línea 7 se realiza la lectura de los datos iniciales provenientes de un archivo de texto base.
- En las línea 9 se declaran dos vectores de enteros de tamaño nv que guardan la información si el nodo de la frontera es del tipo *Dirichlet* o *Neumann*. Luego, se llama a la función `boundary_nodes` que verifica la existencia de elementos (triángulos) vecinos a cada nodo.
- En el bloque siguiente (líneas 15 - 17) se declara un puntero a una matriz de dimensión $3 \times 2 \times nt$ donde se almacenará el valor de cada vector normal y un vector de dimensión nt donde se almacenará el valor de las áreas de cada triángulo. Posteriormente se realiza la llamada a la función `get_normals_areas` que realiza el cálculo de estas áreas.
- Posteriormente se declara un vector de dobles de dimensión nv llamado RHS *(Right Hand Side)* y se llama a la función `get_rhs` que realiza el cálculo del lado derecho de la ecuación (ver (4.16)).

- En las líneas 21-22 se determina el número de elementos no-cero por cada columna en la matriz principal, hecho de importancia ya que al utilizarse una matriz *sparse* es necesario obtener una aproximación de la cantidad de elementos de ésta con el fin de optimizar los recursos de cómputo.
- Luego, se declara la matriz *sparse* K y se realiza el ensamble de la matriz (ver (4.13) - (4.14)), para posteriormente ser resuelta junto con el vector RHS.
- En las líneas 30-35 se realiza un guardado del número de vértices y número de triángulos del sistema. En la línea 36 se actualiza la cantidad de grados de libertad alcanzados (nv), luego se genera un gráfico de la solución visible en ParaView, condicionado al número máximo de grados de libertad. En el caso de no ser alcanzado, se refina la malla, se elimina el vector de normales y la función llega a su término, retornando la cantidad de grados de libertad alcanzados en esa iteración.

```
1  void lee_mall(int& nv, int& nt, int& nbf,MatrixXd& vertices,VectorXi& previous,VectorXi&
       subdomain,MatrixXi& elements,VectorXi& boundary_subdomain,MatrixXi& boundary){
2      std::ifstream infile("mesh.txt");
3      infile >> nv;
4      vertices.resize(nv,2);
5      for (int i=0; i<nv; i++) {
6          infile >> vertices(i,0) >> vertices(i,1);
7      }
8
9      infile >> nt;
10     previous.resize(nt);
11     subdomain.resize(nt);
12     elements.resize(nt,3);
13     for (int i=0; i<nt; i++) {
14         infile >> previous(i) >> subdomain(i) >> elements(i,0) >> elements(i,1) >> elements(i
               ,2);
15     }
16
17     infile >> nbf;
18     boundary_subdomain.resize(nbf);
19     boundary.resize(nbf,2);
20     for (int i=0; i<nbf; i++) {
21         infile >> boundary_subdomain(i) >> boundary(i,0) >> boundary(i,1);
22
23     }
24     infile.close();
25 }
```

LISTING 3. Función lee_mall

En esta función se realiza la lectura de los datos existentes en un archivo de texto (en este caso, `mesh.txt`). Estos datos cuentan con el formato de la figura 10 que contiene la información para una malla básica de 5 vértices y 4 elementos, tal como se muestra en la figura 9.

- La función realiza la apertura del archivo, correspondiendo la primera lectura al número de vértices nv. A partir de este valor se le da la dimensión apropiada a la matriz `vertices` (que es pasada por referencia a la función). Posteriormente, se leen las coordenadas x e y de cada vértice.
- En el bloque siguiente se lee el número de elementos de la malla (nt), información necesaria para establecer el tamaño de los vectores `previous`, `subdomain` y la matriz `elements`. Luego, (líneas 13-14) se leen secuencialmente el identificador del elemento, el subdominio del elemento (si corresponde), y el número de los vértices del elemento, para ser almacenados en la matriz en sentido antihorario.
- Finalmente, se lee el número de caras en el borde, se ajusta la dimensión de la matriz `boundary`, y se almacenan secuencialmente el subdominio del eje y los vértices de los extremos de cada eje.

A partir de este proceso de lectura se obtienen los valores para las variables enumeradas en la tabla 1.

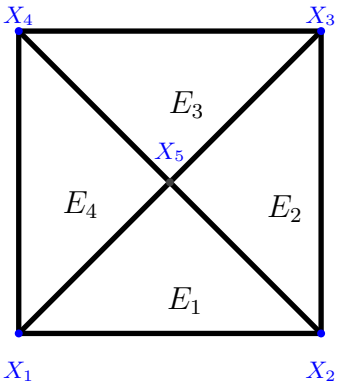

FIGURA 9. Ejemplo de malla inicial

```
5
0 0
1 0
1 1
0 1
0.5 0.5
4
0 1 1 2 5
1 1 2 3 5
2 1 3 4 5
3 1 4 1 5
4
1 1 2
2 2 3
3 3 4
4 4 1
```

FIGURA 10. Información de la malla inicial.

```
1  void get_normals_areas(const MatrixXi& elements,const MatrixXd& vertices,Matrix<double, 3, 2>
       normals[],VectorXd& areas,int nt){
2      for (int k=0; k<nt; k++) {
3        normals[k] << vertices(elements(k,2),1)-vertices(elements(k,1),1), vertices(elements(k
             ,1),0)-vertices(elements(k,2),0),
4                      vertices(elements(k,0),1)-vertices(elements(k,2),1), vertices(elements(k
                           ,2),0)-vertices(elements(k,0),0),
5                      vertices(elements(k,1),1)-vertices(elements(k,0),1), vertices(elements(k
                           ,0),0)-vertices(elements(k,1),0);
6            areas(k)=(normals[k](0,1)*normals[k](2,0)-normals[k](0,0)*normals[k](2,1))/2;
7        }
8  }
```

LISTING 4. Función get_normals_areas

Esta función realiza el cálculo de los vectores tangente y normal para cada lado del triángulo, tomando en cuenta las siguientes consideraciones:

En base al esquema de la figura 8, se observa que en cada triángulo T cada vértice $\boldsymbol{x}_i$ tiene un vector tangente $\boldsymbol{t}_i$, siendo necesario calcular el valor de cada vector normal $\boldsymbol{n}_i$.

De esta forma, el código recorre cada triángulo y almacena los datos de cada vector normal del triángulo. Posteriormente, y a partir del cálculo de los vectores normales se obtiene el valor del área de cada triángulo mediante el producto vectorial entre dos vectores normales, por ejemplo, en el caso de la figura 8:

$$|T| = \frac{1}{2}\left(\boldsymbol{n}_1 \times -\boldsymbol{n}_3\right)$$

```
1  void get_stiffnessmatrix(int nv,int nbf,const VectorXi& boundary_subdomain,const MatrixXi&
       boundary,int nt,const MatrixXi& elements,const MatrixXd& vertices,std::vector<double>&
       RHS,SparseMatrix<double>& K,const MatrixXi& edgeelement,const MatrixXi& edgeedge,const
       Matrix<double, 3, 2> normals[],const VectorXd& areas,const VectorXi& Dnodes){
2      for (int k=0; k<nt; k++) {
3          for (int local_node_a=0; local_node_a<3; local_node_a++) {
4              if (Dnodes(elements(k,local_node_a))>=0){
5                  for (int local_node_b=0; local_node_b<3; local_node_b++) {
6                      double sum=0;
7                      double val=(normals[k].row(local_node_a).dot(normals[k].row(local_node_b)
                           )/(4*areas(k)))*get_epsilon();
8                      sum+=val;
9                      double int_B=0.0;
10                     for (int t=0;t<3;t++){
11                         Vector2d local_coordinate;
12                         local_coordinate(0)=vertices(elements(k,t),0);
13                         local_coordinate(1)=vertices(elements(k,t),1);
14                         Vector2d B=get_b(local_coordinate);
15                         if (t==local_node_a){
16                             int_B+=-B.dot(normals[k].row(local_node_b))/12.0;
17                         }
18                         else{
19                             int_B+=-B.dot(normals[k].row(local_node_b))/24.0;
20                         }
21                     }
22                     sum+=int_B;
23                     if (elements(k,local_node_a)==elements(k,local_node_b)){
24                         sum+=areas(k)/6.0;
25                     }
26                     else{
27                         sum+=areas(k)/12.0;
28                     }
29                     if (Dnodes(elements(k,local_node_b))>=0) {
30                         K.coeffRef(elements(k,local_node_a),elements(k,local_node_b))+=sum;
31                     }
32                     else{
33                         RHS[elements(k,local_node_a)]-=sum*get_uD(vertices.row(elements(k,
                               local_node_b)),-Dnodes(elements(k,local_node_b)));
34                     }
35                 }
36             }
37             else {
38                 K.coeffRef(elements(k,local_node_a),elements(k,local_node_a))=1;
39                 RHS[elements(k,local_node_a)]=get_uD(vertices.row(elements(k,local_node_a)),-
                       Dnodes(elements(k,local_node_a)));
40             }
41         }
42     }
43 }
```

LISTING 5. Función get_stiffnessmatrix

Esta función se encarga de realizar el ensamble de la matriz B de acuerdo al esquema presentado en las fórmulas 4.9 y 4.13. En el código se ve que recorre cada triángulo (línea 3), cada nodo del k-ésimo triángulo (línea 4), para revisar algún vértice tiene condiciones Dirichlet (línea 5). En caso de que el nodo lo sea, se recorre cada nodo (línea 6) para calcular el elemento correspondiente de la matriz (línea 8) como se indica en 4.13.

Si al recorrer alguno de los nodos del triángulo, se encuentra con alguno que sea Dirichlet (línea 10), se suma el valor calculado anteriormente (línea 9) en la posición

correspondiente a la matriz(línea 11). En el caso de que haya un nodo Neumann (línea 13), se resta al valor del lado derecho el valor calculado en la linea 9, multiplicado por el valor de la función u (línea 14).

Alternativamente, al fallar la condición de la línea 5 (línea 18), se coloca el valor 1 en la matriz (línea 19) para forzar que la matriz resuelva esa ecuación igual a cero.

```
1  void get_RHS(int nt,const VectorXi& subdomain,const MatrixXi& elements,const MatrixXd&
       vertices,int nbf,const VectorXi& boundary_subdomain,const MatrixXi& boundary,std::vector<
       double>& RHS, const Matrix<double, 3, 2> normals[],const VectorXd& areas){
2      for (int k=0; k<nt; k++) {
3          Vector3d fmom = Vector3d::Zero();
4          for (int j=0; j<NTRIQP; j++) {
5              Vector2d X=triquad[j][0]*vertices.row(elements(k,0))+triquad[j][1]*vertices.row(
                   elements(k,1))+triquad[j][2]*vertices.row(elements(k,2));
6              double w_f=triweight[j]*get_f(X,subdomain(k));
7              for (int local_node=0; local_node<3; local_node++) {
8                  fmom(local_node)+=w_f*triquad[j][local_node];
9              }
10         }
11         for (int local_node=0; local_node<3; local_node++) {
12             RHS[elements(k,local_node)]+=areas(k)*fmom(local_node);
13         }
14     }
15     for (int i=0; i<nbf; i++){
16         if (get_Dirichlet(boundary_subdomain(i))==0){
17             double h_F=get_hf(vertices.row(boundary(i,0)),vertices.row(boundary(i,1)));
18             Vector2d du_n_mom = Vector2d::Zero();
19             for (int j=0; j<NLINQP; j++){
20                 Vector2d X=linquad[j][0]*vertices.row(boundary(i,0))+linquad[j][1]*vertices.
                       row(boundary(i,1));
21                 double w_du_n=linweight[j]*get_uN(X,boundary_subdomain(i));
22                 for (int local_node=0;local_node<2;local_node++){
23                     du_n_mom(local_node)+=w_du_n*linquad[j][local_node];
24                 }
25             }
26             for (int local_node=0;local_node<2;local_node++){
27                 RHS[boundary(i,local_node)]+=h_F*du_n_mom(local_node);
28             }
29         }
30     }
31 }
```

LISTING 6. Función get_RHS

- En esta función se calcula el vector **f**, a partir de la fórmula mostrada en la ecuación 4.16, donde se recorren los elementos de la malla (línea 2) y para cada elemento se calcula el valor de **f** mediante el cálculo del producto de los puntos de integración, los pesos de Gauss (almacenados como constantes en un archivo de headers) por el valor de la función (líneas 4 - 8).
- Luego, en la línea 11 se recorre cada nodo en el triángulo y se suma (línea 12) el valor calculado anteriormente en la línea 8.
- Posteriormente, se recorren los vértices que se encuentran en la frontera de la malla y se verifica si son nodos Dirichlet (líneas 15 y 16) con el objeto de realizar el cálculo de su aporte al vector **f**, esta vez en la frontera de la malla.

4.5. Ejemplos numéricos. En esta sección, nos ocuparemos de testear nuestro código de elementos finitos, con un ejemplo en donde construiremos una solución análitica y otros dos ejemplos, sin solución analítica, pero con una aplicación física, como modelamiento de contaminantes y temperatura.

Ejemplo 1: Consideraremos como dominio $\Omega = (0,1) \times (0,1)$, $\varepsilon = 1$, $\boldsymbol{b} = [1,0]^T$ y como solución real

$$\mathsf{u}(x,y) = xy(1-x)(1-y).$$

Con dicha función, es que podemos calcular el lado derecho f, utilizando (1.2). Hacemos notar que para el cálculo del lado derecho utilizamos una regla de cuadratura con 73 puntos de Gauss y que se cumple la siguiente relación entre el largo caracteristico de la partición y el número total de vértices

$$h \approx \frac{1}{\mathsf{Nv}^{1/2}}.$$

En la Figura 11 se muestran una serie de soluciones de elementos finitos para el problema en distintas mallas y en la Figura 12, se muestra la tasa de convergencia a medida que refinamos el largo característico de la malla, en la cual se puede apreciar la convergencia lineal óptima del método.

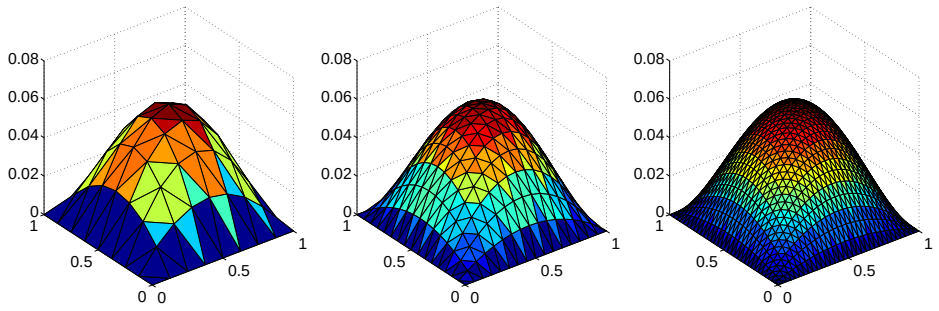

FIGURA 11. Ejemplo 1: Solución de elementos finitos $\mathsf{u}_{\mathscr{T}}$ para distintas particiones: con 512 elementos (izquierda), con 1024 elementos (centro) y 2048 elementos (derecha).

En los ejemplos que siguen, vamos a resolver el siguiente problema:

Hallar u ***tal que,***

$$\left\{\begin{array}{rcl} -\varepsilon\Delta\mathsf{u}(\boldsymbol{x}) + \boldsymbol{b}(\boldsymbol{x})\cdot\nabla\mathsf{u}(\boldsymbol{x}) + \mathsf{u}(\boldsymbol{x}) &=& 0 \quad \forall\, \boldsymbol{x} \in \Omega, \\ \mathsf{u}(\boldsymbol{x}) &=& \mathsf{u}_D(\boldsymbol{x}) \quad \forall\, \boldsymbol{x} \in \partial\Omega_D, \\ \nabla\mathsf{u}(\boldsymbol{x})\cdot\boldsymbol{n} &=& 0 \quad \forall\, \boldsymbol{x} \in \partial\Omega_N, \end{array}\right. \tag{4.27}$$

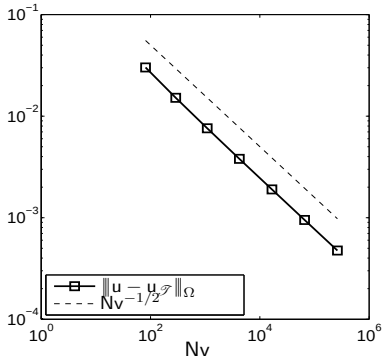

FIGURA 12. Ejemplo 1: Convergencia del método de elementos finitos, donde la convergencia lineal corresponde a $h = \frac{1}{\mathsf{Nv}^{1/2}}$.

donde $\boldsymbol{n}$ es el vector normal unitario a la frontera $\partial\Omega$, la cual cumple con $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, en donde en $\partial\Omega_D$ se impondrá uno condición del tipo Dirichlet, y en $\partial\Omega_N$ se impondrá uno condición del tipo Neumann.

La manera de introducir una condición de frontera en nuestro problema es como sigue: suponer que existe una extensión $\mathcal{E}(\mathsf{u}_D)$ del dato Dirichlet a todo el dominio Ω, tal que $\mathcal{E}(\mathsf{u}_D)|_{\partial\Omega_D} = \mathsf{u}_D$. Luego, hacemos el siguiente cambio de variable

$$\hat{\mathsf{u}} = \mathsf{u} - \mathcal{E}(\mathsf{u}_D) \in H_0^1(\Omega).$$

Luego, discretizaremos el siguiente problema débil:

Encontrar $\hat{\mathsf{u}} \in H_0^1(\Omega)$ tal que

$$\mathcal{B}(\hat{\mathsf{u}}, \mathsf{v}) = \mathcal{F}(\mathsf{v}) - \mathcal{B}(\mathcal{E}(\mathsf{u}_D), \mathsf{v}), \quad \forall\ \mathsf{v} \in H_0^1(\Omega), \tag{4.28}$$

donde

$$\mathcal{B}(\mathsf{u}, \mathsf{v}) := \int_\Omega (\varepsilon \nabla \mathsf{u} \cdot \nabla \mathsf{v} + \boldsymbol{b} \cdot \nabla \mathsf{u}\mathsf{v} + \mathsf{u}\mathsf{v})\ d\boldsymbol{x},$$
$$\mathcal{F}(\mathsf{v}) := \int_\Omega \mathsf{f}\mathsf{v}\ d\boldsymbol{x}. \tag{4.29}$$

Ejemplo 2: Consideraremos como dominio Ω, el descrito en la figura 13. En la figura, notemos que ponemos en la chimenea en la colina de la izquierda una condición de Dirichlet $\mathsf{u}_D = 1000$, en la parte izquierda y tapa superior, una condición de Dirichlet homogenea $\mathsf{u}_D = 0$, y en el resto de la frontera una condición de Neumann

nula, es decir, $\partial_n \mathsf{u} = 0$. Como datos para nuestra ecuación, consideramos:

$$\varepsilon = 1, \quad \boldsymbol{b} = [25 - x, \cos(2\pi x)]^T.$$

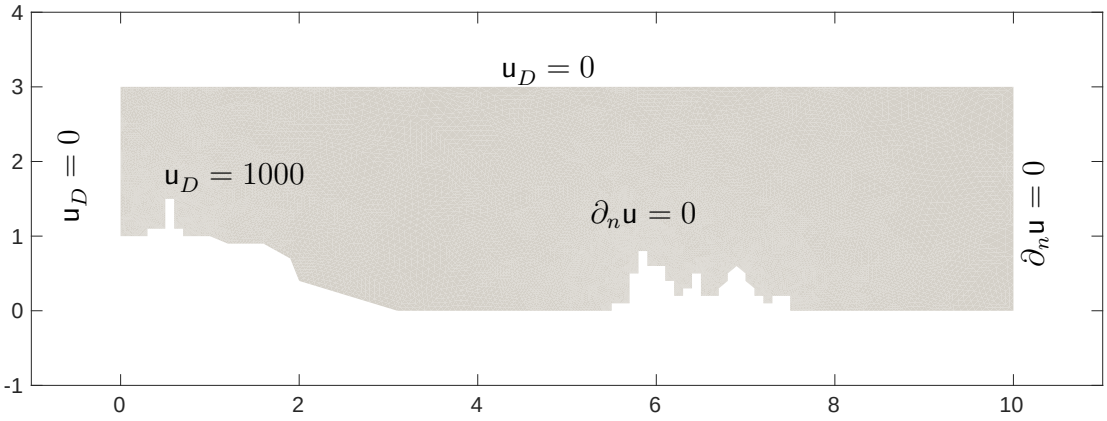

FIGURA 13. Dominio y condiciones de frontera para el ejemplo 2.

En la Figura 14, mostramos la malla de elementos finitos en la cual resolveremos nuestro problema, la cual contiene 15971 puntos y 31280 triángulos. El problema en cuestión puede ser mirado desde dos puntos de vista. El primero como un problema asociado a difusión de temperatura, o como un segundo problema asociado a la difusión de un contaminante, en la cual la chimenea de la colina es la cual difunde dicha temperatura/contaminante, en dirección a la ciudad, ya que el campo convectivo $\boldsymbol{b}$, es de alguna manera el campo vectorial asociado al viento que mueve la sustancia temperatura/contaminante.

En la Figura 15, mostramos la solución de elementos finitos obtenida al resolver el problema en la malla de elementos finitos mostrada en la Figura 14.

Ejemplo 3: Consideraremos como dominio Ω, el descrito en la figura 16. En el dominio, el cual simula un rio con una serie de brazos, es que consideramos en el brazo inferior izquierdo una condición de Dirichlet no homogenea, y en el resto de la frontera del dominio una condición de homogenea de Neumann. Como datos para nuestra ecuación, consideramos:

$$\varepsilon = 1, \quad \boldsymbol{b} = [15, 15]^T.$$

En la Figura 17, mostramos la malla de elementos finitos en la cual resolveremos nuestro problema, la cual contiene 1486 puntos y 2204 triángulos.

En la Figura 18, mostramos la solución de elementos finitos obtenida al resolver el problema en la malla de elementos finitos mostrada en la Figura 14.

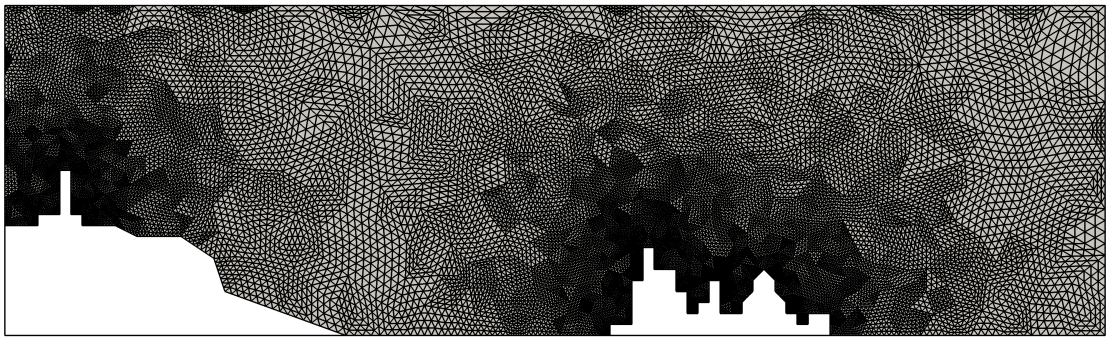

FIGURA 14. Problema 2: malla de elementos finitos con 15971 puntos y 31280 triángulos.

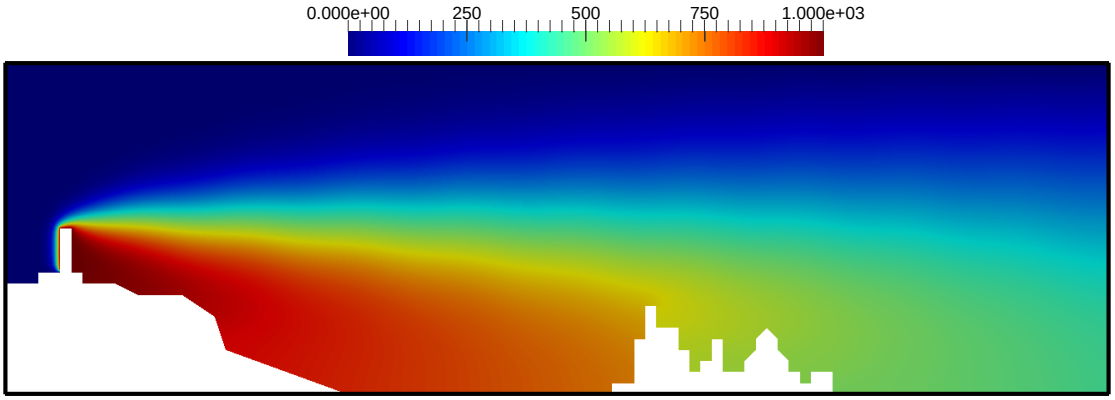

FIGURA 15. Ejemplo 2: solución de elementos finitos $\mathsf{u}_{\mathscr{T}}$, sobre la malla de la Figura 14.

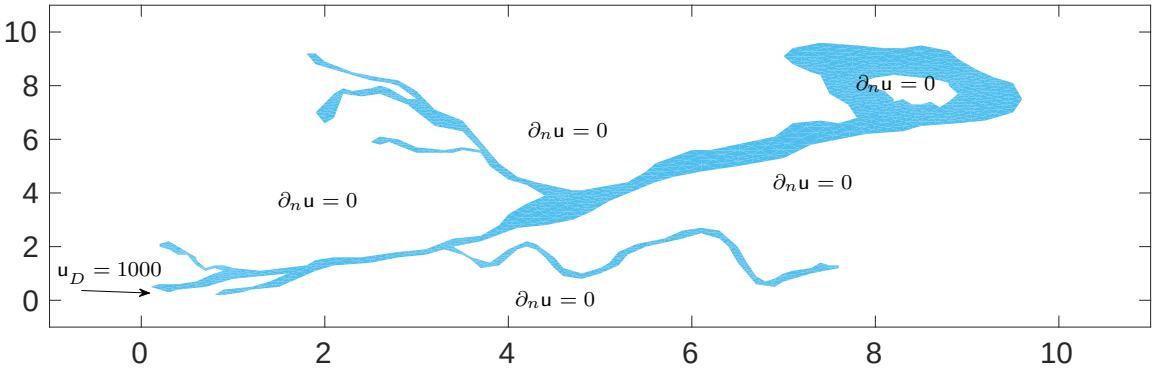

FIGURA 16. Dominio y condiciones de frontera para el ejemplo 3.

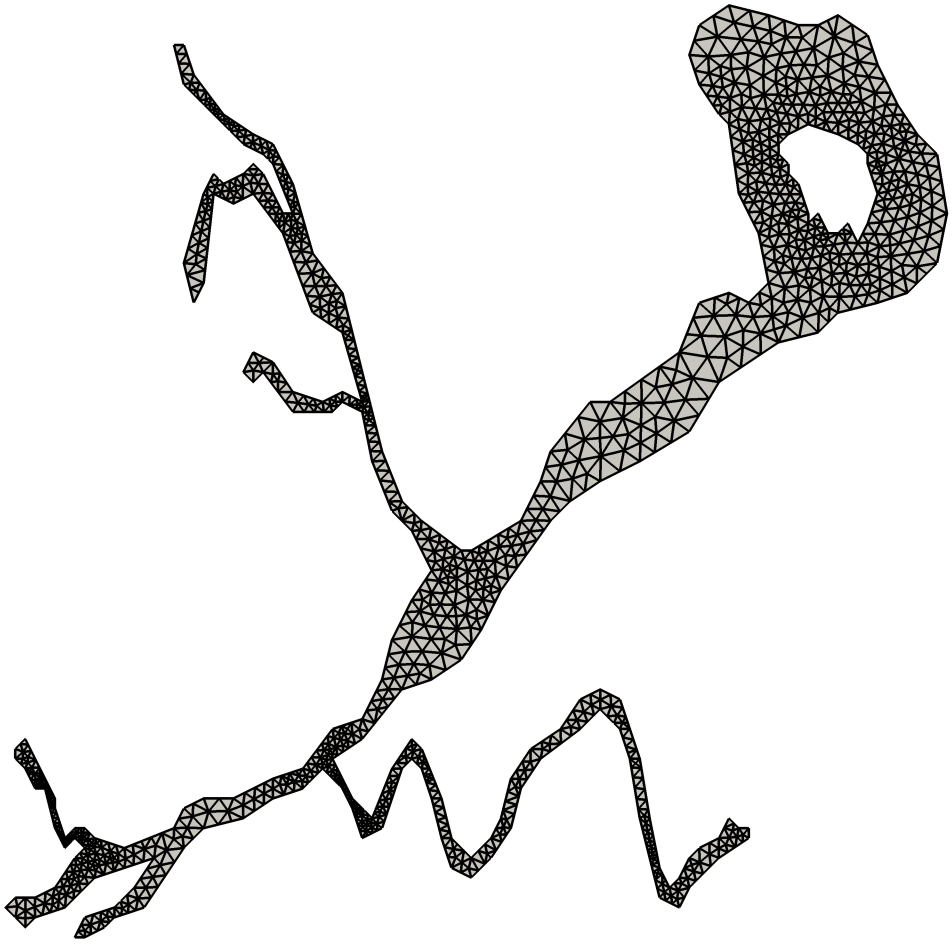

FIGURA 17. Malla de elementos finitos con 1486 puntos y 2204 triángulos para el Ejemplo 3.

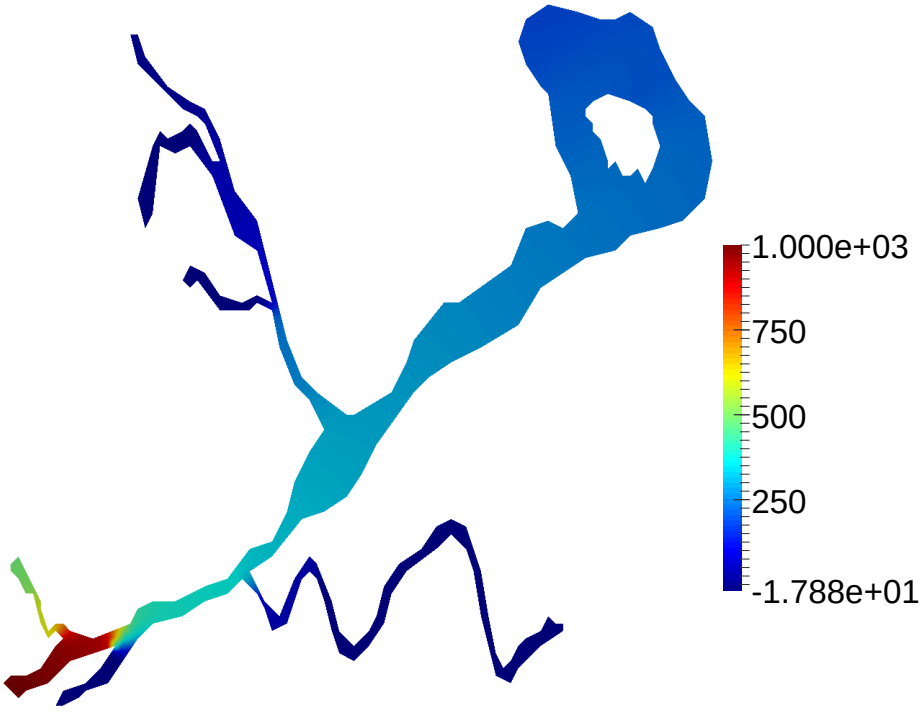

FIGURA 18. Solución de elementos finitos $\mathsf{u}_{\mathscr{T}}$, sobre la malla de la Figura 17.

5. Un Método de elementos finitos estabilizado para advección–reacción–difusión

Los métodos de elementos finitos estabilizados nacen de la necesidad de entregar una mayor robustez del esquema clásico de Galerkin, en casos en donde esté presente el comportamiento denominado de capa limíte, que corresponde a soluciones en las cuales existen fuertes cambios de gradiente en la solución, que justamente es un caso habitual en modelación de contaminantes y en la dinámica de un fluido.

Los elementos finitos estabilizados ya tienen una larga data de teoría de aproximación y propiedades relacionadas, en donde la idea fundamental es aumentar el esquema clásico de Galerkin, discutido en la sección anterior, mediante términos que dependan de la partición del dominio. Uno de los esquemas más clásicos y utilizados en la ingeniería, corresponden a métodos que agregan términos residuales a la formulación en conjunto con los llamados parámetros de estabilización (ver [15]).

5.1. Formulación de Galerkin estabilizada. Para ser más concretos consideraremos el siguiente esquema de elementos finitos estabilizados del tipo residual:

Encontrar $\mathsf{u}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T})$ tal que

$$\mathcal{B}_h(\mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = \mathcal{F}_h(\mathsf{v}_{\mathscr{T}}), \quad \text{para toda } \mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T}), \tag{5.1}$$

donde las nuevas formas están dadas por

$$\begin{aligned} \mathcal{B}_h(\mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) &= \mathcal{B}(\mathsf{u}, \mathsf{v}) + \sum_{T\in\mathscr{T}} \delta_T \int_T (-\varepsilon\Delta \mathsf{u}_{\mathscr{T}} + \boldsymbol{b}\cdot\nabla \mathsf{u}_{\mathscr{T}} + \mathsf{u}_{\mathscr{T}})(\boldsymbol{b}\cdot\nabla \mathsf{v}_{\mathscr{T}}) d\boldsymbol{x} \\ \mathcal{F}_h(\mathsf{v}) &= \mathcal{F}(\mathsf{v}) + \sum_{T\in\mathscr{T}} \delta_T \int_T (\mathsf{f})(\boldsymbol{b}\cdot\nabla \mathsf{v}_{\mathscr{T}})\, d\boldsymbol{x}, \end{aligned} \tag{5.2}$$

donde el parámetro de estabilización $\delta_T > 0$.

Asumiendo que la solución u de (4.1), es regular, en el sentido que

$$-\varepsilon\Delta \mathsf{u} + \boldsymbol{b}\cdot\nabla \mathsf{u} + \mathsf{u} = \mathsf{f} \quad \text{en } L^2(T),\ \forall\ T \in \mathscr{T},$$

luego, claramente se cumplirá que

$$\mathcal{B}_h(\mathsf{u}, \mathsf{v}_{\mathscr{T}}) = \mathcal{F}_h(\mathsf{v}_{\mathscr{T}}) \quad \forall\ \mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T}), \tag{5.3}$$

lo que permite concluir la siguiente propiedad de ortogonalidad, con respecto a la solución $\mathsf{u}_{\mathscr{T}}$ de (5.1),

$$\mathcal{B}_h(\mathsf{u} - \mathsf{u}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) = 0 \quad \forall\ \mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T}). \tag{5.4}$$

Ahora, para estudiar las propiedades de esta nueva solución es que utilizaremos la nueva norma

$$|||\mathsf{v}|||_{\Omega,St} = \left(|||\mathsf{v}|||_{\Omega}^2 + \sum_{T\in\mathscr{T}} \delta_T \|\boldsymbol{b}\cdot\nabla\mathsf{v}\|_{L^2(\Omega)}^2 \right)^{1/2}, \tag{5.5}$$

donde $|||\cdot|||_{\Omega}$ está definida en (4.3).

Para obtener existencia y unicidad de esta nueva solución en el espacio de Hilbert $(\mathbb{V}(\mathscr{T}), |||\mathsf{v}|||_{\Omega,St})$, es que tenemos que demostrar nuevamente las propiedades expuestas en el Teorema 1, es decir, tenemos que demostrar continuidad y coercividad de la forma $\mathcal{B}_h$ y continuidad del funcional lineal $\mathcal{F}_h$, pero ahora usando la norma $|||\mathsf{v}|||_{\Omega,St}$.

Continuidad de $\mathcal{B}_h$: para obtener la continuidad de la forma estabilizada, recordemos que tenemos que demostrar que $\mathcal{B}_h(\mathsf{u}_{\mathscr{T}},\mathsf{v}_{\mathscr{T}}) \leq C|||\mathsf{u}_{\mathscr{T}}|||_{\Omega,St}|||\mathsf{v}_{\mathscr{T}}|||_{\Omega}$ para toda $\mathsf{u}_{\mathscr{T}},\mathsf{v}_{\mathscr{T}} \in \mathbb{V}(\mathscr{T})$. Ahora, para dicho efecto, haciendo uso de (5.2), $\Delta\mathsf{u}_{\mathscr{T}} = 0$ ya que $\mathsf{u}_{\mathscr{T}}$ es lineal, en conjunto con la desigualdad de Cauchy–Schwarz, se tiene que

$$\begin{aligned}
&\mathcal{B}_h(\mathsf{u}_{\mathscr{T}},\mathsf{v}_{\mathscr{T}}) \\
&= \mathcal{B}(\mathsf{u}_{\mathscr{T}},\mathsf{v}_{\mathscr{T}}) + \sum_{T\in\mathscr{T}} \delta_T \int_T (-\varepsilon\Delta\mathsf{u}_{\mathscr{T}} + \boldsymbol{b}\cdot\nabla\mathsf{u}_{\mathscr{T}} + \mathsf{u}_{\mathscr{T}})(\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}})d\boldsymbol{x} \\
&\leq \mathtt{C}_{up}|||\mathsf{u}_{\mathscr{T}}|||_{\Omega}|||\mathsf{v}_{\mathscr{T}}|||_{\Omega} + \sum_{T\in\mathscr{T}} \delta_T^{1/2}\|\boldsymbol{b}\cdot\nabla\mathsf{u}_{\mathscr{T}}\|_{L^2(T)}\delta_T^{1/2}\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)} \\
&\qquad + \sum_{T\in\mathscr{T}} \delta_T^{1/2}\|\mathsf{u}_{\mathscr{T}}\|_{L^2(T)}\delta_T^{1/2}\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)} \\
&\leq \mathtt{C}_{up}|||\mathsf{u}_{\mathscr{T}}|||_{\Omega}|||\mathsf{v}_{\mathscr{T}}|||_{\Omega} + \left(\sum_{T\in\mathscr{T}} \delta_T\|\boldsymbol{b}\cdot\nabla\mathsf{u}_{\mathscr{T}}\|_{L^2(T)}^2\right)^{1/2}\left(\sum_{T\in\mathscr{T}} \delta_T\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2\right)^{1/2} \\
&\qquad + \left(\sum_{T\in\mathscr{T}} \delta_T\|\mathsf{u}_{\mathscr{T}}\|_{L^2(T)}^2\right)^{1/2}\left(\sum_{T\in\mathscr{T}} \delta_T\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2\right)^{1/2} \\
&\leq (\mathtt{C}_{up}+1)|||\mathsf{u}_{\mathscr{T}}|||_{\Omega,St}|||\mathsf{v}_{\mathscr{T}}|||_{\Omega,St} + \left(\sum_{T\in\mathscr{T}} \delta_T\|\mathsf{u}_{\mathscr{T}}\|_{L^2(T)}^2\right)^{1/2} |||\mathsf{v}_{\mathscr{T}}|||_{\Omega,St}.
\end{aligned}$$

Ahora, asumiendo que los parámetros de estabilización satisfacen que

$$\max_{T\in\mathscr{T}}\{\delta_T\} \leq \mathtt{C}_{\delta},$$

donde la constante $\mathtt{C}_\delta$ no depende de la partición ni ninguna variable asociada al problema, entonces podemos concluir que

$$\mathcal{B}_h(\mathtt{u}_\mathscr{T}, \mathtt{v}_\mathscr{T}) \leq \max\{\mathtt{C}_{up}+1, \mathtt{C}_\delta\} |\!|\!| \mathtt{u}_\mathscr{T} |\!|\!|_{\Omega,St} |\!|\!| \mathtt{v}_\mathscr{T} |\!|\!|_{\Omega,St}.$$

Coercividad de $\mathcal{B}_h$: para cualquier elemento $\mathtt{v}_\mathscr{T} \in \mathbb{V}(\mathscr{T})$, usando (4.5) y el hecho que $\Delta \mathtt{v}_\mathscr{T} = 0$ ya que $\mathtt{v}_\mathscr{T} \in \mathbb{P}_1(\mathscr{T})$, se tiene que

$$\begin{aligned}
\mathcal{B}_h(\mathtt{v}_\mathscr{T}, \mathtt{v}_\mathscr{T}) &= \mathcal{B}(\mathtt{v}_\mathscr{T}, \mathtt{v}_\mathscr{T}) + \sum_{T\in\mathscr{T}} \delta_T \int_T (-\varepsilon \Delta \mathtt{v}_\mathscr{T} + \boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T} + \mathtt{v}_\mathscr{T})(\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}) d\boldsymbol{x} \\
&= \mathcal{B}(\mathtt{v}_\mathscr{T}, \mathtt{v}_\mathscr{T}) + \sum_{T\in\mathscr{T}} \delta_T \|\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}\|^2_{L^2(T)} + \delta_T \int_T (\mathtt{v}_\mathscr{T})(\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}) d\boldsymbol{x} \\
&\geq \mathtt{C}_{lw} |\!|\!| \mathtt{v}_\mathscr{T} |\!|\!|^2_\Omega + \sum_{T\in\mathscr{T}} \delta_T \|\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}\|^2_{L^2(T)} + \sum_{T\in\mathscr{T}} \delta_T \int_T (\mathtt{v}_\mathscr{T})(\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}) d\boldsymbol{x} \\
&\geq \mathtt{C}_{lw} |\!|\!| \mathtt{v}_\mathscr{T} |\!|\!|^2_\Omega + \sum_{T\in\mathscr{T}} \delta_T \|\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}\|^2_{L^2(T)} - \left| \sum_{T\in\mathscr{T}} \delta_T \int_T (\mathtt{v}_\mathscr{T})(\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}) d\boldsymbol{x} \right|.
\end{aligned}$$

Ahora, usando la desigualdad $ab \leq \frac{1}{2}(a^2+b^2)$, se tiene que

$$\begin{aligned}
\left| \sum_{T\in\mathscr{T}} \delta_T \int_T (\mathtt{v}_\mathscr{T})(\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}) d\boldsymbol{x} \right| &\leq \sum_{T\in\mathscr{T}} \frac{\delta_T}{2} \int_T \left(\mathtt{v}^2_\mathscr{T} + (\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T})^2 \right) d\boldsymbol{x} \\
&= \sum_{T\in\mathscr{T}} \frac{\delta_T}{2} \|\mathtt{v}_\mathscr{T}\|^2_{L^2(T)} + \frac{\delta_T}{2} \|\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}\|^2_{L^2(T)},
\end{aligned}$$

la cual si se multiplica por menos, se tiene que

$$- \left| \sum_{T\in\mathscr{T}} \delta_T \int_T (\mathtt{v}_\mathscr{T})(\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}) d\boldsymbol{x} \right| \geq \sum_{T\in\mathscr{T}} -\frac{\delta_T}{2} \|\mathtt{v}_\mathscr{T}\|^2_{L^2(T)} - \frac{\delta_T}{2} \|\boldsymbol{b}\cdot\nabla \mathtt{v}_\mathscr{T}\|^2_{L^2(T)},$$

desigualdad que se puede insertar en la desigualdad de coercividad, para así concluir que

$$
\begin{aligned}
&\mathcal{B}_h(\mathsf{v}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) \\
&\geq \mathtt{C}_{lw}\|\!|\mathsf{v}_{\mathscr{T}}|\!\|_{\Omega}^2 + \sum_{T\in\mathscr{T}} \delta_T \|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 + \sum_{T\in\mathscr{T}} -\frac{\delta_T}{2}\|\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 - \frac{\delta_T}{2}\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 \\
&= \mathtt{C}_{lw}\left(\varepsilon\|\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(\Omega)}^2 + \|\mathsf{v}_{\mathscr{T}}\|_{L^2(\Omega)}^2\right) - \sum_{T\in\mathscr{T}} \frac{\delta_T}{2}\|\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 + \sum_{T\in\mathscr{T}} \frac{\delta_T}{2}\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 \\
&= \mathtt{C}_{lw}\left(\sum_{T\in\mathscr{T}} \varepsilon\|\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 + \|\mathsf{v}_{\mathscr{T}}\|_{L^2(\mathscr{T})}^2\right) - \sum_{T\in\mathscr{T}} \frac{\delta_T}{2}\|\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 + \sum_{T\in\mathscr{T}} \frac{\delta_T}{2}\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 \\
&= \sum_{T\in\mathscr{T}} \mathtt{C}_{lw}\varepsilon\|\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 + \sum_{T\in\mathscr{T}} \left(\mathtt{C}_{lw} - \frac{\delta_T}{2}\right)\|\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2 + \sum_{T\in\mathscr{T}} \frac{\delta_T}{2}\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2.
\end{aligned}
$$

Asumiendo que, para todo elemento $T \in \mathscr{T}$, se cumple que

$$
\mathtt{C}_{lw} - \frac{\delta_T}{2} = \text{mín}\{1, \mathtt{C}_b\} - \frac{\delta_T}{2} \geq \mathtt{C}_{st} > 0, \tag{5.6}
$$

entonces, se tiene que

$$
\begin{aligned}
\mathcal{B}_h(\mathsf{v}_{\mathscr{T}}, \mathsf{v}_{\mathscr{T}}) &\geq \text{mín}\{\mathtt{C}_{lw}, \mathtt{C}_{st}, 1/2\}\left(\|\!|\mathsf{v}_{\mathscr{T}}|\!\|_{\Omega}^2 + \sum_{T\in\mathscr{T}} \delta_T\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2\right) \\
&= \mathtt{C}_{lw,St}\|\!|\mathsf{v}|\!\|_{\Omega,St}^2.
\end{aligned}
$$

Continuidad de $\mathcal{F}_h$: usando (5.2), $\text{máx}_{T\in\mathscr{T}}\{\delta_T\} \leq \mathtt{C}_\delta$ y la desigualdad de Cauchy–Schwarz, se tiene que

$$
\begin{aligned}
\mathcal{F}_h(\mathsf{v}_{\mathscr{T}}) &= \mathcal{F}(\mathsf{v}_{\mathscr{T}}) + \sum_{T\in\mathscr{T}} \delta_T \int_T (\mathsf{f})(\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}})\, d\boldsymbol{x} \\
&\leq \|\mathsf{f}\|_{L^2(\Omega)}\|\!|\mathsf{v}_{\mathscr{T}}|\!\|_{\Omega} + \left(\sum_{T\in\mathscr{T}} \delta_{\mathscr{T}}\|\mathsf{f}\|_{L^2(T)}^2\right)^{1/2}\left(\sum_{T\in\mathscr{T}} \delta_{\mathscr{T}}\|\boldsymbol{b}\cdot\nabla\mathsf{v}_{\mathscr{T}}\|_{L^2(T)}^2\right)^{1/2} \\
&\leq \mathtt{C}\|\!|\mathsf{v}_{\mathscr{T}}|\!\|_{\Omega}.
\end{aligned}
$$

Entonces, como tenemos todas las propiedades del Teorema de Lax-Milgram, tenemos existencia y unicidad de una solución discreta.

5.2. Convergencia del método estabilizado. Para obtener un resultado de convergencia del método estabilizado, primero recordamos el siguiente resultado de interpolación local (ver Remark 1.105 en [9]):

Teorema. Existe un operador de interpolación $I_L : H_0^1(T) \to \mathbb{P}_1(T)$ tal que (5.7)

$$\|\mathsf{u} - I_L(\mathsf{u})\|_{L^2(T)} + h_T\|\nabla(\mathsf{u} - I_L(\mathsf{u}))\|_{L^2(T)} + h_T^2\|\Delta(\mathsf{u} - I_L(\mathsf{u}))\|_{L^2(T)} \leq \mathtt{C}_{L,T}h_T^2|\mathsf{u}|_{H^2(T)},$$

para todo elemento $T \in \mathscr{T}$.

Ahora, se quiere obtener una tasa de convergencia para el error $|||\mathsf{u} - \mathsf{u}_{\mathscr{T}}|||_{\Omega,St}$, en la cual, haciendo uso de una desigualdad triangular con $I_L(\mathsf{u})$, se tiene que

$$|||\mathsf{u} - \mathsf{u}_{\mathscr{T}}|||_{\Omega,St} \leq |||\mathsf{u} - I_L(\mathsf{u})|||_{\Omega,St} + |||I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}|||_{\Omega,St} := I + II. \tag{5.8}$$

Cota para I: La idea consiste en poder demostrar que el error satisface que

$$|||\mathsf{u} - I_L(\mathsf{u})|||_{\Omega,St} \leq Ch.$$

Para esto, notemos que haciendo uso del Teorema anterior, se puede concluir que

$$\begin{aligned}
&|||\mathsf{u} - I_L(\mathsf{u})|||^2_{\Omega,St} \\
&= \varepsilon\|\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(\Omega)} + \|\mathsf{u} - I_L(\mathsf{u})\|^2_{L^2(\Omega)} + \sum_{T\in\mathscr{T}} \delta_T\|\boldsymbol{b}\cdot\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(T)} \\
&= \sum_{T\in\mathscr{T}} \varepsilon\|\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(T)} + \|\mathsf{u} - I_L(\mathsf{u})\|^2_{L^2(T)} + \delta_T\|\boldsymbol{b}\cdot\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(T)} \\
&\leq \sum_{T\in\mathscr{T}} \varepsilon\mathtt{C}^2_{L,T}h_T^2|\mathsf{u}|^2_{H^2(T)} + \mathtt{C}^2_{L,T}h_T^4|\mathsf{u}|^2_{H^2(T)} + \sum_{T\in\mathscr{T}} \delta_T\|\boldsymbol{b}\cdot\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(T)}.
\end{aligned}$$

Para el último término anterior, haciendo uso del hecho que $(a+b)^2 \leq 2(a^2+b^2)$ y que $\boldsymbol{b} = (b_1, b_2)$, se tiene que

$$\begin{aligned}
\|\boldsymbol{b}\cdot\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(T)} &= \int_T (\boldsymbol{b}\cdot\nabla(\mathsf{u} - I_L(\mathsf{u}))(\boldsymbol{b}\cdot\nabla(\mathsf{u} - I_L(\mathsf{u}))\, d\boldsymbol{x} \\
&= \int_T (b_1\partial_x(\mathsf{u} - I_L(\mathsf{u})) + b_2\partial_y(\mathsf{u} - I_L(\mathsf{u})))^2\, d\boldsymbol{x} \\
&\leq 2\int_T \left(b_1^2\partial_x(\mathsf{u} - I_L(\mathsf{u}))^2 + b_2^2\partial_y(\mathsf{u} - I_L(\mathsf{u}))^2\right) d\boldsymbol{x} \\
&\leq 2\|\boldsymbol{b}\|^2_{L^\infty(T)}\int_T \left(\partial_x(\mathsf{u} - I_L(\mathsf{u}))^2 + \partial_y(\mathsf{u} - I_L(\mathsf{u}))^2\right) d\boldsymbol{x} \\
&= 2\|\boldsymbol{b}\|^2_{L^\infty(T)}\|\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(T)},
\end{aligned}$$

la cual en conjunto con el resultado de interpolación local, permite concluir finalmente que

$$\begin{aligned}
&\|\!|\mathsf{u} - I_L(\mathsf{u})|\!\|^2_{\Omega,St} \\
&\leq \sum_{T\in\mathscr{T}} \varepsilon \mathtt{C}^2_{L,T} h_T^2 |\mathsf{u}|^2_{H^2(T)} + \mathtt{C}^2_{L,T} h_T^4 |\mathsf{u}|^2_{H^2(T)} + 2\delta_T \|\boldsymbol{b}\|^2_{L^\infty(T)} \varepsilon \mathtt{C}^2_{L,T} h_T^2 |\mathsf{u}|^2_{H^2(T)} \\
&\leq Ch^2 |\mathsf{u}|^2_{H^2(\Omega)}. \qquad (5.9)
\end{aligned}$$

Cota para II: La idea consiste en poder demostrar que el error satisface que

$$\|\!|I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}|\!\|_{\Omega,St} \leq Ch.$$

Ahora, haciendo uso de la propiedad de coercividad anterior tomando $\mathsf{v}_{\mathscr{T}} = I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}$ junto con (5.3) y (5.4), se tiene que

$$\mathtt{C}_{lw,St} \|\!|I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}|\!\|^2_{\Omega,St} \leq \mathcal{B}_h(I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}, I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}) = \mathcal{B}_h(I_L(\mathsf{u}) - \mathsf{u}, I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}).$$

Haciendo uso de la desigualdad de Cauchy–Schwarz, primero estimaremos el primer término de la forma estabilizada, es decir,

$$\begin{aligned}
&\varepsilon \int_\Omega \nabla(I_L(\mathsf{u}) - \mathsf{u}) \cdot \nabla(I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}))\, d\boldsymbol{x} \\
&\leq \varepsilon^{1/2} \|\nabla(I_L(\mathsf{u}) - \mathsf{u})\|_{L^2(\Omega)} \varepsilon^{1/2} \|\nabla(I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}})\|_{L^2(\Omega)} \\
&\leq \varepsilon^{1/2} \|\nabla(I_L(\mathsf{u}) - \mathsf{u})\|_{L^2(\Omega)} \|\!|\nabla(I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}})|\!\|_{\Omega,St}.
\end{aligned}$$

la cual en conjunto con el resultado de interpolación local, permite concluir que

$$\varepsilon \int_\Omega \nabla(I_L(\mathsf{u}) - \mathsf{u}) \cdot \nabla(I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}}))\, d\boldsymbol{x} \leq \varepsilon^{1/2} \mathtt{C}_{L,T} h_T |\mathsf{u}|_{H^2(\mathscr{T})} \|\!|\nabla(I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}})|\!\|_{\Omega,St}.$$

Ahora, acotaremos el segundo y tercer término, es decir,

$$
\begin{aligned}
&\int_\Omega (\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u})+I_L(\mathsf{u})-\mathsf{u})(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}\\
&=\int_\Omega (\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}))(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}+\int_\Omega (I_L(\mathsf{u})-\mathsf{u})(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}\\
&=-\int_\Omega (\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}))(I_L(\mathsf{u})-\mathsf{u})\,d\boldsymbol{x}-\int_\Omega \nabla\cdot\boldsymbol{b}(I_L(\mathsf{u})-\mathsf{u})(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}\\
&\quad+\underbrace{\int_{\partial\Omega}(I_L(\mathsf{u})-\mathsf{u})(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\boldsymbol{b}\cdot\boldsymbol{n}\,ds}_{=0}+\int_\Omega (I_L(\mathsf{u})-\mathsf{u})(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}\\
&=\int_\Omega [(1-\nabla\cdot\boldsymbol{b})(I_L(\mathsf{u})-\mathsf{u})](I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}-\int_\Omega (I_L(\mathsf{u})-\mathsf{u})(\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}))\,d\boldsymbol{x}\\
&=\sum_{T\in\mathscr{T}}\int_T [(1-\nabla\cdot\boldsymbol{b})(I_L(\mathsf{u})-\mathsf{u})](I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}\\
&\qquad\qquad-\int_T (I_L(\mathsf{u})-\mathsf{u})(\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}))\,d\boldsymbol{x}
\end{aligned}
$$

Ahora, en cada elemento de la partición podemos usar la desigualdad de Cauchy–Schwarz, para acotar de la siguiente manera

$$
\begin{aligned}
&\int_T [(1-\nabla\cdot\boldsymbol{b})(I_L(\mathsf{u})-\mathsf{u})](I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}-\int_T (I_L(\mathsf{u})-\mathsf{u})(\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}))\,d\boldsymbol{x}\\
&\le C\|I_L(\mathsf{u})-\mathsf{u}\|_{L^2(T)}\|I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}\|_{L^2(T)}\\
&\qquad\qquad+\frac{1}{\delta_T}\|I_L(\mathsf{u})-\mathsf{u}\|_{L^2(T)}\delta_T\|\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\|_{L^2(T)},
\end{aligned}
$$

desigualdad que puede ser utilizada en las igualdades anteriores y que permite concluir, nuevamente usando la desigualdad de Cauchy–Schwarz y la definición de la

norma (5.5) y (5.7), que

$$
\begin{aligned}
&\int_\Omega (\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u})+I_L(\mathsf{u})-\mathsf{u})(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\,d\boldsymbol{x}\\
&\le C\sum_{T\in\mathscr{T}}\Big(\|I_L(\mathsf{u})-\mathsf{u}\|_{L^2(T)}\|I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}\|_{L^2(T)}\\
&\qquad+\frac{1}{\sqrt{\delta_T}}\|I_L(\mathsf{u})-\mathsf{u}\|_{L^2(T)}\sqrt{\delta_T}\|\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\|_{L^2(T)}\Big)\\
&\le C\Bigg(\left(\sum_{T\in\mathscr{T}}\|I_L(\mathsf{u})-\mathsf{u}\|^2_{L^2(T)}\right)^{1/2}\|I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}\|_{L^2(\Omega)}\\
&\qquad+\left(\sum_{T\in\mathscr{T}}\frac{1}{\delta_T}\|I_L(\mathsf{u})-\mathsf{u}\|^2_{L^2(T)}\right)^{1/2}\left(\sum_{T\in\mathscr{T}}\delta_T\|\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\|^2_{L^2(T)}\right)^{1/2}\Bigg)\\
&\le C\left(\left(\sum_{T\in\mathscr{T}}\|I_L(\mathsf{u})-\mathsf{u}\|^2_{L^2(T)}\right)^{1/2}+\left(\sum_{T\in\mathscr{T}}\frac{1}{\delta_T}\|I_L(\mathsf{u})-\mathsf{u}\|^2_{L^2(T)}\right)^{1/2}\right)|\!|\!|I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}|\!|\!|_{\Omega,St}\\
&\le C\left(\left(\sum_{T\in\mathscr{T}}\mathtt{C}^2_{L,T}h_T^4|\mathsf{u}|^2_{H^2(T)}\right)^{1/2}+\left(\sum_{T\in\mathscr{T}}\frac{1}{\delta_T}\mathtt{C}^2_{L,T}h_T^4|\mathsf{u}|^2_{H^2(T)}\right)^{1/2}\right)|\!|\!|I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}|\!|\!|_{\Omega,St}\\
&\le 2C\left(\sum_{T\in\mathscr{T}}\mathtt{C}^2_{L,T}h_T^4\left(1+\frac{1}{\delta_T}\right)|\mathsf{u}|^2_{H^2(T)}\right)^{1/2}|\!|\!|I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}|\!|\!|_{\Omega,St}.
\end{aligned}
$$

De la misma forma, podemos acotar el último término

$$
\begin{aligned}
&\sum_{T\in\mathscr{T}}\delta_T\int_T(-\varepsilon\Delta(I_L(\mathsf{u})-\mathsf{u})+\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u})+(\mathsf{u})-\mathsf{u}))\,(\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}))\;d\boldsymbol{x}\\
&=\sum_{T\in\mathscr{T}}\delta_T^{1/2}\Big(\varepsilon\|I_L(\mathsf{u})-\mathsf{u}\|_{L^2(T)}+\|\boldsymbol{b}\|_{L^\infty(T)}\|\nabla(I_L(\mathsf{u})-\mathsf{u})\|_{L^2(T)}\\
&\qquad+\|I_L(\mathsf{u})-\mathsf{u}\|_{L^2(T)}\Big)\delta_T^{1/2}\|\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\|_{L^2(T)}\\
&=\sum_{T\in\mathscr{T}}\delta_T^{1/2}\mathtt{C}_{L,T}\left(\varepsilon+\|\boldsymbol{b}\|_{L^\infty(T)}h_T+h_T^2\right)|\mathsf{u}|_{H^2(T)}\delta_T^{1/2}\|\boldsymbol{b}\cdot\nabla(I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}})\|_{L^2(T)}\\
&\le\left(\sum_{T\in\mathscr{T}}\left(\delta_T^{1/2}\mathtt{C}_{L,T}\left(\varepsilon+\|\boldsymbol{b}\|_{L^\infty(T)}h_T+h_T^2\right)\right)^2|\mathsf{u}|^2_{H^2(T)}\right)^{1/2}|\!|\!|I_L(\mathsf{u})-\mathsf{u}_{\mathscr{T}}|\!|\!|_{\Omega,St}.
\end{aligned}
$$

Asumiendo ahora que

$$\varepsilon\delta_T\le Ch_T^2\quad\forall\,T\in\mathscr{T},$$

luego,

$$\begin{aligned}\delta_T^{1/2}\mathtt{C}_{L,T}\left(\varepsilon+\|\boldsymbol{b}\|_{L^\infty(T)}h_T+h_T^2\right) &\leq C\mathtt{C}_{L,T}\left(\sqrt{\varepsilon}\sqrt{\varepsilon\delta_T}+\delta_T^{1/2}h_T+\delta_T^{1/2}h_T^2\right)\\ &\leq C\mathtt{C}_{L,T}\left(\sqrt{\varepsilon}\sqrt{\varepsilon\delta_T}+2\delta_T^{1/2}h_T\right)\\ &\leq 2C\mathtt{C}_{L,T}\left(\sqrt{\varepsilon}+\delta_T^{1/2}\right)h_T\end{aligned}$$

lo cual insertándola en el acotamiento del último término permite concluir que

$$\begin{aligned}&\sum_{T\in\mathscr{T}}\delta_T\int_T\left(-\varepsilon\Delta(I_L(\mathtt{u})-\mathtt{u})+\boldsymbol{b}\cdot\nabla(I_L(\mathtt{u})-\mathtt{u})+(\mathtt{u})-\mathtt{u})\right)\left(\boldsymbol{b}\cdot\nabla(I_L(\mathtt{u})-\mathtt{u}_{\mathscr{T}})\right)\,d\boldsymbol{x}\\ &\leq\left(\sum_{T\in\mathscr{T}}\left(\delta_T^{1/2}\mathtt{C}_{L,T}\left(\varepsilon+\|\boldsymbol{b}\|_{L^\infty(T)}h_T+h_T^2\right)\right)^2|\mathtt{u}|^2_{H^2(T)}\right)|\!|\!|I_L(\mathtt{u})-\mathtt{u}_{\mathscr{T}}|\!|\!|_{\Omega,St}\\ &\leq C\left(\sum_{T\in\mathscr{T}}\left(\varepsilon+\delta_T\right)h_T^2|\mathtt{u}|^2_{H^2(T)}\right)^{1/2}|\!|\!|I_L(\mathtt{u})-\mathtt{u}_{\mathscr{T}}|\!|\!|_{\Omega,St}\end{aligned}$$

Finalmente, bajo todos los supuestos anteriores y juntando todos los resultados anteriores con (5.8), (5.9) y (5.10), se puede concluir que

$$|\!|\!|\mathtt{u}-\mathtt{u}_{\mathscr{T}}|\!|\!|_{\Omega,St}\leq Ch|\mathtt{u}|_{H^2(\Omega)}. \tag{5.10}$$

En la práctica, notemos que se tiene que hacer una elección del parámetro de estabilización, para el cual, una elección usual es, dado

$$Pe_T=\frac{\|\boldsymbol{b}\|_{L^\infty(T)}h_T}{2\varepsilon},$$

luego se define el parámetro de estabilización, para todo elemento $T\in\mathscr{T}$, como

$$\delta_T=\begin{cases}\frac{h_T}{2\|\boldsymbol{b}\|_{L^\infty(T)}} & \text{si } Pe_T>1,\\ \frac{h_T^2}{12\varepsilon} & \text{si } Pe_T\leq 1.\end{cases} \tag{5.11}$$

5.3. Código del Esquema Estabilizado. Las diferencias en el código se limitan a la adición de ciertas funciones de apoyo al cálculo y a variaciones en el código del ensamble de la matriz y el lado derecho de la ecuación

```
double get_hk(Vector2d X0,Vector2d X1,Vector2d X2){
    double hk = (X1-X0).norm();
    if (hk < (X2-X0).norm()) {
       hk = (X2-X0).norm();
   }
   if (hk < (X2-X1).norm()) {
       hk = (X2-X1).norm();
   }
   return hk;
}
double get_Peclet(Vector2d X0, Vector2d X1, Vector2d X2) {
    double hk = get_hk(X0,X1,X2);
    MatrixXd b_k(2,3);
    b_k.col(0) = get_b(X0);
    b_k.col(1) = get_b(X1);
    b_k.col(2) = get_b(X2);
    double b_K_norm = b_k.cwiseAbs().maxCoeff();
    double Peclet = (b_K_norm * hk) / (2.0 * get_epsilon());
    return Peclet;
}
```

LISTING 7. Funciones de soporte adicionales

```
void get_RHS(int nt,const VectorXi& subdomain,const MatrixXi& elements,const MatrixXd&
    vertices,int nbf,const VectorXi& boundary_subdomain,const MatrixXi& boundary,std::vector<
    double>& RHS, const Matrix<double, 3, 2> normals[],const VectorXd& areas){
    for (int k=0; k<nt; k++) {
        Vector3d fmom = Vector3d::Zero();
        Vector2d X0 = vertices.row(elements(k,0));
        Vector2d X1 = vertices.row(elements(k,1));
        Vector2d X2 = vertices.row(elements(k,2));
        double h_K = get_hk(X0,X1,X2);
        double delta_K = (h_K/24.0) * (get_Peclet(X0,X1,X2) > 1) + (pow(h_K, 2) / (24.0*
            get_epsilon())) * (get_Peclet(X0,X1,X2) <= 1);
        for (int j=0; j<NTRIQP; j++) {
            Vector2d X=triquad[j][0]*vertices.row(elements(k,0))+triquad[j][1]*vertices.row(
                elements(k,1))+triquad[j][2]*vertices.row(elements(k,2));
            double w_f=triweight[j]*get_f(X,subdomain(k));
            Vector2d b_X = triquad[j][0]*get_b(vertices.row(elements(k,0)))+triquad[j][1]*
                get_b(vertices.row(elements(k,1)))+triquad[j][2]*get_b(vertices.row(elements(
                k,2)));
            for (int local_node=0; local_node<3; local_node++) {
                fmom(local_node)+=w_f*(triquad[j][local_node] + delta_K * b_X.dot((-0.5/areas
                    (k))*normals[k].row(local_node)));
            }
        }
        for (int local_node=0; local_node<3; local_node++) {
            RHS[elements(k,local_node)]+=areas(k)*fmom(local_node);
        }
    }
    for (int i=0; i<nbf; i++){
        if (get_Dirichlet(boundary_subdomain(i))==0){
            double h_F=get_hf(vertices.row(boundary(i,0)),vertices.row(boundary(i,1)));
            Vector2d du_n_mom = Vector2d::Zero();
            for (int j=0; j<NLINQP; j++){
                Vector2d X=linquad[j][0]*vertices.row(boundary(i,0))+linquad[j][1]*vertices.
                    row(boundary(i,1));
                double w_du_n=linweight[j]*get_uN(X,boundary_subdomain(i));
                for (int local_node=0;local_node<2;local_node++){
                    du_n_mom(local_node)+=w_du_n*linquad[j][local_node];
                }
```

```
            }
            for (int local_node=0;local_node<2;local_node++){
                RHS[boundary(i,local_node)]+=h_F*du_n_mom(local_node);
            }
        }
    }
}
```

LISTING 8. Ensamble de lado derecho para esquema estabilizado

```
void get_stiffnessmatrix(int nv,int nbf,const VectorXi& boundary_subdomain,const MatrixXi&
    boundary,int nt,const MatrixXi& elements,const MatrixXd& vertices,std::vector<double>&
    RHS,SparseMatrix<double>& K,const MatrixXi& edgeelement,const MatrixXi& edgeedge,const
    Matrix<double, 3, 2> normals[],const VectorXd& areas,const VectorXi& Dnodes){

    for (int k=0; k<nt; k++) {
        Vector2d X0 = vertices.row(elements(k,0));
        Vector2d X1 = vertices.row(elements(k,1));
        Vector2d X2 = vertices.row(elements(k,2));
        double h_K = get_hk(X0,X1,X2);
        double delta_K = h_K * (get_Peclet(X0,X1,X2) > 1) + (pow(h_K, 2) / get_epsilon()) * (
            get_Peclet(X0,X1,X2) <= 1);

        for (int local_node_a=0; local_node_a<3; local_node_a++) {
            if (Dnodes(elements(k,local_node_a))>=0)
                {
                for (int local_node_b=0; local_node_b<3; local_node_b++) {
                    double sum=0;
                    double val=(get_epsilon() * normals[k].row(local_node_a).dot(normals[k].
                        row(local_node_b)))/(4*areas(k));
                    sum+=val;

                    double int_B=0.0;
                    for (int t=0;t<3;t++)
                    {
                        Vector2d local_coordinate;
                        local_coordinate(0)=vertices(elements(k,t),0);
                        local_coordinate(1)=vertices(elements(k,t),1);
                        Vector2d B=get_b(local_coordinate);
                        double B_dot_lambda_i = (-0.5/areas(k)) * normals[k].row(local_node_a
                            ).dot(B);
                        if (t==local_node_a)
                        {
                            int_B+=-B.dot(normals[k].row(local_node_b))/12.0;
                        }
                        else
                        {
                            int_B+=-B.dot(normals[k].row(local_node_b))/24.0;
                        }
                        sum += delta_K * B_dot_lambda_i * (areas(k) / 6.0) * ((t ==
                            local_node_b) + 0.5 * (t != local_node_b));
                        for (int s = 0; s < 3; s++) {
                            Vector2d local_coordinate;
                            local_coordinate(0)=vertices(elements(k,t),0);
                            local_coordinate(1)=vertices(elements(k,t),1);
                            Vector2d B=get_b(local_coordinate);
                            double B_dot_lambda_j = (-0.5/areas(k)) * normals[k].row(
                                local_node_b).dot(B);
                            sum += delta_K * B_dot_lambda_i * B_dot_lambda_j * (areas(k) /
                                6.0) * ((t == s) + 0.5 * (t != s));
                        }
                    }
                    sum+=int_B;

                    if (elements(k,local_node_a)==elements(k,local_node_b))
                    {
```

```
48                          sum+=areas(k)/6.0;
49                      }
50                      else
51                      {
52                          sum+=areas(k)/12.0;
53                      }
54                      if (Dnodes(elements(k,local_node_b))>=0)
55                                          {
56                                              K.coeffRef(elements(k,local_node_a),elements(k,
                                                    local_node_b))+=sum;
57                      }
58                      else{
59                          RHS[elements(k,local_node_a)]-=sum*get_uD(vertices.row(elements(k,
                                local_node_b)),-Dnodes(elements(k,local_node_b)));
60                      }
61                  }
62              }
63              else {
64                  K.coeffRef(elements(k,local_node_a),elements(k,local_node_a))=1;
65                  RHS[elements(k,local_node_a)]=get_uD(vertices.row(elements(k,local_node_a)),-
                        Dnodes(elements(k,local_node_a)));
66              }
67          }
68      }
69  }
```

LISTING 9. Ensamble de matriz para esquema estabilizado

5.4. Ejemplos numéricos. Para validar numéricamente nuestro código es que primero vamos a utilizar dos soluciones exactas para nuestro problema. Las dos soluciones exactas son de tal forma que se puedan apreciar el fenómeno de fuertes cambios en los gradientes de las funciones, es decir, el comportamiento de capa límite, tanto cerca de la frontera como en el interior.

Ejemplo 1: consideraremos que el dominio de nuestro problema es el cuadrado unitario $\Omega = (0,1) \times (0,1)$ y se tiene la siguiente solución analítca

$$\mathsf{u}(x,y) = xy(1-x)(1-y)\mathsf{tg}^{-1}((x-0{,}5)/\varepsilon),$$

en donde, para el cálculo del lado derecho usamos que $\varepsilon = 10^{-3}$ y $\boldsymbol{b} = [1,1]^T$. Este problema contiene una capa límite dentro del dominio a lo largo de la recta $x = 0{,}5$.

Ejemplo 2: consideraremos que el dominio de nuestro problema es el cuadrado unitario $\Omega = (0,1) \times (0,1)$ y se tiene la siguiente solución analítca

$$\mathsf{u}(x,y) = y(1-y)\left(x - \frac{e^{-(1-x)/\varepsilon} - e^{-1/\varepsilon}}{1 - e^{-1/\varepsilon}}\right).$$

en donde, para el cálculo del lado derecho usamos que $\varepsilon = 2 \times 10^{-3}$ y $\boldsymbol{b} = [1,1]^T$. Este problema contiene una capa límite cerca de la frontera del dominio cuando $x = 1$.

Por último, recordemos que para el cálculo del lado derecho utilizamos una regla de cuadratura con 73 puntos de Gauss y que se cumple la siguiente relación entre el largo característico de la partición y el número total de vértices

$$h \approx \frac{1}{\mathsf{Nv}^{1/2}}.$$

En las Figuras 19 y 21 se muestran una serie de soluciones de elementos finitos, sin y con estabilización, para los dos problema en distintas mallas, en las cuales se puede apreciar claramente que la solución de elementos finitos estabilizada aproxima de mejor forma y sin oscilaciones las soluciones, tanto de capa límite interior como de frontera. En las Figuras 20 y 22, se muestra la tasa de convergencia a medida de refinamos el largo característico de la malla, en la cual se puede apreciar la convergencia lineal óptima de cada método, pero siendo siempre mejor la aproximación de elementos finitos estabilizada.

Ejemplo 3: Consideramos ahora el **Ejemplo 2** de la sección anterior, con la única modificación de considerar ahora los siguientes parámetros para nuestra ecuación:

$$\varepsilon = 10^{-3}, \quad \boldsymbol{b} = [25 - x, \cos(2\pi x)]^T.$$

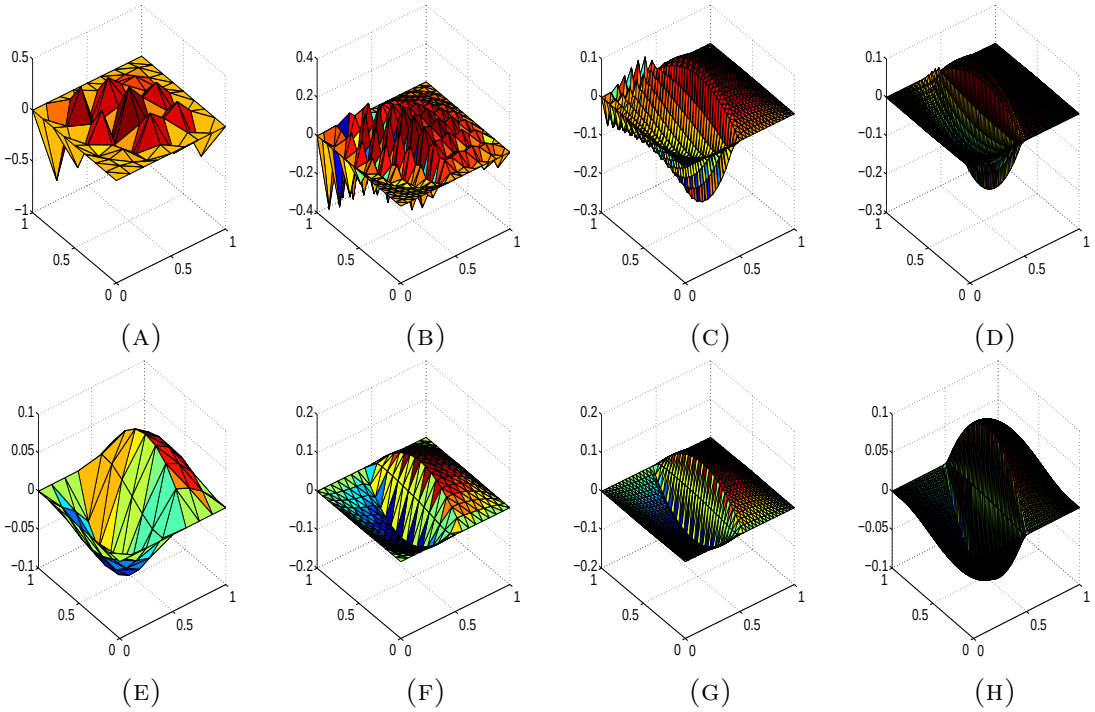

FIGURA 19. Ejemplo 1: Solución de elementos finitos sin estabilizar (a–d) y solución estabilizada (e–h), para distintas particiones: con 128 elementos (a y e), 512 elementos (b y f), 2048 elementos (c y g) y 8192 elementos (d y h).

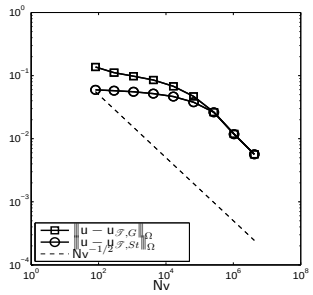

FIGURA 20. Ejemplo 1: Convergencia del método de elementos finitos tanto del método de Galerkin ($\mathsf{u}_{\mathscr{T},G}$) y del esquema estabilizado ($\mathsf{u}_{\mathscr{T},St}$), donde la convergencia lineal corresponde a $h = \frac{1}{\mathsf{Nv}^{1/2}}$.

En la Figura 23, mostramos nuevamente las condiciones de frontera, y la solución de elementos finitos estandar $\mathsf{u}_{\mathscr{T},G}$ y la solución de elementos finitos estabilizada $\mathsf{u}_{\mathscr{T},St}$,

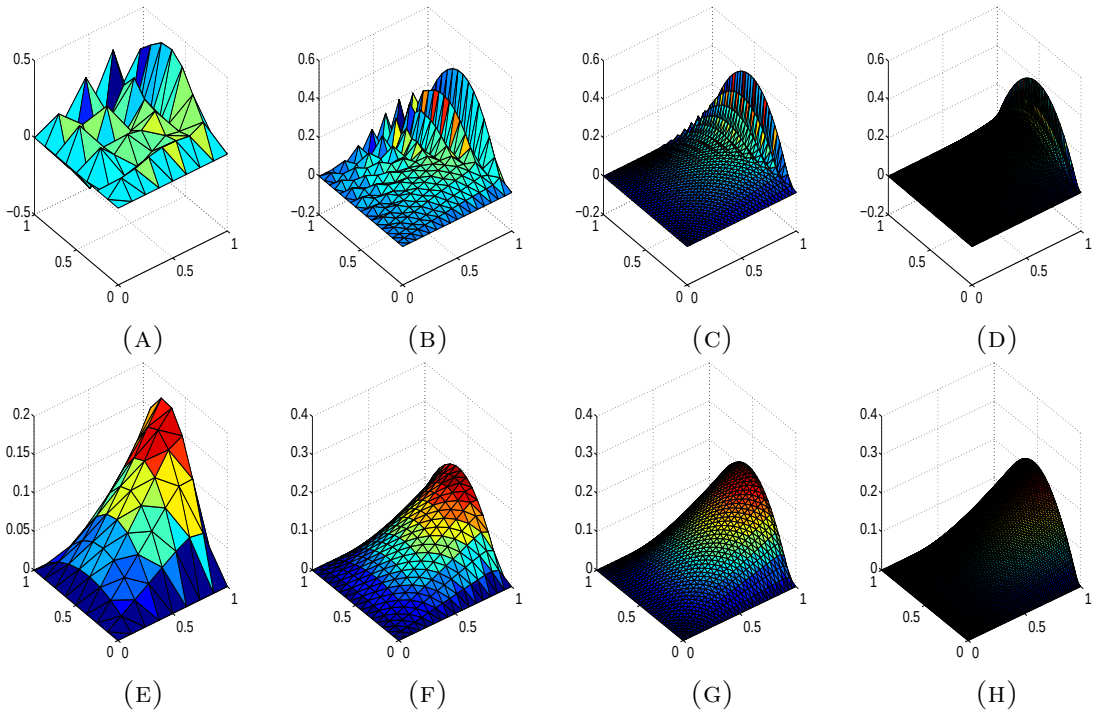

FIGURA 21. Ejemplo 2: Solución de elementos finitos sin estabilizar (a–d) y solución estabilizada (e–h), para distintas particiones: con 128 elementos (a y e), 512 elementos (b y f), 2048 elementos (c y g) y 8192 elementos (d y h).

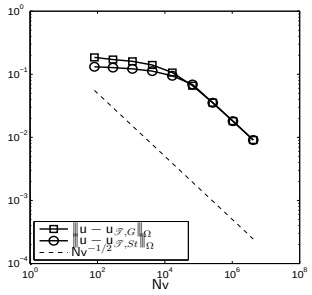

FIGURA 22. Ejemplo 2: Convergencia del método de elementos finitos tanto del método de Galerkin ($\mathsf{u}_{\mathscr{T},G}$) y del esquema estabilizado ($\mathsf{u}_{\mathscr{T},St}$), donde la convergencia lineal corresponde a $h = \frac{1}{\mathsf{Nv}^{1/2}}$.

la cual claramente entrega un resultado mucho más fidedigno, miestras la primera está completamente contaminada.

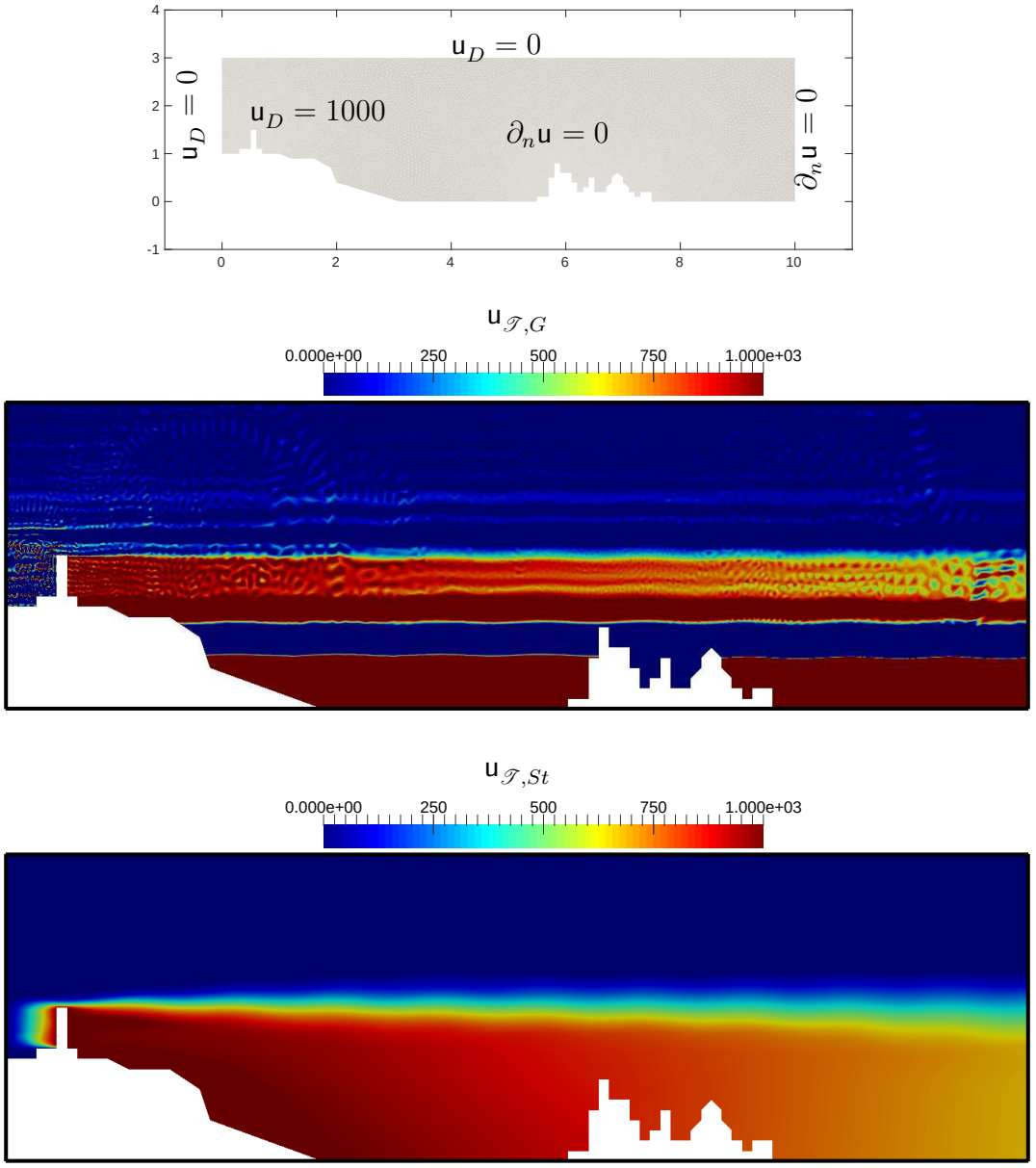

FIGURA 23. Ejemplo 3: Condiciones de borde (superior), solución de elementos finitos sin estabilización $u_{\mathscr{T},G}$ (centro), y solución de elementos finitos estabilizada $u_{\mathscr{T},St}$ (inferior).

Ejemplo 4: Consideramos ahora el **Ejemplo 3** de la sección anterior, con la única modificación de considerar ahora los siguientes parámetros para nuestra ecuación:

$$\varepsilon = 10^{-3}, \quad \boldsymbol{b} = [15, 15]^T.$$

En la Figura 24, mostramos la solución de elementos finitos estandar $\mathsf{u}_{\mathscr{T},G}$ y la solución de elementos finitos estabilizada $\mathsf{u}_{\mathscr{T},St}$, la cual claramente entrega un resultado mucho más fidedigno nuevamente, miestras la primera está completamente contaminada.

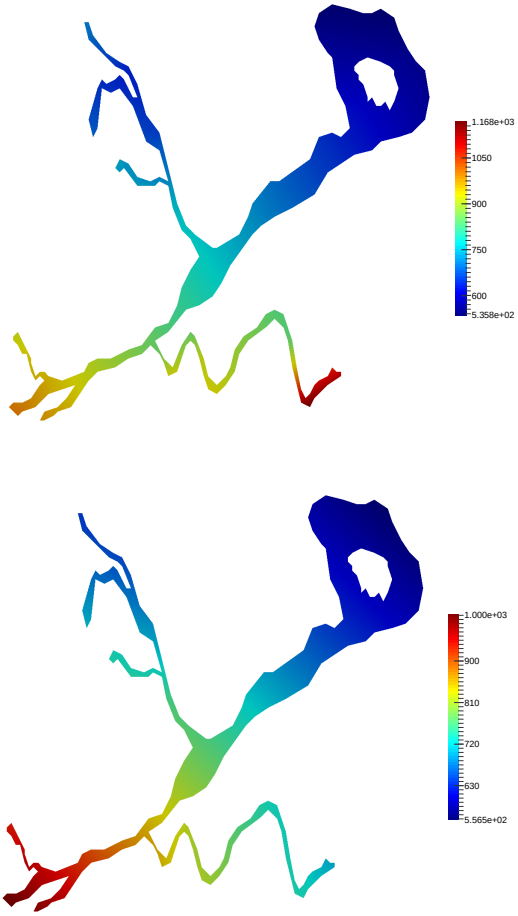

FIGURA 24. Ejemplo 4: solución de elementos finitos sin estabilización $\mathsf{u}_{\mathscr{T},G}$ (superior), y solución de elementos finitos estabilizada $\mathsf{u}_{\mathscr{T},St}$ (inferior).

6. Un método de elementos finitos estabilizados para un problema de Stokes

En ésta sección, nos ocuparemos de construir un método de elementos finitos para un problema lineal de fluidos. Para presentar el problema, primero consideraremos las siguientes definiciones de operadores diferenciales:

- si $\mathbf{v} : \mathbb{R}^d \to \mathbb{R}^d$, donde $\mathbf{v}(\boldsymbol{x}) = [v_1(\boldsymbol{x}), .., v_d(\boldsymbol{x})]^T$ con $\boldsymbol{x} = [x_1, ..., x_d]^T$, luego

$$\text{div } \mathbf{v}(\boldsymbol{x}) = \sum_{i=1}^{d} \frac{\partial v_i}{\partial x_i}, \quad \nabla \mathbf{v} = \begin{bmatrix} \nabla v_1(\boldsymbol{x}) \\ \vdots \\ \nabla v_d(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \cdots & \frac{\partial v_1}{\partial x_d} \\ \vdots & & \vdots \\ \frac{\partial v_d}{\partial x_1} & \cdots & \frac{\partial v_d}{\partial x_d} \end{bmatrix},$$

y

$$\Delta \mathbf{v} = \begin{bmatrix} \Delta v_1(\boldsymbol{x}) \\ \vdots \\ \Delta v_d(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{d} \frac{\partial v_1}{\partial x_i} \\ \vdots \\ \sum_{i=1}^{d} \frac{\partial v_d}{\partial x_i} \end{bmatrix}.$$

Ahora, en base a las definiciones anteriores, es que consideraremos el siguiente problema, dada una fuerza externa $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$, donde $\mathbf{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), .., f_d(\boldsymbol{x})]^T$:

Hallar $(\mathbf{u}, p)$ ***tal que,***

$$(6.1) \qquad \begin{cases} -\varepsilon \Delta \mathbf{u}(\boldsymbol{x}) + \nabla p &= \mathbf{f}(\boldsymbol{x}) \quad \text{para todo } \boldsymbol{x} \in \Omega, \\ \text{div } \mathbf{u}(\boldsymbol{x}) &= 0 \quad \text{para todo } \boldsymbol{x} \in \Omega, \\ \mathsf{u}(\boldsymbol{x}) &= \mathbf{0} \quad \text{para todo } \boldsymbol{x} \in \partial\Omega. \end{cases}$$

En el problema anterior, el cual de denomina un problema de Stokes, corresponde al modelamiento de un fluido incompresible, en donde la variable $\mathbf{u}$ corresponde a la velocidad del fluido y p corresponde a la presión del fluido y asumiremos que $\mathbf{f} \in [L^2(\Omega)]^d$. Dichas variables satisfacen el sistema de ecuaciones vectoriales, en donde la primera ecuación es denominada la ecuación de momentum y la segunda ecuación es denominada la ecuación de conservación de masa.

Al igual que en la Sección (4.1), primero obtendremos una formulación débil del problema. Para dicho efecto, primero consideraremos el par de funciones $(\mathbf{v}, q)$, lo suficientemenete suaves, y multiplicaremos la primera ecuación por $\mathbf{v}$ y la segunda por q, para posteriormente integrar sobre todo el dominio y después integrar por partes usando en cada componente (3.11). En base a lo anterior, y asumiento que

$\mathbf{v}|_{\partial\Omega} = 0$, se cumple que

$$\int_\Omega -\varepsilon\Delta\mathbf{u}\cdot\mathbf{v}\ d\boldsymbol{x} + \int_\Omega \nabla p\cdot\mathbf{v}\ d\boldsymbol{x} = \varepsilon\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v}\ d\boldsymbol{x} - \int_\Omega p\ \mathrm{div}\ \mathbf{v}\ d\boldsymbol{x}$$
$$= \varepsilon\sum_{i=1}^{d}\int_\Omega \nabla u_i\cdot\nabla v_i\ d\boldsymbol{x} - \int_\Omega p\ \mathrm{div}\ \mathbf{v}\ d\boldsymbol{x}.$$

De forma similar se tiene para la segunda ecuación que

$$\int_\Omega q\ \mathrm{div}\ \mathbf{u}\ d\boldsymbol{x} = 0.$$

Luego, y en base a la Sección 3, es que el problema débil asociado se lee como sigue:

Encontrar $(\mathbf{u}, p) \in [H_0^1(\Omega)]^d \times L_0^2(\Omega)$ tal que

$$\text{(6.2)}\qquad \begin{cases} \mathcal{A}(\mathbf{u},\mathbf{v}) + \mathcal{B}(p,\mathbf{v}) &=& \mathcal{F}(\mathbf{v}), & \forall\ \mathbf{v}\in[H_0^1(\Omega)]^d,\\ \mathcal{B}(q,\mathbf{u}) &=& 0, & \forall\ q\in L_0^2(\Omega), \end{cases}$$

donde las formas bilineales están dadas por

$$\mathcal{A}(\mathbf{u},\mathbf{v}) := \varepsilon\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v}\ d\boldsymbol{x}, \quad \mathcal{B}(p,\mathbf{v}) := -\int_\Omega p\ \mathrm{div}\ \mathbf{v}\ d\boldsymbol{x},$$

y el funcional lineal es

$$\mathcal{F}(\mathbf{v}) := \int_\Omega \mathbf{f}\cdot\mathbf{v}\ d\boldsymbol{x}.$$

Notemos que la variable presión $p \in L_0^2(\Omega)$, en donde el espacio se define como

$$\text{(6.3)}\qquad L_0^2(\Omega) := \left\{ q\in L^2(\Omega) :\ \int_\Omega q\ d\boldsymbol{x} = 0 \right\}.$$

El hecho de ocupar dicho espacio corresponde a la siguiente situación. Si $(\mathbf{u}, p)$ es solución de (6.1), luego $(\mathbf{u}, p + c)$, con $c \in \mathbb{R}$, es también solución. Luego, el espacio natural para estudiar dicho problema corresponde a

$$L^2(\Omega)/\mathbb{R} := \left\{ [q] :\ [q] = \{q - c,\ c\in\mathbb{R},\ q\in L^2(\Omega)\} \right\},$$

donde cualquier elemento $[q] \in L^2(\Omega)/\mathbb{R}$ es llamada una clase de equivalencia, para la cual la diferencia entre cualquier par de elementos que vivan en la clase es una constante y cualquier elemento $q \in [q]$ es llamado un representante de la clase. Dicho espacio se puede dotar de la norma cociente

$$\|[q]\|_{L^2(\Omega)/\mathbb{R}} = \inf_{c\in\mathbb{R}} \|q - c\|_{L^2(\Omega)}.$$

Ahora, recordemos que, como las siguientes son equivalentes (ver Proposición A.31 en [9]),

$$\|q - P(q)\|_{L^2(\Omega)} = \min_{c\in\mathbb{R}} \|q - c\|_{L^2(\Omega)} \Leftrightarrow \int_\Omega P(q)\xi\ d\boldsymbol{x} = \int_\Omega q\xi\ d\boldsymbol{x}, \quad \forall\ \xi\in\mathbb{R},$$

donde $P : L^2(\Omega) \to \mathbb{R}$ corresponde a la proyección ortogonal, luego

$$P(q) = \frac{1}{|\Omega|} \int_\Omega q \; d\boldsymbol{x}.$$

En base a lo anterior, es que podemos concluir, usando (6.3), que

$$\|[q]\|_{L^2(\Omega)\setminus\mathbb{R}} = \inf_{c\in\mathbb{R}} \|q - c\|_{L^2(\Omega)} = \left\| q - \frac{1}{|\Omega|} \int_\Omega q \; d\boldsymbol{x} \right\|_{L^2(\Omega)} = \|q\|_{L^2(\Omega)} \quad \forall \; q \in L^2_0(\Omega).$$

Lo anterior nos permite identificar biyectivamente los espacios $L^2_0(\Omega)$ y $L^2(\Omega) \setminus \mathbb{R}$. Más aún, mantendiendo la norma, es decir, de forma isométrica.

Para estudiar la existencia y unicidad del problema, es que se tiene que recurrir a una generalización del Lemma de Lax–Milgram, la cual es conocida como la teoría inf-sup.

En lo que sigue, notemos que la norma para un elemento vectorial, corresponde a

$$\|\mathbf{v}\|^2_{L^2(\Omega)} = \int_\Omega \mathbf{v} \cdot \mathbf{v} \; d\boldsymbol{x},$$

y la norma para su gradiente, corresponde a

$$\|\nabla \mathbf{v}\|^2_{L^2(\Omega)} = \sum_{i=1}^{d} \|\nabla v_i\|^2_{L^2(\Omega)}, \quad \forall \; \mathsf{v} \in [H^1_0(\Omega)]^d.$$

Teorema 2 (Teorema de Brezzi)**.** *El problema* (6.2) *tiene solución única, si y solamente si, existen constantes* $\alpha, \beta > 0$ *tales que*

$$\inf_{\mathbf{v}\in Ker(\mathbb{B})} \sup_{\mathbf{v}\in Ker(\mathbb{B})} \frac{\mathcal{A}(\mathbf{v},\mathbf{w})}{\|\nabla \mathbf{v}\|_{L^2(\Omega)} \|\nabla \mathbf{w}\|_{L^2(\Omega)}} = \sup_{\mathbf{v}\in Ker(\mathbb{B})} \inf_{\mathbf{v}\in Ker(\mathbb{B})} \frac{\mathcal{A}(\mathbf{v},\mathbf{w})}{\|\nabla \mathbf{v}\|_{L^2(\Omega)} \|\mathbf{w}\|_{L^2(\Omega)}} = \alpha > 0,$$

y

$$\inf_{q\in L^2_0(\Omega)} \sup_{\mathbf{v}\in[H^1_0(\Omega)]^d} \frac{\mathcal{B}(q,\mathbf{v})}{\|q\|_{L^2(\Omega)} \|\nabla \mathbf{v}\|_{L^2(\Omega)}} = \beta > 0,$$

donde

$$Ker(\mathbb{B}) := \{\mathbf{v} \in [H^1_0(\Omega)]^d : \; \mathcal{B}(q,\mathbf{v}) = 0, \; \forall \; q \in L^2_0(\Omega)\}.$$

Es sabido que el problema (6.2) cumple el teorema anterior, por lo cual existe una solución única para el problema (ver [12, 17, 4, 5, 9]).

La desventaja al ocupar la teoría inf-sup anterior, es que al ser utilizada en espacios de elementos finitos, resulta en una elección particular de espacios para aproximar la velocidad y la presión. De hecho, el método de elementos finitos inf-sup estable, es decir, que satisfacen el Teorema de Brezzi, es llamado el par Taylor–Hood, el cual corresponde aproximar la velocidad con polinomios de grado dos, y la presión mediante polinomios de grado uno, ambos cotinuos.

Para tener una mayor libertad en la elección de los espacios de elementos finitos, es que vamos a recurrir a la idea del Capítulo anterior, la cual corresponde a estabilizar el método de elementos finitos.

6.1. Un método de elementos finitos estabilizado. Para construir un método de elementos finitos estabilizados, vamos a considerar los siguientes espacios de elementos finitos:

$$\mathbb{V}(\mathscr{T})^d := \{\mathbf{v}_{\mathscr{T}} \in \mathcal{C}(\overline{\Omega})^d : \ \mathbf{v}_{\mathscr{T}} \in \mathbb{P}_1(T)^d, \ \forall\, T \in \mathscr{T}, \ \mathbf{v}_{|\partial\Omega} = 0\},$$

$$\mathbb{Q}(\mathscr{T}) := \{q_{\mathscr{T}} \in L^2_0(\Omega) : \ q_{\mathscr{T}} \in \mathbb{P}_0(T)^d, \ \forall\, T \in \mathscr{T}\}.$$

Ahora, para poder introducir el esquema estabilizado, es que definiremos el siguiente termino, llamado de salto (ver Figura 25):

$$[\![\phi]\!]_\gamma = \phi_{K^+} - \phi_{K^-}. \tag{6.4}$$

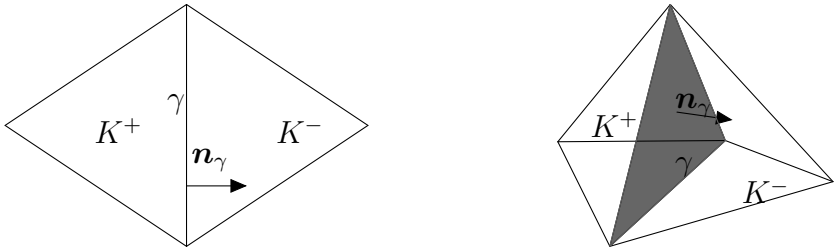

FIGURA 25. Para $\gamma = \partial K^+ \cap \partial K^-$, fijando un normal unitario $\boldsymbol{n}_\gamma$ apuntando desde K^+ a K^-, y así definiremos el salto de una función como $[\![\phi]\!]_\gamma = \phi_{K^+} - \phi_{K^-}$.

En base a los espacios de elementos finitos anteriores y la definición del salto en (6.4), es que consideraremos el siguiente esquema de Galerkin estabilizado:

Encontrar $(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}}) \in [\mathbb{V}(\mathscr{T})]^d \times \mathbb{Q}(\mathscr{T})$ tal que

$$\left\{ \begin{array}{rcll} \mathcal{A}(\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(p_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) & = & \mathcal{F}(\mathbf{v}_{\mathscr{T}}), & \forall\ \mathbf{v}_{\mathscr{T}} \in [\mathbb{V}(\mathscr{T})]^d, \\ \mathcal{B}(q_{\mathscr{T}}, \mathbf{u}_{\mathscr{T}}) + \mathcal{S}(p_{\mathscr{T}}, q_{\mathscr{T}}) & = & 0, & \forall\ q_{\mathscr{T}} \in \mathbb{Q}(\mathscr{T}), \end{array} \right. \tag{6.5}$$

donde

$$\mathcal{S}(p_{\mathscr{T}}, q_{\mathscr{T}}) = -\sum_{\gamma \in E_I} \tau_\gamma \int_\gamma [\![p_{\mathscr{T}}]\!]_\gamma [\![q_{\mathscr{T}}]\!]_\gamma \, ds,$$

y E_I denota el conjunto de todos los lados interiores de la partición $\mathscr{T}$, y τ_γ es el parámetro de estabilización, el cual es una constante positiva.

Para poder estudiar la existencia y unicidad de una solución para el problema (6.5), es que vamos a recurrir nuevamente al teorema de Lax-Milgram. Para ésto, definiremos una nueva forma bilineal, dada como

$$\mathfrak{B}((\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})) = \mathcal{A}(\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(p_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) - \mathcal{B}(q_{\mathscr{T}}, \mathbf{u}_{\mathscr{T}}) - \mathcal{S}(p_{\mathscr{T}}, q_{\mathscr{T}}),$$

y el siguiente funcional lineal

$$\mathfrak{F}((\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})) = \mathcal{F}(\mathbf{v}_{\mathscr{T}}).$$

Luego, podemos reformular el problema (6.5), de forma equivalente como sigue:

Encontrar $(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}}) \in \mathbb{X}(\mathscr{T})$ tal que

$$\mathfrak{B}((\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})) = \mathfrak{F}((\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})), \quad \forall\ (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}}) \in \mathbb{X}(\mathscr{T}), \tag{6.6}$$

donde

$$\mathbb{X}(\mathscr{T}) := [\mathbb{V}(\mathscr{T})]^d \times \mathbb{Q}(\mathscr{T}).$$

En el nuevo espacio $\mathbb{X}(\mathscr{T})$, tomaremos como norma

$$\|(\mathbf{v}_{\mathscr{T}}, p_{\mathscr{T}})\|^2_{\mathbb{X}(\mathscr{T})} = \varepsilon \|\nabla \mathbf{v}_{\mathscr{T}}\|^2_{L^2(\Omega)} + \sum_{\gamma \in E_I} \tau_\gamma \|[\![p_{\mathscr{T}}]\!]_\gamma\|^2_{L^2(\gamma)}.$$

Ahora, y de igual forma que en los capítulos anteriores, procedemos a demostrar todas la hipótesis del Teorema de Lax-Milgram.

Continuidad de $\mathfrak{B}$: recordemos que la propiedad de continuidad nos dice que existe una constante positiva C_{up} tal que

$$|\mathfrak{B}((\mathbf{w}_{\mathscr{T}}, t_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}}))| \leq \mathrm{C}_{up} \|(\mathbf{w}_{\mathscr{T}}, t_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})} \|(\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})},$$

para todo $(\mathtt{w}_{\mathscr{T}}, t_{\mathscr{T}}), (\mathtt{v}_{\mathscr{T}}, q_{\mathscr{T}})) \in \mathbb{X}(\mathscr{T})$. Usando la desigualdad de Cauchy–Schwarz y la definición de la forma bilineal, se tiene que

$$\begin{aligned}
&|\mathfrak{B}((\mathbf{w}_{\mathscr{T}}, t_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}}))| \\
&\leq \sqrt{\varepsilon}\|\mathbf{w}_{\mathscr{T}}\|_{L^2(\Omega)} \sqrt{\varepsilon}\|\mathbf{v}_{\mathscr{T}}\|_{L^2(\Omega)} + \left|\int_\Omega t_{\mathscr{T}} \nabla \cdot \mathbf{v}_{\mathscr{T}}\right| + \left|\int_\Omega q_{\mathscr{T}} \nabla \cdot \mathbf{w}_{\mathscr{T}}\right| \\
&\quad + \sum_{\gamma \in E_I} h_\gamma^{1/2} \|[\![t_{\mathscr{T}}]\!]_\gamma\|_{L^2(\gamma)} h_\gamma^{1/2} \|[\![q_{\mathscr{T}}]\!]_\gamma\|_{L^2(\gamma)}
\end{aligned}$$

Ahora, usando integración por partes y hecho que $t_{\mathscr{T}|K} \in \mathbb{P}_0(\mathscr{T})$, se tiene que

$$\begin{aligned}\int_\Omega t_{\mathscr{T}} \nabla \cdot \mathbf{v}_{\mathscr{T}} &= \sum_{K\in\mathscr{T}} \int_K t_{\mathscr{T}} \nabla \cdot \mathbf{v}_{\mathscr{T}} \\ &= \sum_{K\in\mathscr{T}} -\int_K \nabla t_{\mathscr{T}} \cdot \mathbf{v}_{\mathscr{T}} + \int_{\partial K} t_{\mathscr{T}} (\mathbf{v}_{\mathscr{T}} \cdot \boldsymbol{n}_\gamma^K) \\ &= \sum_{\gamma\in E_I} \int_\gamma [\![t_{\mathscr{T}}]\!] \mathbf{v}_{\mathscr{T}} \cdot \boldsymbol{n}_\gamma \\ &\leq \sum_{\gamma\in E_I} \tau_\gamma^{1/2} \|t_{\mathscr{T}}\|_{L^2(\gamma)} \tau_\gamma^{-1/2} \|\mathbf{v}_{\mathscr{T}}\|_{L^2(\gamma)}.\end{aligned}$$

Como $\mathbf{v}_{\mathscr{T}|K} \in \mathbb{P}_1(K)$, usando (4.14) y (4.15), se tiene que se cumple la siguiente desigualdad, usualmente denominada como inversa,

$$\|\mathbf{v}_{\mathscr{T}}\|_{L^2(\gamma)} \leq C h_K^{-1/2} \|\mathbf{v}_{\mathscr{T}}\|_{L^2(K)}.$$

Usando la desiguald inversa anterior y la desigualdad de Poincaré (3.15), podemos concluir que

$$\begin{aligned}\int_\Omega t_{\mathscr{T}} \nabla \cdot \mathbf{v}_{\mathscr{T}} &\leq \left(\sum_{\gamma\in E_I} \tau_\gamma \|[\![t_{\mathscr{T}}]\!]\|^2_{L^2(\gamma)} \right)^{1/2} \left(\sum_{K\in\mathscr{T}} \sum_{\gamma\in\partial K} \tau_\gamma^{-1} h_K^{-1} \|\mathbf{v}_{\mathscr{T}}\|^2_{L^2(K)} \right)^{1/2} \\ &\leq 3 \max_{K\in\mathscr{T}} \{\tau_\gamma^{-1} h_K^{-1}\} \left(\sum_{\gamma\in E_I} \tau_\gamma \|[\![t_{\mathscr{T}}]\!]\|^2_{L^2(\gamma)} \right)^{1/2} \|\mathbf{v}_{\mathscr{T}}\|^2_{L^2(\Omega)} \\ &\leq 3\mathrm{C}_{P,\Omega} \max_{K\in\mathscr{T}} \{\tau_\gamma^{-1} h_K^{-1}\} \left(\sum_{\gamma\in E_I} \tau_\gamma \|[\![t_{\mathscr{T}}]\!]\|^2_{L^2(\gamma)} \right)^{1/2} \|\nabla \mathbf{v}_{\mathscr{T}}\|^2_{L^2(\Omega)}\end{aligned}$$

Finalmente, agrupando todas las estimaciones anteriores y de la definición de la norma $\|\cdot\|_{\mathbb{X}(\mathscr{T})}$, se tiene que

$$|\mathfrak{B}((\mathbf{w}_{\mathscr{T}}, t_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}}))| \leq \mathrm{C}_{up} \|(\mathbf{w}_{\mathscr{T}}, t_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})} \|(\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})}.$$

Coercividad de $\mathfrak{B}$: Recordemos que la coercividad nos dice que exise una constante C_{lw}, tal que

$$\mathfrak{B}((\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})) \geq \mathrm{C}_{lw} \|(\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})\|^2_{\mathbb{X}(\mathscr{T})}.$$

Notemos, que de la definición, tenemos que

$$\mathfrak{B}((\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})) = \varepsilon \|\nabla \mathbf{v}_{\mathscr{T}}\|^2_{L^2(\Omega)} + \sum_{\gamma\in E_I} h_\gamma^2 \|[\![t_{\mathscr{T}}]\!]\|^2_{L^2(\gamma)} = \|(\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})\|^2_{\mathbb{X}(\mathscr{T})}.$$

En base a lo anterior la propiedad se cumple con constante $\mathrm{C}_{lw} = 1$.

Continuidad del funcional $\mathfrak{F}$: recordemos que la propiedad nos dice que

$$\mathfrak{F}((\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})) \leq \mathrm{C}\|(\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})}.$$

De la fenición de nuestro funcional y utilizando la desigualdad de Cauchy–Scharwz y la desigualdad de Poincaré (3.15), tenemos que

$$\begin{aligned}\mathfrak{F}((\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})) &\leq \|\mathbf{f}\|_{L^2(\Omega)}\|\mathbf{v}_{\mathscr{T}}\|_{L^2(\Omega)} \\ &\leq \frac{\mathrm{C}_{P,\Omega}}{\varepsilon^{-1/2}}\|\mathbf{f}\|_{L^2(\Omega)}\varepsilon^{1/2}\|\nabla\mathbf{v}_{\mathscr{T}}\|_{L^2(\Omega)} \\ &\leq \frac{\mathrm{C}_{P,\Omega}}{\varepsilon^{-1/2}}\|\mathbf{f}\|_{L^2(\Omega)}\|(\mathbf{v}_{\mathscr{T}}, t_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})}.\end{aligned}$$

Luego la propiedad se cumple con $\mathrm{C} = \mathrm{C}_{P,\Omega}\|\mathbf{f}\|_{L^2(\Omega)}$.

En base al análisis anterior tenemos que todas las hipótesis del teorema de Lax–Milgram se cumplen, luego existe una solución única para nuestro problem (6.6).

6.2. Convergencia del método estabilizado. Sea $(\mathbf{u}, p)$ solución del problema (6.2), y sea $(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}})$ su aproximación de elementos finitos, solución del problema (6.5). Supongamos que la solución débil satisface la siguiente suponsición

$$(\mathbf{u}, p) \in \left[[H^2(\Omega)]^d \times [H_0^1(\Omega)]^d\right] \times [H^1(\Omega) \times L_0^2(\Omega)].$$

Ahora, en base al supuesto anterior, se tiene que

$$\sum_{\gamma \in E_I} \int_{\gamma} [\![p]\!]_{\gamma}\, ds = 0.$$

Luego, podemos concluir que

$$\mathfrak{B}((\mathbf{u} - \mathbf{u}_{\mathscr{T}}, p - p_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})) = \mathfrak{B}((\mathbf{e}^u, e^p), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})) = 0, \quad \forall\ (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}}) \in \mathbb{X}(\mathscr{T}),$$

es decir, se cumple la denominada ortogonalidad de Galerkin, también llamada relación de consistencia.

Ahora, recordemos que existe un interpolador $I_L : H_0^1(\Omega) \to \mathbb{V}(\mathscr{T})$, el cual cumple con [5, Teorema 4.4.20]:

$$\|v - I_L(v)\|_{L^2(\Omega)} + h\|\nabla(v - I_L(v))\|_{L^2(\Omega)} \leq \mathrm{C}_L h^2 |v|_{H^2(\Omega)}, \quad \forall\ v \in H^2(\Omega).$$

De forma similar, usaremos la siguiente proyección ortogonal $\Pi : L^2(\Omega) \to \mathbb{Q}(\mathscr{T})$, que se define, elemento a elemento, como

$$\Pi(v)|_T = \frac{1}{|T|}\int_T v\ d\boldsymbol{x}.$$

Dicha proyección satisface que (ver [5]):

$$\|v - \Pi(v)\|_{L^2(\Omega)} \leq \mathrm{C}_\Pi h \|\nabla v\|_{L^2(\Omega)}, \quad \forall\, v \in H^1(\Omega). \tag{6.7}$$

De igual forma, dentro del posterior análisis, es que necesitaremos de la siguiente desigualdad, llamada de traza ([5, Ecuación 10.3.8]):

$$\|\xi\|^2_{L^2(\gamma)} \leq C\left(h_T^{-1}\|\xi\|^2_{L^2(T)} + h_T\|\nabla\xi\|^2_{H^1(T)}\right).$$

Notemos que de la desigualdad anterior y (6.7), podemos concluir inmediatamente que

$$\sum_{\gamma\in E_I} h_\gamma \|[\![v - \Pi(v)]\!]\|^2_{L^2(\gamma)} \leq \mathrm{C}_{\Pi,\mathscr{T}} h^2 \|\nabla v\|^2_{L^2(\Omega)}, \quad \forall\, v \in H^1(\Omega). \tag{6.8}$$

Ahora, como en el caso estabilizado, usando una desigualdad triangular, se tiene que

$$\begin{aligned}
\|(\mathsf{e}^u, e^p)\|^2_{\mathbb{X}(\mathscr{T})} &= \|(\mathsf{u} - \mathsf{u}_{\mathscr{T}}, p - p_{\mathscr{T}})\|^2_{\mathbb{X}(\mathscr{T})} \\
&= \varepsilon\|\nabla(\mathsf{u} - \mathsf{u}_{\mathscr{T}})\|^2_{L^2(\Omega)} + \|p - p_{\mathscr{T}}\|^2_{L^2(\Omega)} + \sum_{\gamma\in E_I} h_\gamma \|[\![p - p_{\mathscr{T}}]\!]\|^2_{L^2(\gamma)} \\
&\leq 2\Big(\varepsilon\|\nabla(\mathsf{u} - I_L(\mathsf{u}))\|^2_{L^2(\Omega)} + \varepsilon\|\nabla(I_L(\mathsf{u}) - \mathsf{u}_{\mathscr{T}})\|^2_{L^2(\Omega)} \\
&\quad + \|p - \Pi(p)\|^2_{L^2(\Omega)} + \|\Pi(p) - p_{\mathscr{T}}\|^2_{L^2(\Omega)} \\
&\quad + \sum_{\gamma\in E_I} h_\gamma \left(\|[\![p - \Pi(p)]\!]\|^2_{L^2(\gamma)} + \|[\![\Pi(p) - p_{\mathscr{T}}]\!]\|^2_{L^2(\gamma)}\right)\Big) = 2(I + II).
\end{aligned}$$

Antes de continuar, es que vamos a introducir la siguiente notación:

$$\mathbf{e}^u = \mathbf{u} - \mathbf{u}_{\mathscr{T}} = \mathbf{u} - I_L(\mathbf{u}) + I_L(\mathbf{u}) - \mathbf{u}_{\mathscr{T}} = \boldsymbol{\eta}^u + \mathbf{e}^u_{\mathscr{T}},$$
$$e^p = p - p_{\mathscr{T}} = p - \Pi(p) + \Pi(p) - p_{\mathscr{T}} = \eta^p + e^p_{\mathscr{T}}.$$

En base a estas nuevas variables, tenemos que

$$I = \|(\boldsymbol{\eta}^u, \eta^p)\|^2_{\mathbb{X}(\mathscr{T})}, \qquad II = \|(\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})\|^2_{\mathbb{X}(\mathscr{T})}.$$

Primero acotaremos el término I:

$$I = \varepsilon\|\nabla(\mathbf{u} - I_L(\mathsf{u}))\|^2_{L^2(\Omega)} + \|p - \Pi(p)\|^2_{L^2(\Omega)} + \sum_{\gamma\in E_I} h_\gamma \|[\![p - \Pi(p)]\!]\|^2_{L^2(\gamma)}.$$

Usando las propiedades del interpolador I_L, se tiene que

$$\varepsilon\|\nabla(\mathbf{u} - I_L(\mathsf{u}))\|^2_{L^2(\Omega)} \leq \varepsilon \mathrm{C}_L^2 h^2 |\mathbf{u}|^2_{H^2(\Omega)}.$$

Usando ahora las propiedades de la proyección ortogonal Π, se tiene que

$$\|p - \Pi(p)\|^2_{L^2(\Omega)} \leq \mathrm{C}_\Pi^2 h^2 \|\nabla p\|^2_{L^2(\Omega)},$$
$$\sum_{\gamma\in E_I} h_\gamma \|[\![p - \Pi(p)]\!]\|^2_{L^2(\gamma)} \leq \mathrm{C}_{\Pi,\mathscr{T}} h^2 \|\nabla p\|^2_{L^2(\Omega)}.$$

En base a lo anterior, es que podemos concluir que

$$I \leq \text{máx}\{\varepsilon \mathrm{C}_L^2, \mathrm{C}_\Pi^2, \mathrm{C}_{\Pi,\mathscr{T}}\} h^2 \left(|\mathbf{u}|^2_{H^2(\Omega)} + \|\nabla p\|^2_{L^2(\Omega)} \right).$$

Ahora, procedemos a acotar el término II:

$$II = \varepsilon \|\nabla \mathbf{e}^u_{\mathscr{T}}\|^2_{L^2(\Omega)} + \|e^p_{\mathscr{T}}\|^2_{L^2(\Omega)} + \sum_{\gamma \in E_I} h_\gamma \|[\![e^p_{\mathscr{T}}]\!]\|^2_{L^2(\gamma)} = \|(\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})\|^2_{\mathbb{X}(\mathscr{T})}.$$

Recordemos, que la forma bilineal $\mathfrak{B}$ satisface la propiedad de coercividad para funciones discretas. Luego, se tiene que

$$II = \|(\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})\|^2_{\mathbb{X}(\mathscr{T})} \leq \mathfrak{B}((\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}}), (\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})).$$

Ahora, recordemos que

$$\mathsf{e}^u_{\mathscr{T}} = \mathbf{e}^u - \eta^u, \quad e^p_{\mathscr{T}} = e^p - \eta^p.$$

En base a lo anterior, podemos ahora escibir

$$\mathfrak{B}((\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}}), (\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})) = III + IV,$$

donde

$$III = \mathfrak{B}((\mathbf{e}^u, e^p), (\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})) = \mathcal{A}(\mathbf{e}^u, \mathbf{e}^u_{\mathscr{T}}) + \mathcal{B}(e^p, \mathbf{e}^u_{\mathscr{T}}) - \mathcal{B}(e^p, \mathbf{e}^u_{\mathscr{T}}) - \mathcal{S}(e^p, e^p_{\mathscr{T}}),$$

y

$$IV = -\mathfrak{B}((\eta^u, \eta^p), (\mathbf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})) = -\mathcal{A}(\eta^u, \mathbf{e}^u_{\mathscr{T}}) - \mathcal{B}(\eta^p, \mathbf{e}^u_{\mathscr{T}}) + \mathcal{B}(\eta^p, \mathbf{e}^u_{\mathscr{T}}) + \mathcal{S}(\eta^p, e^p_{\mathscr{T}}).$$

Primero acotaremos IV, usando la propiedad de continuidad y la definición del término I, como sigue

$$IV = -\mathfrak{B}((\eta^u, \eta^p), (\mathsf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})) \leq \|(\eta^u, \eta^p)\|_{\mathbb{X}(\mathscr{T})} \|(\mathsf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})} = I \|(\mathsf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})\|_{\mathbb{X}(\mathscr{T})}.$$

Ahora, notemos que de la propiedad de ortogonalidad de Galerkin, tenemos que

$$III = \mathfrak{B}((\mathsf{e}^u, e^p), (\mathsf{e}^u_{\mathscr{T}}, e^p_{\mathscr{T}})) = 0.$$

Con lo cual podemos concluir que

$$II \leq I.$$

Finalmente, juntando todo el análisis anterior, finalmente podemos concluir que

$$\|(\mathsf{e}^u, e^p)\|_{\mathbb{X}(\mathscr{T})} \leq \mathfrak{C}_{max} h \left(|\mathbf{u}|_{H^2(\Omega)} + \|\nabla p\|_{L^2(\Omega)} \right).$$

6.3. Código de Stokes. Las diferencias en el código se limitan a la adición de ciertas funciones de apoyo al cálculo y a variaciones en el código del ensamble de la matriz y el lado derecho de la ecuación

```
1  void get_RHS(int nt,const VectorXi& subdomain,const MatrixXi& elements,const MatrixXd&
       vertices,int nbf,const VectorXi& boundary_subdomain,const MatrixXi& boundary,std::vector<
       double>& RHS, int nv,const Matrix<double, 3, 2> normals[],const VectorXd& areas,int exam,
       const VectorXi& Dnodes){
2      for (int k=0; k<nt; k++) {
3          Vector3d fmom1 = Vector3d::Zero();
4          Vector3d fmom2 = Vector3d::Zero();
5          for (int j=0; j<NTRIQP; j++) {
6              Vector2d X=triquad[j][0]*vertices.row(elements(k,0))+triquad[j][1]*vertices.row(
                   elements(k,1))+triquad[j][2]*vertices.row(elements(k,2));
7              double w_f1=triweight[j]*get_f1(X,subdomain(k),exam);
8              double w_f2=triweight[j]*get_f2(X,subdomain(k),exam);
9              for (int local_node=0; local_node<3; local_node++) {
10                 fmom1(local_node)+=w_f1*triquad[j][local_node];
11                 fmom2(local_node)+=w_f2*triquad[j][local_node];
12             }
13         }
14         for (int local_node=0; local_node<3; local_node++) {
15             if (Dnodes(elements(k,local_node))>=0){
16                 RHS[elements(k,local_node)]+=areas(k)*(fmom1(local_node));
17                 RHS[elements(k,local_node)+nv]+=areas(k)*(fmom2(local_node));
18             }
19             else {
20                 RHS[elements(k,local_node)]=0;
21                 RHS[elements(k,local_node)+nv]=0;
22             }
23         }
24     }
25 }
```

LISTING 10. Ensamble de lado derecho para esquema de stokes

```
1  void get_stiffnessmatrix(int nv,int nbf,const VectorXi& boundary_subdomain,const MatrixXi&
       boundary,int nt,const MatrixXi& elements,const MatrixXd& vertices,SparseMatrix<double>& K
       ,const MatrixXi& edgeelement,const MatrixXi& edgeedge,const Matrix<double, 3, 2> normals
       [],const VectorXd& areas,const VectorXi& Dnodes){
2      int edges1[3]={1,2,0};
3      int edges2[3]={2,0,1};
4      double val;
5      int k_prime;
6      double hf;
7      for (int k=0; k<nt; k++) {
8          for (int local_node_a=0; local_node_a<3; local_node_a++) {
9              k_prime=edgeelement(k,local_node_a);
10             if (k_prime>=0){
11                 hf=get_hf(vertices.row(elements(k,edges1[local_node_a])),vertices.row(
                       elements(k,edges2[local_node_a])));
12                 K.coeffRef(2*nv+k,2*nv+k)+= -pow(hf,2)/(12.0*get_epsilon());
13                 K.coeffRef(2*nv+k,2*nv+k_prime)+= +pow(hf,2)/(12.0*get_epsilon());
14             }
15             if (Dnodes(elements(k,local_node_a))>=0) {
16                 for (int local_node_b=0; local_node_b<3; local_node_b++) {
17                     if (Dnodes(elements(k,local_node_b))>=0) {
18                         val=get_epsilon()*normals[k].row(local_node_a).dot(normals[k].row(
                               local_node_b))/(4*areas(k));
19                         K.coeffRef(elements(k,local_node_a),elements(k,local_node_b))+=val;
20                         K.coeffRef(elements(k,local_node_a)+nv,elements(k,local_node_b)+nv)+=
                               val;
21                     }
22                 }
23                 K.coeffRef(elements(k,local_node_a),2*nv+k)+=0.5*normals[k](local_node_a,0);
```

```
24                  K.coeffRef(elements(k,local_node_a)+nv,2*nv+k)+=0.5*normals[k](local_node_a
                        ,1);
25
26                  K.coeffRef(2*nv+k,elements(k,local_node_a))+= 0.5*normals[k](local_node_a,0);
27                  K.coeffRef(2*nv+k,elements(k,local_node_a)+nv)+= 0.5*normals[k](local_node_a
                        ,1);
28              }
29              else {
30                  K.coeffRef(elements(k,local_node_a),elements(k,local_node_a))=1;
31                  K.coeffRef(elements(k,local_node_a)+nv,elements(k,local_node_a)+nv)=1;
32              }
33          }
34          K.coeffRef(2*nv+k,2*nv+nt)=areas(k);
35          K.coeffRef(2*nv+nt,2*nv+k)=areas(k);
36      }
37  }
```

Listing 11. Ensamble de matriz para esquema de Stokes

6.4. Ejemplos numéricos. En esta sección, nos ocuparemos de testear nuestro código de elementos finitos, con un ejemplo en donde construiremos una solución análitica y otros dos ejemplos, sin solución analítica. Recordamos que la solución a cualquiera de nuestros problemas, nos entregará el campo de velocidades y de presión asociado al fluido presente en una región Ω.

Ejemplo 1: Consideraremos que el dominio de nuestro problema es el cuadrado unitario $\Omega = (0,1) \times (0,1)$ y se tiene la siguiente solución analítca

$$\mathbf{u}(x,y) = \operatorname{curl}\left(xy(1-x)(1-y)\right)^2,$$
$$p(x,y) = xy(1-x)(1-y) - \frac{1}{144}.$$

en donde, para el cálculo del lado derecho usamos que $\varepsilon = 1$.

En la Figura 26, mostramos la solución de elementos finitos estabilizada, en donde se presenta la magnitud del campo vectorial de la velocidad, es decir, $|\mathbf{u}_{\mathscr{T}}|$, como también mostramos las presiones del fluido $p_{\mathscr{T}}$. En la Figura 27 se uestra la tasa de convergencia del método de elementos finitos estabilizado, en donde podemos apreciar la tasa de convergencia óptima del método.

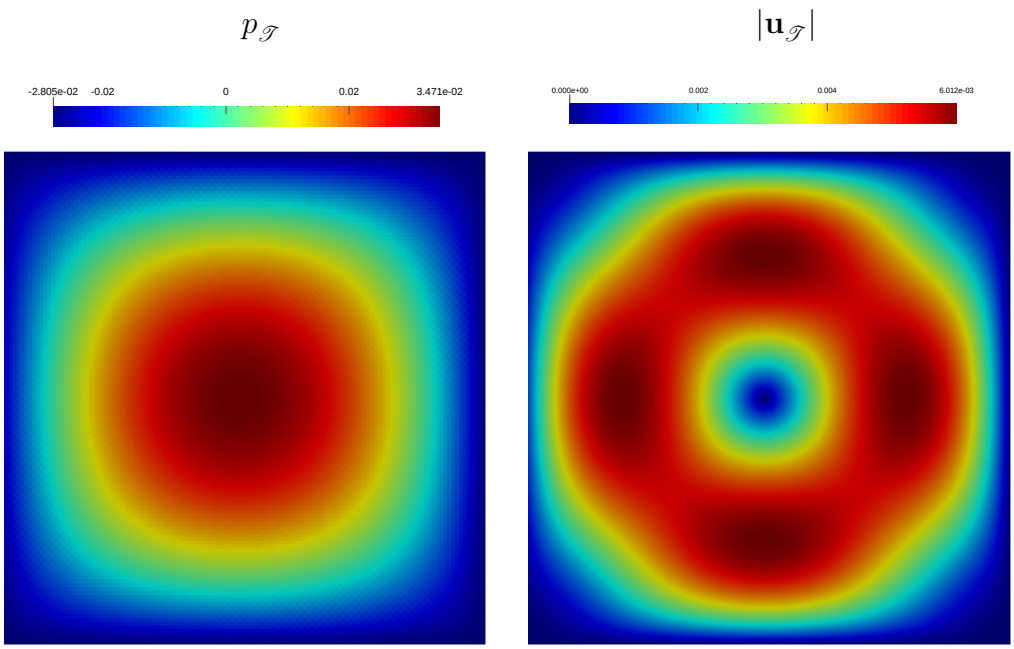

FIGURA 26. Ejemplo 1: Solución del método de elementos finitos estabilizados, con $p_{\mathscr{T}}$ (izquierda) y $\mathbf{u}_{\mathscr{T}}$ (derecha)

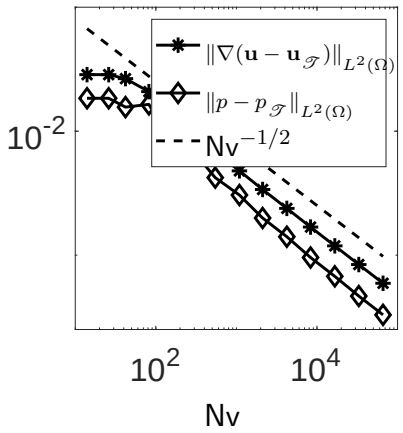

FIGURA 27. Ejemplo 1: Convergencia del método de elementos finitos estabilizado $(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}})$, donde la convergencia lineal corresponde a $h = \frac{1}{\mathrm{Nv}^{1/2}}$.

Ejemplo 2: Consideraremos el mismo dominio de las secciones anteriores, pero con las condiciones de borde descritas en la Figura 28. Notemos, que como el problema se trata de una dinámica de fluido, es que en el lado izquierdo de la ciudad consideramos una entrada del tipo parabólica y una salida de dicho fluido con las mismas características. En el resto de la frontera consideramos una condición de borde nula y tomamos $\varepsilon = 1$. En la Figura 29, se muestran las soluciones del método de elementos finitos estabilizado, tanto para la presión discreta $p_{\mathscr{T}}$, como también para la magnitud del campo de velocidad $|\mathbf{u}_{\mathscr{T}}|$.

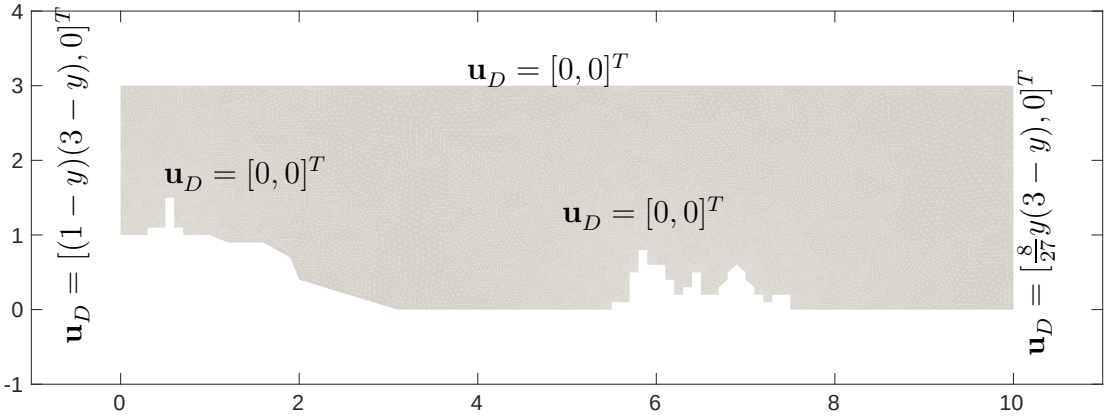

FIGURA 28. Dominio y condiciones de frontera para el Ejemplo 2.

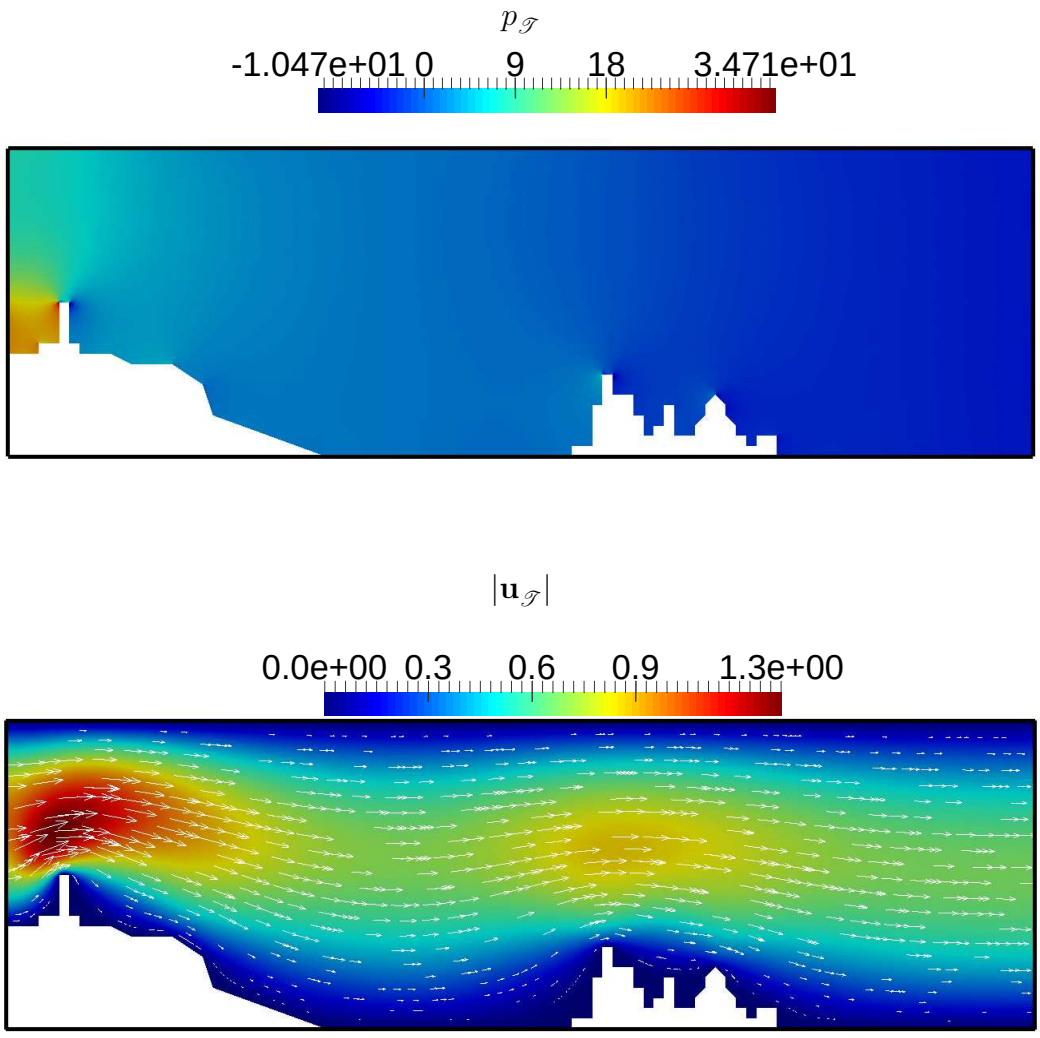

FIGURA 29. Ejemplo 2: solución de elementos finitos estabilizada, para las presiones $p_{\mathscr{T}}$ (superior), y la magnitud del campo de velocidad $|\mathbf{u}_{\mathscr{T}}|$ (inferior).

7. Un método de elementos finitos estabilizados para un problema de Navier–Stokes

Ahora, estamos interazados en resolver el problema no lineal de Navier–Stokes. Es decir, queremos resolver el problema

Hallar $(\mathbf{u}, p)$ ***tal que,***

$$\left\{\begin{array}{rcll} -\varepsilon\Delta\mathbf{u}(\boldsymbol{x}) + \mathbf{u}(\boldsymbol{x})\cdot\nabla\mathbf{u}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) &=& \mathbf{f}(\boldsymbol{x}) & \forall\ \boldsymbol{x}\in\Omega, \\ \text{div } \mathbf{u}(\boldsymbol{x}) &=& 0 & \forall\ \boldsymbol{x}\in\partial\Omega, \\ \mathbf{u}(\boldsymbol{x}) &=& \mathbf{0} & \forall\ \boldsymbol{x}\in\partial\Omega, \end{array}\right.$$

Para resolver el problema de Navier–Stokes, es que vamos a considerar, su formulación débil no lineal, la cual corresponde a:

Encontrar $(\mathbf{u}, p) \in [H_0^1(\Omega)]^d \times L_0^2(\Omega)$ tal que

$$\left\{\begin{array}{rcll} \mathcal{A}(\mathbf{u},\mathbf{v}) + \mathcal{C}(\mathbf{u};\mathbf{u},\mathbf{v}) + \mathcal{B}(p,\mathbf{v}) &=& \mathcal{F}(\mathbf{v}), & \forall\ \mathbf{v}\in[H_0^1(\Omega)]^d, \\ \mathcal{B}(q,\mathbf{u}) &=& 0, & \forall\ q\in L_0^2(\Omega), \end{array}\right. \tag{7.1}$$

donde las formas lineales $\mathcal{A}, \mathcal{B}$ son las mismas que de la Sección anterior, y la nueva forma no lineal $\mathcal{C}$, está dada por:

$$\mathcal{C}(\mathbf{u};\mathbf{w},\mathbf{v}) = \int_\Omega (\mathbf{u}\nabla\mathbf{w})\cdot\mathbf{v}\ d\boldsymbol{x} = \int_\Omega \begin{bmatrix} \mathbf{u}\cdot\nabla w_1 \\ \vdots \\ \mathbf{u}\cdot\nabla w_d \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix} d\boldsymbol{x} = \sum_{i=1}^{d} \int_\Omega \mathbf{u}\cdot\nabla w_i v_i\ d\boldsymbol{x}.$$

Es sabido que la formulación débil del problema de Navier–Stokes, tiene única solución, bajo el siguiente supuesto: si $\mathbf{f}\in[L^2(\Omega)]^d$,

$$\|\nabla\mathbf{u}\|_{L^2(\Omega)} \leq \frac{C_N}{C_{E,P,\Omega}}\varepsilon,$$

donde $C_N \in [0,1)$ y la constante $C_{E,P,\Omega}$ dependen de la constante de Poincaré y una constante de incrustación entre un espacio de Sobolev y un espacio de Lebesgue.

Para la discretización del problema, es que consideraremos la siguiente discretización estabilizada de elementos finitos:

Encontrar $(\mathbf{u}_\mathscr{T}, p_\mathscr{T}) \in [\mathbb{V}(\mathscr{T})]^d \times \mathbb{Q}(\mathscr{T})$ tal que

$$\left\{\begin{array}{rcll} \mathcal{A}(\mathbf{u}_\mathscr{T},\mathbf{v}_\mathscr{T}) + \mathcal{C}(\mathbf{u}_\mathscr{T};\mathbf{u}_\mathscr{T},\mathbf{v}_\mathscr{T}) + \mathcal{B}(p_\mathscr{T},\mathbf{v}_\mathscr{T}) &=& \mathcal{F}(\mathbf{v}_\mathscr{T}), & \forall\ \mathbf{v}_\mathscr{T}\in[\mathbb{V}(\mathscr{T})]^d, \\ \mathcal{B}(q_\mathscr{T},\mathbf{u}_\mathscr{T}) + \mathcal{S}(p_\mathscr{T},q_\mathscr{T}) &=& 0, & \forall\ q_\mathscr{T}\in\mathbb{Q}(\mathscr{T}). \end{array}\right. \tag{7.2}$$

7.1. Un método de punto fijo. Para poder construir un algoritmo iterativo de tipo punto fijo, es que tenemos que construir una aplicación:

$$\begin{array}{rccc} \mathtt{T}: & \mathbb{M} & \to & \mathbb{M} \\ & (\boldsymbol{\zeta}_{\mathscr{T}}, \xi_{\mathscr{T}}) & \to & \mathtt{T}(\boldsymbol{\zeta}_{\mathscr{T}}, \xi_{\mathscr{T}}) = (\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}}) \end{array} \tag{7.3}$$

donde

$$\mathbb{M} \subset [\mathbb{V}(\mathscr{T})]^d \times \mathbb{Q}(\mathscr{T}),$$

y $(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}})$ es solución del problema

$$\begin{cases} \mathcal{A}(\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\boldsymbol{\zeta}_{\mathscr{T}}; \mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(p_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) &= \mathcal{F}(\mathbf{v}_{\mathscr{T}}), & \forall\ \mathbf{v}_{\mathscr{T}} \in [\mathbb{V}(\mathscr{T})]^d, \\ \mathcal{B}(q_{\mathscr{T}}, \mathbf{u}_{\mathscr{T}}) + \mathcal{S}(p_{\mathscr{T}}, q_{\mathscr{T}}) &= 0, & \forall\ q_{\mathscr{T}} \in \mathbb{Q}(\mathscr{T}). \end{cases} \tag{7.4}$$

Notemos que el punto fijo de la aplicación, es decir,

$$\mathtt{T}(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}}) = (\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}}),$$

es justamentamente una solución para el problema no lineal (7.2).

El estudio de la existencia y unicidad para la solución del problema de punto fijo se encuentra completamente tratada en los textos y articulos científicos [17, 7, 15, 13, 11].

Para su aproximación, un esquema usual es considerar un proceso iterativo , en donde se toma $(\mathbf{u}^0_{\mathscr{T}}, p^0_{\mathscr{T}}) \in M$, y definimos una secuencia $\{(\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}})\}$ mediante la siguiente formula iterativa

$$(\mathbf{u}^{n+1}_{\mathscr{T}}, p^{n+1}_{\mathscr{T}}) = \mathtt{T}(\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}}), \quad n = 0, 1, ...$$

El algoritmo de punto fijo anterior, luego se puede reescribir como sigue:

Dado $(\mathbf{u}^0_{\mathscr{T}}, p^0_{\mathscr{T}}) \in [\mathbb{V}(\mathscr{T})]^d \times \mathbb{Q}(\mathscr{T})$, encontrar $(\mathbf{u}^{n+1}_{\mathscr{T}}, p^{n+1}_{\mathscr{T}}) \in [\mathbb{V}(\mathscr{T})]^d \times \mathbb{Q}(\mathscr{T})$ tal que

$$\begin{cases} \mathcal{A}(\mathbf{u}^{n+1}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{T}}; \mathbf{u}^{n+1}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(p^{n+1}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) &= \mathcal{F}(\mathbf{v}_{\mathscr{T}}), & \forall\ \mathbf{v}_{\mathscr{T}} \in [\mathbb{V}(\mathscr{T})]^d, \\ \mathcal{B}(q_{\mathscr{T}}, \mathbf{u}^{n+1}_{\mathscr{T}}) + \mathcal{S}(p^{n+1}_{\mathscr{T}}, q_{\mathscr{T}}) &= 0, & \forall\ q_{\mathscr{T}} \in \mathbb{Q}(\mathscr{T}). \end{cases} \tag{7.5}$$

El algoritmo anterior tendrá como criterio de parada

$$\max_{\boldsymbol{x} \in \mathcal{V}(\mathscr{T})} \|(\mathbf{u}^{n+1}_{\mathscr{T}}, p^{n+1}_{\mathscr{T}}) - (\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}})\|_2 \leq \text{tol}.$$

con una tolerancia dada, que usualmente es $\text{tol} = 10^{-8}$.

Por último, recordaremos un resultado de convergencia para el método de punto fijo.

Teorema 3 (Teorema del punto fijo de Banach)**.** *Supongamos que M es un conjunto cerrado no vacio dentro de un espacio de Banach V, y además, $\mathtt{T} : M \to M$ es un operador contractivo, con constante de contración α, donde $0 \leq \alpha < 1$. Entonces,*

para cualquier $u_0 \in \mathbb{M}$, la sucesión $\{u_n\} \subset K$ definida por $u_{n+1} = \mathtt{T}(u_n)$, $n = 0, 1, ...$, converge a u, es decir,

$$\|u - u_n\|_V \to 0 \text{ para } n \to \infty.$$

Además, la siguiente cota es válidas para el error

$$\|u - u_n\|_V \leq \frac{\alpha^n}{1-\alpha}\|u_0 - u_1\|_V. \tag{7.6}$$

Demostración. Como $\mathtt{T} : \mathbb{M} \to \mathbb{M}$, la sucesión $\{u_n\}$ está bien definida. Hora, demostraremos que $\{u_n\}$ es una sucesión de Cauchy. Usando el hecho que $\mathtt{T}$ es contractiva, se tiene que

$$\|u_{n+1} - u_n\|_V \leq \alpha\|u_n - u_{n-1}\|_V \leq \cdots \leq \alpha^n\|u_1 - u_0\|_V.$$

Luego, para cualquier $m \geq n \geq 1$,

$$\begin{aligned}\|u_m - u_n\|_V &\leq \sum_{j=0}^{m-n-1} \|u_{n+j+1} - u_{n+j}\|_V \\ &\leq \sum_{j=0}^{m-n-1} \alpha^{n+j}\|u_1 - u_0\|_V \\ &\leq \frac{\alpha^n}{1-\alpha}\|u_1 - u_0\|_V.\end{aligned}$$

Ahora, como $\alpha \in [0, 1)$, $\|u_m - u_n\|_V \to 0$ para cuando $m, n \to \infty$, luego $\{u_n\}$ es de Cauchy; y como $\mathbb{M}$ es un cerrado, luego $\{u_n\}$ tiene un límite $u \in K$. Tomando el límite cuando $n \to \infty$ en $u_{n+1} = \mathtt{T}(u_n)$ se tiene que $u = \mathtt{T}(u)$ usando la continuidad de $\mathtt{T}$, luego u es un punto fijo para $\mathtt{T}$. □

7.2. Un método de Newton. Primero, introduciremos un marco general para obtener un método de Newton.

Sean U, V dos espacios de Hilbert, $F : U \to V$ un operador Fréchet diferenciable, es decir, existe $F' \in \mathcal{L}(U, V)$ tal que

$$\lim_{t\to 0} \frac{F(u + th) - F(u)}{t} = F'(u)(h), \quad \forall\ h \in U,$$

donde $\mathcal{L}(U, V)$ es el conjunto de todos los funciones lineales continuos y acotados, que toman elementos de U y los transforma en elementos en V, con norma

$$\|F'(u)\|_{\mathcal{L}(U,V)} = \sup_{h\in U} \frac{\|F'(u)(h)\|_V}{\|h\|_U}.$$

Queremos resolver el siguiente problema

$$F(u) = 0.$$

El método de Newton asociado es como sigue: considerando $u_0 \in U$, para $n = 0, 1, ...$ calcular

$$u_{n+1} = u_n - [F'(u_n)]F(u_n).$$

De forma equivalente, se puede calcular el incremental

$$F'(u_n)(\delta u) = -F(u_n),$$

y luego hacemos

$$u^{n+1} = \delta u + u_n.$$

El siguiente resultado presenta un resultado de convergencia (ver [3, Teorema 4.4.1]).

Teorema 4 (Convergencia local)**.** *Asumamos que u^* es una raiz de la ecuación para la cual $[F'(u^*)]^{-1}$ existe y además es una aplicación continua de V sobre U. Asumamos además que $F'(u)$ es localmente Lipschitz continua en u^*, es decir,*

$$\|F'(u) - F'(v)\|_{\mathcal{L}(U,V)} \leq L\|u - v\|_U, \quad \forall\ u, v \in N(u^*),$$

donde $N(u^*)$ es una vecindad de u^*, y $L > 0$ es una constante. Entonces, existe $\delta > 0$ tal que $\|u_0 - u^*\| \leq \delta$, la sucesión de Newton $\{u_n\}$ está bien definida y converge a u^*. Además, existe $M > 0$ tal que

$$\|u_{n+1} - u^*\|_U \leq M\|u_n - u^*\|_U^2.$$

Demostración. Asumamos que $[F'(u)]^{-1}$ existe en $N(u^*)$ y sea $c_0 = \sup_{u \in N(u^*)} \|[F'(u)]^{-1}\| \leq \infty$. Definamos ahora

$$T(u) = u - [F'(u)]^{-1}F(u), \quad u \in N(u^*).$$

Notar que $T(u^*) = u^*$. Para $u \in N(u^*)$, se tiene que

$$\begin{aligned} T(u) - T(u^*) &= u - u^* - [F'(u)]^{-1}F(u) \\ &= [F'(u)]^{-1}\left\{F(u^*) - F(u) - F'(u)(u^* - u)\right\} \\ &= [F'(u)]^{-1}(u^* - u)\int_0^1 (F'(u + t(u^* - u)) - F'(u))dt, \end{aligned}$$

y ahora tomando norma, se tiene

$$\begin{aligned} \|T(u) - T(u^*)\|_U &\leq \|[F'(u)]^{-1}\|_{\mathcal{L}(V,U)}\|u^* - u\|_U \left\|\int_0^1 (F'(u + t(u^* - u)) - F'(u))dt\right\| \\ &\leq \|[F'(u)]^{-1}\|_{\mathcal{L}(V,U)}\|u^* - u\|_U \left\|\int_0^1 Lt\|u^* - u\|_U dt\right\|, \end{aligned}$$

luego

$$\|T(u) - T(u^*)\|_U \leq \frac{c_0 L}{2}\|u - u^*\|_U^2.$$

Ahora, tomando $\delta < \frac{2}{c_0 L}$ con la propiedad que $\overline{B}(u^*, \delta) \subset N(u^*)$, luego $T : \overline{B}(u^*, \delta) \to \overline{B}(u^*, \delta)$ es una α-contracción con $\alpha = \frac{c_0 L \delta}{2} < 1$. Entonces, del teorema del punto fijo de Banach, T tiene un único punto fijo. □

Notar que de la demostración anterior se tiene convergencia cuadrática para el método de Newton.

Finalmente, para obtener nuestro método de Newton, definimos la siguiente función

$$\mathtt{F}_{\mathscr{J}} : \mathcal{H} \to \mathcal{H}'$$

donde $\mathcal{H}'$ corresponde al espacio dual topológico de $\mathcal{H}$, $\mathcal{H} = [\mathbb{V}(\mathscr{T})]^d \times \mathbb{Q}(\mathscr{T})$ y la acción de la función se define como

$$\langle \mathtt{F}_{\mathscr{J}}(\mathbf{u}_{\mathscr{J}}, p_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}}) \rangle_{\mathcal{H}' \times \mathcal{H}} := \mathfrak{A}((\mathbf{u}_{\mathscr{J}}, p_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}})) - \mathfrak{Z}((\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}})),$$

donde

$$\begin{aligned}\mathfrak{A}((\mathbf{u}_{\mathscr{J}}, p_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}})) = \mathcal{A}(\mathbf{u}_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{C}(\mathbf{u}_{\mathscr{J}}; \mathbf{u}_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{B}(p_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) \\ - \mathcal{B}(q_{\mathscr{J}}, \mathbf{u}_{\mathscr{J}}) - \mathcal{S}(p_{\mathscr{J}}, q_{\mathscr{J}})\end{aligned}$$

y

$$\mathfrak{Z}((\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}})) = \mathcal{F}(\mathbf{v}_{\mathscr{J}}).$$

Recordemos que la iteración de Newton está basada en el cálculo del incremental $(\delta \mathbf{u}_{\mathscr{J}}, \delta p_{\mathscr{J}})$ tal que

$$\langle \mathtt{F}'_{\mathscr{J}}(\mathbf{u}^n_{\mathscr{J}}, p^n_{\mathscr{J}})(\delta \mathbf{u}_{\mathscr{J}}, \delta p_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}}) \rangle_{\mathcal{H}' \times \mathcal{H}} = \langle -\mathtt{F}_{\mathscr{J}}(\mathbf{u}^n_{\mathscr{J}}, p^n_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}}) \rangle_{\mathcal{H}' \times \mathcal{H}}$$

donde $\mathtt{F}'_{\mathscr{J}}(\mathbf{u}^n_{\mathscr{J}}, p^n_{\mathscr{J}})(\delta \mathbf{u}_{\mathscr{J}}, \delta p_{\mathscr{J}})$ denota la derivada de $\mathtt{F}_{\mathscr{J}}$ en $(\mathbf{u}^n_{\mathscr{J}}, p^n_{\mathscr{J}})$ evaluada en la dirección $(\delta \mathbf{u}_{\mathscr{J}}, \delta p_{\mathscr{J}})$, es decir,

$$\begin{aligned}&\langle \mathtt{F}'_{\mathscr{J}}(\mathbf{u}^n_{\mathscr{J}}, p^n_{\mathscr{J}})(\delta \mathbf{u}_{\mathscr{J}}, \delta p_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}}) \rangle_{\mathcal{H}' \times \mathcal{H}} \\ &= \lim_{t \to 0} \frac{\langle \mathtt{F}_{\mathscr{J}}(\mathbf{u}^n_{\mathscr{J}} + t\delta \mathbf{u}_{\mathscr{J}}, p^n_{\mathscr{J}} + t\delta p_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}}) \rangle_{\mathcal{H}' \times \mathcal{H}} - \langle \mathtt{F}_{\mathscr{J}}(\mathbf{u}^n_{\mathscr{J}}, p^n_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}}) \rangle_{\mathcal{H}' \times \mathcal{H}}}{t}.\end{aligned}$$

Utilizando la definición de la aplicación $\mathtt{F}_{\mathscr{J}}$, se tiene que

$$\begin{aligned}&\langle \mathtt{F}_{\mathscr{J}}(\mathbf{u}^n_{\mathscr{J}} + t\delta \mathbf{u}_{\mathscr{J}}, p^n_{\mathscr{J}} + t\delta p_{\mathscr{J}}), (\mathbf{v}_{\mathscr{J}}, q_{\mathscr{J}}) \rangle_{\mathcal{H}' \times \mathcal{H}} \\ &= \mathcal{A}(\mathbf{u}^n_{\mathscr{J}} + t\delta \mathbf{u}_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{J}} + t\delta \mathbf{u}_{\mathscr{J}}; \mathbf{u}^n_{\mathscr{J}} + t\delta \mathbf{u}_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{B}(p^n_{\mathscr{J}} + t\delta p_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) \\ &\quad - \mathcal{B}(q_{\mathscr{J}}, \mathbf{u}^n_{\mathscr{J}} + t\delta \mathbf{u}_{\mathscr{J}}) - \mathcal{S}(p^n_{\mathscr{J}} + t\delta p_{\mathscr{J}}, q_{\mathscr{J}}) - \mathcal{F}(\mathbf{v}_{\mathscr{J}}) \\ &= \mathcal{A}(\mathbf{u}^n_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{J}}; \mathbf{u}^n_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{B}(p^n_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) - \mathcal{B}(q_{\mathscr{J}}, \mathbf{u}^n_{\mathscr{J}}) - \mathcal{S}(p^n_{\mathscr{J}}, q_{\mathscr{J}}) - \mathcal{F}(\mathbf{v}_{\mathscr{J}}) \\ &+ t(\mathcal{A}(\delta \mathbf{u}_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{J}}; \delta \mathbf{u}_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) + \mathcal{C}(\delta \mathbf{u}_{\mathscr{J}}; \mathbf{u}^n_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) \\ &\qquad + \mathcal{B}(\delta p_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}}) - \mathcal{B}(q_{\mathscr{J}}, \delta \mathbf{u}_{\mathscr{J}}) - \mathcal{S}(\delta p_{\mathscr{J}}, q_{\mathscr{J}})) \\ &+ t^2 \mathcal{C}(\delta \mathbf{u}_{\mathscr{J}}; \delta \mathbf{u}_{\mathscr{J}}, \mathbf{v}_{\mathscr{J}})\end{aligned}$$

De forma similar, tenemos que

$$\begin{aligned}&\langle \mathtt{F}_{\mathscr{T}}(\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})\rangle_{\mathcal{H}'\times\mathcal{H}}\\ &= \mathcal{A}(\mathbf{u}^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{T}}; \mathbf{u}^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(p^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) - \mathcal{B}(q_{\mathscr{T}}, \mathbf{u}^n_{\mathscr{T}}) - \mathcal{S}(p^n_{\mathscr{T}}, q_{\mathscr{T}}) - \mathcal{F}(\mathbf{v}_{\mathscr{T}})\end{aligned}$$

Luego, restando los dos términos anteriores, se tiene que

$$\begin{aligned}&\langle \mathtt{F}_{\mathscr{T}}(\mathbf{u}^n_{\mathscr{T}} + t\delta\mathbf{u}_{\mathscr{T}}, p^n_{\mathscr{T}} + t\delta p_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})\rangle_{\mathcal{H}'\times\mathcal{H}} - \mathtt{F}_{\mathscr{T}}(\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})\rangle_{\mathcal{H}'\times\mathcal{H}}\\ &= t(\mathcal{A}(\delta\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{T}}; \delta\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\delta\mathbf{u}_{\mathscr{T}}; \mathbf{u}^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(\delta p_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) - \mathcal{B}(q_{\mathscr{T}}, \delta\mathbf{u}_{\mathscr{T}})\\ &\quad - \mathcal{S}(\delta p_{\mathscr{T}}, q_{\mathscr{T}}))\\ &+ t^2\mathcal{C}(\delta\mathbf{u}_{\mathscr{T}}; \delta\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}})\end{aligned}$$

Diviendo la expresión anterior por t y haciendo $t \to 0$, podemos concluir que

$$\begin{aligned}&\langle \mathtt{F}'_{\mathscr{T}}(\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}})(\delta\mathbf{u}_{\mathscr{T}}, \delta p_{\mathscr{T}}), (\mathbf{v}_{\mathscr{T}}, q_{\mathscr{T}})\rangle_{\mathcal{H}'\times\mathcal{H}}\\ &= \mathcal{A}(\delta\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{T}}; \delta\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\delta\mathbf{u}_{\mathscr{T}}; \mathbf{u}^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(\delta p_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) - \mathcal{B}(q_{\mathscr{T}}, \delta\mathbf{u}_{\mathscr{T}})\\ &\quad - \mathcal{S}(\delta p_{\mathscr{T}}, q_{\mathscr{T}})\end{aligned}$$

Finalmente, podemos establecer el siguiente método de Newton:

Dado $(\mathbf{u}^0_{\mathscr{T}}, p^0_{\mathscr{T}})$, calcular $(\delta\mathbf{u}_{\mathscr{T}}, \delta p_{\mathscr{T}})$, con $n = 0, 1, 2, ...$ solución de:

$$\begin{aligned}&\mathcal{A}(\delta\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\mathbf{u}^n_{\mathscr{T}}; \delta\mathbf{u}_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{C}(\delta\mathbf{u}_{\mathscr{T}}; \mathbf{u}^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(\delta p_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) - \mathcal{B}(q_{\mathscr{T}}, \delta\mathbf{u}_{\mathscr{T}})\\ &\quad - \mathcal{S}(\delta p_{\mathscr{T}}, q_{\mathscr{T}})\\ &= \mathcal{F}(\mathbf{v}_{\mathscr{T}}) - \mathcal{A}(\mathbf{u}^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) - \mathcal{C}(\mathbf{u}^n_{\mathscr{T}}; \mathbf{u}^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) - \mathcal{B}(p^n_{\mathscr{T}}, \mathbf{v}_{\mathscr{T}}) + \mathcal{B}(q_{\mathscr{T}}, \mathbf{u}^n_{\mathscr{T}}) + \mathcal{S}(p^n_{\mathscr{T}}, q_{\mathscr{T}}).\end{aligned}$$

La siguiente iteración es

$$(\mathbf{u}^{n+1}_{\mathscr{T}}, p^{n+1}_{\mathscr{T}}) = (\delta\mathbf{u}_{\mathscr{T}}, \delta p_{\mathscr{T}}) + (\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}}).$$

El criterio de parada es:

$$\|(\mathbf{u}^{n+1}_{\mathscr{T}}, p^{n+1}_{\mathscr{T}}) - (\mathbf{u}^n_{\mathscr{T}}, p^n_{\mathscr{T}})\|_2 \leq \text{tol}.$$

7.3. Código del método estabilizado para Navier Stokes. En este caso además de producirse los esperados cambios en la sección de ensamble de la matriz y el lado derecho, es necesario incorporar el código del método de iteración de punto fijo para resolver el problema.

```
1  void get_RHS_flow(int nt, int nv, const VectorXi& subdomain,const MatrixXi& elements,const
       MatrixXd& vertices,int nbf,const VectorXi& boundary_subdomain,const MatrixXi& boundary,
       std::vector<double>& RHS) {
2      VectorXi Dnodes(nv);
3      for (int i=0; i<nv; i++) {
4          Dnodes(i)=i;
5      }
6      for (int i=0; i<nbf; i++) {
7          if (get_Dirichlet(boundary_subdomain(i))==1) {
8              for (int j=0; j<2; j++) {
9                      Dnodes(boundary(i,j))=-boundary_subdomain(i);
10             }
11         }
12     }
13     for (int k = 0; k < nt; k++) {
14         double area = get_area(vertices.row(elements(k, 0)), vertices.row(elements(k, 1)),
               vertices.row(elements(k, 2)));
15         Matrix<double, 3, 2> normal = get_normal(vertices.row(elements(k, 0)), vertices.row(
               elements(k, 1)), vertices.row(elements(k, 2)));
16         Vector3d fmom1 = Vector3d::Zero();
17         Vector3d fmom2 = Vector3d::Zero();
18         for (int j = 0; j < NTRIQP; j++) {
19             Vector2d X = triquad[j][0] * vertices.row(elements(k,0))
20                        + triquad[j][1] * vertices.row(elements(k,1))
21                        + triquad[j][2] * vertices.row(elements(k,2));
22             double w_f1 = triweight[j] * get_f1(X,subdomain(k));
23             double w_f2 = triweight[j] * get_f2(X,subdomain(k));
24             for (int local_node = 0; local_node < 3; local_node++) {
25                 fmom1(local_node) += w_f1 * triquad[j][local_node];
26                 fmom2(local_node) += w_f2 * triquad[j][local_node];
27             }
28         }
29         for (int local_node = 0; local_node < 3; local_node++) {
30             RHS[elements(k,local_node)] += area * fmom1(local_node);
31             RHS[elements(k,local_node) + nv] += area * fmom2(local_node);
32         }
33     }
34     for (int i = 0; i < nbf; i++) {
35         if (get_Dirichlet(boundary_subdomain(i)) == 0) {
36             double length_F = get_hf(vertices.row(boundary(i,0)), vertices.row(boundary(i,1))
                   );
37             Vector2d du1_n_mom = Vector2d::Zero();
38             Vector2d du2_n_mom = Vector2d::Zero();
39             for (int j = 0; j < NLINQP; j++){
40                 Vector2d X = linquad[j][0] * vertices.row(boundary(i, 0)) + linquad[j][1] *
                       vertices.row(boundary(i, 1));
41                 double w_du1_n = linweight[j] * get_uN1(X, boundary_subdomain(i));
42                 double w_du2_n = linweight[j] * get_uN2(X, boundary_subdomain(i));
43                 for (int local_node = 0; local_node < 2; local_node++) {
44                     du1_n_mom(local_node) += w_du1_n * linquad[j][local_node];
45                     du2_n_mom(local_node) += w_du2_n * linquad[j][local_node];
46                 }
47             }
48             for (int local_node = 0; local_node < 2; local_node++) {
49                 RHS[boundary(i, local_node)] += length_F * du1_n_mom(local_node);
50                 RHS[boundary(i, local_node) + nv] += length_F * du2_n_mom(local_node);
51             }
52         }
53     }
54     for (int i = 0; i < nv; i++) {
55         if (Dnodes(i) < 0) {
56             RHS[i] = get_uD1(vertices.row(i), -Dnodes(i));
```

```
57              RHS[i + nv] = get_uD2(vertices.row(i), -Dnodes(i));
58          }
59      }
60  }
```

LISTING 12. Ensamble de lado derecho para las ecuacioens de Navier Stokes utilizando método de iteración de punto fijo

```
1   void get_stiffnessmatrix_Stokes(int nv, int nt, int nbf,const VectorXi& boundary_subdomain,
        const MatrixXi& boundary, const MatrixXi& elements,const MatrixXd& vertices, const
        MatrixXi& edgeelement, std::vector<double>& RHS,SparseMatrix<double>& K) {
2       int edges1[3]={1,2,0};
3       int edges2[3]={2,0,1};
4       VectorXi Dnodes(nv);
5       for (int i=0; i<nv; i++) {
6           Dnodes(i)=i;
7       }
8       for (int i=0; i<nbf; i++) {
9           if (get_Dirichlet(boundary_subdomain(i))==1) {
10              for (int j=0; j<2; j++) {
11                  Dnodes(boundary(i,j))=-boundary_subdomain(i);
12              }
13          }
14      }
15      for (int k = 0; k < nt; k++) {
16          double area = get_area(vertices.row(elements(k, 0)), vertices.row(elements(k, 1)),
                vertices.row(elements(k, 2)));
17          Matrix<double, 3, 2> normal = get_normal(vertices.row(elements(k,0)), vertices.row(
                elements(k,1)), vertices.row(elements(k,2)));
18          for (int local_node_a = 0; local_node_a < 3; local_node_a++) {
19              double sum_a1 = (1.0 / 2.0) * normal(local_node_a, 0);
20              double sum_a2 = (1.0 / 2.0) * normal(local_node_a, 1);
21              if (Dnodes(elements(k, local_node_a)) >= 0) {
22                K.coeffRef(elements(k, local_node_a), 2 * nv + k) += sum_a1;
23                K.coeffRef(elements(k, local_node_a) + nv, 2 * nv + k) += sum_a2;
24
25                K.coeffRef(2 * nv + k,elements(k, local_node_a)) += sum_a1;
26                K.coeffRef(2 * nv + k,elements(k, local_node_a) + nv) += sum_a2;
27              }
28              else {
29                RHS[2 * nv + k] -= sum_a1 * get_uD1(vertices.row(elements(k, local_node_a)), -
                      Dnodes(elements(k, local_node_a)));
30                RHS[2 * nv + k] -= sum_a2 * get_uD2(vertices.row(elements(k, local_node_a)), -
                      Dnodes(elements(k, local_node_a)));
31              }
32              if (Dnodes(elements(k,local_node_a)) >= 0) {
33                  for (int local_node_b = 0; local_node_b < 3; local_node_b++) {
34                      double val = get_epsilon() * area * (normal.row(local_node_a).dot(normal.
                            row(local_node_b)) / (4.0 * pow(area, 2)));
35                      val += get_kappa_u() * area * (1.0 / 6.0) * ((local_node_a ==
                            local_node_b) * 1 + (local_node_a != local_node_b) * 0.5);
36                      if (Dnodes(elements(k,local_node_b)) >= 0) {
37                          K.coeffRef(elements(k,local_node_a), elements(k,local_node_b)) += val
                                ;
38                          K.coeffRef(elements(k,local_node_a) + nv, elements(k,local_node_b) +
                                nv) += val;
39                      }
40                      else {
41                          RHS[elements(k, local_node_a)] -= get_uD1(vertices.row(elements(k,
                                local_node_b)), -Dnodes(elements(k,local_node_b))) * val;
42                          RHS[elements(k, local_node_a) + nv] -= get_uD2(vertices.row(elements(
                                k, local_node_b)), -Dnodes(elements(k,local_node_b))) * val;
43                      }
44                  }
45              } else {
```

```
46                  K.coeffRef(elements(k, local_node_a), elements(k, local_node_a)) = 1;
47                  K.coeffRef(elements(k, local_node_a) + nv,elements(k,local_node_a) + nv) = 1;
48              }
49          }
50          for (int F = 0; F < 3; F++) {
51              int k_prime = edgeelement(k, F);
52              if (k_prime >= 0) {
53                  double hf = get_hf(vertices.row(elements(k, edges1[F])), vertices.row(
                        elements(k, edges2[F])));
54                  double tau_f = hf / 12.0;
55                  K.coeffRef(2 * nv + k, 2 * nv + k) += -hf * tau_f;
56                  K.coeffRef(2 * nv + k, 2 * nv + k_prime) += + hf * tau_f;
57              }
58          }
59          K.coeffRef(2 * nv + k, 2 * nv + nt) = area;
60          K.coeffRef(2 * nv + nt, 2 * nv + k) = area;
61      }
62  }
63
64  void get_stiffnessmatrix_Oseen_from_Stokes(int nv,int nbf,const VectorXi& boundary_subdomain,
        const MatrixXi& boundary,int nt,const MatrixXi& elements,const MatrixXd& vertices, const
        vector<double>& field, std::vector<double>& RHS,  SparseMatrix<double>& K_Stokes) {
65      VectorXi Dnodes(nv);
66      for (int i=0; i<nv; i++) {
67          Dnodes(i)=i;
68      }
69      for (int i=0; i<nbf; i++) {
70          if (get_Dirichlet(boundary_subdomain(i))==1) {
71              for (int j=0; j<2; j++) {
72                      Dnodes(boundary(i,j))=-boundary_subdomain(i);
73              }
74          }
75      }
76      for (int k = 0; k < nt; k++) {
77          double area = get_area(vertices.row(elements(k, 0)), vertices.row(elements(k, 1)),
                vertices.row(elements(k, 2)));
78          Matrix<double, 3, 2> normal = get_normal(vertices.row(elements(k,0)), vertices.row(
                elements(k,1)), vertices.row(elements(k,2)));
79          Matrix<double, 3, 2> grad_lambda;
80          grad_lambda = (-1.0 / (2.0 * area)) * normal;
81          Matrix<double, 2, 3> field_K;
82          for (int node = 0; node < 3; node++) {
83              field_K.col(node) << field[elements(k, node)], field[elements(k, node) + nv];
84          }
85          for (int local_node_a = 0; local_node_a < 3; local_node_a++) {
86              if (Dnodes(elements(k,local_node_a)) >= 0) {
87                  for (int local_node_b = 0; local_node_b < 3; local_node_b++) {
88                      double val = 0;
89                      val += get_convective_term_phi_phi_K(area, local_node_a, local_node_b,
                            field_K, grad_lambda);
90                      if (Dnodes(elements(k,local_node_b)) >= 0) {
91                          K_Stokes.coeffRef(elements(k,local_node_a), elements(k,local_node_b))
                                += val;
92                          K_Stokes.coeffRef(elements(k,local_node_a) + nv, elements(k,
                                local_node_b) + nv) += val;
93                      } else {
94                          RHS[elements(k, local_node_a)] -= get_uD1(vertices.row(elements(k,
                                local_node_b)), -Dnodes(elements(k,local_node_b))) * val;
95                          RHS[elements(k, local_node_a) + nv] -= get_uD2(vertices.row(elements(
                                k, local_node_b)), -Dnodes(elements(k,local_node_b))) * val;
96                      }
97                  }
98              } else {
99                  K_Stokes.coeffRef(elements(k, local_node_a), elements(k, local_node_a)) = 1;
100                 K_Stokes.coeffRef(elements(k, local_node_a) + nv,elements(k,local_node_a) +
                        nv) = 1;
101             }
102         }
103     }
```

```
104 }
```

LISTING 13. Ensamble de matriz para las ecuaciones de Navier Stokes utilizando método de iteración de punto fijo

```
1  void Fixed_point_iteration(int nv, int nt,int nbf, const VectorXi& subdomain,const VectorXi&
       boundary_subdomain,const MatrixXi& boundary,const MatrixXi& elements,const MatrixXd&
       vertices, SparseMatrix<double> K_Stokes, vector<double>& previous_flow,  vector<double>
       RHS_flow, double& norm) {
2      get_stiffnessmatrix_Oseen_from_Stokes(nv, nbf, boundary_subdomain, boundary, nt, elements
           , vertices, previous_flow, RHS_flow, K_Stokes);
3      MUMPS(K_Stokes, RHS_flow);
4      norm = get_step_norm(previous_flow,  RHS_flow);
5      double omega = 1;
6      for (size_t i = 0; i < RHS_flow.size(); i++) {
7          previous_flow[i] = omega * RHS_flow[i] + (1.0 - omega) * previous_flow[i];
8      }
9  }
10
11 void Fixed_point_resolution(int nt, int nv,int nbf, const VectorXi& subdomain,const VectorXi&
       boundary_subdomain,const MatrixXi& boundary,const MatrixXi& elements,const MatrixXd&
       vertices, const MatrixXi& edgeelement, double tol, int& fixed_point_iterations, vector<
       double>& sol_flow) {
12     vector<double> RHS_flow(get_ss_stokes(nv, nt));
13     get_RHS_flow(nt, nv, subdomain, elements, vertices, nbf, boundary_subdomain, boundary,
           RHS_flow);
14     SparseMatrix<double> K_Stokes(get_ss_stokes(nv, nt), get_ss_stokes(nv, nt));
15     VectorXi nnzpc_flow(get_ss_stokes(nv, nt));
16     get_nnzpc_flow(nv, nt, elements, nbf, boundary_subdomain, boundary, nnzpc_flow);
17     K_Stokes.reserve(nnzpc_flow);
18     get_stiffnessmatrix_Stokes(nv, nt, nbf, boundary_subdomain, boundary, elements, vertices,
           edgeelement, RHS_flow, K_Stokes);
19
20     vector<double> initial_flow = get_initial_flow(nv, nt, nbf, boundary_subdomain, boundary,
           elements, vertices, K_Stokes, RHS_flow);
21     double step_norm = tol;
22     fixed_point_iterations = 0;
23     while (step_norm >= tol && fixed_point_iterations <= get_max_iterations()) {
24         Fixed_point_iteration(nv, nt, nbf, subdomain, boundary_subdomain, boundary, elements,
               vertices, K_Stokes, initial_flow, RHS_flow, step_norm);
25         ++fixed_point_iterations;
26     }
27     sol_flow = initial_flow;
28 }
```

LISTING 14. Código de iteración de punto fijo para las ecuaciones de Navier Stokes

Adicionalmente, el código relativo al Método de Newton es el siguiente:

```
1  void get_RHS_Newton(int nt, int nv, const VectorXi& subdomain,const MatrixXi& elements,const
       MatrixXd& vertices,int nbf,const VectorXi& boundary_subdomain,const MatrixXi& boundary,
       const MatrixXi& edgeelement, std::vector<double>& RHS, const std::vector<double>&
       previous_flow) {
2      VectorXi Dnodes(nv);
3      for (int i=0; i<nv; i++) {
4          Dnodes(i)=i;
5      }
6      for (int i=0; i<nbf; i++) {
7          if (get_Dirichlet(boundary_subdomain(i))==1) {
8              for (int j=0; j<2; j++) {
9                      Dnodes(boundary(i,j))=-boundary_subdomain(i);
```

```
10              }
11          }
12      }
13      for (int k = 0; k < nt; k++) {
14          double area = get_area(vertices.row(elements(k, 0)), vertices.row(elements(k, 1)),
                vertices.row(elements(k, 2)));
15          Matrix<double, 3, 2> normal = get_normal(vertices.row(elements(k, 0)), vertices.row(
                elements(k, 1)), vertices.row(elements(k, 2)));
16
17          Matrix<double, 3, 2> base_functions_gradient = normal * (-1.0 / (2.0 * area));
18
19          Matrix<double, 2, 3> velocidad_K;
20          velocidad_K << previous_flow[elements(k, 0)], previous_flow[elements(k, 1)],
                previous_flow[elements(k, 2)],
21                         previous_flow[elements(k, 0) + nv], previous_flow[elements(k, 1) + nv
                               ], previous_flow[elements(k, 2) + nv];
22          Matrix2d grad_velocidad_K = velocidad_K * base_functions_gradient;
23
24
25          Vector3d fmom1 = Vector3d::Zero();
26          Vector3d fmom2 = Vector3d::Zero();
27          for (int j = 0; j < NTRIQP; j++) {
28              Vector2d X = triquad[j][0] * vertices.row(elements(k,0))
29                         + triquad[j][1] * vertices.row(elements(k,1))
30                         + triquad[j][2] * vertices.row(elements(k,2));
31
32              Vector3d lambda_i_X(triquad[j][0], triquad[j][1], triquad[j][2]);
33              double w_f1 = triweight[j] * get_f1(X,subdomain(k));
34              double w_f2 = triweight[j] * get_f2(X,subdomain(k));
35              Vector2d convectivo = triweight[j] * (grad_velocidad_K * velocidad_K * lambda_i_X
                    );
36              for (int local_node = 0; local_node < 3; local_node++) {
37                  fmom1(local_node) += (w_f1 - convectivo(0)) * triquad[j][local_node] -
                        triweight[j] * (get_epsilon() * grad_velocidad_K.row(0).dot(
                        base_functions_gradient.row(local_node)) - previous_flow[2 * nv + k] *
                        base_functions_gradient(local_node, 0));
38                  fmom2(local_node) += (w_f2 - convectivo(1)) * triquad[j][local_node] -
                        triweight[j] * (get_epsilon() * grad_velocidad_K.row(1).dot(
                        base_functions_gradient.row(local_node)) - previous_flow[2 * nv + k] *
                        base_functions_gradient(local_node, 1));
39              }
40          }
41          for (int local_node = 0; local_node < 3; local_node++) {
42              RHS[elements(k,local_node)] += area * fmom1(local_node);
43              RHS[elements(k,local_node) + nv] += area * fmom2(local_node);
44          }
45          for (size_t edges = 0; edges < 3; edges++) {
46              if (edgeelement(k, edges) >= 0) {
47                  double k_prime = edgeelement(k, edges);
48                  Vector2d X0 = vertices.row(elements(k, (edges + 1) % 3 ));
49                  Vector2d X1 = vertices.row(elements(k, (edges + 2) % 3 ));
50                  double hf = get_hf(X0, X1);
51                  RHS[2 * nv + k] += hf * (hf / 12.0) * (previous_flow[2 * nv + k] -
                        previous_flow[2 * nv + k_prime]);
52              }
53          }
54          RHS[2 * nv + k] += area * (grad_velocidad_K(0, 0) + grad_velocidad_K(1, 1));
55      }
56      for (int i = 0; i < nv; i++) {
57          if (Dnodes(i) < 0) {
58              RHS[i] = 0;
59              RHS[i + nv] = 0;
60          }
61      }
62  }
```

LISTING 15. Código del método de Newton para las ecuaciones Navier Stokes utilizando método de Newton

```
1  void get_stiffnessmatrix_Newton_from_Stokes(int nv,int nbf,const VectorXi& boundary_subdomain
       ,const MatrixXi& boundary,int nt,const MatrixXi& elements,const MatrixXd& vertices, const
        vector<double>& field, SparseMatrix<double>& K_Stokes) {
2      VectorXi Dnodes(nv);
3      for (int i=0; i<nv; i++) {
4          Dnodes(i)=i;
5      }
6      for (int i=0; i<nbf; i++) {
7          if (get_Dirichlet(boundary_subdomain(i))==1) {
8              for (int j=0; j<2; j++) {
9                      Dnodes(boundary(i,j))=-boundary_subdomain(i);
10             }
11         }
12     }
13     for (int k = 0; k < nt; k++) {
14         double area = get_area(vertices.row(elements(k, 0)), vertices.row(elements(k, 1)),
               vertices.row(elements(k, 2)));
15         Matrix<double, 3, 2> normal = get_normal(vertices.row(elements(k,0)), vertices.row(
               elements(k,1)), vertices.row(elements(k,2)));
16         Matrix<double, 3, 2> grad_lambda;
17         grad_lambda = (-1.0 / (2.0 * area)) * normal;
18         Matrix<double, 2, 3> field_K;
19         for (int node = 0; node < 3; node++) {
20             field_K.col(node) << field[elements(k, node)], field[elements(k, node) + nv];
21         }
22         for (int local_node_a = 0; local_node_a < 3; local_node_a++) {
23             if (Dnodes(elements(k,local_node_a)) >= 0) {
24                 for (int local_node_b = 0; local_node_b < 3; local_node_b++) {
25                     double val = get_convective_term_field_phi_phi_K(area, local_node_a,
                           local_node_b, field_K, grad_lambda);
26                     Matrix2d C_ij = get_convective_term_phi_field_phi_K(k, nv, elements, area
                           , local_node_a, local_node_b, field_K, grad_lambda);
27                     if (Dnodes(elements(k,local_node_b)) >= 0) {
28                         K_Stokes.coeffRef(elements(k,local_node_a), elements(k,local_node_b))
                               += val;
29                         K_Stokes.coeffRef(elements(k,local_node_a) + nv, elements(k,
                               local_node_b) + nv) += val;
30                         K_Stokes.coeffRef(elements(k,local_node_a), elements(k,local_node_b))
                                +=  C_ij(0, 0);
31                         K_Stokes.coeffRef(elements(k,local_node_a), elements(k,local_node_b)
                               + nv) += C_ij(0, 1);
32                         K_Stokes.coeffRef(elements(k,local_node_a) + nv, elements(k,
                               local_node_b)) += C_ij(1, 0);
33                         K_Stokes.coeffRef(elements(k,local_node_a) + nv, elements(k,
                               local_node_b) + nv) +=  C_ij(1, 1);
34                     }
35                 }
36             } else {
37                 K_Stokes.coeffRef(elements(k, local_node_a), elements(k, local_node_a)) = 1;
38                 K_Stokes.coeffRef(elements(k, local_node_a) + nv,elements(k,local_node_a) +
                       nv) = 1;
39             }
40         }
41     }
42 }
```

LISTING 16. Código de ensamble de matriz del método de Newton para las ecuaciones Navier Stokes

```
1  void Newton_iteration(int nv, int nt,int nbf, const VectorXi& subdomain,const VectorXi&
       boundary_subdomain,const MatrixXi& boundary,const MatrixXi& elements,const MatrixXd&
       vertices, const MatrixXi& edgeelement, SparseMatrix<double> K_Stokes, vector<double>&
       previous_flow, double& norm) {
2      vector<double> RHS_Newton(get_ss_stokes(nv, nt), 0);
```

```
3       get_RHS_Newton(nt, nv, subdomain, elements, vertices, nbf, boundary_subdomain, boundary,
            edgeelement, RHS_Newton, previous_flow);
4       VectorXi nnzpc_Newton(get_ss_stokes(nv, nt));
5       get_nnzpc_Newton(nv, nt, elements, nbf, boundary_subdomain, boundary, nnzpc_Newton);
6       K_Stokes.reserve(nnzpc_Newton);
7       get_stiffnessmatrix_Newton_from_Stokes(nv, nbf, boundary_subdomain, boundary, nt,
            elements, vertices, previous_flow, K_Stokes);
8
9       MUMPS(K_Stokes, RHS_Newton);
10      for (size_t i = 0; i < RHS_Newton.size() - 1; i++) {
11          previous_flow[i] += RHS_Newton[i];
12      }
13      previous_flow[RHS_Newton.size() - 1] = RHS_Newton[RHS_Newton.size() - 1];
14      norm = get_punctual_norm(RHS_Newton);
15  }
16  vector<double> Newton_resolution(int nt, int nv,int nbf, const VectorXi& subdomain,const
        VectorXi& boundary_subdomain,const MatrixXi& boundary,const MatrixXi& elements,const
        MatrixXd& vertices, const MatrixXi& edgeelement, double tol, int& newton_iterations) {
17      VectorXi nnzpc_flow(get_ss_stokes(nv, nt));
18      get_nnzpc_flow(nv, nt, elements, nbf, boundary_subdomain, boundary, nnzpc_flow);
19      SparseMatrix<double> K_Stokes(get_ss_stokes(nv, nt), get_ss_stokes(nv, nt));
20      K_Stokes.reserve(nnzpc_flow);
21      vector<double> initial_flow = get_initial_flow(nv, nt, nbf, subdomain, boundary_subdomain
            , boundary, elements, vertices, edgeelement, K_Stokes);
22
23      double step_norm = tol;
24      newton_iterations = 0;
25      VectorXi nnzpc_Newton(get_ss_stokes(nv, nt));
26      get_nnzpc_Newton(nv, nt, elements, nbf, boundary_subdomain, boundary, nnzpc_Newton);
27      K_Stokes.reserve(nnzpc_Newton);
28      while (step_norm >= tol && newton_iterations <= get_max_iterations()) {
29          Newton_iteration(nv, nt, nbf, subdomain, boundary_subdomain, boundary, elements,
                vertices, edgeelement, K_Stokes, initial_flow, step_norm);
30          ++newton_iterations;
31      }
32      return initial_flow;
33  }
```

LISTING 17. Código de resolución del método de Newton para las ecuaciones de Navier Stokes

7.4. Ejemplos numéricos. En esta sección, nos ocuparemos de testear nuestro código de elementos finitos, con un ejemplo en donde construiremos una solución análitica y otros ejemplo sin solución analítica. Recordamos que la solución a cualquiera de nuestros problemas, nos entregará el campo de velocidades y de presión asociado al fluido presente en una región Ω.

Ejemplo 1: Consideraremos que el dominio de nuestro problema es el cuadrado unitario $\Omega = (0,1) \times (0,1)$ y se tiene la siguiente solución analítca

$$\mathbf{u}(x,y) = \operatorname{curl}\left(xy(1-x)(1-y)\right)^2,$$
$$p(x,y) = x^2y^2(1-x)(1-y) - \frac{1}{144}.$$

en donde, para el cálculo del lado derecho usamos que $\varepsilon = 1$.

En las Figuras 30, 31 mostramos la solución de elementos finitos estabilizada, en donde se presenta la magnitud del campo vectorial de la velocidad, es decir, $|\mathbf{u}_{\mathscr{T}}|$, como también mostramos las presiones del fluido $p_{\mathscr{T}}$.

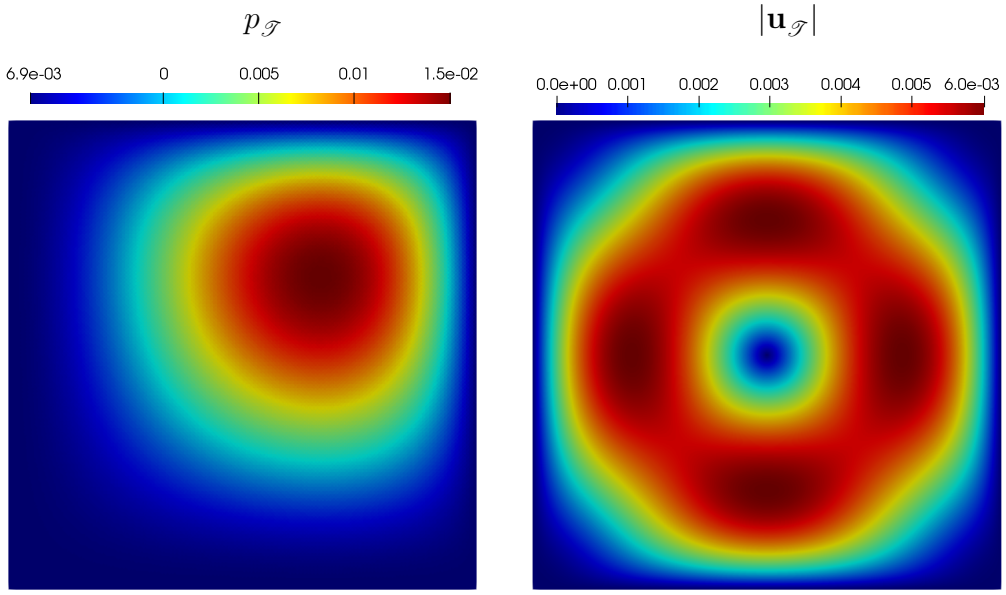

FIGURA 30. Ejemplo 1: Solución del método de elementos finitos estabilizados para Navier Stokes con iteración de punto fijo, con $p_{\mathscr{T}}$ (izquierda) y $\mathbf{u}_{\mathscr{T}}$ (derecha)

Análogamente, para el método de Newton el ejemplo anterior muestra:

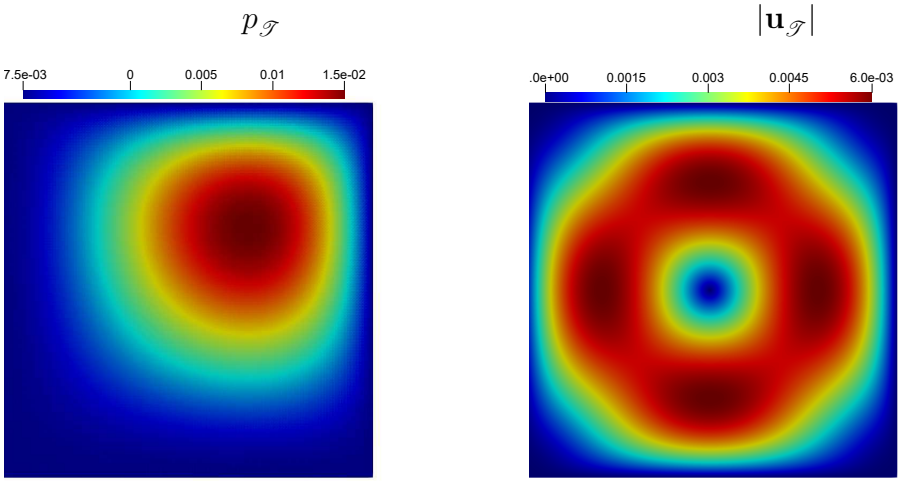

FIGURA 31. Ejemplo 1: Solución del método de elementos finitos estabilizados para Navier Stokes con método de Newton, con $p_{\mathscr{T}}$ (izquierda) y $\mathbf{u}_{\mathscr{T}}$ (derecha)

En la Figura 32 se muestra la tasa óptima de convergencia del método de elementos finitos estabilizado para Navier Stokes con iteración de punto fijo, mientras que en la Figura 33 se muestra la tasa de convergencia del método de elementos finitos estabilizado para Navier Stokes con método de Newton.

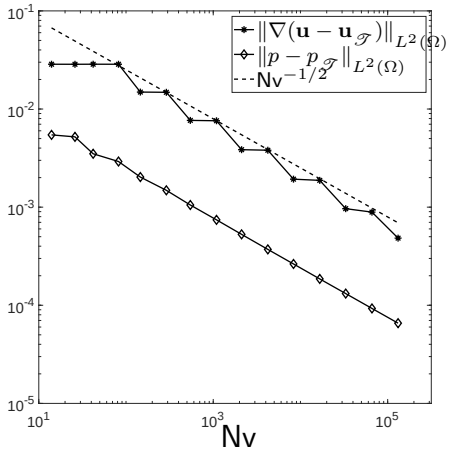

FIGURA 32. Ejemplo 1: Convergencia del método de elementos finitos estabilizado para Navier Stokes con iteración de punto fijo $(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}})$, donde la convergencia lineal corresponde a $h = \frac{1}{\mathrm{Nv}^{1/2}}$.

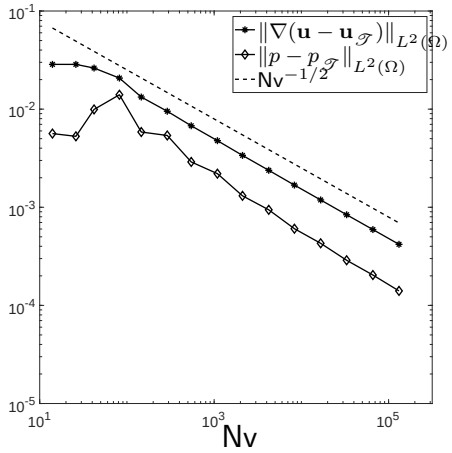

FIGURA 33. Ejemplo 1: Convergencia del método de elementos finitos estabilizado para Navier Stokes con método de Newton $(\mathbf{u}_{\mathscr{T}}, p_{\mathscr{T}})$, donde la convergencia lineal corresponde a $h = \frac{1}{\mathsf{Nv}^{1/2}}$.

Ejemplo 2: Consideraremos el mismo dominio de las secciones anteriores, pero con las condiciones de borde descritas en la Figura 28. Notemos, que como el problema se trata de una dinámica de fluidos, es que en el lado izquierdo de la ciudad consideramos una entrada del tipo parabólica y una salida de dicho fluido con las mismas características. En el resto de la frontera consideramos una condición de borde nula. En la Figura 34, se muestran las soluciones del método de elementos finitos estabilizado para Navier Stokes .

Análogamene, se considera el ejemplo anterior para el esquema de Navier Stokes con método de Newton

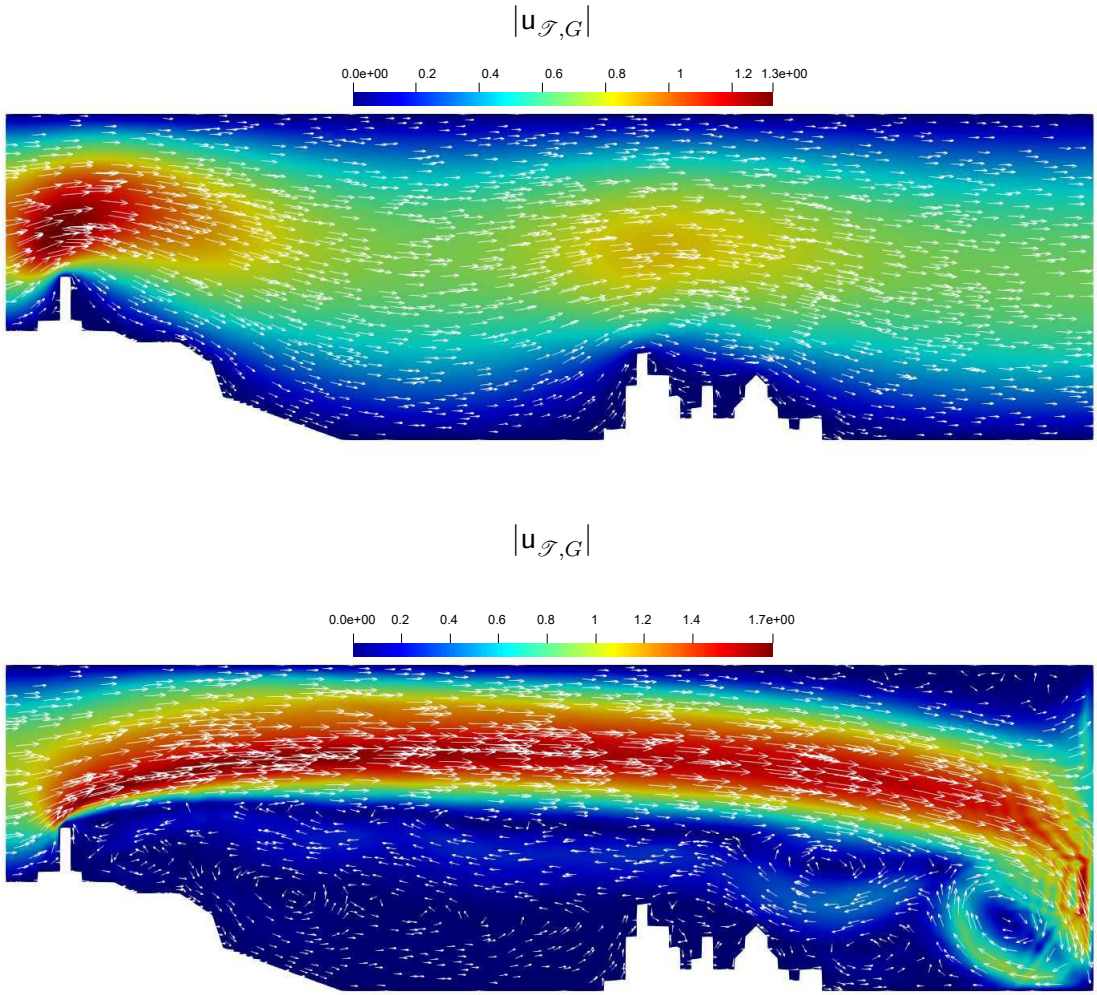

FIGURA 34. Ejemplo 2: solución de elementos finitos estabilizada con iteración de punto fijo para la magnitud del campo de velocidad $|\mathsf{u}_{\mathscr{T}}|$, con $\varepsilon = 1$ (superior) y $\varepsilon = 0{,}001$ (inferior).

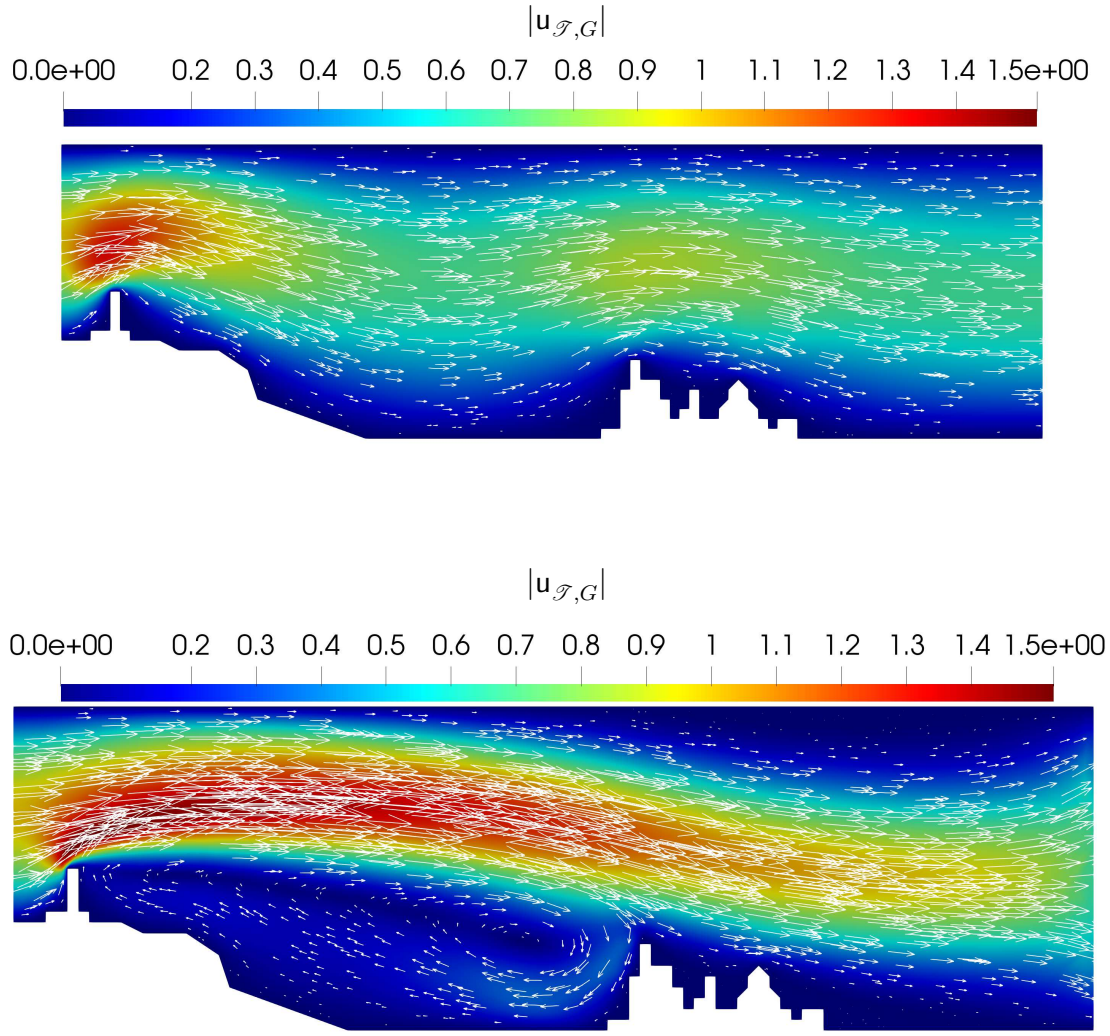

FIGURA 35. Ejemplo 2: elementos finitos estabilizado para Navier Stokes con método de Newton para la magnitud del campo de velocidad $|\mathbf{u}_{\mathscr{T}}|$, con $\varepsilon = 1$ (superior) y $\varepsilon = 0{,}01$ (inferior).

8. Conclusiones

En este trabajo se presentaron una serie de esquemas de elementos finitos los cuales se usaron para aproximar la solución a problemas asociados con contaminación y dinámica de fluidos, para lo cual consideramos tanto problemas lineales, como las ecuaciones de Stokes, y problemas no lineales, como las ecuaciones de Navier–Stokes. Todo lo anterior, en dominios con distintas geometrías.

Dentro de los esquemas considerados, los cuales fueron el esquema clásico de Galerkin en conjunto con un esquema estabilizado, se realizó un estudio de convergencia de los métodos de forma general. Entonces, en base a todo el trabajo anterior, es que podemos concluir que:

- El Método de Elementos Finitos es un sólido método numérico para aproximar las soluciones a ecuaciones diferenciales.
- Dentro de la gran gama de métodos, los esquemas de elementos finitos estabilizados, entregan una mayor robustez en presencia de problemas que contengan capas límites.
- A partir de la estructura del Método de Elementos Finitos presentada se construyó un código estructurado y modular cuya ventaja es la versatilidad, permitiendo que a partir de un código base, las modificaciones necesarias para adaptar la resolución a un nuevo problema sean mínimas, incluso en el caso de problemas más complejos como los nolineales.
- El código construido resuelve de forma satisfactoria los problemas asociados a las Ecuaciones de Navier–Stokes utilizando el Método de Elementos finitos, con convergencia lineal.

En la actualidad, el código es utilizado por académicos e investigadores del Departamento de Matemática en el área de Métodos Numéricos con éxito. Las ventajas que presenta la modularización de este código permiten independizar la etapa de resolución del sistema lineal, lo que conlleva a eventuales mejoras en el rendimiento y robustez de la solución entregada, ya que es posible emplear otros solvers que pueden ofrecer mejor rendimiento que el utilizado en este caso.

Se observó que al construir la implementación uno de los obstáculos más importantes a vencer fue el manejo de los datos anexos al problema, como por ejemplo el manejo de las mallas de datos, puesto que el proceso de refinado ofreció desafíos al momento de asignar nuevos identificadores de los elementos triangulares.

También se concluye que, indudablemente, el núcleo de la solución consiste en el correcto ensamble de la matriz de rigidez, proceso que puede ser llevado a cabo sin dificultades mayores conociendo las expresiones matriciales de las formas bilineales correspondientes al problema.

Como trabajos futuros se proponen:

- Realizar la adaptación del código de Elementos Finitos existente para resolver problemas en 3 dimensiones.
- Evaluar el desempeño del código (manejo de memoria y uso de CPU) con el objeto de buscar optimizaciones que mejoren su rendimiento general.
- Evaluar el desempeño del código realizando pruebas que incluyan distintos solver que pueden traer mejoras al rendimiento en algunos problemas.
- Optimizar los procesos para construir adaptación que sea compatible con mecanismos de paralelismo.
- Ofrecer alternativas de salida de datos que sean compatibles con otros software de visualización.

Referencias

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41 (electronic), 2001.

[2] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Comput.*, 32(2):136–156, 2006.

[3] Kendall Atkinson and Weimin Han. *Theoretical numerical analysis*, volume 39 of *Texts in Applied Mathematics*. Springer, Dordrecht, third edition, 2009. A functional analysis framework.

[4] Daniele Boffi, Franco Brezzi, and Michel Fortin. *Mixed finite element methods and applications*, volume 44 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2013.

[5] Susanne C. Brenner and L. Ridgway Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, third edition, 2008.

[6] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Universitext. Springer, New York, 2011.

[7] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32(1-3):199–259, 1982. FENOMECH '81, Part I (Stuttgart, 1981).

[8] Philippe G. Ciarlet. *The finite element method for elliptic problems*, volume 40 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002. Reprint of the 1978 original [North-Holland, Amsterdam; MR0520174 (58 #25001)].

[9] A. Ern and J.-L. Guermond. *Theory and practice of finite elements*, volume 159 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2004.

[10] L. C. Evans. *Partial Differential Equations: Second Edition*. Graduate Studies in Mathematics.

[11] Leopoldo P. Franca and Sérgio L. Frey. Stabilized finite element methods. II. The incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 99(2-3):209–233, 1992.

[12] Gabriel N Gatica. A simple introduction to the mixed finite element method. *Theory and Applications, Springer-Verlag, Berlin*, 2014.

[13] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations*, volume 5 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1986. Theory and algorithms.

[14] Stig Larsson and Vidar Thomée. *Partial differential equations with numerical methods*, volume 45 of *Texts in Applied Mathematics*. Springer-Verlag, Berlin, 2003.

[15] H.-G. Roos, M. Stynes, and L. Tobiska. *Robust numerical methods for singularly perturbed differential equations*, volume 24 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2008. Convection-diffusion-reaction and flow problems.

[16] Jennifer A Scott, Yifan Hu, and Nicholas IM Gould. An evaluation of sparse direct symmetric solvers: an introduction and preliminary findings. In *International Workshop on Applied Parallel Computing*, pages 818–827. Springer, 2004.

[17] Roger Temam. *Navier-Stokes equations. Theory and numerical analysis*. North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977. Studies in Mathematics and its Applications, Vol. 2.