TRABAJO DE FIN DE GRADO

Grado en Matemáticas

Título: HDG para problemas de ondas: Análisis de dispersión numérica y aplicaciones

Autor: Guillem Belda Ferrín

Director: Sonia Fernández-Méndez

Departamento: Matemàtica Aplicada III

Convocatoria: 2013 /2014

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Trabajo final de grado

HDG para problemas de ondas
Análisis de dispersión numérica y aplicaciones

Guillem Belda Ferrín

Directora: Sonia Fernández-Méndez

Departament de Matemàtica Aplicada III

A Nieves y Paco.

Resumen

Palabras clave: Hybridizable Discontinuous Galerkin, Continuous Galerkin, error de dispersión, número de onda, número de onda numérico, métodos de elementos finitos, discretización

MSC2000: 35, 65N30

En este trabajo se realiza un análisis de dispersión para los métodos Continuous Galerkin (CG) y Hybridizable Discontinuous Galerkin (HDG) al ser aplicados a la ecuación de Helmholtz en 1D. Se calcula la expresión del error de dispersión con aproximaciones numéricas de alto orden para CG y HDG. Luego, se realizan comparativas entre los métodos en función del parámetro de estabilidad utilizando aproximaciones de segundo, tercer y quinto orden. Finalmente, se presentan pruebas numéricas para diferentes grados de aproximación numérica, parámetro de estabilidad y nodos por longitud de onda que confirman los resultados.

Abstract

Keywords: Hybridizable Discontinuous Galerkin, Continuous Galerkin, wave propagation, wavenumber, numerical wavenumber, finite element method, discretization

MSC2000: 35, 65N30

In this work, a dispersion analysis is performed for Continuous Galerkin (CG) and Hybridizable Discontinuous Galerkin (HDG) methods when applied to Helmholtz 1-dimensional equation. A dispersion error is computed for high-order numerical aproximations for CG and HDG. Later on, we compare between these two proposed methods depending on stability parameter using second, third and fifth-order aproximations. Finally, numerical proves for different aproximation degrees, stability parameters and nodes per wavelength are presented confirming the results.

Índice general

Capítulo 1
Introducción

La modelización de muchas aplicaciones de la física e ingeniería, como las ondas acústicas, el electromagnetismo o las vibraciones, se realiza mediante ecuaciones de ondas. La hipótesis de una solución harmónica da como resultado la ecuación de Helmholtz

$$\Delta \mathbf{u} + k^2 \mathbf{u} = 0.$$

La resolución numérica de dicha ecuación para frecuencias altas conlleva dificultades numéricas debido a que hay un desfase entre la onda de la solución numérica y la analítica. Es el llamado *error de dispersión* (Figura 1).

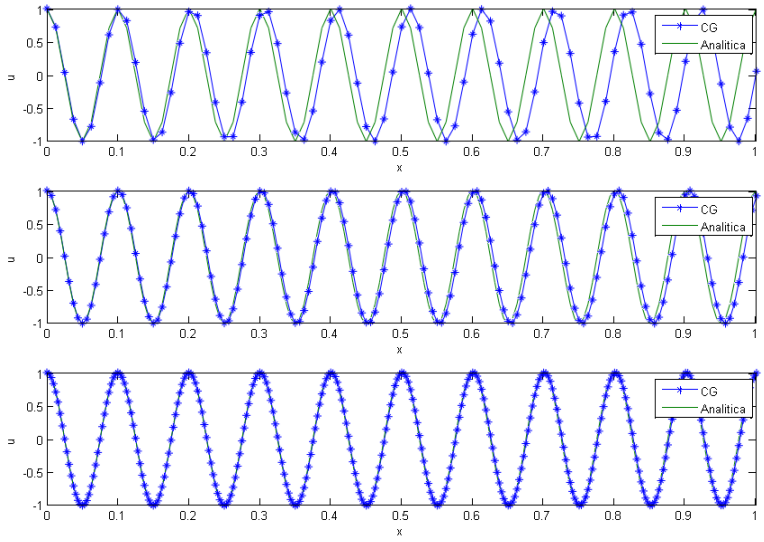

FIGURA 1. En la primera imagen vemos un desfase entre las ondas, siendo la onda de color verde la onda de la solución analítica y la azul la de la solución numérica. En la segunda imagen seguimos viendo desfase, aunque no tan notable como en la primera imagen. En la tercera imagen no se aprecia desfase alguno entre las dos ondas.

El término k de la ecuación de Helmholtz se conoce como *número de onda* de la onda analítica, e indica el número de oscilaciones por longitud. Por lo tanto, nuestra onda numérica también tendrá un número de onda el cual denotaremos $\tilde{k}$, que suele diferir de k. Esto es lo que se conoce como error de dispersión numérica, el cual depende del número de onda numérico $\tilde{k} = kh$; es decir, de la cantidad de nodos por longitud de onda utilizados en la aproximación numérica del problema.

El método más habitual para la resolución de la ecuación de Helmholtz es Elementos Finitos Continuos (CH) con aproximación lineal a trozos. En la Figura 1 anterior se ha utilizado CG con aproximación lineal con 80, 160 y 320 elementos; es decir, con 8, 16 y 32 nodos por longitud de onda respectivamente. Como se ve, a mayor número de nodos mayor precisión se obtiene y menor es el error de dispersión. Giorgio Giorgiani concluye que la utilización de una aproximación numérica de alto orden, $p > 2$, es más eficiente que el aumento de los nodos utilizados por longitud de onda para problemas de ondas mediante CG (Ver [**1**]).

Tal y como se expone en [**1**], la reciente aparición de Hybridizable Discontinuous Galerkin (HDG) para problemas de ondas abre una nueva ventana de los Discontinuous Galerkin (DG) frente los CG. Los DG son métodos que, para alto orden de aproximación numérica, son estables y tienen superconvergencia, pero pierden terreno frente los CG debido a su alto coste computacional. En cambio, HDG es competitivo frente CG para un alto orden de aproximación numérica, ya que tiene un coste computacional y precisión parecidos a CG, aunque HDG es computacionalmente más eficiente que CG en la misma discretización. El hecho de que nadie haya calculado analíticamente el error de dispersión de la ecuación de Helmholtz con HDG y los resultados obtenidos en [**1**] son una motivación y un reto para llevar a cabo este trabajo.

En este trabajo se ha realizado una comparación del error de dispersión en la ecuación de Helmholtz entre CG y HDG. Esto se ha llevado a cabo mediante el cálculo analítico del error de dispersión para CG y HDG, obteniendo gráficas del error entre los métodos para diversos grados en la aproximación numérica. También se ha implementado la resolución numérica unidimensional de CG y HDG para el siguiente caso práctico

$$\begin{cases} u''(x) + k^2 u(x) = 0 \quad \text{con } x \in (0,1) \\ u(0) = 1 \\ u'(0) - iku(1) = 0 \end{cases} \tag{1}$$

cuya solución analítica es $u(x) = e^{ikx}$. Esto ha permitido obtener gráficas de los dos métodos para poder observar el error de dispersión entre CG y HDG en función de la cantidad de nodos por longitud de onda.

En la primera sección de este capítulo se explicará mediante un ejemplo como se obtiene la expresión analítica del error de dispersión para CG. En el Capítulo 2 explicaremos la discretización de la ecuación de Helmholtz con CG, y la posterior implementación de la *condensación estática*. Esto nos permitirá reducir los grados de libertad del sistema y dar una expresión analítica del error de dispersión para aproximaciones de alto orden en CG. En el Capítulo 3 se explicará la discretización y notación del método HDG para la ecuación de Helmholtz, detallando su problema local y global. También se realizará el caso unidimensional con aproximaciones de alto orden, lo que nos permitirá obtener la expresión analítica del error de dispersión para HDG. Por último, en el Capítulo 4 realizaremos comprobaciones numéricas mediante las expresiones analíticas y las gráficas del error para los métodos estudiados.

1. Discretización y notación. Cálculo del error de dispersión.

Sea $\Omega \subset \mathbb{R}^n$ un dominio acotado (y $\partial\Omega$ su frontera) sobre el cual definimos la ecuación de Helmholtz

$$\begin{cases} \Delta \mathbf{u} + k^2 \mathbf{u} = 0 & \text{en } \Omega \\ \mathbf{u} = \mathbf{u_D} & \text{en } \partial\Omega \end{cases} \tag{2}$$

donde $\mathbf{u}$ representa la amplitud de la onda y $k > 0$ es el número de onda. Consideraremos a partir de ahora que $\Omega = [a, b] \subset \mathbb{R}$, de manera que $\mathbf{u}(\mathbf{x})$ será una función escalar $u(x)$. Consideremos una discretización equiespaciada de Ω en $N + 1$ elementos de la forma $\Omega_e = [x_e, x_{e+1}]$, $e = 0, \ldots, N$, con $x_0 = a$, $x_{N+1} = b$ y $h = (b - a)/N$ (Figura 2).

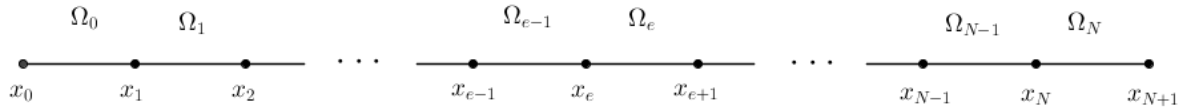

FIGURA 2. Discretización de $\Omega = [a, b]$ en $N + 1$ elementos equiespaciados.

Sea u^h una solución numérica de la forma $u^h(x) = e^{i\tilde{k}x}$; es decir, una solución para la cual los valores nodales cumplen [1]

$$\begin{array}{lll} u_e = u^h(x_e) & = & e^{i\tilde{k}x_e}, \\ u_{e+1} = u^h(x_{e+1}) & = & e^{i\tilde{k}h} u_e, \\ u_{e-1} = u^h(x_{e-1}) & = & e^{-i\tilde{k}h} u_e, \end{array} \tag{3}$$

donde $\tilde{k}$ es el número de onda numérico. Asumiendo que la ecuación discreta es de la forma

$$a u_{e-1} + b u_e + a u_{e+1} = 0$$

y substituyendo (3), uno llega a

$$\tilde{k}h = \arccos -\frac{b}{2a} \tag{4}$$

donde los coeficientes a y b dependen del método numérico considerado. Haciendo Taylor del término de la derecha llegamos a que el error de dispersión en cada elemento es de la forma

[1] $x_e = a + eh$ para $e = 0 \ldots, N$

$$\frac{\tilde{k}-k}{k} = C_1(kh)^2 + C_2(kh)^4 + \cdots \tag{5}$$

Siendo $C_1, C_2, \ldots$ constantes reales independientes de k y h.

Capítulo 2
Discretización con CG de la ecuación de Helmholtz

El interés de este capítulo reside en obtener la expresión analítica del error de dispersión en los elementos interiores de nuestra discretización utilizando CG por tal de poder realizar un análisis de dispersión posteriormente.

1. Discretización con CG. Error de dispersión.

Como acabamos de ver, el error de dispersión de un elemento Ω_e de la discretización se calcula mediante los valores nodales de los extremos en cada elemento Ω_e. En el error de dispersión previamente calculado hemos utilizado una aproximación lineal de la solución $u(x)$ en cada elemento, de manera que la interpolación no generaba nodos interiores.

Si interpolamos con un polinomio de grado p en Ω_e generaremos $p-1$ nodos interiores, de manera que tendremos $p+1$ nodos en Ω_e siendo x_e y x_{e+1} los nodos de los extremos de Ω_e. Por lo tanto, nos interesa expresar de forma explícita los $p-1$ nodos internos de cada elemento en función de sus nodos extremos. Este proceso se conoce como *condensación estática* y reduce considerablemente el número global de incógnitas en cada elemento Ω_e.

Sean $N_i \in \mathcal{P}^p(\Omega_e)$ los polinomios interpoladores [1] de grado p en Ω_e tales que toman el valor 0 en todos los nodos excepto en el i-ésimo nodo donde valen 1, cumpliéndose para todo $i = 1, \ldots, p+1$. Para cada elemento tomamos la función de peso $v_e = N_i$, $i = 1, \ldots, p+1$, siendo N_i la i-ésima función de forma del elemento Ω_e. Expresando la función u^h en cada elemento Ω_e como su aproximación por funciones de peso elementales tenemos que

$$u_e^h = \sum_{i=1}^{p+1} u_i^e N_i(x) \in \mathcal{V}^h(\Omega_e) \tag{6}$$

[1] $N_i \in \mathcal{H}^1(\Omega_e)$.

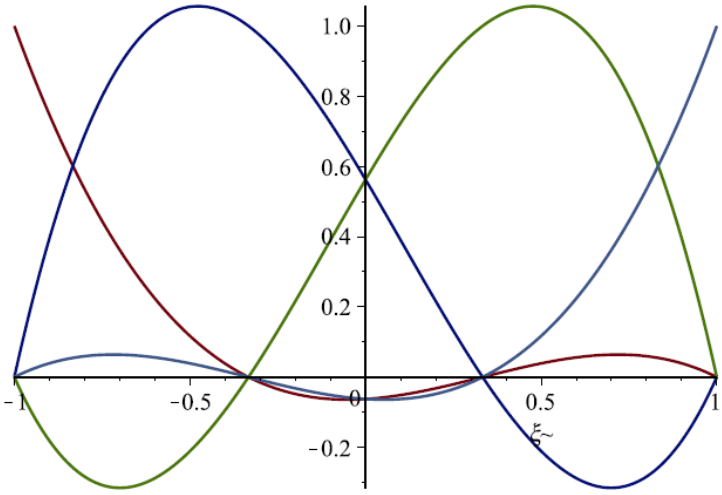

FIGURA 1. Interpolación del intervalo $[-1,1]$ con polinomios de grado $p=3$. Podemos observar que tenemos 2 nodos interiores.

donde u_i^e, con $i=1,\ldots,p+1$, son las incógnitas asociadas a la función u^h en cada elemento Ω_e; es decir, el valor de los $p+1$ nodos del elemento Ω_e. Dado que la función u_e^h está definida sobre los extremos de Ω_e, denotaremos $u_e^h(x_{e+1}) = u_{p+1}^e$ y $u_e^h(x_e) = u_1^e$. Si planteamos la formulación débil de la ecuación tenemos que

$$\int_{\Omega_e} v_e((u_e^h)'' + k^2 u_e^h)dx = 0 \tag{7}$$

Desarrollando (7) y teniendo en cuenta que $v_e = N_i$ para $i=1,\ldots,p+1$, llegamos a

$$\begin{aligned} &\int_{x_e}^{x_{e+1}} N_i \left(\sum_{j=1}^{p+1} u_i^e N_j''\right) dx + \int_{x_e}^{x_{e+1}} N_i \left(\sum_{j=1}^{p+1} k^2 u_i^e N_j\right) dx = \\ &\sum_{j=1}^{p+1} u_i^e \left(\int_{x_e}^{x_{e+1}} N_i' N_j' dx\right) + k^2 \sum_{j=1}^{p+1} u_i^e \left(\int_{x_e}^{x_{e+1}} N_i N_j dx\right) = 0 \end{aligned} \tag{8}$$

Si tomamos las matrices $\mathbf{K^e} = [k_{ij}]$ y $\mathbf{M^e} = [m_{ij}]$ definidas como

$$k_{ij} = \int_{x_e}^{x_{e+1}} N_i' N_j' dx, \quad m_{ij} = \int_{x_e}^{x_{e+1}} N_i N_j dx \quad \text{con } 1 \le i,j \le p+1,$$

las cuales son simétricas, obtenemos el sistema

$$\mathbf{A^e U^e} = 0, \tag{9}$$

con $\mathbf{A^e} = \mathbf{K^e} + k^2\mathbf{M^e}$ y $\mathbf{U^e} = [u_1^e, \ldots, u_{p+1}^e]^T$. Para poder expresar explícitamente los nodos interiores en función de los nodos extremos, consideremos la partición siguiente de la matriz $\mathbf{A^e}$:

$$\mathbf{A^e U^e} = \left[\begin{array}{c|ccc|c} a_{1,1} & a_{1,2} & \cdots & a_{1,p} & a_{1,p+1} \\ \hline a_{1,1} & a_{1,1} & \cdots & a_{1,p} & a_{1,p+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{p,1} & a_{p,1} & \cdots & a_{p,p} & a_{p,p+1} \\ \hline a_{p+1,1} & a_{p+1,1} & \cdots & a_{p+1,p} & a_{p+1,p+1} \end{array}\right] \left[\begin{array}{c} u_1^e \\ u_2^e \\ \vdots \\ u_p^e \\ u_{p+1}^e \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{array}\right]$$

De esta manera podemos aislar los nodos $u_2^e, \ldots, u_p^e$ en función de los nodos extremos u_1^e y u_{p+1}^e a través de un sistema de la forma

$$\widetilde{\mathbf{A}}^{\mathbf{e}}\widetilde{\mathbf{U}}^{\mathbf{e}} = -\mathbf{c_0}u_1^e - \mathbf{c_{p+1}}u_{p+1}^e, \tag{10}$$

donde

$$\widetilde{\mathbf{A}}^{\mathbf{e}} = \left[\begin{array}{ccc} a_{2,2} & \cdots & a_{2,p} \\ \vdots & \ddots & \vdots \\ a_{p,2} & \cdots & a_{p,p} \end{array}\right], \widetilde{\mathbf{U}}^{\mathbf{e}} = \left[\begin{array}{c} u_2^e \\ \vdots \\ u_p^e \end{array}\right], \mathbf{c_0} = \left[\begin{array}{c} a_{2,1} \\ \vdots \\ a_{p,1} \end{array}\right], \mathbf{c_{p+1}} = \left[\begin{array}{c} a_{2,p+1} \\ \vdots \\ a_{p,p+1} \end{array}\right].$$

De esta manera tenemos que $\widetilde{\mathbf{U}}^{\mathbf{e}} = [\widetilde{\mathbf{A}}^{\mathbf{e}}]^{-1}[-\mathbf{c_0}u_1^e - \mathbf{c_{p+1}}\mathbf{u_{p+1}}^e]$. Substituyendo en nuestro vector de valores nodales, utilizando (3) y que $u_1^e = u_{p+1}^{e-1} = u_e$ llegamos a

$$\mathbf{U^{e-1}} = \left[\begin{array}{c} u_{e-1} \\ \widetilde{\mathbf{U}}^{\mathbf{e-1}} \\ u_e \end{array}\right], \mathbf{U^e} = \left[\begin{array}{c} u_e \\ \widetilde{\mathbf{U}}^{\mathbf{e}} \\ u_{e+1} \end{array}\right].$$

Como los valores del nodo u_e en los elementos Ω_{e-1} y Ω_e vienen dados por la última y primera ecuación del sistema $\mathbf{A^e U^e} = 0$, tenemos que

$$\sum_{i=1}^{p+1} a_{p+1,i}\, u_i^{e-1} + a_{1,i}\, u_i^e = 0,$$

llegando a una ecuación discreta de la forma

$$au_{e-1} + bu_e + au_{e+1} = 0,$$

de la cual deducimos la fórmula del error de dispersión (5) tal y como se ha hecho en la primera sección de este capítulo. Estos son algunos ejemplos para diferentes grados de interpolación del error de dispersión en los elementos interiores Ω_e:

$$\begin{cases} p=2 \Rightarrow & \dfrac{\tilde{k}-k}{k} = -\dfrac{1}{1440}(kh)^4 + \dfrac{11}{302400}(kh)^6 - \dfrac{1}{20736000}(kh)^8 + O((kh)^{10}), \\ p=3 \Rightarrow & \dfrac{\tilde{k}-k}{k} = -\dfrac{1}{201600}(kh)^6 + \dfrac{11}{63504000}(kh)^8 - \dfrac{67}{32598720000}(kh)^{10} + O((kh)^{12}), \\ p=5 \Rightarrow & \dfrac{\tilde{k}-k}{k} = -\dfrac{1}{20118067200}(kh)^{10} + \dfrac{1}{924712588800}(kh)^{12} - \dfrac{571}{552096094556160000}(kh)^{14} + O((kh)^{16}). \end{cases}$$

Por lo tanto, para grado $p > 1$

$$\frac{\tilde{k}-k}{k} = C_1(kh)^{2p} + O((kh)^{2p+2}). \tag{11}$$

Capítulo 3
Discretización con HDG de la ecuación de Helmholtz

En los métodos de elementos finitos también encontramos los Discontinuous Galerkin (DG), métodos de elementos finitos discontinuos; es decir, consideran cada elemento de la discretización de manera independiente y resuelven la formulación débil del problema elemento a elemento. Estos métodos son estables y permiten conseguir un alto orden de precisión, y transmiten su información de elemento a elemento mediante el flujo numérico. El inconveniente de los DG frente a los CG reside en que tienen un número elevado de grados de libertad ya que genera nuevos nodos interiores en los elementos para aproximaciones de alto orden.

Por otra parte, la reciente aparición de la hibridización permite reducir los grados de libertad en las fronteras de los elementos (funciones de traza) en los DG, de una manera similar a la condensación estática en los CG. Estos métodos se conocen como Hybridizable Discontinuous Galerkin (HDG). Además, también incrementan la convergencia en norma $\mathcal{L}^2$ del gradiente de la solución, siendo esta convergencia del mismo orden que la solución.

En este capítulo explicaremos cómo se resuelve la ecuación de Helmholtz con HDG con condiciones de Dirichlet constantes y explicaremos detalladamente el caso unidimensional, mediante el cual encontraremos la expresión analítica de los errores de dispersión para HDG con diferentes grados de aproximación numérica en nuestra solución.

1. Discretización y notación

Sea $\Omega \subset \mathbb{R}^n$ un abierto acotado con frontera $\partial\Omega$. Consideremos una partición de Ω en n_{el} elementos disjuntos Ω_e con fronteras $\partial\Omega_e$. Las siguientes definiciones y notaciones serán usadas para la construcción del dominio y frontera utilizados en HDG.

$$\overline{\Omega} = \bigcup_{e=1}^{n_{el}} \overline{\Omega}_e,\ \Omega_e \cap \Omega_{e'} = \emptyset \text{ para } e \neq e'$$

Como se ha comentado previamente, el método HDG plantea una formulación débil en cada elemento Ω_e de forma independiente ya que es un método DG. Por tal de reducir el número de grados de libertad

introducimos una nueva variable $\hat{\mathbf{u}}$, la cual es una aproximación de la traza de $\mathbf{u}$ en el conjunto Γ, definido como

$$\Gamma = \bigcup_{e=1}^{n_{el}} \partial\Omega_e$$

y que representa el esqueleto de nuestra malla (Figura 1).

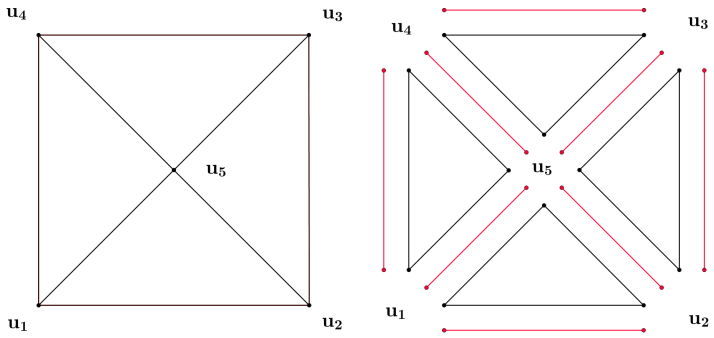

FIGURA 1. Representación de cuatro elementos triangulares que comparten caras interiores en CG y HDG de izquierda a derecha respectivamente. En rojo las caras que generan Γ; es decir, las trazas.

De esta manera podemos plantear un problema local en cada elemento Ω_e, resolviendo la forma débil con las condiciones de contorno asociadas a Ω_e

$$\mathbf{u} = \hat{\mathbf{u}} \text{ en } \partial\Omega e. \tag{12}$$

Nótese que $\hat{\mathbf{u}}$ es unievaluada en cada cara de Γ, de manera que (12) nos asegura la continuidad entre dos nodos.

Definimos el operador *salto* $[\![\cdot]\!]$, el cual actúa en cada cara de Γ utilizando el valor de los elementos de la izquierda y de la derecha (llamémoslos Ω_L y Ω_R) de manera que

$$[\![\circledcirc]\!] = \circledcirc_L + \circledcirc_R$$

implicando el vector normal $\boldsymbol{n}$.

Los espacios de funciones definidas en el interior de los elementos Ω_e y de la función de traza en Γ los definimos, respectivamente, como

$$\mathcal{V}_h = \{v \in \mathcal{H}^1(\Omega) : v|_{\Omega_i} \in \mathcal{P}^k(\Omega_i), \text{ para } i = 1, \ldots, n_{el}\},$$
$$\Lambda_h = \{\bar{\tau} \in \mathcal{L}^2(\Gamma) : \bar{\tau}|_{\Gamma_i} \in \mathcal{P}^k(\Gamma_i), \text{ para } i = 1, \ldots, n_{ca}\},$$

donde $\mathcal{P}^k$ es el espacio de polinomios de grado k, y n representa la dimensión de $\Omega \subset \mathbb{R}^n$ (Figura 2).

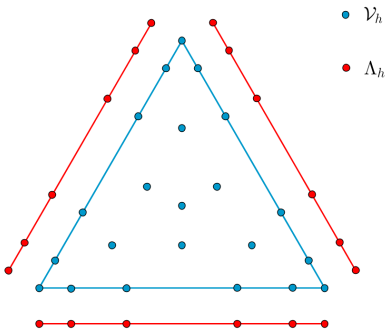

FIGURA 2. Nodos representantes de los espacios $\mathcal{V}_h$ y Λ_h para un elemento triangular de grado $p = 5$.

2. Formulación HDG para la ecuación de Helmholtz

Consideremos la ecuación de Helmholtz sobre un dominio $\Omega \subset \mathbb{R}^n$

$$\begin{cases} \Delta \mathbf{u} + k^2 \mathbf{u} = 0 & \text{en } \Omega \\ \mathbf{u} = \mathbf{u}_D & \text{en } \partial\Omega \end{cases} \tag{13}$$

donde $\mathbf{u} \in \mathcal{H}^1(\Omega)$. En la formulación débil del problema introducimos la variable $\mathbf{q}$ cumpliendo que $k\mathbf{q} = \boldsymbol{\nabla}\mathbf{u}$. De esta manera, transformamos la ecuación (13) en las siguientes dos ecuaciones de primer orden

$$\begin{cases} \boldsymbol{\nabla}\mathbf{u} - k\mathbf{q} = 0 & \text{en } \Omega \\ \boldsymbol{\nabla}\cdot \mathbf{q} + k\mathbf{u} = 0 & \text{en } \Omega \\ \mathbf{u} = \mathbf{u}_D & \text{en } \partial\Omega \end{cases}$$

las cuales se resolverán para cada elemento Ω_e. También impondremos $\mathbf{u} = \hat{\mathbf{u}}$ en Γ, lo cual se entiende como la imposición de las condiciones de contorno en el problema local para cada elemento.

Estas soluciones locales han de relacionarse de alguna manera por tal de obtener una solución global, para eso impondremos la continuidad del flujo normal entre elementos; es decir, $[\![\mathbf{q} \cdot \boldsymbol{n}]\!] = 0$ en las caras interiores y las condiciones de contorno del problema (13) en las caras exteriores.

De esta manera obtendremos un sistema lineal de ecuaciones la solución del cual será la solución de (13) en cada elemento ($\mathbf{u} \in \mathcal{V}_h$), el flujo ($\mathbf{q} \in [\mathcal{V}_h]^n$) y la solución en el esqueleto del dominio ($\hat{\mathbf{u}} \in \Lambda_h$).

3. Problema local

Supongamos $\hat{\mathbf{u}}$ conocida y el problema (13) en el elemento Ω_e.

$$\begin{cases} \boldsymbol{\nabla}\mathbf{u} - k\mathbf{q} = 0 & \text{en } \Omega_e \quad (a) \\ \boldsymbol{\nabla}\cdot\mathbf{q} + k\mathbf{u} = 0 & \text{en } \Omega_e \quad (b) \\ \mathbf{u} = \hat{\mathbf{u}} & \text{en } \partial\Omega_e \end{cases} \tag{14}$$

Consideremos la ecuación (a) y la multiplicamos por una función test $\boldsymbol{\tau}$, luego integramos sobre Ω_e

$$\int_{\Omega_e} \boldsymbol{\tau}\cdot(\boldsymbol{\nabla}\mathbf{u} - k\mathbf{q})\, d\Omega = 0, \quad \forall\boldsymbol{\tau} \in [\mathcal{H}^1(\Omega_e)]^n. \tag{15}$$

Aplicando integración por partes en (15) obtenemos

$$\int_{\Omega_e} \boldsymbol{\nabla}\cdot\boldsymbol{\tau}\,\mathbf{u}\, d\Omega + k\int_{\Omega_e} \boldsymbol{\tau}\cdot\mathbf{q}\, d\Omega = \int_{\partial\Omega_e} \hat{\mathbf{u}}\,\boldsymbol{\tau}\cdot\boldsymbol{n}\, d\Gamma \quad \forall\boldsymbol{\tau} \in [\mathcal{H}^1(\Omega_e)]^n. \tag{16}$$

Para la ecuación (b) procedemos de la misma manera

$$\int_{\Omega_e} \boldsymbol{v}\cdot(\boldsymbol{\nabla}\cdot\mathbf{q} + k\mathbf{u})\, d\Omega = 0, \quad \forall\boldsymbol{v} \in \mathcal{H}^1(\Omega_e). \tag{17}$$

Aplicando integración por partes en (17) obtenemos que

$$\int_{\Omega_e} \boldsymbol{\nabla}\cdot\boldsymbol{v}\mathbf{q}\, d\Omega + k\int_{\Omega_e} \boldsymbol{v}\mathbf{u}\, d\Omega = \int_{\partial\Omega_e} \boldsymbol{v}\mathbf{q}\cdot\boldsymbol{n}\, d\Gamma. \tag{18}$$

Sea $\hat{\mathbf{q}}$ el *flujo numérico* definido como

$$\hat{\mathbf{q}} = \mathbf{q} + \tau(\hat{\mathbf{u}} - \mathbf{u})\boldsymbol{n}, \tag{19}$$

siendo τ un parámetro llamado *parámetro de estabilización*. Imponiendo el flujo numérico (19) en (18) y deshaciendo la integración por partes llegamos a

$$\int_{\Omega_e} \boldsymbol{v}\boldsymbol{\nabla}\cdot\mathbf{q}\, d\Omega + k\int_{\Omega_e} \boldsymbol{v}\mathbf{u}\, d\Omega - \tau\int_{\partial\Omega_e} \boldsymbol{v}\mathbf{u}\cdot\boldsymbol{n}\, d\Gamma = -\tau\int_{\partial\Omega_e} \boldsymbol{v}\hat{\mathbf{u}}\cdot\boldsymbol{n}\, d\Gamma, \forall\boldsymbol{v} \in \mathcal{H}^1(\Omega_e). \tag{20}$$

Entonces, para las ecuaciones (a) y (b) obtenemos dos sistemas de la forma

$$\begin{cases} \mathbf{A_{uu}}\bar{\mathbf{u}}^{\mathbf{e}} & +\mathbf{A_{uq}}\bar{\mathbf{q}}^{\mathbf{e}} & = \mathbf{A}^{\mathbf{e}}_{\mathbf{u\hat{u}}}\hat{\mathbf{u}}_{F_e} \\ \mathbf{A_{qu}}\bar{\mathbf{u}}^{\mathbf{e}} & +\mathbf{A_{qq}}\bar{\mathbf{q}}^{\mathbf{e}} & = \mathbf{A}^{\mathbf{e}}_{\mathbf{q\hat{u}}}\hat{\mathbf{u}}_{F_e} \end{cases}$$

Juntando dichos sistemas llegamos a un único sistema lineal de la forma

$$\begin{bmatrix} \mathbf{A_{uu}} & \mathbf{A_{uq}} \\ \mathbf{A_{qu}} & \mathbf{A_{qq}} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{u}}^{\mathbf{e}} \\ \bar{\mathbf{q}}^{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{\mathbf{e}}_{\mathbf{u\hat{u}}} \\ \mathbf{A}^{\mathbf{e}}_{\mathbf{q\hat{u}}} \end{bmatrix} \hat{\mathbf{u}}_{F_e} \tag{21}$$

donde $\mathbf{A_{uu}}$ y $\mathbf{A_{uq}}$ son las matrices asociadas a la función test $\boldsymbol{v} \in \mathcal{H}^1(\Omega_e)$ de $\mathbf{u}$ y $\mathbf{q}$, $\mathbf{A_{uq}}$ y $\mathbf{A_{qq}}$ las matrices asociadas a la función test $\boldsymbol{\tau} \in [\mathcal{H}^1(\Omega_e)]^n$ de $\mathbf{u}$ y $\mathbf{q}$. También definimos $\bar{\mathbf{u}}^{\mathbf{e}}$, $\bar{\mathbf{q}}^{\mathbf{e}}$, $\hat{\mathbf{u}}_{F_e}$ como los vectores formados por los valores nodales de la aproximación numérica de $\mathbf{u}$, $\mathbf{q}$ y $\hat{\mathbf{u}}$, respectivamente, en el elemento Ω_e.

Resolviendo el sistema (21) tenemos que

$$\begin{bmatrix} \bar{\mathbf{u}}^{\mathbf{e}} \\ \bar{\mathbf{q}}^{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{A_{uu}} & \mathbf{A_{uq}} \\ \mathbf{A_{qu}} & \mathbf{A_{qq}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{A}^{\mathbf{e}}_{\mathbf{u\hat{u}}} \\ \mathbf{A}^{\mathbf{e}}_{\mathbf{q\hat{u}}} \end{bmatrix} \hat{\mathbf{u}}_{F_e}. \tag{22}$$

De esta manera, la solución local para cada elemento solo depende de los términos de $\hat{\mathbf{u}}$, el cual desconocemos.

4. Problema global

Una vez resuelto el problema local para cada elemento Ω_e en función de la incógnita $\hat{\mathbf{u}}$, resolvemos el problema global imponiendo la continuidad del flujo entre elementos; es decir, $[\![\mathbf{q}\cdot\boldsymbol{n}]\!] = 0$ (Figura 3). También imponemos las condiciones de contorno de Dirichlet $\hat{\mathbf{u}} = \mathbf{u}_D$ en $\partial\Omega$. Tomando $\mathbf{q} = \hat{\mathbf{q}}$ tenemos que

$$\sum_e \int_{\partial\Omega_e} \mu\hat{\mathbf{q}} \cdot \boldsymbol{n}\, d\Gamma \tag{23}$$

para toda función de traza μ, función test de $\hat{\mathbf{u}}$.

Sean Ω_L y Ω_R dos elementos que comparten la cara interior F (Figura 3). Si $\mu \neq 0$ en F, entonces nos queda

$$\int_F \mu\hat{\mathbf{q}}_L \cdot \boldsymbol{n_L}\, d\Gamma + \int_F \mu\hat{\mathbf{q}}_R \cdot \boldsymbol{n_R}\, d\Gamma = 0 \tag{24}$$

donde $\boldsymbol{n_L} = \boldsymbol{n_R}$. De (24) llegamos a

$$\int_F \mu(\hat{\mathbf{q}}_L - \hat{\mathbf{q}}_R) \cdot \boldsymbol{n_L}\, d\Gamma = 0, \quad \forall\mu \in \Lambda_h(F). \tag{25}$$

Substituyendo (19) y (25) en (23) llegamos a

$$\int_F \mu[(\mathbf{q}_L - \tau(\hat{\mathbf{u}} - \mathbf{u})\boldsymbol{n_L}) - (\mathbf{q}_R - \tau(\hat{\mathbf{u}} - \mathbf{u})\boldsymbol{n_R})] \cdot \boldsymbol{n_L}\, d\Gamma = 0, \quad \forall\mu \in \Lambda_h(F).$$

Desarrollando esta expresión tenemos que

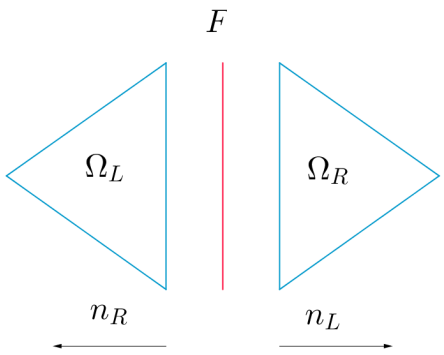

FIGURA 3. Tenemos que $\boldsymbol{n_L} = -\boldsymbol{n_R}$ en la cara interior F.

$$\int_F \mu(\mathbf{q}_L - \mathbf{q}_R)\cdot \boldsymbol{n_L}\, d\Gamma + \int_F \mu\tau(\hat{\mathbf{u}}_F - \mathbf{u}_R)\boldsymbol{n_L}\cdot \boldsymbol{n_L}\, d\Gamma - \int_F \mu\tau(\hat{\mathbf{u}}_F - \mathbf{u}_R)\boldsymbol{n_R}\cdot \boldsymbol{n_L}\, d\Gamma =$$
$$\int_F \mu(\mathbf{q}_L - \mathbf{q}_R)\cdot \boldsymbol{n}\, d\Gamma + \int_F \mu\tau(\hat{\mathbf{u}}_F - \mathbf{u}_R)\, d\Gamma + \int_F \mu\tau(\hat{\mathbf{u}}_F - \mathbf{u}_R)\, d\Gamma =$$
$$\int_F \mu[(\mathbf{q}_L - \mathbf{q}_R)\cdot \boldsymbol{n} + 2\tau\hat{\mathbf{u}}_F - \tau\mathbf{u}_L - \tau\mathbf{u}_R\, d\Gamma =$$
$$\int_F \mu\mathbf{q}_L\cdot \boldsymbol{n}\, d\Gamma - \int_F \mu\mathbf{q}_R\cdot \boldsymbol{n}\, d\Gamma + 2\tau\int_F \mu\hat{\mathbf{u}}_F\, d\Gamma - \tau\int_F \mu\mathbf{u}_L\, d\Gamma - \tau\int_F \mu\mathbf{u}_R\, d\Gamma = 0.$$

A partir de aquí, obtenemos el sistema lineal

$$\mathbf{A}^{\mathbf{L}}_{\hat{\mathbf{u}}\mathbf{q}}\bar{\mathbf{q}}^{\mathbf{L}} + \mathbf{A}^{\mathbf{R}}_{\hat{\mathbf{u}}\mathbf{q}}\bar{\mathbf{q}}^{\mathbf{R}} + \mathbf{A}^{\mathbf{L}}_{\hat{\mathbf{u}}\mathbf{u}}\bar{\mathbf{u}}^{\mathbf{L}} + \mathbf{A}^{\mathbf{R}}_{\hat{\mathbf{u}}\mathbf{u}}\bar{\mathbf{u}}^{\mathbf{R}} + \mathbf{A}_{\hat{\mathbf{u}}\hat{\mathbf{u}}}\hat{\mathbf{u}}_F = 0. \tag{26}$$

Substituyendo la solución del problema local (23) en (26) obtenemos un sistema

$$\mathbf{A}\hat{\mathbf{u}} = \mathbf{h} \tag{27}$$

cuyas incógnitas son los nodos de las caras interiores o exteriores. Una vez calculada la solución del esqueleto Γ; es decir, una vez resuelto el sistema (27), substituimos la solución en (23) y obtenemos los valores de $\mathbf{u}$ y $\mathbf{q}$ en cada elemento.

5. Ecuación de Helmholtz unidimensional

A partir de ahora desarrollaremos el caso unidimensional, de manera que expresaremos nuestra ecuación en función de la variable x, denotando la solución y el gradiente del problema como $\mathbf{u} = u$ y $\mathbf{q} = q$ y la traza $\hat{\mathbf{u}} = \hat{u}$. Impondremos también condiciones de contorno de Dirichlet constantes u_D en la frontera $\partial\Omega$. Sea $\Omega = [a, b] \subset \mathbb{R}$ y la discretización equiespaciada en $N+1$ elementos de la forma $\Omega_e = [x_e, x_{e+1}]$, $e = 0, \ldots, N$, con $x_0 = a$, $x_{N+1} = b$ y $h = (b-a)/N$ (Figura 4).

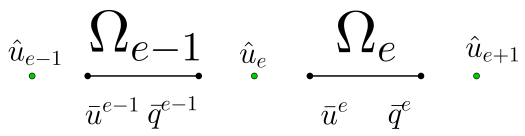

FIGURA 4. Discretización de Ω.

Nuestro problema ahora será

$$\begin{cases} u' - kq = 0 & \text{en } (a,b) \quad (a) \\ q' + ku = 0 & \text{en } (a,b) \quad (b) \\ u(a) = u(b) = u_D \end{cases} \tag{28}$$

La ecuación de salto $[\![\mathbf{q}\cdot\boldsymbol{n}]\!] = 0$ en Γ ahora será

$$[\![qn]\!] = 0 \text{ en } \Gamma \tag{29}$$

En la formulación local del problema (28), donde los elementos Ω_e son de la discretización antes definida, tenemos que la formulación débil para la ecuación (a) es

$$\int_{x_e}^{x_{e+1}} v'u\,dx \,-\, k\int_{x_e}^{x_{e+1}} vq\,dx \,=\, v(x_{e+1})\hat{u}(x_{e+1}) - v(x_e)\hat{u}(x_e), \tag{30}$$

$\forall v \in \mathcal{H}^1([x_e, x_{e+1}])$. Y para la ecuación (b), considerando el flujo numérico como $\hat{q} = q + \tau(\hat{u} - u)$

$$\int_{x_e}^{x_{e+1}} \nu q'\,dx + k\int_{x_e}^{x_{e+1}} \nu u\,dx - \tau[\nu(x_{e+1})u(x_{e+1}) + \nu(x_e)u(x_e)] \,=\, -\tau[\nu(x_{e+1})\hat{u}(x_{e+1}) + \nu(x_e)\hat{u}(x_e)], \tag{31}$$

$\forall \nu \in \mathcal{H}^1([x_e, x_{e+1}])$. Una vez tenemos la solución local de cada elemento, planteamos el problema global. Imponemos los valores nodales con condiciones de Dirichlet de manera que $\hat{u} = u_D$ en Γ. Para las caras interiores tomamos $\hat{q} = q$ y procedemos como en el capítulo anterior

$$\sum_e \int_{\partial\Omega_e} \mu\hat{q}\,d\Gamma \tag{32}$$

para toda función de traza μ, función test de $\hat{u}$, la cual dependerá de si nos encontramos en una cara interior o en la frontera de Dirichlet. Imponiendo la expresión del flujo numérico en (32) llegamos a

$$\sum_e \int_{\partial\Omega_e} \mu[q + \tau(\hat{u} - u)]\,d\Gamma, \tag{33}$$

la ecuación global del problema.

Sean $N_i \in \mathcal{P}^p(\Omega_e)$ los polinomios interpoladores de grado p en Ω_e, obteniendo $p+1$ nodos, tales que toman el valor 0 en todos los nodos excepto en el i-ésimo nodo donde valen 1 cumpliéndose para todo $i = 1, \ldots, p+1$. Para cada elemento tomamos la función de peso $v_e = N_i$, $i = 1, \ldots, p+1$, siendo N_i la i-ésima función de forma del elemento Ω_e.

Expresando las funciones u y q en cada elemento Ω_e por sus aproximaciones por funciones de peso elementales tenemos que

$$u^e = \sum_{i=1}^{p+1} u_i^e N_i(x), \quad q^e = \sum_{i=1}^{p+1} q_i^e N_i(x) \tag{34}$$

donde u_i^e y q_i^e son los valores nodales de las funciones u y q en el elemento Ω_e. Dado que las funciones μ y $\hat{u}$ están definidas sobre los vértices x_e de la discretización, denotaremos su valor explícito en x_e como $\hat{u}_e$. De esta manera podemos reescribir la ecuación (30) introduciendo las ecuaciones de (34) como

$$\sum_{j=1}^{p+1} u_j^e \left(\int_{x_e}^{x_{e+1}} N_i' N_j \, dx \right) + k \sum_{j=1}^{p+1} q_j^e \left(\int_{x_e}^{x_{e+1}} N_i N_j \, dx \right) = N_i(x_{e+1}) \hat{u}_{e+1} - N_i(x_e) \hat{u}_e. \tag{35}$$

Y la ecuación (31) como

$$\sum_{j=1}^{p+1} q_j^e \left(\int_{x_e}^{x_{e+1}} N_i N_j' \, dx \right) + k \sum_{j=1}^{p+1} u_j^e \left(\int_{x_e}^{x_{e+1}} N_i N_j \, dx \right) - \tau [N_i(x_{e+1}) u(x_{e+1}) + N_i(x_e) u(x_e)] = \tag{36}$$

$$= -\tau [N_i(x_{e+1}) \hat{u}_{e+1} + N_i(x_e) \hat{u}_e)].$$

Introduciendo la notación $\delta_{i,j}$ satisfaciendo $\delta_{i,j} = 1$ si $i = j$ y $\delta_{i,j} = 0$ si $i \neq j$, y teniendo en cuenta que $N_i(x_e) = \delta_{i,e}$ por construcción, tenemos que (35) y (36) se reescriben, respectivamente, como

$$\sum_{j=1}^{p+1} u_j^e \left(\int_{x_e}^{x_{e+1}} N_i' N_j \, dx \right) + k \sum_{j=1}^{p+1} q_j^e \left(\int_{x_e}^{x_{e+1}} N_i N_j \, dx \right) = \delta_{i,p+1} \hat{u}_{e+1} - \delta_{i,1} \hat{u}_e. \tag{37}$$

$$\sum_{j=1}^{p+1} q_j^e \left(\int_{x_e}^{x_{e+1}} N_i N_j' \, dx \right) + k \sum_{j=1}^{p+1} u_j^e \left(\int_{x_e}^{x_{e+1}} N_i N_j \, dx \right) - \tau [\delta_{i,p+1} u_{p+1}^e + \delta_{i1} u_1^e] = -\tau [\delta_{i,p+1} \hat{u}_{e+1} + \delta_{i,1} \hat{u}_e]. \tag{38}$$

siendo u_1^e y u_{p+1}^e el primer y último nodo del elemento Ω_e. Definimos $\mathbf{M^e} = [m_{ij}]$, $\mathbf{C^e} = [c_{ij}]$ como

$$m_{ij} = \int_{x_e}^{x_{e+1}} N_i N_j \, dx \,, \quad c_{ij} = \int_{x_e}^{x_{e+1}} N_i' N_j \, dx.$$

Entonces tenemos los siguientes sistemas de ecuaciones

$$\left\{ \begin{array}{lll} \mathbf{A_{uu}}\bar{u}^e & +\mathbf{A_{uq}}\bar{q}^e & = \mathbf{A^e_{u\hat{u}}}\hat{u}^e \\ \mathbf{A_{qu}}\bar{u}^e & +\mathbf{A_{qq}}\bar{q}^e & = \mathbf{A^e_{q\hat{u}}}\hat{u}^e \end{array} \right.$$

donde $\mathbf{A_{uu}} = k\mathbf{M^e} - \tau\mathbf{I_1}$, $\mathbf{A_{uq}} = [\mathbf{C^e}]^T$, $\mathbf{A^e_{u\hat{u}}} = -\tau\mathbf{I_2}$, $\mathbf{A_{qu}} = \mathbf{C^e}$, $\mathbf{A_{qq}} = k\mathbf{M^e}$ y $\mathbf{A^e_{q\hat{u}}} = \mathbf{I_3}$ con

$$\mathbf{I_1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \mathbf{I_2} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{I_3} = \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \bar{u}^e = \begin{bmatrix} u_1^e \\ u_2^2 \\ \vdots \\ u_p^e \\ u_{p+1}^e \end{bmatrix}, \bar{q}^e = \begin{bmatrix} q_1^e \\ q_2^2 \\ \vdots \\ q_p^e \\ q_{p+1}^e \end{bmatrix}, \bar{u}^e = \begin{bmatrix} \hat{u}_e \\ 0 \\ \vdots \\ 0 \\ \hat{u}_{e+1} \end{bmatrix}$$

Juntando dichos sistemas llegamos a un único sistema lineal de la forma

$$\begin{bmatrix} \mathbf{A_{uu}} & \mathbf{A_{uq}} \\ \mathbf{A_{qu}} & \mathbf{A_{qq}} \end{bmatrix} \begin{bmatrix} \bar{u}^e \\ \bar{q}^e \end{bmatrix} = \begin{bmatrix} \mathbf{A^e_{u\hat{u}}} \\ \mathbf{A^e_{q\hat{u}}} \end{bmatrix} \hat{u}^e \tag{39}$$

Por lo tanto nuestras soluciones son

$$\begin{bmatrix} \bar{u}^e \\ \bar{q}^e \end{bmatrix} = \begin{bmatrix} \mathbf{A_{uu}} & \mathbf{A_{uq}} \\ \mathbf{A_{qu}} & \mathbf{A_{qq}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{A^e_{u\hat{u}}} \\ \mathbf{A^e_{q\hat{u}}} \end{bmatrix} \hat{u}^e \tag{40}$$

Considerando (25) unidimensional $[\![\hat{q}n]\!] = 0$ y que $\hat{q}_L = \hat{q}_{e-1}$, $\hat{q}_R = \hat{q}_e$, pues Ω_{e-1} y Ω_e son elementos consecutivos que comparten el nodo x_e, llegamos a

$$[\![\hat{q}n]\!]_{|\, x=x_e} = q_{p+1}^{e-1} - q_1^e + 2\tau\hat{u}_e - \tau u_{p+1}^{e-1} - \tau u_1^e = 0. \tag{41}$$

Substituyendo la solución del problema local (40) en (41) llegamos a la ecuación discreta

$$a\hat{u}_{e-1} + b\hat{u}_e + a\hat{u}_{e+1} = 0, \tag{42}$$

la cual se resuelve de la misma manera que (5). Por lo tanto, el error nos queda en función del parámetro de estabilidad τ. Estos son algunos ejemplos del error en función del parámetro τ y p.

Para $p = 2$ tenemos que

$$\frac{\tilde{k}-k}{k} = \begin{cases} \dfrac{1-\tau^2}{14400\tau}(kh)^5 + \dfrac{7\tau^4 - 10\tau^2 + 7}{1008000\tau^2}(kh)^6 + O((hk)^7) & \text{para } \tau \text{ cualquiera} \\ \dfrac{1}{12960}(kh)^4 - \dfrac{407}{4898880}(kh)^6 + O((kh)^8) & \text{para } \tau = kh, \\ \dfrac{1}{252000}(kh)^6 + \dfrac{1}{16200000}(kh)^8 + O((kh)^{10}) & \text{para } \tau = 1. \end{cases}$$

Para $p = 3$ tenemos que

$$\frac{\tilde{k}-k}{k} = \begin{cases} \dfrac{1-\tau^2}{2822400\tau}(kh)^7 + \dfrac{9\tau^4-14\tau^2+9}{355622400\tau^2}(kh)^8 + O((kh)^9) & \text{para } \tau \text{ cualquiera} \\ \dfrac{1}{2620800}(kh)^6 - \dfrac{4369}{10732176000}(kh)^8 + O((kh)^{10}) & \text{para } \tau = kh, \\ \dfrac{1}{88905600}(kh)^8 - \dfrac{1}{26622288000}(kh)^{10} + O((kh)^{12}) & \text{para } \tau = 1. \end{cases}$$

Para $p = 5$ tenemos que

$$\frac{\tilde{k}-k}{k} = \begin{cases} \dfrac{1-\tau^2}{442597478400\tau}(kh)^{11} + \dfrac{13\tau^4-22\tau^2+13}{126582878822400\tau^2}(kh)^{12} + O((kh)^{13}) & \text{para } \tau \text{ cualquiera} \\ \dfrac{1}{422479411200}(kh)^{10} - \dfrac{14197}{5709175523251200}(kh)^{12} + O((kh)^{14}) & \text{para } \tau = kh, \\ \dfrac{1}{31645719705600}(kh)^{12} - \dfrac{1}{3313671996096000}(kh)^{14} + O((kh)^{16}) & \text{para } \tau = 1. \end{cases}$$

Capítulo 4
Análisis de resultados

Una vez realizado los cálculos del error de dispersión para la ecuación de Helmholtz con CG y HDG, utilizando diferentes grados para la aproximación numérica de la solución, procedemos a un estudio con gráficas del error que nos ayudarán a concluir los resultados obtenidos. Antes de todo, recordemos que la longitud de onda k y la longitud h de cada elemento se definen como

$$k = \frac{2\pi}{L}, \quad h = \frac{L}{n}$$

siendo L la longitud de la onda y n el número de nodos por longitud de onda. En nuestras gráficas tomaremos en el eje OX el valor de $\xi = kh$, definido en el intervalo

$$\xi \in [0, \frac{\pi}{2}].$$

La interpretación física de la elección de este intervalo radica en que para $\xi = \pi/4$ tendremos 8 nodos por longitud de onda y para $\xi = \pi/2$ tendremos 4 nodos por longitud de onda, siendo 8 nodos por longitud de onda el valor más extendido en la práctica, de manera que nuestro interés se centrará en el comportamiento a partir de $\xi = \pi/4$ ya que eso nos indicará la precisión de los métodos con menos nodos por longitud de onda.

En el intervalo OY tomaremos el valor

$$\frac{\tilde{k}}{k},$$

el cual nos indicará el error de dispersión. Recordemos que se define el error de dispersión como

$$\frac{\tilde{k} - k}{k} = \frac{\tilde{k}}{k} - 1 = C_1(kh)^2 + C_2(kh)^4 + \cdots$$

Por lo tanto, tendremos que

$$\frac{\tilde{k}}{k} = 1 + C_1\xi^2 + C_2\xi^4 + \cdots$$

Por lo tanto, para valores cercanos a 1 tendremos que el error de dispersión es pequeño y que el método numérico utilizado es preciso y eficiente. Procedamos pues a explicar los resultados obtenidos.

Primero de todo ilustraremos la gráfica del error para CG entre los grados $p = 1$ y $p = 2$ (Figura 1).

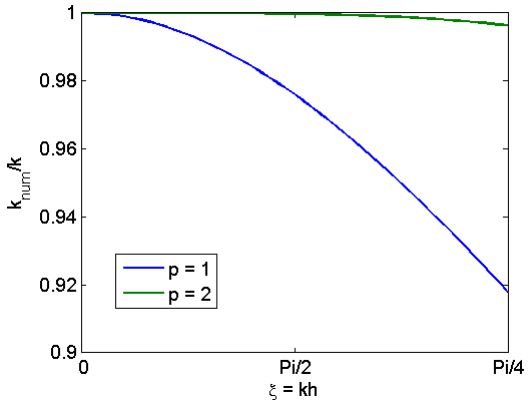

FIGURA 1. Gráfica del error de dispersión con CG para $p = 1$ y $p = 2$, en azul y verde respectivamente.

Podemos observar que utilizando CG con $p = 1$ obtenemos un resultado muy poco preciso con 8 nodos por longitud de onda, y que para 4 nodos por longitud de onda nuestro error de dispersión es bastante elevado y no obtenemos los resultados deseados. Por lo tanto, descartaremos aproximaciones de orden lineal $p = 1$ ya que no son precisas ni eficientes. A partir de ahora nos centraremos en los grados $p = 2, 3$ y 5.

Veamos ahora la gráfica del error utilizando CG de aproximaciones numéricas para los grados $p = 2, 3$ y 5 (Figura 2).

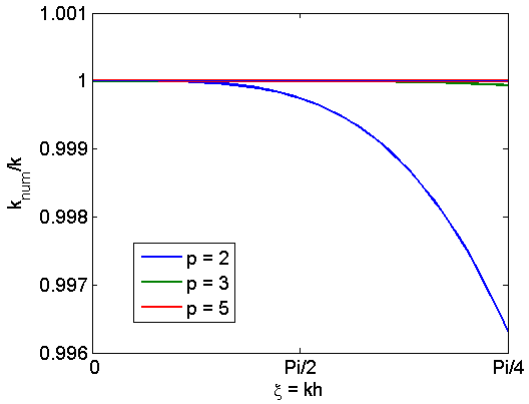

FIGURA 2. Gráfica del error de dispersión con CG para $p = 2, 3$ y 5, en azul, verde y rojo respectivamente.

Podemos observar que para 8 y 4 nodos por longitud de onda los grados de aproximación numérica 3 y 5 son muy precisos. En cambio, para una aproximación numérica de grado 2 y 8 nodos por longitud de onda los resultados obtenidos no son los deseados en comparación con los resultados obtenidos para los grados 3 y 5. Para 4 nodos por longitud de onda y grado 2 los errores de dispersión empiezan a ser significativos.

Veamos ahora los errores de dispersión para HDG utilizando aproximaciones numéricas de orden $2, 3$ y 5. Recordemos que el error de dispersión para HDG dependía del parámetro de estabilidad τ. Se han realizado gráficas del error para HDG tomando el valor τ igual a $\tau = kh$ y $\tau = 1$. Mostraremos primero la gráfica del error para $\tau = kh$ (Figura 3).

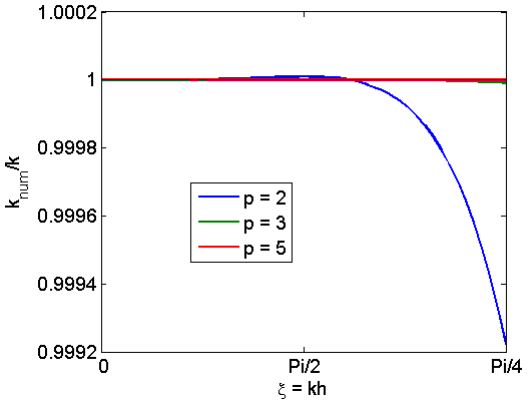

FIGURA 3. Gráfica del error de dispersión con HDG para los grados $p = 2, 3$ y 5, en azul, verde y rojo respectivamente, utilizando $\tau = kh$.

La gráfica muestra que utilizando 8 nodos por longitud de onda el método HDG, con $\tau = xi$, no tiene error de dispersión para los grados $2, 3$ y 5. Para 4 nodos por longitud de onda la aproximación de grado 2 empieza a tener errores de dispersión, pero fijándonos en los valores de la escala del eje OY vemos que el error de dispersión es muy pequeño. Para los grados 3 y 5 no hay error de dispersión alguno para 8 y 4 nodos por longitud de onda.

Veamos ahora HDG para el valor $\tau = 1$ (Figura 4).

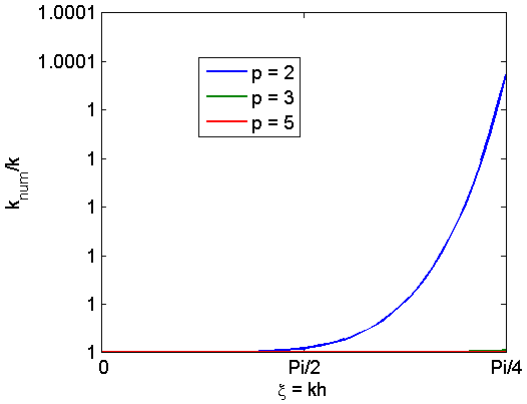

FIGURA 4. Gráfica del error de dispersión con HDG para los grados $p = 2, 3$ y 5, en azul, verde y rojo respectivamente, utilizando $\tau = 1$.

La interpretación de esta gráfica del error de dispersión reside en la observación de los valores que toma el eje OY, los cuales son 1 y 1,0001. Esto significa que HDG con $\tau = 1$ no tiene error de dispersión alguno.

Una vez realizada la comparativa gráfica del error de dispersión para diferentes métodos, valores de estabilidad y grados de aproximación numérica, concluimos que el método HDG con parámetro de estabilidad $\tau = 1$ es el método con menor error de dispersión, independientemente del grado de la aproximación numérica y de la cantidad de nodos por longitud de onda. Por último, mostraremos la comparación entre CG, HDG con $\tau = kh$ y HDG con $\tau = 1$, para diferentes grados, en la Figura 5.

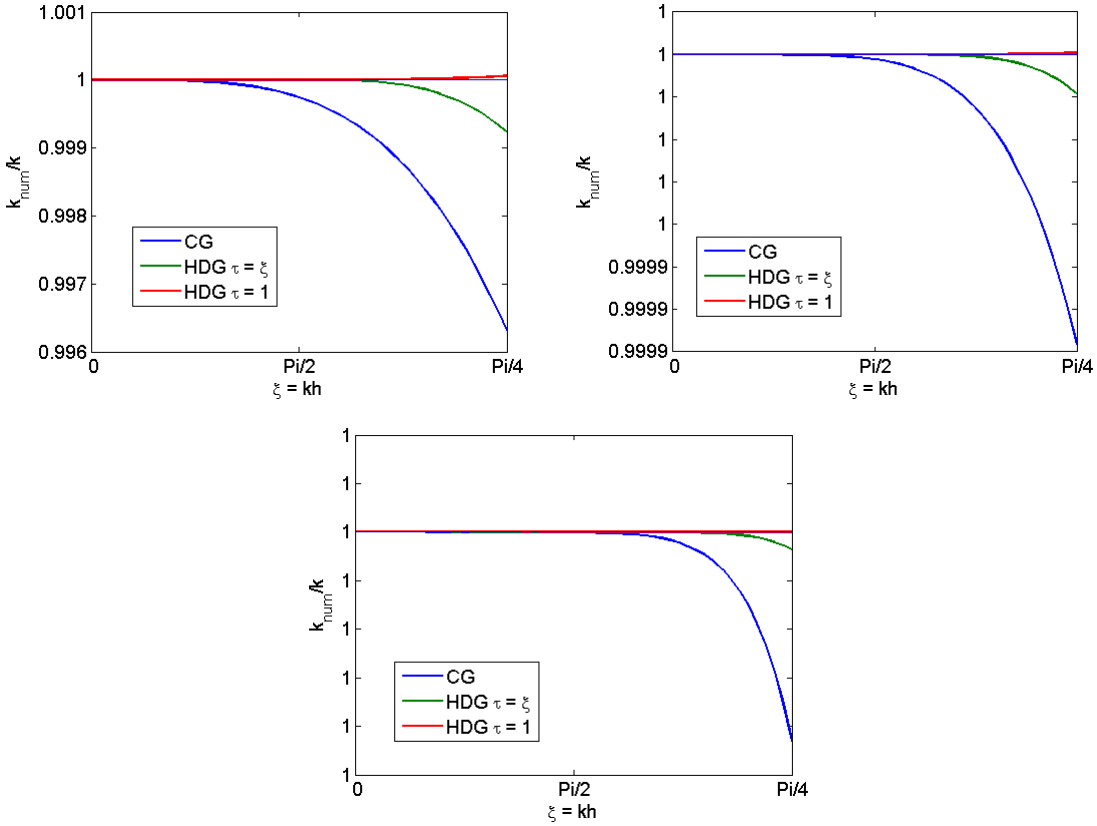

FIGURA 5. Grado 2, 3 y 5 respectivamente.

Conclusiones

Las aproximaciones lineales con CG son poco precisas y presentan errores de dispersión. Las aproximaciones numéricas de alto orden en CG son mucho más precisas y no presentan errores de dispersión con 8 nodos por longitud de onda por lo cual descartamos la utilización de aproximaciones numéricas lineales. HDG es eficiente y no presenta error de dispersión numérico. La variabilidad del parámetro de estabilidad proporciona a HDG una mayor precisión que CG para aproximaciones numéricas de alto orden para 8 y 4 nodos por longitud de onda.

Bibliografía

[1] Giorgiani, G.; Modesto, D.; Fernández-Méndez, S. and Huerta, A., *High-order continuous and discontinuous Galerkin methods for wave problems*. International Journal for Numerical Methods in Fluids, Vol. 73, Issue 10, pp. 883–903, 2013.

[2] Cadenas, C.; S. Fernández-Méndez, S and Huerta, A., *Análisis de dispersión de LDG para la ecuación de Helmholtz 1D*. Documento interno de LaCàN.

[3] Deumal, P., *Estudi d'alternatives per a la resolució de problemes no lineals discretitzats amb HDG*. Projecte Tecnológic, pp. 7–13, 2013.

Anexos

Anexo A: Códigos de Maple

CG para la ecuación de Helmholtz

Este código nos permite calcular el error en los elementos de la discretización realizando condensación estática.

cmtt

```
restart: with('LinearAlgebra'):

#Podemos asumir que xi > 0 pues xi =kh, donde h > 0 es una longitud y k > 0 por construcción.

assume(xi > 0);

#Aquí definimos el grado de los polinomios interpoladores y los construimos.

p := 2;

if p = 1 then
    L := [(xi+1)/2,(1-xi)/2];
else
    Ax := 2/p:
    xs := [seq(-1+Ax*i, i = 0..p)];
    L_i := product((xi - xs[i+1]), i = 1..p): L_i := L_i/subs(xi = xs[1], L_i);
    L := [L_i];
    for i from 1 to p-1 do
        L_i := product((xi - xs[j+1]),j = 0..i-1)*product((xi - xs[j+1]),j = i+1..p):
        L_i := L_i/subs(xi = xs[i+1], L_i);
        L := [op(L), L_i]:
    od:
    L_i := product((xi - xs[j+1]), j = 0..p-1): L_i := L_i/subs(xi = xs[p+1], L_i);
    L := [op(L), L_i]:
fi:

#Definimos N como la función de forma elemental y dN su derivada. Se utilizan para construir las
    matrices de masa y difusión.

N := L:
dN := diff(N, xi):

#Dibujamos los polinomios N en el intervalo [-1,1]

plot(N, xi = -1..1):
```

```
35
36 #Construimos la matriz de masa M
37
38 M := Matrix(1 .. p+1, 1 .. p+1):
39 for i from 1 to p+1 do
40     for j from 1 to i do
41         M[i,j] := int(N[i]*N[j], xi = -1..1);
42         M[j,i] := M[i,j];
43     od:
44 od:
45
46 hMe := (h^2/2)*M:
47
48 s := 0:
49 for i from 1 to p+1 do
50     for j from 1 to p+1 do
51         s := s + M[i,j];
52     od:
53 od: s;
54
55 K := Matrix(1 .. p+1, 1 .. p+1):
56 for i from 1 to p+1 do
57     for j from 1 to i do
58         K[i,j] := int(dN[i]*dN[j], xi = -1..1);
59         K[j,i] := K[i,j];
60     od:
61 od:
62
63 hKe := 2*K:
64
65 #Creamos la matriz Ae y su submatriz Ae_int
66
67 Ae := (-hKe + (xi^2/h^2)*hMe);
68 Ae_int := SubMatrix(Ae, [2..p], [2..p]);
69
70 #Creamos los vectores que tienen las componentes de las ecuaciones del primer y último nodo en
       cada elemento
71
72 Au[e-1] := Transpose(convert(SubMatrix(Transpose(Ae),[1..p+1],[p+1]),Vector)):
73 Au[e+1] := Transpose(convert(SubMatrix(Transpose(Ae),[1..p+1],[1]),Vector)):
74
75 #Creamos los vectores columna para resolver el sistema de los nodos interiores
76
77 c[initial] := convert(SubMatrix(Ae,[2..p],[1]),Vector):
78 c[final] := convert(SubMatrix(Ae,[2..p],[p+1]),Vector):
79
80 #Resolvemos y encontramos el valor de los nodos interiores para los elementos e-1 y e
81
82 U[e-1] := collect(Vector([u[e-1], LinearSolve(Ae_int, -u[e-1]*c[initial] - u[e]*c[final]), u[e]]),
       {u[e-1], u[e]});
83 U[e+1] := collect(Vector([u[e], LinearSolve(Ae_int, -u[e]*c[initial] - u[e+1]*c[final]), u[e+1]]),
       {u[e], u[e+1]});
84
85 #Generamos la ecuación discreta
86
87 P := collect(simplify(Au[e-1].U[e-1] + Au[e+1].U[e+1]), {u[e-1], u[e], u[e+1]}):
88
89 #Obtenemos los valores a y b de la ecuación au_{e-1}+bu_{e}+au_{e+1} = 0
90
91 a := simplify(coeff(P, u[e-1], 1));
92 b := simplify(coeff(P, u[e], 1));
```

```
93
94 #Calculamos el error
95
96 xi_num0 := arccos(simplify(-b/(2*a))):
97 xi_num := simplify(taylor(xi_num0,xi = 0, 15));
```

HDG para la ecuación de Laplace

Este código nos permite calcular la solución numérica de la ecuación de Laplace. Este código fue creado para *testear* el HDG en la ecuación de Helmholtz.

cmtt

```
1  restart: with('LinearAlgebra'):
2
3  #Podemos asumir que xi > 0 pues xi =kh, donde h > 0 es una longitud y k > 0 por construcción.
4
5  assume(xi > 0);
6
7  #Aquí definimos el grado de los polinomios interpoladores y los construimos.
8
9  p := 2:
10
11 if p = 1 then
12     L := [(xi+1)/2, (1-xi)/2];
13 else
14   Ax := 2/p:
15   xs := [seq(-1 + Ax*i, i = 0..p)];
16   L_i := product((xi - xs[i+1]),i = 1..p):
17   L_i := L_i/subs(xi = xs[1], L_i);
18   L := [L_i];
19   for i from 1 to p-1 do
20       L_i := product((xi - xs[j+1]),j = 0..i-1)*product((xi - xs[j+1]),j = i+1..p):
21       L_i := L_i/subs(xi = xs[i+1], L_i);
22       L := [op(L), L_i]:
23   od:
24   L_i := product((xi - xs[j+1]), j = 0..p-1):
25   L_i := L_i/subs(xi = xs[p+1], L_i);
26   L := [op(L), L_i]:
27 fi:
28
29 #Definimos N como la función de forma elemental y dN su derivada. Se utilizan para construir las
       matrices de masa y difusión.
30
31 N := L:
32 dN := diff(N, xi):
33
34 #Dibujamos los polinomios N en el intervalo [-1,1]
35
36 plot(N, xi = -1..1):
37
38 #Construimos la matriz de masa M
39
40 M := Matrix(1 .. p+1, 1 .. p+1):
41 for i from 1 to p+1 do
42     for j from 1 to p+1 do
43         M[i,j] := int(N[i]*N[j], xi = -1..1);
44     od:
45 od:
46 s := 0:
47 for i from 1 to p+1 do
```

```
48      for j from 1 to p+1 do
49          s := s + M[i,j];
50      od:
51  od:
52
53  C := Matrix(1 .. p+1, 1 .. p+1):
54  for i from 1 to p+1 do
55      for j from 1 to p+1 do
56          C[i,j] := int(dN[i]*N[j], xi = -1..1);
57      od:
58  od:
59
60  #Construimos los vectores canónicos e_1 y e_{p+1} y las matrices auxiliares I1, I2, I3, las cuales
        se utilizan para sumar/restar los valores de u_e y q_e en los extremos del elemento.
61
62  e[1] := Transpose(Vector(p+1, shape = unit[1])):
63  e[p+1] := Transpose(Vector(p+1, shape = unit[p+1])):
64  I1 := Matrix(p+1,2,[Vector(p+1, shape = unit[1]), Vector(p+1, shape = unit[p+1])]):
65  I2 := Matrix([Vector(p+1, shape = unit[1]), Matrix(p+1,p-1), Vector(p+1, shape = unit[p+1])]):
66  I3 := Matrix(p+1,2,[-Vector(p+1, shape = unit[1]), Vector(p+1, shape = unit[p+1])]):
67
68  #Definimos las matrices que formaran nuestro sistema, siendo Ae_uq y Ae_uu las matrices que
        involucran a u_e, y Ae_qu y Ae_qq las matrices que involucran a q.
69
70  Ae_uq := Transpose(C):
71  Ae_uu := - tau*I2:
72
73  Ae_qu := C:
74  Ae_qq := (xi/2)*M:
75
76  #Concatenamos las matrices anteriores en una matriz de cuatro bloques.
77
78  Ae := Matrix([[Ae_uu,Ae_uq],[Ae_qu,Ae_qq]]):
79  Am1 := MatrixInverse(Ae):
80
81  #Definimos los valores del vector û_e en el sistema lineal Ae [u_e, q_e] = Âe[û_e]
82
83  Ae_uû := -tau*I1:
84  Ae_qû := I3:
85
86  #Concatenamos las matrices anteriores en una matriz de dos bloques.
87
88  Âe := Matrix([[Ae_uû],[Ae_qû]]):
89
90  #Creamos los vectores de dimensión 2x1 de los elementos \Omega_{e-1} y \Omega_{e}, siendo [û_{e
        -1},û_{e}] y [û_{e},û_{e+1}] dichos vectores
91
92  ûem1 := Matrix(2,1,[û[e-1], û[e]]):
93  ûe := Matrix(2,1,[û[e], û[e+1]]):
94
95  Ûem1 := Âe.ûem1:
96  Ûe := Âe.ûe:
97
98  #Definimos las matrices tales que u_e = Bû_e y q_e = Cû_e
99
100 B := SubMatrix(Am1,[1..p+1],[1..2*(p+1)]):
101 C := SubMatrix(Am1,[(p+2)..2*(p+1)],[1..2*(p+1)]):
102
103 #Definimos los valores de u_e y q_e en el elemento e-1
104
105 Uem1 := simplify(convert(B.Ûem1,Vector)):
```

```
Qem1 := simplify(convert(C.Ûem1, Vector)):

#Definimos los valores de u_e y q_e en el elemento e

Ue := simplify(convert(B.Ûe, Vector)):
Qe := simplify(convert(C.Ûe, Vector)):

#Ensamblamos las ecuaciones para obtener la ecuación discreta P que nos permitirá calcular el
    error en cada elemento de la malla

Qleft := collect(simplify(e[p+1].(Qem1 -tau*Uem1) + tau*û[e]),{û[e-1],û[e]}):
Qright := collect(simplify(e[1].(-Qe - tau*Ue) + tau*û[e]),{û[e],û[e+1]}):

P := collect(simplify(Qleft + Qright),{û[e-1],û[e],û[e+1]}):

#Obtenemos los valores a y b de la ecuación au_{e-1}+bu_{e}+cu_{e+1} = 0, donde a = c

a := simplify(coeff(P,û[e-1],1)):
b := simplify(coeff(P,û[e],1)):
c := simplify(coeff(P,û[e+1],1)):

#Definimos los valores de a y b en los extremos para poder dibujar la solución en MatLab

a_boundary := simplify(coeff(Qleft,û[e-1],1)):
b_boundary := simplify(coeff(Qleft,û[e],1)):
```

HDG para la ecuación de Helmholtz

Este código calcula el error en los elementos de la discretización para HDG en la ecuación de Helmholtz. También obtiene los valores numéricos de las condiciones de contorno por tal de poder implementar dichos valores en los códigos de Matlab.

cmtt

```
restart: with('LinearAlgebra'):

#Podemos asumir que xi > 0 pues xi =kh, donde h > 0 es una longitud y k > 0 por construcción.

assume(xi > 0);

#Aquí definimos el grado de los polinomios interpoladores y los construimos.

p := 2:

if p = 1 then
    L := [(xi+1)/2, (1-xi)/2];
else
  Ax := 2/p:
  xs := [seq(-1 + Ax*i, i = 0..p)];
  L_i := product((xi - xs[i+1]),i = 1..p):
  L_i := L_i/subs(xi = xs[1], L_i);
  L := [L_i];
  for i from 1 to p-1 do
      L_i := product((xi - xs[j+1]),j = 0..i-1)*product((xi - xs[j+1]),j = i+1..p):
      L_i := L_i/subs(xi = xs[i+1], L_i);
      L := [op(L), L_i]:
  od:
  L_i := product((xi - xs[j+1]), j = 0..p-1):
  L_i := L_i/subs(xi = xs[p+1], L_i);
  L := [op(L), L_i]:
fi:
```

```
28
29 #Definimos N como la función de forma elemental y dN su derivada. Se utilizan para construir las
       matrices de masa y difusión.
30
31 N := L:
32 dN := diff(N, xi):
33
34 #Dibujamos los polinomios N en el intervalo [-1,1]
35
36 plot(N, xi = -1..1):
37
38 #Construimos la matriz de masa M
39
40 M := Matrix(1 .. p+1, 1 .. p+1):
41 for i from 1 to p+1 do
42     for j from 1 to p+1 do
43         M[i,j] := int(N[i]*N[j], xi = -1..1);
44     od:
45 od:
46 s := 0:
47 for i from 1 to p+1 do
48     for j from 1 to p+1 do
49        s := s + M[i,j];
50     od:
51 od:
52
53 C := Matrix(1 .. p+1, 1 .. p+1):
54 for i from 1 to p+1 do
55     for j from 1 to p+1 do
56         C[i,j] := int(dN[i]*N[j], xi = -1..1);
57     od:
58 od:
59
60 #Construimos los vectores canónicos e_1 y e_{p+1} y las matrices auxiliares I1, I2, I3, las cuales
       se utilizan para sumar/restar los valores de u_e y q_e en los extremos del elemento.
61
62 e[1] := Transpose(Vector(p+1, shape = unit[1])):
63 e[p+1] := Transpose(Vector(p+1, shape = unit[p+1])):
64 I1 := Matrix(p+1,2,[Vector(p+1, shape = unit[1]), Vector(p+1, shape = unit[p+1])]):
65 I2 := Matrix([Vector(p+1, shape = unit[1]), Matrix(p+1,p-1), Vector(p+1, shape = unit[p+1])]):
66 I3 := Matrix(p+1,2,[-Vector(p+1, shape = unit[1]), Vector(p+1, shape = unit[p+1])]):
67
68 #Definimos las matrices que formaran nuestro sistema, siendo Ae_uq y Ae_uu las matrices que
       involucran a u_e, y Ae_qu y Ae_qq las matrices que involucran a q.
69
70 Ae_uq := Transpose(C):
71 Ae_uu := (xi/2)*M - tau*I2:
72
73 Ae_qu := C:
74 Ae_qq := (xi/2)*M:
75
76 #Concatenamos las matrices anteriores en una matriz de cuatro bloques.
77
78 Ae := Matrix([[Ae_uu,Ae_uq],[Ae_qu,Ae_qq]]):
79 Am1 := MatrixInverse(Ae):
80
81 #Definimos los valores del vector û_e en el sistema lineal Ae [u_e, q_e] = Âe[û_e]
82
83 Ae_uû := -tau*I1:
84 Ae_qû := I3:
85
```

```
#Concatenamos las matrices anteriores en una matriz de dos bloques.

Âe := Matrix([[Ae_uû],[Ae_qû]]):

#Creamos los vectores de dimensión 2x1 de los elementos \Omega_{e-1} y \Omega_{e}, siendo [û_{e
    -1},û_{e}] y [û_{e},û_{e+1}] dichos vectores

ûem1 := Matrix(2,1,[û[e-1], û[e]]):
ûe := Matrix(2,1,[û[e], û[e+1]]):

Ûem1 := Âe.ûem1:
Ûe := Âe.ûe:

#Definimos las matrices tales que u_e = Bû_e y q_e = Cû_e

B := SubMatrix(Am1,[1..p+1],[1..2*(p+1)]):
C := SubMatrix(Am1,[(p+2)..2*(p+1)],[1..2*(p+1)]):

#Definimos los valores de u_e y q_e en el elemento e-1

Uem1 := simplify(convert(B.Ûem1,Vector)):
Qem1 := simplify(convert(C.Ûem1,Vector)):

#Definimos los valores de u_e y q_e en el elemento e

Ue := simplify(convert(B.Ûe,Vector)):
Qe := simplify(convert(C.Ûe,Vector)):

#Ensamblamos las ecuaciones para obtener la ecuación discreta P que nos permitirá calcular el
    error en cada elemento de la malla

Qleft := collect(simplify(e[p+1].(Qem1 -tau*Uem1) + tau*û[e]),{û[e-1],û[e]}):
Qright := collect(simplify(e[1].(-Qe - tau*Ue) + tau*û[e]),{û[e],û[e+1]}):

P := collect(simplify(Qleft + Qright),{û[e-1],û[e],û[e+1]}):

#Obtenemos los valores a y b de la ecuación au_{e-1}+bu_{e}+cu_{e+1} = 0, donde a = c

a := simplify(coeff(P,û[e-1],1)):
b := simplify(coeff(P,û[e],1)):
c := simplify(coeff(P,û[e+1],1)):

#Definimos los valores de a y b en los extremos para poder dibujar la solución en MatLab

a_boundary := simplify(coeff(Qleft,û[e-1],1)):
b_boundary := simplify(coeff(Qleft,û[e],1)):

#Calculamos el error dejando tau como un parámetro

xi_num0 := arccos(simplify(-b/(2*a))):
xi_num_tau := simplify(taylor(xi_num0,xi = 0,10)):

#Calculamos el error siendo tau del orden de xi

xi_num0 := arccos(simplify(subs(tau = xi,-b/(2*a)))):
xi_num_xi := simplify(taylor(xi_num0,xi = 0,10)):

#Calculamos el error siendo tau = 1

xi_num0 := arccos(simplify(subs(tau = 1,-b/(2*a)))):
```

```
144  xi_num_1 := simplify(taylor(xi_num0, xi = 0,10)):
```

Anexo B: Códigos de Matlab

CG para la ecuación de Helmholtz

Código para la ecuación de Helmholtz con CG que permite obtener el gráfico de la solución numérica.

cmtt

```
1
2  clear all, clc, close all
3  %Solucion analitica u(x) = exp(ikx)
4
5  %Condicion de contorno
6  u0 = 1;
7
8  k = 10;
9
10 %Preproceso (malla)
11
12 n = input('Numero de elementos = ');
13 p = input('Grado = ');
14 h = 1/n;
15 xi = k*h;
16
17 if p == 1
18     a = 1+xi^2/6;
19     b = -2+2*xi^2/3;
20 elseif p == 2
21     a = -(1/24)*(xi^4+16*xi^2+240)/(xi^2-10);
22     b = (1/12)*(3*xi^4-104*xi^2+240)/(xi^2-10);
23 elseif p == 3
24     a = (1/60)*(xi^6+30*xi^4+1080*xi^2+25200)/(xi^4-52*xi^2+420);
25     b = (2/15)*(xi^6-135*xi^4+2880*xi^2-6300)/(xi^4-52*xi^2+420);
26 elseif p == 5
27     a = (1/210)*(xi^10+70*xi^8+6720*xi^6+604800*xi^4+42336000*xi^2+1676505600)/(xi^8-352*xi
       ^6+35808*xi^4-1128960*xi^2+7983360);
28     b = (2/105)*(3*xi^10-2345*xi^8+430080*xi^6-24645600*xi^4+397958400*xi^2-838252800)/(xi^8-352*
       xi^6+35808*xi^4-1128960*xi^2+7983360);
29 end
30
31 uno = ones(n+1,1);
32 A = spdiags([a*uno, b*uno, a*uno],-1:1,n+1,n+1);
33 A(1,1) = b/2;
34 A(end,end)=b/2;
35 f = zeros(n+1,1);
36
37 %Condicion de contorno en x = 1
38
39 A(end,end)=A(end,end)-i*xi;
40
41 %Condicions de contorn en x = 0
42 f1 = A(2:end,1);
43 A = A(2:end,2:end);
44 f = f(2:end)-u0*f1;
45
46 %Resolucio sistema
47 u = A\f;
48
49 %Postproces
50 x = 0:h:1;
51 u = [u0;u];
```

```
figure(1)
plot(x,real(u),'-r*',x,real(exp(i*k*x)),'LineSmoothing','on'), legend('MEF','analitica')
%DibujaSol1D(X,T,u)
```

HDG para la ecuación de Helmholtz

Código para la ecuación de Helmholtz con HDG que permite obtener la solución numérica.

cmtt

```
%Solucion analitica u(x) = exp(ikx)
clear all, clc, close all
%Condicion de contorno
u0 = 1;

%Numero de onda
k = 100;

%Tau
tau = 1;

%Preproces (malla)
n = input('Numero de elementos = ');
p = input('Grado = ');
h = 1/n;
xi = k*h;

if p == 1
      tau = 0
      a = -(2*(tau^2*xi^3-66*tau^2*xi+14*tau*xi^2-xi^3+36*tau-6*xi))/((-xi+6*tau)*(2*tau*xi-xi
          ^2+12)*xi);
      b = -(2*(4*tau^2*xi^3-tau*xi^4-156*tau^2*xi+56*tau*xi^2-4*xi^3-72*tau+12*xi))/((-xi+6*tau)
          *(2*tau*xi-xi^2+12)*xi);
      a_boundary = -(2*(tau^2*xi^3-66*tau^2*xi+14*tau*xi^2-xi^3+36*tau-6*xi))/((-xi+6*tau)*(2*tau*
          xi-xi^2+12)*xi);
      b_boundary = -(4*tau^2*xi^3-tau*xi^4-156*tau^2*xi+56*tau*xi^2-4*xi^3-72*tau+12*xi)/((-xi+6*
          tau)*(2*tau*xi-xi^2+12)*xi);
elseif p == 2
    a =(3*(tau^2*xi^5+16*tau^2*xi^3+2*tau*xi^4-xi^5+240*tau^2*xi+96*tau*xi^2-16*xi^3+2400*tau-240*
        xi))/((72*tau^2*xi^3-18*tau*xi^4+xi^5-720*tau^2*xi+912*tau*xi^2-72*xi^3-7200*tau+720*xi)*
        xi);
    b = -(2*(9*tau^2*xi^5-tau*xi^6-312*tau^2*xi^3+162*tau*xi^4-9*xi^5+720*tau^2*xi-3312*tau*xi
        ^2+312*xi^3+7200*tau-720*xi))/((72*tau^2*xi^3-18*tau*xi^4+xi^5-720*tau^2*xi+912*tau*xi
        ^2-72*xi^3-7200*tau+720*xi)*xi);
    a_boundary = (3*(tau^2*xi^5+16*tau^2*xi^3+2*tau*xi^4-xi^5+240*tau^2*xi+96*tau*xi^2-16*xi
        ^3+2400*tau-240*xi))/((72*tau^2*xi^3-18*tau*xi^4+xi^5-720*tau^2*xi+912*tau*xi^2-72*xi
        ^3-7200*tau+720*xi)*xi);
    b_boundary = -(9*tau^2*xi^5-tau*xi^6-312*tau^2*xi^3+162*tau*xi^4-9*xi^5+720*tau^2*xi-3312*tau*
        xi^2+312*xi^3+7200*tau-720*xi)/((72*tau^2*xi^3-18*tau*xi^4+xi^5-720*tau^2*xi+912*tau*xi
        ^2-72*xi^3-7200*tau+720*xi)*xi);
end

uno = ones(n+1,1);
A = spdiags([a*uno, b*uno, a*uno],-1:1,n+1,n+1);
f= zeros(n+1,1);

%Condicions de contorn en x=0
f1 = A(2:end,1);
A = A(2:end,2:end);
f = f(2:end)-u0*f1;

```

```
%Condicion de contorno en x=1
A(end,end-1) = a_boundary;
A(end,end) = b_boundary;
A(end,end) = A(end,end)-i;

%Resolucio sistema
u=A\f;

%Postproces
x = 0:h:1;
u = [u0;u];
figure(1)
plot(x,real(u),'-r*',x,real(exp(i*k*x)),'LineSmoothing','on'), legend('HDG','analitica');
```