Année scolaire 2023/2024 Rapport de stage

UFR de mathématique et d'informatique Université de Strasbourg

IRMA Institut de Recherche Mathématique Avancée

Stage de fin d'étude de Master 1 CSMI

Sujet: Galerkine discontinu en 1D

Auteur :
Adama Dieng

Superviseur :
Joubine Aghili

Date : August 20, 2024

Contents

Chapter 1

Context

Pour realiser ce rapport, on va suivre l'article de Cockburn [2]

1.1 Introduction

1.1.1 Informations Préliminaires

Remerciements
Je tiens à exprimer ma gratitude à mon encadrant, M. Joubine Aghili, pour son accompagnement et ses enseignements durant ce stage. Je suis également très reconnaissant pour sa patience et sa disponibilité lorsque j'avais besoin d'aide, ainsi que pour m'avoir offert l'opportunité de travailler sur un sujet qui répond à mes intérêts académiques.

A propos du stage et du laboratoire de recherche
Ce stage fait partie du cursus du Master CSMI de l'Universitéde Strasbourg. Il s'agit d'un stage de 2 mois, qui a débuté le 3 juin et s'est terminé le 3 août 2024.

Il a eu lieu au sein du laboratoire **IRMA**, qui est une unité de recherche de CNRS(Centre national de la recherche scientifique) et de l'Université de Strasbourg. Le centre est situé au campus de l'université de Strasbourg, à Strasbourg, en France.

1.1.2 Contexte du stage

Les méthodes numériques sont indispensables pour l'analyse et la modélisation des systèmes physiques complexes, en particulier dans le cadre de la résolution des équations différentielles partielles (EDP).

Ces dernières apparaissent fréquemment dans divers domaines scientifiques et techniques, allant de la mécanique des fluides à la propagation des ondes.

Parmi les méthodes numériques disponibles, les méthodes de Galerkine ont longtemps été privilégiées pour leur capacité à fournir des solutions précises avec un coût computationnel raisonnable. Toutefois, les méthodes classiques de Galerkine, comme les éléments finis continus, présentent des limitations lorsqu'elles sont confrontées à des solutions avec des discontinuités ou des gradients abrupts.

La méthode de Galerkine discontinue (DG) est une variante qui se distingue par sa capacité à traiter efficacement de tels défis de discontinuité. Contrairement aux méthodes continues, la méthode DG permet l'utilisation de fonctions de base discontinues, offrant ainsi une flexibilité accrue dans la représentation des solutions. Cette caractéristique est particulièrement utile dans les contextes où les solutions présentent des discontinuités naturelles, telles que les chocs ou les interfaces de phase, comme c'est souvent le cas dans les équations de conservation hyperboliques.

Ce rapport vise à explorer en détail la méthode de Galerkine discontinue dans le contexte unidimensionnel (1D). L'objectif principal est de fournir une compréhension claire et complète des principes sous-jacents de cette méthode, de démontrer son implémentation pratique, et d'examiner ses avantages par rapport aux autres méthodes numériques. Nous aborderons également les défis spécifiques associés à la mise en œuvre de la méthode DG, notamment le traitement des flux numériques et la stabilisation des solutions.

En première partie, nous introduirons les fondements théoriques de la méthode de Galerkine discontinue, en expliquant les concepts clés et en mettant en lumière ses avantages distinctifs. Ensuite, la deuxième partie sera consacrée à l'implémentation pratique de cette méthode en 1D, avec une description détaillée des étapes de discrétisation et des exemples d'application. Enfin, nous conclurons par une discussion sur les résultats obtenus, les limitations rencontrées, et les perspectives d'amélioration et d'extension de la méthode DG.

Cette exploration détaillée nous permettra de mieux comprendre comment la méthode de Galerkine discontinue peut être utilisée pour résoudre efficacement une large gamme de problèmes d'EDP, en mettant en lumière son potentiel et ses domaines d'application future.

1.1.3 Approximation discontinue par méthode Galerkin Discontinue

L'idée fondamentale de la méthode Galerkin Discontinu (DG) consiste, comme pour une méthode des éléments finis, à mettre la problème sous forme variationnelle, et à réaliser une approximation polynomiale de la solution du système.

Toutefois, contrairement aux éléments finis, où la solution est continue sur tout le domaine, l'approximation DG est une approximation locale dans chaque cellule du maillage, la valeur de la solution aux interfaces est à priori **multi-valuée** et la solution globale est donc discontinue sur les faces des éléments volumiques. De ce fait, des flux numériques sont calculés aux interfaces de chaque cellule, tel que cela est réaliser dans les méthodes de type volumes finis.

Par exemple en 1D, l'idée est de diviser le domaine de calcul (un intervalle) en plusieurs elements $[x^{k-1}, x^k], [x^k, x^{k+1}], etc$ sur lesquels on va construire des u^k qui sont des approximations de la solution dans chaque élément. **figure1.1**. Contrairement aux méthodes d'éléments finis classiques, la méthode DG autorise des discontinuités aux frontières des éléments. Cela signifie que la valeur de la solution peut changer brusquement d'un élément à l'autre. Les points $x^{k-1}, x^k, x^{k+1} etc$ sont les frontières (**les interfaces**) entre les éléments et les solution u^k sont des polynomes de degré k sur chaque élément.

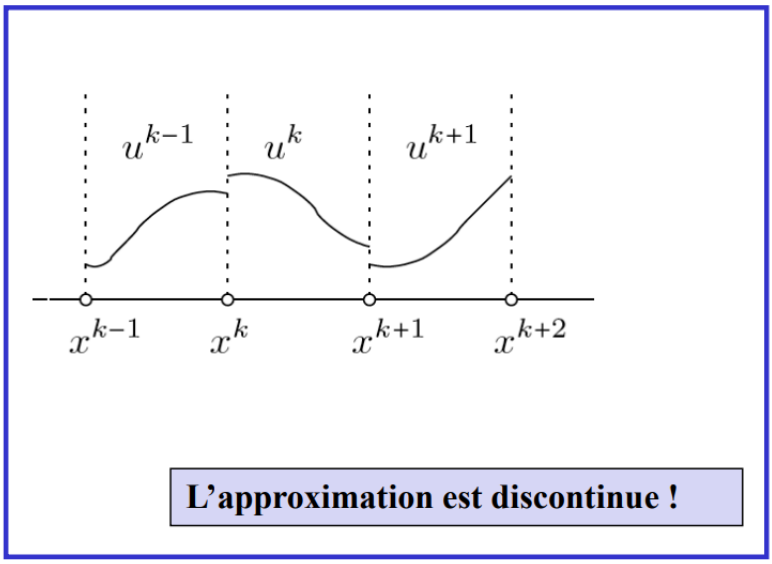

Figure 1.1: approximation discontinue en 1D [1]

Chapter 2

Rappels sur la méthode des éléments finis MEF

2.1 La méthode des éléments finis (MEF)

La méthode des éléments finis (MEF) est une technique numérique utilisée pour trouver des solutions approximatives aux équations différentielles partielles (EDP) et intégrales. Elle est largement employée dans les domaines de la mécanique, la physique, l'ingénierie, et plus particulièrement dans la résolution de problèmes liés aux déformations, à la diffusion de la chaleur, à la dynamique des fluides, et autres phénomènes physiques modélisables par des EDP.

2.1.1 Principe Fondamental de la MEF en 1D

En une dimension (1D), la MEF consiste à diviser le domaine de calcul, qui est souvent un intervalle sur l'axe des abscisses, en un certain nombre de sous-domaines appelés éléments finis. Sur chaque élément, la solution est approximée par une fonction de forme simple, généralement un polynôme.

Soit une équation différentielle ordinaire (EDO) linéaire de la forme :

$$-\frac{d}{dx}\left(k(x)\frac{du}{dx}\right) + c(x)u = f(x), \quad x \in (a,b)$$

avec les conditions aux limites appropriées. L'idée principale est de multiplier cette équation par une fonction test $v(x)$, puis d'intégrer sur le domaine $[a,b]$. Cela conduit à la formulation dite **faible** ou **variationnelle** du problème. Celle-ci est de la forme:

$$-\int_a^b \frac{d}{dx}\left(k(x)\frac{du}{dx}\right)v(x)dx + \int_a^b c(x)u(x)v(x)dx = \int_a^b f(x)v(x)dx$$

pour tout $v(x)$ appartenant à un espace de fonctions tests approprié.

2.1.2 Discrétisation du Domaine

Le domaine $[a,b]$ est discrétisé en un nombre fini de sous-intervalles $[x_i, x_{i+1}]$, formant ainsi un maillage. La solution approchée $u_h(x)$ est construite comme une combinaison linéaire de fonctions de forme associées à chaque nœud du maillage :

$$u_h(x) = \sum_{i=1}^{n} u_i \phi_i(x)$$

où $\phi_i(x)$ sont les fonctions de forme locales, et u_i sont les coefficients inconnus (les valeurs de la solution approchée aux nœuds).

2.1.3 Assemblage des Équations

En substituant l'expression de $u_h(x)$ dans la formulation faible et en choisissant les fonctions tests $v(x)$ égales aux fonctions de forme $\phi_j(x)$, on obtient un système d'équations linéaires pour les coefficients u_i. Ce système est généralement écrit sous forme matricielle :

$$\mathbf{Ku} = \mathbf{F}$$

où $\mathbf{K}$ est la matrice de raideur, $\mathbf{u}$ est le vecteur des inconnues, et $\mathbf{F}$ est le vecteur des forces.

2.1.4 Extension en 2D

La méthode des éléments finis peut être étendue en deux dimensions (2D) en considérant des éléments finis comme des triangles ou des quadrilatères couvrant le domaine de calcul. La formulation et les concepts restent similaires, mais nécessitent une intégration numérique sur **des surfaces plutôt que sur des segments**.

Chapter 3

Méthode Galerkin Discontinue (DG)

3.1 Equation differentielle ordinaire (EDO) en 1D

3.1.1 Probleme abstrait et formulation faible

Considérons une EDO en 1D de la forme:

$$\begin{cases} \frac{d}{dt}u(t) = f(t)u(t), \quad t \in (0,T), \\ u(0) = u_0, \end{cases} \tag{3.1}$$

On suppose que nous voulons determiner une approximation u_h de la solution u de l'EDO en utilisant une méthode de Galerkin.

Pour cela, on va faire un discrétisation de l'intervalle $(0,T)$ en petit intervalles $I^n = (t_n, t_{n+1})$, pour tout $n = 0, 1, \ldots, N-1$.

On fait une sorte de reformulation faible de l'EDO en multipliant l'equation par une fonction v (polynome de degré au plus egale à k^n), car on cherche une fonction approximative u_h qui est de degré au plus égale à k^n sur chaque intervalle I^n. Ensuite on intègre sur chaque intervalle I^n.

On a donc:

$$\int_{I^n} \frac{d}{dt}u(t)v(t)dt = \int_{I^n} f(t)u(t)v(t)dt \tag{3.2}$$

En utilisant la formule d'integration par partie, on a:

$$-\int_{I^n} u(t)\frac{d}{dt}v(t)dt + [u(t)v(t)]_{t_n}^{t_{n+1}} = \int_{I^n} f(t)u(t)v(t)dt \tag{3.3}$$

On cherche à approcher u par u_h tel que:

$$u_h = \sum_{n=0}^{N} u_i \phi_i(t) \tag{3.4}$$

Où les fonctions $\tilde{\phi}_i$ sont des polynomes par morceaux de degré au plus égale à k^n, connue localelement sur chaque intervalle I^n mais elles sont discontinues globalement sur (0,T).

Dans la discrétisation de l'intervalle $(0,T)$, il est important aussi d'abserver que la frontiere de chaque element I^n est partagee par deux intervalles.

Par conséquent, il est necessaire de prendre en compte ce qui se passe aux interfaces de ces intervalles I^n.

Sur la base de cette partittion, les méthodes DG permettent de gerer les valeurs des solutions aux interfaces des intervalles I^n en introduisant le terme de **trace numérique** $\widehat{u}_h$.

On cherche alors u_h qui, sur l'intervalle I^n, est de degré au plus égale à k^n determinée en imposant que:

$$-\int_{I^n} u_h(s)\frac{d}{ds}v(s)ds + \widehat{u}_h v|_{t_n}^{t_{n+1}} = \int_{I^n} f(s)u_h(s)v(s)ds \tag{3.5}$$

pour tout polynome v de degré au plus égale à k^n. Pour completer cette definition de la méthode DG, il est necessaire de determiner la trace numérique $\widehat{u}_h$. Notons que la méthode etablit un lien entre les valeurs de u_h dans les differents intervalles uniquement par le biais de la trace numérique $\widehat{u}_h$.

3.1.2 Trace numérique $\widehat{u}_h$

Pour une EDO, l'information voyage " du passé vers le futur", on peut donc determiner $\hat{u}_h$ comme suit:

$$\widehat{u}_h(t_n) = \begin{cases} u_0, & \text{si} \quad t_n = 0, \\ lim_{\epsilon \to 0} u_h(t_n - \epsilon), & \text{sinon.} \end{cases} \tag{3.6}$$

Cela achève la definition de la méthode DG. Dans cet exemple, on peut deja identifier les elements clés de la méthode :

- L'utilisation d'aproximations discontinues pour la solution u_h;
- L'application de l'équation différentielle sur chaque intervalle via une formulation faible de Galerkin;
- L'introduction et la definiton adéquate de ce qu'on appelle la trace numérique $\hat{u}_h$.

La sélection de la trace numérique est probablement l'aspect le plus sensible et crucial de la méthode DG, car elle peut influencer sa cohérence, sa stabilité, et même sa précision. Le choix simple que nous avons effectué ici est bien adapté à ce cas et conduit à une méthode efficace.

On aborde d'abord la question de **la consistance** de la méthode de DG. Comme c'est habituel pour la majorité des méthodes des éléments finis, une méthode est dite consistante si la solution exacte u de l'EDO peut etre remplacée par la solution approximative u_h dans la formulation faible de Galerkin. Nous constatons immédiatement que cela est vérifié si et seulement si $\hat{u} = u$.

Quant à **la stabilité** de la méthode, un aspect plus delicat, l'approche consiste à etablir d'abord une propriété de stabilité pour l'equaton **(3.1)** avant de l'etendre à la méthode DG **(3.5)** en definissant de maniere appropriée la trace numérique $\hat{u}_h$. Si on multiplions l'EDO par u et integrant sur $(0, T)$, on obtient:

$$\frac{1}{2}u^2(T) - \frac{1}{2}u_0^2 = \int_0^T f(s)u^2(s)ds \tag{3.7}$$

où la stabilité-L^∞ de l'EDO est garantie par le fait que le terme de droite est positif. Pour la méthode DG, il est necessaire d'obtenir un égalité similaire. Pour y parvenir, il suffit de poser $v = u_h$ dans l'equation **(3.5)**, integrant par partie et en sommmant sur n. On obtient alors:

$$\sum_{n=0}^{N-1} \left(-\frac{1}{2}u_h^2 + \hat{u}_h u_h \right) |_{t_n}^{t_{n+1}} = \frac{1}{2}u_h^2(T^-) - \frac{1}{2}u_h^2(T^-) + \sum_{n=0}^{N-1} \left(-\frac{1}{2}u_h^2 + \hat{u}_h u_h \right) |_{t_n}^{t_{n+1}} + \frac{1}{2}u_0^2 - \frac{1}{2}u_0^2 = \int_0^T f(s)u_h^2(s)d(s) \tag{3.8}$$

Autrment dit :

$$\sum_{n=0}^{N-1} \left(-\frac{1}{2}u_h^2 + \hat{u}_h u_h \right) |_{t_n}^{t_{n+1}} = \frac{1}{2}u_h^2(T^-) + \Theta_h(T) - \frac{1}{2}u_0^2 = \int_0^T f(s)u_h^2(s)d(s) \tag{3.9}$$

où

$$\Theta_h(T) = -\frac{1}{2}u_h^2(T^-) + \sum_{n=0}^{N-1} \left(-\frac{1}{2}u_h^2 + \hat{u}_h u_h \right) |_{t_n}^{t_{n+1}} + \frac{1}{2}u_0^2$$

Quel critere sur $\Theta_h(t)$ pour determiner cet trace numérique $\hat{u}_h$?

Il est important de noter que si $\Theta_h(T)$ etait une quantité non-négative, l'egalité ci-dessus impliquerait la stabilité de la méthode DG. Autrement dit, la stabilité de la méthode DG est garantie si nous pouvons definir la trace numérique $\hat{u}_h$ de sorte que $\Theta_h(T) \geq 0$, [2].

Proposision 3.1: ($\mathbf{L}^2-$stabilité)[6]
Nous avons:

$$\frac{1}{2}||u_h||^2_{L^2(0,T)} + \Theta_h(T) \leq \frac{1}{2}||u_0||^2_{L^2(0,T)}$$

où $\Theta_h(T) \geq 0$.

En supposant que:

$$u_h(t) = u_0, \quad \text{si } t < 0,$$

et **en notant**:

$$\{u_h\} = \frac{1}{2}(u_h^+ + u_h^-), \quad [u_h] = u_h^- - u_h^+, \quad u_h^\pm = \lim_{\epsilon \to 0} u_h(t \pm \epsilon) \tag{3.10}$$

On peut ainsi réecrire $\Theta_h(T)$ comme suit:

$$\begin{aligned}\Theta_h(T) = &-\frac{1}{2}u_h^2(T^-) \\ &+ \left(-\frac{1}{2}u_h^2(T^-) + \hat{u}_h(T)u_h(T^-)\right) \\ &+ \sum_{n=1}^{N-1}\left(-\frac{1}{2}\left[u_h\right]^2 + \hat{u}_h\left[u_h\right]\right)(t_n) \\ &- \left(-\frac{1}{2}u_h^2(0^+) + \hat{u}_h(0)u_h(0^+)\right) + \frac{1}{2}u_0^2\end{aligned}$$

Avec l'identité suivante (obtenue avec les notations précédentes):

$$\left[u_h^2\right] = (u_h^+)^2 - (u_h^+)^2 = (u_h^+ - u_h^-)(u_h^+ + u_h^-) = 2\left\{u_h\right\}\left[u_h\right]$$

On obtient:

$$\Theta_h(T) = (\hat{u}_h(T) - u_h(T^-))u_h(T^-) + \sum_{n=1}^{N-1}\left((\hat{u}_h - \{u_h\})\left[u_h\right]\right)(t_n) - (\hat{u}_h(0) - u_0)u_h(0^+) + \frac{1}{2}\left[u_h\right]^2(0) \tag{3.11}$$

Il est clair que [2] si nous prenons:

$$\hat{u}_h(t_n) = \begin{cases} u_0, \quad \text{si} \quad t_n = 0, \\ (\{u_h\} + C^n\left[u_h\right])(t_n), \quad \text{si } t_n \in]0, T[, \\ u_h(T^-), \quad \text{si} \quad t_n = T, \end{cases} \tag{3.12}$$

Où $C^n \geq 0$, on peut voir que:

$$\Theta_h(T) = \sum_{n=1}^{N-1} C^n\left[u_h\right]^2(t_n) + \frac{1}{2}\left[u_h\right]^2(0) = \sum_{n=0}^{N-1} C^n\left[u_h\right]^2(t_n) \geq 0$$

avec $C^0 = \frac{1}{2}$.

En effet, en substituant $\hat{u}_h$ dans l'expression de $\Theta_h(T)$, on obtient:

- Pour $t^n = T$:

$$\hat{u}_h(T) = u_h(T^-)$$

donc,

$$(\hat{u}_h(T) - u_h(T^-))u_h(T^-) = (u_h(T^-) - u_h(T^-))u_h(T^-) = 0$$

- Pour $t_n = 0$:

$$\hat{u}_h(0) = u_0$$

donc,

$$(\hat{u}_h(0) - u_0)u_h(0^+) = (u_0 - u_0)u_h(0^+) = 0$$

- Pour $t_n \in (0, T)$:

$$\hat{u}_h(t_n) = (\{u_h\} + C^n [u_h])(t_n)$$

Alors,

$$(\hat{u}_h - \{u_h\})(t_n) = (\{u_h\} + C^n [u_h] - \{u_h\})(t_n) = C^n [u_h](t_n)$$

Et donc,

$$(\hat{u}_h - \{u_h\}) [u_h](t_n) = C^n [u_h]^2 (t_n)$$

Ainsi on a bien:

$$\Theta_h(T) = 0 + \sum_{n=1}^{N-1} C^n [u_h]^2 (t_n) - 0 + \frac{1}{2} [u_h]^2 (0) = \sum_{n=0}^{N-1} C^n [u_h]^2 (t_n) \geq 0$$

Le choix des C^n reste important pour pour une meilleure de convergnece de la méthode DG. Avec quelques tests numériques suivant les valeurs de C^n, on va voir que la solution est proches de la solution exacte selon les valeurs de C^n.

En effet, il est possible de définir la trace numérique $\hat{u}_h$ afin de garantir la stabilité de la méthode. On note que le choix de $C^n = \frac{1}{2}$ correspond à la trace numérique choisie initialement **(3.6)**, soit $\hat{u}_h(t) = \hat{u}_h(t^-)$.

Les méthodes DG qui en résulte sont non seulement stables mais aussi cohérentes pour toutes les valeurs de $C^n \geq 0$, car la condition $\hat{u} = u$ est satisfaite.

De plus, en cherchant à assurer la stabilité, il a été trouvé naturellement que la trace numérique $\hat{u}_h$ ne peut dépendre que des deux traces de $\hat{u}$ à t, plus précisement de $\hat{u}(t^-)$ et $\hat{u}(t^+)$.

Parlons maintenant de l'importance du choix des coefficients C^n sur la precision de la méthode. Il a été demontré que si $C^n = \frac{1}{2}$, l'ordre de la méthode aux points t_n est de $2k + 1$. Cependant, si $C^n = 0$, l'ordre est de $2k + 2$ [1]. Il est important de noter dans ce cas, on perda la possibilité de résoudre le flux $\hat{u_h}$ intervalle par intervalle, car si $C^n \neq \frac{1}{2}$, il devient nécessaire de résoudre $\hat{u_h}$ sur l'ensemble du domaine de calcul $(0, T)$ en une seule fois.

Par conséquent, si on souhaite une méthode DG qui soit cohérente, stable et capable de traiter la solution approximative intervalle par intervalle, il est nécessaire de fixer $C^n = \frac{1}{2}$.

Pour finir, on souhaite souligner trois propriétés essentielles des méthodes DG qui s'appliquent également aux cas multidimensionnels et à divers types de problèmes. Premièrement, la solution approchée des méthodes DG n'est pas contrainte par la continuité entre les éléments. Cela permet à la méthode d'être hautement parallélisable, particulièrement pour les problèmes hyperboliques dépendant du temps, et facilite l'utilisation de différentes approximations dans chaque élément, ce qui est idéal pour l'adaptativité hp.

Deuxièmement, les méthodes DG sont localement conservatrices. Cette propriété découle du fait que la méthode applique l'équation élément par élément et utilise une trace numérique. Dans notre cadre simple, cette propriété se lit comme suit:

$$\hat{u}|_{t_n}^{t_{n+1}} = \int_{I^n} f(s)u_h(s)ds,$$

et est est obtenu en posant $v = 1$ dans l'équation **(3.5)**.

Enfin, la troisièmement qui n'a pas été abordée suffisamment dans la litttérature disponible sur les méthodes DG, concerne la relation étroite entre les résidus de u_h à l'intérieur des intervalles et ses discontinuités aux interfaces. Pour mettre en évidence cette relation, on intégre par partie le premier terme de l'équation **(3.5)** afin d'obtenir:

$$\int_{I^n} u_h(s)\frac{d}{ds}v(s)ds - u_h v|_{t_n}^{t_{n+1}} + \hat{u}_h v|_{t_n}^{t_{n+1}} = \int_{I^n} f(s)u_h(s)v(s)ds$$

$\Longrightarrow$

$$\int_{I^n} v(s)\frac{d}{ds}u_h(s)ds - \int_{I^n} f(s)u_h(s)v(s)ds = (u_h - \hat{u}_h)v|_{t_n}^{t_{n+1}}$$

$\Longrightarrow$

$$\int_{I^n} R_h(s)v(s)ds = (u_h - \hat{u}_h)v|_{t_n}^{t_{n+1}}$$

où $R_h = (\frac{d}{ds}u_h - f(s)u_h)$ est le résidu de u_h sur l'intervalle I^n. Si on prends $v = 1$ et on utilise la definition de la trace numérique $\hat{u}_h$, on obtient:

$$\int_{I^n} R_h(s)ds = [\![u_h]\!](t_n)$$

En d'autres termes, l'intégrale du résidu de u_h sur l'intervalle I^n est égale au saut de u_h, noté $[\![u_h]\!]$, en t_n. Cela signifie que si l'EDO a été bien approximée dans l'intervalle I^n, alors le saut $[\![u_h]\!]$ sera trés faible. En revanche, si ce n'est pas le cas, c'est-à-dire si l'EDO n'a pas été bien approximée dans cette intervalle, le saut $[\![u_h]\!]$ sera grand, et la méthode DG deviendra plus dissipative, comme on peut bien le constater directement à partir de l'identité de stabilité:

$$\frac{1}{2}u_h^2(T^-) - \frac{1}{2}\sum_{n=0}^{N-1}[\![u_h]\!]^2(t_n) - \frac{1}{2}u_0^2 = \int_0^T f(s)u_h^2(s)ds$$

Pour simplifier, on peut dire que les discontinuités $[\![u_h]\!]$ fonctionnent comme des amortisseurs, stabilisant la méthode DG lorsque l'intégration devient difficile et que l'EDO n'est pas correctement approchée. Cela assure une grande stabilité à la méthode tout en maintenant sa précision. En passant aux dimensions supérieures, le résidu devient une fonction linéaire plus complexe des discontinuités, mais le mécanisme dissipatif décrit reste principalement le même. Ce mécanisme est aussi présent dans les méthodes d'éléments finis stabilisées, couramment utilisées pour les problèmes de convection et de second ordre.

3.1.3 Assemblages locales des matrices

On discrétise l'intervalle $(0, T)$ en $N - 1$ intervalles I^n:

$$0 = t_0 < t_1 < \ldots < t_{N-1} = T$$

où les elements sont donnés par:

$$I^n = (t_n, t_{n+1}), \quad n \in [0, N-2]$$

On a alors:

$$(0, T) = \bigcup_{n=0}^{N-2} I^n$$

L'equation **(3.5)** peut se réecrire comme suit:

$$\int_{I^n} u_h(s)\frac{d}{ds}v(s)ds + \int_{I^n} f(s)u_h(s)v(s)ds - \hat{u}_h v|_{t_n}^{t_{n+1}} = 0 \tag{3.13}$$

En substituant **(3.4)** dans **(3.13)**, on obtient:

$$\int_{I^n}\left(\sum_{i=0}^{p} u_i^n\phi_i^n(s)\right)\frac{d}{ds}\phi_j^n(s)ds + \int_{I^n} f(s)\left(\sum_{i=0}^{p} u_i^n\phi_i^n(s)\right)\phi_j^n(s)ds - \hat{u}_h v|_{t_n}^{t_{n+1}} = 0$$

qu'on peut réecrire encore:

$$\sum_{i=0}^{p}\int_{I^n} u_i^n\left(\frac{d}{ds}\phi_j^n(s) + f(s)\phi_j^n(s)\right)\phi_i^n(s)ds + \hat{u}_h(t_n)\phi_j^n(t_n) - \hat{u}_h(t_{n+1})\phi_j^n(t_{n+1}) = 0 \tag{3.14}$$

Sur chaque intervalle de la partition, on on approche "localement" la solution:

- **Sur le premier intervalle $I^0 = (t_0, t_1)$:**

$$u_h = \sum_{i=0}^{p} u_i^0 \phi_i^0, \quad v = \phi_j^0$$

l'equation **(3.14)** devient:

$$\sum_{i=0}^{p} u_i^0 \int_{I^0} u_i^0 \left(f(s)\phi_j^0(s) + \frac{d}{ds}\phi_j^0(s) \right) ds + \hat{u}_h(t_0)\phi_j^0(t_0) - \hat{u}_h(t_1)\phi_j^0(t_1) = 0 \tag{3.15}$$

or d'après **(3.12)**, on a:

$$\hat{u}_h(t_0) = u_0$$

et

$$\hat{u}_h(t_1)\phi_j^0(t_1^-) = \left(\{u_h\} + C^1 [u_h] \right) \phi_j^0(t_1^-)$$

alors,

$$\sum_{i=0}^{p} u_i^0 \int_{I^0} u_i^0 \left(f(s)\phi_j^0(s) + \frac{d}{ds}\phi_j^0(s) \right) ds + u_0\phi_j^0(t_0) - \hat{u}_h(t_1)\phi_j^0(t_1) = 0$$

Si on aborde les notations dans **(3.10)**, on obtient:

$$\begin{aligned}
-\hat{u}_h(t_1)\phi_j^0(t_1^-) &= -\left(\{u_h\} + C^1 [u_h] \right) \phi_j^0(t_1^-) \\
&= -\left(\frac{1}{2}(u_h(t_1^-) + u_h(t_1^+)) + C^1(u_h(t_1^-) - u_h(t_1^+)) \right) \phi_j^0(t_1^-) \\
&= \left[(-\frac{1}{2} - C^1)u_h(t_1^-) + (-\frac{1}{2} + C^1)u_h(t_1^+) \right] \phi_j^0(t_1^-) \\
&= \left[(C^1 - \frac{1}{2})u_h(t_1^+) + (-\frac{1}{2} - C^1)u_h(t_1^-) \right] \phi_j^0(t_1^-)
\end{aligned}$$

Si on remplace u_h par son expression, on obtient:

$$-\hat{u}_h(t_1)\phi_j^0(t_1^-) = (C^1 - \frac{1}{2}) \sum_{i=0}^{p} \mathbf{u_i^1} \phi_i^1(t_1^+)\phi_j^0(t_1^-) + (-\frac{1}{2} - C^1) \sum_{i=0}^{p} \mathbf{u_i^0} \phi_i^0(t_1^-)\phi_j^0(t_1^-)$$

Par suite, on peut reecrire l'équation **(3.15)** sur l'intervalle I^0 comme suit:

$$\begin{aligned}
\sum_{i=0}^{p} \left\{ \int_{I^0} \left(f(s)\phi_i^0(s) + \frac{d}{ds}\phi_i^0(s) \right) \phi_j^0(s) ds \right\} \mathbf{u_i^0} + \left(-\frac{1}{2} - C^1 \right) \sum_{i=0}^{p} \phi_i^0(t_1^-)\phi_j^0(t_1^-)\mathbf{u_i^0} \\
+ \left(C^1 - \frac{1}{2} \right) \sum_{i=0}^{p} \phi_i^1(t_1^+)\phi_j^0(t_1^-)\mathbf{u_i^1} = -u_0\phi_j^0(t_0^+)
\end{aligned}$$

$\Longrightarrow$

$$\boxed{\sum_{i=0}^{p} \left\{ \int_{I^0} \left(f(s)\phi_i^0(s) + \frac{d}{ds}\phi_i^0(s) \right) \phi_j^0(s) ds + \left(-\frac{1}{2} - C^1 \right) \phi_i^0(t_1^-)\phi_j^0(t_1^-) \right\} \mathbf{u_i^0}}$$

$$+ \boxed{\left(C^1 - \frac{1}{2} \right) \sum_{i=0}^{p} \phi_i^1(t_1^+)\phi_j^0(t_1^-)\mathbf{u_i^1}}$$

$$= \boxed{-u_0\phi_j^0(t_0^+)}$$

Sous forme matricielle, on obtient:

$$\begin{bmatrix} Q_1 & Q_2 & 0 & \dots & 0 \\ 0 & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u^0} \\ \mathbf{u^1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ 0 \\ \vdots \end{bmatrix}$$

où

$$(Q_1)_{ij} = \left\{ \int_{I^0} \left(f(s)\phi_i^0(s) + \frac{d}{ds}\phi_i^0(s) \right) \phi_j^0(s) ds \right\} + \left(-\frac{1}{2} - C^1 \right) \phi_i^0(t_1^-)\phi_j^0(t_1^-)$$

$$(Q_2)_{ij} = \left(C^1 - \frac{1}{2} \right) \phi_i^1(t_1^+)\phi_j^0(t_1^-)$$

et

$$(\mathbf{b})_j = -u_0\phi_j^0(t_0^+)$$

Remarque: Il faudra noter que $Q1$ et $Q2$ sont des blocs de matrices $(p \times p)$, et $\mathbf{b}$ est un vecteur de taille p et que U^0 et U^1 sont aussides vecteurs de taille p.

- **Sur les intervalles interieurs $I^n = (t_n, t_{n+1})$ avec $n \in [1, N-3]$:**

$$u_h = \sum_{i=0}^{p} u_i^n \phi_i^n, \quad v = \phi_j^n$$

L'équation **(3.14)** devient:

$$\sum_{i=0}^{p} u_i^n \int_{I^n} \left(f(s)\phi_j^n(s) + \frac{d}{ds}\phi_j^n(s) \right) \phi_i^n(s) ds + \hat{u}_h(t_n)\phi_j^n(t_n) - \hat{u}_h(t_{n+1})\phi_j^n(t_{n+1}) = 0 \tag{3.16}$$

or, toujours d'après **(3.12)**, on a:

$$\begin{aligned} \hat{u}_h(t_n) &= \{u_h\} + C^n [u_h] \\ &= \frac{1}{2}(u_h(t_n^-) + u_h(t_n^+)) + C^n(u_h(t_n^-) - u_h(t_n^+)) \\ &= (\frac{1}{2} + C^n)u_h(t_n^-) + (\frac{1}{2} - C^n)u_h(t_n^+) \end{aligned}$$

Donc,

$$\begin{aligned} \hat{u}_h(t_n)\phi_j^n(t_n) &= \left[(\frac{1}{2} + C^n)u_h(t_n^-) + (\frac{1}{2} - C^n)u_h(t_n^+) \right] \phi_j^n(t_n^+) \\ &= \left[(\frac{1}{2} + C^n) \sum_{i=0}^{p} \mathbf{u_i^{n-1}} \phi_i^{n-1}(t_n^-) + (\frac{1}{2} - C^n) \sum_{i=0}^{p} \mathbf{u_i^n} \phi_i^n(t_n^+) \right] \phi_j^n(t_n^+) \\ &= (\frac{1}{2} + C^n) \sum_{i=0}^{p} \mathbf{u_i^{n-1}} \phi_i^{n-1}(t_n^-)\phi_j^n(t_n^+) + (\frac{1}{2} - C^n) \sum_{i=0}^{p} \mathbf{u_i^n} \phi_i^n(t_n^+)\phi_j^n(t_n^+) \end{aligned}$$

et de même,

$$\begin{aligned} \hat{u}_h(t_{n+1})\phi_j^n(t_{n+1}) &= \left[(\frac{1}{2} + C^{n+1})u_h(t_{n+1}^-) + (\frac{1}{2} - C^{n+1})u_h(t_{n+1}^+) \right] \phi_j^n(t_{n+1}^+) \\ &= (\frac{1}{2} + C^{n+1}) \sum_{i=0}^{p} \mathbf{u_i^n} \phi_i^n(t_{n+1}^-)\phi_j^n(t_{n+1}^+) + (\frac{1}{2} - C^{n+1}) \sum_{i=0}^{p} \mathbf{u_i^{n+1}} \phi_i^{n+1}(t_{n+1}^+)\phi_j^n(t_{n+1}^+) \end{aligned}$$

En substituant ces deux expressions dans l'équation **(3.16)**, on obtient:

$$\boxed{\sum_{i=0}^{p}\left\{\left(\frac{1}{2}+C^n\right)\phi_i^{n-1}(t_n^-)\phi_j^n(t_n^+)\right\}\mathbf{u_i^{n-1}}}+$$

$$\boxed{\sum_{i=0}^{p}\left\{\int_{I^n}\phi_i^n(s)\left(f(s)\phi_j^n(s)+\frac{d}{ds}\phi_j^n(s)\right)ds+\left(\frac{1}{2}-C^n\right)\phi_i^n(t_n^+)\phi_j^n(t_n^+)-\left(\frac{1}{2}+C^{n+1}\right)\phi_i^n(t_{n+1}^-)\phi_j^{n+1}(t_{n+1}^+)\right\}\mathbf{u_i^n}}$$

$$+\boxed{\sum_{i=0}^{p}\left\{-\left(\frac{1}{2}-C^{n+1}\right)\phi_i^{n+1}(t_{n+1}^+)\phi_j^{n+1}(t_{n+1}^+)\right\}\mathbf{u_i^{n+1}}}=0$$

Sous forme matricielle, on obtient:

$$\begin{bmatrix}0&\cdots&\cdots&\cdots&\cdots&\cdots&0\\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\ 0&\cdots&Q1&Q2&Q3&\cdots&0\\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\ 0&\cdots&\cdots&\cdots&\cdots&\cdots&0\end{bmatrix}\begin{bmatrix}\vdots\\ \mathbf{u^{n-1}}\\ \mathbf{u^n}\\ \mathbf{u^{n+1}}\\ \vdots\end{bmatrix}=\begin{bmatrix}\vdots\\0\\0\\0\\\vdots\end{bmatrix}$$

Avec:

$$(Q_1)_{ij}=\left\{\left(\frac{1}{2}+C^n\right)\phi_i^{n-1}(t_n^-)\phi_j^n(t_n^+)\right\}$$

$$(Q_2)_{ij}=\left\{\int_{I^n}\phi_i^n(s)\left(f(s)\phi_j^n(s)+\frac{d}{ds}\phi_j^n(s)\right)ds+\left(\frac{1}{2}-C^n\right)\phi_i^n(t_n^+)\phi_j^n(t_n^+)-\left(\frac{1}{2}+C^{n+1}\right)\phi_i^n(t_{n+1}^-)\phi_j^{n+1}(t_{n+1}^+)\right\}$$

$$(Q_3)_{ij}=-\left(\frac{1}{2}-C^{n+1}\right)\phi_i^{n+1}(t_{n+1}^+)\phi_j^{n+1}(t_{n+1}^+)$$

Remarque: Même remarque que pour l'assemblage sur l'intervalle I^0: $Q1$, $Q2$ et $Q3$ sont des blocs de matrices $(p\times p)$, et les solutions U^{n-1}, U^n et U^{n+1} sont des vecteurs de taille p.

- **Sur le dernier intervalle $I^{N-2}=(t_{N-2},t_{N-1})=(t_{N-2},T)$:**

$$u_h=\sum_{i=0}^{p}u_i^{N-2}\phi_i^{N-2},\quad v=\phi_j^{N-2}$$

L'équation **(3.14)** devient:

$$\sum_{i=0}^{p}\mathbf{u_i^{N-2}}\int_{I^{N-2}}\left(f(s)\phi_j^{N-2}(s)+\frac{d}{ds}\phi_j^{N-2}(s)\right)\phi_i^{N-2}(s)ds+\hat{u}_h(t_{N-2})\phi_j^{N-2}(t_{N-2})-\hat{u}_h(T)\phi_j^{N-2}(T)=0\quad(3.17)$$

or

$$\hat{u}_h(T)=u_h(T^-)\quad\text{d'apres }\mathbf{(3.12)}$$

donc,

$$\hat{u}_h(T)\phi_j^{N-2}(T)=\sum_{i=0}^{p}\mathbf{u_i^{N-2}}\phi_i^{N-2}(T^-)\phi_j^{N-2}(T^-)$$

Par ailleurs,

$$\begin{aligned}\hat{u}_h(t^{N-2})&=\{u_h\}+C^{N-2}\,[u_h]\\&=\frac{1}{2}(u_h(t_{N-2}^-)+u_h(t_{N-2}^+))+C^{N-2}(u_h(t_{N-2}^-)-u_h(t_{N-2}^+))\end{aligned}$$

$$= (\frac{1}{2} + C^{N-2})u_h(t^-_{N-2}) + (\frac{1}{2} - C^{N-2})u_h(t^+_{N-2})$$

donc,

$$\begin{aligned}
\hat{u}_h(t_{N-2})\phi_j^{N-2}(t^+_{N-2}) &= \left[(\frac{1}{2} + C^{N-2})u_h(t^-_{N-2}) + (\frac{1}{2} - C^{N-2})u_h(t^+_{N-2})\right]\phi_j^{N-2}(t^+_{N-2}) \\
&= \left[(\frac{1}{2} + C^{N-2})\sum_{i=0}^{p}\mathbf{u_i^{N-3}}\phi_i^{N-3}(t^-_{N-2}) + (\frac{1}{2} - C^{N-2})\sum_{i=0}^{p}\mathbf{u_i^{N-2}}\phi_i^{N-2}(t^+_{N-2})\right]\phi_j^{N-2}(t^+_{N-2}) \\
&= (\frac{1}{2} + C^{N-2})\sum_{i=0}^{p}\mathbf{u_i^{N-3}}\phi_i^{N-3}(t^-_{N-2})\phi_j^{N-2}(t^+_{N-2}) + (\frac{1}{2} - C^{N-2})\sum_{i=0}^{p}\mathbf{u_i^{N-2}}\phi_i^{N-2}(t^+_{N-2})\phi_j^{N-2}(t^+_{N-2})
\end{aligned}$$

En remplaçant ces expressions dans l'équation **(3.17)**, on obtient:

$$\boxed{\sum_{i=0}^{p}\left\{\left(\frac{1}{2} + C^{N-2}\right)\phi_i^{N-3}(t_-^{N-2})\phi_j^{N-2}(t_+^{N-2})\right\}\mathbf{u_i^{N-3}}} +$$

$$\boxed{\sum_{i=0}^{p}\left\{\int_{I^{N-2}}\phi_i^{N-2}(s)\left(f(s)\phi_j^{N-2}(s) + \frac{d}{ds}\phi_j^{N-2}(s)\right)ds + \left(\frac{1}{2} - C^{N-2}\right)\phi_i^{N-2}(t_+^{N-2})\phi_j^{N-2}(t_+^{N-2}) - \phi_i^{N-2}(T^-)\phi_j^{N-2}(T^-)\right\}}$$

$= 0$

Sous forme matricielle, on obtient:

$$\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & Q1 & Q2 \end{bmatrix}\begin{bmatrix} \vdots \\ \\ \vdots \\ \mathbf{u^{N-3}} \\ \mathbf{u^{N-2}} \end{bmatrix}$$

Avec:

$$(Q1)_{ij} = \left\{\left(\frac{1}{2} + C^{N-2}\right)\phi_i^{N-3}(t_-^{N-2})\phi_j^{N-2}(t_+^{N-2})\right\}$$

$$\begin{aligned}
(Q2)_{ij} = &\int_{I^{N-2}}\phi_i^{N-2}(s)\left(f(s)\phi_j^{N-2}(s) + \frac{d}{ds}\phi_j^{N-2}(s)\right)ds \\
&+ \left(\frac{1}{2} - C^{N-2}\right)\phi_i^{N-2}(t_+^{N-2})\phi_j^{N-2}(t_+^{N-2}) \\
&- \phi_i^{N-2}(T^-)\phi_j^{N-2}(T^-)
\end{aligned}$$

3.1.4 Implementation du schema DG en 1D

Le schéma a été implémenté en Python et disponible à l'adresse suivante:
`https://gitlab.math.unistra.fr/aghili/dg/-/blob/dieng-main-patch-68158/dg1d.py?ref_type=heads`

Dans cette section, nous présentons une implémentation de la méthode de Galerkin discontinue (DG) en une dimension. Le code utilise des fonctions de base discontinues, la formulation faible, et des flux numériques pour résoudre l'EDO **(3.1)**. Voici les diffèrentes étapes de l'implémentation:

- **Etape 1:** On commence par initialiser la classe, définir les paramètres du problème et les fonctions de base.
- **Etape 2:** On construit les matrices locales $Q1$, $Q2$ et $Q3$ selon chaque intervalle I^n.
- **Etape 3:** On assemble les matrices locales pour obtenir la matrice globale A.
- **Etape 4:** On resoud le systeme lineaire $A\mathbf{u} = \mathbf{b}$ et estimer les erreurs.

3.1.5 Résultats Numériques

On reprends le problème (3.1) avec $f(t) = -t^2, \quad u_0 = 1 \quad$ et $\quad T = 5$.
Dans la figure (3.1), on affiche l'erreur $||u_h - u||^2_{L^2}$ en fonction du pas d'espace h allant de $h = \frac{1}{2^2}$ jusqu'à $h = \frac{1}{2^8}$ à échelles logarithmiques pour $k = 0, 1, ..., 5$ et pour $C = 0$.

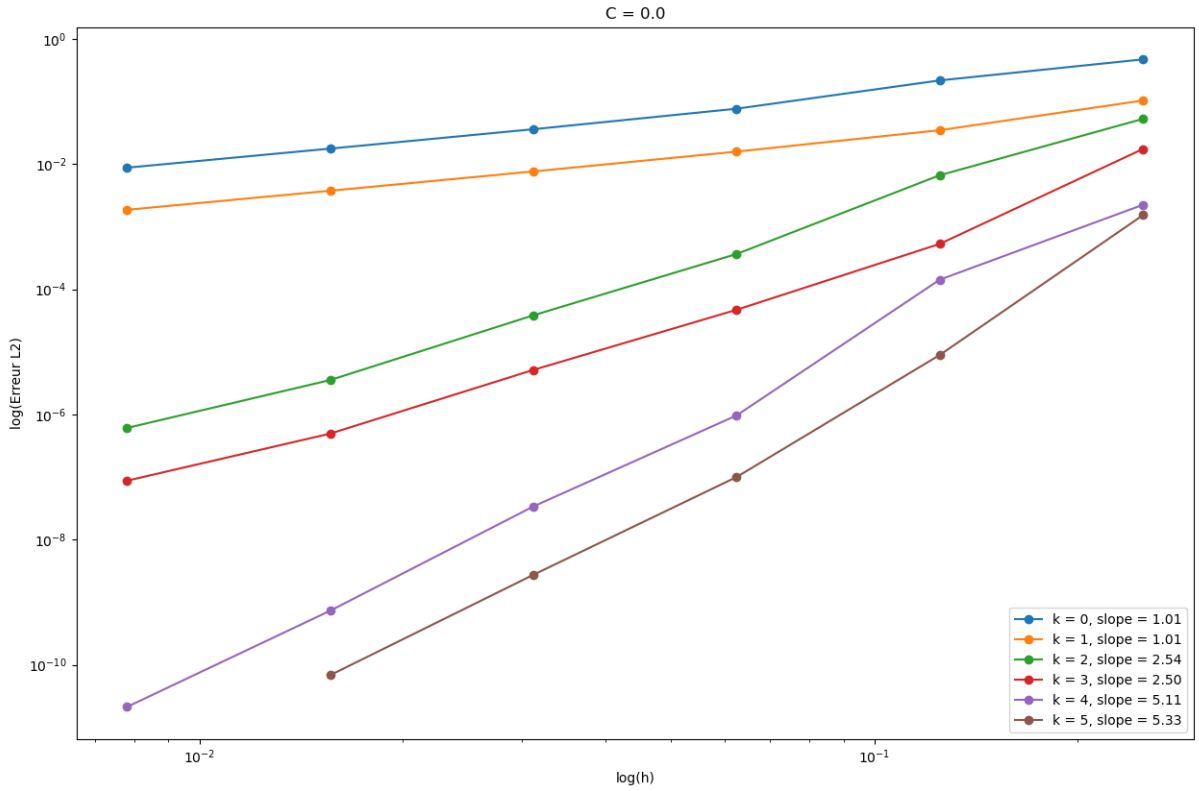

Figure 3.1: Erreurs et Pentes pour $C = 0$ pour chaque ordre polynomial k

De même dans la figure (3.2), on affiche les mêmes variations pour $C = 0.5$.
.

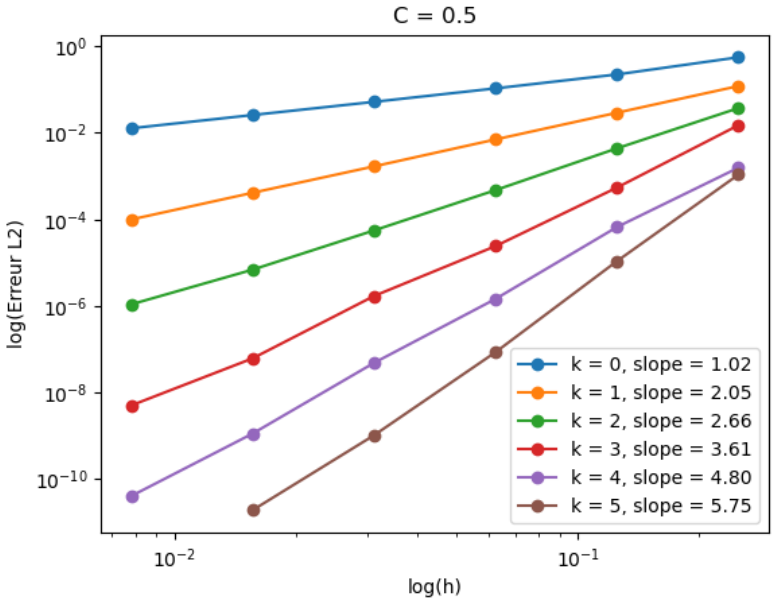

Figure 3.2: Erreurs et Pentes pour $C = 0.5$ pour chaque ordre polynomial k

Interpretations: Dans un premier temps, on peut remarquer que les ordres d'erreurs diminuent progressivement lorsque le degré k des polynômes de base augmente, ce qui confirme que les méthodes DG sont efficaces pour des polynômes de degré élevé.
Ensuite, on peut constater que le choix de $C = 0.5$ conduit à des résultats **sous-optimaux**, dans le sens où les erreurs sont plus petites et les ordres d'erreurs se rapprochent de $2k + 2$. Cette sous-optimalité peut être justifiée par le choix (non optimal) des fonctions de base, qui pourrait être amélioré pour atteindre une meilleure précision.

3.2 L'équation de transport 2D

3.2.1 Formulation faible du problème

Dans cette cette section on considere les méthodes de DG pour la resolution de l'équation de transport 2D:

$$\begin{cases} \frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{a}u) = 0, & \text{dans } \Omega \times (0,T) \\ u(t=0) = u_0, & \text{dans } \Omega \end{cases} \tag{3.18}$$

où $\mathbf{a} = (a_x, a_y)$ est un vecteur fixé, $\Omega = [0,1] \times [0,1]$ et

$$\nabla = \frac{\partial}{\partial x}\hat{\mathbf{i}} + \frac{\partial}{\partial y}\hat{\mathbf{j}}$$

avec $\hat{\mathbf{i}}$ et $\hat{\mathbf{j}}$ les vecteurs unitaires dans les directions x et y respectivement.

On se concentre sur trois propriétés clés de cette approche:

Premièrement, les méthodes DG sont étroitement liées aux méthodes classiques de volumes finis.

Deuxièmement, l'utilisation de polynômes de haut degré permet aux méthodes DG d'atteindre une grande précision tout en restant hautement parallélisables.

Troisièmement, la viscosité artificielle de la méthode est déterminée par l'amplitude des discontinuités, qui sont elles-mêmes liées aux résidus à l'intérieur des éléments.

Ainsi, lorsque le degré polynomial de la solution approchée augmente, la viscosité artificielle diminue, même en présence de discontinuités.

Pour discrétiser l'équation de transport dans l'espace à l'aide d'une méthode DG, on commence par subdiviser le domaine Ω en un maillage triangulaire $\mathcal{T}_h$. On cherche ensuite une solution approchée discontinue u_h, qui appartient à un espace $V(K)$ pour chaque élément K du maillage. Cet espace $V(K)$ n'est pas contraint, bien que le choix le plus courant soit l'espace des polynômes de degré au plus k, noté $P_k(K)$.

Comme cas la secton précédente, on commence par une formulation faible donnée par:

$$\int_K (u_h)_t v + \int_K \nabla \cdot (\mathbf{a} u_h) v = 0 \tag{3.19}$$

où v est une fonction test appartenant à un espace de fonctions $V(K)$.

Par une integration par partie de l'équation ci-desus, on obtient:

$$\int_K (u_h)_t v - \int_K \mathbf{a} u_h \cdot \nabla v + \int_{\partial K} \widehat{\mathbf{a} u_h} \cdot \hat{\mathbf{n}}^k v = 0 \tag{3.20}$$

où $\hat{\mathbf{n}}^k = (\hat{n}_x, \hat{n}_y)$ est la normale unitaire sortante de la frontière ∂_k de l'element K. En 2D, le rôle des vecteurs normaux entre en jeu. Les différents vecteurs normaux d'un élément triangulaire ainsi que d'un de ses éléments voisins, comme illustré dans la **figure 3.4**.

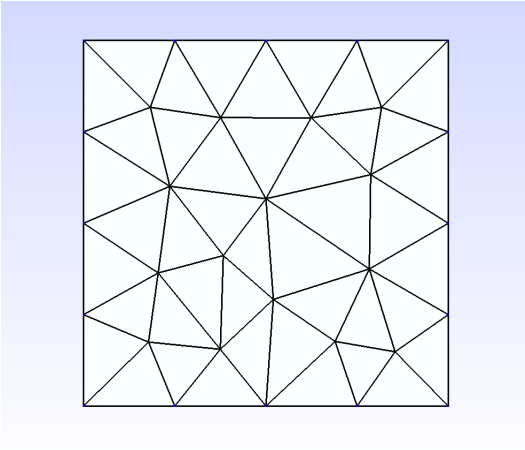

Figure 3.3: Maillage triangulaire du domaine Ω

Pour completer la definition de la méthode DG, il est necessaire de definir le flux numérique $\widehat{\mathbf{a} u_h}$ sur les bords de chaque element K. Pour cela, on procède comme dans la section précédente en obtenant un résultat de stabilité.

Pour ce faire, on multiplie par u et on intègre sur l'espace et le temps pour obtenir:

$$\frac{1}{2}\int_{\mathbf{R}^2} u^2 + \frac{1}{2}\int_0^T \int_{\mathbf{R}^2} \nabla \cdot \mathbf{a} u^2 = \frac{1}{2}\int_{\mathbf{R}^2} u_0^2$$

Maintenant, imitons la procédure précédente pour la méthode DG considèrée. En prenant $v = u_h$ dans l'équation **(3.20)**, et en additionnant sur les éléments K, on obtient:

$$\frac{1}{2}\int_K u_h^2(x,y,T) + \frac{1}{2} int_0^T \int_K \nabla \cdot \mathbf{a} u_h^2(x,y,t) dx dy dt \int_0^T \Theta_h(t) dt = \frac{1}{2}\int_K u_0^2(x,y) dx dy$$

où

$$\Theta_h(t) = \sum_{K \in \mathcal{T}_h} \left(-\frac{1}{2}\int_K \nabla \cdot (\mathbf{a} u_h)(x,y,t) dx dy + \int_{\partial K} \widehat{\mathbf{a} u_h} \cdot \mathbf{n} u_h(x,y,t) ds \right)$$

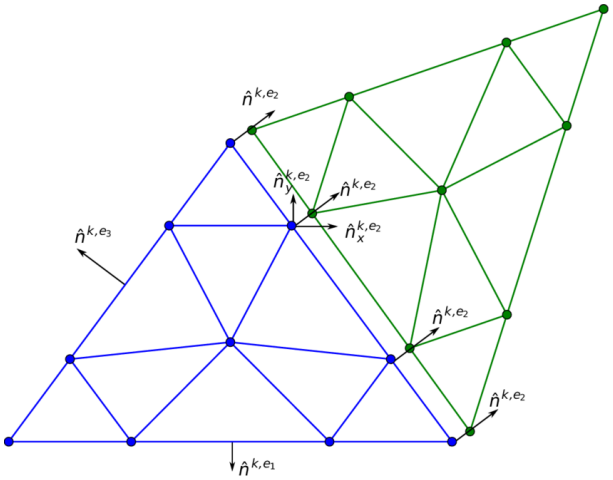

Figure 3.4: Vecteurs normaux du deuxième bord (e2) du **K**-ième élément (bleu) avec son élément voisin (vert) [1]

Ensuite, on definit le flux numérique $\widehat{\mathbf{a}u_h}$ de sorte que cette quantité $\Theta_h(t)$ soit positive.

Considèrons un point (x, y) situé sur l'ensemble $e = \overline{\partial K^+} \cap \overline{\partial K^-}$. Soient n^+ et n^- les vecteurs normaux sortants de K^+ et K^- respectivement en (x, y). La valeur de $u_h^{\pm}(x, y)$ est définie comme $\lim_{\epsilon \to 0} u_h(x - \epsilon n^+, y - \epsilon n^+)$.
On definit le flux comme suit:

$$\widehat{\mathbf{a}u_h^n} = \mathbf{a}\{u_h\} + C[\![u_h]\!] \quad avec \quad [\![u_h]\!] = u_h^- \mathbf{n}^- + u_h^+ \mathbf{n}^+ \quad et \quad \{u_h\} = \frac{1}{2}(u_h^+ + u_h^-)$$

Enfin soit $\mathcal{E}_h$ l'ensemble des ensembles $e = \overline{\partial K^+} \cap \overline{\partial K^-}$ pour tout K^+ et K^- dans $\mathcal{T}_h$. On peut alors écrire:

$$\begin{aligned}
\Theta_h(t) &= \sum_{K \in \mathcal{T}_h} \left(-\frac{1}{2} \int_K \nabla \cdot (\mathbf{a}u_h)(x, y, t) dx dy + \int_{\partial K} \widehat{\mathbf{a}u_h} \cdot \mathbf{n} u_h(x, y, t) ds \right) \\
&= \sum_{K \in \mathcal{T}_h} \int_{\partial K} \left(\widehat{\mathbf{a}u_h} \cdot \mathbf{n} u_h - \frac{1}{2} \mathbf{a} u_h^2 \cdot \mathbf{n} \right) ds \\
&= \sum_{e \in \mathcal{E}_h} \int_e [\![\widehat{\mathbf{a}u_h} u_h - \frac{1}{2} \mathbf{a} u_h^2]\!] ds \\
&= \sum_{e \in \mathcal{E}_h} \int_e \left(\widehat{\mathbf{a}u_h} [\![u_h]\!] - \frac{1}{2} [\![\mathbf{a} u_h^2]\!] \right) ds
\end{aligned}$$

Comme

$$\frac{1}{2}[\![u_h^2]\!] = [\![u_h]\!]\{u_h\} \quad \text{et} \quad [\![\mathbf{a}u_h^2]\!] = [\![u_h]\!]\{\mathbf{a}u_h\}$$

alors, on obtient:

$$\begin{aligned}
\Theta_h(t) &= \sum_{e \in \mathcal{E}_h} \int_e \left(\widehat{\mathbf{a}u_h} [\![u_h]\!] - \frac{1}{2} [\![\mathbf{a} u_h^2]\!] \right) ds \\
&= \sum_{e \in \mathcal{E}_h} \int_e \left(\widehat{\mathbf{a}u_h} [\![u_h]\!] - [\![u_h]\!]\{\mathbf{a}u_h\} \right) ds \\
&= \sum_{e \in \mathcal{E}_h} \int_e \left(\widehat{\mathbf{a}u_h} - \mathbf{a}\{u_h\} \right) [\![u_h]\!] ds
\end{aligned}$$

En prenant

$$\widehat{\mathbf{a}u_h} = \mathbf{a}\{u_h\} + C[\![u_h]\!] \tag{3.21}$$

on obtient que

$$\Theta_h(t) = \sum_{e\in\mathcal{E}_h} \int_e C[\![u_h]\!]^2 ds \geq 0$$

pour $C \geq 0$.
Ainsi, le choix du flux numérique dans **(3.21)** garantit la stabilité et la consistance de la méthode DG.

3.2.2 Assemblage des matrices locales

On cherche une solution approchée $u_h^n(x,y) \approx u(x,y,t^n)$ telle que restreinte à l'element K, elle s'ecrit:

$$u_h^n(x,y)\Big|_K = \alpha^n + \beta^n x + \gamma^n y$$

On fait l'approximation en temps suivante:

$$(u_h)_t = \frac{u_h^{n+1} - u_h^n}{\delta t}$$

On reecrit alors l'équation **(3.20)** comme suit:

$$\int_K \frac{u_h^{n+1} - u_h^n}{\delta t} v - \int_K \mathbf{a} u_h^n \cdot \nabla v + \int_{\partial K} \widehat{\mathbf{a} u_h^n} \cdot \hat{\mathbf{n}}^k v = 0 \quad \forall v \in V = P_k(K)$$

Pour tout $v \in V = P_k(K)$, on a:

$$\int_K \frac{(\alpha^{n+1} - \alpha^n) + (\beta^{n+1} - \beta^n)x + (\gamma^{n+1} - \gamma^n)y}{\delta t} \cdot v - \int_K \mathbf{a}\left(\alpha^n + \beta^n x + \gamma^n y\right) \cdot \nabla v + \int_{\partial K} \widehat{\mathbf{a} u_h^n} \cdot \hat{\mathbf{n}}^k \cdot v = 0 \quad (3.22)$$

Ce qui equivaut à:

$$\begin{aligned}
\int_K \frac{(\alpha^{n+1} - \alpha^n) + (\beta^{n+1} - \beta^n)x + (\gamma^{n+1} - \gamma^n)y}{\delta t} \cdot v & \\
- \int_K a_x \left(\alpha^n + \beta^n x + \gamma^n y\right) \frac{\partial v}{\partial x} & \\
- \int_K a_y \left(\alpha^n + \beta^n x + \gamma^n y\right) \frac{\partial v}{\partial y} & \\
+ \sum_{Arete\in\partial K} \int_{Arete} \left(\widehat{\mathbf{a} u_h^n} \cdot \hat{\mathbf{n}}^k\right) \cdot v = 0 &
\end{aligned}$$

Supposons un element K avec les aretes e_1, e_2, e_3 qui sont en commun avec les elements voisins L,M, et N respectivement et posons $\mathbf{n} = \hat{\mathbf{n}}^k$ pour simplifier la notation. On a alors:

$$\begin{aligned}
\int_{e_1} \left(\widehat{\mathbf{a} u_h^n}\right) \cdot \mathbf{n} \cdot v &= \int_{e_1} \left(\mathbf{a}\{u_h\} + C[\![u_h]\!]\right) \cdot \mathbf{n} \cdot v \\
&= \int_{e_1} \left(\mathbf{a}\{u_h\}\right) \cdot \mathbf{n} \cdot v + C \int_{e_1} [\![u_h]\!] \cdot \mathbf{n} \cdot v \\
&= \int_{e_1} \tfrac{1}{2} \mathbf{a} \left(u_h^+ + u_h^-\right) \cdot \mathbf{n} \cdot v + C \int_{e_1} \left(u_h^- \mathbf{n}^- + u_h^+ \mathbf{n}^+\right) \cdot \mathbf{n} \cdot v \\
&= \int_{e_1} \tfrac{1}{2} \mathbf{a} \left(u_h^n\Big|_K + u_h^n\Big|_L\right) \cdot \mathbf{n} \cdot v + C \int_{e_1} \left(u_h^n\Big|_K \mathbf{n}^- + u_h^n\Big|_L \mathbf{n}^+\right) \cdot \mathbf{n} \cdot v \\
&= \int_{e_1} \tfrac{1}{2} \mathbf{a} \left((\alpha^{K,n} + \beta^{K,n} x + \gamma^{K,n} y) + (\alpha^{L,n} + \beta^{L,n} x + \gamma^{L,n} y)\right) \cdot \mathbf{n} \cdot v \\
&\quad + C \int_{e_1} \left((\alpha^{K,n} + \beta^{K,n} x + \gamma^{K,n} y)\mathbf{n}^- + (\alpha^{L,n} + \beta^{L,n} x + \gamma^{L,n} y)\mathbf{n}^+\right) \cdot \mathbf{n} \cdot v \\
&= \int_{e_1} \tfrac{1}{2} \mathbf{a} \left((\alpha^{K,n} + \alpha^{L,n}) + (\beta^{K,n} + \beta^{L,n})x + (\gamma^{K,n} + \gamma^{L,n})y\right) \cdot \mathbf{n} \cdot v \\
&\quad + C \int_{e_1} \left((\alpha^{K,n}\mathbf{n}^- + \alpha^{L,n}\mathbf{n}^+) + (\beta^{K,n}\mathbf{n}^- + \beta^{L,n}\mathbf{n}^+)x + (\gamma^{K,n}\mathbf{n}^- + \gamma^{L,n}\mathbf{n}^+)y\right) \cdot \mathbf{n} \cdot v
\end{aligned}$$

Par analogie sur les aretes e_2 (commune avec M) et e_3(commune avec N), on obtient respectivement:

$$\begin{aligned}\int_{e_2}\left(\widehat{\mathbf{a}u_h^n}\right)\cdot\mathbf{n}\cdot v &= \int_{e_2}\tfrac{1}{2}\mathbf{a}\left((\alpha^{K,n}+\alpha^{M,n})+(\beta^{K,n}+\beta^{M,n})x+(\gamma^{K,n}+\gamma^{M,n})y\right)\cdot\mathbf{n}\cdot v\\ &+C\int_{e_2}\left((\alpha^{K,n}\mathbf{n}^-+\alpha^{M,n}\mathbf{n}^+)+(\beta^{K,n}\mathbf{n}^-+\beta^{M,n}\mathbf{n}^+)x+(\gamma^{K,n}\mathbf{n}^-+\gamma^{M,n}\mathbf{n}^+)y\right)\cdot\mathbf{n}\cdot v\end{aligned}$$

$$\begin{aligned}\int_{e_3}\left(\widehat{\mathbf{a}u_h^n}\right)\cdot\mathbf{n}\cdot v &= \int_{e_3}\tfrac{1}{2}\mathbf{a}\left((\alpha^{K,n}+\alpha^{N,n})+(\beta^{K,n}+\beta^{N,n})x+(\gamma^{K,n}+\gamma^{N,n})y\right)\cdot\mathbf{n}\cdot v\\ &+C\int_{e_3}\left((\alpha^{K,n}\mathbf{n}^-+\alpha^{N,n}\mathbf{n}^+)+(\beta^{K,n}\mathbf{n}^-+\beta^{N,n}\mathbf{n}^+)x+(\gamma^{K,n}\mathbf{n}^-+\gamma^{N,n}\mathbf{n}^+)y\right)\cdot\mathbf{n}\cdot v\end{aligned}$$

3.2.3 Assemblage d'un systeme lineaire:

Nous considérons une solution approchée $u_h^n(x,y)$ dans l'élément K à un instant t^n donnée par:

$$u_h^n(x,y)=\alpha^{K,n}+\beta^{K,n}x+\gamma^{K,n}y=\sum_{i=1}^{3}\alpha_i^{K,n}\phi_i^K(x,y)$$

où $\phi_i^K(x,y)$ sont les fonctions de base associées à l'élément K et $\alpha_i^{K,n}$ sont les coefficients de la solution approchée sur l'élément K à l'instant t^n.
Soit $v=\phi_j^K(x,y)$pour $j=1,2,3$ une fonction test dans l'élément K. En substituant u_h^n dans l'équation **(3.22)**, on obtient:

$$\int_K\frac{u_h^{n+1}-u_h^n}{\delta t}\phi_j^K(x,y)-\int_K\mathbf{a}\cdot\nabla u_h^n\phi_j^K(x,y)+\sum_{Arete\in\partial K}\int_{Arete}\left(\widehat{\mathbf{a}u_h^n}\cdot\hat{\mathbf{n}}^k\right)\cdot\phi_j^K(x,y)=0$$

En remplaçant les expressions de u_h^n dans l'équation ci-dessus, on obtient:

$$\sum_{i=1}^{3}\alpha_i^{K,n+1}\int_K\phi_i^K(x,y)\phi_j^K(x,y)-\sum_{i=1}^{3}\alpha_i^{K,n}\int_K\mathbf{a}\cdot\nabla\phi_i^K(x,y)\phi_j^K(x,y)+\sum_{Arete\in\partial K}\int_{Arete}\left(\widehat{\mathbf{a}u_h^n}\cdot\hat{\mathbf{n}}^k\right)\cdot\phi_j^K(x,y)=0$$

Matrice de Masse:
La matrice de masse locale sur un élément triangulaire K est definie par:

$$M=(M_{ij})\quad\text{où}\quad M_{ij}=\int_K\phi_i^K(x,y)\phi_j^K(x,y)dxdy$$

Matrice d'advection
La matrice d'advection est donnée par:

$$S=(S_{ij})\quad\text{où}\quad S_{ij}=\int_K\mathbf{a}\cdot\nabla\phi_i^K(x,y)\phi_j^K(x,y)dxdy$$

Matrice de Flux aux interfaces:
Pour chaque bord $e_i\quad(i=1,2,3)$, on a:

$$F=(F_{e_i})\quad\text{où}\quad F_{e_i}=\int_{e_i}\left(\widehat{\mathbf{a}u_h^n}\cdot\hat{\mathbf{n}}^k\right)\cdot\phi_j^K(x,y)ds$$

Pour une arete e_i, $(i=1,2,3)$ partagée par K et L, On decompose cette expression en deux parties:
La moyenne des flux:

$$\int_{e_i}\mathbf{a}\left(u_h^n\Big|_K+u_h^n\Big|_L\right)\cdot\mathbf{n}\cdot\phi_j^K(x,y)ds=\int_{e_i}\mathbf{a}u_h^n\Big|_K\cdot\mathbf{n}\cdot\phi_j^K(x,y)ds+\int_{e_i}\mathbf{a}u_h^n\Big|_L\cdot\mathbf{n}\cdot\phi_j^K(x,y)ds$$

En remplaçant les expressions de u_h^n sur les éléments K et L, on obtient:

$$\int_{e_i} \frac{1}{2}\mathbf{a}\left(\sum_{i=1}^{3}\alpha_i^{K,n}\phi_i^K(x,y)+\sum_{i=1}^{3}\alpha_i^{L,n}\phi_i^L(x,y)\right)\cdot\mathbf{n}\cdot\phi_j^K(x,y)ds$$

En termes matriciels, cela se traduit par une multiplication vecteurs/matrices:

$$\mathbf{F_1}=\frac{1}{2}\mathbf{n^T a}\left(\mathbf{M}_{KL}\alpha^{K,n}+\mathbf{M}_{LK}\alpha^{L,n}\right)$$

où:

- $\mathbf{M}_{KL}$ est la matrice des intégrales de $\phi_i^K(x,y)\phi_j^L(x,y)$ sur l'arete e_i:

$$(\mathbf{M}_{KL})_{ij}=\int_{e_i}\phi_i^K(x,y)\phi_j^L(x,y)ds$$

- $\alpha^{K,n}$ et $\alpha^{L,n}$ sont les vecteurs des coefficients de u_h^n sur les éléments K et L respectivement
- $\mathbf{n^T a}$ est le produit scalaire entre le vecteur normal $\mathbf{n}$ et le vecteur $\mathbf{a}$.

Et le terme de saut:

$$\int_{e_i} C(u_h^-\mathbf{n}^- + u_h^+\mathbf{n}^+)\cdot\mathbf{n}\cdot\phi_j^K(x,y)ds$$

Encore une fois, en remplaçant les expressions de u_h^- et u_h^+, on obtient:

$$\int_{e_i} C\left(\sum_{i=1}^{3}\alpha_i^{K,n}\phi_i^K(x,y)\mathbf{n}^-+\sum_{i=1}^{3}\alpha_i^{L,n}\phi_i^L(x,y)\mathbf{n}^+\right)\cdot\mathbf{n}\cdot\phi_j^K(x,y)ds$$

En termes matriciels:

$$\mathbf{F_2}=C\left(\mathbf{n}^-\mathbf{M}_K\alpha^{K,n}+\mathbf{n}^+\mathbf{M}_L\alpha^{L,n}\right)$$

où $\mathbf{M}_K$ et $\mathbf{M}_L$ sont les matrices associées aux éléments K et L respectivement:

$$(\mathbf{M}_K)_{ij}=\int_{e_i}\phi_i^K(x,y)\phi_j^K(x,y)ds,\quad (\mathbf{M}_L)_{ij}=\int_{e_i}\phi_i^L(x,y)\phi_j^L(x,y)ds$$

Triangle de référence:

Pour calculer ces matrices M et S, on se place tout d'abords dans un triangle «simple», appelé **triangle de référence** $\hat{K}$, dont les sommets sont $\hat{s}_0=(0,0),\hat{s}_1=(1,0),\hat{s}_2=(0,1)$, ordonné dans le sens trigonométrique. Pour différencier ce triangle d'un triangle du maillage, on lui adjoint un repère (ξ,η), dit **repère paramétrique**.

Notons $\hat{\phi}_i\in P^1(\hat{K})$ les trois fonctions de bases associées aux sommets $\hat{s}_i$, pour $i=0,1,2$, definies par $\hat{\phi}_i(\hat{s}_j)=\delta_{ij}$. Ces fonctions $\hat{\phi}_i$ étant des polynômes de degré 1, elles sont données par:

$$\begin{cases}\hat{\phi}_1(\xi,\eta)=1-\xi-\eta\\ \hat{\phi}_2(\xi,\eta)=\xi\\ \hat{\phi}_3(\xi,\eta)=\eta\end{cases}\tag{3.23}$$

Transformation affine entre triangle de réference et élément du Maillage Pour passer du triangle de référence à un élément K du maillage (élément physique), on utilise une transformation affine. Cette transformation permet de cartographier chaque point du triangle de référence $\hat{K}$ vers un point de l'élément K.

On considère un élément K du maillage avec les sommets $s_0=(x_0,y_0),s_1=(x_1,y_1),s_2=(x_2,y_2)$. la transformation affine entre $\hat{K}$ et K est donnée par:

$$\begin{pmatrix}x\\y\end{pmatrix}=T\begin{pmatrix}\xi\\\eta\end{pmatrix}+\begin{pmatrix}x_0\\y_0\end{pmatrix}$$

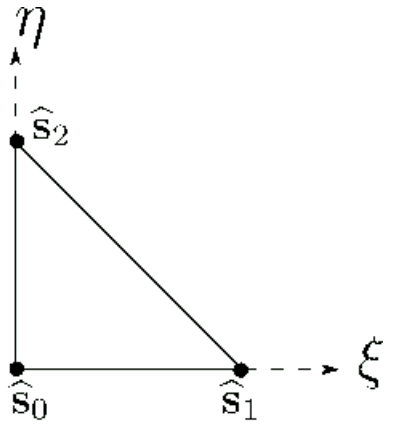

Figure 3.5: Triangle de référence $\hat{K}$

où T est la matrice Jacobienne de la transformation, definie par:

$$T = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}$$

et $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ est le vecteur de translation.

Calcul du Jacobien de la transformation et de son déterminant Le Jacobien de la transformation est important pour calculer les intégrales. Le determinant $|J_K| = \det(T)$ mesure le rapport entre l'élément physique K et le triangle de référence $\hat{K}$.
En particulier lorsqu'on intégre une fonction $f(x, y)$ sur l'élément physique K, on a:

$$\int_K f(x,y)dxdy = \int_{\hat{K}} f(T(\xi,\eta) + (x_0, y_0))|J_K|d\xi d\eta$$

Transformation du gradient des fonctions de base Lorsqu'on calcule le gradient d'une fonction de base sur l'élément réel, il faut le transformer depuis le triangle de référence. Ainsi, si $\nabla_{\xi,\eta}\hat{\phi}_j$ est le gradient de la fonction de base $\hat{\phi}_j$ dans l'élément de référence, alors le gradient dans l'élément réel est donnée par:

$$\nabla_{x,y}\phi_j = T^{-1}\nabla_{\xi,\eta}\hat{\phi}_j$$

Cette relation sera utilisée pour calculer la matrice d'advection.

Intégrations sur le triangle de référence

Dans le triangle de référence, les matrices de masse local et d'advection sont données respectivement par $\hat{M} = (\hat{M}_{ij})_{1\leq i,j\leq 3}$ et $\hat{S} = (\hat{S}_{ij})_{1\leq i,j\leq 3}$, où:

$$\hat{M}_{ij} = |J_K| \int_{\hat{K}} \hat{\phi}_i(\xi,\eta)\hat{\phi}_j(\xi,\eta)d\xi d\eta$$

$$\hat{S}_{ij} = |J_K| \int_{\hat{K}} \mathbf{a} \cdot (T^{-1}\nabla_{\xi,\eta}\hat{\phi}_j(\xi,\eta))\hat{\phi}_i(\xi,\eta)d\xi d\eta$$

La matrice de masse elementaire $\hat{M}$ est donnée par:

$$\hat{M} = \frac{1}{24}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

en effet, pour $i = j = 2$, soit $\hat{\phi}_i(\xi,\eta) = \hat{\phi}_j(\xi,\eta) = \xi$. Dans ce cas on a:

$$\hat{M}_{22} = |J_K| \int_{\hat{K}} \xi^2 d\xi d\eta = \int_0^1 \int_0^{1-\xi} \xi^2 d\eta d\xi = \left[\frac{\xi^3}{3} - \frac{\xi^4}{4}\right]_0^1 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}$$

Les calculs sont similaires pour $i = 1$, et $i = 3$. Prenons maintenant le cas $i \neq j$, par exemple $i = 3$ et $j = 2$:

$$\hat{M}_{32} = |J_K| \int_{\hat{K}} \hat{\phi}_3(\xi,\eta)\hat{\phi}_2(\xi,\eta)d\xi d\eta = \int_0^1 \left(\int_0^{1-\xi} \eta d\eta \right) \xi d\xi = \frac{1}{2}\int_0^1 \xi(1-\xi)^2 d\xi = \frac{1}{2}\left[\frac{1}{2} - \frac{2}{3} + \frac{1}{4}\right] = \frac{1}{24}$$

Les calculs sont similaires pour les autres coéfficients $M_{11}, M_{12}, M_{13}, M_{21}, M_{23}$ et M_{31}.

Système linéaire: $A\mathbf{U}^{n+1} = F$

En assemblant les matrices élémentaires, on obtient le système linéaire suivant:

$$A\mathbf{U}^{n+1} = F$$

où $\mathbf{U}^{n+1} = [\alpha^{K,n+1}, \beta^{K,n+1}, \gamma^{K,n+1}]^T$ est le vecteur des coefficients de la solution approchée à l'instant t^{n+1}, A est la matrice de taille $3N \times 3N$ et F est le vecteur de taille $3N$ donné par:

$$F = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix}$$

où F_i est le vecteur de taille N associé à l'élément i:

$$F_i = \begin{pmatrix} F_{e_1} \\ F_{e_2} \\ F_{e_3} \end{pmatrix}$$

et N est le nombre total d'éléments dans le maillage.
La matrice A est donnée par:

$$A = \begin{pmatrix} M_1 + S_1 & -F_1 & 0 \\ -F_1 & M_2 + S_2 & -F_2 \\ 0 & -F_2 & M_3 + S_3 \end{pmatrix}$$

où M_i et S_i sont les matrices de masse et d'advection associées à l'élément i respectivement.

Chapter 4

Implémentation de la méthode DG pour l'équation de transport 2D

Pour illustrer la méthode DG pour l'equaiton de transport 2D, voici quelques extraits de codes que nous avons eu à faire pour la résolution du problème de transports (3.18)

```
        class dG2DimTransport(AbstractProblem):
            def __init__(self, T=1.0, a=(1., 1.), k=1, N=20):
                self.steady = False
                super().__init__([(0., 1.), (0., 1.)], N)
                self.a = a

                self.phi_hat = [
                    lambda xi, eta: 1 - xi - eta,
                    lambda xi, eta: xi,
                    lambda xi, eta: eta
                ]

                self.dphi_xi = [
                    lambda xi, eta: -1,
                    lambda xi, eta: 1,
                    lambda xi, eta: 0
                ]

                self.dphi_eta = [
                    lambda xi, eta: -1,
                    lambda xi, eta: 0,
                    lambda xi, eta: 1
                ]

                self.p = len(self.phi_hat)
                self.dofs = self.mesh.ne * self.p
                self.T = T  # Temps final

                self.u = np.ones((self.mesh.ne, self.p))  #
                self.u0 = lambda x, y: 1  # Condition initiale
                self.ue = lambda t, x, y: self.u0(x - t * self.a[0], y - t * self.a[1])

```

Ceci permets d'initialiser la classe dG2DimTransport qui hérite de la classe AbstractProblem.
Ensuite les quelques méthodes importantes pour assembler les matrices élémentaires M, S, et F.

```
        def eval_at_point(self, t, x, y):
            n = self.mesh.get_index((x, y))

            x0, y0 = self.mesh.get_vertex(n, 0)
            x1, y1 = self.mesh.get_vertex(n, 1)
            x2, y2 = self.mesh.get_vertex(n, 2)

```

```
            T, B = self.transformer((x0, y0), (x1, y1), (x2, y2))
            xi_eta = np.linalg.inv(T) @ (np.array([x, y]) - B)

            phi_values = np.array([phi(xi_eta[0], xi_eta[1]) for phi in self.phi_hat])
            u_t = self.u[n, :]

            ut = u_t @ phi_values
            return ut

        def transformer(self, X0, X1, X2):
            x0, y0 = X0
            x1, y1 = X1
            x2, y2 = X2

            T = np.array([[x1 - x0, x2 - x0], [y1 - y0, y2 - y0]])
            B = np.array([x0, y0])

            return T, B

        def transform_to_global(self, T, B, xi, eta):
            global_coords = T @ np.array([xi, eta]) + B
            return global_coords[0], global_coords[1]

        def integrand_M(self, i, j, T, B, xi, eta):
            return self.phi_hat[i](xi, eta) * self.phi_hat[j](xi, eta)

        def assemble_Sa_local(self, i, j, T, a, xi, eta):
            T_inv_T = np.linalg.inv(T).T
            integrand_Sa = (a @ (T_inv_T @ np.array([self.dphi_xi[j](xi, eta), self.dphi_eta[j](xi,
 eta)]))) * self.phi_hat[i](xi, eta)
            return integrand_Sa

        def assembly_local(self, C=0.5, element=0):
            x0, y0 = self.mesh.get_vertex(element, 0)
            x1, y1 = self.mesh.get_vertex(element, 1)
            x2, y2 = self.mesh.get_vertex(element, 2)

            M_local = np.zeros((self.p, self.p))
            Sa_local = np.zeros((self.p, self.p))
            F_e = np.zeros(self.p)

            X0, X1, X2 = (x0, y0), (x1, y1), (x2, y2)
            T, B = self.transformer(X0, X1, X2)
            detT = np.abs(np.linalg.det(T))

            for i in range(self.p):
                for j in range(self.p):
                    M_local[i, j] = integrate.dblquad(lambda xi, eta: self.integrand_M(i, j, T, B,
 xi, eta) * detT,
                                                      0, 1, lambda xi: 0, lambda xi: 1 - xi)[0]
                    Sa_local[i, j] = integrate.dblquad(lambda xi, eta: self.assemble_Sa_local(i, j,
 T, self.a, xi, eta) * detT,
                                                       0, 1, lambda xi: 0, lambda xi: 1 - xi)[0]

            voisins = self.mesh.voisins[element]
            for index, voisin in enumerate(voisins):
                if voisin == -1:
                    continue
                else:
                    x0_v, y0_v = self.mesh.get_vertex(voisin, 0)
                    x1_v, y1_v = self.mesh.get_vertex(voisin, 1)
                    x2_v, y2_v = self.mesh.get_vertex(voisin, 2)

                    T_v, B_v = self.transformer((x0_v, y0_v), (x1_v, y1_v), (x2_v, y2_v))

                    # Normalisation de la normale
                    nx, ny = y1 - y0, x0 - x1
                    norme = math.sqrt(nx**2 + ny**2)
```

```
                    nx, ny = nx / norme, ny / norme

                    for i in range(self.p):
                        integrand_flux = lambda s: (0.5 * (self.a[0] * nx + self.a[1] * ny) * self.
    phi_hat[i](s, 0) +
                                                    C * (nx**2 + ny**2) * self.phi_hat[i](s, 0))
                        F_e[i] += integrate.quad(integrand_flux, 0, 1, limit=100)[0]

            return M_local, Sa_local, F_e


```

Une assemblage des matrices élémentaires pour obtenir la matrice globale A et le vecteur F.

```
        def assemblage_global(self, C=0.5):
            n_dofs = self.mesh.ne * self.p
            A = np.zeros((n_dofs, n_dofs))
            S = np.zeros((n_dofs, n_dofs))
            F = np.zeros(n_dofs)

            for K in range(self.mesh.ne):
                MK, SK, Fe = self.assembly_local(C, K)
                for i in range(self.p):
                    dof_i = K * self.p + i
                    F[dof_i] += Fe[i]
                    for j in range(self.p):
                        dof_j = K * self.p + j
                        A[dof_i, dof_j] += MK[i, j]
                        S[dof_i, dof_j] += SK[i, j]

            return A, S, F


```

Voir le code complet à l'adresse suivante:
https://gitlab.math.unistra.fr/aghili/dg/-/blob/dieng-main-patch-68158/EquationTransport.py?ref_type=heads

et le repo gitlab complet se trouve à l'adresse suivante:
https://gitlab.math.unistra.fr/aghili/dg/-/tree/dieng-main-patch-68158?ref_type=heads

Quelques tests unitaires pour vérifier les méthodes:
Afin de vérifier ces méthodes implémentées, nous avons fait quelques tests unitaires pavec **pytest** pour vérifier les assemblages des matrices élémentaires.

```
        import pytest
        import numpy as np
        from scipy import integrate
        from EquationTransport import dG2DimTransport

        expected_matrix = (1/24) * np.array([[2,1, 1],
                                    [1, 2, 1],
                                    [1, 1, 2]])

        def setup_test_problem():
            T = 1.0
            a = (0.25, 0.25)
            k = 1
            N = 20
            problem = dG2DimTransport(T=T, a=a, k=k, N=N)
            return problem

        def test_integrand_M():
            problem = setup_test_problem()
            element = 0
            x0, y0 = problem.mesh.get_vertex(element, 0)
```

```
            x1, y1 = problem.mesh.get_vertex(element, 1)
            x2, y2 = problem.mesh.get_vertex(element, 2)
            X0, X1, X2 = (x0, y0), (x1, y1), (x2, y2)
            A, B = problem.transformer(X0, X1, X2)

            for i in range(problem.p):
                for j in range(problem.p):
                    result = integrate.dblquad(lambda x, y: problem.integrand_M(i, j, A, B, x, y),
    0, 1, lambda x: 0, lambda x: 1 - x)[0]
                    assert np.isclose(result, expected_matrix[i, j]), f"Integrand M({i},{j}) = {
    result}, expected {expected_matrix[i, j]}"

        if __name__ == "__main__":
        pytest.main()

```

Dans la sortie de la commande **pytest**, on obtient le résultat suivant qui indique bien que cette assemblage de la matrice de masse est correcte.

```
            ========================== test session starts ==========================
            platform linux -- Python 3.10.9, pytest-7.1.2, pluggy-1.0.0
            rootdir: /home/mai-pret/Bureau/stage/StageDG/dg
            plugins: anyio-3.5.0
            collected 1 item

            test_dG2DimTransport_integrals.py .                              [100%]

            ========================== 1 passed in 0.80s ===========================

```

Chapter 5

Conclusion et perspectives

Conclusion

Ce travail a permis d'explorer en profondeur les aspects théoriques et pratiques liés à l'application de la méthode de Galerkin discontinue (DG) pour la résolution d'équations différentielles ordinaires et d'équations de transport en 2D.
Ce travail a permis d'explorer en profondeur les aspects théoriques et pratiques liés à l'application de la méthode de Galerkin discontinue (DG) pour la résolution

Du point de vue de l'implémentation, nous avons d'abord traité le cas en 1D, ce qui nous a permis de reproduire les ordres d'erreurs théoriques attendus. Cette étape a été déterminante pour valider notre approche et ajuster les paramètres de la méthode DG. Toutefois, il a été constaté que le choix des fonctions de base n'était pas optimal, ce qui a conduit à des résultats sous-optimaux. Une amélioration de ce choix pourrait potentiellement augmenter la précision de la méthode.

Le passage au cas 2D a introduit de nouvelles difficultés, notamment liées à la complexité accrue des flux aux interfaces et à l'assemblage des matrices. Ces défis ont nécessité une adaptation des techniques utilisées en 1D pour les rendre compatibles avec les exigences d'une discrétisation en deux dimensions.

Perspectives

Les perspectives de ce travail sont multiples:

- **Amélioration en 1D:** Un des premiers objectifs serait d'optimiser le choix des fonctions de base afin de réduire les erreurs et d'atteindre des ordres de convergence plus proches des prévisions théoriques.
- **Finalisation en 2D:** L'application de la méthode DG à un cas simple en 2D, tel que l'advection scalaire constante, constitue une prochaine étape logique. Ce test permettra de vérifier la robustesse de l'approche développée et d'identifier les ajustements nécessaires avant de passer à des problèmes plus complexes.

Bibliography

[1] Bharat Tripathi.
Discontinuous Galerkin Method for Propagation of Acoustical Shock Waves in Complex Geometry. Acoustics [physics.class-ph]. Université Pierre et Marie Curie - Paris VI, 2015. English. ffNNT : 2015PA066344ff. fftel-01297050f

[2] Bernardo Cockburn.
Discontinuous Galerkin methods. Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, 2003, 83 (11), pp.731-754. 10.1002/zamm.200310088. hal-01352444

[3] https://fr.wikipedia.org/wiki/M%C3%A9thode_de_Galerkine_discontinue

[4] Delfour, M., W. Hager, and F. Trochu.
"Discontinuous Galerkin Methods for Ordinary Differential Equations. " Mathematics of Computation 36, no. 154 (1981): 455–73. https://doi.org/10.2307/2007652

[5] Jean-Baptiste Clément, Mehmet Ersoy, Frederic Golay, Damien Sous.
Discontinuous Galerkin Method for steady-state Richards Equation. Topical Problems of Fluid Mechanics 2019, Feb 2019, Prague, Czech Republic. pp.53-62, 10.14311/TPFM.2019.008. hal-02075109

[6] Bernardo Cockburn.
An Introduction to the Discontinuous Galerkin Method for Convection-Dominated Problems School of Mathematics, University of Minnesota,