

MÉTHODE STILS POUR L'ÉQUATION
DE TRANSPORT : COMPARAISONS ET ANALYSES.
ETUDE D'UN MODÈLE DE FERMETURE POUR LA LOI DE DARCY

THESE
présentée à la faculté des sciences, pour obtenir
le grade de docteur ès sciences, par

Gautier de Montmollin

sous la direction du Professeur

Olivier Besson

UNIVERSITÉ DE NEUCHÂTEL
Institut de Mathématiques
Emile-Argand 11
2007 NEUCHÂTEL

Février 2001

IMPRIMATUR POUR LA THESE

**Méthode STILS pour l'équation de transport:
comparaisons et analyses. Etude d'un modèle de
fermeture pour la loi de Darcy**

de M. Gautier de Montmollin

UNIVERSITE DE NEUCHÂTEL

FACULTE DES SCIENCES

La Faculté des sciences de l'Université de
Neuchâtel sur le rapport des membres du jury,

MM. O. Besson (directeur de thèse),
P. Perrochet, P. Lesaint (Besançon F) et
P. Azérad (Perpignan F)

autorise l'impression de la présente thèse.

Neuchâtel, le 28 mai 2001

Le doyen:



J.-P. Derendinger

Sommaire

Résumé	9
Introduction	11
1 Construction de la méthode STILS	19
1.1 Equation de transport	19
1.2 Rappels et compléments sur la méthode STILS	20
1.2.1 Vers une formulation faible	21
1.2.2 Equation faible de STILS et éléments finis	22
1.2.3 Une version marche-en-temps de STILS	24
2 Comparaisons entre STILS et d'autres méthodes	27
2.1 Éléments finis continus en espace	27
2.1.1 Streamline diffusion	28
2.1.2 Shock capturing [29, 32, 33]	28
2.1.3 Une méthode de caractéristiques	29
2.2 Expériences numériques	30
2.2.1 Cylindre fendu	31
2.2.2 Conservativité	35
2.2.3 Cône absorbé par 6 tourbillons	36
2.2.4 Plug-flow	40
2.3 Conclusion à STILS d'ordre 1 en temps	44
3 Comparaison fine et stabilité	45
3.1 Moindres carrés et espace-temps : apports respectifs	45
3.2 Analyse de stabilité des méthodes TMQ1	47
3.2.1 Amplitude	51
3.2.2 Déphasage	52
3.2.3 Lien entre analyse de Fourier et analyse matricielle	53
3.3 Conclusion	54
4 STILS pour les degrés supérieurs en temps	57
4.1 Degré 2 en temps : formulation STILS-TMQ2	57
4.2 Propriétés des matrices pour un degré M quelconque	59
4.3 Degré 3 en temps : formulation STILS-TMQ3	60
4.4 Comparaisons entre STILS-TMQ1, TMQ2 et TMQ3	62
4.5 Stabilité comparée de STILS-TMQ1, TMQ2 et TMQ3	63
4.5.1 Amplification de STILS-TMQ2	63

4.5.2	Amplification de STILS-TMQ3	64
4.5.3	Stabilité	65
5	Essais de méthodes discontinues	71
6	STILS pour des équations non linéaires	75
6.1	Equation de Burgers	76
6.1.1	Séparation espace-temps	77
6.2	Essais numériques de STILS-TMQ1-Burgers	79
6.2.1	Onde de choc	80
6.2.2	Onde de raréfaction	81
6.2.3	Recours à la linéarisation	82
6.2.4	Semi-linéarisation	84
7	Modèle de fermeture pour la loi de Darcy	93
7.1	Énoncé du problème	93
7.2	Point de vue local	94
7.2.1	Obtention de la perméabilité K	95
7.2.2	Effets de la périodicité	95
7.2.3	Problème local : formulation faible	95
7.2.4	Propriétés des espaces $H_{P,\Gamma}$, $V_{P,\Gamma}$, $V_{P,\Gamma,\text{div}}$	97
7.2.5	Problème variationnel sans ∇d	97
7.3	Résultats d'existence et d'unicité pour D et d	97
7.3.1	Existence et unicité de d définie sur Ω , cas purement local	98
7.3.2	Existence et unicité de d définie sur Ω , cas périodique localisé	100
	Conclusion et perspectives	107
	Annexes	107
A	Programmes de résolution	109
A.1	Structure	109
A.2	Fonctionnement	111
A.3	Formats de fichiers	111
A.4	Utilitaires	114
B	Modules, ossature	115
B.1	Matrices usuelles	115
B.1.1	Cadre générique	115
B.1.2	Instanciation particulière	116
B.2	BandMatr : matrices bandes	116
B.3	Sparse : matrices creuses	118
B.4	Maillage et réduction de largeur de bande	120
B.5	Programme FT	121

C	Routines importantes de FT	129
C.1	Construction des matrices élémentaires	129
C.2	Assemblage des matrices globales	136
C.3	Remplissage des matrices globales	140
C.4	Paramètres pour l'ensemble du projet Flux-Transport	141
C.5	STILS-Burgers	146

Résumé

Les cinq premiers chapitres de la présente thèse sont consacrés à la méthode STILS (Space-Time Integrated Least Squares). C'est une méthode d'éléments finis permettant de résoudre de manière approchée l'équation de transport

$$\frac{\partial c}{\partial t} + (\nabla c | v) = f$$

qui est une équation aux dérivées partielles linéaire, dite de convection pure. Le lecteur intéressé par la mise en œuvre pratique de STILS pourra se pencher particulièrement sur la fin du premier chapitre et sur les chapitres 2 et 4.

Dans le chapitre 6, on étudie l'application de STILS pour le cas non-linéaire.

Le chapitre 7 est une contribution théorique à la détermination du tenseur K dans l'équation de diffusion

$$\operatorname{div}(K \nabla h) = S_s \frac{\partial h}{\partial t} + q.$$

Il n'est cependant pas tout à fait étranger au reste de la thèse puisque, pour les problèmes, d'hydrogéologie notamment, où une équation de transport est couplée à une équation de diffusion par la relation $v = -K \nabla h$, on a la chaîne de dépendances $c \rightarrow v \rightarrow h \rightarrow K$, autrement dit, il faut obtenir K avant même de pouvoir essayer de trouver c .

Découpage en chapitres

Voici la liste des différents chapitres. Nous marquons d'un « M » les chapitres, plutôt théoriques, qui s'adressent essentiellement aux mathématiciens, physiciens, et généralement à ceux qui sont familiers avec les espaces de Sobolev. Les chapitres contenant des formulations numériques sont marquées d'un « N ».

Construction de la méthode STILS	1	M	N
Comparaisons entre STILS marche-en-temps et d'autres méthodes	2		N
Comparaison fine et stabilité	3		
STILS pour les degrés supérieurs en temps	4		N
Essais de méthodes discontinues	5	M	
STILS pour des équations non-linéaires	6		N
Modèle de fermeture pour la loi de Darcy	7	M	
Annexes informatiques	A,B,C		N

Introduction

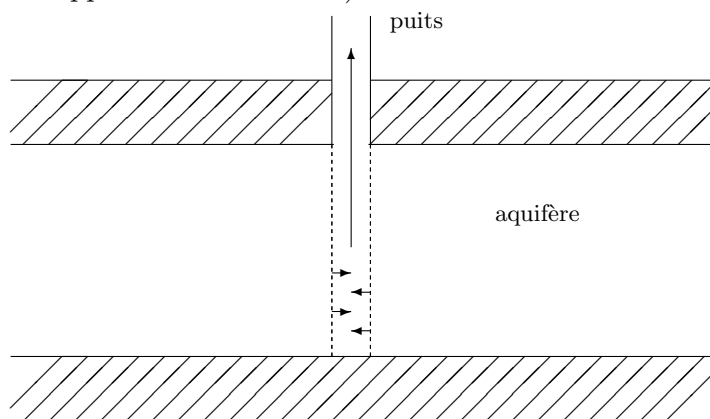
Parmi les phénomènes naturels fondamentaux, la diffusion ou la convection apparaissent dans des contextes extraordinairement différents ¹. Des scientifiques d’horizons tout aussi divers ont observé, formulé des lois empiriques, établi des notations et un jargon. Il a fallu parfois des décennies pour que quelqu’un remarque les coïncidences entre des lois, dissimulées par les notations et le cloisonnement — bien humain — des sciences.

Prenons un exemple d’hydrogéologie, l’écoulement en milieux poreux. Ce sera, disons-le d’emblée, le terrain de choix pour héberger les équations traitées dans cette thèse, car la diffusion comme la convection y baignent ensemble.

Vers 1856, Henri Darcy, en étudiant les fontaines de Dijon, énonce une loi expérimentale sur le débit de l’eau s’écoulant à travers un massif de sable [15].

En 1935, soit près de 80 ans plus tard, Ch. V. Theis observe (« coup d’œil fondamental »[34]) que cette loi n’est autre que l’équation de la chaleur ; il suffit de remplacer « température » par « pression hydraulique ». « Ainsi, écrit-il [59], la théorie mathématique de la conduction de la chaleur développée par Fourier et les auteurs subséquents est largement applicable à la théorie hydraulique ».

On finira d’ailleurs par étayer la loi de Darcy par les équations de Navier-Stokes qui régissent l’écoulement des fluides [17]. Mais revenons à Theis qui accompagne son propos d’un exemple très intéressant. On considère un puits servant à pomper de l’eau en permanence d’un aquifère saturé (une nappe d’eau souterraine).



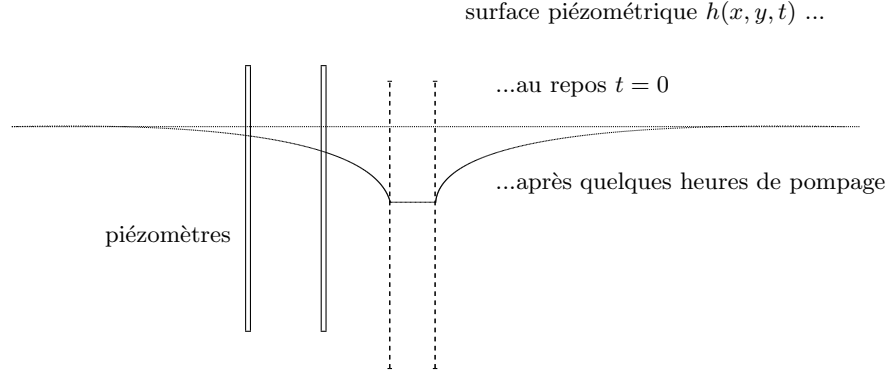
On fait quelques hypothèses §8.1 de [17], [34] :

- pour le terrain : il est confiné, d’étendue infinie, homogène, isotrope, d’épaisseur uniforme et avec une surface piézométrique ² horizontale — c’est à dire, une pression

¹Parmi les domaines d’application, citons en vrac la simulation industrielle, la météorologie, l’hydrologie, l’hydrogéologie, la finance

²le piézomètre est un simple trou foré dans le sol, parfois renforcé par un tubage perforé. Le niveau d’eau

hydraulique uniforme avant le pompage — ;
 – pour le puits : débit constant, profondeur égale à l'épaisseur du terrain.
 On s'attend alors à avoir le profil piézométrique suivant sous l'effet du pompage :



Il est temps de préciser le modèle physico-mathématique sous-jacent. Définissons les variables physiques :

symbole	dimension	nature	unités physiques
v	vecteur	vitesse de filtration	m/s
h	scalaire	charge hydraulique	m
K	tenseur	tenseur de conductivité fluide	m/s
S_s	scalaire	coefficient d'emmagasinement spécifique	$1/m$
q	scalaire	terme de source	$1/s$
ϱ	scalaire	densité de la phase liquide	Kg/m^3
g	scalaire	norme de la pesanteur	m/s^2

Toutes peuvent dépendre du lieu (x, y, z) et du temps t , mais il y aura des simplifications. Dans notre cas, v et h sont inconnues tandis que les autres variables sont des données.

La **loi de Darcy** (stationnaire) s'énonce $v = -K\nabla h$: la vitesse de filtration dérive du potentiel de charge hydraulique. L'**équation de continuité** dit : $-\text{div } v = S_s \frac{\partial h}{\partial t} + q$. Au total, on a l'**équation d'écoulement**

$$\text{div}(K\nabla h) = S_s \frac{\partial h}{\partial t} + q \quad (1)$$

que Theis a évidemment reconnue comme l'équation de la chaleur ! En fait, on joue sur la chronologie : ces notations, plus récentes, cachent des trésors d'ingéniosité, de la maturation, des choix et quelques approximations, comme l'invariabilité de la densité ϱ par rapport au temps. pour l'équation de continuité. Dans le problème de Theis, K est le tenseur identité I (terrain isotrope) fois une *constante* T (terrain homogène). Il s'ensuit : $\text{div}(K\nabla h) = T\Delta h$. Il y a aussi une symétrie centrale sur le plan (x, y) si on regarde le terrain à vol d'oiseau. On a donc avantage à passer des coordonnées cartésiennes (x, y, z) aux coordonnées cylindriques (r, φ, z) , dans lequel le laplacien $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ s'écrit

$$\Delta = \frac{1}{r} \frac{\partial}{\partial r} + \frac{\partial^2}{\partial r^2} + \frac{1}{r^2} \frac{\partial^2}{\partial \varphi^2} + \frac{\partial^2}{\partial z^2}.$$

indique la pression par la relation bien connue $p = \varrho g(h + h_0)$; on choisit h_0 pour que $p = 0$ corresponde à la pression atmosphérique.

Au vu des hypothèses et de la géométrie du problème, il est raisonnable de supposer que h est indépendant de φ et de z . Ainsi, exit les coordonnées φ , z et les dérivées de h en ces variables. On a maintenant $\Delta h = \frac{1}{r} \frac{\partial h}{\partial r} + \frac{\partial^2 h}{\partial r^2}$. Dans sa formulation, Theis fait du puits une source ponctuelle en $(0, 0)$ dans le plan (x, y) . Il n'inclut pas q comme terme dans l'équation, mais en fait une condition de Neumann au bord d'un puits infiniment étroit. L'équation (1) s'écrit dans ce contexte

$$\frac{1}{r} \frac{\partial h}{\partial r} + \frac{\partial^2 h}{\partial r^2} = S_s \frac{\partial h}{\partial t} \quad (2)$$

et on peut définir le **problème mathématique** suivant : trouver $h(r, t)$ dans $]0, \infty[\times]0, \infty[$, solution de (2) et telle que

$$h(r, 0) = 0 \quad \text{Condition initiale pour tout } r > 0 \quad (3)$$

$$\lim_{r \rightarrow 0} r \frac{\partial h}{\partial r}(r, t) = \frac{q}{2\pi T} \quad \text{Condition au bord (le point 0)} \quad (4)$$

$$\lim_{r \rightarrow \infty} h(r, t) = 0 \quad \text{Condition au bord (l'infini)} \quad (5)$$

— c'est exactement le problème (5.1) - (5.3) tel que reproduit dans la documentation FE-FLOW : [34]. En coordonnées cartésiennes, le problème pour h est posé sur le plan privé de l'origine $\mathbb{R}^2 \setminus 0$. Il admet une solution analytique

$$h(r, t) = \frac{-q}{4\pi T} W(u(r, t)), \quad u(r, t) = \frac{r^2 S}{4Tt}, \quad W(u) = \int_u^\infty \frac{e^{-x}}{x} dx \quad (6)$$

qui est unique par la théorie de l'équation de la chaleur. Cela nous permettra d'y comparer une approximation numérique. Il n'y a pas de difficulté, les méthodes convergent et une méthode simple d'éléments finis, par exemple, convient très bien — ce n'est pas pour rien qu'on qualifie l'équation de la chaleur d'équation de débutant !

Entre en scène un second phénomène directement lié au champ de vecteur v donné par l'équation d'écoulement : **la convection**. Soit c la concentration chimique d'une substance soluble dans l'eau — par exemple du sel. C'est une quantité pure entre 0 et 1.

symbole	dimension	nature	unités physiques
c	scalaire	concentration chimique	1

Si on « pollue » une portion de terrain au temps initial $t = 0$, en posant $c = 1$ sur cette portion, 0 ailleurs, l'évolution de cette concentration c dans l'espace et le temps obéira à **l'équation de transport**

$$\frac{\partial c}{\partial t} + \operatorname{div}(cv) = f. \quad (7)$$

C'est sur la résolution **numérique** de cette équation que porte la présente thèse (hormis le dernier chapitre). On s'en doute, cette équation est moins aimable que l'équation d'écoulement.

Concluons cette introduction sur une simulation du problème de Theis et d'une « pollution » transportée par la solution dudit problème. C'est un cas heureux où l'on a en même temps, outre la **simulation numérique**, des **expériences** faites dans le monde réel et une **solution analytique** explicite du problème mathématique. Comme la pierre de Rosette qui comporte le même texte traduit en trois langues, il est possible ici de confronter deux à deux réalité, modèle mathématique et simulation.

Pour une simulation numérique — du moins sur les ordinateurs actuels —, il faut passer

du continu au discret et de l'infini au fini. Pour ce faire, on dispose des points qui s'espacent de manière exponentielle à partir du puits et régulière sur des cercles concentriques. Pour la finitude, on pose, comme le font Pinder et Frind [49], le rayon du puits — devenu non-ponctuel — à un pied ($0,304m$), assorti de la condition au bord

$$r \frac{\partial h}{\partial r}(r, t) = \frac{q}{2\pi T}, \quad r = r_m$$

et le rayon extrême du terrain à mille pieds ($304m$), assorti de la condition

$$h(r, t) = 0, \quad r = r_M.$$

Notre maillage, pour 28 anneaux découpés en 60 secteurs est de forme radiale. Il y a un peu plus de rayons du côté du 3^e quadrant — on en bénéficiera plus tard. Les anneaux sont proportionnels à leur distance au centre, c'est pour tenir compte des différentes échelles de grandeur en jeu (des rapports de 1 à 1000). Le maillage se présente ainsi :

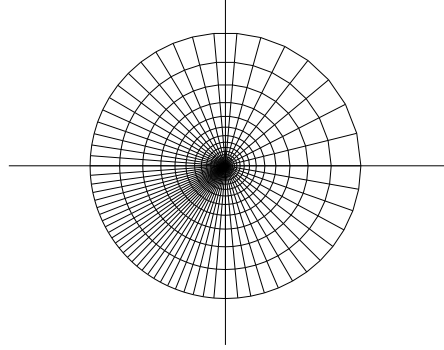
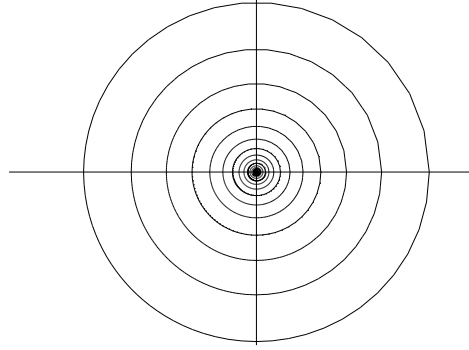


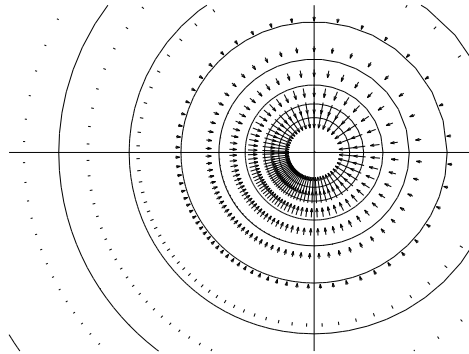
FIG. 1 – Maillage pour le problème de Theis ; rayon : $304m$

On l'a dit, la résolution de l'équation d'écoulement (1) — ou plus généralement d'une équation à terme diffusif dominant — n'est plus un défi majeur : la méthode de Galerkin la plus classique donne une très bonne approximation d'une solution exacte ³. Avant tout, montrons la solution h et la vitesse de filtration $v = -T\nabla h$ après six jours de pompage. On a une convergence rapide vers la solution stationnaire où $\text{div } v = 0$.

³Notons que dans ce problème particulier, la condition à l'infini du modèle théorique de Theis est difficilement approchable

FIG. 2 – Vingt courbes de niveau de h après 6 jours

NB : il ne faut pas trop chercher les vecteurs vitesse à l'échelle de $304m$; si on les agrandissait, ceux qui sont près du centre envahiraient l'image. Sur chaque cercle du maillage, on a une valeur constante de h . Numériquement, on a, depuis l'extérieur, $0, -0.59, -1.18, -1.78, \dots, -13.6, -14.3, -14.8, -15, 4, -16.0, -16.6m$ de charge hydraulique : une progression linéaire d'un cercle à l'autre, donc logarithmique par rapport au rayon puisque les cercles sont disposés exponentiellement.

FIG. 3 – v et vingt courbes de niveau de h après 6 jours, au bord du puits

Vers le bord du puits, on a $h = -16.6$ mètres de charge hydraulique. Ce n'est qu'à quelques mètres du puits que la vitesse v devient « importante » avec $6.50310^{-3} m/s$.

Simulons à présent l'équation de transport (1.1). Comme condition initiale, on met une concentration chimique $c = 1$ sur le quart de plan inférieur gauche $[-\infty; -20] \times [-\infty; 0]$ (en mètres). On laisse le courant v l'emporter vers le centre. Au départ, on a les courbes de niveau suivantes (on voit que les mailles, trop grossières, s'adaptent mal au rectangle!) :

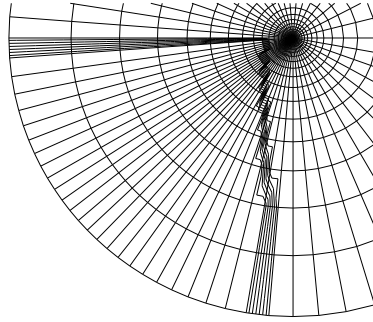


FIG. 4 – Concentration chimique c au départ : $c = 1$ en bas à gauche, $c = 0$ ailleurs

Sur le front de la concentration, d'une maille où $c = 1$ à une autre où $c = 0$, l'algorithme de tracé de courbes de niveau interpole les valeurs ; si le problème n'était pas discrétisé, nous n'aurions besoin d'en représenter qu'une seule à ce stade.

Après six jours, la simulation par la méthode STILS (telle que nous la verrons aux chapitres 1 et 2) nous donne :

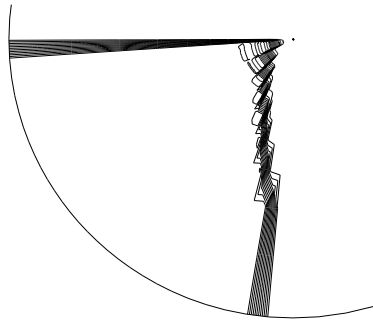


FIG. 5 – Concentration chimique c après 30 heures

Désormais, ce qui se passe près du puits devient crucial. Il nous faut faire un « zoom » significatif pour tenter d'y voir clair — d'abord à l'échelle de $30m$:

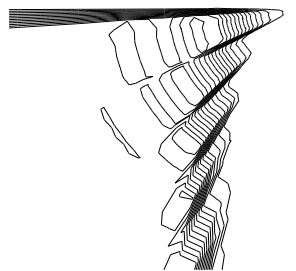


FIG. 6 – Concentration chimique c après 36 heures

Pour les courbes de niveau, il faut imaginer des marches d'escalier descendantes. On

observe déjà quelques perturbations numériques sur le front de la pollution, ainsi qu'un « étalement » de celui-ci. Il faut encore s'approcher (échelle : $10m$) pour voir le premier contact de la pollution ($c > 0$) avec le bord du puits. En fait, si la convection était parfaitement simulée, on devrait avoir un passage brusque de $c = 0$ à $c = 1$, c'est-à-dire des courbes de niveau resserrées au maximum (au pire, d'une maille à l'autre).

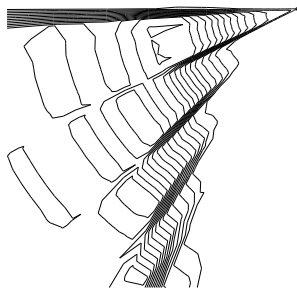


FIG. 7 – Concentration chimique c après 42 heures

Il faut donc encore laisser passer beaucoup d'heures pour affirmer que le bord du puits a été touché. Après 144 heures (6 jours) on a la situation

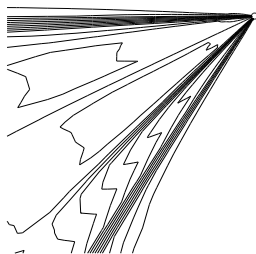


FIG. 8 – Concentration chimique c après 6 jours

qui nécessite un nouveau rapprochement à l'échelle de 3 mètres.

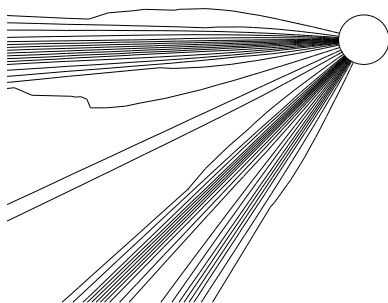


FIG. 9 – Concentration chimique c après 6 jours

On voit bien qu’une partie du bord (environ $\frac{1}{12}$ du cercle) est atteinte par $c = 1$. A vrai dire, on remarque des « îlots » où c peut atteindre la valeur 1.18. De même, sur les côtés on peut avoir dans ce calcul jusqu’à $c = -0.034$. Ces défauts numériques sont ici largement provoqués par le maillage ; il plaide coupable : on devrait mieux l’adapter à la condition initiale de c , mais cela ne gommerait pas totalement ces défauts.

On a, au total, observé deux phénomènes gênants, purement numériques. D’abord la concentration chimique c déborde de l’intervalle $[0; 1]$, quand bien même on prouve facilement qu’elle ne doit pas le faire si elle résout l’équation de transport. Puis on observe une certaine diffusion ; elle ne devrait pas avoir lieu.

Le présent travail (sans le dernier chapitre) entend mettre en lumière ces phénomènes, communs aux méthodes numériques courantes, et les corriger quelque peu. Par exemple, mieux on maîtrise la diffusion numérique, mieux on peut corriger *localement* les débordements.

Remerciements

D’abord, il me faut remercier mon directeur de thèse, le professeur Olivier Besson, dont l’expérience et le flair mathématique m’ont toujours permis de me repérer au cours de mes recherches. De nos discussions ont émergé (et j’espère émergeront encore !) des belles idées, et ce de manière détendue. Il me faut bien sûr remercier les professeurs P. Azérad, P. Lesaint et P. Perrochet d’avoir bien voulu faire parti du jury et donc relire ces pages j’espère point trop absconses.

Mes remerciements les plus chaleureux à ma fiancée, Dr Eva Schläpfer pour sa relecture attentive et critique des épreuves de cette thèse.

Un grand merci à Stéphane Perret, magicien ès L^AT_EX qui a notamment résolu le problème épineux de placer $\square$ au bon endroit .

Ce travail aurait été bien austère sans la joyeuse humeur distillée, notamment au lieu-dit le Saloon, par mes collègues (dans l’ordre alphabétique pour ne pas faire de jaloux) Priska Ammann, Yves Bolle, Eve Chédel, Roger Filliger, Anne Gertsch, Marc Gindraux, Xavier Hainaut, Jean-François Hämmerli, Martin Jakob, Alexandre Junod, Souleye Kane, Nicolas Louvet, Florian Martin, Michel Matthey — et je crains d’en oublier.

Il me faut aussi remercier les collaborateurs du CHYN (centre d’hydrogéologie de l’Université de Neuchâtel), Farid Achour, Francesco Kimmeier, et Mahmoud Bouzelboudjen pour leur fructueuses conversations sur la cohabitation au quotidien entre les équations aux dérivées partielles et les réalités naturelles — et également pour la riche documentation du logiciel FEFLOW.

Chapitre 1

Construction de la méthode STILS

The poor lady [...] had [...] quite simple but not unattractive features of a type that may be defined as a weak solution of Marlene Dietrich.

Vladimir Nabokov, *Lolita*, ch. 10

Le but, en étudiant la méthode STILS, est de voir si elle améliore la qualité de la simulation numérique, au moins dans un certain contexte : équation de convection pure, méthodes des éléments finis. On en verra la confirmation expérimentale. De plus, cet « optimum relatif » apparaît dès le départ, dans la formulation mathématique de la méthode.

Nous présentons donc dans ce chapitre un rappel détaillé sur la méthode STILS et une discrétisation par éléments finis à l'aide de fonctions de base affines en temps. Le prochain chapitre est consacré à une comparaison expérimentale entre cette méthode et des méthodes usuelles.

1.1 Equation de transport

Soit $\Omega \subset \mathbb{R}^d$ un domaine borné ayant un bord $\mathcal{C}^1$ par morceaux. On prend comme temps initial 0 et comme temps final T . Le problème physique comprend comme données $v : \Omega \rightarrow \mathbb{R}^d$, un champ de vitesse stationnaire de classe $\mathcal{C}^1$ et $f : \Omega \times]0, T[\rightarrow \mathbb{R}^d$ un terme de source¹. On suppose que la fonction f est de carré intégrable.

Le problème consiste à trouver une fonction $c : \Omega \times]0, T[\rightarrow [0, 1]$ qui satisfait à l'équation aux dérivées partielles dite de **transport** vue en introduction :

$$\frac{\partial c}{\partial t} + \operatorname{div}(cv) = f \quad (1.1)$$

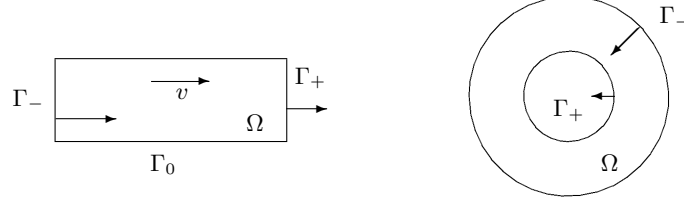
et aux conditions initiale et au bord

$$c(x, 0) = c_0(x) \quad \text{pour } x \text{ dans } \Omega \quad (1.2)$$

$$c(x, t) = c_b(x, t) \quad \text{pour } x \text{ sur } \Gamma_- = \{x \in \partial\Omega : (n(x)|v(x)) < 0 \text{ p. p.}\} \quad (1.3)$$

¹En fait, pour v , on peut se passer de la continuité. Si, comme ce sera aussi le cas ici pour la construction de STILS, on suppose v à divergence nulle, l'appartenance à $L^2(\Omega)$ suffit, pour autant que $(n|v)$ ait un sens adéquat sur $\partial\Omega$ [6].

où $n(x)$ est la normale sortante du bord $\partial\Omega$ au point x . Γ_- est la frontière influx, de même que $\Gamma_0 = \{x \in \partial\Omega : (n(x)|v(x)) = 0 \text{ presque partout}\}$ couvre la partie à flux nul et $\Gamma_+ = \{x \in \partial\Omega : (n(x)|v(x)) > 0 \text{ presque partout}\}$ la partie hors-flux ; par exemple



Pour simplifier, nous considérerons le problème homogène — ce qui suppose $c_b = 0$ — le cas général étant obtenu par superposition. Le terme ayant une dérivée partielle en t est souvent multiplié par un facteur de porosité que nous pouvons inclure dans v qui est donné.

La fonction c représente par exemple une concentration chimique transportée par un flux liquide dans le domaine Ω .

Il est bien connu que les méthodes numériques pour un problème hyperbolique de ce type peuvent être étendues convenablement au cas parabolique avec un terme d'ordre 2 $-\operatorname{div}(D\nabla c)$ où D est un tenseur de diffusion-dispersion, pourvu que ce terme d'ordre 2 soit fortement dominé par le terme de convection. Cette situation est typique pour les écoulements souterrains (Ouvrage de référence : *Hydrogéologie quantitative*, [17] §9.2 Ecoulement de fluides miscibles, §9.2.1 Lois du transport en solution des éléments conservatifs ; ou [30]).

La vitesse v peut être la solution d'équations de Navier-Stokes ou dériver d'un potentiel. Par exemple, dans le contexte d'écoulement en milieu poreux, v sera une vitesse de filtration de Darcy comme vu en introduction.

1.2 Rappels et compléments sur la méthode STILS

Rappelons les principes de la méthode STILS (Space-time integrated least-squares) pour l'équation de transport (1.1). La suite chronologique de son développement est [45], dans le cas unidimensionnel et avec un terme de diffusion, puis [48, 2, 3].

Dans une description spatio-temporelle, le temps est simplement vu comme la $(d+1)$ ème dimension spatiale. Bien sûr, la vitesse en espace-temps est $\tilde{v} = (v_1, \dots, v_n, 1)$. Le domaine en espace-temps est $\tilde{\Omega} = \Omega \times]0, T[$. La frontière influx de celui-ci est naturellement

$$\tilde{\Gamma}_- = \left\{ x \in \partial\tilde{\Omega} : (\tilde{n}(x)|\tilde{v}(x)) < 0 \right\}.$$

Il s'ensuit

$$\tilde{\Gamma}_- = \Gamma_- \times [0, T] \cup \Omega \times \{0\}.$$

La condition initiale à l'intérieur du domaine (c'est-à-dire sur $\Omega \times \{0\}$) devient une partie de la condition aux limites sur la frontière influx en espace-temps.

En partant d'une situation simplifiée de flot à divergence nulle ($\operatorname{div} v = 0$), on a

$$\underbrace{c \operatorname{div}(v)}_0 + (\nabla c | v) = \operatorname{div}(cv)$$

et donc l'équation de transport (1.1) devient $\frac{\partial c}{\partial t} + (\nabla c|v) = f$, c'est-à-dire

$$(\tilde{\nabla} c|\tilde{v}) = f \quad (1.4)$$

où $\tilde{\nabla} = (\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n}, \frac{\partial}{\partial t})$ est le gradient en espace-temps.

Posons

$$H(v, \tilde{\Omega}) = \left\{ u \in L^2(\tilde{\Omega}) : (\tilde{\nabla} u|\tilde{v}) \in L^2(\tilde{\Omega}) \right\}$$

muni de la norme du graphe de $u \mapsto (\tilde{\nabla} u|\tilde{v})$ définie par $\|u\|^2 = \|u\|_0^2 + \|(\tilde{\nabla} u|\tilde{v})\|_0^2$ et en fait générée par le produit scalaire sous-jacent $((u, w)) = (u, w)_0 + ((\tilde{\nabla} u|\tilde{v}), (\tilde{\nabla} w|\tilde{v}))_0$.

Définissons le sous-espace

$$H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-) = \left\{ u \in H(v, \tilde{\Omega}) : u = 0 \text{ sur } \tilde{\Gamma}_- \right\}$$

correspondant à une condition de Dirichlet nulle sur la frontière influx.

Définissons la fonctionnelle linéaire

$$\mathcal{A} : u \mapsto (\tilde{\nabla} u|\tilde{v}).$$

De fait, l'équation (1.4) peut s'écrire $\mathcal{A}u = f$. Dans L^2 , une solution de cette équation correspond à un zéro de la fonctionnelle convexe, positive suivante :

$$J(u) = \frac{1}{2} \int_{\tilde{\Omega}} (\mathcal{A}u - f)^2 \, dx \, dt.$$

L'approche STILS revient à chercher un minimum de J . Nous verrons que sous une condition simple de régularité sur v , on a l'existence et l'unicité d'un tel minimum, et que sous la même condition, ce minimum est 0.

1.2.1 Vers une formulation faible

Notons $[DJ(u)] : H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-) \rightarrow \mathbb{R}$ la dérivée directionnelle de la fonctionnelle J en la fonction u , définie par

$$[DJ(u)]w = \lim_{h \rightarrow 0} \frac{1}{h} \{J(u + hw) - J(u)\}. \quad (1.5)$$

La fonction w doit être vue comme une direction de dérivation. Après un calcul simple, on obtient

$$[DJ(u)]w = \int_{\tilde{\Omega}} (\mathcal{A}u - f) \mathcal{A}w \, dx \, dt.$$

Lemme 1 $[DJ(u)]$ est une forme linéaire bornée sur $H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-)$.

Preuve : par Cauchy-Schwartz,

$$|[DJ(u)]w| \leq (\|\mathcal{A}u\|_0 + \|f\|_0) \|\mathcal{A}w\|_0 \leq (\|\mathcal{A}u\|_0 + \|f\|_0) \|\mathcal{A}w\|.$$

□

Par conséquent, $[DJ(u)]$ est une dérivée au sens de Gâteaux comme définie dans le livre d'Ekeland et Temam, *Analyse convexe et problèmes variationnels* [22]².

Lemme 2 *J est convexe.*

Preuve : pour $\alpha \geq 0$ et $\beta \geq 0$ tels que $\alpha + \beta = 1$, on a

$$\alpha J(u) + \beta J(w) - J(\alpha u + \beta w) = \frac{\alpha\beta}{2} \int_{\tilde{\Omega}} (\mathcal{A}u - \mathcal{A}w)^2 \, dx \, dt \geq 0.$$

□

Corollaire 3 *Une condition suffisante pour trouver un minimum u de J est : $[DJ(u)]w = 0$ pour toute fonction w dans $H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-)$.*

Preuve :

Un résultat connu d'analyse convexe — proposition 5.4 du livre [22], ou théorème 2.6 du livre [13] — dit que J est convexe **si et seulement si**

$$J(u + w) \geq J(u) + [DJ(u)](w) \quad \text{pour toutes fonctions } u, w \in H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-).$$

□

A ce stade, on a éclairci un point crucial : le corollaire 3 nous dit que pour répondre à la question « Quelle est la solution de l'équation de transport ? » au sens des moindres-carrés ($J(u)$ min.), il suffit de résoudre le problème faible « trouver un u tel que $[DJ(u)]w = 0$ pour tout w » que nous allons formuler comme méthode d'éléments finis.

1.2.2 Equation faible de STILS et éléments finis

Maintenant, considérons la forme bilinéaire

$$\mathcal{B}(u, w) = \int_{\tilde{\Omega}} \mathcal{A}u \mathcal{A}w \, dx \, dt = \int_{\tilde{\Omega}} (\tilde{\nabla}u|\tilde{v})(\tilde{\nabla}w|\tilde{v}) \, dx \, dt$$

et la forme linéaire

$$\mathcal{L}(w) = \int_{\tilde{\Omega}} f \cdot \mathcal{A}w \, dx \, dt = \int_{\tilde{\Omega}} f \cdot (\tilde{\nabla}w|\tilde{v}) \, dx \, dt.$$

On a $DJ(u) = 0$, donc solution au sens des moindres-carrés, si et seulement si

$$\mathcal{B}(u, w) = \mathcal{L}(w) \quad \text{pour tout } w \in H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-). \quad (1.6)$$

C'est l'équation faible sur laquelle se base la méthode STILS. Les formes $\mathcal{B}$ et $\mathcal{L}$ sont continues puisque $|\mathcal{B}(u, w)| \leq \|u\| \|w\|$ et $|\mathcal{L}(w)| \leq \|f\|_0 \|w\|$.

Enonçons le *problème faible* de STILS :

$$\text{Trouver } u \in H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-) \text{ telle que (1.6) est satisfaite.} \quad (1.7)$$

²Voir également l'*Introduction au calcul des variations* de B. Dacorogna [14]

Voici le résultat-clé de la théorie autour de STILS, dû à Pascal Azérad [2].

Théorème 4 *Pour les hypothèses mentionnées au début du chapitre (§1.1), et si de plus v est à divergence nulle ($\operatorname{div} v = 0$), alors le problème faible (1.7) possède une unique solution $u = u_{\text{faible}}$.*

Ce théorème a une double portée :

- La solution de la formulation STILS existe, est unique et coïncide presque partout avec la solution exacte de l'équation de transport (1.1).
- Dans la réalité des ordinateurs binaires, on a bien sûr une approximation, avec des données et des solutions en quantité discrètes. La méthode des éléments finis consiste à prendre un sous-espace fonctionnel de dimension finie et de résoudre (1.7) sous forme de système d'équations linéaires. Alors, cette méthode pour la formulation STILS donnera automatiquement la *meilleure* approximation au sens des moindres-carrés pour ce sous-espace. La qualité dépend bien sûr du type de fonctions de base et du maillage de discrétisation. Si par hasard la solution exacte (en espace-temps) trouve sa place dans ce sous-espace, la méthode *numérique approchée* livrera la solution *exacte*!

Preuve 4 :

Les composantes de la preuve se trouvent dans la thèse de P. Azérad [2] ainsi que dans la note subséquente [3].

D'abord, le champ de vitesse $\tilde{v}$ est borné dans sa composante temporelle (qui vaut 1) et a donc une projection constante sur le vecteur constant $(0, \dots, 0, 1)$. Cela assure (proposition 7 dans [2], proposition 1 dans [3]) que $\tilde{\Omega}$ est rempli par les caractéristiques de la manière suivante : il existe un $S > 0$ tel que pour presque tout point $\tilde{x} = (x, t)$ de $\tilde{\Omega}$, il existe une courbe intégrale $\xi : [0, S] \rightarrow \tilde{\Omega}$ de

$$\frac{\partial \xi}{\partial s} = \tilde{v}(\xi)$$

qui relie $\tilde{x}$ à la frontière influx en espace-temps : $\xi(0) \in \tilde{\Gamma}_-$ et $\xi(s) = \tilde{x}$ pour un certain $s \in [0, S]$.

Puis, cette propriété de flux $\tilde{\Omega}$ -remplissant permet, via un redressement du champ $\tilde{v}$ le long de t (ici) d'obtenir (théorème 14 dans [2], théorème 1 dans [3]) une constante $C > 0$ telle que pour tout u dans $H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-)$ l'inégalité « courbe » de Poincaré a lieu

$$\|u\|_0 \leq C \|(\tilde{\nabla} u | \tilde{v})\|_0. \quad (1.8)$$

Une conséquence immédiate est que la forme $\mathcal{B}$ est **coercitive**

$$\|u\|^2 = \|u\|_0^2 + \|(\tilde{\nabla} u | \tilde{v})\|_0^2 \leq (C^2 + 1) \|(\tilde{\nabla} u | \tilde{v})\|_0^2 = (C^2 + 1) \mathcal{B}(u, u).$$

Le théorème de Lax-Milgram donne alors l'existence et l'unicité d'une solution u_{faible} au problème faible (1.7). Ainsi, nous avons un unique minimum u_{faible} à la fonctionnelle J . $\square$

Corollaire 5 *De surcroît, si la source f est nulle, cette solution faible u_{faible} coïncide avec la solution c du problème fort correspondant à l'équation (1.4) et à la condition $c = 0$ sur $\tilde{\Gamma}_-$.*

Remarque : Il est plus intéressant de considérer le cas $c \neq 0$ sur $\tilde{\Gamma}_-$.

Preuve 5 : Par $\tilde{\Omega}$ -remplissement, les caractéristiques donnent une solution exacte $c : \tilde{\Omega} \rightarrow \mathbb{R}$ au problème fort (1.4) — la raison est simple, voir un peu plus loin au §2.1.3. Comme par

hypothèse la source f est nulle, cette solution c est nulle presque partout puisque constante sur les caractéristiques, et ce sur le borné $\tilde{\Omega}$.

Ainsi $c \in H(v, \tilde{\Omega})$. Puisque $J(c) = 0$, ce qui est le minimum absolu pour une fonctionnelle positive, on a clairement $u_{\text{faible}} = c$. $\square$

Remarque : Ces résultats sont en général faux dans le cas **stationnaire**, où $\tilde{\Gamma}_- = \Gamma_-$. Par exemple, un point x au voisinage duquel v est nulle n'est l'aboutissement d'aucune caractéristique venant de la frontière (en espace) et est de ce fait indéterminé. C'est alors une « zone morte ». Donc, on n'a pas automatiquement Ω -remplissement dans le cas stationnaire. En espace-temps, il n'y a pas ce problème. Au pire, sur de tels points x avec $v(x, t) = 0$, on a une caractéristique « verticale » qui traverse le temps sans quitter ce point : c'est $\{x\} \times [0; T]$.

1.2.3 Une version marche-en-temps de STILS

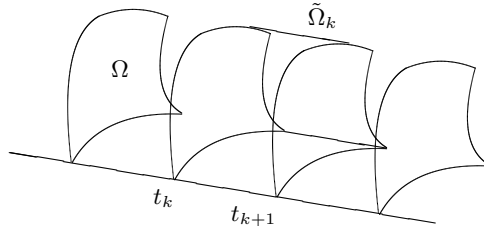
Une approche complètement en espace-temps peut mener à des tailles matricielles problématiques. De plus, de par sa composante temporelle 1, le champ de vitesse $\tilde{v}$ transporte les valeurs de $c(x, t)$ ³ uniquement du passé vers le futur (heureusement !). Il vient donc l'idée de résoudre à chaque pas de temps un problème « local en temps » où la condition initiale est l'état au pas de temps courant et l'inconnue est l'état au pas de temps suivant. Une réalisation particulière à une dimension spatiale est faite dans [2, 48]. Voici une version générale en espace — présentée dans [18].

Pour l'espace, choisissons une famille libre $\{\psi_1 \dots \psi_N\}$ de fonctions de $H(v, \Omega)$. On pense bien sûr aux éléments finis de Galerkin et à des fonctions de base dont la j ème, ψ_j , interpole, par exemple polynomialement sur chaque élément, les couples $(x_1, 0), \dots, (x_j, 1), \dots, (x_N, 0)$ où $x_1 \dots x_N$ sont les nœuds du maillage. Pour le degré 1 ce sont les fonctions-pyramides de sommet 1 sur le nœud x_j . Notons V_h l'espace de fonctions engendré par la base $\{\psi_1 \dots \psi_N\}$.

Quant au temps, nous considérerons successivement les intervalles $[t_0, t_1], [t_1, t_2], \dots$ pour résoudre (1.7) dans une approche marche-en-temps, ce que permet la composante 1 de $\tilde{v}$. Définissons une base affine sur l'intervalle $[t_k, t_{k+1}]$

$$\begin{aligned} a_k(t) &= \frac{(t_{k+1} - t)}{\tau} \\ a_{k+1}(t) &= \frac{(t - t_k)}{\tau} \end{aligned} \tag{1.9}$$

avec $\tau = t_{k+1} - t_k$. On considèrera le **problème local** dans le domaine $\tilde{\Omega}_k = \Omega \times]t_k, t_{k+1}[$, avec une condition initiale sur $\Omega \times \{t_k\} \subseteq (\tilde{\Gamma}_k)_- = \Gamma_- \times]t_k, t_{k+1}[\cup \Omega \times \{t_k\}$.



³le lecteur attentif aura remarqué l'usage de la lettre plus « mathématicienne » u pour la solution c de l'équation de transport (1.4)

Soit E_2 le sous-espace de dimension 2 de $\mathcal{C}[t_k, t_{k+1}]$ engendré par les fonctions a_k et a_{k+1} . Une approximation du problème (1.7) consiste à chercher une fonction c_h dans l'espace fonctionnel $H_{0,h} = V_h \otimes E_2 \cap H_0(v, \tilde{\Omega}_k, \tilde{\Gamma}_{k,-})$ telle que

$$\mathcal{B}(c_h, w_h) = \mathcal{L}(w_h) \quad \text{pour tout } w_h \in H_{0,h}. \quad (1.10)$$

Plus explicitement, on a l'équation

$$\int_{\Omega} \int_{t_k}^{t_{k+1}} \left(\tilde{\nabla} c_h | \tilde{v} \right) \left(\tilde{\nabla} w_h | \tilde{v} \right) dt dx = \int_{\Omega} \int_{t_k}^{t_{k+1}} f \cdot \left(\tilde{\nabla} w_h | \tilde{v} \right) dt dx. \quad (1.11)$$

pour tout $w_h \in H_{0,h}$.

Pour résoudre (1.11) par une méthode marche-en-temps, on choisira w_h indépendante de a_k . Ainsi w_h s'annule en $t = t_k$. Pour motiver ce choix, on regarde le problème avec condition de Dirichlet sur $\tilde{\Gamma}_{k,-}$. Si la solution approchée c_h est dans $H_{0,h}$, cela veut dire que la condition est homogène (nulle sur la toute frontière influx en espace-temps). Dans ce contexte, c_h s'écrit

$$c_{0,h}(x, t) = c_h(x, t) = \sum_{j=1}^N \psi_j(x) a_{k+1}(t) c_j^{k+1}.$$

Il suffit, pour les fonctions-test, de considérer les fonctions de base de $H_{0,h}$:

$$w_h(x, t) = \psi_i(x) \cdot a_{k+1}(t), \quad i \in \{1 \dots N\}, \quad x_i \notin \Gamma_-.$$

Dans le cas inhomogène (en temps, toujours), on a un vecteur c^k qui contient l'état connu au temps $t = t_k$.

$$c_h(x, t) = \sum_{j=1}^N \psi_j(x) \left(a_k(t) c_j^k + a_{k+1}(t) c_j^{k+1} \right). \quad (1.12)$$

On peut alors écrire le système linéaire correspondant au problème (1.10) :

$$\sum_{j=1}^N \mathcal{B}(\psi_j a_{k+1}, \psi_i a_{k+1}) \cdot c_j^{k+1} = \mathcal{L}(\psi_i a_{k+1}) - \sum_{j=1}^N \mathcal{B}(\psi_j a_k, \psi_i a_{k+1}) \cdot c_j^k. \quad (1.13)$$

Dans (1.13), on peut calculer directement les intégrales en temps sous-jacentes à la forme $\mathcal{B}$. Il reste celles en espace, qui dépendent de la géométrie du problème. A la clé, on a le système linéaire d'une méthode d'éléments finis marche-en-temps de degré 1, « STILS-TMQ1 » :

$$\begin{aligned} & \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{\tau}{3} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{1}{2} (\nabla \psi_j | v) \psi_i + \frac{1}{2} \psi_j (\nabla \psi_i | v) + \frac{1}{\tau} \psi_j \psi_i \right] dx \\ &= \sum_{j=1}^N c_j^k \int_{\Omega} \left[\frac{-\tau}{6} (\nabla \psi_j | v) (\nabla \psi_i | v) - \frac{1}{2} (\nabla \psi_j | v) \psi_i + \frac{1}{2} \psi_j (\nabla \psi_i | v) + \frac{1}{\tau} \psi_j \psi_i \right] dx + l_i \end{aligned} \quad (1.14)$$

pour tous i dans $\{1 \dots N\}$, avec le terme de source

$$l_i = \mathcal{L}(\psi_i a_{k+1})$$

Remarque : Si v est nulle, il ne reste dans le système (1.14) que les termes $\psi_j \psi_i$, ce qui fait $c^{k+1} = c^k$: solution constante.

Matriciellement, le système est de la forme

$$Ac^{k+1} = Bc^k + l \quad (1.15)$$

avec

$$\begin{aligned} a_{ij} &= \mathcal{B}(\psi_j a_{k+1}, \psi_i a_{k+1}) \\ b_{ij} &= -\mathcal{B}(\psi_j a_k, \psi_i a_{k+1}). \end{aligned} \quad (1.16)$$

Remarque : A est symétrique par symétrie de $\mathcal{B}$. Mieux : elle est définie positive grâce à la coercitivité de $\mathcal{B}$. En effet, pour un vecteur $z \in \mathbb{R}^N$, on a par bilinéarité de $\mathcal{B}$ et par l'inégalité de Poincaré courbe

$$(Az|z) = \mathcal{B} \left(\sum_{j=1}^N \psi_j(x) a_{k+1}(t) z_j, \sum_{i=1}^N \psi_i(x) a_{k+1}(t) z_i \right) \geq \frac{1}{C^2 + 1} \left\| \sum_{j=1}^N \psi_j(x) a_{k+1}(t) z_j \right\|^2 \quad (1.17)$$

Le fait que A soit symétrique et définie positive est un avantage considérable puisqu'on peut appliquer une méthode appropriée pour résoudre le système (1.15) et ne calculer et conserver que la moitié de la matrice. Par exemple, on pourra appliquer une méthode de gradient conjugué au lieu d'une méthode de gradient bi-conjugué, plus fragile et plus lente.

Une condition non triviale de Dirichlet sur la frontière influx en espace Γ_- sera traitée de la manière usuelle par superposition d'une fonction c_0 qui s'annule presque partout sauf peut-être sur Γ_- . Après discrétisation, cela signifie que les valeurs imposées aux nœuds seront toutes nulles hormis sur des nœuds qui sont sur Γ_- . Sous forme matricielle, le système linéaire s'écrit

$$Ac^{k+1} = Bc^k - Ac_0^{k+1} + Bc_0^k.$$

Chapitre 2

Comparaisons entre STILS et d'autres méthodes

Pour mener à bien la comparaison de STILS (marche en temps) avec d'autres méthodes classiques, nous sommes amenés à faire quelques rappels sur celles-ci — même si souvent ce sont des « classiques ».

2.1 Eléments finis continus en espace

Nous nous contenterons, pour ces méthodes très connues, de formuler les équations faibles et les composantes des matrices de masse et de rigidité. On se référera aux ouvrages de base ([32, 50, 27]).

Définissons d'abord $H_0 = \{u \in H^1(\Omega) : u = 0 \text{ sur } \Gamma_-\}$ l'espace des fonctions de base H^1 ayant une condition de Dirichlet nulle sur la frontière influx.

Ainsi, le problème faible pour l'équation de transport à divergence nulle (1.4) peut s'écrire : trouver $c \in L^2(0; T; H^1(\Omega) \cap H_0)$ tel que

$$\int_{\Omega} \left[(\nabla c | v) + \frac{\partial c}{\partial t} - f \right] w \, dx = 0 \quad \text{pour tout } w \in H_0. \quad (2.1)$$

La discrétisation spatiale, faite par le moyen d'une famille linéairement indépendante de fonctions $\{\psi_1 \dots \psi_N\}$ et d'une approximation de $c(x, t)$ par $\sum_{j=1}^N c_j(t) \psi_j(x)$ conduit au système différentiel

$$Rc + Mc' = b \quad (2.2)$$

avec

$$\begin{aligned} r_{ij} &= (\psi_i, (\nabla \psi_j | v)) \\ m_{ij} &= (\psi_i, \psi_j). \end{aligned} \quad (2.3)$$

Sans modification, cette méthode de Galerkin simple est notoirement inadaptée, à cause de son instabilité, à un problème ayant un terme hyperbolique dominant, surtout au voisinage de discontinuités — c'est aussi sûrement un défaut de coercitivité, cf la discussion au §3.1 sur la formulation faible (3.7). C'est pourquoi nous avons, pour cette comparaison, implémenté des

améliorations, désormais classiques, telles que de la diffusion artificielle le long des lignes de courant (streamline diffusion) et un raffinement de celle-là (shock capturing) qui tient compte des endroits du domaine où une solution peut avoir de forts gradients. Rappelons brièvement ces améliorations.

2.1.1 Streamline diffusion

On sait qu'ajouter de la diffusion artificielle contribue à réduire les oscillations numériques indésirées, jusqu'à les supprimer. Mais, plus on en ajoute, plus on fausse le problème de base qui est convectif et tout sauf diffusif. Il vaudrait donc mieux filtrer la diffusion transversale aux lignes de courant pour éviter d'« infecter » les lignes voisines avec des valeurs incorrectes. La fonction-test w est complétée par un terme $\delta(\nabla w|v)$, δ étant un facteur arbitraire à fixer. Des recherches sont encore en cours pour obtenir la valeur de δ la plus petite possible tout en étant efficace. Les valeurs usuelles sont de l'ordre du diamètre d'un élément cf §9.7 de [32]. L'équation faible correspondante est

$$\int_{\Omega} \left[(\nabla c|v) + \frac{\partial c}{\partial t} - f \right] [w + \delta(\nabla w|v)] \, dx = 0 \quad \text{pour tout } w \in H_0. \quad (2.4)$$

C'est l'approche de Petrov-Galerkine connue aussi sous le nom de *Streamline diffusion* ou encore l'abréviation SUPG (Streamline Upwind Petrov Galerkin). En écrivant

$$\psi_i^\delta = \psi_i + \delta(\nabla \psi_i|v), \quad (2.5)$$

le système discrétisé en espace devient

$$R^\delta c + M^\delta c' = b \quad (2.6)$$

avec

$$\begin{aligned} r_{ij}^\delta &= \left(\psi_i^\delta, (\nabla \psi_j|v) \right) \\ m_{ij}^\delta &= \left(\psi_i^\delta, \psi_j \right). \end{aligned} \quad (2.7)$$

On a de fait un terme de type diffusif avec $\delta \int_{\Omega} (\nabla c|v)(\nabla w|v) \, dx$. À noter que pour cette équation de transport (1.4) purement convective, donc sans terme de diffusion, l'approche GLS (Galerkin Least Squares) décrite dans [28] coïncide avec *Streamline diffusion*.

2.1.2 Shock capturing [29, 32, 33]

L'idée est de compléter l'effet précédent lorsqu'il y a de fortes variations locales de c en direction des lignes de courant. Posons

$$v_{//} = \begin{cases} \frac{(v|\nabla c^H)}{\|\nabla c^H\|_2^2} \nabla c^H, & \nabla c^H \neq 0 \\ 0, & \nabla c^H = 0 \end{cases} \quad (2.8)$$

la projection de v sur ∇c^H , où $c^H = \sum_j c_j \psi_j$ est l'approximation courante en espace. On combine ensuite la diffusion le long de v et celle le long de $v_{//}$, avec des poids appropriés. En

réutilisant la notation précédente, on peut écrire les fonctions de base modifiées

$$\psi_i^\delta = \psi_i + \delta_1 (\nabla \psi_i | v) + \delta_2 (\nabla \psi_i | v_{//}). \quad (2.9)$$

A nouveau, il n'y a pas (encore) de recette infaillible pour déterminer δ_1 et δ_2 — d'où l'intérêt de l'approche STILS au moins pour la convection pure.

Notons également que le calcul de $v_{//}$ demande le recalcul de M et R à chaque pas de temps, ce qui peut ralentir quelque peu la résolution numérique.

2.1.3 Une méthode de caractéristiques

Cette méthode « ancestrale » a le défaut de ne résoudre *a priori* le problème que sur les lignes de courant et non sur des points choisis, comme les nœuds d'un maillage d'éléments finis. Elle a en revanche le mérite de donner la valeur exacte de la solution le long des lignes de courant. On a ainsi un outil de comparaison pour savoir si une autre méthode connaît du retard ou de l'avance sur la convection. De plus, une méthode de caractéristiques permet de détecter les cas où le champ v est mal approché, signe, par exemple, d'un mauvais maillage.

Considérons la courbe $(x(t), t)$ qui vérifie $x'(t) = v(x(t), t)$. En d'autres termes, la courbe (dite caractéristique) intègre le champ de vitesses v . Soit $c(x, t)$ une solution de $\operatorname{div}(-cv) = \frac{\partial c}{\partial t}$ sur $\Omega \times]0, T[$. Alors nous avons, comme évolution de c sur le parcours de $(x(t), t)$:

$$\begin{aligned} \frac{dc}{dt}(x(t), t) &= \sum_{j=1}^d \frac{\partial c}{\partial x_j} \cdot \frac{dx_j}{dt} + \frac{\partial c}{\partial t} = \sum_{j=1}^d \frac{\partial c}{\partial x_j} \cdot v_j + \frac{\partial c}{\partial t} \\ &= (\nabla c | v) + \frac{\partial c}{\partial t} = \operatorname{div}(cv) + \frac{\partial c}{\partial t} - c \operatorname{div}(v) \\ &= f(x(t), t) - c(x(t), t) \cdot \operatorname{div}(v(x(t), t)) \end{aligned} \quad (2.10)$$

en vertu de l'équation de transport (1.1). Cela montre que le long des caractéristiques, c est une solution d'une EDO simple. De plus, pour un flux v à divergence nulle (par exemple, dérivant d'un potentiel harmonique) et si la source f est nulle, c est même constante le long des caractéristiques.

L'implémentation numérique est faite au moyen d'un flot de particules. Chaque particule porte une valeur de c déterminée par une condition initiale ou au bord. Les caractéristiques sont intégrées à l'aide de la méthode de Dormand-Prince à pas variable telle qu'étudiée dans le livre de Hairer, Nørsett et Wanner, *Solving ordinary differential equations I*, [25].

La méthode de Dormand-Prince a l'avantage d'être très précise et aussi très rapide en raison de son pas variable. Le principe est d'ajuster le pas de temps à l'aide du calcul de l'erreur entre deux extrapolations (ici une d'ordre 7 et une autre d'ordre 8). On trouve une analyse détaillée de cette méthode dans [25] — et d'autres méthodes encore plus poussées. A titre de comparaison, d'après nos expériences, une méthode de Runge-Kutta classique d'ordre 4 est beaucoup trop imprécise ; de plus le pas fixe en fait un moyen d'intégration beaucoup trop lent. Quand le champ v est obtenu approximativement par éléments finis, il est nécessaire, pour connaître la vitesse d'une particule sur son trajet, dans quel élément elle se situe à un moment donné. Dans ce but, nous avons écrit un algorithme de recherche dichotomique permettant de retrouver les coordonnées *locales* de la particule dans un élément candidat — ou de conclure qu'elle n'y est pas. Une fois l'élément et la position locale est trouvée, on obtient facilement v en interpolant les valeurs calculées aux nœuds de l'élément à l'aide des fonctions de base.

Aux pas de temps principaux t_k , les valeurs de $c(x, t_k)$ aux nœuds sont estimées à l'aide des particules avoisinantes. Une telle opération est nécessaire car rien n'oblige une caractéristique à relier deux nœuds, ce qui donnerait la solution de manière univoque ¹. Sur ce point, on voit qu'il vaudrait mieux parler d'une méthode des caractéristiques, car les façons de définir la pondération attribuée à chaque particule pour contribuer à la valeur d'un nœud sont innombrables. L'important est d'avoir des poids non-négatifs et 1 comme somme des poids. On peut voir ces poids comme des probabilités et la valeur au nœud, une espérance. Comme les nouvelles particules apparaissant au cours du temps sur la frontière influx sont distribuées aléatoirement, on a affaire en fait à une forme de méthode de Monte-Carlo. Cette approche piétonne ne se réfère cependant pas aux théories connues pour résoudre des E.D.P. par une approche de Monte-Carlo [36].

En répartissant parcimonieusement les particules — c'est à dire en les concentrant près des discontinuités de la condition initiale — on obtient parfois des performances intéressantes. L'inconvénient majeur que nous avons rencontré est que les particules se font rares aux endroits où on a le plus besoins d'elles : les zones à fort gradient de pression, donc à grande vitesse v . C'est naturel, puisqu'elles passent très vite dans de telles zones, leur densité (nombre / volume) est faible.

Au sujet des caractéristiques on peut lire des ouvrages sur les lois de conservation [23, 24, 41].

2.2 Expériences numériques

Les calculs ont été effectués sur un Compaq AXP 8420 sous OpenVMS 7.1-2. La précision est double (15 chiffres significatifs). Tous les tests de bornes ont été supprimés et l'optimisation en temps a été activée pour avoir la meilleure mesure du temps algorithmique possible. A cause de facteurs tels que mémoire cache, mémoire virtuelle et de choix algorithmiques, les chronométrages en temps CPU doivent être pris avec précaution. Il y a notamment un « coût fixe » de préparation des données allant d'une fraction de seconde à une minute selon les exemples traités.

En vue de permettre la reproductibilité, donnons les détails-clés sur l'implémentation numérique.

- Les δ_1, δ_2 dans les méthodes streamline diffusion et shock capturing sont de l'ordre du diamètre maximal des éléments, que nous noterons δ_M (variable `diamax` dans le programme) ; plus précisément les valeurs $\delta_1 = \frac{\delta_M}{\sqrt{2}} \cdot 0.6$ et $\delta_2 = \frac{\delta_M}{\sqrt{2}} \cdot 0.8$ ont été fixées.
- Les systèmes linéaires sont résolus de deux manières selon leur format de stockage :
 - **Format de matrice bande** : on sait que seul un certain nombre de diagonales au-dessus et au-dessous de la diagonale principale sont susceptibles d'être non-nulles. On ne mémorise donc que celles-là et on peut faire une décomposition de Cholesky $A = L^T L$ dans le cas symétrique défini positif (routine DGBTRF de la bibliothèque LAPACK) ou $A = LU$ dans le cas général (routine DPBTRF). Ces décompositions sont possibles pour ce format de mémorisation car elles conservent la largeur de bande.
 - **Format de matrice creuse** : en plus de la largeur de bande, on sait en outre que la plupart des composantes de la matrice A sont nulles. Ce sont dans notre contexte les

¹On peut donc imaginer un générateur de mailles qui propage des nœuds placés sur la frontière influx à l'aide des caractéristiques. Cela ne vaut que pour un champ v stationnaire et il faut compenser les grandes valeurs de v par des pas de temps plus petits pour éviter un maillage trop grossier aux points cruciaux.

a_{ij} pour lesquels les nœuds i et j ne sont pas dans un même élément. La résolution se fait par méthodes itératives, ce qui ne nécessite aucune transformation sur A . Pour une matrice symétrique, on fait appel à la méthode du gradient conjugué et pour une matrice asymétrique, à la méthode du gradient bi-conjugué stabilisée. Ces méthodes itératives sont plus rapides que la résolution sur format bande de $LUx = y$ mais parfois plus délicates (voir A.1).

- Les méthodes d'éléments finis classiques sont discrétisées en temps sur un schéma BDF (backward differentiation formula) d'ordre 2, aussi appelé schéma de Gear. Pour une équation différentielle ordinaire scalaire $u'(t) = f(u(t), t)$ ce schéma s'écrit

$$\frac{3}{2}u^{k+1} - 2u^k + \frac{1}{2}u^{k-1} = \tau f(u^{k+1}).$$

Le schéma BDF pour le système $Rc + Mc' = b$ est donc

$$\left(\frac{3}{2\tau}M + R\right)c^{k+1} = \frac{1}{\tau}M\left(2c^k - \frac{1}{2}c^{k-1}\right) + b^{k+1}.$$

Comme tous les schémas BDF jusqu'à l'ordre 6, il est stable (Théorème 3.4, §III.2 dans [25]).

- L'intégration des fonctions de base est faite à l'aide d'une méthode de Gauss à trois points : $-\sqrt{3/5}, 0, \sqrt{3/5}$, racines du polynôme de Legendre $\frac{1}{2}(5x^3 - 3x)$ sur $[-1, 1]$. La méthode est appliquée en chaque dimension spatiale, ce qui correspond à une intégrale multiple. Rappelons que la méthode de Gauss à trois points est *exacte* pour des polynômes au moins jusqu'au degré 5 ; ainsi on a des composantes de matrices correctes pour des éléments de géométrie parallélépipédique et de type Q_1 et Q_2 .
- Lorsque le champ v est solution d'une équation d'écoulement (1), on résout celle-là par une méthode de Galerkin simple en espace et, à moins qu'elle ne soit stationnaire, par le même schéma BDF d'ordre 2 en temps (schéma de Gear).

2.2.1 Cylindre fendu

Inaugurons les essais numériques sur l'équation de transport (1.1) par cet exemple qui provient d'un article de P. Hansbo [26] sur la méthode CSD (characteristic streamline diffusion). Nous prendrons soin de garder les paramètres de discrétisation pour faire la comparaison — cf la remarque p. 33.

Le domaine est le carré $] -1, 1[^2$ et discrétisé en 100×100 éléments de type Q_1 (sur le domaine, les fonctions sont continues et, sur chaque élément, au plus linéaires en chaque variable).

La condition initiale est

$$c(x, y, 0) = c_0(x, y) = \begin{cases} 1 & \text{si } (|x| > 0.05 \text{ ou } y > 0.7) \text{ et } R \leq 0.3 \\ 0 & \text{ailleurs} \end{cases}$$

où $R = \sqrt{x^2 + (y - 0.5)^2}$.

Le champ de vitesse est une rotation de vitesse proportionnelle au rayon $v(x, y, t) = (-y, x)$ dont les courbes intégrales sont évidemment des cercles.

Le temps final est $T = 2\pi$.

Il faut s'attendre à un phénomène d'oscillations aux alentours des discontinuités lors de la résolution numérique, du fait qu'on projette une fonction discontinue c dans un sous-espace

à dimension finie de $H(v, \tilde{\Omega})$. C'est un phénomène de Gibbs analogue à ce qu'on observe sur les développements en sommes partielles des séries de Fourier sur $L^2(\mathbb{R})$.

Outre le résultat proprement dit, on prendra note de la variation relative qui est la variation maximale de la concentration intégrale sur tous les pas de temps, divisée par la valeur minimale. C'est fondamental pour voir à quel degré les algorithmes sont conservatifs.

Ici, idéalement la variation devrait être nulle puisqu'il n'y a ni entrée ni sortie de concentration : la solution exacte à l'intérieur du disque unité est simplement une rotation de la condition initiale ; on a donc même $c = 0$ partout sur le bord spatial $\partial\Omega$. La source f étant nulle, le théorème de la divergence assure une variation nulle :

$$\frac{\partial}{\partial t} \left(\int_{\Omega} c \, dx \right) = \int_{\Omega} \frac{\partial}{\partial t} c \, dx = - \int_{\Omega} \operatorname{div}(cv) \, dx = - \int_{\partial\Omega} c (v|n) \, d\gamma = 0. \quad (2.11)$$

Le flot v est à divergence nulle, donc $\operatorname{div}(cv) = c \operatorname{div} v + (\nabla c|v) = (\nabla c|v)$ l'équation de transport $\frac{\partial c}{\partial t} + \operatorname{div}(cv) = 0$ (1.1) coïncide avec celle sur laquelle se basent les méthodes numériques : $(\tilde{\nabla} c|\tilde{v}) = 0$ (1.4). Ainsi, on devrait converger, en raffinant le maillage, vers une variation nulle. On verra plus loin dans quelle mesure on a cette convergence (§2.2.2).

Voici les résultats numériques pour l'équation (1.1) par les méthodes décrites ci-dessus.

Méthode	# pas de temps	min	max	variation relative	temps CPU (format ma- trices bande)	temps CPU (format ma- trices creuses)
Galerkine	40 x 20	-0.234	1.213	$6.86 \cdot 10^{-3}$	2h13	0h41
SUPG	40 x 20	-0.147	1.080	$8.01 \cdot 10^{-5}$	2h14	0h43
SUPG-DC	40 x 20	-0.139	1.064	$8.35 \cdot 10^{-6}$	3h20	1h29
STILS-TMQ1	40 x 20	-0.261	1.239	$4.29 \cdot 10^{-4}$	2h18	0h33
C-Convection	≤ 100 x 20	0.000	1.000	$1.41 \cdot 10^{-2}$	0h58	

TAB. 2.1 – Résultats – cylindre fendu

Le « $\leq$ » dans le nombre de pas de temps pour la méthode des caractéristiques est dû à l'usage de pas de temps variables pour intégrer les caractéristiques. Il est donc sans relation avec les pas de temps pour les méthodes d'éléments finis. La méthode qui a un terme de *shock-capturing*, SUPG-DC, présente une surcharge de temps CPU due au recalcul des matrices qui évoluent au fil des itérations en temps à cause du terme $(\nabla\psi_i|v_{//})$ de (2.9).

Voici un aperçu de la solution numérique, calculée par les différentes méthodes en jeu, au temps final $T = 2\pi$ où la solution exacte coïncide avec la condition initiale. On montre d'abord le champ de vecteurs v et les courbes de niveau de la concentration c initiale (pour $t = 0$). On a $c = 0$ hors de la forme et 1 dans la forme (Fig. 0 et 1).

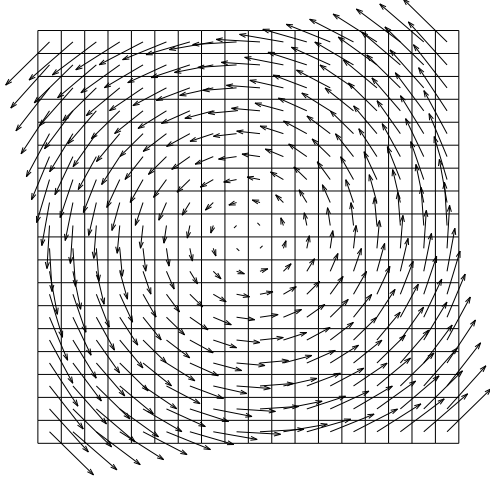


Fig. 0 Champ de vitesses v (maillage plus grossier)

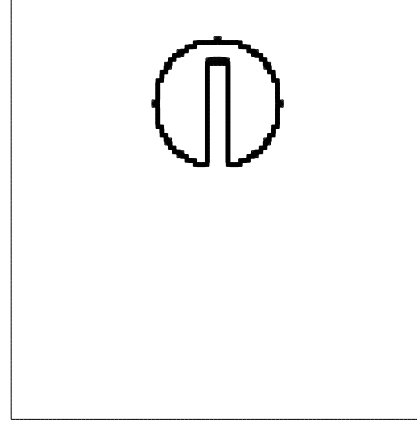


Fig. 1 Condition initiale — courbes de niveau

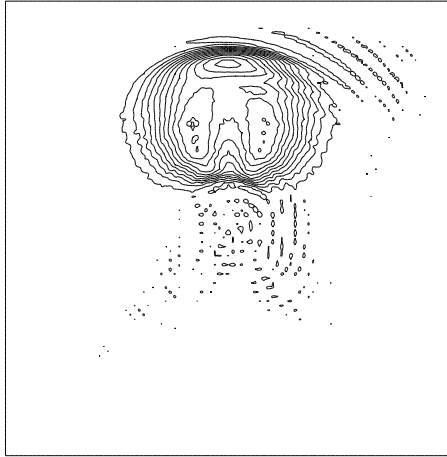


Fig. 2 Méthode de Galerkin simple

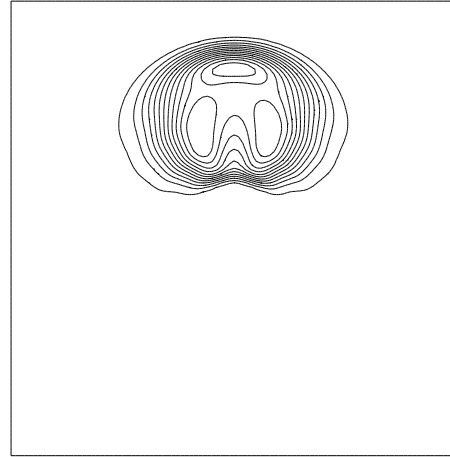


Fig. 3 Str. diff., $\delta = \frac{\delta_M}{\sqrt{2}} \cdot 0.6$

Remarque : Les méthodes qui mettent en œuvre une diffusion artificielle le long des lignes de courant pour gommer les oscillations (figures 3 et 4) ont l'effet escompté, bien visible ici. Ces résultats sont du reste très semblables à ceux produits par la méthode CSD — characteristic streamline diffusion —, pour la même finesse en espace (100x100 nœuds) : voir figures 8 et 9 de [26].

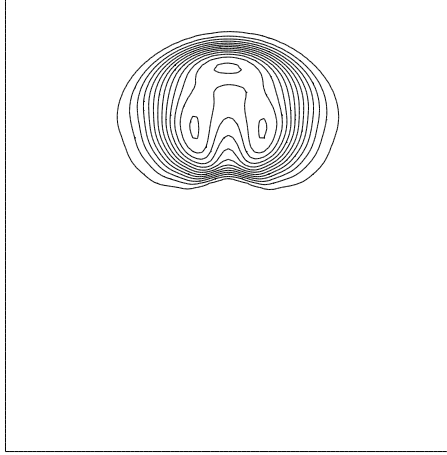


Fig. 4 Str. diff. et shock capturing,
 $\delta_1 = \frac{\delta_M}{\sqrt{2}} \cdot 0.6$, $\delta_2 = \frac{\delta_M}{\sqrt{2}} \cdot 0.8$

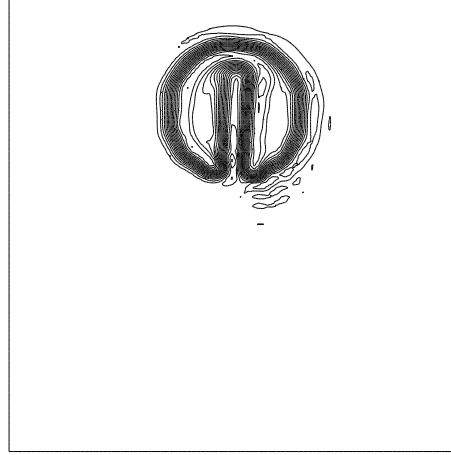


Fig. 5 STILS-TMQ1

La méthode STILS-TMQ1 est ici remarquable (figure 5). La forme du cylindre est bien préservée. Néanmoins, une comparaison entre des schémas semi-discrétisés (Ici Galerkin, Streamline diffusion) et un schéma en espace-temps, marche-en-temps (Ici STILS-TMQ1) est quelque peu biaisé, même en ayant exactement des mailles identiques en espace et en temps. En effet, le réglage entre l'approche de Galerkin et les moindres-carrés est une chose ; le choix de discrétiser en temps sur un schéma semi-discrétisé ou de marcher en temps sur une formulation en espace-temps en est une autre. Ces deux aspects peuvent être ajustés séparément ; il faut donc examiner et comparer leurs variantes systématiquement et séparément, même si l'approche STILS les réunit de manière naturelle. Nous développerons donc une méthodologie de comparaison plus fine au chapitre 3.

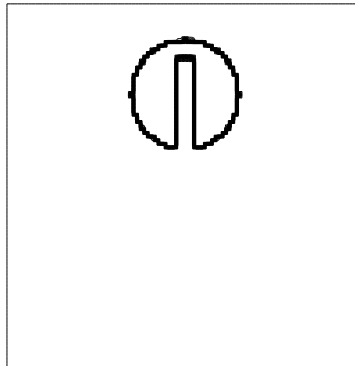


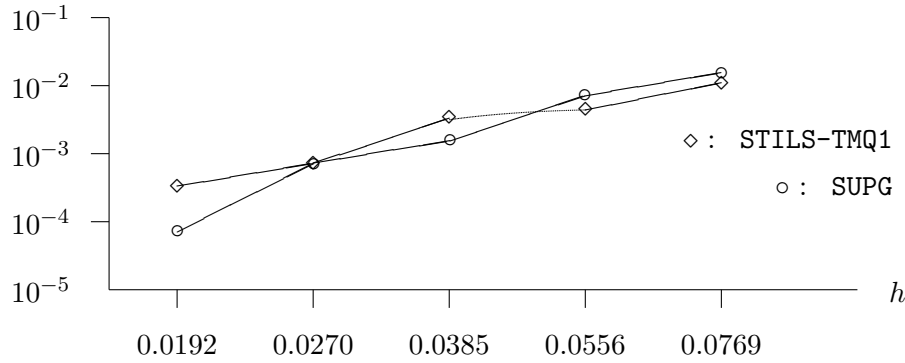
Fig. 6 Méthode des caractéristiques

Quant à la méthode des caractéristiques, elle a le loisir, dans cet exemple, de développer toute sa précision. Grâce à la nature du champ v , le faisceau de caractéristiques est dans une situation favorable et produit la solution exacte pour chaque pas de temps (figure 6). En effet, la densité du flot de particules est préservée, puisque v n'engendre qu'une rotation.

2.2.2 Conservativité

Etudions l'évolution de la conservativité (variation de la concentration intégrale de la table 2.1) en fonction de la taille d'une maille (h) et du pas de temps (τ). Nous comparons STILS-TMQ1 à SUPG. D'abord, observons l'évolution en fonction de h , pour un pas de temps $\tau = \frac{2\pi}{800} \simeq 0.00785$. Posons $C(x) = \frac{\tau \|v(x)\|_2}{h}$, le nombre de Courant et $C_M = \max_{x \in \Omega} |C(x)|$ son maximum sur le domaine. On rappellera au chapitre suivant pourquoi il est sain d'avoir $C_M \leq 1$. Pour des tailles h grossissant par pas de $\sqrt{2}$, on a les résultats de la table 2.2.

	$h = 2/104$	$h = 2/74$	$h = 2/52$	$h = 2/36$	$h = 2/26$
	$h \simeq 0.0192$	$h \simeq 0.0270$	$h \simeq 0.0385$	$h \simeq 0.0556$	$h \simeq 0.0769$
	$C_M \simeq 0.408$	$C_M \simeq 0.290$	$C_M \simeq 0.204$	$C_M \simeq 0.141$	$C_M \simeq 0.102$
SUPG	$7.17 * 10^{-5}$	$5.32 * 10^{-4}$	$1.29 * 10^{-3}$	$6.89 * 10^{-3}$	$1.59 * 10^{-2}$
STILS	$3.26 * 10^{-4}$	$6.80 * 10^{-4}$	$3.22 * 10^{-3}$	$4.47 * 10^{-3}$	$1.05 * 10^{-2}$



TAB. 2.2 – Conservativité en fonction de h

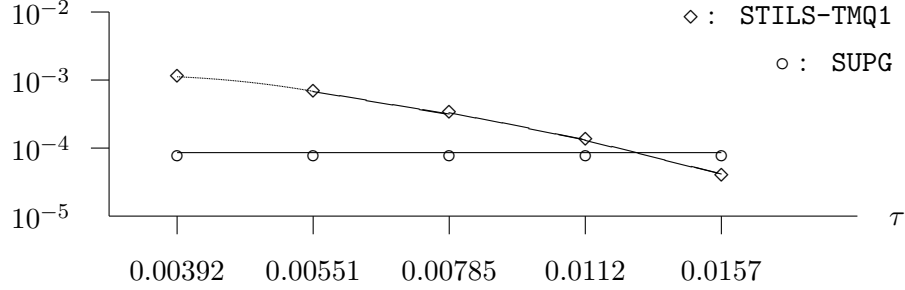
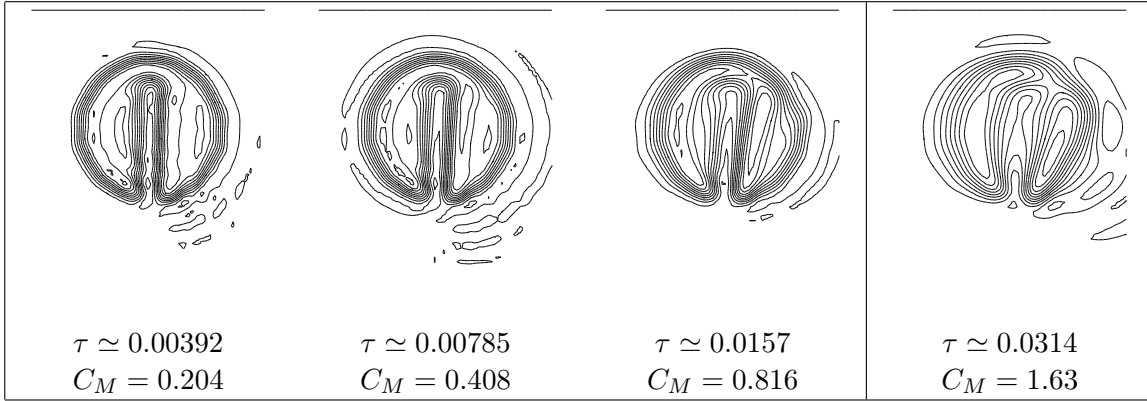
On voit un effet attendu pour les deux méthodes : la conservativité se dégrade quand h grossit, et ce de manière moins forte pour STILS. Pour quantifier cette évolution, on peut chercher l'ordre de la dépendance entre la conservativité k et h . En supposant un instant que $k(h) = \text{cte} \cdot h^d$ où h^d est l'ordre cherché, on trouve, en comparant les paires de valeurs mesurées, un degré d entre 1.7 et 3.2 pour STILS et entre 3.6 et 4.2 pour SUPG.

Si on fait évoluer τ en fixant $h = 2/104$ (maillage 104×104), on a les résultats de la table 2.3.

L'évolution de la conservativité globale k en fonction de τ paraît surprenante pour les deux méthodes. Pour SUPG, $k(\tau)$ est presque constante. On peut l'expliquer par la forte diffusion artificielle qui semble « fixer » k pour un maillage spatial donné.

Le comportement de k pour STILS est aussi surprenant : la conservativité globale s'améliore (donc décroît), même, quand τ croît, avec un degré d allant de -1.71 à -2.92 . Cela semble aberrant mais en regardant les diagrammes en courbes de niveau de la solution numérique en $t = T$, (tableau 2.4), on s'aperçoit qu'en revanche la conservativité **locale** se dégrade quand τ croît. Cette diffusion numérique peut avoir pour effet de diminuer les fluctuations de la concentration intégrale approchée au cours de la simulation. En effet, celle-ci est calculée par intégration de la solution numérique c_h sur un maillage qui reste fixe alors que la solution effectue une rotation en fonction de t_k . On remarque par ailleurs, dans le diagramme pour lequel $C_M > 1$, le retard pris par la solution numérique, particulièrement

	$\tau \simeq 0.00392$	$\tau \simeq 0.00551$	$\tau \simeq 0.00785$	$\tau \simeq 0.0112$	$\tau \simeq 0.0157$
	$C_M \simeq 0.204$	$C_M \simeq 0.286$	$C_M \simeq 0.408$	$C_M \simeq 0.583$	$C_M \simeq 0.816$
SUPG	$7.70 * 10^{-5}$	$7.43 * 10^{-5}$	$7.17 * 10^{-5}$	$7.04 * 10^{-5}$	$6.92 * 10^{-5}$
STILS	$1.07 * 10^{-3}$	$6.55 * 10^{-4}$	$3.26 * 10^{-4}$	$1.30 * 10^{-4}$	$4.29 * 10^{-5}$

TAB. 2.3 – Conservativité en fonction de τ TAB. 2.4 – Evolution de la solution numérique par STILS, au temps final $t = T$, en fonction du pas de temps τ

dans la zone $r = \sqrt{x_1^2 + x_2^2} > \frac{1}{C_M}$, dans laquelle $|C(r)| = r \cdot C_M$ est supérieur à 1 (condition CFL violée, cf le chapitre suivant).

2.2.3 Cône absorbé par 6 tourbillons

Introduit par P. Smolarkiewicz [54], cet exemple est devenu un classique connu sous le nom de « flux déformant » [55, 26]. Sa particularité est de déformer la masse de concentration initiale, disposée en un cône, par un champ de vecteurs simple à formuler qui l'absorbe dans des tourbillons de sens opposés.

Le domaine est le rectangle $]25, 75[\times]12.5, 87.5[$, discrétisé en 50×75 éléments (donc, de taille 1×1) de type Q_1 .

La condition initiale forme un cône centré en $(50, 50)$.

$$c(x, y, 0) = c_0(x, y) = \begin{cases} 1 - R/15 & \text{si } R \leq 15 \\ 0 & \text{ailleurs} \end{cases}$$

où $R = \sqrt{(x - 50)^2 + (y - 50)^2}$. La figure 8 représente les courbes de niveau de $c_0(x, y)$. Le champ de vitesse est

$$v(x, y, t) = \frac{8\pi}{25} \left[\sin\left(\frac{\pi x}{25}\right) \sin\left(\frac{\pi y}{25}\right), \cos\left(\frac{\pi x}{25}\right) \cos\left(\frac{\pi y}{25}\right) \right].$$

On a ainsi les centres de 6 tourbillons dont le sens est déterminé par la parité de $(x/25) (y/25)$. A nouveau, le champ v est à divergence nulle. La figure 7 représente v pour un maillage plus grossier, et le domaine est tourné d'un quart de tour.

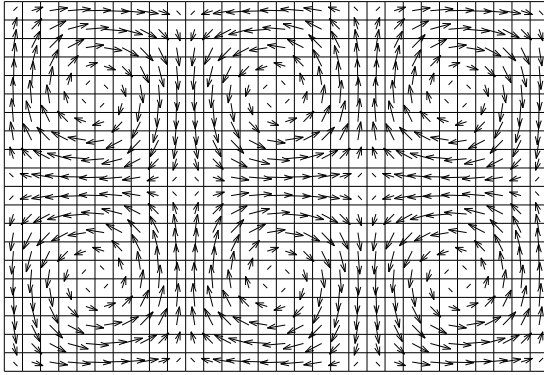


Fig. 7 Champ de vitesses v (maillage plus grossier 20×30)

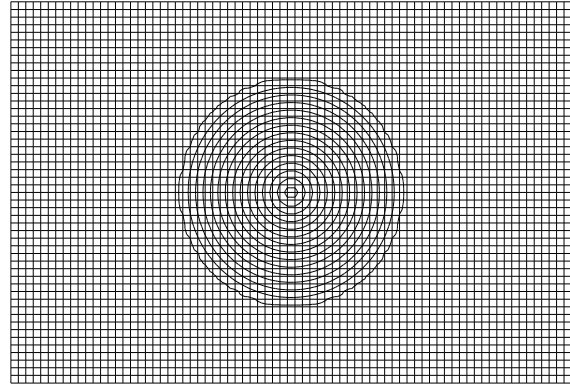


Fig. 8 Condition initiale et maillage 50×75

Le temps final est $T = 200 \cdot 1.3188 = 263.76$ (toutes ces données sont celles utilisées dans [26]).

Les calculs numériques présentent les résultats généraux suivants :

Méthode	# pas de temps	min	max	variation relative	temps CPU (format ma- trices bande)
Galerkine	40 x 20	-1.032	1.140	$2.52 \cdot 10^{-9}$	0h24
SUPG	40 x 20	-0.193	0.967	$6.80 \cdot 10^{-10}$	0h24
SUPG-DC	40 x 20	-0.159	0.967	$6.21 \cdot 10^{-10}$	0h38
STILS-TMQ1	40 x 20	-0.256	1.023	$7.40 \cdot 10^{-10}$	0h25
C-Convection	$\leq 100 \times 20$	0.000	1.000	$2.66 \cdot 10^{-1}$	0h26

TAB. 2.5 – Résultats – flux déformant

La variation relative théorique est 0 puisqu'on a les mêmes conditions que pour le flot

tournant, donc la relation (2.11) est aussi valable. La concentration intégrale vaut $\int_{\Omega} c \, dx = \pi \cdot 75 \simeq 235.6$ et est, on le voit, particulièrement bien conservée par les méthodes d'éléments finis.

Voici un aperçu de la solution de l'équation de transport au temps $T/2$ et au temps final T , pour les méthodes de Galerkin, Streamline diffusion (SUPG), Shock-capturing (SUPG-DC), STILS-TMQ1, caractéristiques (C-Convection).

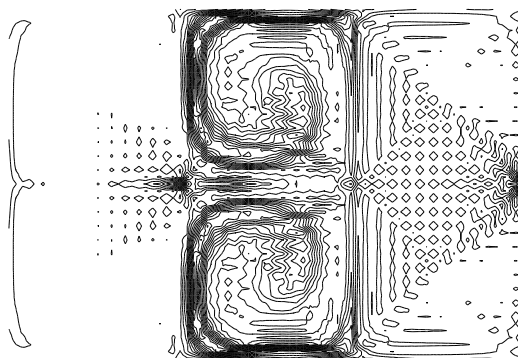


Fig. 9 Méthode de Galerkin simple
(200ème pas)

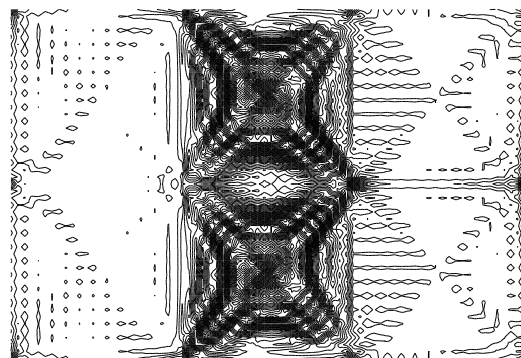


Fig. 10 Méthode de Galerkin simple
(800ème pas)

Notons que les oscillations bien connues de Galerkin pour l'équation de transport se manifestent très tôt.

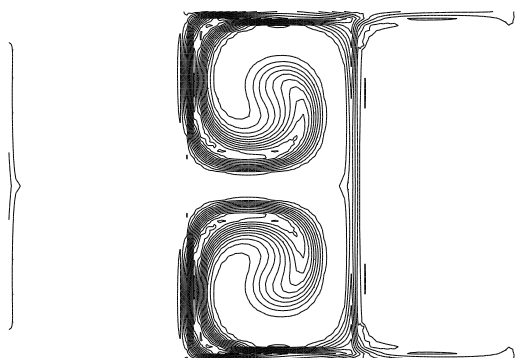


Fig. 11 Streamline diffusion (200ème
pas)

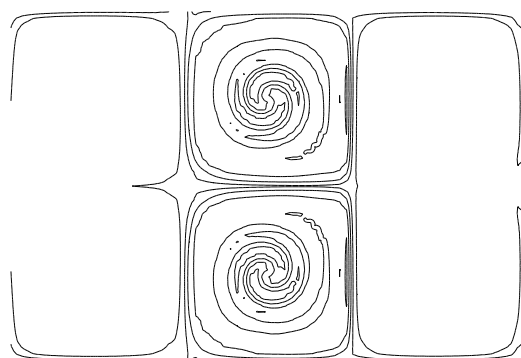


Fig. 12 Streamline diffusion (800ème
pas)

Il y a là un phénomène typique d'une diffusion artificielle : l'effet stabilisant permet une solution cohérente au début, mais la diffusion finit par l'aplatir. Idem pour le shock-capturing :

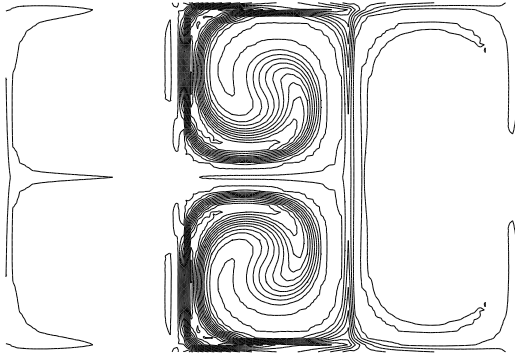


Fig. 13 Streamline diffusion, shock capturing (200ème pas)

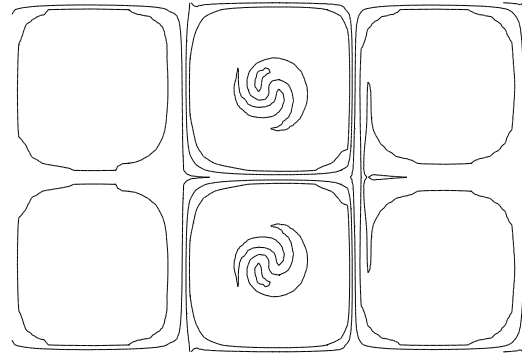


Fig. 14 Streamline diffusion, shock capturing (800ème pas)

On en vient à STILS.

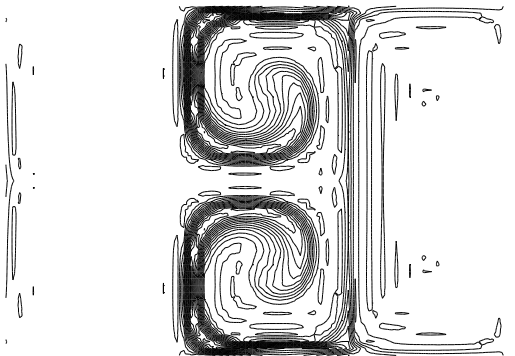


Fig. 15 STILS-TMQ1 (200ème pas)

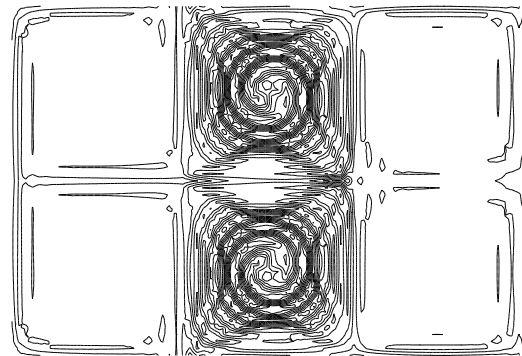


Fig. 16 STILS-TMQ1 (800ème pas)

La « tenue de route » de STILS marche-en-temps est spectaculaire. On peut être surpris par la finesse du résultat de 800 opérations matricielles compte tenu de la relative grossièreté du maillage spatial et du fait qu'il n'y a aucun recalcul de matrices au cours de l'avance en temps de style shock-capturing.

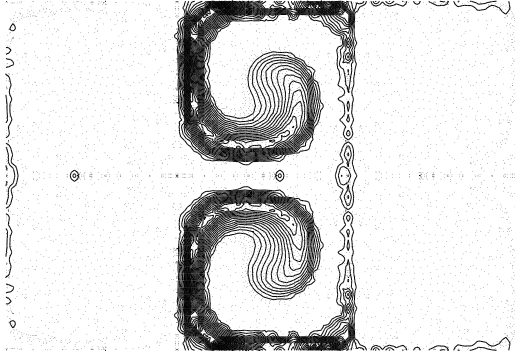


Fig. 17 Caractéristiques (200ème pas)

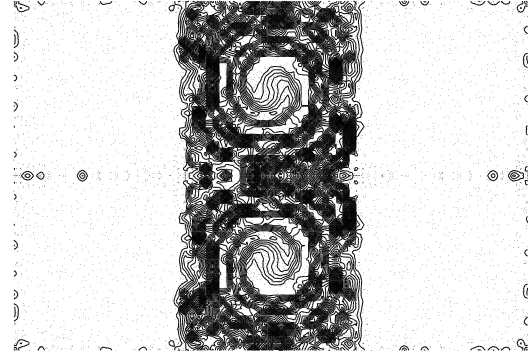


Fig. 18 Caractéristiques (800ème pas)

Commentaire général : Toutes les méthodes ont des difficultés après un certain temps : les bandes issues du cône initial deviennent de plus en plus minces à mesure qu'elles sont enroulées par les tourbillons ; ainsi leur largeur finit par passer au-dessous de la taille d'un élément. L'effet est un mélange complet de c_h . Cela n'empêche pas la concentration intégrale d'être très bien conservée par les méthodes d'éléments finis.

2.2.4 Plug-flow

Cet exemple, qui provient de [47], est une simulation de flux et de transport en milieu poreux ². Résumons le contexte rocheux (voir figure 19) : une couche de rocher imperméable est prise entre deux couches de terrain poreux et traversée par une fissure à forte perméabilité. Le champ de vitesses v dérive d'un potentiel h qui est la solution de l'équation elliptique de flux stationnaire (1) de l'introduction. Le tenseur K , constant par morceaux, est défini de la manière suivante : $K = 0.001 \cdot I$ dans les deux zones perméables, $K = 10^{-8} \cdot I$ dans la partie rocheuse et $K = 0.01$ dans la fissure. La condition aux bords pour le flux est une condition de Neumann nulle (non-flux) partout sur $\partial\Omega$, sauf des conditions de Dirichlet aux coins inférieurs ($h = 100$) et supérieurs ($h = 50$). Cela induit un flux du bas vers le haut passant par la fissure.

²Lire également [48, 2]

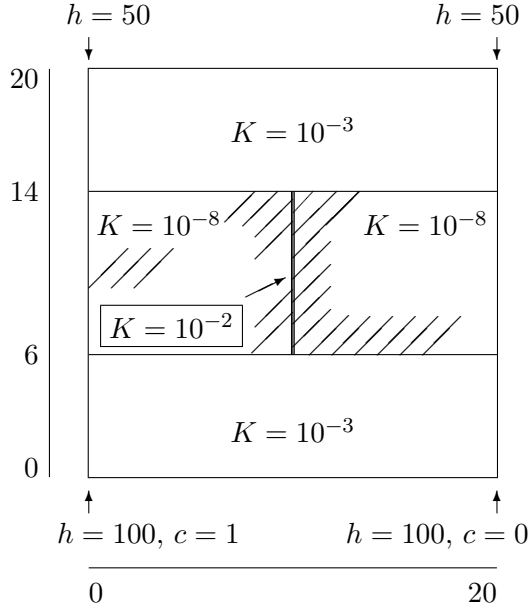


Fig. 19 Perméabilité et conditions aux bords

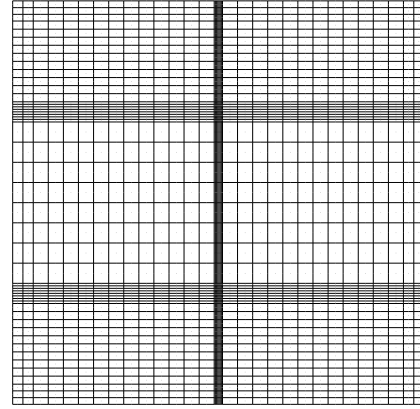
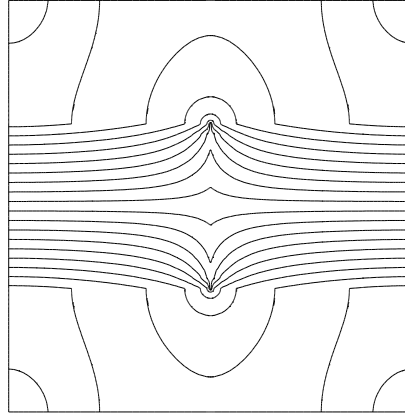


Fig. 20 Maillage : 7171 nœuds, 1750 éléments

Fig. 21 Courbes de niveau du potentiel h , solution de (1)

Pour l'équation de transport, la condition initiale est $c = 0$ partout. La condition aux limites sur la frontière influx, c'est à dire sur de « petites » zones aux coins inférieurs, est $c = 1$ à gauche et $c = 0$ à droite. On a donc une source illimitée de fluide de concentration 1 d'un côté et de concentration 0 de l'autre. La discrétisation en espace est faite au moyen d'éléments Q_2 (sur le domaine, les fonctions sont continues et, sur chaque élément, au plus quadratiques en chaque variable). Le maillage est représenté à la figure 20. Une différence importante avec le modèle dans [48] est qu'il n'y a pas de mélange entre éléments à une et deux dimensions, pas plus qu'il n'y a de conditions au bord ponctuelles. La fissure est un rectangle mince et les points d'injection sont de petits coins du bord du domaine. Ceci est pour préserver l'intégrité de nos discrétisations.

Méthode	# pas de temps	min	max	concentration finale	temps CPU (format ma- trices bande)
SUPG-DC	20 x 20	-3.870	4.599	114.3	1h21
STILS-TMQ1	20 x 20	-1.097	1.326	114.6	0h40
C-Convection	≤ 5000 x 20	0.000	1.000	104.6	5h27

TAB. 2.6 – Résultats – plug-flow

Le temps final est $T = 7200$ secondes, c'est-à-dire deux heures. La table 2.6 montre les résultats pour les méthodes de shock-capturing (SUPG-DC), STILS-TMQ1 et caractéristiques (C-Convection).

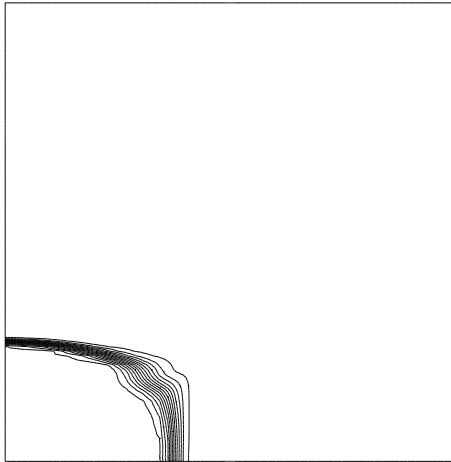


Fig. 22 Streamline diffusion et shock capturing (100ème pas)

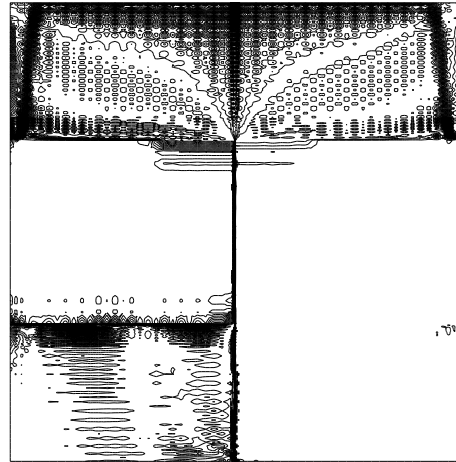


Fig. 23 Streamline diffusion et shock capturing (400ème pas)

La fissure est, on le voit, fatale à la méthode de Galerkin malgré les termes de diffusion artificielle : la solution est criblée d'oscillations. Les valeurs extrémales de la table 2.6 indiquent bien la divergence. Venons-en aux deux seules méthodes de cette comparaison qui s'en sortent :

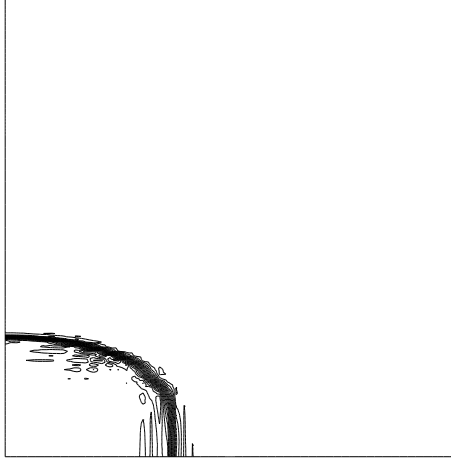


Fig. 24 STILS-TMQ1 (100ème pas)

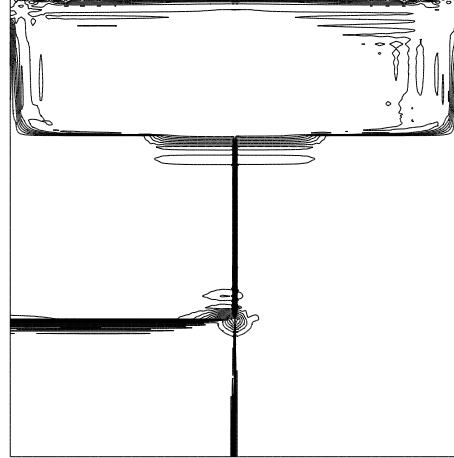


Fig. 25 STILS-TMQ1 (400ème pas)

Remarque : Pour les deux méthodes d'éléments finis, le flot à concentration 1 (à gauche) et à concentration 0 (à droite) se mélangent dans la fissure avec pour résultat une zone à concentration de 0,5 dans la partie supérieure (figures 23 et 25). Même si c'est naturel, il s'agit d'une erreur d'approximation si on considère la convection pure. En effet, si v est continu, les caractéristiques ne s'intersectent pas et donc les valeurs ne devraient jamais se mélanger.

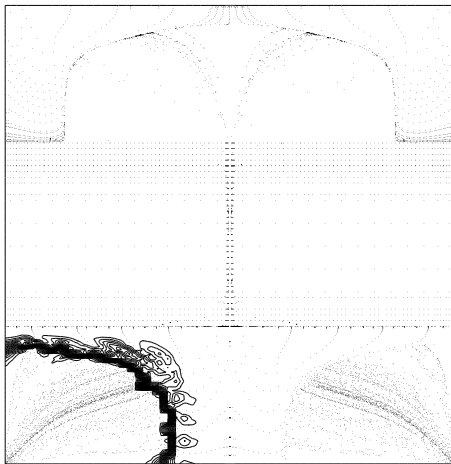


Fig. 26 Caractéristiques (100ème pas)

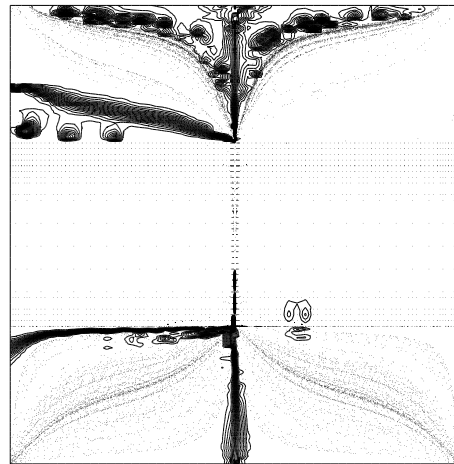


Fig. 27 Caractéristiques (400ème pas)

Commentaire général :

La solution STILS, quoique fort stable, montre des dépassements significatifs aux régions critiques du domaine (extrémités de la fissure). Néanmoins, ils restent très *localisés*. Il y a donc moyen de tronquer après calcul la solution numérique pour obtenir un résultat chimiquement sensé : c entre 0 et 1.

Comme mentionné plus haut, le champ de vitesse v est une approximation ; précisément, c'est une solution de Galerkin de l'équation stationnaire (1) utilisant le même maillage. La méthode des caractéristiques s'est révélée utile pour raffiner convenablement le maillage (pour toutes les méthodes). Un champ v mal approché au voisinage de la fissure conduit les caractéristiques, telles qu'intégrées numériquement, à traverser l'axe de symétrie vertical ou à finir agglutinées dans la roche. L'erreur sur v est donc « visualisée » par une méthode de caractéristiques, alors qu'on pourrait l'attribuer — faussement — à un problème numérique lié à l'équation de convection, lorsque résolue par éléments finis. Néanmoins, pour la résolution même de l'équation de transport, le flot de particules qui suit les caractéristiques présente, on l'a dit plus haut, l'inconvénient d'être peu dense — parce que rapide — justement là où on aurait besoin du maximum d'information en des points donnés du domaine, dans les zones à fort gradient. On le voit bien ici et la simulation du puits dans l'exemple de l'introduction est encore plus flagrante.

2.3 Conclusion à STILS d'ordre 1 en temps

Récoltons quelques observations sur STILS-TMQ1, indépendantes des exemples.

1. On a clairement une meilleure précision, dans le sens que la diffusion numérique ³ est bien plus petite que pour les solutions par méthodes d'éléments finis classiques, même lorsqu'aucune diffusion artificielle n'est ajoutée.
2. La stabilité semble meilleure. A vrai dire, les estimations de stabilité faites dans [48] par une analyse de Fourier et par des matrices d'amplification le prédisent déjà. Nous y reviendrons au prochain chapitre.
3. Les oscillations — lorsqu'il y en a — tendent à se concentrer très près des discontinuités. C'est un moindre mal correspondant au phénomène de Gibbs qui voit apparaître des dépassements au points de discontinuité d'une fonction L^2 , projetée sur une base en sin ou cos. Ici on a aussi des fonctions de base continues, mais en l'occurrence polynomiales.

Ces expériences et observations sont encourageantes, surtout si l'on pense au fait que toutes les méthodes utilisent le même maillage en espace et — à l'exception des caractéristiques avec leurs pas adaptatif — la même discrétisation en temps.

On soulignera que la discrétisation en temps pour les méthodes classiques d'éléments finis est d'ordre 2 (BDF) alors que STILS en, ici, une d'ordre 1, ce qui devrait la pénaliser.

Il est temps de voir comment on peut développer la dimension temporelle et quelle améliorations cela apporte.

³On peut définir la diffusion numérique par l'accumulation d'erreurs et l'effet d'approximations au cours des pas de temps qui font ressembler la solution numérique à celle d'une équation ayant un terme de diffusion.

Chapitre 3

Comparaison fine et stabilité

Tout cela est bien beau dans la pratique,
mais est-ce que ça marchera en théorie ?

(ingénieur anonyme)

3.1 Moindres carrés et espace-temps : apports respectifs

Dans le chapitre précédent, les comparaisons de méthodes étaient uniquement empiriques ; d'autre part on a essentiellement comparé STILS (intégration en *espace-temps* et *moindres carrés*) à des variantes de Galerkin intégrées en espace seulement, puis discrétisées totalement en temps au moyen d'une méthode scalaire pour les équations différentielles ordinaires.

En considérant des méthodes d'éléments finis qui sont toutes intégrées en espace-temps, nous pouvons mieux cerner ce qui distingue STILS d'autres méthodes d'éléments finis, à la fois empiriquement et au moyen de calculs quantitatifs sur les systèmes linéaires correspondants.

En somme, on aimerait répondre aux questions : « Qu'apportent les moindres-carrés ? », « Qu'apporte la résolution en espace-temps ? » et aussi « Qu'apporte la combinaison des deux ? »

La démarche est illustrée par le tableau suivant :

	E.F. en espace, EDO en temps	E.F. en espace-temps , marche en temps
Fonction-test w	Galerkine problème faible : (2.1)	Galerkine en espace-temps problème faible approché : (3.1), $\delta = 0$
w et $(v \nabla w)$	Streamline diffusion problème faible : (2.4)	Streamline diff. espace-temps problème faible approché : (3.1)
$(v \nabla w)$ seul. Moindres-carrés	« SILS » problème faible : (3.4)	STILS problème faible : (1.7)

TAB. 3.1 – Démarche de comparaison entre espace-temps et moindres-carrés

On voit au passage qu'il faut pour compléter le tableau avoir une méthode « SILS » de moindres carrés en espace seulement. Par l'analyse de stabilité ou par l'expérimentation (par exemple la comparaison de la fig. 3.2) on s'apercevra vite qu'elle ne mérite pas le détour.

Avec des fonctions de base affines en temps, la méthode Streamline diffusion en espace-temps, « marche-en-temps », correspond au problème faible approché suivant — en employant les mêmes notations que celles pour construire **STILS-TMQ1** — problème (1.11) du §1.2.3, p. 25 :

Problème : chercher $c_h \in H_{0,h} = V_h \otimes E_2 \cap H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-)$ tel que

$$\int_{\Omega} \int_{t_k}^{t_{k+1}} \left(\tilde{\nabla} c_h | \tilde{v} \right) \left(w_h + \delta \left(\tilde{\nabla} w_h | \tilde{v} \right) \right) dt dx = \int_{\Omega} \int_{t_k}^{t_{k+1}} f \left(w_h + \delta \left(\tilde{\nabla} w_h | \tilde{v} \right) \right) dt dx. \quad (3.1)$$

pour tout $w_h \in H_h$. Le facteur $\delta \geq 0$ est arbitraire.

Dans le cas de source nulle $f = 0$ on aboutit au système linéaire

$$\begin{aligned} & \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{\delta \tau}{3} (\nabla \psi_j | v) (\nabla \psi_i | v) + \left(\frac{\tau}{3} + \frac{\delta}{2} \right) (\nabla \psi_j | v) \psi_i + \frac{\delta}{2} \psi_j (\nabla \psi_i | v) + \left(\frac{1}{2} + \frac{\delta}{\tau} \right) \psi_j \psi_i \right] dx \\ &= \sum_{j=1}^N c_j^k \int_{\Omega} \left[\frac{-\delta \tau}{6} (\nabla \psi_j | v) (\nabla \psi_i | v) - \left(\frac{\tau}{6} + \frac{\delta}{2} \right) (\nabla \psi_j | v) \psi_i + \frac{\delta}{2} \psi_j (\nabla \psi_i | v) + \left(\frac{1}{2} + \frac{\delta}{\tau} \right) \psi_j \psi_i \right] dx \end{aligned} \quad (3.2)$$

pour tous i dans $\{1 \dots N\}$.

Notons que si $\delta = 0$ on a une méthode de Galerkin très simple en espace-temps

$$\sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{\tau}{3} (\nabla \psi_j | v) \psi_i + \frac{1}{2} \psi_j \psi_i \right] dx = \sum_{j=1}^N c_j^k \int_{\Omega} \left[\frac{-\tau}{6} (\nabla \psi_j | v) \psi_i + \frac{1}{2} \psi_j \psi_i \right] dx \quad (3.3)$$

que nous reverrons bientôt. A l'inverse, si $\delta \rightarrow \infty$, on tombe naturellement sur **STILS-TMQ1** (1.14). On a donc complété la colonne « espace-temps ».

Quant à la méthode « SILS », il suffit d'imiter le raisonnement de **STILS** en espace seulement, en passant le terme dérivé en temps au second membre : $(\nabla c | v) = f - \frac{\partial c}{\partial t}$. On obtient le problème faible

$$\int_{\Omega} \left[(\nabla c | v) + \frac{\partial c}{\partial t} - f \right] (\nabla w | v) dx = 0 \quad (3.4)$$

correspondant à (1.7) p. 22 (mais en espace uniquement !), puis le schéma semi-discretisé

$$Rc_h + Mc'_h = b \quad (3.5)$$

avec

$$\begin{aligned} r_{ij} &= ((\nabla \psi_i | v), (\nabla \psi_j | v)) \\ m_{ij} &= ((\nabla \psi_i | v), \psi_j). \end{aligned} \quad (3.6)$$

On reconnaît un Streamline Diffusion où la fonction-test non dérivée disparaît (autrement dit, où $\delta \rightarrow \infty$). On voit déjà surgir un sérieux problème : si $v = 0$ sur les éléments entourant un nœud i , les i èmes lignes de M et de R sont nulles ; lorsque $v = 0$ sur Ω au temps t , l'évolution de c_h est complètement indéterminée alors qu'en fait c répond à l'équation ordinaire $c'(t) = f$. Ceci n'arrive bien sûr pas pour **STILS** puisque $\tilde{v}$ ne s'annule jamais et n'est jamais

proche de 0. Sa norme (l_1, l_2 ou l_∞) vaut même toujours au moins 1 ! Ce défaut remonte d'ailleurs certainement au problème faible *avant* la discrétisation. En effet, si on définit la la forme bilinéaire

$$\mathcal{B}_{\text{esp}}(u, w) := \int_{\Omega} (\nabla u | v) [\epsilon w + \delta (\nabla w | v)] \, dx,$$

on peut exprimer tous les problèmes semi-discrétisés de ce chapitre par l'équation faible suivante

$$\mathcal{B}_{\text{esp}}(u, w) = \int_{\Omega} \left[f - \frac{\partial u}{\partial t} \right] \cdot [\epsilon w + \delta (\nabla w | v)] \, dx. \quad (3.7)$$

On a le schéma de Galerkin (2.1) si $\epsilon = 1$ et $\delta = 0$, Streamline diffusion (2.4) si $\delta > 0$, et enfin « SILS » (3.4) si $\epsilon = 0$ et $\delta = 1$. Indépendamment du choix de l'espace de Sobolev s'appliquant à (3.7), on peut essayer de mettre en défaut la coercitivité de $\mathcal{B}_{\text{esp}}$. Par exemple, si $\mathcal{B}_{\text{esp}}(u, u) = 0$ on n'a peut-être pas toujours $u = 0$. Pour SILS, $\mathcal{B}_{\text{esp}}$ n'est pas coercitive si v s'annule localement.

Pour la comparaison visuelle, les courbes de niveaux pour le problème du cylindre fendu avec les six méthodes (pour le cas de la discrétisation totale, la méthode BDF d'ordre 2 est utilisée pour l'EDO) sont réunies dans le tableau 3.2. On voit bien les difficultés de « SILS » au centre ($v \rightarrow 0$).

On peut en tirer la conjecture suivante : dans l'approche de discrétisation totale sur un schéma semi-discrétisé en espace, il y a une optimum pour un certain δ réel ; dans l'approche espace-temps avec marche en temps, le mieux est de faire $\delta \rightarrow \infty$, c'est-à-dire de faire disparaître la fonction test non dérivée de l'équation faible, c'est-à-dire encore de résoudre au sens des moindres carrés.

L'un dans l'autre, la combinaison STILS semble meilleure que les autres approches.

3.2 Analyse de stabilité des méthodes TMQ1

Pour la simplicité des calculs, nous nous ramènerons au cas de la dimension spatiale 1. L'analyse contribue néanmoins à comprendre le comportement dans les dimensions supérieures. Dans le cas d'un champ v constant par morceaux, par exemple, les caractéristiques répondent à des EDOs indépendantes d'une dimension spatiale à l'autre : $x_i(t) = v_i * t + x_i^0$. Le transport peut donc s'étudier, dans ce cas, sur chaque composante spatiale.

En dimension 1, on a la base d'éléments finis la plus simple possible : Q_1 en espace et Q_1 en temps. Le domaine Ω est l'intervalle $]0; 1[$ discrétisé en N points équidistants de coordonnées $x_1 = \frac{1}{N+1}, \dots, x_N = \frac{N}{N+1}$, la vitesse d'advection est un scalaire v que nous supposons constant, les intégrales des fonctions de base sont, en posant $h = \frac{1}{N+1}$, telles que calculées dans le tableau 3.3.

	$j = i - 1$	$j = i$	$j = i + 1$
$\int (\nabla \psi_j v) (\nabla \psi_i v) \, dx = \int \psi'_j \psi'_i v^2 \, dx$	$\frac{-v^2}{h}$	$\frac{2v^2}{h}$	$\frac{-v^2}{h}$
$\int (\nabla \psi_j v) \psi_i \, dx = \int \psi'_j \psi_i v \, dx$	$\frac{-v}{2}$	0	$\frac{v}{2}$
$\int \psi_j (\nabla \psi_i v) \, dx = \int \psi_j \psi'_i v \, dx$	$\frac{v}{2}$	0	$\frac{-v}{2}$
$\int \psi_j \psi_i \, dx$	$\frac{h}{6}$	$\frac{2h}{3}$	$\frac{h}{6}$

TAB. 3.3 – Intégrales des fonctions de base Q_1 sur maillage régulier (h)

Pour la discrétisation en temps des schémas semi-discrétisés, nous choisissons le schéma « théta » puisqu'il a le même degré en temps que les schémas en espace-temps considérés. Rappelons qu'il consiste à approcher un système de type (3.5), qui est une équation différentielle ordinaire à inconnue vectorielle c_h , par la relation

$$\left(\frac{1}{\tau} M + \vartheta R \right) c^{k+1} = \left(\frac{1}{\tau} M + (\vartheta - 1) R \right) c^k \quad (3.8)$$

où le paramètre réel ϑ fait le réglage entre explicite ($\vartheta = 0$) et implicite ($\vartheta = 1$). Les matrices R et M sont définies pour $\delta = 0$, $\delta > 0$ et $\delta \rightarrow \infty$ par (2.3), (2.7) et (3.6) respectivement.

On pose

$$C = \frac{\tau v}{h},$$

le nombre de Courant, et, pour les méthodes qui diffusent sur les lignes de courant,

$$D = \frac{\delta v}{h} \quad \text{et} \quad E = \frac{\delta}{\tau} > 0.$$

Les tables 3.5 et 3.4 regroupent les schémas aux différences obtenus dans les six contextes. On observera que pour $\vartheta = 2/3$, les variantes schéma « théta » et marche en temps coïncident pour Galerkin simple. C'est aussi vrai pour le cas général en espace : il suffit de comparer le système (3.3) au système (3.8) pour les matrices (2.3). Ce n'est plus vrai pour $\delta > 0$. On peut dire, déjà expérimentalement, que le chemin vers les moindres-carrés commence au même point pour le semi-discrétisé que pour le marche-en-temps mais aboutit à des schémas de qualité diamétralement opposée. L'analyse de stabilité des schémas 1D va nous en donner une confirmation.

Galerkine $\delta = 0$	$(1 - 3\vartheta C) c_{i-1}^{k+1} + 4 c_i^{k+1} + (1 + 3\vartheta C) c_{i+1}^{k+1} =$ $(1 - 3(\vartheta - 1)C) c_{i-1}^k + 4 c_i^k + (1 + 3(\vartheta - 1)C) c_{i+1}^k$
Streamline diffusion $\delta > 0$	$(1 - 3\vartheta C + 3D(1 - 2\vartheta C)) c_{i-1}^{k+1} + (4 + 3D(4\vartheta C)) c_i^{k+1} +$ $(1 + 3\vartheta C + 3D(-1 - 2\vartheta C)) c_{i+1}^{k+1} =$ $(1 - 3(\vartheta - 1)C + 3D(1 - 2(\vartheta - 1)C)) c_{i-1}^k + (4 + 3D(4(\vartheta - 1)C)) c_i^k +$ $(1 + 3(\vartheta - 1)C + 3D(-1 - 2(\vartheta - 1)C)) c_{i+1}^k$
Moindres- carrés SILS $ D \rightarrow \infty$	$(1 - 2\vartheta C) c_{i-1}^{k+1} + 4\vartheta C c_i^{k+1} + (-1 - 2\vartheta C) c_{i+1}^{k+1} =$ $(1 - 2(\vartheta - 1)C) c_{i-1}^k + 4(\vartheta - 1)C c_i^k + (-1 - 2(\vartheta - 1)C) c_{i+1}^k$

TAB. 3.4 – Schémas aux différences 1D, schéma théta en temps

Galerkine $\delta = 0$	$(1 - 2C) c_{i-1}^{k+1} + 4 c_i^{k+1} + (1 + 2C) c_{i+1}^{k+1} =$ $(1 + C) c_{i-1}^k + 4 c_i^k + (1 - C) c_{i+1}^k$
Streamline diffusion $\delta > 0$	$(1 - 2C + 2E(1 - 2C^2)) c_{i-1}^{k+1} + (4 + 2E(4 + 4C^2)) c_i^{k+1} +$ $(1 + 2C + 2E(1 - 2C^2)) c_{i+1}^{k+1} =$ $(1 + C + 2E(1 + 3C + C^2)) c_{i-1}^k + (4 + 2E(4 - 2C^2)) c_i^k +$ $(1 + C + 2E(1 - 3C + C^2)) c_{i+1}^k$
Moindres- carrés STILS $E \rightarrow \infty$	$(1 - 2C^2) c_{i-1}^{k+1} + (4 + 4C^2) c_i^{k+1} + (1 - 2C^2) c_{i+1}^{k+1} =$ $(1 + 3C + C^2) c_{i-1}^k + (4 - 2C^2) c_i^k + (1 - 3C + C^2) c_{i+1}^k$

TAB. 3.5 – Schémas aux différences 1D, TMQ1

Afin d'analyser ces schémas, rappelons brièvement les principes de l'analyse de Fourier – von Neumann. A l'aide de la transformée de Fourier discrète de la solution numérique c_h^k ,

$$\hat{c}^k(\omega) = \sum_{m \in \mathbb{Z}} c_m^k e^{-im\omega}, \quad (3.9)$$

on condense *toute* l'information du schéma aux différences dans une fonction d'amplification complexe $\gamma(\omega)$ ¹ :

$$\hat{c}^{k+1}(\omega) = \gamma(\omega) \hat{c}^k(\omega) \quad (3.10)$$

Les fonctions d'amplification des six schémas aux différences (tables 3.4 et 3.5) traités dans ce chapitre sont représentées dans les tables 3.6 et 3.7.

¹aussi appelée *symbole* [16]

Galerkine $\delta = 0$	$\gamma(\omega) = \frac{2+\cos \omega+3i(\vartheta-1)C \sin \omega}{2+\cos \omega+3i\vartheta \quad C \sin \omega}$
Streamline diffusion $\delta > 0$	$\gamma(\omega) = \frac{2+\cos \omega+6CD(\vartheta-1)(1-\cos \omega)+3i((\vartheta-1)C-D) \sin \omega}{2+\cos \omega+6CD\vartheta \quad (1-\cos \omega)+3i(\vartheta \quad C-D) \sin \omega}$
Moindres- carrés SILS $ D \rightarrow \infty$	$\gamma(\omega) = \frac{2(\vartheta-1)C(1-\cos \omega)-i \sin \omega}{2\vartheta \quad C(1-\cos \omega)-i \sin \omega}$

TAB. 3.6 – Fonctions d'amplification des schémas, schéma théta en temps

Galerkine $\delta = 0$	$\gamma(\omega) = \frac{2+\cos \omega-iC \sin \omega}{2+\cos \omega+2iC \sin \omega}$
Streamline diffusion $\delta > 0$	$\gamma(\omega) = \frac{2EC^2(\cos \omega-1)+(1+2E)(\cos \omega+2)-i(C+6EC) \sin \omega}{-4EC^2(\cos \omega-1)+(1+2E)(\cos \omega+2)+i2C \quad \sin \omega}$
Moindres- carrés STILS $E \rightarrow \infty$	$\gamma(\omega) = \frac{2-C^2+(1+C^2) \cos \omega-3iC \sin \omega}{2+2C^2+(1-2C^2) \cos \omega}$

TAB. 3.7 – Fonctions d'amplification des schémas TMQ1

Dans ces schémas, c_i^{k+1} ne dépend que de c_{i-1}^k , c_i^k et c_{i+1}^k . Il est donc impossible que l'information se propage sur le maillage en espace-temps plus vite qu'à la vitesse h/τ . Si $|C| > 1$, c'est-à-dire si $|v| > h/\tau$, une solution numérique ne peut être que fautive car retardée. Cela fournit déjà la condition **CFL** nécessaire au bon fonctionnement de ces schémas : $|C| \leq 1$.

La condition suffisante de **stabilité** de von Neumann est $|\gamma(\omega)| \leq 1$ voire $|\gamma(\omega)| \leq 1 + \text{cte } \tau$. Dans tous les cas la limite de stabilité $|\gamma(\omega)| = 1$ est atteinte pour $C = 0$ (c'est-à-dire $v = 0$) ou $\sin \omega = 0$. Mis à part ces situations particulières, on vérifie aisément que les schémas Galerkin et SILS discrétisés en temps par un schéma théta sont stables si et seulement si $\vartheta \geq 1/2$. Ainsi donc pour Galerkin-TMQ1 qui coïncide avec le cas $\vartheta = 2/3$. Pour Streamline-théta, on a un peu mieux avec la condition : $|\gamma(\omega)| \leq 1$ si et seulement si

$$0 \leq |D| + (\vartheta - \frac{1}{2})3 \left[4|D| \frac{1 - \cos \omega}{1 + \cos \omega} + |C|(1 - \cos \omega) \right].$$

En revanche Streamline-TMQ1 devient conditionnellement stable (selon E et C , donc selon le choix de δ , τ , h et selon les circonstances : valeur de v) avec la condition $|\gamma(\omega)| \leq 1$ si et seulement si

$$0 \leq E^2 C^4 (1 - \cos \omega) + EC^2 (1 + 2E)(2 + \cos \omega) + \left(\frac{C^2}{4} - EC - 3E^2 C^2 \right) (1 + \cos \omega).$$

On retrouve une stabilité inconditionnelle, c'est-à-dire indépendante de C , pour STILS-TMQ1

(ce qui correspond aussi à Streamline-TMQ1 avec $E \rightarrow \infty$) puisqu'on a en toute généralité

$$|\gamma(\omega)| \leq 1 \quad \Leftrightarrow \quad 0 \leq (1 - \cos \omega)C^2(C^2 + 1) \quad (3.11)$$

3.2.1 Amplitude

Comparons les graphes des symboles de ces schémas. La figure ci-dessous 3.1 (obtenue par MathCAD) représente les courbes dans le plan complexe des symboles $\bar{\gamma}$ pour Galerkin-TMQ1, Streamline-TMQ1 avec $D = \frac{\delta v}{h} = 0.1$, STILS-TMQ1 avec ω variant de 0 à π — c'est un intervalle pour toutes les fréquences non-redondantes à considérer. Les courbes sont tracées pour quelques valeurs du nombre de Courant C : 0.2, 0.5, 0.9, 2.

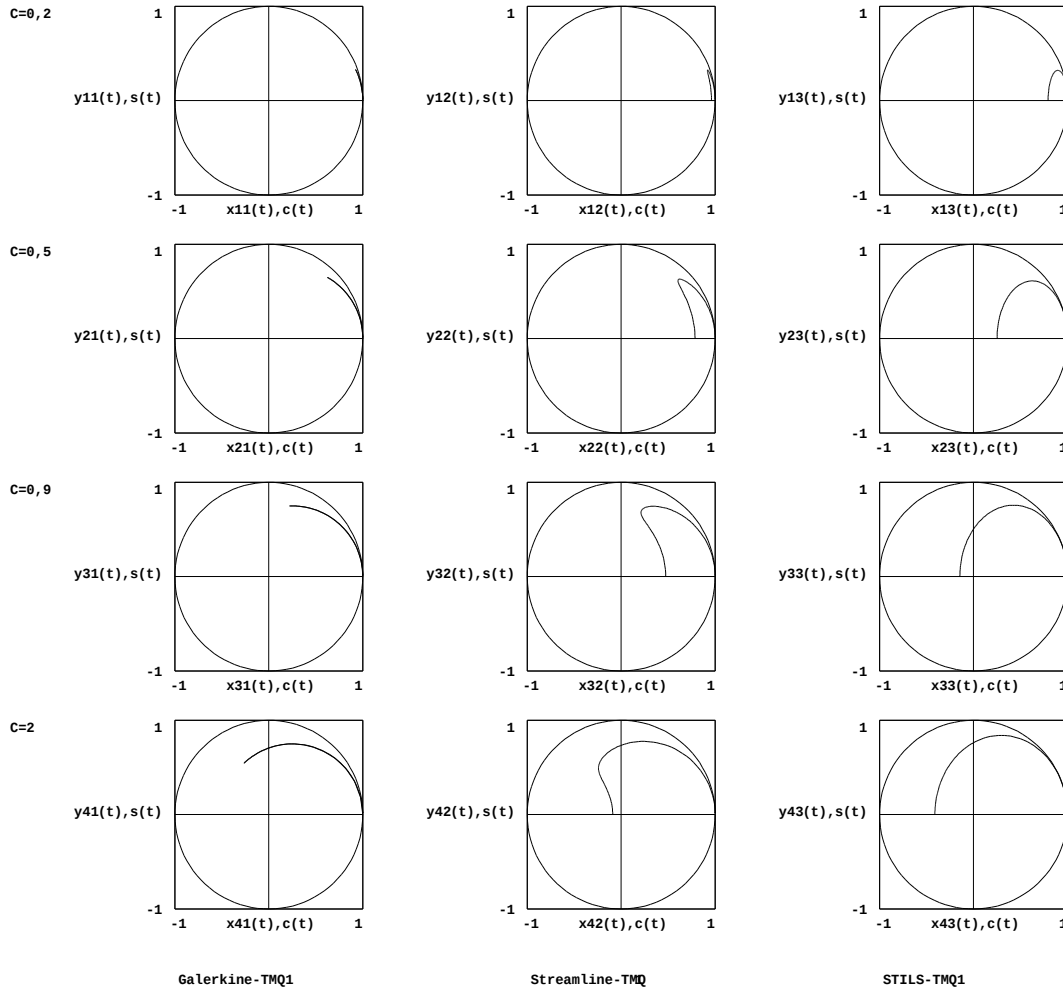


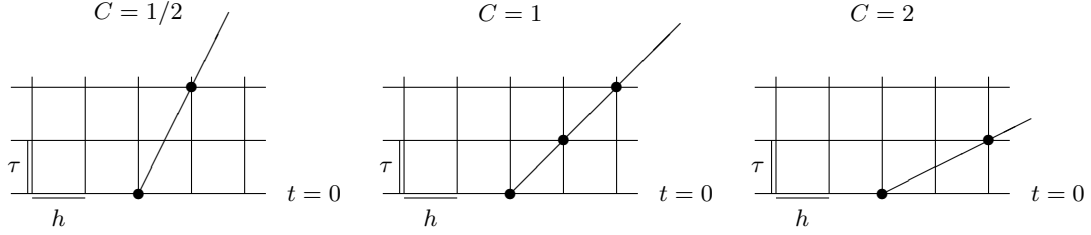
FIG. 3.1 – Amplification $\bar{\gamma}$ pour Galerkin-TMQ1, Streamline-TMQ1, STILS-TMQ1 ; $C = 0.2$, $C = 0.5$, $C = 0.9$, $C = 2$

On observe que la phase est incomplète pour $\delta = 0$ (Galerkin). A vrai dire, la courbe revient en arrière. On le verra avec les diagrammes 3.2 et 3.3, mais on le devine par le comportement du cas $\delta > 0$. Ce n'est qu'avec STILS-TMQ1 qu'on a pour C proche de 1 une courbe ressemblant à un demi-cercle, certes décentré. Bien sûr, il y a déphasage pour les autres valeurs de C — par exemple, on devrait idéalement avoir un tour complet pour $C = 2$

— mais ceci est un autre problème.

3.2.2 Déphasage

La fonction cercle $\xi(t) = e^{-iCt}$ est, rappelons-le, le symbole du schéma « modèle ». On s'en convainc en prenant d'abord C entier ou inverse d'entier, puis rationnel. $\xi(t)$ est alors le symbole d'un schéma exact, du moins aux points d'espace-temps où une caractéristique de vitesse v , issue d'un point du maillage pour $t = 0$, rencontre à nouveau le maillage.



On voit mieux ce qui se passe au niveau du déphasage par rapport à $\xi(t) = e^{-iCt}$ en comparant séparément les parties réelles et imaginaires de γ — figures 3.2 et 3.3. On garde bien sûr $D = \frac{\delta v}{h} = 0.1$. Sur la partie réelle, comparons les graphes $(t, \text{Re}(\gamma(t)))$ aux graphes $(t, \cos(t))$

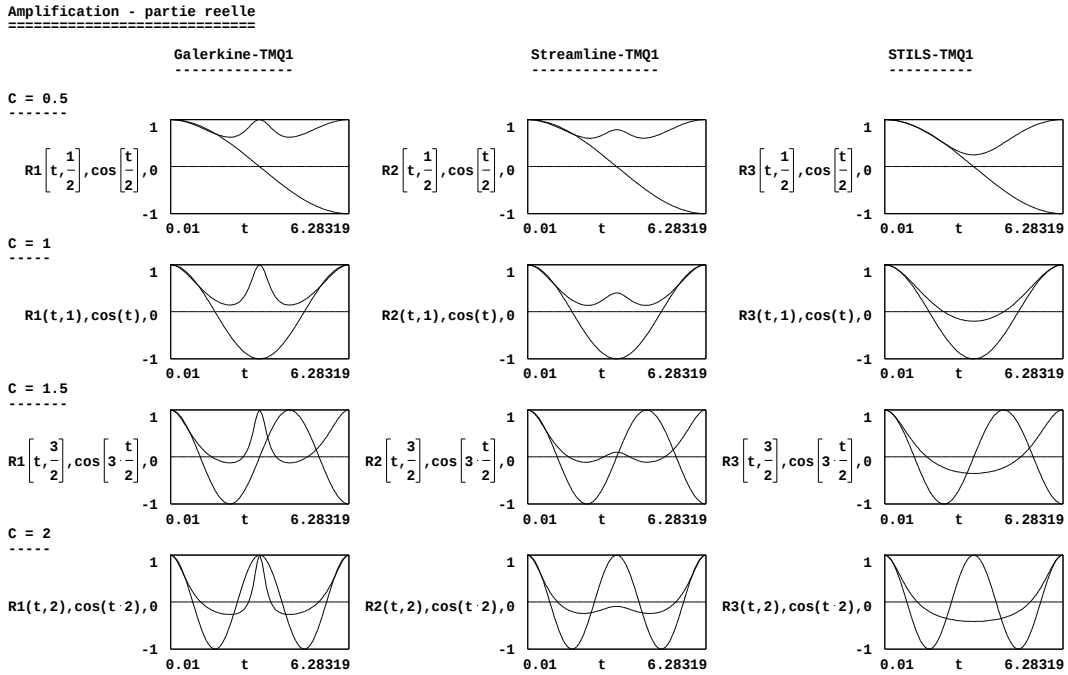
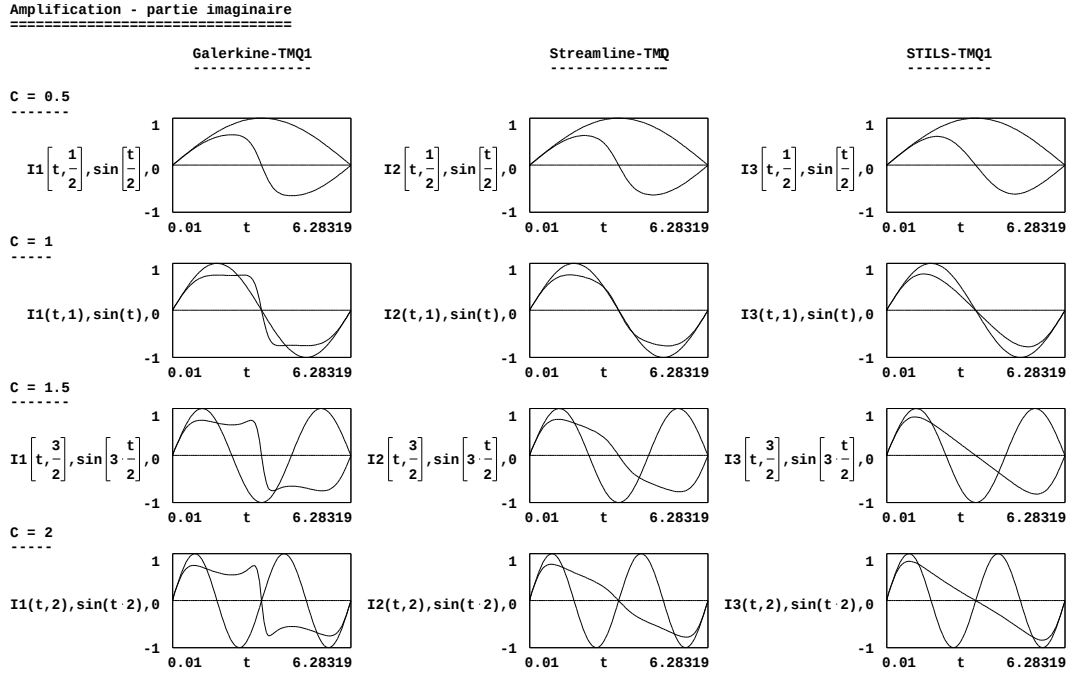


FIG. 3.2 – Déphasage sur $\mathbb{R}$ de Galerkine-TMQ1, Streamline-TMQ1, STILS-TMQ1

La partie réelle pour STILS est la plus satisfaisante pour $C = 1$, car il n'y a plus le retour en arrière repéré plus haut qui correspond ici à une remontée avec un sommet en $t = \pi$.

Quant aux graphes de la partie imaginaire $(t, -\text{Im}(\gamma(t)))$ comparés aux graphes $(t, \sin(t))$, on a

FIG. 3.3 – Déphasage sur $i\mathbb{R}$ de Galerkine-TMQ1, Streamline-TMQ1, STILS-TMQ1

Pour la partie imaginaire uniquement, on pourrait dire qu'un $\delta < \infty$ est plus satisfaisant ; mais comme il faut considérer γ dans $\mathbb{C}$, il en résulte que STILS a le moins mauvais déphasage de cette classe de méthodes.

On mesure souvent le déphasage par une fonction $\Delta(\omega) = \frac{\arg(\gamma(\omega)) - (-C\omega)}{|C\omega|}$ mais la non-injectivité de la fonction $\arg$ fait perdre l'essentiel de l'information sur l'avance ou le retard dès que γ a accompli un tour sur le plan complexe. C'est notamment le cas pour les degrés supérieurs en temps étudiés au chapitre suivant où des diagrammes sont présentés pour les mêmes valeurs de C , ce qui permet de voir une certaine évolution.

3.2.3 Lien entre analyse de Fourier et analyse matricielle

Faisons un bref rappel de deux aspects bien connus (ch. 2 de [60], ou [56]) mais rarement mis ensemble.

Quitte à prendre des conditions aux limites périodiques, la matrice d'amplification devient diagonalisable, de valeurs propres $\gamma(\frac{2\pi l}{N})$, $l \in \{0 \dots N-1\}$. Pour détailler cela, on se souvient des matrices A et B du système linéaire générique (1.15), qui est $Ac^{k+1} = Bc^k$. On tire alors parti du fait que pour la dimension 1 en espace, les matrices A et B sont circulantes, c'est-à-dire de la forme

$$\begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \dots & \alpha_{N-2} & \alpha_{N-1} \\ \alpha_{N-1} & \alpha_0 & \alpha_1 & \dots & \alpha_{N-3} & \alpha_{N-2} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_{N-1} & \alpha_0 \end{pmatrix}$$

et de valeurs propres $\lambda_l = \sum_{k=0}^{N-1} \alpha_k e^{ilk2\pi/N}$ et de vecteurs propres correspondants

$$v^l = \left(1, e^{il2\pi/N}, \dots, e^{il(N-1)2\pi/N}\right)^T$$

qui sont indépendants des valeurs $\{\alpha_0, \dots, \alpha_{N-1}\}$. Par le déterminant de Vandermonde, on voit que les $\{v^l\}$ forment une base. Toujours avec les matrices A et B de (1.15), dont les lignes, pour notre étude, correspondent aux schémas des tables 3.4 et 3.5, on peut poser $G = A^{-1}B$, la matrice d'amplification. On vérifie aisément que

$$Gv^l = \frac{\lambda_l(B)}{\lambda_l(A)} v^l = \gamma\left(\frac{2\pi l}{N}\right) v^l.$$

On a donc d'une part

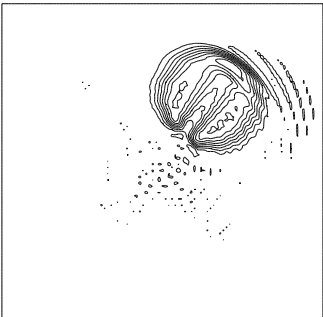
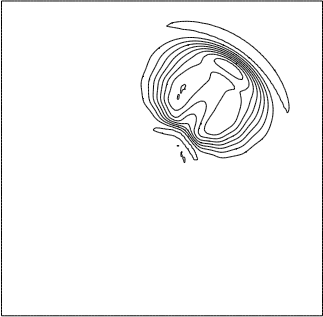
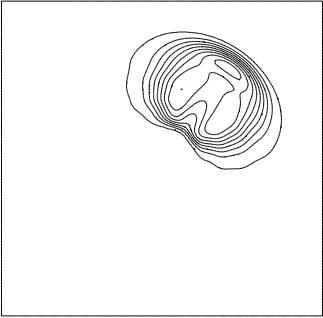
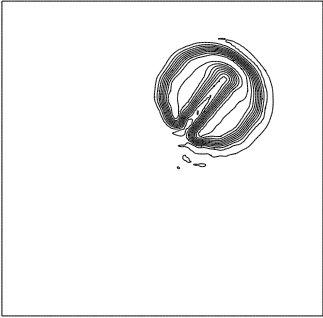
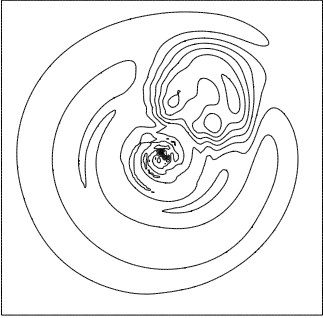
$$\varrho(G) \leq \|G\|_2$$

(avec égalité si G est symétrique) ; d'autre part

$$\varrho(G) = \max_l \left| \gamma\left(\frac{2\pi l}{N}\right) \right| \leq \max_{\omega \in [0; \pi]} |\gamma(\omega)|.$$

3.3 Conclusion

En explorant plus systématiquement l'approche STILS dans ses différences avec des alternatives classiques — sans moindre-carrés ou seulement semi-discrétisées, puis discrétisées en temps — on voit se confirmer les impressions empiriques du chapitre précédent : dans le contexte des éléments finis pour résoudre une équation de transport purement convective, STILS est certainement la meilleure approche qui conserve le temps de calcul.

	E.F. en espace, (semi-discrétisé) EDO en temps, sol. BDF 2 (Gear)	E.F. en espace-temps , marche en temps
Galerkine, $\delta = 0$	 $c \in [-0.23; 1.21]$	 $c \in [-0.27; 1.33]$
Stream. diff. $\delta = \frac{\text{diamax}}{\sqrt{2}} \cdot 0.6$	 $c \in [-0.2; 1.09]$	 $c \in [-0.25; 1.24]$
Stream. diff. $\delta = \frac{\text{diamax}}{\sqrt{2}}$	 $c \in [-0.18; 1.09]$	 $c \in [-0.25; 1.24]$
Moindres-carrés $\delta \rightarrow \infty$	 $c \in [-0.56; 1.12]$	 STILS-TMQ1 $c \in [-0.26; 1.24]$

TAB. 3.2 – Moindres-carrés, espace-temps sur le problème du cylindre

Chapitre 4

STILS pour les degrés supérieurs en temps

Après avoir construit la forme la plus simple de STILS en marche-en-temps pour une base quelconque de fonctions en espace et, en temps, une base d'éléments finis de Lagrange affine par morceaux, passons aux degrés polynomiaux 2 et 3 en temps. Outre le meilleur degré d'approximation, l'avantage recherché est de bénéficier de meilleures pentes pour transporter des discontinuités, comme on le voit ci-dessous (fig. 4.1).

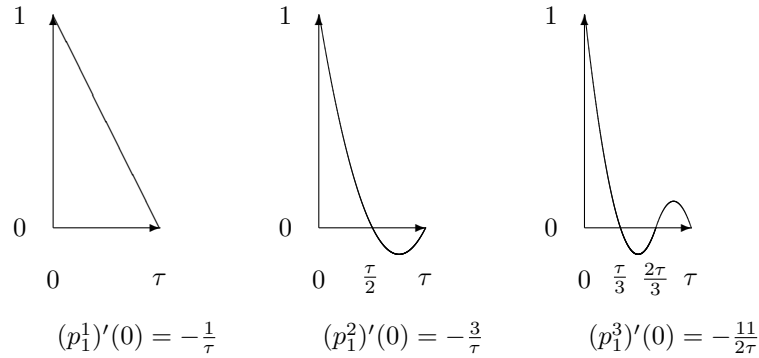


FIG. 4.1 – Pentes des fonctions de base polynomiales au point 0 pour les degrés 1,2,3

4.1 Degré 2 en temps : formulation STILS-TMQ2

L'intervalle temporel de travail est à nouveau $[t_k, t_{k+1}]$, de largeur $\tau = t_{k+1} - t_k$. Plaçons un nœud temporel intermédiaire en $t_k + \frac{\tau}{2}$ noté $t_{k+\frac{1}{2}}$. Pour simplifier les notations, reportons-nous sur $[0, \tau]$. On a les fonctions de base quadratiques

$$\begin{aligned}
 a_k(t) &= \frac{(t - \frac{\tau}{2})(t - \tau)}{(0 - \frac{\tau}{2})(0 - \tau)} \\
 a_{k+\frac{1}{2}}(t) &= \frac{(t - 0)(t - \tau)}{(\frac{\tau}{2} - 0)(\frac{\tau}{2} - \tau)} \\
 a_{k+1}(t) &= \frac{(t - 0)(t - \frac{\tau}{2})}{(\tau - \frac{\tau}{2})(\tau - \frac{\tau}{2})}.
 \end{aligned} \tag{4.1}$$

Ces trois fonctions en temps engendrent un sous-espace E_2 de $\mathcal{C}[t_k, t_{k+1}]$. L'espace H_h des solutions approchées est alors engendré par tous les produits des fonctions $a_k(t), a_{k+\frac{1}{2}}(t)$ ou $a_{k+1}(t)$ par les fonctions spatiales $\psi_j(x)$ liées au maillage d'éléments finis. Notons encore V_h l'espace de fonctions engendré par la base $\{\psi_1 \dots \psi_N\}$.

On résoud donc à nouveau le problème approché de STILS (1.11) pour l'espace

$$H_{0,h} = V_h \otimes E_2 \cap H_0(v, \tilde{\Omega}, \tilde{\Gamma}_-).$$

La solution approchée s'écrit

$$c_h(x, t) = \sum_{j=1}^N \psi_j(x) \left(c_j^k a_k(t) + c_j^{k+\frac{1}{2}} a_{k+\frac{1}{2}}(t) + c_j^{k+1} a_{k+1}(t) \right). \quad (4.2)$$

Selon le même principe que pour le degré 1 en temps, conduisant au système linéaire (1.14), on choisira parmi les fonctions-test celles qui s'annulent en t_k .

En utilisant l'indépendance et la génération de la base spatio-temporelle, il suffit de vérifier l'équation faible (1.7) avec les fonctions-test de base associées aux N nœuds du maillage et aux temps $t_{k+\frac{1}{2}}$ et t_{k+1} . Cela fournit $2N$ équations linéaires, pour les $2N$ inconnues que sont $\{c_j^{k+1/2}, c_j^{k+1}\}_{j=1}^N$. Les c_j^k sont, rappelons-le, les valeurs connues aux nœuds pour le temps t_k .

Le système d'équations correspondant au cas où la fonction-test w_h est $\psi_i \cdot a_{k+1/2}$ s'écrit

$$\begin{aligned} 0 &= \sum_{j=1}^N c_j^k \int_{\Omega} \left[\frac{\tau}{15} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{2}{3} (\nabla \psi_j | v) \psi_i + \frac{-2}{3} \psi_j (\nabla \psi_i | v) + \frac{-8}{3\tau} \psi_j \psi_i \right] dx \\ &+ \sum_{j=1}^N c_j^{k+1/2} \int_{\Omega} \left[\frac{8\tau}{15} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{16}{3\tau} \psi_j \psi_i \right] dx \\ &+ \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{\tau}{15} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{-2}{3} (\nabla \psi_j | v) \psi_i + \frac{2}{3} \psi_j (\nabla \psi_i | v) + \frac{-8}{3\tau} \psi_j \psi_i \right] dx \end{aligned} \quad (4.3)$$

et, pour w_h est $\psi_i \cdot a_{k+1}$:

$$\begin{aligned} 0 &= \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{-\tau}{30} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{-1}{6} (\nabla \psi_j | v) \psi_i + \frac{1}{6} \psi_j (\nabla \psi_i | v) + \frac{1}{3\tau} \psi_j \psi_i \right] dx \\ &+ \sum_{j=1}^N c_j^{k+1/2} \int_{\Omega} \left[\frac{\tau}{15} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{2}{3} (\nabla \psi_j | v) \psi_i + \frac{-2}{3} \psi_j (\nabla \psi_i | v) + \frac{-8}{3\tau} \psi_j \psi_i \right] dx \\ &+ \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{2\tau}{15} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{1}{2} (\nabla \psi_j | v) \psi_i + \frac{1}{2} \psi_j (\nabla \psi_i | v) + \frac{7}{3\tau} \psi_j \psi_i \right] dx. \end{aligned} \quad (4.4)$$

NB : les calculs ont été fait à la main et vérifiés à l'aide de Mathematica.

La question de la résolution pratique se pose. On a en effet un système par blocs de la

forme

$$\begin{pmatrix} A & B & C \\ D & E & F \end{pmatrix} \begin{pmatrix} c^k \\ c^{k+1/2} \\ c^{k+1} \end{pmatrix} = 0$$

qui équivaut à

$$\begin{pmatrix} B & C \\ E & F \end{pmatrix} \begin{pmatrix} c^{k+1/2} \\ c^{k+1} \end{pmatrix} = \begin{pmatrix} -A & 0 \\ 0 & -D \end{pmatrix} \begin{pmatrix} c^k \\ c^k \end{pmatrix}. \quad (4.5)$$

On observe que $E = A$ et $C = A^T$. Le système (4.5) peut s'écrire avec une légère simplification

$$\begin{pmatrix} B & A^T \\ A & F \end{pmatrix} \begin{pmatrix} c^{k+1/2} \\ c^{k+1} \end{pmatrix} = \begin{pmatrix} -A & 0 \\ 0 & -D \end{pmatrix} \begin{pmatrix} c^k \\ c^k \end{pmatrix}. \quad (4.6)$$

Notons que la matrice de gauche est symétrique car B et F le sont. Ce n'est pas étonnant, car c'est une conséquence de la symétrie de la forme $\mathcal{B}$. La matrice est aussi définie positive. Nous détaillons cela juste après : §4.2. Puisqu'on n'a pas besoin de $c^{k+1/2}$, on peut penser à réduire le système par blocs, ce qui correspond, par exemple, au système

$$(A^T - BA^{-1}F) c^{k+1} = (BA^{-1}D - A) c^k \quad (4.7)$$

pour autant que A soit régulière. C'est — au contenu des matrices près — le même système que pour le degré 1 en temps de STILS. On remarque au passage qu'une méthode pour un degré quelconque en temps se réduira à ce type de système linéaire, ce qui laisse présager que la qualité de la solution numérique ne va pas s'améliorer indéfiniment avec le degré en temps.

Du fait qu'on utilise la faible largeur de bande des matrices ou le fait qu'elles sont creuses (très faible densité de coefficients non nuls, d'où possibilité de compresser l'information), il n'est pas raisonnable, pour de grandes tailles, de calculer explicitement les matrices A^{-1} ou le complément de Schur $S = (A^T - BA^{-1}F)$. En effet, A^{-1} et S peuvent être quasiment pleines, ce qui pose un problème de stockage, en sus du temps pris pour les calculer. S'il n'y avait pas A^T dans S , on pourrait faire une résolution directe, mais ce n'est pas le cas. Le second membre dans (4.7) peut en revanche être directement calculé puisqu'on peut résoudre $Av = Dc^k$ et poser $w = Bv - Ac^k$.

On voit donc qu'il est plus simple et plus efficace, ici, de résoudre directement le système par blocs (4.5) que le système réduit (4.7).

4.2 Propriétés des matrices pour un degré M quelconque

Si on considère plus généralement le degré M , on a des fonctions de base en temps $a_{k+\frac{0}{M}}(t)$, $a_{k+\frac{1}{M}}(t)$, $\dots$, $a_{k+1}(t)$. Il suffit alors d'adapter la construction formelle du degré 1 faite au §1.2.3. Pour une solution

$$c_h(x, t) = \sum_{m=0}^M \sum_{j=1}^N \psi_j(x) a_{k+\frac{m}{M}}(t) c_j^{k+\frac{m}{M}} \quad (4.8)$$

(les c_j^k étant des « données » provenant de la frontière influx en temps, c'est-à-dire l'état en $t = t_k$) et une fonction-test de base

$$w_h(x, t) = \psi_i(x) a_{k+\frac{i}{M}}(t) \quad (4.9)$$

avec (l, i) dans $\{1 \dots M\} \times \{1 \dots N\}$, le problème approché (1.10) qui est, rappelons-le

$$\mathcal{B}(c_h, w_h) = \mathcal{L}(w_h) \quad \text{pour tout } w_h \in H_{0,h}$$

s'écrit plus explicitement

$$\sum_{m=1}^M \sum_{j=1}^N \mathcal{B} \left(\psi_j a_{k+\frac{m}{M}}, \psi_i a_{k+\frac{l}{M}} \right) \cdot c_j^{k+\frac{m}{M}} = \mathcal{L} \left(\psi_i a_{k+\frac{l}{M}} \right) - \sum_{j=1}^N \mathcal{B} \left(\psi_j a_k, \psi_i a_{k+\frac{l}{M}} \right) \cdot c_j^k. \quad (4.10)$$

On voit immédiatement que la matrice blocs d'indices (i, l) , (j, m) du membre de gauche est symétrique. Elle est aussi définie positive par coercitivité de $\mathcal{B}$: le calcul est identique à celui pour le degré 1, (1.17).

4.3 Degré 3 en temps : formulation STILS-TMQ3

Expérience faite, le degré $M = 3$ apporte un changement infime aux solutions numériques par rapport au degré 2. En comparant sur le cas du cylindre fendu, le maximum des différences ponctuelles sur l'ensemble des nœuds et des pas de temps est de 1/1000.

Présentons néanmoins les trois systèmes linéaires à résoudre qui correspondent, pour les fonctions-test, aux nœuds temporels $t_{k+1/3}, t_{k+2/3}, t_{k+1}$.

Les fonctions de base en temps $a_k, a_{k+1/3}, a_{k+2/3}, a_{k+1}$ sont comme toujours les polynômes d'interpolation pour les quadruples $(1; 0; 0; 0), (0; 1; 0; 0), (0; 0; 1; 0), (0; 0; 0; 1)$, respectivement, aux quadruples de temps $(t_n; t_{k+1/3}; t_{k+2/3}; t_{k+1})$.

Pour $w_h = \psi_i \cdot a_{k+1/3}$, c'est

$$\begin{aligned} 0 &= \sum_{j=1}^N c_j^k \int_{\Omega} \left[\frac{33\tau}{560} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{57}{80} (\nabla \psi_j | v) \psi_i + \frac{-57}{80} \psi_j (\nabla \psi_i | v) + \frac{-189}{40\tau} \psi_j \psi_i \right] dx \quad (4.11) \\ &+ \sum_{j=1}^N c_j^{k+1/3} \int_{\Omega} \left[\frac{27\tau}{70} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{54}{5\tau} \psi_j \psi_i \right] dx \\ &+ \sum_{j=1}^N c_j^{k+2/3} \int_{\Omega} \left[\frac{-27\tau}{560} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{-81}{80} (\nabla \psi_j | v) \psi_i + \frac{81}{80} \psi_j (\nabla \psi_i | v) + \frac{-297}{40\tau} \psi_j \psi_i \right] dx \\ &+ \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{-3\tau}{140} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{3}{10} (\nabla \psi_j | v) \psi_i + \frac{-3}{10} \psi_j (\nabla \psi_i | v) + \frac{27}{20\tau} \psi_j \psi_i \right] dx. \end{aligned}$$

Pour $w_h = \psi_i \cdot a_{k+2/3}$, on a

$$\begin{aligned} 0 &= \sum_{j=1}^N c_j^k \int_{\Omega} \left[\frac{-3\tau}{140} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{-3}{10} (\nabla \psi_j | v) \psi_i + \frac{3}{10} \psi_j (\nabla \psi_i | v) + \frac{27}{20\tau} \psi_j \psi_i \right] dx \quad (4.12) \\ &+ \sum_{j=1}^N c_j^{k+1/3} \int_{\Omega} \left[\frac{-27\tau}{560} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{81}{80} (\nabla \psi_j | v) \psi_i + \frac{-81}{80} \psi_j (\nabla \psi_i | v) + \frac{-297}{40\tau} \psi_j \psi_i \right] dx \end{aligned}$$

$$\begin{aligned}
& + \sum_{j=1}^N c_j^{k+2/3} \int_{\Omega} \left[\frac{27\tau}{40} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{54}{5\tau} \psi_j \psi_i \right] dx \\
& + \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{33\tau}{560} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{-57}{80} (\nabla \psi_j | v) \psi_i + \frac{57}{80} \psi_j (\nabla \psi_i | v) + \frac{-189}{40\tau} \psi_j \psi_i \right] dx.
\end{aligned}$$

Enfin, pour $w_h = \psi_i \cdot a_{k+1}$:

$$\begin{aligned}
0 & = \sum_{j=1}^N c_j^k \int_{\Omega} \left[\frac{19\tau}{1680} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{7}{80} (\nabla \psi_j | v) \psi_i + \frac{-7}{80} \psi_j (\nabla \psi_i | v) + \frac{-13}{40\tau} \psi_j \psi_i \right] dx \quad (4.13) \\
& + \sum_{j=1}^N c_j^{k+1/3} \int_{\Omega} \left[\frac{-3\tau}{140} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{-3}{10} (\nabla \psi_j | v) \psi_i + \frac{3}{10} \psi_j (\nabla \psi_i | v) + \frac{27}{20\tau} \psi_j \psi_i \right] dx \\
& + \sum_{j=1}^N c_j^{k+2/3} \int_{\Omega} \left[\frac{33\tau}{560} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{57}{80} (\nabla \psi_j | v) \psi_i + \frac{-57}{80} \psi_j (\nabla \psi_i | v) + \frac{-189}{40\tau} \psi_j \psi_i \right] dx \\
& + \sum_{j=1}^N c_j^{k+1} \int_{\Omega} \left[\frac{8\tau}{105} (\nabla \psi_j | v) (\nabla \psi_i | v) + \frac{1}{2} (\nabla \psi_j | v) \psi_i + \frac{1}{2} \psi_j (\nabla \psi_i | v) + \frac{37}{10\tau} \psi_j \psi_i \right] dx.
\end{aligned}$$

NB : les calculs pour le degré 3 ont été faits à l'aide de Mathematica.

Cela s'écrit en matrices blocs sous la forme

$$\begin{pmatrix} A & B & C & D \\ E & F & G & H \\ J & K & L & M \end{pmatrix} \begin{pmatrix} c^k \\ c^{k+1/3} \\ c^{k+2/3} \\ c^{k+1} \end{pmatrix} = 0$$

qui équivaut à

$$\begin{pmatrix} B & C & D \\ F & G & H \\ K & L & M \end{pmatrix} \begin{pmatrix} c^{k+1/3} \\ c^{k+2/3} \\ c^{k+1} \end{pmatrix} = \begin{pmatrix} -A & 0 & 0 \\ 0 & -E & 0 \\ 0 & 0 & -J \end{pmatrix} \begin{pmatrix} c^k \\ c^k \\ c^k \end{pmatrix} \quad (4.14)$$

si on veut séparer l'implicite de l'explicite. On retrouve dans (4.14) la forme du système général (4.10). Le système (4.14) est un véritable mini-problème STILS discrétisé, à $N \cdot 4$ nœuds, avec c^k comme condition au bord temporel. On observe les mêmes similitudes et symétries que pour les degrés 1 et 2. On le sait, la matrice de gauche est bel et bien symétrique, définie positive. Ainsi le système peut s'écrire

$$\begin{pmatrix} B & C & D \\ C^T & B & A^T \\ D^T & A & M \end{pmatrix} \begin{pmatrix} c^{k+1/3} \\ c^{k+2/3} \\ c^{k+1} \end{pmatrix} = \begin{pmatrix} -A & 0 & 0 \\ 0 & -D^T & 0 \\ 0 & 0 & -J \end{pmatrix} \begin{pmatrix} c^k \\ c^k \\ c^k \end{pmatrix} \quad (4.15)$$

et on peut tirer parti de la définie positivité pour la résolution du système (4.10).

4.4 Comparaisons entre STILS-TMQ1, TMQ2 et TMQ3

Commençons par le plus illustratif de nos tests, le cylindre fendu « plongé » dans un champ rotatif. La conservation de la concentration intégrale est presque parfaite pour les trois degrés ; les erreurs pourraient être attribuées à la formule de quadrature utilisée pour la calculer. Notons que les cases vides du tableau 4.1 sont dues à un temps de calcul excessif.

Comme indiqué plus haut, la différence expérimentale entre STILS-TMQ2 et TMQ3 est infime — en tout cas en gardant des éléments Q_1 en espace. Les contrastes entre degrés 1 et 2 sont autrement plus intéressants. Puisque le temps de calcul pour STILS-TMQ2 est environ 4 fois plus grand, il convient aussi de se demander s’il ne serait pas meilleur d’utiliser STILS-TMQ1 avec un pas de temps plus petit, par exemple 2 fois pour avoir le même nombre de pas effectif de temps, ou 4 fois pour avoir la même complexité du système linéaire.

Les expériences représentés sur le tableau 4.1 semblent indiquer que STILS-TMQ2 est préférable à STILS-TMQ1 avec un pas de temps plus petit. Une explication possible est que, pour un pas de temps plus petit, le nombre de Courant $C = \frac{\tau v}{h}$ devient également petit et qu’on s’approche de la limite de stabilité $\gamma(\omega) = 1$ en raison du facteur C^2 dans l’inéquation (3.11). De plus, le fait de raffiner le pas de temps ne corrige pas le retard apparent de TMQ1 qui croît lorsqu’on s’éloigne du centre. Avec TMQ2 on trouve une symétrie bien meilleure.

Qu’en est-il du schéma marche-en-temps de degré 3 STILS-TMQ3 ? La résolution prend plus de temps : les matrices sont 9/4 plus grandes que pour le degré 2 et expérimentalement on a même un quadruplement du temps de calcul. Du côté de la qualité, la différence est infime : sur tous les exemples, l’écart point par point entre les valeurs de TMQ2 et TMQ3 ne dépasse pas 1/1000. Cette stagnation persiste lorsqu’on augmente le degré en espace : voir la figure 4.2 ci-dessous. La conclusion (expérimentale) est évidente : il ne faut pas chercher d’amélioration dans les hauts degrés en temps. L’analyse de Fourier des trois schémas (§4.5) permettra de dire pourquoi, du moins dans le contexte unidimensionnel en espace.

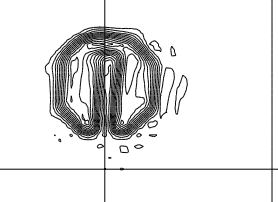
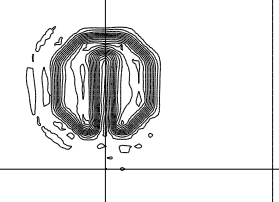
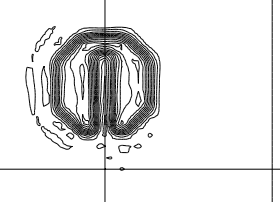
	TMQ1	TMQ2	TMQ3
$\tau = \frac{\pi}{400}$ $ C \leq \tau \cdot 50$ $\simeq 0,3927$			
	CPU = 12 min.	CPU = 106 min.	CPU = 481 min.
	$c_{\min} = -0,301$	$c_{\min} = -0,307$	$c_{\min} = -0,307$
	$c_{\max} = 1,330$	$c_{\max} = 1,334$	$c_{\max} = 1,334$

FIG. 4.2 – Q_2 en espace pour STILS-TMQ1, TMQ2 et TMQ3 ; état en $t = 2\pi$

La dimension 2 en espace a le désavantage visuel de ne pas montrer de manière univoque les phénomènes de dépassement de nos méthodes numériques. Aussi allons-nous passer à la dimension $d = 1$ pour les révéler et les comparer d’une méthode à l’autre. Posons la vitesse de convection $v = 1$ constante et définissons une condition initiale en forme de marche descendante $c_0(x) = \mathbb{1}_{x < 1/2}$. La solution « exacte » au sens des distributions de l’équation de transport à divergence nulle (1.4) est bien sûr le déplacement de ladite marche vers la droite, à vitesse $v = 1$, autrement dit $c(x, t) = c_0(x - t)$.

La figure 4.3 montre une comparaison de solutions numériques sur le domaine espace-temps $\Omega =]0; 1[\times]0; 1/2[$ produites par toutes les méthodes marche-en-temps étudiées jusqu'à présent : Galerkin TMQ1, Streamline diffusion TMQ1, STILS-TMQ1, TMQ2 et TMQ3. Le maillage en espace est de 100 nœuds et en temps, de 80 pas de temps. Les pas de temps affichés sont $t = 1/8 = 0,125$, $t = 3/10 = 0,3$ et $t = 19/40 = 0,475$. Pour $t = 1/2$ on doit avoir la marche sur le point $x = 1$. On voit que la solution numérique, pour toutes les méthodes, avance à la bonne vitesse, mais qu'une diffusion numérique a lieu. Les dépassements de STILS-TMQ1 sont particulièrement asymétriques, c'est-à-dire qu'ils s'attaquent surtout au haut de la marche. Streamline diffusion marche-en-temps est bien à mi-chemin entre Galerkin simple et STILS, comme expliqué au chapitre précédent. STILS-TMQ2 corrige sensiblement les dépassements et l'asymétrie de STILS-TMQ1, tandis que le degré 3, une fois de plus, n'apporte quasi rien.

4.5 Stabilité comparée de STILS-TMQ1, TMQ2 et TMQ3

Comme on l'a vu au chapitre précédent, il est judicieux de se ramener au cas à une dimension spatiale, $d = 1$.

4.5.1 Amplification de STILS-TMQ2

Le système aux différences obtenu pour les fonctions de base Q_1 (intégrales calculées dans la table 3.3) dans les formules pour le degré 2 en temps, (4.4) et (4.5), s'écrit

$$\begin{aligned}
 0 &= c_{i-1}^k (-6C^2 - 60C - 40) + c_i^k (12C^2 - 160) + c_{i+1}^k (-6C^2 + 60C - 40) + \\
 &\quad c_{i-1}^{k+1/2} (-48C^2 + 80) + c_i^{k+1/2} (96C^2 + 320) + c_{i+1}^{k+1/2} (-48C^2 + 80) + \\
 &\quad c_{i-1}^{k+1} (-6C^2 + 60C - 40) + c_i^{k+1} (12C^2 - 160) + c_{i+1}^{k+1} (-6C^2 - 60C - 40) \\
 0 &= c_{i-1}^k (3C^2 + 15C + 5) + c_i^k (-6C^2 + 20) + c_{i+1}^k (3C^2 - 15C + 5) + \\
 &\quad c_{i-1}^{k+1/2} (-6C^2 - 60C - 40) + c_i^{k+1/2} (12C^2 - 160) + c_{i+1}^{k+1/2} (-6C^2 + 60C - 40) + \\
 &\quad c_{i-1}^{k+1} (-12C^2 + 35) + c_i^{k+1} (24C^2 + 140) + c_{i+1}^{k+1} (-12C^2 + 35)
 \end{aligned} \tag{4.16}$$

où c^k est connu, $c^{k+1/2}$ et c^{k+1} sont inconnus. On reconnaît les blocs $A \ B \ C$ puis $D \ E \ F$, ici tridiagonaux, du système (4.5). En prenant la transformée de Fourier discrète telle qu'on l'a définie au §3.2, (3.9), on a la relation

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \varrho_A & \varrho_B & \varrho_C \\ \varrho_D & \varrho_E & \varrho_F \end{pmatrix} \begin{pmatrix} \hat{c}^k \\ \hat{c}^{k+1/2} \\ \hat{c}^{k+1} \end{pmatrix} \tag{4.17}$$

(analogue au cas unidimensionnel 3.10) où

$$\begin{aligned}
 \varrho_A(\omega) &= (-12C^2 - 80) \cos \omega + (120C) \ i \sin \omega + (12C^2 - 160) \\
 \varrho_B(\omega) &= (-96C^2 + 160) \cos \omega + (96C^2 + 320) \\
 \varrho_C(\omega) &= (-12C^2 - 80) \cos \omega + (-120C) \ i \sin \omega + (12C^2 - 160) \\
 \varrho_D(\omega) &= (6C^2 + 10) \cos \omega + (-30C) \ i \sin \omega + (-6C^2 + 20) \\
 \varrho_E(\omega) &= (-12C^2 - 80) \cos \omega + (120C) \ i \sin \omega + (12C^2 - 160) \\
 \varrho_F(\omega) &= (-24C^2 + 70) \cos \omega + (24C^2 + 140).
 \end{aligned}$$

On observe que $\varrho_B = \overline{\varrho_B}$ et $\varrho_F = \overline{\varrho_F}$ puisque $B = B^T$ et $F = F^T$, ainsi que $\varrho_C = \overline{\varrho_A}$ et $\varrho_E = \varrho_A$ car $C = A^T$ et $E = A$.

En réduisant le système (4.17), on trouve la relation $\hat{c}^{k+1}(\omega) = \gamma(\omega) \hat{c}^k(\omega)$ d'amplification sur le pas de temps τ avec

$$\gamma = \frac{\varrho_A^2 - \varrho_D \varrho_B}{\varrho_F \varrho_B - |\varrho_A|^2}.$$

4.5.2 Amplification de STILS-TMQ3

Pour le degré 3 en temps on obtient comme schéma unidimensionnel en espace le système triple

$$\begin{aligned}
0 = & \begin{array}{llll} c_{i-1}^k & (-99C^2 - 1197C - 1323) & + & c_i^k (198C^2 - 5292) & + & c_{i+1}^k (-99C^2 + 1197C - 1323) & + \\ c_{i-1}^{k+1/3} & (-648C^2 + 3024) & + & c_i^{k+1/3} (1296C^2 + 12096) & + & c_{i+1}^{k+1/3} (-648C^2 + 3024) & + \\ c_{i-1}^{k+2/3} & (81C^2 + 1701C - 2079) & + & c_i^{k+2/3} (-162C^2 - 8316) & + & c_{i+1}^{k+2/3} (81C^2 - 1701C - 2079) & + \\ c_{i-1}^{k+1} & (36C^2 - 504C + 378) & + & c_i^{k+1} (-72C^2 + 1512) & + & c_{i+1}^{k+1} (36C^2 + 504C + 378) \end{array} \\
0 = & \begin{array}{llll} c_{i-1}^{k+1} & (36C^2 + 504C + 378) & + & c_i^{k+1} (-72C^2 + 1512) & + & c_{i+1}^{k+1} (36C^2 - 504C + 378) \\ c_{i-1}^{k+2/3} & (81C^2 - 1701C - 2079) & + & c_i^{k+2/3} (-162C^2 - 8316) & + & c_{i+1}^{k+2/3} (81C^2 + 1701C - 2079) & + \\ c_{i-1}^{k+1/3} & (-648C^2 + 3024) & + & c_i^{k+1/3} (1296C^2 + 12096) & + & c_{i+1}^{k+1/3} (-648C^2 + 3024) & + \\ c_{i-1}^{k+1} & (-99C^2 + 1197C - 1323) & + & c_i^{k+1} (198C^2 - 5292) & + & c_{i+1}^{k+1} (-99C^2 - 1197C - 1323) \end{array} \quad (4.18) \\
0 = & \begin{array}{llll} c_{i-1}^k & (-19C^2 - 147C - 91) & + & c_i^k (38C^2 - 364) & + & c_{i+1}^k (-19C^2 + 147C - 91) & + \\ c_{i-1}^{k+1} & (36C^2 + 504C + 378) & + & c_i^{k+1} (-72C^2 + 1512) & + & c_{i+1}^{k+1} (36C^2 - 504C + 378) \\ c_{i-1}^k & (-99C^2 - 1197C - 1323) & + & c_i^k (198C^2 - 5292) & + & c_{i+1}^k (-99C^2 + 1197C - 1323) & + \\ c_{i-1}^{k+1} & (-128C^2 + 1036) & + & c_i^{k+1} (256C^2 + 4144) & + & c_{i+1}^{k+1} (-128C^2 + 1036) \end{array}
\end{aligned}$$

à partir de (4.12), (4.13) et (4.14). La relation en transformée de Fourier s'écrit alors

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \varrho_A & \varrho_B & \varrho_C & \varrho_D \\ \varrho_E & \varrho_F & \varrho_G & \varrho_H \\ \varrho_J & \varrho_K & \varrho_L & \varrho_M \end{pmatrix} \begin{pmatrix} \hat{c}^k \\ \hat{c}^{k+1/3} \\ \hat{c}^{k+2/3} \\ \hat{c}^{k+1} \end{pmatrix} \quad (4.19)$$

où

$$\begin{aligned}
\varrho_A(\omega) &= (-198C^2 - 2646) \cos \omega + (2394C) i \sin \omega + (198C^2 - 5292) \\
\varrho_B(\omega) &= (-1296C^2 + 6048) \cos \omega + (1296C^2 + 12096) \\
\varrho_C(\omega) &= (162C^2 - 4158) \cos \omega + (-3402C) i \sin \omega + (-162C^2 - 8316) \\
\varrho_D(\omega) &= (72C^2 + 756) \cos \omega + (1008C) i \sin \omega + (-72C^2 + 1512) \\
\varrho_E(\omega) &= \overline{\varrho_D(\omega)} \\
\varrho_F(\omega) &= \overline{\varrho_C(\omega)} \\
\varrho_G(\omega) &= \varrho_B(\omega) \\
\varrho_H(\omega) &= \overline{\varrho_A(\omega)} \\
\varrho_J(\omega) &= (-38C^2 - 182) \cos \omega + (294C) i \sin \omega + (38C^2 - 364) \\
\varrho_K(\omega) &= \overline{\varrho_D(\omega)} \\
\varrho_L(\omega) &= \varrho_A(\omega) \\
\varrho_M(\omega) &= (-256C^2 + 2072) \cos \omega + (256C^2 + 4144).
\end{aligned}$$

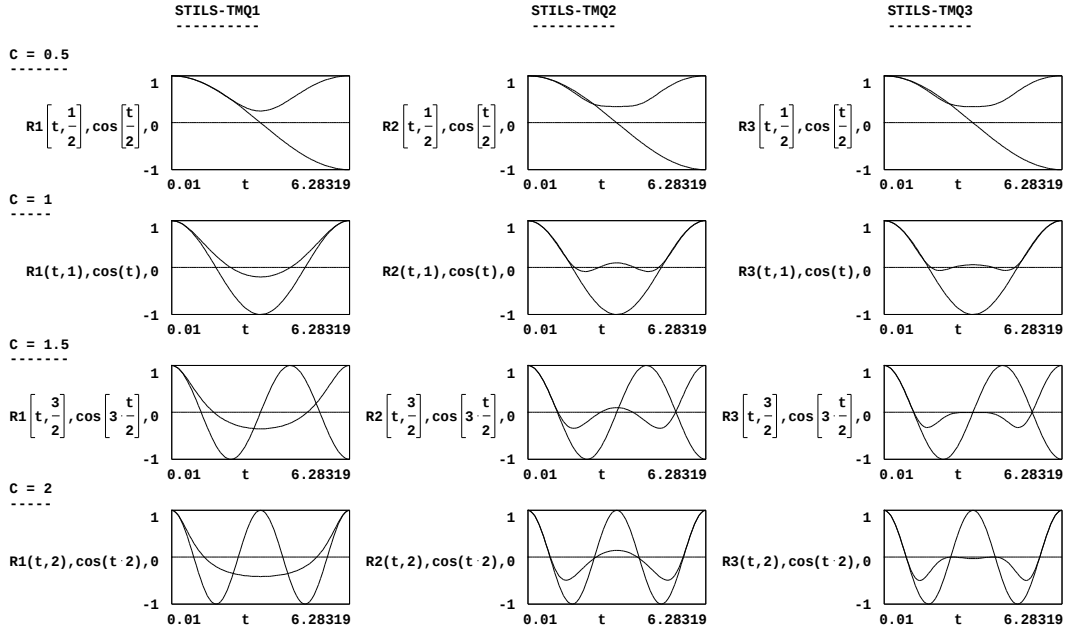
En réduisant (4.19), on obtient la relation $\hat{c}^{k+1}(\omega) = \gamma(\omega) \hat{c}^k(\omega)$ avec

$$\gamma = \frac{(\varrho_A \varrho_B - \varrho_C \overline{\varrho_D}) \cdot (\overline{\varrho_D} \varrho_B - \varrho_A \overline{\varrho_C}) - (\varrho_J \varrho_B - \varrho_A \overline{\varrho_D}) \cdot (\varrho_B^2 - \varrho_C \overline{\varrho_C})}{(\varrho_M \varrho_B - \varrho_D \overline{\varrho_D}) \cdot (\varrho_B^2 - \varrho_C \overline{\varrho_C}) - (\varrho_A \varrho_B - \varrho_C \overline{\varrho_D}) \cdot (\overline{\varrho_A} \varrho_B - \varrho_D \overline{\varrho_C})}.$$

4.5.3 Stabilité

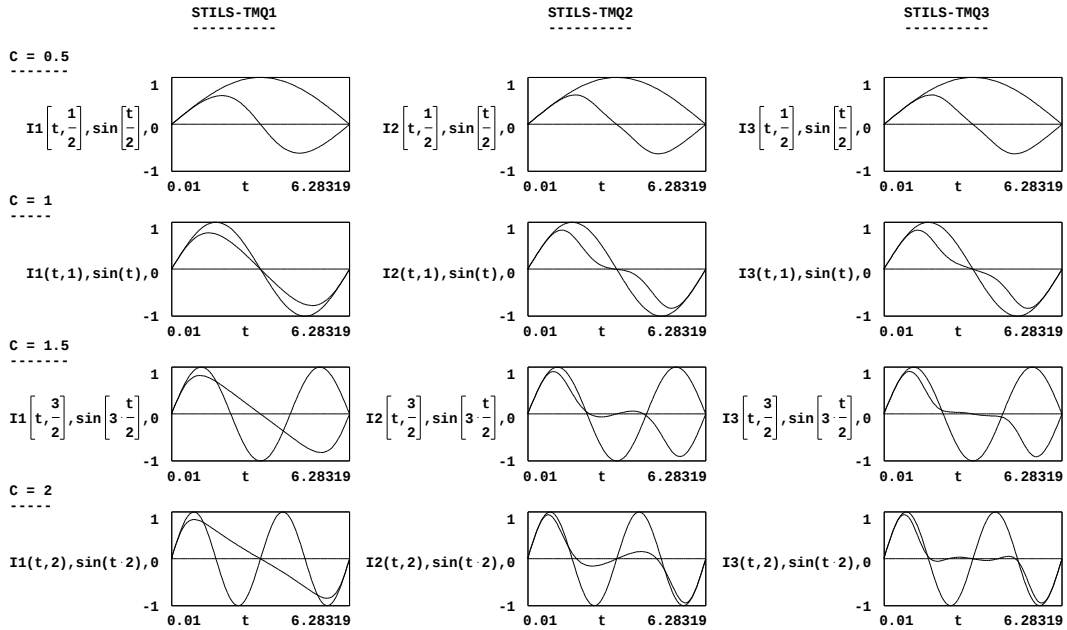
Pour l'heure une preuve explicite de la stabilité pour les degrés 2 et 3 n'a pas pu être faite, mais numériquement il est évident que le module de $\gamma(\omega)$ est borné par 1 pour toutes les valeurs du nombre de Courant C . La figure 4.4 représente les courbes des fonctions d'amplification $\overline{\gamma_{Q_1}}$, $\overline{\gamma_{Q_2}}$, $\overline{\gamma_{Q_3}}$ (de TMQ1, 2, 3) pour les valeurs de C suivantes : 0.2, 0.5, 0.9, 2. Comme pour les schémas Q_1 en temps, on a avantage à séparer les parties réelles et imaginaires de γ pour voir le déphasage — figures 4.5 et 4.6. On compare, pour la partie réelle, les graphes $(t, \text{Re}(\gamma(t)))$ aux graphes $(t, \cos(t))$:

Amplification - partie réelle

FIG. 4.5 – Graphe de $Re(\gamma)$ comparé à $\cos$ pour TMQ1 (R1), TMQ2 (R2), TMQ3 (R3)

On voit bien que, indépendamment du degré, un nombre de Courant C le plus proche de 1 est le plus souhaitable. C'est aussi le cas pour la partie imaginaire, dont les graphes $(t, -Im(\gamma(t)))$ comparés aux graphes $(t, \sin(t))$ nous donnent

Amplification - partie imaginaire

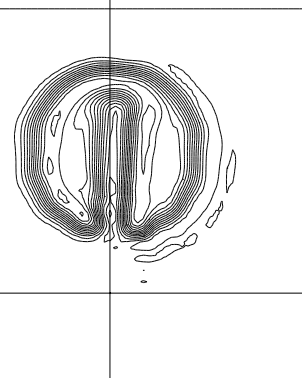
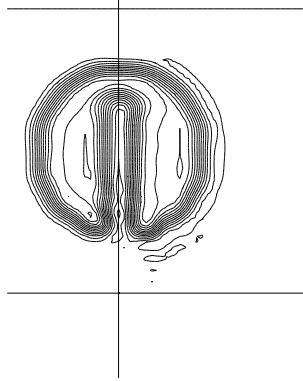
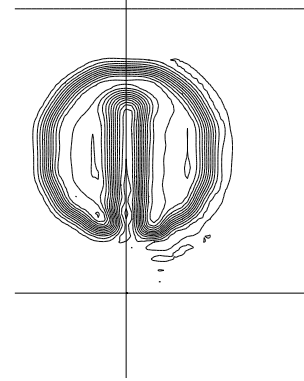
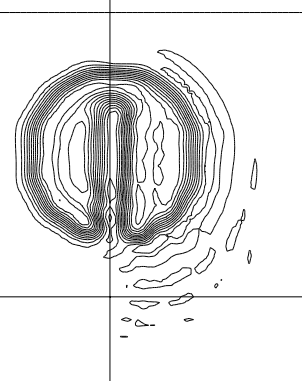
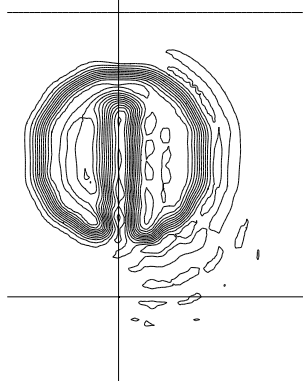
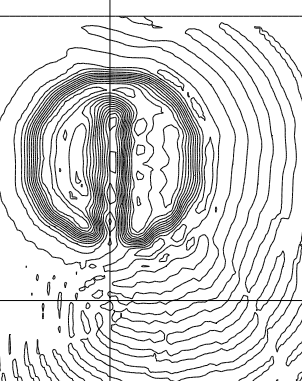
FIG. 4.6 – Graphe de $-Im(\gamma)$ comparé à $\sin$ pour TMQ1 (I1), TMQ2 (I2), TMQ3 (I3)

La ressemblance des graphes des fonctions d'amplification γ_{Q_2} et γ_{Q_3} indique une bonne

convergence selon le degré en temps.

Une estimation explicite à partir de la différence explicite $\gamma_{Q_2} - \gamma_{Q_3}$ rebute les logiciels de calcul formel les plus ambitieux. En tout cas la faible différence observée numériquement pour chaque valeur-test de C explique la ressemblance des solutions numériques de l'équation de transport, puisque, nous l'avons vu en 3.2.3, le système linéaire pour une CL périodique est diagonalisable et les N valeurs propres sont précisément $\gamma(\frac{2\pi j}{N})$, $k \in \{1 \dots N\}$, auxquels correspondent des vecteurs propres invariants par rapport au système.

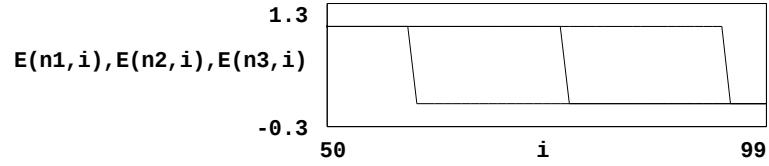
On peut en conclure que si on veut mettre en œuvre des fonctions de base polynomiales sur les éléments et partout continues, le degré deux en temps semble être un optimum ; le degré trois apporte un changement infime pour un temps de calcul considérablement plus grand.

	TMQ1	TMQ2	TMQ3
$\tau = \frac{\pi}{400}$ $ C \leq \tau \cdot 50$ $\simeq 0,3927$	 CPU = 40 min. $c_{\min} = -0,261$ $c_{\max} = 1,239$	 CPU = 166 min. $c_{\min} = -0,253$ $c_{\max} = 1,239$	 CPU = 693 min. $c_{\min} = -0,253$ $c_{\max} = 1,239$
$\tau = \frac{\pi}{800}$ $ C \leq \tau \cdot 50$ $\simeq 0,1963$	 CPU = 81 min. $c_{\min} = -0,302$ $c_{\max} = 1,293$	 CPU = 381 min. $c_{\min} = -0,299$ $c_{\max} = 1,292$	—
$\tau = \frac{\pi}{1600}$ $ C \leq \tau \cdot 50$ $\simeq 0,0982$	 CPU = 159 min. $c_{\min} = -0,336$ $c_{\max} = 1,332$	—	—

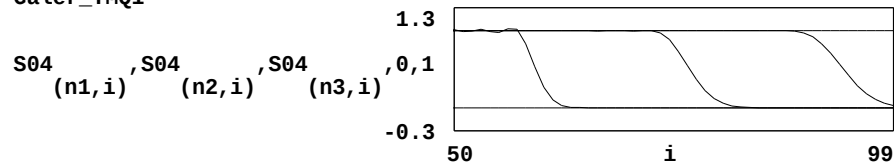
TAB. 4.1 – Problème du cylindre pour STILS-TMQ1, TMQ2 et TMQ3; divers pas de temps τ ; état en $t = 2\pi$

Profils 1D
=====

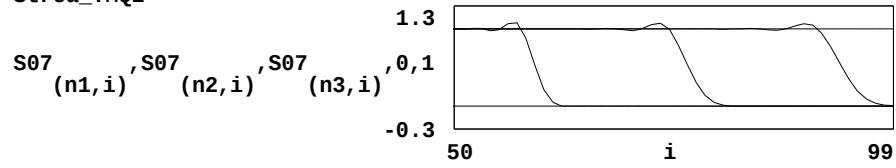
Exacte



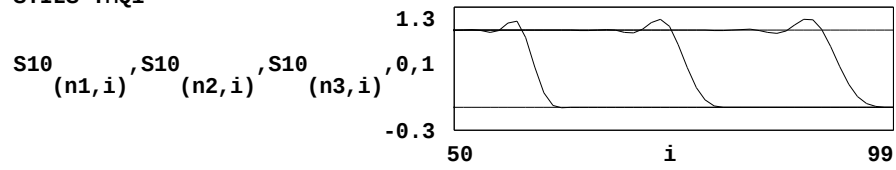
Galer_TMQ1



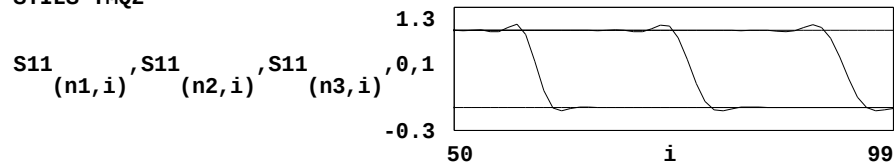
Strea_TMQ1



STILS-TMQ1



STILS-TMQ2



STILS-TMQ3

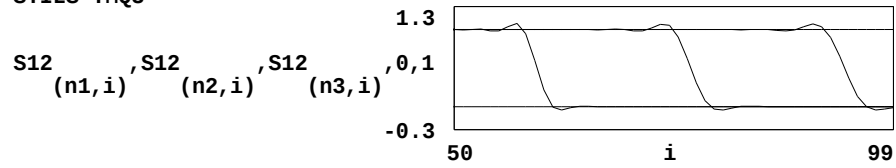
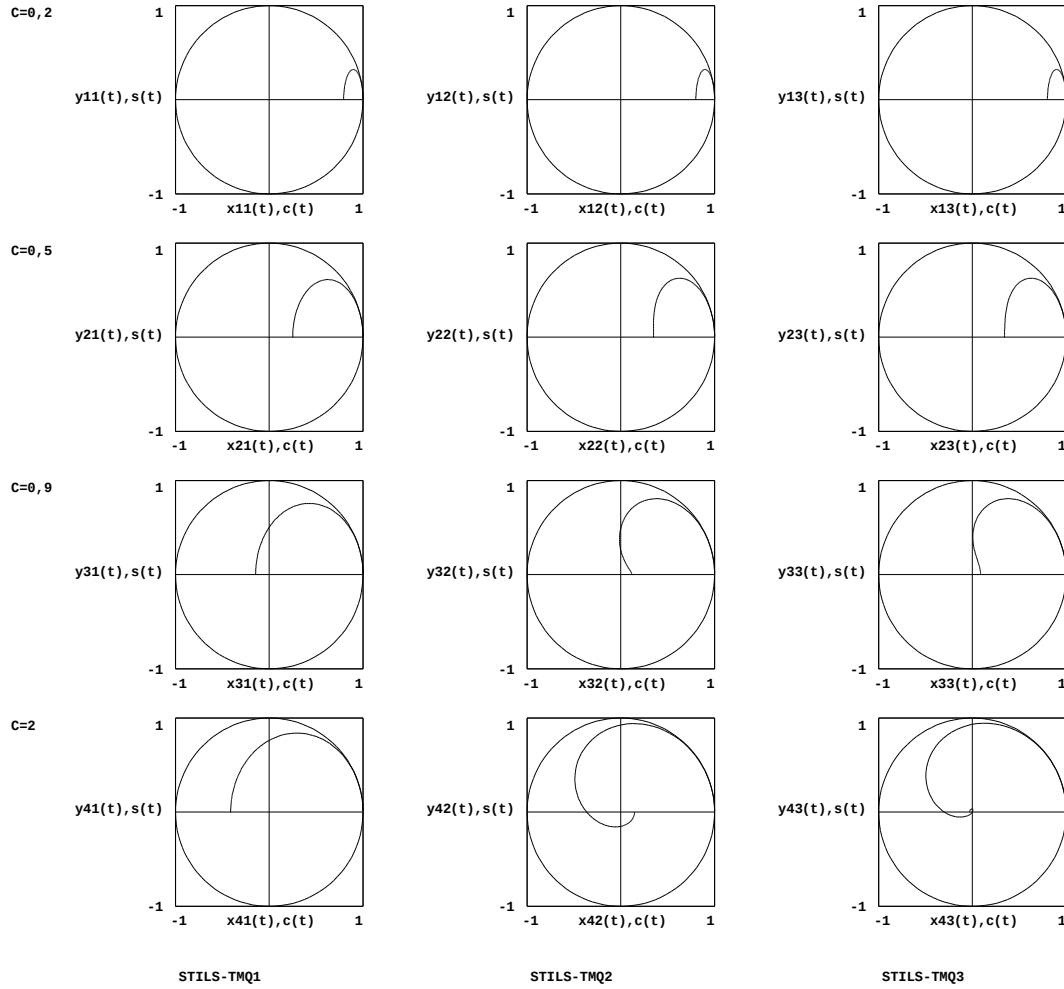


FIG. 4.3 – Solution « exacte » et méthodes marche-en-temps, pour $t = 0,125$ (n1), $t = 0,3$ (n2) et $t = 0,475$ (n3)

FIG. 4.4 – Amplification $\overline{\gamma_{Q_1}}$, $\overline{\gamma_{Q_2}}$, $\overline{\gamma_{Q_3}}$; $C = 0.2$, $C = 0.5$, $C = 0.9$, $C = 2$

Chapitre 5

Essais de méthodes discontinues

Il est évident que des fonctions de base continues présentent un défaut majeur lorsqu'il s'agit d'approcher la solution d'une équation de transport : les discontinuités sont grossièrement approchées. Le phénomène de Gibbs provoque les oscillations bien connues autour de celles-ci. Par ailleurs, une équation de pure convection transporte les discontinuités le long des caractéristiques; on ne se débarrasse donc jamais ni des discontinuités, ni des phénomènes numériques qui les accompagnent. Il convient alors d'enrichir la base de fonctions en admettant qu'elles puissent avoir un saut aux frontières spatiales ou temporelles entre les éléments. Selon les différentes sources sur le sujet, l'idée remonte à [40].

Une idée d'application peut être de prendre des fonctions discontinues en temps seulement. On peut y voir les atouts suivants :

- cette approche fonctionne pour la méthode de Galerkin simple en espace-temps (Discontinuous Galerkin, DG)
- pour un maillage non-trivial, il est plus simple de garder la continuité en espace, sans quoi il faut tenir compte des flux entrants et sortants de chaque élément et faire la résolution élément par élément le long des caractéristiques selon la technique décrite au §9.8 du livre de Johnson [32].

Prenons le domaine en espace-temps $\tilde{\Omega} = \Omega \times]0, T[$ et restons attentifs au fait que les fonctions auront des sauts aux temps $0 < t_1 < \dots < t_k < t_{k+1} < \dots < t_F = T$. Si on dégage les termes temporels du problème faible (1.7), on obtient

$$\int_{\tilde{\Omega}} [(\nabla u|v)(\nabla w|v) + (\nabla u|v)\partial_t w + \partial_t u(\nabla w|v) + \partial_t u\partial_t w] \, dx \, dt = \int_{\tilde{\Omega}} f \cdot [(\nabla w|v) + \partial_t w] \, dx \, dt. \quad (5.1)$$

La première difficulté (légère) est que l'espace fonctionnel H pour une solution u et les fonctions-test w ne peut pas être $H(v, \tilde{\Omega})$ ou un sous-espace de celui-ci, puisque des distributions $u \in H$ peuvent avoir des sauts en $t = t_k$, donc des dérivées qui sont des masses de Dirac. Il faut donc enrichir $H(v, \tilde{\Omega})$. Une distribution u de H s'écrira

$$u(x, t) = \bar{u}(x, t) + \sum_{k=1}^F h_{t_k}(t) [\bar{u}]_{t_k}(x)$$

où $\bar{u}$ est défini hors des pas de temps, c'est-à-dire dans $H(v, \Omega \times]0, t_1[\cup \dots \cup]t_{F-1}, t_F[)$, $h_{t_k}(t)$ est la fonction de Heaviside, qui vaut 0 pour $t < t_k$ et 1 pour $t \geq t_k$, et enfin $[\bar{u}]_{t_k}(x)$ est le saut de $\bar{u}$ par-dessus $t = t_k$ et au point $x \in \Omega$. Pour Galerkin, on peut formuler l'équation

faible en espace-temps sur un pas de temps, avec $\tilde{\Omega}_k = \Omega \times]t_k, t_{k+1}[$:

$$\int_{\tilde{\Omega}_k} ((\nabla u|v) + \partial_t u) w \, dx \, dt + \int_{\Omega_k} [u]_{t_k}(x) w \, dx = \int_{\tilde{\Omega}_k} f \cdot w \, dx \, dt. \quad (5.2)$$

et en tirer une méthode d'éléments finis un peu meilleure que pour des fonctions partout continues [32]. Dans le programme Flux-Transport, c'est la méthode `Galer_TDQ1`.

Malheureusement cela ne passe pour STILS puisqu'on a dans (5.1) des carrés de masses de Dirac, c'est-à-dire l'infini ! Il faut donc changer fondamentalement de formulation.

Une bonne idée, pour ce faire, est de *découpler* u et $p = \partial_t u$ dans une approche d'éléments mixtes. Partant du cadre théorique du chapitre II de [9] *Approximation of saddle point problems*, et spécialement le cadre général donné au §II.1.2, on peut poser le problème suivant : trouver $u \in V$ et $p \in Q$ tels que

$$a(u, w) + b(w, p) = 0 \quad \text{pour tout } w \in V \quad (5.3)$$

$$b(u, q) - c(p, q) = 0 \quad \text{pour tout } q \in Q \quad (5.4)$$

où $V = V_h \otimes P_0$ et $Q = V_h \otimes Q_1$. Il s'agit de donner un contenu aux formes a , b , c , mais rappelons d'abord que V_h désigne l'ensemble des fonctions exprimables dans la base d'éléments finis en espace $\{\psi_1 \dots \psi_N\}$. P_0 est ici l'ensemble des fonctions constantes sur les intervalles $]t_k, t_{k+1}[$, $k = 0 \dots F-1$ et Q_1 est l'ensemble des fonctions affines sur ces mêmes intervalles et continues.

Le choix d'une meilleure régularité (Q_1) pour p que pour u (P_0) peut sembler étrange puisque p est censée être la dérivée de u , néanmoins il est usuel pour les formulations mixtes puisqu'on *recherche* cette régularité — une fois de plus les inventions et la pratique de l'ingénieur précèdent les explications du mathématicien. Le choix ci-dessus est directement inspiré par celui utilisé pour résoudre l'équation elliptique $-u''(x) = f(x)$ — cf *Mixed finite element methods for Dirichlet's problem*, §IV.1.2 de [9].

On voit assez vite que le seul choix qui tienne pour les formes a , b , c est

$$a(u, w) = \int_{\tilde{\Omega}} [(\nabla u|v)(\nabla w|v) + (\nabla u|v)\partial_t w + \partial_t u(\nabla w|v)] \, dx \, dt \quad (5.5)$$

$$b(w, p) = \int_{\tilde{\Omega}} p \cdot \partial_t w \, dx \, dt \quad (5.6)$$

$$c(p, q) = \int_{\tilde{\Omega}} q \cdot p \, dx \, dt. \quad (5.7)$$

Ainsi (5.3) recouvre l'équation STILS (5.1) et (5.4) est l'équation de couplage. Il faut être attentifs bien sûr aux sauts et donc au développement des termes $\partial_t u$ et $\partial_t w$, mais au moins on a esquivé le problème du « Dirac au carré ».

Les expériences numériques sont totalement divergentes, ce qui fait penser que l'approche a un défaut fondamental. Un remède pourrait être de considérer des fonctions discontinues en temps *et* en espace. En effet pour $d = 1$ et $v = 1$ déjà, (l'équation de transport s'écrit alors $\partial_x u + \partial_t u = 0$) la solution pour une condition initiale en forme de marche est discontinue aussi bien en x qu'en t .

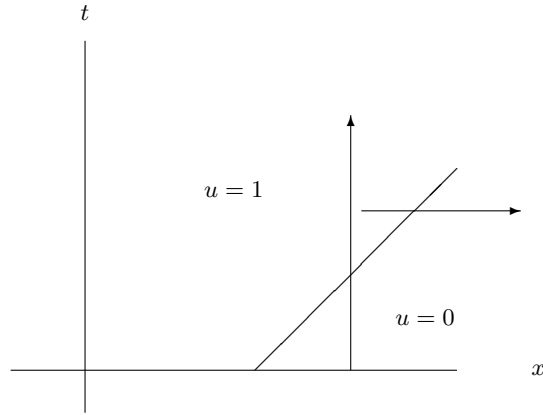


FIG. 5.1 – Discontinuité dans les deux dimensions

Une piste pour STILS (qui est une formulation de type diffusive) pourrait être le développement récent de méthodes du type LDG (Local Discontinuous Galerkin) pour des problèmes diffusifs, introduit par Oden, Babuška et Baumann [46] et étudié par Cockburn et Shu [11].

Chapitre 6

STILS pour des équations non linéaires

Jusqu'à présent, nous avons pratiqué des éléments finis avec moindres-carrés et intégration en espace-temps sur des équations d'advection (ou convection) linéaires. Qu'en est-il des équations de premier ordre non linéaires ?

Comme point de départ, nous considérerons la dimension spatiale 1 où les équations sont aussi désignées par le nom de *loi de conservation scalaire*. C'est sous ce nom que les équations de type

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \quad (6.1)$$

apparaissent dans la littérature (Ouvrages de référence : [23, 24, 35]).

Rappelons le fait le plus intéressant à ce propos : une solution d'un problème à valeur initiale

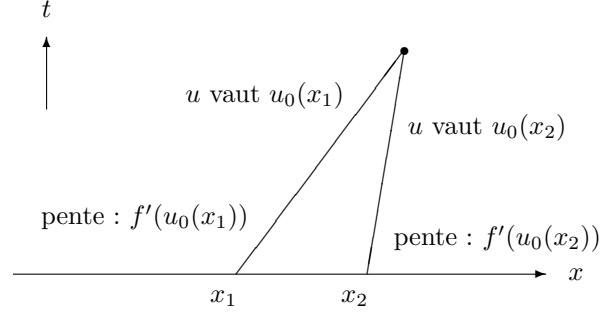
$$u(x, 0) = u^0(x) \text{ pour tout } x \in \mathbb{R}$$

est constante sur les trajectoires $(x(t), t)$ d'équation

$$x'(t) = f'(u(x(t), t)).$$

Il s'ensuit que ces trajectoires — les caractéristiques — sont des segments de droites. Issues de $(x, t) = (x_0, 0)$, leur pente dans le plan (t, x) vaut $f'(u^0(x_0))$. On l'a vu dans le contexte du transport avec l'équation caractéristique (2.10).

Dans l'équation de convection, f est linéaire, de sorte que les droites caractéristiques ne s'intersectent jamais. En fait, on a un contexte plus général avec $f(u, x, t) = v(x, t) \cdot u(x, t)$, où le champ de vitesse v se résume à un scalaire, mais ne dépend pas de u . L'intégration des caractéristiques peut se faire indéfiniment du moment que v est C^1 . Dans le cas non-linéaire, les pentes des droites sont déterminées par la valeur de la solution en un de leur points. Mais alors, elles peuvent s'intersecter malgré une condition initiale lisse. Au-delà du point d'intersection, on ne sait ni la valeur, ni donc la pente de la droite où elle doit se propager !



On a dans ce cas une onde de choc, c'est à dire une discontinuité qui naît naturellement de l'équation. Une solution ne peut plus être définie comme une fonction de classe C^1 vérifiant *stricto sensu* l'équation de conservation. On a avantage à la définir au sens des distributions. En effet, en intégrant contre une fonction-test à support compact $\varphi \in \mathcal{D}(\mathbb{R} \times [0; \infty[)$ ¹ on obtient une équation faible pour la loi de conservation (6.1) :

$$-\langle \partial_t T_u + \partial_x T_{f(u)}, \varphi \rangle_{\mathbb{R} \times [0; \infty[} = \langle u, \partial_t \varphi \rangle + \langle f(u), \partial_x \varphi \rangle + \int_{\mathbb{R}} u(x, 0) \varphi(x, 0) dx = 0. \quad (6.2)$$

Il est alors intéressant d'essayer des méthodes numériques basées sur la formulation STILS, à partir de la loi de conservation (6.1), et de comparer le résultat à une solution faible (solution de (6.2) lorsqu'on en a une à disposition).

Afin de situer la qualité de STILS pour les lois de conservation scalaires, nous l'aborderons à la lumière de notions connues comme les ondes de choc, la condition de Rankine-Hugoniot et l'entropie.

6.1 Equation de Burgers

Pour avoir une bonne idée de STILS dans le contexte non linéaire, faisons appel à l'équation aux dérivées partielles non linéaire qui sans doute allie le mieux une formulation simple et un comportement retors : l'équation de Burgers. C'est (6.1) avec $f(u) = \frac{u^2}{2}$:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0. \quad (6.3)$$

L'approche STILS consistant à résoudre cette équation sous forme intégrée et au sens des moindres-carrés, on peut suivre les premières étapes du §1.2. La fonctionnelle $\mathcal{A}$ est donc, dans le cas de l'équation de Burgers,

$$\mathcal{A}u = \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x}$$

et la fonctionnelle d'énergie J , définie par

$$J(u) = \frac{1}{2} \int_{\Omega} (\mathcal{A}u)^2 dx dt$$

¹Rappelons que sur un fermé tel que $\mathbb{R} \times [0; \infty[$, il faut comprendre : φ fonction lisse à support compact sur le plan ($\in \mathcal{D}(\mathbb{R}^2)$), mais restreinte à $\mathbb{R} \times [0; \infty[$, ce qui fait qu'on n'a pas de valeur forcément nulle sur $t = 0$

a ici pour dérivée directionnelle (définie par (1.5))

$$[DJ(u)]w = \lim_{h \rightarrow 0} \frac{1}{h} (J(u + hw) - J(u)) = \int_{\tilde{\Omega}} \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \right) \left(\frac{\partial w}{\partial t} + w \frac{\partial u}{\partial x} + u \frac{\partial w}{\partial x} \right) dx dt \quad (6.4)$$

L'équation faible de recherche d'un point stationnaire de (6.3) dans un espace fonctionnel H convenu est donc

$$[DJ(u)]w = 0 \quad \text{pour tout } w \in H. \quad (6.5)$$

Remarque : La forme $\mathcal{B}(u, w) = [DJ(u)]w$ est *non-linéaire*, contrairement à ce qu'on a pour une EDP linéaire (§1.2.2).

Passons à une forme approchée de l'équation faible au sens de STILS (6.5).

6.1.1 Séparation espace-temps

Pour aborder l'approximation, on considère le problème local sur $\mathbb{R} \times]t_n, t_{n+1}[$. Comme dans le cas d'une EDP linéaire (§1.2.3), on choisit en espace une base $\{\psi_1 \dots \psi_N\}$ finie, quelconque, d'un sous-espace V_h de $L^2(\mathbb{R})$, et une base affine

$$\begin{aligned} a_n(t) &= \frac{1}{\tau}(t_{n+1} - t) \\ a_{n+1}(t) &= \frac{1}{\tau}(t - t_n) \end{aligned}$$

pour le temps $t \in [t_n, t_{n+1}]$, avec $\tau = t_{n+1} - t_n$. On pose E_2 le sous-espace de dimension 2 de $\mathcal{C}[t_n, t_{n+1}]$ engendré par a et b .

Une approximation du problème faible de Burgers (6.5) consiste à chercher $u_h \in H_h = V_h \otimes E_2 \cap L^2(\mathbb{R})$ tel que

$$[DJ(u_h)]w_h = 0 \quad (6.6)$$

pour tout w_h dans H_h .

Pour alléger les calculs, on prendra $[0, \tau]$ au lieu de $[t_n, t_{n+1}]$.

La solution approchée dans H_h peut s'écrire

$$u_h(x, t) = \sum_{j=1}^N \psi_j(x) s_j(t)$$

où

$$s_j(t) = s_j^n a_n(t) + s_j^{n+1} a_{n+1}(t).$$

Les coefficients s_j^n , $j \in \{1 \dots N\}$ sont connus tandis que les s_j^{n+1} , $j \in \{1 \dots N\}$ sont les inconnues. Les fonctions-test s'annulant en $t = 0$ (choix), il suffira de vérifier l'équation faible (6.6) pour les toutes les fonctions de base $w_h(x, t) = \psi_i(x) \cdot a_{n+1}(t)$, $i \in \{1 \dots N\}$.

L'équation (6.1) correspond alors à un système de N équations non linéaires à N inconnues (le vecteur s^{n+1}) qui s'écrit

$$f(s^{n+1}) = 0 \quad (6.7)$$

où $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$ est définie par $f = (f_1, \dots, f_N)^\top$ avec

$$\begin{aligned} f_i(s^{n+1}) &= [DJ(u_h)](\psi_i b) = \sum_{j=1}^N \int_0^\tau s'_j(t) a'_{n+1}(t) dt \int_{\mathbb{R}} \psi_j(x) \psi_i(x) dx + \\ &\quad \sum_{j,l=1}^N \int_0^\tau s'_j(t) s_l(t) a_{n+1}(t) dt \int_{\mathbb{R}} (\psi_j(x) \psi'_l(x) \psi_i(x) + \psi_j(x) \psi_l(x) \psi'_i(x)) dx + \\ &\quad \sum_{j,k=1}^N \int_0^\tau s_j(t) s_k(t) a'_{n+1}(t) dt \int_{\mathbb{R}} \psi_j(x) \psi'_k(x) \psi_i(x) dx + \\ &\quad \sum_{j,k,l=1}^N \int_0^\tau s_j(t) s_k(t) s_l(t) a_{n+1}(t) dt \int_{\mathbb{R}} (\psi_j(x) \psi'_k(x) \psi'_l(x) \psi_i(x) + \psi_j(x) \psi'_k(x) \psi_l(x) \psi'_i(x)) dx. \end{aligned}$$

Un calcul des intégrales en temps montre que

$$\begin{aligned} f_i(s^{n+1}) &= \sum_{j=1}^N \left(\frac{s_j^{n+1} - s_j^n}{\tau} \right) \int_{\mathbb{R}} \psi_j(x) \psi_i(x) dx + \\ &\quad \sum_{j,l=1}^N \frac{1}{6} (s_j^{n+1} - s_j^n) (s_l^n + 2s_l^{n+1}) \int_{\mathbb{R}} (\psi_j(x) \psi'_l(x) \psi_i(x) + \psi_j(x) \psi_l(x) \psi'_i(x)) dx + \\ &\quad \sum_{j,k=1}^N \frac{1}{6} (s_j^k (2s_k^n + s_k^{n+1}) + s_j^{k+1} (s_k^n + 2s_k^{n+1})) \int_{\mathbb{R}} \psi_j(x) \psi'_k(x) \psi_i(x) dx + \\ &\quad \sum_{j,k,l=1}^N \frac{\tau}{60} [s_j^n (3s_k^n s_l^n + 2s_k^{n+1} s_l^n + 2s_k^n s_l^{n+1} + 3s_k^{n+1} s_l^{n+1}) + \\ &\quad s_j^{n+1} (s_k^n (2s_l^n + 3s_l^{n+1}) + 3s_k^{n+1} (s_l^n + 4s_l^{n+1}))] \\ &\quad \int_{\mathbb{R}} (\psi_j(x) \psi'_k(x) \psi'_l(x) \psi_i(x) + \psi_j(x) \psi'_k(x) \psi_l(x) \psi'_i(x)) dx. \end{aligned}$$

En notant $u = s^{n+1} = (s_1^{n+1}, \dots, s_N^{n+1})^\top$, $s = s^n$ et

$$\begin{aligned} \alpha_{i,j} &= \int_{-\infty}^{\infty} \psi_j(x) \psi_i(x) dx \\ \beta_{i,j,l} &= \int_{-\infty}^{\infty} \psi_j(x) (\psi'_l(x) \psi_i(x) + \psi_l(x) \psi'_i(x)) dx \\ \gamma_{i,j,k} &= \int_{-\infty}^{\infty} \psi_j(x) \psi'_k(x) \psi_i(x) dx \end{aligned} \tag{6.8}$$

$$\varepsilon_{i,j,k,l} = \int_{-\infty}^{\infty} \psi_j(x) \psi'_k(x) (\psi'_l(x) \psi_i(x) + \psi_l(x) \psi'_i(x)) \, dx$$

on obtient simplement

$$\begin{aligned} f_i(u) = & \sum_{j=1}^N \left(\frac{u_j - s_j}{\tau} \right) \alpha_{i,j} + \\ & \sum_{j,l=1}^N \frac{1}{6} (u_j - s_j) (s_l + 2u_l) \beta_{i,j,l} + \\ & \sum_{j,k=1}^N \frac{1}{6} \left(s_j^k (2s_k + u_k) + u_j (s_k + 2u_k) \right) \gamma_{i,j,k} + \\ & \sum_{j,k,l=1}^N \frac{\tau}{60} [s_j (3s_k s_l + 2u_k s_l + 2s_k u_l + 3u_k u_l) + u_j (s_k (2s_l + 3u_l) + 3u_k (s_l + 4u_l))] \varepsilon_{i,j,k,l}. \end{aligned} \quad (6.9)$$

Pour trouver une solution à l'équation vectorielle non-linéaire (6.7), la méthode de Newton convient très bien. Pour cela, il nous faut le jacobien Df de f , que voici :

$$\begin{aligned} \frac{\partial}{\partial u_m} f_i(u) = & \frac{1}{\tau} \alpha_{i,m} + \\ & \sum_{j=1}^N \left[\frac{1}{3} u_j (\beta_{i,m,j} + \beta_{i,j,m} + \gamma_{i,m,j} + \gamma_{i,j,m}) + \frac{1}{6} s_j (\beta_{i,m,j} - 2\beta_{i,j,m} + \gamma_{i,m,j} + \gamma_{i,j,m}) \right] + \\ & \sum_{j,k=1}^N \frac{\tau}{60} [2s_j s_k (\varepsilon_{i,j,m,k} + \varepsilon_{i,j,k,m} + \varepsilon_{i,m,j,k}) + \\ & 3s_j u_k (\varepsilon_{i,j,m,k} + \varepsilon_{i,j,k,m} + \varepsilon_{i,k,j,m} + \varepsilon_{i,m,j,k} + \varepsilon_{i,k,m,j} + \varepsilon_{i,m,k,j})]. \end{aligned} \quad (6.10)$$

L'algorithme de Newton consiste, pour chaque pas de temps n , à prendre un vecteur $x = s^n$ (qui est une approximation sensée de s^{n+1}) et répéter

$$\text{résoudre } r : [Df(x)] r = -f(x); \quad x \leftarrow x + r$$

jusqu'à ce que $\|r\|$ soit inférieur à un seuil ε choisi. On pose alors $s^{n+1} = x$.

Puisqu'on avance dans le temps au moyen des fonctions de base en temps affines, la résolution de (6.7) est bien la méthode STILS-TMQ1 développée pour l'équation de Burgers.

6.2 Essais numériques de STILS-TMQ1-Burgers

Pour les essais, nous avons programmé la méthode pour une base de fonctions pyramidales affines $\{\psi_1 \dots \psi_N\}$ où ψ_j est la fonction associée au point $x_j = j \cdot h$, $h > 0$. Le programme complet se trouve dans l'annexe C.5.

La première condition initiale à essayer est la fonction marche qui provoque les deux phénomènes fondamentaux d'une loi de conservation : l'onde de choc (provoquée par la marche descendante pour l'équation de Burgers) et l'onde de raréfaction (marche ascendante). On a pour cette condition initiale une solution explicite permettant la comparaison.

Nous comparons aussi STILS à la méthode classique de Godounov — ce qui donne aussi un repère dans le cas de conditions initiales non-triviales qui privent l'équation d'une solution entropique explicite. Le principe de cette méthode (lire aussi [41]) consiste à résoudre explicitement l'équation (6.1) sur $\tilde{\Omega} = \mathbb{R} \times]t_n; t_{n+1}[$ en prenant comme condition initiale la fonction $x \mapsto \tilde{u}(x, t_n)$ constante sur les intervalles $]x_j - \frac{h}{2}; x_j + \frac{h}{2}[$ et définie sur ces intervalles par $\tilde{u}(x_j, t_n) = u_j^n$, la valeur numérique calculée pour $t = t_n$. En intégrant (6.1) sur $\tilde{\Omega}_j =]x_j - \frac{h}{2}; x_j + \frac{h}{2}[\times]t_n; t_{n+1}[$, on obtient

$$\int_{x_j - \frac{h}{2}}^{x_j + \frac{h}{2}} \tilde{u}(x, t_{n+1}) dx - \int_{x_j - \frac{h}{2}}^{x_j + \frac{h}{2}} \underbrace{\tilde{u}(x, t_n)}_{=u_j^n} dx + \int_{t_n}^{t_{n+1}} \left[f(\tilde{u}(x_j + \frac{h}{2}, t)) - f(\tilde{u}(x_j - \frac{h}{2}, t)) \right] dt = 0 \quad (6.11)$$

On choisit comme solution numérique en $t = t_{n+1}$ la moyenne de $\tilde{u}$:

$$u_j^{n+1} := \frac{1}{h} \int_{x_j - \frac{h}{2}}^{x_j + \frac{h}{2}} \tilde{u}(x, t_{n+1}) dx,$$

d'où la relation

$$u_j^{n+1} = u_j^n - \frac{1}{h} \int_{t_n}^{t_{n+1}} \left[f(\tilde{u}(x_j + \frac{h}{2}, t)) - f(\tilde{u}(x_j - \frac{h}{2}, t)) \right] dt. \quad (6.12)$$

Comme la fonction $x \mapsto \tilde{u}(x, t_n)$ est en escalier, on a comme solution entropique unique une fonction $\tilde{u}(x, t)$ qui est une juxtaposition d'ondes de choc et de raréfaction (ces deux cas sont détaillés plus loin : §6.2.1 et §6.2.2). L'important ici est que si le pas de temps τ est assez petit (vérifiant $\max_{x \in \mathbb{R}} |u^0(x)| < h/\tau$), il n'y a pas de collision possible d'ondes sur $t \in]t_n; t_{n+1}[$. On a alors une propriété remarquable permettant de simplifier (6.12) : quel que soit le type d'onde apparaissant sur $\tilde{\Omega}_j$, la fonction $\tilde{u}(x, t)$ est **constante** sur les segments $\{x_j - \frac{h}{2}\} \times]t_n; t_{n+1}[$. Cette valeur ne dépend que des valeurs u_{j-1}^n et u_j^n de la solution numérique et s'obtient facilement en déterminant le type d'onde et la position du segment par rapport à l'onde. Notons-la $v(u_{j-1}^n, u_j^n)$. On a alors le schéma de Godounov

$$u_j^{n+1} = u_j^n - \frac{\tau}{h} [f(v(u_j^n, u_{j+1}^n)) - f(v(u_{j-1}^n, u_j^n))] . \quad (6.13)$$

6.2.1 Onde de choc

La figure 6.2 montre une comparaison entre la méthode de Godounov et STILS dans le cas d'une onde de choc. La condition initiale la plus simple pour provoquer une onde de choc est

$$u^0(x) = \begin{cases} u_G & x < 0 \\ u_D & x > 0 \end{cases} \quad (6.14)$$

avec $1 = u_G > u_D = 0$ (marche descendante).

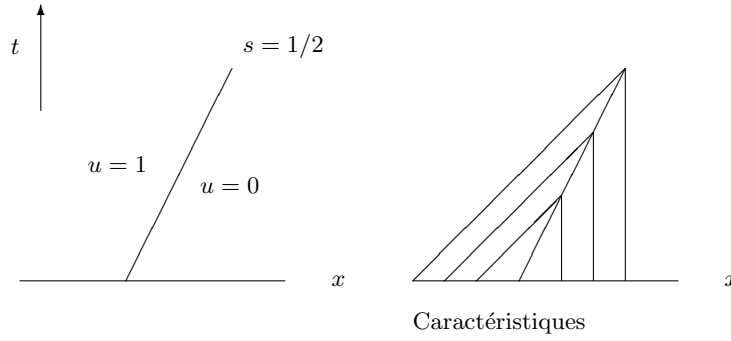


FIG. 6.1 – Solution d'onde de choc

L'unique solution faible, au sens des distributions, de (6.2) est dans ce cas (figure 6.1) le déplacement de cette marche à une vitesse constante σ :

$$u(x, t) = \begin{cases} u_G & x < \sigma t \\ u_D & x > \sigma t \end{cases}, \quad t > 0.$$

La vitesse σ de propagation du choc est, par la condition de Rankine-Hugoniot,

$$\sigma = \frac{f(u_D) - f(u_G)}{u_D - u_G} \quad (6.15)$$

qui est nécessaire et suffisante à ce que u soit solution faible. Ici donc, $\sigma = \frac{(u_D^2)/2 - (u_G^2)/2}{u_D - u_G} = 1/2$. Rappelons aussi que dans une situation d'onde de choc, la méthode de Godounov tient compte explicitement de s . D'autre part la qualité de cette méthode est comparable à Streamline diffusion + shock capturing, tout en diffusant davantage (cf comparaisons fig. 13.6 de [32]).

Or, malgré un choix soigneux des paramètres, STILS se montre plutôt instable dès lors que l'onde de choc a passé quelques mailles (cf figure 6.2). On a ici $h = 0,003$ et $\tau = 0,000025$; donc $C = 0,0078$. Pour C plus grand l'instabilité est encore plus forte.

6.2.2 Onde de raréfaction

Le pendant de l'onde de choc est l'onde de raréfaction, solution faible « naturelle » pour la même condition initiale en forme de marche (6.14), mais cette fois montante avec $0 = u_G < u_D = 1$.

$$u(x, t) = \begin{cases} u_G & x < f'(u_G)t \\ (f')^{-1}(x/t) & f'(u_G)t < x < f'(u_D)t \\ u_D & x > f'(u_D)t \end{cases}.$$

Pour l'équation de Burgers, l'onde de raréfaction s'écrit simplement

$$u(x, t) = \begin{cases} u_G & x < u_G t \\ (x/t) & u_G t < x < u_D t \\ u_D & x > u_D t \end{cases}. \quad (6.16)$$

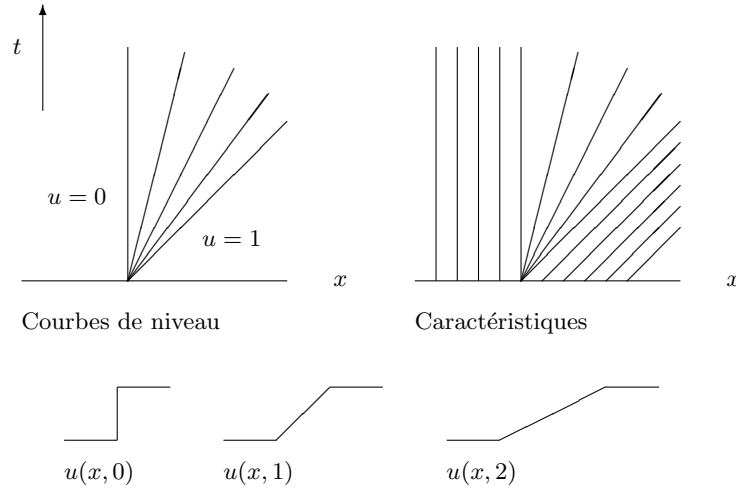


FIG. 6.3 – Solution d'onde de raréfaction

Les caractéristiques (figure 6.3) forment un éventail à partir du point $(x; t) = (0; 0)$. Comme il n'y a pas d'unicité de solution faible dans ce cas, on a défini des conditions d'entropie générales pour choisir la solution appropriée parmi les solutions faibles possibles. Cette question est étudiée depuis longtemps [38, 23, 41] ; une référence récente et solide est le livre de Godlewski et Raviart, *Numerical approximation of hyperbolic systems of conservation laws* [24] : §3.1 pour la définition d'entropie, §3.3 pour l'existence et l'unicité d'une solution entropique dans le cas scalaire. L'important ici est de savoir que l'onde de raréfaction (6.16) est la solution faible entropique et qu'il convient donc de la comparer à une solution numérique.

A nouveau, la méthode de Godounov tire parti de l'onde de raréfaction de manière explicite, d'où un résultat conforme, à laquelle néanmoins s'ajoute une diffusion numérique. Comme on le voit sur la figure 6.4, l'instabilité de STILS se manifeste également avec une certaine vigueur ; on observe la formation d'oscillations sur la zone de raréfaction tandis que la méthode de Godounov ne présente que de la diffusion. Rappelons-le, la solution entropique exacte a une pente $1/t$ pour $x \in [0; t]$.

6.2.3 Recours à la linéarisation

La méthode STILS, pour cette EDP non linéaire, présente de fortes instabilités, essentiellement sur l'onde de choc. Le fait que les fonctions de base soient continues y contribue peut-être, bien que pour l'équation de Burgers les impuretés disparaissent *naturellement* dans l'onde de choc. Il se pourrait aussi que la fonctionnelle J ne soit pas convexe, ce qui conduirait l'algorithme de Newton à des extrema *locaux* de J .

Un moyen d'améliorer la situation peut être de linéariser le problème ; pour DJ , on a, au lieu de (6.4),

$$[DJ(u)]w \simeq \int_{\tilde{\Omega}} \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \right) \left(\frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} \right) dx dt. \quad (6.17)$$

Le démarche consiste à prendre u comme vitesse de convection « donnée », ce qui produit la formulation STILS usuelle, quitte à considérer en fait u comme inconnue.

On peut d'autre part faire une semi-linéarisation en gardant l'équation faible STILS-Burgers (6.4), mais dans une approche marche-en-temps où u^* est l'état connu de la solution

pour préparer le pas de temps suivant. Cela permet de rendre linéaire l'équation faible

$$[DJ(u)]w \simeq \int_{\tilde{\Omega}} \left(\frac{\partial u}{\partial t} + u^* \frac{\partial u}{\partial x} \right) \left(\frac{\partial w}{\partial t} + w \frac{\partial u^*}{\partial x} + u^* \frac{\partial w}{\partial x} \right) dx dt. \quad (6.18)$$

On peut aussi imaginer une linéarisation totale

$$[DJ(u)]w \simeq \int_{\tilde{\Omega}} \left(\frac{\partial u}{\partial t} + u^* \frac{\partial u}{\partial x} \right) \left(\frac{\partial w}{\partial t} + u^* \frac{\partial w}{\partial x} \right) dx dt. \quad (6.19)$$

Cette approche « simpliste » est souvent employée pour essayer toutes sortes de méthodes sur l'équation de Burgers (parmi d'autres, [37, 7]) au moment de discrétiser en temps, en général sans discuter de l'effet du retard de u^* par rapport à u . Il s'avère en tout cas que c'est une bonne piste pour STILS. La semi-linéarisation (6.18) améliore cela sans tomber dans l'instabilité de la version complètement non-linéaire vue précédemment. On le verra au §6.2.4.

Commençons l'expérimentation par la linéarisation totale (6.19). L'avantage est qu'on peut essayer sans autres toutes les méthodes vues pour les EDP linéaires, en prenant comme champ v , unidimensionnel ici, la valeur u^* .

Pour l'onde de choc (figure 6.5) on observe un certain retard avec les méthodes autres que Godounov, mais cela s'explique par le fait que la vitesse utilisée pour l'intervalle $[t_k; t_{k+1}]$ est égale à la valeur de la solution approchée u^* en $t = t_k$. Une correction de type prédiction-correction et/ou un emprunt au déterminisme de la méthode de Godounov² permettrait sans doute de corriger ce phénomène.

Pour l'onde de raréfaction (figure 6.6), STILS-TMQ1 confine à la perfection. Les degrés supérieurs en font trop, peut-on dire (légères oscillations au-dessus de la valeur 1). On remarque particulièrement la diffusion du schéma de Godounov en $t = 19/21 < 1$.

Un aspect fondamental est aussi la **formation d'ondes de choc**. Par un raisonnement simple sur la pente des caractéristiques, on voit que même si la condition initiale u^0 n'est pas discontinue, alors la « tranche au temps t » $u^t : x \mapsto u(x, t)$ peut l'être pour un certain $t > 0$. En fait, en prenant $u^0 \in C^1$ par morceaux, il est clair que si u^0 est croissante, alors u^t l'est aussi pour tout $t > 0$ (il suffit de suivre les caractéristiques, dont le faisceau se disperse); sinon, deux caractéristiques se rencontrent en un $t > 0$. On peut montrer ([41] p. 25) que le temps auquel apparaît la première discontinuité est

$$t = \frac{-1}{\inf_{x \in \mathbb{R}} u^{0'}(x)}.$$

²ou une de ses variantes comme Godounov-Perthame [62]

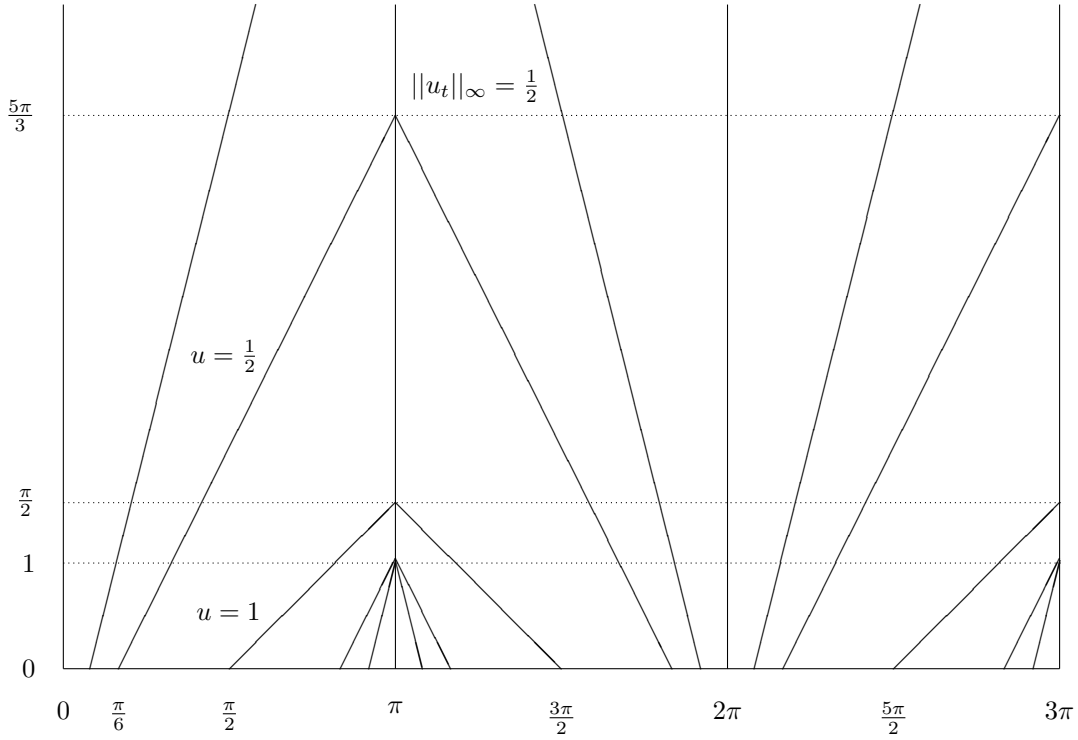


FIG. 6.7 – Caractéristiques de l'équation de Burgers pour la CI $u^0 = \sin$. Temps intéressants : $t = 1$, $t = \pi/2$ (rupture de la vague) et $t = 5\pi/3$

Prenons $u^0(x) = \sin(x)$. Des ondes de choc doivent se former exactement aux points $(x, t) = (\pi + 2k\pi, 1)$, $k \in \mathbb{Z}$, avec une pente 0 (moyenne des valeurs transportées par les caractéristiques).

Décrivons (figure 6.7) la solution entropique pour la CI $u^0 = \sin$, ceci permettra de bien contrôler les comportements numériques.

Avant $t = 1$, il n'y a pas de choc : u^t est continue. Puis, pour $t > 1$, il y a propagation du choc sur les demi-droites $\{\pi + 2k\pi\} \times [1; \infty[$. Le moment où une caractéristique issue de $x \in]0; 2\pi[$ entre dans le choc en $x = \pi$ est $t = \frac{\pi-x}{\sin x}$. A partir de $t = \pi/2$, u^t est croissant sur les intervalles $]2k\pi; 2(k+1)\pi[$, on a donc des zones de raréfaction séparées par des ondes de choc. $\|u^t\|_\infty$ décroît pour $t \geq \pi/2$. Par exemple, $\|u^{5\pi/3}\|_\infty = 1/2$.

En comparant les solutions numériques — figure 6.8 — on voit que STILS linéarisé approche la solution entropique avec quelques oscillations autour du choc, qui augmentent avec le degré de la base en temps. A part cela, le comportement pour $t = 1$, $t = \pi/2$ et $t = 5\pi/3$ est parfaitement ce qu'on attend.

6.2.4 Semi-linéarisation

Nous avons testé la variante semi-linéarisée (6.18) en faisant une approximation Q_1 en temps. Après un calcul sur les mêmes fonctions de bases en temps que pour la convection (1.9), on obtient un système similaire à (1.14), mais avec deux termes en plus pour les matrices

A et B (1.16). Précisément, on a le système

$$Au^{k+1} = Bu^k$$

où, pour $u^*(x)$ fonction interpolée sur Ω à l'aide des valeurs connues u^k , on pose

$$\begin{aligned} a_{ij} &= \int_{\Omega} \left[\frac{\tau}{3} (\psi'_j u^*) (\psi'_i u^*) + \frac{1}{2} (\psi'_j u^*) \psi_i + \frac{1}{2} \psi_j (\psi'_i u^*) + \frac{1}{\tau} \psi_j \psi_i \right. \\ &\quad \left. + \frac{1}{2} \partial_x u^* \psi_j \psi_i + \frac{\tau}{3} u^* \cdot \partial_x u^* \psi'_j \psi_i \right] dx \\ b_{ij} &= \int_{\Omega} \left[\frac{-\tau}{6} (\psi'_j u^*) (\psi'_i u^*) - \frac{1}{2} (\psi'_j u^*) \psi_i + \frac{1}{2} \psi_j (\psi'_i u^*) + \frac{1}{\tau} \psi_j \psi_i \right. \\ &\quad \left. + \frac{1}{2} \partial_x u^* \psi_j \psi_i + \frac{-\tau}{6} u^* \cdot \partial_x u^* \psi'_j \psi_i \right] dx \end{aligned} \quad (6.20)$$

Le résultat surprenant est pour l'onde de choc — figure 6.9 — puisqu'on gomme, par rapport à la linéarisation totale, l'oscillation des dépassements autour du choc, qui se développe lentement au cours du temps. Au temps final $t = 1$, le dépassement est tiré vers le haut pour la solution totalement linéarisée, alors que la semi-linéarisation maintient les oscillations stables et assez modestes : 0,096 au maximum de dépassement par le haut et 0,085 au maximum par le bas. Il reste cependant un certain retard sur la solution entropique, dû au fait qu'à chaque pas de temps t_k , u^* est utilisée pour simuler la solution sur l'intervalle $]t_k; t_{k+1}]$. Pour l'onde de raréfaction, — figure 6.10 — le résultat est impeccable et les termes supplémentaires de semi-linéarisation, par rapport à la variante totalement linéarisée, ne provoquent pas d'oscillations comme on pourrait le craindre dans un contexte complètement non-linéaire (figure 6.4).



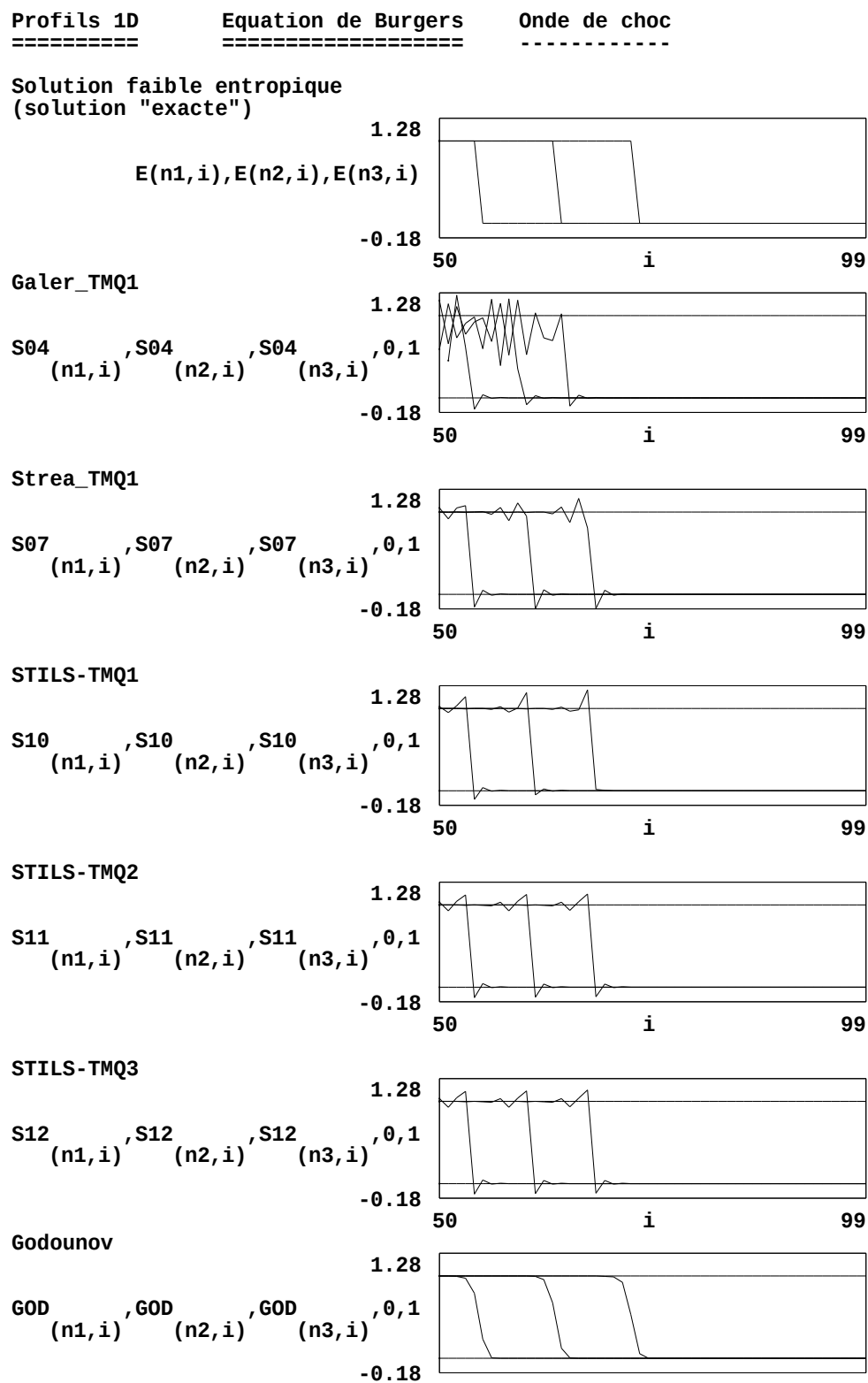


FIG. 6.5 – Equation de Burgers linéarisée, onde de choc, diverses méthodes , pas de temps $t = 5/21$ (n1), $t = 12/21$ (n2), $t = 19/21$ (n3)

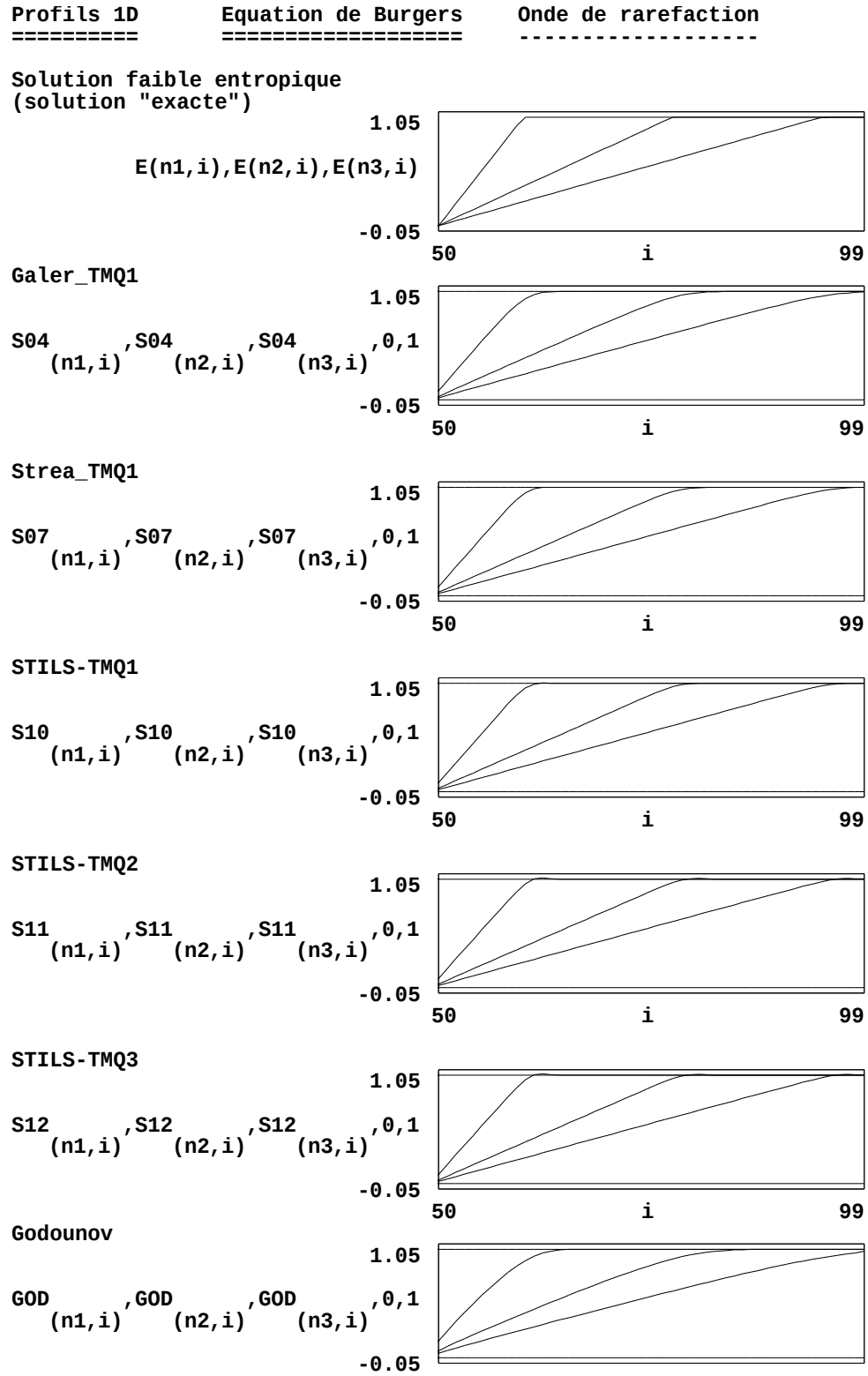


FIG. 6.6 – Equation de Burgers linéarisée, onde de raréfaction, diverses méthodes , pas de temps $t = 5/21$ (n1), $t = 12/21$ (n2), $t = 19/21$ (n3)

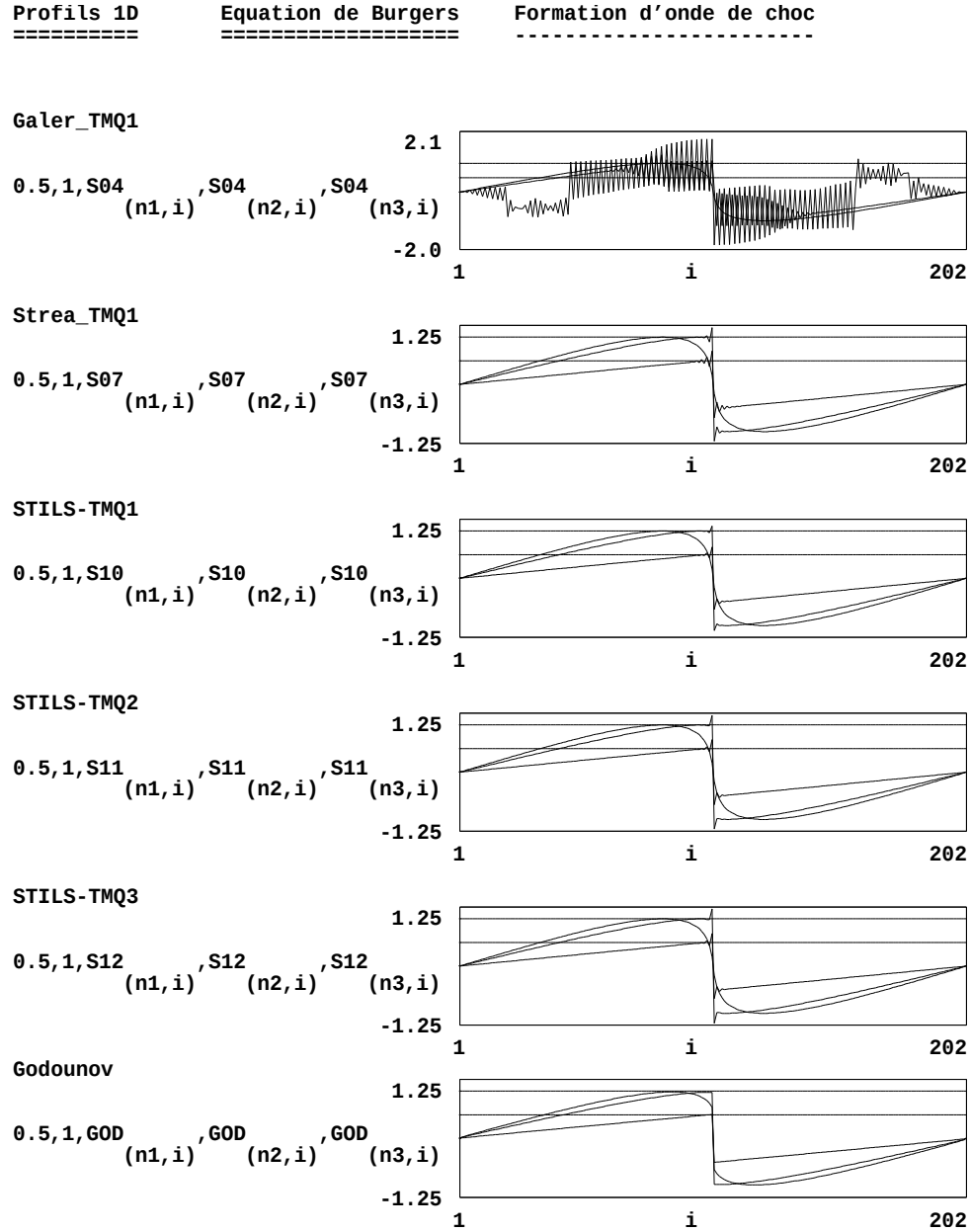


FIG. 6.8 – Equation de Burgers linéarisée, $u^0 = \sin$; simulation de u^t par diverses méthodes et pour $t = 1$ (forme arrondie), $t = \pi/2$ (rupture de la vague) et $t = 5\pi/3$ (dent de scie à hauteur 1/2)

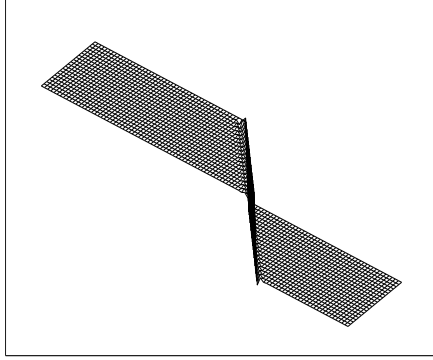
Linearisations de STILS-TMQ1 sur l'équation de Burgers - onde de choc

```

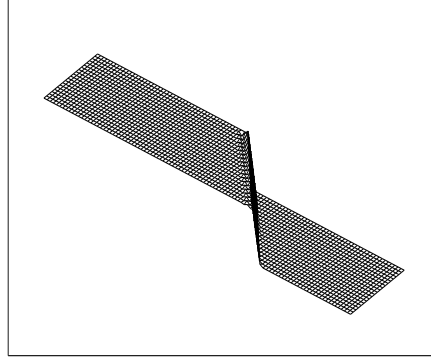
L := READPRN(stils_burg_lin)    rows(L) = 21    min(L) = -0.1272
                                cols(L) = 99    max(L) = 1.237
N := READPRN(stils_burg_nli)   rows(N) = 21    min(N) = -0.0848
                                cols(N) = 99    max(N) = 1.096

```

N: semi-linearise



L: linearise



```

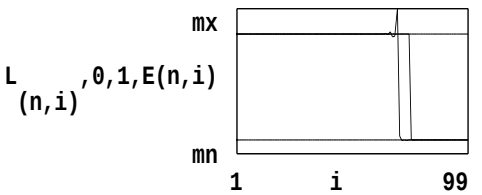
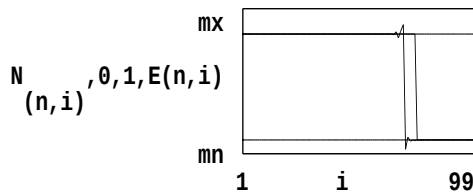
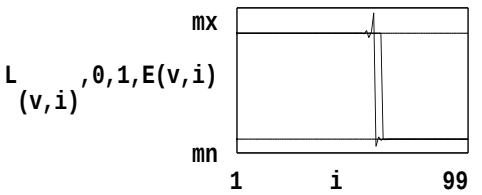
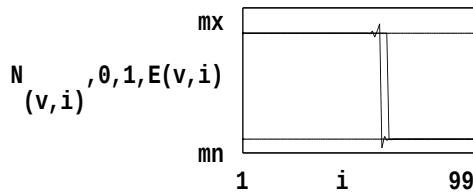
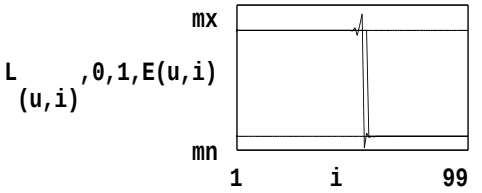
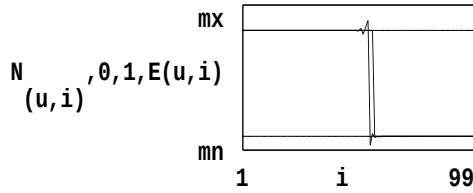
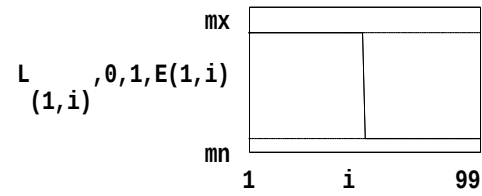
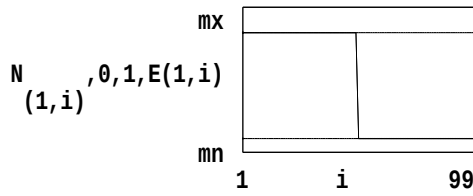
N
      i := 1 .. cols(L)
n := rows(L)      u := ceil(n·0.25)      v := ceil(n·0.5)
t=1 :      n = 21      t=0.2 :      u = 6      t=0.5 :      v = 11

```

```

mins := (min(N) min(L))    mn := min(mins)    mn = -0.1272
maxs := (max(N) max(L))    mx := max(maxs)    mx = 1.237

```



N: STILS-TMQ1 semi-linearise

L: STILS-TMQ1 linearise

FIG. 6.9 – Equation de Burgers linéarisée et semi-linéarisée pour STILS-TMQ1 ; onde de choc, solution entropique (E)

Linearisations de STILS-TMQ1 sur l'équation de Burgers - onde de rarefaction

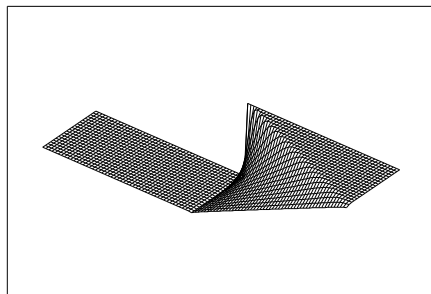
```

L := READPRN(stils_burg_lin)    rows(L) = 21    min(L) = -0.0206
                                cols(L) = 99    max(L) = 1.011

N := READPRN(stils_burg_nli)    rows(N) = 21    min(N) = -0.0107
                                cols(N) = 99    max(N) = 1.011

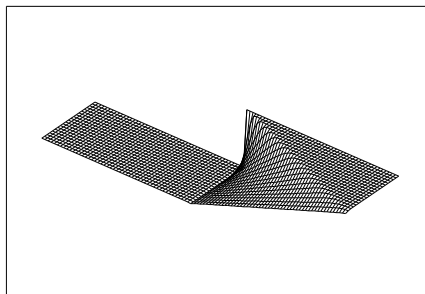
```

N: semi-linearise



N

L: linearise



L

```

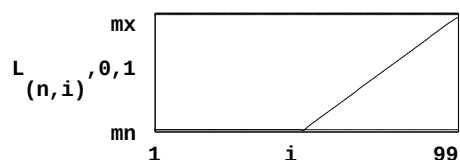
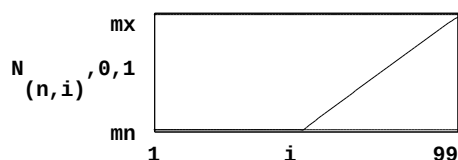
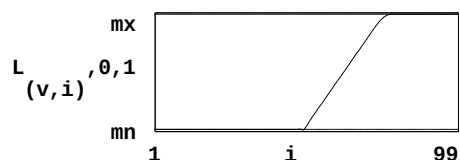
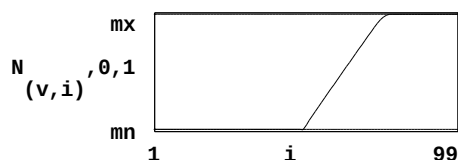
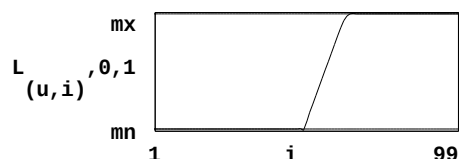
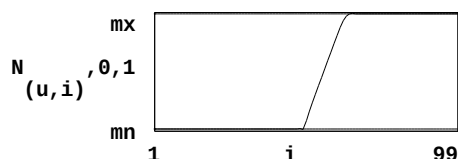
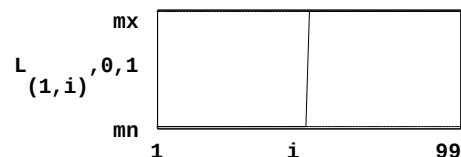
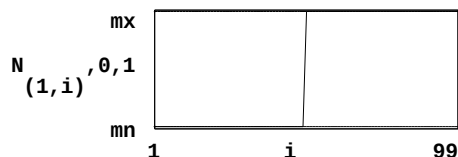
i := 1 .. cols(L)    n := rows(L)    u := ceil(n*0.25)    u = 6
v := ceil(n*0.5)    v = 11

```

```

mins := (min(N) min(L))    mn := min(mins)    mn = -0.0206
maxs := (max(N) max(L))    mx := max(maxs)    mx = 1.011

```



STILS semi-linearise

STILS linearise

FIG. 6.10 – Equation de Burgers linéarisée et semi-linéarisée pour STILS-TMQ1; onde de raréfaction

Chapitre 7

Modèle de fermeture pour la loi de Darcy

7.1 Énoncé du problème

L'écoulement en milieu poreux est régi par la loi de Darcy qui permet de prédire la répartition d'une charge hydraulique h (c'est-à-dire, à facteur près, une pression) à l'aide d'un tenseur de perméabilité K et de conditions aux limites. Combinée avec l'équation de continuité, on obtient l'équation instationnaire

$$\operatorname{div}(K \nabla h) = \frac{\partial h}{\partial t}$$

dans sa forme la plus simple — c'est l'équation (1) de l'introduction.

Le tenseur K est une donnée de l'équation, mais on peut avoir à le déterminer. En particulier, si l'on étudie un milieu homogène, les particularités liées aux limites (d'un domaine borné) disparaissent au profit de conditions périodiques. La perméabilité et la pression deviennent ensemble des inconnues ; la disposition d'obstacles intérieurs et répétés périodiquement permet de les déterminer, ainsi que montre S. Whitaker dans l'article [63], que poursuit [5].

En termes mathématiques, on se placera dans l'espace $\mathbb{R}^n$, $n = 2$ ou 3 , et fixera n directions $l_1 \dots l_n$ indépendantes (vecteurs de $\mathbb{R}^n$) suivant lesquelles on aura périodicité. Notons la période de base

$$P = \left\{ \sum_{i=1}^n \alpha_i l_i, \alpha_1 \dots \alpha_n \in]0; 1[\right\}.$$

Pour séparer la phase liquide où a lieu l'écoulement de la phase solide — les obstacles, qui sont inertes — nous définirons les équations dans un ouvert connexe $\tilde{\Omega}$ de $\mathbb{R}^n$ et P -périodique. La partie $\tilde{\Omega}$ représente ainsi la phase liquide du milieu et son complémentaire $\mathbb{R}^n \setminus \tilde{\Omega}$, la phase solide.

Cela s'exprime par la condition qu'un point x de $\mathbb{R}^n$ est dans le domaine $\tilde{\Omega}$ si et seulement si $x + l_i$ s'y trouve pour n'importe quel i . Le bord $\tilde{\Gamma} = \partial \tilde{\Omega}$, sera supposé C^1 par morceaux et $\tilde{\Gamma} \cap \bar{P}$ sera supposé de mesure non nulle. Ainsi, l'interface liquide-solide $\tilde{\Gamma}$ entoure au moins un obstacle plein répété périodiquement.

Une formulation simple du problème de détermination se trouve dans [5], relations (2.21) à (2.24). Il s'agit de trouver $d : \tilde{\Omega} \rightarrow \mathbb{R}^n$ et $D : \tilde{\Omega} \rightarrow M_n(\mathbb{R})$ suffisamment dérivables tels

que

$$\begin{cases} -\nabla d + \Delta D = I & \text{dans } \tilde{\Omega} \\ \operatorname{div} D = 0 & \text{dans } \tilde{\Omega} \\ D = 0 & \text{sur } \tilde{\Gamma} \\ D \text{ et } d & P\text{-périodiques} \end{cases} \quad (7.1)$$

Le tenseur D représente la déviation en vitesse et d la déviation en pression. Nous indiquons au §7.2.1 comment obtenir le tenseur de perméabilité K à l'aide du tenseur D . Contrairement à la loi de Darcy mentionnée au début du chapitre, le problème (7.1) est *stationnaire*. Ce n'est pas ennuyeux dans la mesure où la perméabilité K , qui est une *donnée* pour la loi de Darcy, n'évolue en général pas dans l'intervalle de temps $[0, T]$ d'une simulation. Observons enfin que le problème est une forme tensorielle de l'équation de Stokes.

7.2 Point de vue local

Ramenons ce problème à un problème local situé dans la cellule P , en posant $\Omega = \tilde{\Omega} \cap P$ et $\Gamma = \tilde{\Gamma} \cap \partial\Omega$.

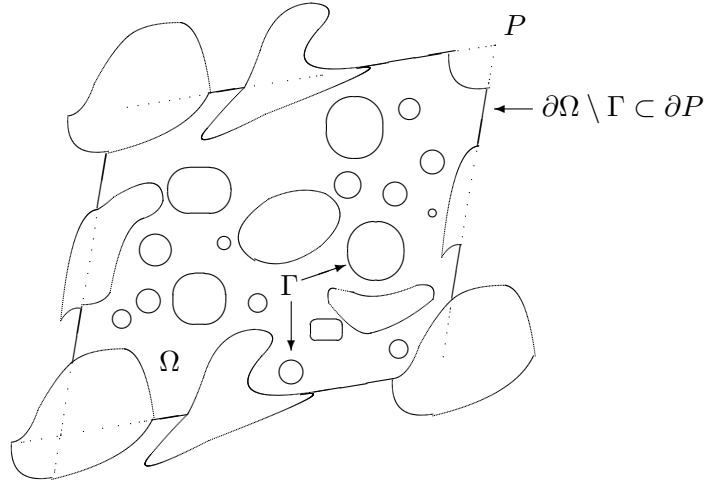


FIG. 7.1 – Cellule P , domaine localisé Ω

Après l'hypothèse de périodicité qui permet d'aborder un problème sur un domaine vaste mais homogène, c'est une seconde étape évidente puisque toute solution dans $\tilde{\Omega}$ est déterminée par ce qui se passe dans P . De plus, on pourra tirer parti du fait qu' Ω est borné. Enfin on remarquera que flux périodique équivaut au sens intégral à flux nul sur le bord de P , ce qui permettra de d'attaquer le problème avec des conditions aux bords classiques (Neumann périodique et Dirichlet homogène).

Il est clair que les points du bord de Ω appartiennent à celui de P (frontière « artificielle » produite par la localisation) ou à l'interface locale Γ (frontière « naturelle »), de manière non exclusive.

Posons le problème localisé. On cherche $d : \bar{\Omega} \rightarrow \mathbb{R}^n$ et un tenseur $D : \bar{\Omega} \rightarrow M_n(\mathbb{R})$

suffisamment dérivables tels que

$$\begin{cases} -\nabla d + \Delta D = I & \text{dans } \Omega \\ \operatorname{div} D = 0 & \text{dans } \Omega \\ D = 0 & \text{sur } \Gamma \\ D \text{ et } d & P\text{-périodiques} \end{cases} \quad (7.2)$$

7.2.1 Obtention de la perméabilité K

Précisons maintenant comment on obtient le tenseur de perméabilité intrinsèque K à partir du tenseur D solution du problème localisé (7.2). Il suffit de reprendre les formulations de [63] et [5]. Définissons la *moyenne de phase intrinsèque* (c'est (2.7) dans [63])

$$\langle D \rangle^{\text{liq}} = \frac{1}{\operatorname{Vol}(\Omega)} \int_{\Omega} D \, dx \quad (7.3)$$

et notons la *fraction de volume de la phase liquide* ((2.9) dans [63])

$$\varepsilon_{\text{liq}} = \frac{\operatorname{Vol}(\Omega)}{\operatorname{Vol}(P)}. \quad (7.4)$$

Alors on a

$$K = -\varepsilon_{\text{liq}} \langle D \rangle^{\text{liq}} \quad (7.5)$$

(formules (2.20) et (A.24) dans [5]).

7.2.2 Effets de la périodicité

Il est évident que si un champ de vitesse est P -périodique, les flux se compensent au travers de la frontière « artificielle », $\partial\Omega \setminus \Gamma$.

Lemme 6 : *Soit un champ de vecteurs $a : \partial P \rightarrow \mathbb{R}^n$ intégrable et P -périodique.*

$$\text{Alors } \int_{\partial\Omega \setminus \Gamma} (a|n) \, d\gamma = 0.$$

Preuve : Chaque point de $\partial\Omega \setminus \Gamma$ est sur une face de P , donc a a au moins une composante 0 ou 1 dans la base $l_1 \dots l_n$. Ce point a un correspondant sur la face opposée avec une normale sortante n de signe opposé. Comme a est périodique, les intégrales prises sur les faces opposées se compensent deux à deux. $\square$

Intuitivement, le flux sortant d'un côté rentre par le côté opposé, ce qui fait au total une compensation ; au sens intégral, cela coïncide avec une condition de Neumann homogène.

En résumé, sur le bord de Ω , on a une condition de type Neumann périodique sur la frontière « artificielle », de contribution totale nulle, et une condition de type Dirichlet sur l'interface Γ (la frontière « naturelle »). Le lemme 6 ne nous servira qu'à passer du problème fort (7.5) à un problème faible où D sera défini dans un espace de Sobolev d'ordre 1 approprié.

7.2.3 Problème local : formulation faible

Définissons à présent des espaces fonctionnels appropriés, en vue d'établir un problème faible et de le résoudre.

- Définissons l'équivalent périodique et localisé des fonctions lisses à support compact [51]. Soit d'abord l'espace fonctionnel $\widetilde{\mathcal{D}}_P$ des fonctions v de $\mathcal{D}(\tilde{\Omega})$ où, pour tout n -uplet d'indices α de $\mathbb{N}^n$, les valeurs de $\partial^\alpha v$ sont égales sur les faces opposées de $\partial P \cap \partial\Omega$ (voir la figure 7.1). On a la périodicité. On peut maintenant poser l'espace de fonctions localisées

$$\mathcal{D}_P = \left\{ u = v \Big|_{\Omega} : v \in \widetilde{\mathcal{D}}_P \right\}.$$

Le fait de demander l'égalité sur les faces opposées pour toutes les dérivées de v permet d'obtenir, en prolongeant une fonction $u \in \mathcal{D}_P$ par périodicité sur $\tilde{\Omega}$, une fonction lisse, périodique, à support dans $\tilde{\Omega}$ (en particulier, elle s'annule avec toutes ses dérivées sur $\tilde{\Gamma}$).

- Posons $H_{P,\Gamma}$ l'adhérence dans $H^1(\Omega)$ de $\mathcal{D}_P$. Définissons sur $H_{P,\Gamma}$ la forme bilinéaire symétrique

$$((u, v)) = (\nabla u, \nabla v) = \sum_{l=1}^n \left(\frac{\partial u}{\partial x_l}, \frac{\partial v}{\partial x_l} \right) = \sum_{l=1}^n \int_{\Omega} \frac{\partial u}{\partial x_l} \frac{\partial v}{\partial x_l} dx$$

où $(\cdot, \cdot)$ désigne le produit scalaire sur $L^2(\Omega)$ ou sur $L^2(\Omega)^n$.

- Soit $V_{P,\Gamma} = M_n(H_{P,\Gamma})$ l'espace des matrices $n \cdot n$ à composantes dans $H_{P,\Gamma}$; soit la forme bilinéaire $V_{P,\Gamma} \times V_{P,\Gamma} \rightarrow \mathbb{R}$ induite par celle de $H_{P,\Gamma}$

$$((D, E)) = \sum_{i,j=1}^n ((D_{ij}, E_{ij})) = \sum_{i,j,l=1}^n \int_{\Omega} \frac{\partial D_{ij}}{\partial x_l} \frac{\partial E_{ij}}{\partial x_l} dx$$

- Enfin l'ensemble des tenseurs de $V_{P,\Gamma}$ à divergence nulle

$$V_{P,\Gamma,\text{div}} = \{D \in V_{P,\Gamma} : \text{div} D = 0\}$$

Si on cherche un tenseur D et un vecteur d solution de l'équation (7.5), on est amené à écrire cette dernière au sens faible :

$$\int_{\Omega} (\Delta D | B) dx - \int_{\Omega} (\nabla d | B) dx - \int_{\Omega} (I | B) dx = 0 \quad \text{pour tout } B \in M_n(\mathcal{D}_P)$$

avec B tenseur-test lisse. On suppose pour l'instant que D et d sont lisses. Les deux premières intégrales peuvent s'exprimer au sens des distributions :

$$\langle \Delta D, B \rangle - \langle \nabla d, B \rangle - (I, B) = 0 \tag{7.6}$$

On peut développer les termes, en invoquant le théorème de Green :

$$\begin{aligned} \langle \Delta D, B \rangle &= \sum_{i,j=1}^n \langle \Delta D_{ij}, B_{ij} \rangle = \sum_{i,j=1}^n \int_{\partial\Omega} \frac{\partial D_{ij}}{\partial n} B_{ij} d\gamma - ((D, B)) \\ \langle \nabla d, B \rangle &= \sum_{i,j=1}^n \langle \partial_j d_i, B_{ij} \rangle = \sum_{i,j=1}^n \int_{\partial\Omega} d_i B_{ij} n_j d\gamma - (d, \text{div} B) \\ (I, B) &= \sum_{i=1}^n \int_{\Omega} B_{ii} dx \end{aligned}$$

Les intégrales sur le bord de Ω s'annulent grâce au lemme 6 pour la partie artificielle (bord de P) et au support de B pour la partie naturelle. Ainsi $\langle \Delta D, B \rangle = -((D, B))$ et $\langle \nabla d, B \rangle = -(d, \operatorname{div} B)$. L'équation (7.6) s'écrit alors :

$$-((D, B)) + (d, \operatorname{div} B) - (I, B) = 0$$

avec des intégrales définies dans $L^2(\Omega)$. Les inconnues sont $D \in V_{P,\Gamma,\operatorname{div}}$ et $d \in L^2(\Omega)^n$. Les espaces fonctionnels pour D et d correspondent aux conditions minimales de validité des produits scalaires. L'équation doit être vérifiée pour tout $B \in M_n(\mathcal{D}_P)$. En remplaçant $M_n(\mathcal{D}_P)$ par $V_{P,\Gamma}$ qui le contient, on obtient le problème local faible de trouver $D \in V_{P,\Gamma,\operatorname{div}}$ et $d \in L^2(\Omega)^n$ tels que

$$((D, B)) - (d, \operatorname{div} B) + (I, B) = 0 \quad \text{pour tout } B \in V_{P,\Gamma} \quad (7.7)$$

7.2.4 Propriétés des espaces $H_{P,\Gamma}$, $V_{P,\Gamma}$, $V_{P,\Gamma,\operatorname{div}}$

Considérons la norme $\|u\|_{1,\Omega} = \sqrt{(u, u) + ((u, u))}$ et la semi-norme $|u|_{1,\Omega} = \sqrt{((u, u))}$.

Lemme 7 *La norme $\|\cdot\|_{1,\Omega}$ et la semi-norme $|\cdot|_{1,\Omega}$ sont équivalentes — et donc $|\cdot|_{1,\Omega}$ est une norme.*

Par conséquent, $H_{P,\Gamma}$ muni de la norme $|\cdot|_{1,\Omega}$ est un espace de Hilbert, ainsi que $V_{P,\Gamma}$ et $V_{P,\Gamma,\operatorname{div}}$.

Preuve : Comme Γ est de mesure non nulle dans $\partial\Omega$, on a une inégalité de Friedrichs-Poincaré $\|u\|_{1,\Omega} \leq c(\Omega)|u|_{1,\Omega}$. Comme d'autre part $|u|_{1,\Omega} \leq \|u\|_{1,\Omega}$, on a l'équivalence. $\square$

7.2.5 Problème variationnel sans ∇d

Si l'on exprime le problème local faible (7.7) en ne prenant que les B à divergence nulle, le terme $-(d, \operatorname{div} B)$ disparaît et on obtient le problème de trouver $D \in V_{P,\Gamma,\operatorname{div}}$ tel que

$$((D, B)) + (I, B) = 0 \quad \text{pour tout } B \in V_{P,\Gamma,\operatorname{div}}. \quad (7.8)$$

7.3 Résultats d'existence et d'unicité pour D et d

Pour se retrouver dans les résultats établis dans la suite, plaçons-les dans le tableau suivant en mettant en évidence le plus intéressant :

	B à divergence nulle	Terme d apparent
Condition nulle sur ∂P	Théorème 9	Théorème 11
Condition périodique sur ∂P	Théorème 8	Théorème 22

Théorème 8 *Le problème (7.8) admet une unique solution $D \in V_{P,\Gamma,\operatorname{div}}$.*

Preuve : La forme $((\cdot, \cdot))$ est, par le lemme 7, un produit scalaire, évidemment continu dans $V_{P,\Gamma,\operatorname{div}}$ et coercitif. La forme linéaire $B \mapsto l(B) = -(I, B)$ est continue dans $V_{P,\Gamma,\operatorname{div}}$:

$$|(I, B)| \leq \|I\|_{0,\Omega} \cdot \|B\|_{0,\Omega} \leq \|I\|_{0,\Omega} \cdot c(\Omega) \cdot |B|_{1,\Omega} \quad \text{pour tout } B \in V_{P,\Gamma,\operatorname{div}}. \quad (7.9)$$

Donc, par Lax-Milgram, il existe un unique D solution de (7.8).

□

Remarque 1 : Ce résultat suffit à déterminer le tenseur de perméabilité intrinsèque K . En effet, seul D est utilisé pour calculer K ; d'autre part on obtient le même D si on traite le cas plus général avec $\nabla d \neq 0$ qui est étudié ci-après.

Remarque 2 : La technique de la preuve est la même que pour résoudre l'équation de Stokes, en prenant des fonctions-test à divergence nulle (théorème I.2.1 de [58]).

7.3.1 Existence et unicité de d définie sur Ω , cas purement local

On fait ici l'équivalent de l'obtention de la pression dans le problème de Stokes. L'idée est d'ailleurs de s'y ramener. La difficulté est qu'on sait le traiter normalement pour une trace nulle sur tout le bord $\partial\Omega$ ou une partie non négligeable de celui-ci. C'est en fait la situation d'un problème de Dirichlet homogène dans un domaine borné Ω .

Nous résolvons le problème tensoriel dans ce cas particulier, plus simple, puis le cas général, périodique. Il est intéressant de faire ainsi car non seulement le principe de la démonstration y apparaît complètement, mais aussi il permet déjà de trouver un K satisfaisant, car, d'après [5] (rem. (2) p. 216), les conditions périodiques ne devraient avoir qu'une influence *mineure* sur D dans Ω .

Soit $V_0 = M_n(H_0^1(\Omega))$ l'espace des matrices $n \cdot n$ à composantes dans $H_0^1(\Omega)$. C'est bien sûr un sous-ensemble de $V_{P,\Gamma}$: les fonctions y ont une trace nulle non seulement sur Γ mais aussi sur ∂P . Nous munissons donc V_0 du produit scalaire de $V_{P,\Gamma}$, ce qui en fait un sous-espace de Hilbert. Comme sous-espace de V_0 nous avons l'espace $V_{0,\text{div}}$ des tenseurs à divergence nulle.

Le problème faible purement local qui fait pendant au problème (7.8) est de trouver $D \in V_{0,\text{div}}$ tel que

$$((D, B)) + (I, B) = 0 \quad \text{pour tout } B \in V_{0,\text{div}}. \quad (7.10)$$

On a immédiatement l'existence et l'unicité d'une solution : c'est l'équivalent du théorème 8 dont il suffit de recopier la preuve en remplaçant $V_{P,\Gamma,\text{div}}$ par $V_{0,\text{div}}$.

Théorème 9 *Le problème (7.10) admet une unique solution $D \in V_{0,\text{div}}$.*

Il faut encore rappeler le résultat fondamental d'existence du potentiel pression, que nous utiliserons pour le problème tensoriel.

Lemme 10 *Soit U ouvert de $\mathbb{R}^n$ et une forme $f \in H^{-1}(U)^n$ telle que $(f, v) = 0$ pour tout $v \in \mathcal{D}(U)^n$ tel que $\text{div } v = 0$.*

Alors il existe $p \in L_{\text{loc}}^2(U)$ telle que $f = \nabla p$ au sens des distributions : $\langle \nabla p, v \rangle = -(p, \text{div } v)$ pour tout $v \in H_0^1(U)^n$.

Si de plus U est borné et à bord lipschitzien, alors $p \in L^2(U)$.

Cette « récupération de la pression » est un résultat classique dans le contexte des équations de Stokes et Navier-Stokes, basé sur un théorème de G. de Rham [19]. L'énoncé correspond à la remarque I.1.4 du livre de Temam, *Navier-Stokes equations* [58], qui renvoie pour les éléments de preuve à [20, 43, 44]. On a l'équivalent dans [42, 52, 57].

Maintenant, nous pouvons énoncer le théorème d'obtention de la déviation en pression d .

Théorème 11 *Il existe un unique $D \in V_{0,\text{div}}$ et un $d \in L^2(\Omega)^n$ unique à vecteur constant près tels que*

$$((D, B)) + (I, B) - (d, \text{div } B) = 0 \quad \text{pour tout } B \in V_0. \quad (7.11)$$

Preuve : Considérons le tenseur D donné par le théorème 9.

Définissons la forme linéaire $L(B) = ((D, B)) + (I, B)$, $B \in V_0$. Elle est bornée puisque :

$$|L(B)| \leq |D|_{1,\Omega} \cdot |B|_{1,\Omega} + \|I\|_{0,\Omega} \cdot \|B\|_{0,\Omega} \leq (|D|_{1,\Omega} + \|I\|_{0,\Omega} \cdot c(\Omega)) \cdot |B|_{1,\Omega} \quad (7.12)$$

Par le théorème 8, $L(B) = 0$ pour tout $B \in V_{0,\text{div}}$. Il s'agit de trouver $d \in L^2(\Omega)^n$ tel que

$$L(B) = (d, \text{div} B) \text{ pour tout } B \in V_0. \quad (7.13)$$

Or, on a

$$L(B) = \sum_{i=1}^n \left(\sum_{j,k=1}^n (\partial_k D_{ij}, \partial_k B_{ij}) - \int_{\Omega} B_{ii} dx \right). \quad (7.14)$$

Posons $f_i(b) = \sum_{j,k=1}^n (\partial_k D_{ij}, \partial_k b_j) - \int_{\Omega} b_i dx$ pour $b \in (H_0^1(\Omega))^n$. On voit que

$$f_i(b) = ((d^i, b)) - (1, b_i) \quad (7.15)$$

où d^i est la i ème ligne du tenseur D . En considérant un tenseur $B \in V_{0,\text{div}}$ ligne à ligne, on a, en notant b^i la i ème ligne de B ,

$$L(B) = \sum_{i=1}^n f_i(b^i).$$

f_i est clairement une forme linéaire bornée sur $H_0^1(\Omega)^n$:

$$|f_i(b)| \leq (|d^i|_{1,\Omega} + \sqrt{\text{Vol}(\Omega)} \cdot c(\Omega)) \cdot |b|_{1,\Omega} \quad (7.16)$$

On a donc une forme linéaire $f_i \in H^{-1}(\Omega)^n$ qui s'annule pour tout $b \in H_0^1(\Omega)^n$ tel que $\text{div} b = 0$. On peut alors faire appel au lemme 10 avec $U = \Omega$, domaine borné et à bord lipschitzien puisque de classe C^1 par morceaux. Il existe donc une fonction $p_i \in L^2(\Omega)$ telle que $-(p_i, \text{div} b) = f_i(b)$. En restreignant à $b \in \mathcal{D}(\Omega)^n$, on a même $f_i = \nabla p_i$ au sens des distributions : $\langle \nabla p_i, b \rangle = -(p_i, \text{div} b) = f_i(b)$.

Rassemblons les résultats pour tous les $i \in \{1 \dots n\}$: en posant $d = -p = -(p_1, \dots, p_n)^T$, on a pour tout $B \in V_0$:

$$(d, \text{div} B) = (-p, \text{div} B) = \sum_{i=1}^n (-p_i, \text{div} b^i) = \sum_{i=1}^n f_i(b^i) = L(B) = ((D, B)) + (I, B).$$

Maintenant que l'existence de d est établie, reste son unicité à vecteur constant près.

Si on se restreint à $B \in M_n(\mathcal{D}(\Omega))$, on a $L(B) = (d, \text{div} B) = \langle -\nabla d, B \rangle$, ce qui fait que si d_a et d_b sont des solutions d de (7.11), $\nabla d_a - \nabla d_b = 0$, donc d_a et d_b ne peuvent différer que d'une constante sur Ω qui est connexe. $\square$

7.3.2 Existence et unicité de d définie sur Ω , cas périodique localisé

Pour obtenir d , il a fallu travailler sur un domaine borné Ω — ce qu'on a obtenu en localisant — mais aussi sur des espaces de type H_0^1 qui ont une trace nulle sur tout le bord de ce domaine, y compris sur la frontière « artificielle » $\partial\Omega \setminus \Gamma$. Cette version ne reflète pas le problème original (7.1), même si elle en donne une solution sensée : d'après la remarque (2) p. 216 de [5], les conditions périodiques sur (7.1) sont extrêmement faibles et ne devraient avoir qu'une influence mineure dans la cellule représentative. Il n'est néanmoins pas satisfaisant de se contenter de cette simplification du problème sur $\partial\Omega \setminus \Gamma$.

Pour prouver le résultat dans le cas général (périodique localisé), il nous faut une version périodique du lemme 10. Adaptons l'approche constructive de J. Simon (*Démonstration constructive d'un théorème de G. de Rham*, [52]). Soit R le réseau engendré par $\{l_1, \dots, l_n\}$. Rappelons qu'une distribution T sur $\tilde{\Omega}$ qui est P -périodique (c'est-à-dire invariante par translation par une combinaison entière des $l_1, \dots, l_n$) s'identifie à une distribution simple $\dot{T}$ sur $\dot{\Omega} = \tilde{\Omega}/R$, le quotient pour l'addition dans $\mathbb{R}^n$. Ce sujet est abordé pour les séries de Fourier (Ouvrages de référence : [51, 61]).

L'avantage de travailler sur le tore topologique restreint $\dot{\Omega}$ est que les fonctions-test de $\mathcal{D}(\dot{\Omega})$ vérifient automatiquement la condition sur Γ , mais aussi celle de P -périodicité. On verra que c'est crucial pour trouver la primitive d'une distribution de $\mathcal{D}'(\dot{\Omega})^n$ s'annulant sur les vecteurs-test à divergence nulle et périodiques.

Théorème 12 *Soit une distribution $q \in \mathcal{D}'(\dot{\Omega})^n$ telle que*

$$\langle q, \psi \rangle = 0 \quad \text{pour toute } \psi \in \mathcal{D}(\dot{\Omega})^n \text{ telle que } \operatorname{div} \psi = 0. \quad (7.17)$$

Alors il existe $f \in \mathcal{D}'(\dot{\Omega})$ telle que $q = \nabla f$.

Preuve.

Il serait bien sûr infructueux de ne pas suivre la démonstration de J. Simon [52], qu'on peut reprendre sans aucune modification majeure. Il faut néanmoins préciser à chaque étape les distributions et fonctions qui sont périodiques et celles qui n'ont pas besoin de l'être — en fait, celles qui servent à faire des convolutions. Lorsqu'il le faudra, on vérifiera qu'après transformation, les distributions de $\mathcal{D}'(\dot{\Omega})$ conservent leur périodicité.

1. Symétrie des dérivées

Lemme 13 *Sous l'hypothèse (7.17), les dérivées de q sont symétriques : $\partial_i q_j = \partial_j q_i$.*

Preuve :

Soit $\varphi \in \mathcal{D}(\dot{\Omega})$ et $i \neq j$. Posons $\psi \in \mathcal{D}(\dot{\Omega})^d$ ainsi : $\psi_i = \partial_j \varphi$, $\psi_j = -\partial_i \varphi$ et $\psi_k = 0$ pour les autres indices : $i \neq k \neq j$. Immédiatement, on a $\operatorname{div} \psi = 0$. Donc, par (7.17), on obtient

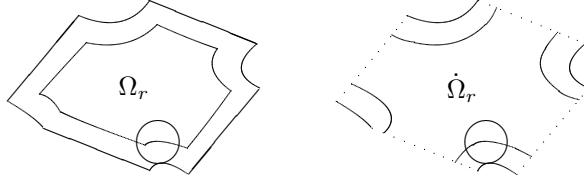
$$0 = \langle q, \psi \rangle = \langle q_i, \partial_j \varphi \rangle - \langle q_j, \partial_i \varphi \rangle = \langle \partial_i q_j - \partial_j q_i, \varphi \rangle.$$

Cela étant vrai pour toute fonction-test $\varphi \in \mathcal{D}(\dot{\Omega})$ on a donc identification des distributions : $\partial_i q_j - \partial_j q_i = 0$. $\square$

On voit déjà là que cette identification tomberait si l'on devait imposer *hors définition* une condition de périodicité à φ (par exemple si on avait $\varphi \in \mathcal{D}(\Omega)$ au lieu de $\varphi \in \mathcal{D}(\dot{\Omega})$).

2. Convolution locale

Pour tout $r > 0$, on définit $\dot{\Omega}_r = \{x \in \mathbb{R}^d/R : \dot{B}(x, r) \subset \dot{\Omega}\}$, où la boule $\dot{B}(x, r)$ est évidemment $B(x, r)/R$.



Pour la suite, on impose r assez petit pour que $\dot{\Omega}_{2r}$ ne soit pas vide. Le choix de $2r$ est dû au diamètre de $\dot{B}(x, r)$ et, on le devine, au support des distributions sur $\dot{B} = \dot{B}(0, r)$. La fonction de localisation α est définie sur le tore : $\alpha \in \mathcal{C}^\infty(\dot{\Omega})$ telle que $\alpha = 1$ sur $\dot{\Omega}_r$ et $\text{Supp } \alpha \subset \dot{\Omega}$. Si $v \in \mathcal{D}'(\dot{\Omega})$, alors le produit αv est aussi une distribution de $\mathcal{D}'(\dot{\Omega})$ et le prolongé par 0, $\widetilde{\alpha v}$, est défini sur $\mathbb{R}^d/R$. La convolution $\widetilde{\alpha v} * \mu$ est bien définie dans $\mathcal{D}'(\mathbb{R}^d/R)$ pour toutes les distributions périodiques $\mu \in \mathcal{D}'(\mathbb{R}^d/R)$ (et même pour les non-périodiques ($\mu \in \mathcal{D}'(\mathbb{R}^d)$)), car la convolution conserve la périodicité : si $U \in \mathcal{D}'(\mathbb{R}^d)$ est P -périodique, alors on a un translaté périodique

$$\tau_{l_j}(U * V) = \delta_{l_j} * (U * V) = (\delta_{l_j} * U) * V = (\tau_{l_j}U) * V = U * V$$

pour n'importe quelle distribution V et pour n'importe lequel des vecteurs l_j engendrant le réseau R . Ceci assure la bonne définition de la convolution locale

$$v \underset{\text{loc}}{*} \mu = (\widetilde{\alpha v} * \mu) \Big|_{\dot{\Omega}_{2r}}$$

dans $\mathcal{D}'(\dot{\Omega}_{2r})$.

Lemme 14 *La convolée locale ne dépend pas du choix de α .*

Preuve :

Soient $\alpha, \beta \in \mathcal{C}^\infty(\dot{\Omega})$ valant 1 sur $\dot{\Omega}_r$ et à support dans $\dot{\Omega}$. Alors, pour $\varphi \in \mathcal{D}(\dot{\Omega}_{2r})$, on a

$$\begin{aligned} \left\langle \widetilde{(\alpha - \beta)v} * \mu, \varphi \right\rangle_{\dot{\Omega}_{2r}} &= \left\langle \mu(y), \left\langle \widetilde{(\alpha - \beta)v}(x), \varphi(x + y) \right\rangle_{\dot{\Omega}_{2r}} \right\rangle_{\dot{B} = \dot{B}(0, r) \subset \mathbb{R}^d} \\ &= \left\langle \mu(y), \langle v(x), (\alpha - \beta)(x) \varphi(x + y) \rangle_{\dot{\Omega}_{2r}} \right\rangle_{\dot{B}} \end{aligned}$$

On voit que quand $x + y \notin \dot{\Omega}_{2r}$, $\varphi(x + y)$ s'annule ; sinon, $x \in \dot{\Omega}_{2r} + \dot{B} \subseteq \dot{\Omega}_r$ et alors c'est $(\alpha - \beta)$ qui s'annule. On a ainsi $(\alpha - \beta)v * \mu = 0$ □

Lemme 15 *Conservation de la convolée locale par restriction : Pour un ouvert $\dot{\omega}$ de $\dot{\Omega}$,*

$$v \Big|_{\dot{\omega}} \underset{\text{loc}, \dot{\omega}}{*} \mu = (v \underset{\text{loc}, \dot{\Omega}}{*} \mu) \Big|_{\dot{\omega}_{2r}}$$

La preuve est élémentaire : il suffit d'évaluer la différence des deux membres contre une fonction test $\varphi \in \mathcal{D}(\dot{\omega}_{2r})$.

Lemme 16 *La propriété suivante, connue pour la convolution usuelle, est valable pour convolée locale :*

$$\partial_j(v \underset{\text{loc}}{*} \mu) = v \underset{\text{loc}}{*} (\partial_j \mu) = (\partial_j v) \underset{\text{loc}}{*} \mu$$

A nouveau, la preuve est simple et ne contient pas de détour propre à la périodicité. On considère séparément $\partial_j(\widetilde{\alpha v} * \mu) = \widetilde{\alpha v} * \partial_j \mu$ (immédiat) et sur $\dot{\Omega}_{2r}$ où α est constante,

$$\left. \partial_j (\widetilde{\alpha v} * \mu) \right|_{\dot{\Omega}_{2r}} = \left. (\partial_j \widetilde{\alpha v}) * \mu \right|_{\dot{\Omega}_{2r}} = \left. (\widetilde{\alpha \partial_j v}) * \mu \right|_{\dot{\Omega}_{2r}}.$$

3. Primitive locale dans un domaine étoilé

On considère la solution élémentaire ξ de l'équation de Laplace $-\Delta \xi = \delta : \xi(x) = \frac{1}{2\pi} \log \frac{1}{\|x\|}$ pour $d = 2$ et, pour $d > 2$:

$$\frac{1}{(d-2)\text{Vol}(S^{d-1})} \|x\|^{2-d}.$$

Soit $\vartheta \in \mathcal{D}(\dot{B})$ valant 1 sur $\overline{\dot{B}(0, r/2)}$. On pose comme distributions $\gamma = \vartheta \xi$ (solution locale, ξ tronquée) et $\eta = 2(\nabla \vartheta | \nabla \xi) + \xi \Delta \vartheta$ qui est le terme correcteur. En effet

$$-\Delta \gamma = -(\vartheta \Delta \xi + 2(\nabla \vartheta | \nabla \xi) + \xi \Delta \vartheta) = \delta - \eta.$$

Les distributions γ et η restent définies, comme dans l'étape 3 de [52], sur la boule $B = B(0, r)$ de $\mathbb{R}^d$, et non sur une boule $\dot{B}(0, r)$ du tore : elles n'apparaissent par la suite « que » dans le rôle de μ pour la convolution locale, et, on l'a vu, μ peut être non-périodique. La primitive F de q se construit alors sur un ouvert $\dot{\omega}$ dans $\dot{\Omega}_{2r}$ étoilé par rapport à un point a de $\dot{\Omega}_{2r}$. On définit $h \in \mathcal{C}^\infty(\dot{\omega})$

$$h(x) = \int_0^1 \sum_{k=1}^d (x-a)_k (q_k \underset{\text{loc}}{*} \eta)(a + t(x-a)) dt$$

puis enfin la primitive locale $F \in \mathcal{D}'(\dot{\omega})$ de q par

$$F = h - \sum_{k=1}^d q_k \underset{\text{loc}}{*} \partial_k \gamma. \quad (7.18)$$

Lemme 17 *F est une primitive de q dans l'étoile $\dot{\omega} : \nabla F = q$.*

La preuve de ce lemme est identique au cas non-périodique. On vérifie au moyen des lemmes 13 et 16 que

$$(\partial_j h)(x) = \int_0^1 \left[1 + t \frac{d}{dt} \right] (q_j \underset{\text{loc}}{*} \eta)(a + t(x-a)) dt = (q_j \underset{\text{loc}}{*} \eta)(x)$$

pour un point $x \in \dot{\Omega}_{2r}$. Les lemmes 13 et 16 donnent par ailleurs

$$\partial_j (q_k \underset{\text{loc}}{*} \partial_k \gamma) = \partial_j q_k \underset{\text{loc}}{*} \partial_k \gamma = \partial_k q_j \underset{\text{loc}}{*} \partial_k \gamma = q_j \underset{\text{loc}}{*} \partial_k^2 \gamma.$$

Par la définition de F (7.18), on a

$$\partial_j F = q_j \underset{\text{loc}}{*} \eta - q_j \underset{\text{loc}}{*} \sum_{k=1}^d \partial_k^2 \gamma = q_j \underset{\text{loc}}{*} (\eta - \Delta \gamma) = q_j \underset{\text{loc}}{*} \delta = q_j.$$

4. Fonctions-test à divergence nulle

Pour un lacet L dans $\mathbb{R}^d/R$ et une fonction $\varrho \in \mathcal{D}(\dot{B})$ où $\dot{B} = \dot{B}(0, r)$, on définit la

distribution $T \in \mathcal{D}'(\mathbb{R}^d/R)$ par

$$\langle T, \varphi \rangle = \int_L (\varphi \cdot dl) \quad \text{pour toute } \varphi \in \mathcal{D}(\mathbb{R}^d/R)^d. \quad (7.19)$$

et la fonction-test $\Psi \in \mathcal{D}(\mathbb{R}^d/R)$ par

$$\Psi(x) = \int_L \varrho(l - x) dl \quad (7.20)$$

Lemme 18 *On a $\operatorname{div} T = 0$; le support de T est L .*

Lemme 19 *On a $\Psi(x) = (T * \check{\varrho})(x)$ où par définition $\check{\varrho}(x) = \varrho(-x)$. De plus, $\operatorname{div} \Psi = 0$. Enfin, le support de Ψ est contenu dans $L + \dot{B}$.*

Les preuves sont élémentaires. Remarque sur la périodicité : le lacet L servant à définir T doit être vu dans $\mathbb{R}^d/R$ car il sert à relier des paires de connexes disjoints de $\dot{\Omega}$ pour l'ultime étape, le recollement des primitives locales. C'est pourquoi T est défini dans $\mathcal{D}'(\mathbb{R}^d/R)$ et la fonction lisse Ψ à support dans le lacet épaissi d'au plus r est définie dans $\mathcal{D}(\mathbb{R}^d/R)$.

5. Primitive globale

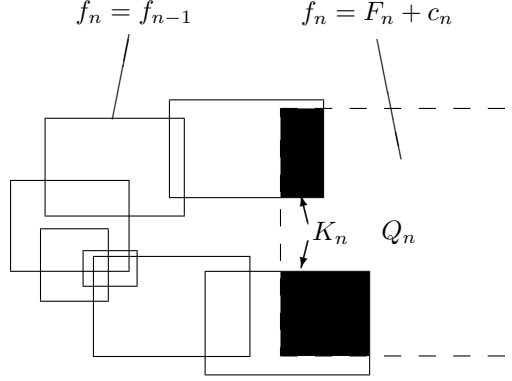
Cette étape consiste à recoller progressivement les primitives locales de q définies sur des ouverts étoilés recouvrant le domaine $\dot{\Omega}$.

Pour ce faire, soit un recouvrement de $\dot{\Omega}$ par une suite de cubes ouverts $\{Q_n\}_n$ tels que $\bar{Q}_n \subset \dot{\Omega}$. On pose $\omega_n = (Q_1 \cup \dots \cup Q_n)$ et on suppose que $Q_{n-1} \cap Q_n$ est non vide.

Pour chaque Q_n , le lemme 17 donne une primitive F_n de q , donc $\nabla F_n = q$ sur Q_n . Comme pour toute constante c , $F_n + c$ est aussi une primitive, on peut essayer de construire une primitive globale par recollement. On prouve que c'est possible par récurrence. Supposons qu'on ait un recollement partiel à savoir une distribution $f_{n-1} \in \mathcal{D}'(\omega_{n-1})$ telle que $\nabla f = q$ dans ω_{n-1} . La partie à recoller pour construire f_n est $K_n = \omega_{n-1} \cap Q_n$. Si la distribution $D_n = (f_{n-1} - F_n)$ est constante ($=: c_n$) sur K_n , alors on a une primitive f_n définie sur ω_n : $f_n = f_{n-1}$ sur ω_{n-1} et $f_n = F_n + c_n$ sur Q_n ; on peut alors conclure par récurrence et définir $f \in \mathcal{D}'(\dot{\Omega})$ primitive de q sur $\dot{\Omega}$. Il reste donc à prouver le

Lemme 20 *La distribution $D_n = (f_{n-1} - F_n)$ est constante sur K_n .*

La difficulté est que K_n n'est plus forcément connexe, mais, comme $\nabla D_n = 0$, il s'ensuit que D_n est une fonction constante sur chaque composante connexe.



Pour deux points a et b pris dans deux composantes connexes distinctes de K_n , on calcule $c = D(b) - D(a)$. Si on prouve que $c = 0$, alors D est la même constante sur (toutes les composantes de) K_n .

En prenant r assez petit pour que $\dot{B}(a, r)$ et $\dot{B}(b, r)$ soient inclus dans l'ouvert K_n , considérons une fonction-test $\varrho \in \mathcal{D}(\dot{B}(0, r))$ d'intégrale 1. Alors $(D_n *_{\text{loc}} \varrho)(a) = D_n(a)$, idem pour b . Alors

$$c = (D_n *_{\text{loc}} \varrho)(b) - (D_n *_{\text{loc}} \varrho)(a).$$

Ecrivons encore c sous la forme $c = c_1 + c_2$ où

$$\begin{aligned} c_1 &= (f_{n-1} *_{\text{loc}} \varrho)(b) - (f_{n-1} *_{\text{loc}} \varrho)(a) \\ c_2 &= (F_n *_{\text{loc}} \varrho)(a) - (F_n *_{\text{loc}} \varrho)(b) \end{aligned}$$

On choisit L_1 , un chemin de a à b passant dans ω_n et L_2 , de b à a passant par Q_n . Quitte à réduire r pour avoir $L_1 \subset (\omega_{n-1})_{2r}$, et $L_2 \subset (Q_n)_{2r}$ on a par les lemmes 15 et 16

$$\begin{aligned} c_1 &= \int_{L_1} \nabla(f_{n-1} *_{\text{loc}} \varrho) \cdot dl = \int_{L_1} (q *_{\text{loc}} \varrho) \cdot dl \\ c_2 &= \int_{L_2} \nabla(F_n *_{\text{loc}} \varrho) \cdot dl = \int_{L_2} (q *_{\text{loc}} \varrho) \cdot dl \end{aligned}$$

Le lacet $L = L_1 \cup L_2$ dans $\mathbb{R}^d/R$ permet d'utiliser la définition de T (7.19) de l'étape précédente et de conclure au moyen des lemmes 18 et 19

$$\begin{aligned} c &= \int_L^0 (q *_{\text{loc}} \varrho) \cdot dl = \langle T, \widetilde{\alpha q} * \rho \rangle_{\mathbb{R}^d/R} = \langle \widetilde{\alpha q}, T * \check{\rho} \rangle_{\mathbb{R}^d/R} \\ &= \langle q, \alpha \cdot (T * \check{\rho}) \rangle_{\dot{\Omega}} = \langle q, \alpha \Psi \rangle_{\dot{\Omega}} = \langle q, \Psi \rangle_{\dot{\Omega}} = 0. \end{aligned}$$

□

Remarque : une variante au théorème 12 peut consister à adapter une preuve récente de J. Simon sur les solutions semi-périodiques de Navier-Stokes [1] : existence, extension et

périodicité de p , pages 350-351.

Le lemme suivant, bien connu pour les équations de Navier-Stokes [58], permet de passer p d'une distribution à une fonction L^2 .

Lemme 21 *Soit U un ouvert borné à bord lipschitzien de $\mathbb{R}^d$ et $p \in \mathcal{D}'(U)$ une distribution. Si toutes les dérivées au premier ordre $\partial_{x_i} p$ sont des fonctions de $L^2(U)$, alors $p \in L^2(U)$ et*

$$\|p\|_{L^2(U)/\mathbb{R}} \leq C \|\nabla p\|_{L^2(U)^d}$$

Ce lemme est le point i) de la proposition I.1.2 de [58].

Théorème 22 *Il existe une unique $D \in V_{P,\Gamma,\text{div}}$ et un $d \in L^2(\Omega)^n$ unique à vecteur constant près tels que*

$$((D, B)) + (I, B) - (d, \text{div } B) = 0 \quad \text{pour tout } B \in V_{P,\Gamma}. \quad (7.21)$$

Preuve : Les principales étapes sont celles du théorème 11. L'existence et l'unicité du tenseur D est cette fois donnée par le théorème 8.

On définit la forme $L(B) = ((D, B)) + (I, B)$, $B \in V_{P,\Gamma}$. Elle est continue, exactement comme son homologue non-périodique (7.12). On peut reprendre sans autres la mécanique de la preuve 11 en remplaçant $H_0^1(\Omega)$ par $H_{\Gamma,P}$, V_0 par $V_{P,\Gamma}$, et ainsi de suite. La forme linéaire $f_i(b) = ((d^i, b)) - (1, b_i)$ est définie et bornée sur $H_{\Gamma,P}^n$:

$$|f_i(b)| \leq \left(|d^i|_{1,\Omega} + \sqrt{\text{Vol}(\Omega)} \right) \cdot |b|_{1,\Omega} \quad (7.22)$$

donc $f_i \in H_{\Gamma,P}'^n$. C'est aussi une distribution de $\mathcal{D}'(\dot{\Omega})^n$ puisque $\mathcal{D}(\dot{\Omega})$ s'identifie à $\mathcal{D}_P$ qui est un sous-espace dense de $H_{\Gamma,P}$ (théorème d'injection canonique de duals). Le théorème 12 donne ainsi une unique distribution $p_i \in \mathcal{D}'(\dot{\Omega})$ telle que $f_i(b) = \langle \nabla p_i, b \rangle = -(p_i, \text{div } b)$ pour $b \in \mathcal{D}'(\dot{\Omega})^n$. Par densité, on obtient $f_i(b) = -(p_i, \text{div } b)$ pour les b vecteurs de $H_{\Gamma,P}^n$. Le lemme 21 permet en outre d'identifier p_i à une fonction de $L^2(\Omega)$, en prenant $U = \Omega$. La fin de la preuve est celle de 11 : en sommant sur i , on obtient $d \in L^2(\Omega)^n$ qui est unique à vecteur constant près.

□

Conclusion et perspectives

La partie du présent travail consacrée à la méthode STILS a contribué, nous l'espérons, à éclaircir quelques aspects de l'énigme qu'est la recherche d'une solution numérique convenable à l'équation de transport. Bien sûr, cette énigme reste entière : les essais de solutions développés au cours des cinquante dernières années (caractéristiques, schémas aux différences et d'innombrables dérivés de la méthode des éléments finis de Galerkin) sont très différents entre eux et une bonne solution (à inventer, donc) sera encore bien différente, sans être forcément compliquée.

Cependant, on peut affirmer que pour une discrétisation en espace-temps et un jeu de fonctions de base donnés, STILS constitue un optimum parmi les méthodes d'éléments finis pour l'équation de transport. Le théorème de P. Azérad le prouve déjà ; les comparaisons expérimentales ont permis en outre de situer les méthodes courantes par rapport à cet optimum.

La marge de manœuvre reste donc dans le choix du maillage et des fonctions de base. Les techniques ne manquent pas dans ce domaine. Les fonctions discontinues en particulier constituent un champ d'investigation prometteur, bien que non dénué d'embûches, comme on l'a vu au chapitre 5.

Par ailleurs, une comparaison théorique et numérique entre STILS « marche-en-temps » et STILS sur tout l'intervalle $]0; T]$ pourrait être intéressante bien que, pour un maillage en espace-temps de type maillage en espace $\times$ maillage en temps, le support d'une fonction de base en temps pour $t = t_k$ n'excède pas l'intervalle $[t_{k-1}; t_{k+1}]$, ce qui rend la méthode peu différente de la marche en temps. Aussi pourrait-on songer à un maillage entrelacé en espace-temps, par exemple. Un champ de comparaison encore plus large, incluant les volumes finis, aurait également son intérêt.

Du côté des équations non-linéaires, STILS n'a pas encore dévoilé tous ses atouts, notamment pour des équations autres ou de forme plus générale que l'équation de Burgers.

Quant à la détermination du tenseur K pour la loi de Darcy, rappelons qu'une information précise sur ce tenseur est, lorsqu'on a couplage entre l'équation de diffusion et l'équation de transport, nécessaire à la résolution de cette dernière. L'existence et l'unicité d'une solution au problème exact sous certaines conditions donnent une base fertile à la résolution du problème approché ; aussi pouvons-nous espérer faciliter la construction de simulations dans ce type de problème.

Annexe A

Programmes de résolution

A la table de Scrabble, en revanche, la même faible et folle Ada se transformait en une sorte d'ordinateur élégant, doué, par surcroît, d'une chance phénoménale et vous préparait les mots les plus longs et les plus alléchants avec les miettes et les rogatons les moins prometteurs.

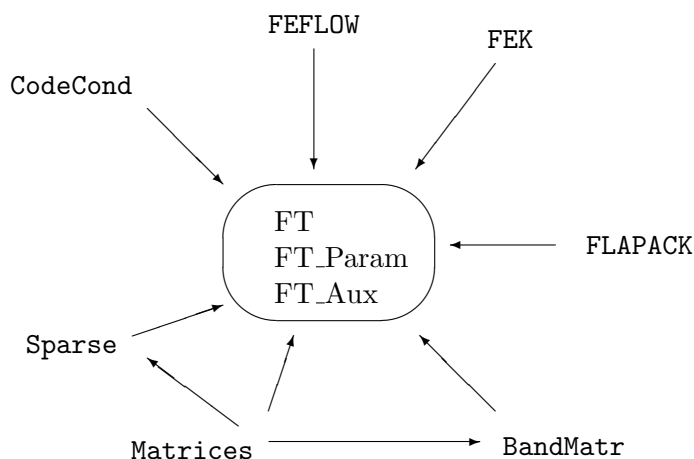
Vladimir Nabokov, *Ada*, I, ch. XXXVI

A.1 Structure

Plutôt que de reproduire tous les listages informatiques impliqués dans les calculs numériques (environ 20'000 lignes), nous ne reproduirons ici que les parties conceptuellement intéressantes ou nouvelles — en particulier la réalisation de STILS marche-en-temps.

Au préalable, voici un aperçu de l'organisation générale du programme de résolution de l'équation d'écoulement (1), par la méthode de Galerkin simple, couplée à celle de transport (1.1), par les diverses méthodes décrites précédemment.

- a) D'abord, résumons par un schéma la structure modulaire du programme FT (pour flux-transport) :



`CodeCond` traite les codes pour les conditions initiales et aux limites, ainsi que les

champs de vitesse et tenseurs : K , conductivité fluide pour l'équation de Darcy (1) et D , diffusion-dispersion pour l'équation de convection (1.1). Les parties **Matrices**, **Sparse** et **BandMatr** ont trait aux matrices sous format rectangulaire, creux et bande — remplissage, produits matrice-vecteur, méthodes du gradient pour les matrices creuses. **FLAPACK** fait le lien avec la bibliothèque **LAPACK** pour la décomposition LU et la résolution directe sur des matrices bandes. **FEK** signifie Finite Element Kernel, bibliothèque de M. Bercovier (Université de Jérusalem) qui permet de calculer les fonctions de base usuelles à partir des coordonnées des nœuds d'un élément. **FEFLOW** contient les paramètres de quelques problèmes parmi les nombreux que fournit la documentation du logiciel **FEFLOW** [21].

- b) Voici l'ensemble (simplifié) des combinaisons d'options possibles qui se sont peu à peu accumulées. Elles sont pour l'essentiel dans le module **FT_Param** (cf C.4).
- Matrices bandes, matrices creuses. Les premières deviennent trop volumineuses, ou la méthode LU trop lente, pour certains problèmes, mais les méthodes itératives de gradient qu'on emploie avec les matrices creuses, quoique plus rapides que les méthodes directes, ne fonctionnent pas toujours. Dans le cas symétrique définie positive (gradient conjugué), on peut ne pas avoir de convergence en N pas comme ce devrait être le cas pour une précision infinie ; dans le cas asymétrique (gradient bi-conjugué) il y a en plus des cas de *rupture* où la taille du pas est quasi nulle — ceci n'affecte pas **STILS** dont la matrice A est symétrique définie positive.
 - Matrices simples, en 2×2 blocs ou 3×3 blocs selon le degré en temps de **STILS**, +1 pour les fonctions discontinues.
 - Schéma en temps : théta, BDF ou provenant d'une formulation en espace-temps.
 - Combinaison de conditions aux limites de Dirichlet et de Neumann pour les deux équations.
 - 14 méthodes de résolution pour l'équation de transport.
 - Champ de vecteurs v provenant d'un potentiel — équation de Darcy (1) —, donné analytiquement, donné sur fichier puis interpolé par les fonctions de base ou, pour l'équation de Burgers (6.3), pris sur la solution au pas de temps précédent.
 - Ecoulement (h, v) stationnaire ou non.
 - Dimensions spatiales 1,2 ou 3 ; éléments Q_1 ou Q_2 .

Tout cela engendre d'abondantes subtilités comme le re-remplissage des matrices pour le Shock-capturing et l'équation de Burgers, ou le fait que le jeu d'inconnues et d'équations linéaires n'est pas le même entre les deux EDP si les types de conditions aux limites (Dirichlet, Neumann) ne se retrouvent pas aux mêmes nœuds, d'où une numérotation indicée par le genre d'EDP et une structure de matrices creuses différente, etc.

Heureusement le typage fort et la conception claire du langage de programmation (Ada) a permis de tenir bon face à une complexité à croissance exponentielle et ce, pour un temps*personnel plutôt avantageux !

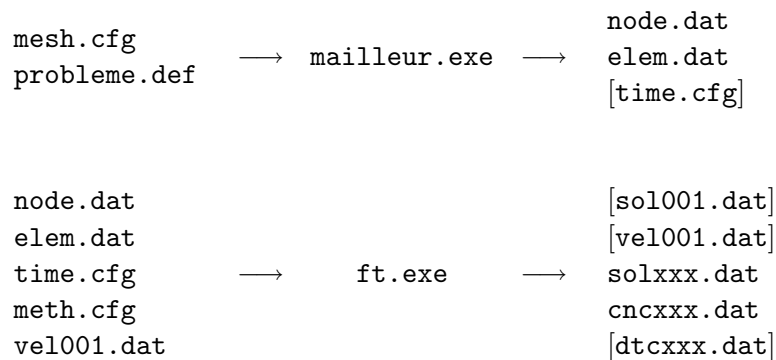
Un mot sur la **portabilité**. Certaines bibliothèques aux noms particuliers à DEC Ada (hors de l'ancienne norme Ada 83) sont intégrées à la norme Ada 95. Il suffit de changer quelques clauses d'importation : `float_math.lib` en `Ada.Numerics.Elementary_functions`, utiliser `Ada.Numerics.Float.Random`.

L'interfaçage avec **LAPACK** (voir **FLAPACK**) et **BLAS** (dans **Sparse**) devrait fonctionner sans autres pour autant que ces bibliothèques soient présentes sur le système. On peut même se passer complètement de **BLAS** et **LAPACK** en choisissant **SparseB0** pour **Sparse** et en utilisant le format de matrices creuses. D'autre part, l'interfaçage a été fait ici uniquement

pour les quelques fonctions utilisées dans le projet. Si on cherche des *bindings* complets pour LAPACK et BLAS, il existe respectivement « LAPACK-Ada » (on le trouve sur de nombreux sites FTP sous les noms **lapack-ada** ou **lapada**) et le récent « BLAS Ada binding » (URL : <http://topo.math.u-psud.fr/~sands/Programs/BLAS/index.html>).

A.2 Fonctionnement

Le processus de simulation passe d'abord par un **mailleur** qui établit un maillage fin à l'aide de macro-éléments décrits pour chaque problème. Si le maillage existe déjà — par exemple produit par un mailleur automatique —, on saute cette étape. La simulation proprement dite est faite par le programme **FT**. Le schéma suivant montre les fichiers en entrée et ceux en sortie pour ces deux étapes.



Que signifient ces noms de fichiers ? **mesh.cfg** contient simplement le nom du fichier de définition du problème, par exemple **probleme.def**. Celui-ci est lu par le **mailleur** qui produit un fichier de coordonnées de noeuds (**node.dat**) et un fichier qui indique les numéros des noeuds qui composent chaque élément (**elem.dat**). Eventuellement, il recopiera de **probleme.def** la configuration des pas de temps, **time.cfg** ; en l'absence d'un fichier de problème (maillage imposé), on modifie directement **time.cfg**.

Pour ce qui est de l'étape principale, le programme **FT** lit **node.dat**, **elem.dat** et **time.cfg** qui donnent la représentation discrétisée précise du domaine $\tilde{\Omega}$ en espace-temps. Il lit aussi **vel001.dat** qui contient le champ de vecteurs v en des points donnés (noeuds ou centres d'éléments). Si v dérive d'un potentiel, il produit **vel001.dat** ainsi que le potentiel h (**sol001.dat**) comme solution de l'équation de flux diffusif, avant de résoudre l'équation de transport. Cette résolution produit les fichiers **solxxx.dat** (solution approchée de c), **cncxxx.dat** (concentration intégrale) et, pour une méthode découplée, **dtxxxx.dat** pour l'approximation de $\frac{\partial c}{\partial t}$. Le « xxx » vaut 002 pour la méthode particulière, 003 pour la méthode de Galerkin et ainsi de suite. Le nom de la méthode est lu dans **meth.cfg**. La liste complète des méthodes se trouve dans **FT.Param** — voir §C.4. Le prochain paragraphe explique le format de ces différents fichiers.

A.3 Formats de fichiers

Tous les fichiers en question sont du texte (non-binaire). Parmi ceux qui suivent, ceux qui sont suffisants pour une simulation sont **mesh.cfg**, **meth.cfg**, (**probleme**).**def**. Le reste est générés par le mailleur (**mailleur.exe**) ou le programme de résolution (**ft.exe**).

- **mesh.cfg** : une ligne de texte ; contient le nom du fichier de maillage (par ex. **probleme.def**). Les autres lignes sont ignorées.

- **meth.cfg** : texte; contient la liste des méthodes à essayer pour résoudre l'équation de transport (liste : §C.4). Les noms sont séparés par des espacements ou des sauts de ligne. Il suffit d'un mot non reconnu pour terminer la liste.
- **time.cfg** : indications sur les pas de temps. Exemple « auto-explicatif » :

```

20          -- pas de temps retenus
1.2566371   -- secondes par pas retenu
159         -- nombre de pas intermediaires Elements Finis
50          -- nombre de pas maximum methode caracteristiques
0           -- nombre vagues d'injection (methode caracteristiques)
*
<--- fin time.cfg, produit par hansbo.def

```

- **probleme.def** : Il y a un petit « langage ». Chaque mot-clé tient sur une ligne et les lignes suivantes sont les paramètres. Une ligne (qui ne sert pas de paramètre) commençant par un espacement est considérée comme commentaire.

gelem : c'est le type d'élément utilisé. Le choix actuel est L2, Q4, Q9, B27 en notation FEK (cf annexe B).

nbeq : nombre d'équations (1 : flux; 2 : flux et transport, même si l'équation de flux est résolue ailleurs ou n'en est pas une).

ci : condition initiale. Les paramètres sont 1 ou 2 (selon **nbeq**) entiers qui contiennent les codes identifiant la fonction de condition initiale pour chaque équation. Voir le module **Codecond** pour les codes (annexe B).

source : idem, pour la fonction source de chaque équation.

tens : idem, pour les tenseurs K et D .

vit : idem, pour la vitesse v (n'a de sens que pour l'équation de transport).

temps : copie dans le fichier **time.cfg** les lignes suivantes jusqu'à une ligne vide.

grid : copie dans le fichier **grid.dat** les lignes suivantes jusqu'à une ligne vide. Le fichier en question contient une description sommaire d'un maillage rectangulaire, qui est utilisée pour la visualisation de solutions par MathCAD sous la forme d'une matrice par pas de temps. La transformation est faite par **dat2mcad**. Voici un échantillon de **grid.dat** :

```

2          -- dimension spatiale
100 -1.0 1.0      -- # subdivisions, bornes en x
100 -1.0 1.0      -- # subdivisions, bornes en y
200 0.0 6.28318530718 -- # pas de temps, bornes en t
<--- fin grid.dat

```

zcoor. Ligne 1 : bornes en z des macro-éléments tridimensionnels subséquents, puis nombre d'éléments (en hauteur) produits au sein des macro-éléments. Ligne 2 : codes de condition limite en bas et en haut. Exemple :

```

zcoor
0.0 1.0    5
-1    -1

```

rec : macro-élément en forme de trapèze ou, en 3D, parallélépipède vertical à base trapézoïdale. Chacune des quatre lignes suivantes commence par les codes de condition aux limites puis les coordonnées d'un sommet du trapèze (en orientation positive, i.e. sens contraire aux aiguilles de la montre, sinon il y a faute de signe en 3D pour les conditions de Neumann sur les faces horizontales!). La cinquième ligne indique le nombre d'éléments à construire dans les coordonnées x et y locales. Exemple :

```

rec
1 0    -1.0    -1.0
0 0     0.0    -1.0
0 0     0.0     0.0
1 -1    -1.0     0.0

```

- 50 50
seg : idem, mais pour un segment 1D. Exemple :
- ```

seg
1 1 0.0
1 1 1.0
198

```
- rem** : remarque (affichée par le mailleur).  
**fin** : ce mot-clé termine le fichier de définition.
- **node.dat**. Normalement produit par le mailleur, ce fichier dédié aux **nœuds** a le format suivant : une ligne avec le nombre de nœuds, puis pour chaque nœud, une ligne avec ses coordonnées et 1 s'il est sur le bord, 0 sinon. Exemple :
 

```

10201
-1.00000E+00 -2.00000E-01 1
-1.00000E+00 -2.20000E-01 1
-9.80000E-01 -2.20000E-01 0
...

```
  - **elem.dat**. Le mailleur produit aussi ce fichier d'**éléments** selon le format suivant : une ligne avec le type d'élément ; une ligne avec le nombre d'éléments puis le nombre d'équations ; une ligne par équation avec les codes CI, source, tenseur, vitesse ; une ligne par élément avec les numéros de nœuds suivis des codes de CL pour les nœuds pouvant être à l'extérieur (0 : non-condition). Exemple :
 

```

Q4
1156 2 = nbelem, nbeq
0 0 0 0 = codes CI / src / tens / vit, eq: 1
8 0 6 1 = codes CI / src / tens / vit, eq: 2
16 7 8 17 1 0 0 0 0 0 0 0
7 2 4 8 1 0 0 0 0 0 0 0
2 1 3 4 1 0 0 0 0 0 0 0
1 5 6 3 1 0 0 0 0 0 0 0
5 13 14 6 1 0 0 0 0 0 0 0
13 25 26 14 1 0 0 0 0 0 0 0
380 327 328 381 0 0 0 1 0 0 0 -1
327 278 279 328 0 0 0 0 0 0 0 0
...

```
  - **solxxx.dat**. Fichier de **solution**. Ligne 1 : rappel du nombre de nœuds et nombre de pas de temps. Puis, pour chaque pas de temps : une ligne avec le pas de temps suivi des valeurs aux nœuds. Exemple :
 

```

1225 21
1
-3.3151E-04 -4.8389E-04 -5.3659E-04 -5.3537E-04 -5.3778E-04 ...

```
  - **velxxx.dat**. Fichier de champ de **vitesse**. Les vecteurs sont aux nœuds ou au centre des éléments mot-clé **vel\_noeuds** ou **vel\_elems** en première ligne. Puis, les composantes des vecteurs. Exemple :
 

```

VEL_ELEMS
9.706E-01 -9.706E-01
9.706E-01 -9.118E-01
...

```
  - **cncxxx.dat**. Fichier des **concentrations** intégrales pour chaque pas de temps. Exemple :
 

```

0 2.56055401384002E-01
1 2.55731231702301E-01
2 2.54567416706331E-01
3 2.54846586018884E-01
4 2.55280947936901E-01

```

...

## A.4 Utilitaires

Voici une liste des programmes indépendants ayant un rôle dans ce projet.

- **varcnc** mesure la variation relative de concentration
- **cs** compare deux solutions point par point en espace-temps ; calcule les extréma et les variations relatives
- **stilburg** résout l'équation de Burgers par STILS (non-linéarisé) par la formule établie dans 6.1
- **theis\_ex** évalue la solution exacte du problème de Theis vu en introduction (6) en vue de la comparer à la solution approchée par éléments finis de FT, au moyen du programme **cs**.
- **tomatrix** convertit tous les fichiers solutions **solxxx.dat** sous format matrice (ligne :  $x_i$ , colonne :  $t_j$ ) pour affichage par MathCAD ; suppose que le maillage est **unidimensionnel** et que les nœuds sont ordonnés canoniquement
- **dat2mcad** convertit les fichiers solutions d'un problème **bidimensionnel** en matrices (une par pas de temps) et crée un fichier MathCAD qui va lire ces matrices pour les afficher
- **radial** produit un maillage proportionnel pour le problème de Theis
- **deupuits** produit un maillage pour un problème à deux puits (injection-pompage)

# Annexe B

## Modules, ossature

Cette annexe contient uniquement les spécifications des différentes composantes du projet. C'est la prochaine qui contient les algorithmes essentiels pour la résolution.

### B.1 Matrices usuelles

#### B.1.1 Cadre générique

Le module générique G\_Matrices comporte quelques opérations de base sur les matrices.

---

```
-- File : gmatrice.ads
-- Description : simple generic matrix package
-- Date / Version : 17-Jan-2000 /
-- 28.I.1998 / 16.XI.1998 / 30.VII.1998 / 22 XII 1997
-- Author : Gautier.deMontmollin@UniNe.CH
```

---

```
generic -- Requires a field with a square root
-- e.g. floating point numbers, fixed point numbers

 type field_elt is private;
 zero, one: field_elt; -- 0 and 1 elements

 with function "-" (a: field_elt) return field_elt; -- unary oper.
 with function sqrt(a: field_elt) return field_elt;

 with function "+" (a,b: field_elt) return field_elt; -- binary oper.
 with function "-" (a,b: field_elt) return field_elt;
 with function "*" (a,b: field_elt) return field_elt;
 with function "/" (a,b: field_elt) return field_elt;
-- no more "=" for Ada 83 compatibility

package G_Matrices is

 TYPE Vector IS ARRAY (Integer RANGE <>) OF field_elt ;
 TYPE Matrix IS ARRAY (Integer RANGE <>, Integer RANGE <>) OF field_elt ;
 TYPE IVector IS ARRAY (Integer RANGE <>) OF Integer ;
 TYPE IMatrix IS ARRAY (Integer RANGE <>, Integer RANGE <>) OF Integer ;

 type p_matrix is access matrix;
 type p_imatrix is access imatrix;
 type p_vector is access vector;
```

```

type p_ivector is access ivector;

-- vector operations

function "*" (l:field_elt; v:vector) return vector;
function "-" (a:vector) return vector;
function "+" (a,b:vector) return vector;
function "-" (a,b:vector) return vector;
function "*" (a,b:vector) return field_elt; -- produit scalaire
function norm(a:vector) return field_elt;
function square_norm(a:vector) return field_elt; -- norme au carr\'e
function dist(a,b:vector) return field_elt;
function square_dist(a,b:vector) return field_elt;

-- matrix operations

function Transpose(A:matrix) return matrix;
function Id(order: positive) return matrix;
function "*" (l:field_elt; A:matrix) return matrix;
function "*" (A,B:matrix) return matrix;
function "+" (A,B:matrix) return matrix;
function "-" (A,B:matrix) return matrix;

-- matrix-vector operations

function "*" (A:matrix; x:vector) return vector;

end;

```

### B.1.2 Instanciation particulière

Le projet FT utilise des réels à double précision (`long_float`), d'où l'instanciation suivante :

```

with G_Matrices;
with Long_float_math_lib; use Long_float_math_lib;

package Matrices is new
 G_Matrices(long_float, 0.0,1.0, "-", Sqrt,"+", "-", "*", "/");

pragma suppress_all;

```

## B.2 BandMatr : matrices bandes

Voici la spécification du module pour le calcul sur matrices à faible largeur de bande.

---

```

-- File : bandmatr.ada
-- Description : Generic package for band matrices
--
-- Symmetric and Asymmetric Lapack band matrix formats
--
-- Date / Version : 3-AUG-2000
-- Author : Gautier.deMontmollin@UniNe.CH

```

---

```

generic
 type real is digits <>;
 type index is range <>;
 type vector is array(index range <>) of real;
 type matrix is array(index range <>, index range <>) of real;

package BandMatr is

 -- Define a band matrix --

 type Band_matrix(max_size, -- size of actual matrix
 half_band, -- half-bandwidth
 multi_band -- n-uple of half-bandwidth
 : index) is

 record
 val: matrix(1..max_size, 1..multi_band); -- transposed w/r Lapack due to Fortran
 symmetric: boolean;
 size: index:= max_size;
 end record;

 -- Lapack:
 -- minimal values for multi_band | 1+3*half_band: asymmetric case
 -- | 1+ half_band: symmetric case

 -- # of super/subdiagonals: half_band-1 (for KU/KL/KD params in Lapack)

 -- Example
 -- DGBTRS(TR=> no_transpose, N=>MB.size,
 -- KL=>MB.half_band - 1,
 -- KU=>MB.half_band - 1,
 -- AB =>MB.val, LDAB=>MB.multi_band,
 -- IPIV=> pivots, B=> sk, LDB=>MB.size);

 type p_Band_matrix is access Band_matrix;

 -- Matrix-vector multiplication --

 -- w:= A*u

 procedure Mult(A: in out Band_matrix; u: vector; w: in out vector);

 -- Put/Add/Get data into/in/from matrix --

 -- procedure Put(A: in out Band_matrix; i,j: index; value: real);
 procedure Add(A: in out Band_matrix; i,j: index; value: real);
 -- function Get(A: in Band_matrix; i,j: index) return real;

end BandMatr;

```

Le module FLAPACK fait l'interface avec les routines LAPACK suivantes :

- DGBTRF décomposition  $A = LU$  pour matrices asymétriques,
- DGBTRS résolution d'un système linéaire asymétrique mis sous forme  $A = LU$
- DPBTRF décomposition de Cholesky  $A = L^T L$  pour matrices symétriques

- DPBTRS résolution d'un système linéaire symétrique mis sous forme  $A = L^T L$

### B.3 Sparse : matrices creuses

Pour les calculs de base et la résolution de systèmes linéaires sur matrices creuses, on a le module **Sparse**. Les matrices ont le format CRS (Compressed Row Storage) dont le principe est de ne retenir, ligne après ligne, que les composantes susceptibles d'être non-nulles. Ces composantes sont mises bout-à-bout (**val**) et repérées par un tableau d'indices (**col\_ind**). Le début de chaque ligne est repéré par un autre tableau (**row\_ptr**) Le stockage en lignes permet de faire un produit  $Ax$  en suivant les données de **val**.

---

```
-- File : sparse.ads
-- Description : Sparse matrices, generic
-- compressed row storage (crs)
--
-- Date / Version : 17-Jan-2000 / 29.III.1999
-- Author : Olivier.Besson@UniNe.CH
-- Ada version : Gautier.deMontmollin@UniNe.CH
```

---

**generic**

```
type real is digits <>;
type index is range <>;
type vector is array(index range <>) of real;
```

**package Sparse is**

```
type index_array is array(index range <>) of index;
```

---

```
-- Define a matrix with compressed rows storage --
```

---

```
type CRS_matrix(rows_max_p1, nnz_max: index) is
record
 val: vector(1..nnz_max);
 col_ind: index_array(1..nnz_max);
 row_ptr: index_array(1..rows_max_p1); -- p1 means +1 -> extra row ptr
 symmetric: boolean;
 rows: index:= rows_max_p1-1; -- for re-use; must be in [1..rows_max_p1-1]
 nnz: index:= nnz_max; -- for re-use; must be in [1..nnz_max]
end record;
```

```
type p_CRS_matrix is access CRS_matrix;
```

```
type t_precond is (none, diag, tridiag);
```

---

```
-- Matrix-vector multiplication --
```

---

```
-- w:= A*u
```

```
procedure Mult(A: in out CRS_matrix; u: vector; w: in out vector);
```

```

-- operator version - warning: uses stack, not for large/fast usage !
-- function "*" (A: CRS_matrix; u: vector) return vector;

-- Put/Add/Get data into/in/from matrix --

procedure Put(A: in out CRS_matrix; i,j: index; value: real);
procedure Add(A: in out CRS_matrix; i,j: index; value: real);
function Get(A: in CRS_matrix; i,j: index) return real;

-- pragma Inline(Put, Add);

position_not_found_in_sparse_matrix: exception;

-- Iterative solving of Ax=b symmetric system by conjugate gradient method --

procedure CG (A : in out CRS_matrix; -- (not "out" - GNAT optm pblm)
 b : vector;
 x : in out vector; -- input: 1st approx;
 -- output: solution of Ax=b
 eps: real; -- tolerance
 precondition: t_precond; -- kind of preconditioning
 precd, precl: vector; -- diag. & lower diag. of precondition.
 itmax: index; -- maximum number of iterations
 ite: out index -- last iteration
);

-- Stabilized biconjugate gradient method (A non symmetric) --

procedure BiCGStab (A : in out CRS_matrix; -- (not "out" - GNAT optm pblm)
 b : vector;
 x : in out vector; -- input: 1st approx;
 -- output: solution of Ax=b
 eps, tol: real; -- tolerance
 precondition: t_precond; -- kind of preconditioning
 precd, precl: vector; -- diag. & lower diag. of precondition.
 itmax: index; -- maximum number of iterations
 ite: out index -- last iteration
);

end Sparse;

Les routines de « bas niveau » d'algèbre linéaire sont, au choix, dans les modules SparseB0 (sans appel à la bibliothèque BLAS) ou SparseB1 (qui y fait appel). Aussi leur interface est-elle exactement la même.

-- Linear Algebra routines for Sparse package, NOT mapped to BLAS library

generic
 type real is digits <>;
 type index is range <>;
 type vector is array(index range <>) of real;

```

```

package SparseB0 is
 procedure Copy(u: in vector; v: out vector);
 function "*" (u,v: vector) return real;
 procedure Add_scaled(factor: real; u: in vector; v: in out vector);
 procedure Scale(factor: real; u: in out vector);
end SparseB0;

```

*-- Linear Algebra routines for Sparse package, mapped to BLAS library*

```

generic
 type real is digits <>;
 type index is range <>;
 type vector is array(index range <>) of real;

```

```

package SparseB1 is
 procedure Copy(u: in vector; v: out vector);
 function "*" (u,v: vector) return real;
 procedure Add_scaled(factor: real; u: in vector; v: in out vector);
 procedure Scale(factor: real; u: in out vector);
end SparseB1;

```

SparseB1 utilise les routines BLAS suivantes :

```

procedure scopy(n: natural; x: in vector; incx: integer;
procedure dcopy(n: natural; x: in vector; incx: integer;
function sdot(n: natural; x: in vector; incx: integer;
function ddot(n: natural; x: in vector; incx: integer;
procedure saxpy(n: natural; a: real; x: in vector; incx: integer;
procedure daxpy(n: natural; a: real; x: in vector; incx: integer;
procedure sscal(n: natural; a: real; x: in out vector; incx: integer);
procedure dscal(n: natural; a: real; x: in out vector; incx: integer);

```

## B.4 Maillage et réduction de largeur de bande

Le module BandRedu exporte l'algorithme de Collin [12] qui renumérote les noeuds de manière à réduire la largeur de bande. Pour deux noeuds  $i$  et  $j$ , l'élément  $a_{ij}$  d'une matrice de masse ou de rigidité ne sera non-nul que si ces noeuds sont dans un même élément. On a donc intérêt à ce que les indices  $i$  et  $j$  soient proches.

---

```

-- File : bandredu.ada
-- Description : Bandwidth reduction for finite elements
-- Date / Version : 24.XI.1998 / 8.I.1998 / 5 VII 1997
-- Author : gautier.demontmollin@unine.ch

```

---

```

with Matrices; use Matrices;

```

```

package BandRedu is

```

```

 procedure Bandwidth_reduction(
 nn, -- number of nodes IN
 nb_elem, -- number of elements
 npe, -- number of nodes per element

```

```

mxnj : in natural; -- dimension in space
x : in out matrix; -- node coordinates IN OUT
elem : in out imatrix; -- elements topology
nhbo, : -- original bandwidth OUT
nhbn : in out natural; -- new bandwidth
jnt, nnjt : in out ivector); -- node numbering new <--> old

-- the array *jnt* contains the new node numbers,
-- their location in the array being equal to the old node numbers
-- the array *nnjt* contains the old node numbers,
-- their locations in the array being equal to the new node numbers

node_unassigned: exception;
end BandRedu;

```

Le programme Mailleur comporte les sous-programmes suivants :

```

function Find_node_number(x: vector; res: lf) return natural is
procedure Trapeze(s,t: lf; x,y: out lf) is
function Noeud(s,t,z: lf) return natural is -- pour constr. elem.
procedure Construction_noeuds is... -- pour 1 macro-element
procedure Construction_elements is... -- pour 1 macro-element
function Diam_max_elem return lf is... -- h= diametre maximal d'un element
function Cle(s,mot: string) return boolean is...
procedure Lire_et_construire is...
procedure Reduire_largeur_bande is...
procedure Sauver is...
procedure Verifier is...

```

## B.5 Programme FT

Pour gérer les conditions initiales et aux limites, les tenseurs et les champs de vitesse, il y a le module CodeCond.

---

```

-- File : codecond.ada
-- Description : Codes for boundary conditions with finite elements
-- Date / Version : 24-Aug-2000 / 25-apr-2000 / 7.IV.1999 / 25 XI 1997
-- Author : Gautier.deMontmollin@UniNe.CH

```

---

```

with FEK; use FEK;
with FEFLOW; use FEFLOW;
with Matrices; use Matrices;
with Long_Float_Math_Lib; use Long_Float_Math_Lib;

package CodeCond is

 type condcode is (free, dirichlet, neumann);

 -- Values of boundary conditions:

 function Dirichlet(code: positive; x,y,z,t: long_float) return long_float;
 function Neumann(code: positive; x,y,z,t: long_float) return long_float;

 -- L'<<ancienne>> methode, 1 code par noeud:

```

```

procedure BC_sides_to_nodes(BC_sides: imatrix; elem: imatrix; nel: natural;
 BC_code: out ivector; gelem: t_element);
 -- entries for
 -- BC_sides: code(side, element)
 -- elem: node(elementary node, element)

procedure Compute_BC_values(BC_node_code: ivector; x: matrix;
 node_num: positive;
 refresh_unsteady_only: boolean;
 time: long_float;
 BC_value: out vector);

-- 28.III.1998: codes multiples: des noeuds peuvent etre a la frontiere
-- entre des conditions tres differentes.

type cubecode is array(1..6) of integer;
type cubeval is array(1..6) of long_float;

type BC_code_pack is record
 dirichlet: integer; --> 1 code par noeud
 neumann: cubecode; --> le prolongement continu de n et de la CL
end record; -- sur le noeud dependent de la face !

type BC_value_pack is record
 dirichlet: long_float;
 neumann: cubeval;
end record;

type t_BC_code is array(positive range <>) of BC_code_pack;
type p_BC_code is access t_BC_code;

type t_BC_value is array(positive range <>) of BC_value_pack;
type p_BC_value is access t_BC_value;

procedure BC_sides_to_nodes(BC_sides: imatrix; elem: imatrix; nel: natural;
 BC_code: out t_BC_code; gelem: t_element);
 -- entries for
 -- BC_sides: code(side, element)
 -- elem: node(elementary node, element)

procedure Compute_BC_values(BC_node_code: t_BC_code; x: matrix;
 node_num: positive;
 refresh_unsteady_only: boolean;
 time: long_float;
 BC_value: in out t_BC_value;
 gelem: t_element);

key_is_steady: constant:= 111; -- key code for "IC = sol. of steady problem"

undefined_dirichlet_code, undefined_neumann_code,
undefined_IC_code, undefined_source_code,
undefined_tensor_code, undefined_velocity_code,
bad_dimension,
IC_must_be_solved: exception;

-- Values of initial conditions: (11.III.1998 annule le 19 - par noeud/elem)

```

```

-- 18.X.1998: global codes for IC / source / tensor

function IC(code: natural; x,y,z: long_float) return long_float;

function Source(code: natural; p: vector) return long_float;

-- 4.XI.1998 : on passe le tenseur d'un coup
-- 25.IV.1999 : on passe un pointeur

procedure Tensor(code: natural; point: vector; m: out p_matrix);

procedure Velocity(code: natural; p: vector; v: out vector);

--| Special codes for velocity vector for NOT using "Velocity(...)" |--

key_vel_derived : constant:= 0; -- "velocity is derived from sol001"
key_vel_on_nodes : constant:= 333; -- "velocity is given by vel001"
key_vel_burgers : constant:= 555; -- "velocity = solution! (1D)" 28-Jun-2000

end CodeCond;

```

Pour ce qui est de l'évaluation des fonctions de base, de leurs dérivées et du jacobien du changement de coordonnées local vers global, c'est le module FEK qui le fait. Le principe est très simple : pour chaque procédure d'évaluation, on donne en entrée les coordonnées des nœuds de l'élément  $\hat{K}$ , selon une numérotation convenue faite sur l'élément modèle  $K$ , et un point en coordonnées locales dans  $K$ . Au retour, on a la valeur des fonctions de base  $\psi_j$  associées aux nœuds, ainsi que, entre autres, leur gradient  $\nabla\psi_j$  et le déterminant du jacobien de la transformation  $K \rightarrow \hat{K}$  également évalué au point donné.

---

```

-- File : fek.ada
-- Description : Finite Elements Kernel (extracts)
-- Date / Version : 25-Apr-2000 / 13.I.1999 / 15.X.1998 / 25.VI.1997
-- Author : M. Bercovier, Jerusalem
-- Ada version by G. de Montmollin (gdemont@hotmail.com)

```

---

```

with Matrices; use Matrices;

package FEK is

 type t_element is (L2, Q4, Q9, B27);

 nodes_per_element: constant array(t_element) of positive:=
 (L2=> 2, Q4=> 4, Q9=> 9, B27=> 27);

 spatial_dimension: constant array(t_element) of positive:=
 (L2=> 1, Q4=> 2, Q9=> 2, B27=> 3);

 -- Pour elements reguliers:

 faces_per_element: constant array(t_element) of natural:=
 (L2=> 2, Q4=> 4, Q9=> 4, B27=> 6);

 nodes_per_face: constant array(t_element) of natural:=

```

```

(L2=> 1, Q4=> 2, Q9=> 3, B27=> 9);

----- Conversions de numeros de faces vers numeros d'elements -----

type t_face_cnv is array(positive range <>, positive range <>) of positive;

-- 1D linear to 0D pointwise vertices numbering conversion:

P1_to_L2: constant t_face_cnv(1..2,1..1):= ((1=> 1),(1=> 2));

-- 2D linear to 1D linear vertices numbering conversion:

L2_to_Q4: constant t_face_cnv(1..4,1..2):=
 ((1, 2), -- face 1: y=0
 (2, 3), -- face 2: x=1
 (3, 4), -- face 3: y=1
 (4, 1)); -- face 4: x=0

-- 2D parabolic to 1D parabolic vertices numbering conversion:

L3_to_Q9: constant t_face_cnv(1..4,1..3):=
 ((1, 5, 2), -- face 1: y=0
 (2, 6, 3), -- face 2: x=1
 (3, 7, 4), -- face 3: y=1
 (4, 8, 1)); -- face 4: x=0

-- 3D parabolic to 2D parabolic vertices numbering conversion:

Q9_to_B27: constant t_face_cnv(1..6,1..9):=
 ((1, 2, 6, 5, 9,18,13,17,25), -- face 1: y=0
 (2, 3, 7, 6,10,19,14,18,22), -- face 2: x=1
 (3, 4, 8, 7,11,20,15,19,26), -- face 3: y=1
 (4, 1, 5, 8,12,17,16,20,24), -- face 4: x=0
 (3, 2, 1, 4,10, 9,12,11,21), -- face 5: z=0 -- orient. corr. 24.X.1997
 (5, 6, 7, 8,13,14,15,16,23)); -- face 6: z=1

type p_face_cnv is access t_face_cnv;
pn_P1_to_L2: constant p_face_cnv:= New t_face_cnv ' (P1_to_L2);
pn_L2_to_Q4: constant p_face_cnv:= New t_face_cnv ' (L2_to_Q4);
pn_L3_to_Q9: constant p_face_cnv:= New t_face_cnv ' (L3_to_Q9);
pn_Q9_to_B27: constant p_face_cnv:= New t_face_cnv ' (Q9_to_B27);

-- One array for all conversions (pointers)

to_face_conv: constant array(t_element) of p_face_cnv:=
 (L2=> pn_P1_to_L2,
 Q4=> pn_L2_to_Q4,
 Q9=> pn_L3_to_Q9,
 B27=> pn_Q9_to_B27);

type shape_opt is (loc_der, loc_der_jac, loc_der_jac_glob_der);

subtype l2_vector is vector(1..2);
subtype l2_grad is matrix(1..2, 1..1);

subtype q4_vector is vector(1..4);
subtype q4_grad is matrix(1..4, 1..2);

subtype q9_vector is vector(1..9);

```

```

subtype q9_grad is matrix(1..9, 1..2);

subtype b27_vector is vector(1..27);
subtype b27_grad is matrix(1..27, 1..3);

permu_q9: constant array(1..3,1..3) of positive:=
 ((1, 8, 4), (5, 9, 7), (2, 6, 3));

permu27: constant array(1..3,1..3,1..3) of positive:=
 (((1, 17, 5), (12, 24, 16), (4, 20, 8)),
 ((9, 25, 13), (21, 27, 23), (11, 26, 15)),
 ((2, 18, 6), (10, 22, 14), (3, 19, 7)));

l_coord: constant array(1..2) of long_float:= (-1.0, 1.0);

q_coord: constant array(1..9,1..2) of long_float:=
 ((-1.0,-1.0), (1.0,-1.0), (1.0, 1.0), (-1.0, 1.0),
 (0.0,-1.0), (1.0, 0.0), (0.0, 1.0), (-1.0, 0.0),
 (0.0, 0.0));

b_coord: constant array(1..27,1..3) of long_float:=
 ((-1.0,-1.0,-1.0), (1.0,-1.0,-1.0), (1.0, 1.0,-1.0), (-1.0, 1.0,-1.0),
 (-1.0,-1.0, 1.0), (1.0,-1.0, 1.0), (1.0, 1.0, 1.0), (-1.0, 1.0, 1.0),
 (0.0,-1.0,-1.0), (1.0, 0.0,-1.0), (0.0, 1.0,-1.0), (-1.0, 0.0,-1.0),
 (0.0,-1.0, 1.0), (1.0, 0.0, 1.0), (0.0, 1.0, 1.0), (-1.0, 0.0, 1.0),
 (-1.0,-1.0, 0.0), (1.0,-1.0, 0.0), (1.0, 1.0, 0.0), (-1.0, 1.0, 0.0),
 (0.0, 0.0,-1.0), (1.0, 0.0, 0.0), (0.0, 0.0, 1.0), (-1.0, 0.0, 0.0),
 (0.0,-1.0, 0.0), (0.0, 1.0, 0.0), (0.0, 0.0, 0.0));

```

---

```

-- Shape functions --

```

---



---

```

-- L2: line element, in one dimension. --
-- linear shape functions. --

```

---

```

-- NOTE, GdM Apr-2000: The original FEK handles the element
-- in 2 dimensions with a wrong determinant (a SQRT too much)
-- but correct global derivatives using it!

```

```

procedure l2shap (x : l2_vector; -- x(i) = coordinate of node i
 s : long_float; -- s = some local point
 shape : out l2_vector; -- shape fcn value at s
 dadl,dadg : in out l2_grad; -- local and global gradients
 det : out long_float; -- jacobian of transformation
 eps : long_float); -- 0 det checking criterion

```

```

procedure l2shap (s: long_float; shape: out l2_vector);

```

---

```

-- Q4: quadrilateral element, in two dimensions. --
-- linear shape functions. --

```

---

```

procedure q4shap (x1,x2 : q4_vector; -- (x1(i),x2(i)) = node i
 s,t : long_float; -- (s,t) = some local point
 shape : out q4_vector; -- shape fcn value at (s,t)

```

```

dadl,dadg : in out q4_grad; -- local and global gradients
det : out long_float; -- jacobian of transformation
calc_shape : boolean; -- calculate shape values ?
opt : shape_opt; -- options
eps : long_float); -- 0 det checking criterion

procedure q4shap (s,t: long_float; shape: out q4_vector);

-- Q9: quadrilateral element, in two dimensions. --
-- parabolic shape functions. --

procedure q9shap (x1,x2 : q9_vector; -- (x1(i),x2(i)) = node i
s,t : long_float; -- (s,t) = some local point
shape : out q9_vector; -- shape fcn value at (s,t)
dadl,dadg : in out q9_grad; -- local and global gradients
det : out long_float; -- jacobian of transformation
calc_shape : boolean; -- calculate shape values ?
opt : shape_opt; -- options
eps : long_float); -- 0 det checking criterion

procedure q9shap (s,t: long_float; shape: out q9_vector);

-- B27: quadrilateral element, in three dimensions. --
-- parabolic shape functions. --

procedure b27shp (x1,x2,x3 : b27_vector; -- (x1(i),x2(i),x3(i)) = node i
r,s,t : long_float; -- (r,s,t) = some local point
shape : out b27_vector; -- shape fcn value at (r,s,t)
dadl,dadg : in out b27_grad; -- local and global gradients
det : out long_float; -- jacobian of transformation
calc_shape : boolean; -- calculate shape values ?
opt : shape_opt; -- options
eps : long_float); -- 0 det checking criterion

procedure b27shp (r,s,t: long_float; shape: out b27_vector);

zero_jacobian, negative_jacobian: exception;

-- Translated from Fortran: marked with *

-- * SUBROUTINE L2SHAP(X1,X2,S,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE L3SHAP(X1,X2,S,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE T3SHAP(X1,X2,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE T6SHAP(X1,X2,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- * SUBROUTINE Q4SHAP(X1,X2,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE Q8SHAP(X1,X2,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- * SUBROUTINE Q9SHAP(X1,X2,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE Q49SHP(X1,X2,NOD,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE TH4SHP(X1,X2,X3,R,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE TH10SH(X1,X2,X3,R,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- * SUBROUTINE B27SHP(X1,X2,X3,R,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE B821SP(X1,X2,X3,NOD,R,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE B8SHAP(X1,X2,X3,R,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE PR18SP(X1,X2,X3,R,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)
-- SUBROUTINE PR6SHP(X1,X2,X3,R,S,T,MNDP,SHAPE,DADL,DADG,DET,IOPT,EPS,IERR)

```

end FEK;

Quelques constantes, notamment liées au problème introductif, se trouvent dans le petit module suivant :

---

```
-- File : FEFLOW.ada
-- Description : Constants in FEFLOW benchmarks
-- Date / Version : 11.III.1998 / first : 11.III.1998
-- Author : Gautier.deMontmollin@Maths.UniNe.CH
```

---

**package FEFLOW is**

-- Constants for Theis problem, FEFLOW, ch. 5 Benchmarks & ex. 5-4, table 5.2

**package Theis is**

```
 r1: constant := 0.304; -- radius of the well [m]
 r2: constant := 304.0; -- external radius [m]
 Q: constant := 0.0141584; -- well pumping rate [m^3/s] (~1.4 litre/sec)
 S: constant := 0.001; -- storage coefficient S
 T: constant := 9.2903e-4; -- transmissivity T (equiv to K)
end Theis;
```

-- Constants for 2-well problem, FEFLOW, ch. 5 Benchmarks & ex. 5-43, table 5.14

**package Well2 is**

```
 r1: constant := 0.005; -- radius of a well [m]
 r2: constant := 1.45; -- external radius [m]
 d: constant := 0.305; -- distance between centres
 Q: constant := -7.45e-5; -- well injection rate [m^3/s]
 T: constant := 1.0e-4; -- transmissivity T (equiv to K)
end Well2;
```

end FEFLOW;

Venons-en au programme Flux-Transport. Le programme FT comporte de nombreux sous-programmes d'importance variable. Les routines importantes sont décrites dans l'annexe C. Ci-dessous, pour donner une idée, on a les noms des sous-programmes de FT. En réalité, tout est englobé dans la procédure `FT_elem` dans laquelle sont définis tous les sous-types de vecteurs et gradients dépendant du genre d'élément du problème. Cela permet de ne pas passer en paramètre des valeurs courantes comme le nombre de dimensions, le nombre de noeuds par élément, etc., ce qui améliore la performance et la lisibilité du programme.

```
procedure FT_elem(gelem: t_element) is
 procedure Controle_CL_Dirichlet is...
 procedure Controle_CL_Neumann is...
 procedure Donnees is...
 procedure Local_to_Global...
 procedure Global_to_local...
 function In_trapezoidal_element(el: positive; gp: sp_vector) return boolean
 is...
 function In_element(el: positive; gp: sp_vector) return boolean is...
 procedure Derive_vitesse...
 procedure Interpole_Vitesse...
 procedure Interpole_Vitesse_Burgers...
 procedure Vitesse_filtration...
 procedure Attribuer_elements_aux_particules is...
```

```

procedure Attribuer_elt_a_1_particule(ppp: p_particule; ad: out boolean)
 is...
procedure Reclassement_particules is... -- <<garbage collection>>
procedure Creer_particule...
procedure Particules_initiales is...
procedure Calcul_diametres is...
procedure Calcul_tables_visibilite is...
procedure Calcul_table_adjacence is...
procedure Calcul_conditions_initiales is...
procedure Numerotation_equations is...
procedure Struc_matr_bande is...
procedure Struc_matr_creuse is...
procedure Indices_matr_creuse...
procedure Indices_matr_AC_BC...
procedure Indices_matr_A2C_B2C...
procedure Struc_matr is...
procedure Assma_bande...
procedure Assma_creux...
procedure Assma...
procedure Assma_Multibloc...
procedure Ematr...
procedure Remplir_matrices...
procedure Decomp_A(eq: genre_equation) is ...
procedure Rempl_et_decomp(eq: genre_equation; stationn: boolean) is ...
procedure Etrac_L2...
procedure Etrac_Q9...
procedure Etrac_Q4...
procedure Etrac_affine_B27...
procedure Etrac_B27...
procedure Elifix...
procedure Secmem(c: in out vector; t:FT_real; eq: genre_equation) is...
procedure Second_membre_système_totalemt_discretise...
procedure Solve(eq: genre_equation; sk: p_vector) is...
procedure Ecrire_vitesses_filtration is...
procedure Resoudre_flux_stationnaire is...
procedure Macro_pas_c_convection is...
function Concentration_integrale return FT_real is
procedure Injection_particules is... -- par la frontiere influx
procedure Injecte_selon_CL(el,nd: positive; pt_loc: sp_vector) is...
procedure Resolution_equation_flux is...
procedure Resolution_equation_transport is...

```

La résolution de l'équation non-linéaire (§6.1) par STILS non-linéarisé est faite dans un programme spécial, *STILBurg*, qui comporte les sous-programmes suivants et dont le listage complet est en annexe C.

```

procedure STILBurg is
 function alpha(i,j: integer) return real is...
 function beta(i,j,l: integer) return real is...
 function gamma(i,j,k: integer) return real is...
 function epsilon(i,j,k,l: integer) return real is...
 function fx(i:integer) return real is...
 function Dfx(i,m: integer) return real is...
 procedure LR(a,b,c: in out vector) is...
 procedure Solve_LR(m,l,r,d: in vector; x: in out vector) is...
 function Norme2(x:vector) return real is...

```

## Annexe C

# Routines importantes de FT

C'est maintenant le tour du contenu « utile » du programme FT

### C.1 Construction des matrices élémentaires

Le principe est, rappelons-le, de parcourir tous les éléments, de calculer les matrices pour chaque élément parcouru, puis d'ajouter ces matrices « élémentaires » dans les matrices « globales ». La linéarité du produit scalaire dans  $L^2$  fait qu'au bout du parcours, les matrices « globales » ont le contenu escompté. Voir plus loin la partie consacrée au remplissage même de ces matrices.

---

--| *Construction des matrices elementaires et du second membre elementaire* |--

---

```
procedure Ematr(
 Me, Re, -- Matrices elementaires (M,R,M3,M4,M5,M6)
 Me3, Me4, Me5, Me6,
 Me7, Me8, Me9, Me10,
 Me11, Me12: in out el_matrix;
 ce : in out el_array; -- Second membre elementaire (c)
 equa_el : el_iarray; -- Vect. elem. des num d'equations
 nde : el_iarray; -- Numeros des noeuds de l'element
 xel : sp_el_array; -- Coordonnees des noeuds de l'elem.
 fl : el_array; -- Force aux noeuds
 calc_second: boolean;
 eq: genre_equation
) is

 delt1: constant FT_real:= diamax * 0.7071; -- facteur du terme streamline * 0.6
 delt2: constant FT_real:= diamax * 0.8; -- facteur du terme DC

 psi : el_array; -- fonctions de base attachees aux noeuds
 stream_psi : el_array; -- termes streamline (V|grad psi) supplementaires
 dc_psi : el_array; -- termes DC supplementaires

 grad_c : sp_vector; -- Gradient de c au point de Gauss (pour DC)
 V_proj : sp_vector; -- projection de V sur grad c = v//

 dl,dg: el_grad;
 det,pds,q,wd, spspm, spm, spv, fact_mij : FT_real;
```

```

norme_2_grad_c, facteur_V_proj : FT_real;
psi_i_corr: FT_real;

loc_tens, loc_K: p_matrix; -- pointeurs sur tenseurs locaux
point_milieu, pgauss: sp_vector;
V: sp_vector; -- vitesse de filtration

begin
 case gelem is
 when L2 => Local_to_global(xel, (1=> 0.0), point_milieu);
 when Q4 => Local_to_global(xel, (0.0,0.0), point_milieu);
 when Q9 => point_milieu:= (xel(1)(iel), xel(2)(iel));
 when B27=> point_milieu:= (xel(1)(iel), xel(2)(iel), xel(3)(iel));
 end case;
 case eq is
 when flux => fact_mij:= Ss;
 when trans => fact_mij:= omega;
 end case;

 -- Mise a Zero !

 -- ce:= (others => 0.0);
 for i in ce'range loop ce(i):= 0.0; end loop; -- 16-Apr-2001 (perf)

 -- Re:= (others => (others => 0.0));
 -- Me:= Re;
 -- Me3:= Re; Me4:= Re; Me5:= Re; Me6:= Re;
 -- Me7:= Re; Me8:= Re; Me9:= Re; Me10:= Re; Me11:= Re; Me12:= Re;

 for i in Re'range(1) loop -- 16-Apr-2001 (perf)
 for j in Re'range(2) loop
 Re(i,j) := 0.0; Me(i,j) := 0.0; Me3(i,j) := 0.0;
 Me4(i,j) := 0.0; Me5(i,j) := 0.0; Me6(i,j) := 0.0;
 Me7(i,j) := 0.0; Me8(i,j) := 0.0; Me9(i,j) := 0.0;
 Me10(i,j):= 0.0; Me11(i,j):= 0.0; Me12(i,j):= 0.0;
 end loop;
 end loop;

 -- Tenseur de permeabilite et/ou de diffusivit\'e au pt milieu
 Tensor(code_tenseur(flux), point_milieu, loc_K);
 Tensor(code_tenseur(eq), point_milieu, loc_tens);

 -- Integration par Gauss

 for i1 in 1..3 loop -- triple somme (pour triple integrale)
 for i2 in 1..3 loop
 for i3 in 1..3 loop
-- fonctions de formes, derivees locales et globales, jacobien
-- au point de gauss dans l'element de reference.

 case gelem is
 when L2 =>
 pds:= poids(i1);
 FEK.l2shap(xel(1),gauss(i1),psi, dl,dg,det,FEK_eps);

 when Q4 =>
 pds:= poids(i1)*poids(i2);
 FEK.q4shap(xel(1),xel(2),gauss(i1),gauss(i2),psi,
 dl,dg,det,true,loc_der_jac_glob_der,FEK_eps);

```

```

when Q9 =>
 pds:= poids(i1)*poids(i2);
 FEK.q9shap(xel(1),xel(2),gauss(i1),gauss(i2),psi,
 dl,dg,det,true,loc_der_jac_glob_der,FEK_eps);
when B27 =>
 pds:= poids(i1)*poids(i2)*poids(i3);
 FEK.b27shp(xel(1),xel(2),xel(3),gauss(i1),gauss(i2),gauss(i3),psi,
 dl,dg,det,true,loc_der_jac_glob_der,FEK_eps);
end case;
if eq=trans then
 case code_vitesse(trans) is
 when key_vel_derived => Derive_vitesse(nde, loc_K.all, dg, 1.0, V);
 when key_vel_on_nodes => Interpole_Vitesse(nde, psi, 1.0, V);
 when key_vel_burgers |
 key_vel_burgers_special => Interpole_Vitesse_Burgers(nde, psi, V);
 when others =>
 case nsd is
 when 1=>
 Local_to_global(xel, (1=>gauss(i1)), pgauss);
 when 2=>
 Local_to_global(xel, (1=>gauss(i1),2=>gauss(i2)), pgauss);
 when 3=>
 Local_to_global(xel,
 (1=>gauss(i1),2=>gauss(i2),3=>gauss(i3)), pgauss);
 when others =>
 raise cas_imprevu;
 end case;
 Velocity(code_vitesse(trans), pgauss, V);
 end case;
end if;

wd:= pds*det;

if calc_second then
 q:= 0.0;
 if eq=flux then
 for l in 1..iel loop
 q:= q + fl(l)*psi(l);
 end loop;
 q:= wd*q;
 end if;
end if;

-- Calcul de la fonction-charge élémentaire q
-- au point de Gauss par interpolation avec les
-- fonctions de base. Autre point de vue :
-- on projette la fonction-charge sur H_h
-- et on calcule la valeur du projeté
-- au point de Gauss

if eq=trans then
 -- Calcul du terme <<stream_psi>> $(v|\nabla\psi_k)$ (2.9)
 case meth_trans is
 when C_Convection | Galer_kine => null;
 -- Terme nul ou inutile pour ces méthodes
 when others =>
 for i in 1..iel loop
 stream_psi(i):= 0.0;
 for l in 1..nsd loop
 stream_psi(i):= stream_psi(i) + dg(i,l) * V(l);
 end loop;
 end loop;
 end case; -- meth_trans

 -- Calcul du terme <<dc_psi>> $(v_{//}|\nabla\psi_k)$ (2.9)
 if meth_trans = SUPG_DC then

```

```

-- (27) de Hughes, Mallet,... [29]

-- Interpolation de grad c (concentration actuelle)
-- au point de Gauss

grad_c:= (others=> 0.0);
for el in 1..iel loop
 for l in 1..nsd loop
 grad_c(l):= grad_c(l) + dg(el,l) * co(nde(el));
 end loop;
end loop; -- el

norme_2_grad_c:= square_norm(grad_c);

-- Calcul V_proj i.e. $v_{//}$ définie dans (2.8)

if norme_2_grad_c < FT_real'epsilon then
 V_proj:= (others=> 0.0);
else
 facteur_V_proj:= 0.0;
 for l in 1..nsd loop -- prod. scal. au numérateur
 facteur_V_proj:= facteur_V_proj + grad_c(l) * V(l);
 end loop;
 facteur_V_proj:= facteur_V_proj / norme_2_grad_c; -- dénominateur
 for l in 1..nsd loop
 V_proj(l):= grad_c(l) * facteur_V_proj;
 end loop;
end if;

for el in 1..iel loop -- c.f. (30) de [29]
 dc_psi(el):= 0.0;
 for l in 1..nsd loop -- encore un prod. scal.
 dc_psi(el):= dc_psi(el) + V_proj(l) * dg(el,l);
 end loop; -- l
end loop; -- el

end if; -- DC
end if; -- eq=trans

-- Boucles (i,j) pour remplir les matrices élémentaires
-- au point de Gauss

for i in 1..iel loop
 if eq=trans then -- Modification de la fonction-test de base ψ_i
 case meth_trans is

 when Streamline => -- formule (2.5)
 psi_i_corr:= psi(i) + delt1 * stream_psi(i);

 when SUPG_DC => -- formule (2.9)
 psi_i_corr:= psi(i) + delt1 * stream_psi(i)
 + delt2 * dc_psi(i); -- Contrib v//

 when SILS => -- formule (3.6)
 psi_i_corr:= stream_psi(i);

 when others =>
 psi_i_corr:= psi(i);
 end case;
 else
 psi_i_corr:= psi(i); -- equation diffu. -> tjrs Galerkin normal
 end if;
end loop;

```

```

end if;

for j in 1..iel loop
 spspm:= 0.0; -- somme prod. scal. et matr. ($K\nabla\psi_j|\nabla\psi_i$)
 spv:= 0.0; -->> (10.XI.1998) somme pr. sc. ($\nabla\psi_j|v$)
 for l in 1..nsd loop
 spm:= 0.0; -->> somme prod. matr. ($T\nabla\psi_j$)l, avec $T = K$ ou D
 for m in 1..nsd loop
 spm:= spm+ loc_tens(l,m) * dg(j,m);
 end loop;

 case eq is
 when flux => null;
 -- when trans => spm:= spm - V(l)*psi(j); -- terme hyperbolique
 -- v. Johnson 10.XI.1998
 when trans => spv:= spv + V(l)*dg(j,l); -- terme hyperbolique
 end case;
 spspm:= spspm + dg(i,l) * eta(l) * spm; -- terme parabolique
 end loop; -- l

 if eq = trans and Space_time_method(meth_trans) then -- E-T !
 declare
 al_ij, bt_ij, ga_ij, de_ij,
 tau_al_ij, iv_tau_de_ij,
 eta_ij, xi_ij, dxu : FT_real;
 -- Termes pour composantes de matrices locales
 begin

 al_ij:= stream_psi(j) * stream_psi(i); -- ($\nabla\psi_j|v$) ($\nabla\psi_i|v$)
 bt_ij:= stream_psi(j) * psi(i); -- ($\nabla\psi_j|v$) ψ_i
 ga_ij:= psi(j) * stream_psi(i); -- ψ_j ($\nabla\psi_i|v$)
 de_ij:= psi(j) * psi(i); -- $\psi_j \psi_i$

 tau_al_ij:= al_ij * tau ;
 iv_tau_de_ij:= de_ij * iv_tau;

 case meth_trans is

 when STILS_TMQ1 =>

 -- Ici M est la matrice multipliant c^{k+1} (implicite) "M=A"
 -- R est la matrice multipliant c^k (explicite) "R=B"
 -- NB : il faut aussi M pour 2nd membre (CL Dirich.) 19.VIII.1999

 Me(i,j):= Me(i,j) + wd *
 ((1.0/3.0)*tau_al_ij + 0.5*bt_ij + 0.5*ga_ij + iv_tau_de_ij);

 Re(i,j):= Re(i,j) + wd *
 ((-1.0/6.0)*tau_al_ij - 0.5*bt_ij + 0.5*ga_ij + iv_tau_de_ij);

 if code_vitesse(trans) = key_vel_burgers_special then
 -- Burgers semi-linéarisé, (6.20)
 -- Att: un peu cavalier : induit changement de methode 15-dec-2000
 dxu:= Interpole_Burgers_non_lin(nde, dg);

 eta_ij:= de_ij * dxu; -- $\partial_x u^* \psi_j \psi_i$
 xi_ij := bt_ij * V(1) * dxu; -- $u^* \partial_x u^* \psi_j \psi_i$

 Me(i,j):= Me(i,j) + wd * (0.5 * eta_ij + (tau/3.0) * xi_ij);
 end if;
 end case;
 end begin;
end if;

```

```

 Re(i,j):= Re(i,j) + wd * (0.5 * eta-ij + (-tau/6.0) * xi-ij);
end if;

-- Autre equation (pour eta(0)=1)

-- Me(i,j):= Me(i,j) + wd *
-- ((-1.0/6.0)*tau-al-ij + 0.5*bt-ij - 0.5*ga-ij + iv-tau-de-ij);
--
-- Re(i,j):= Re(i,j) + wd *
-- ((1.0/3.0)*tau-al-ij - 0.5*bt-ij - 0.5*ga-ij + iv-tau-de-ij);
--
when STILS-TMQ1S2 => -- Methode a deux pas: oscille!

 Me(i,j):= Me(i,j) + wd *
 ((-1.0/6.0)*tau-al-ij + 0.5*bt-ij - 0.5*ga-ij + iv-tau-de-ij);

 Re(i,j):= Re(i,j) + wd *
 ((2.0/3.0)*tau-al-ij + 2.0 * iv-tau-de-ij);

 Me3(i,j):= Me3(i,j) + wd *
 ((1.0/6.0)*tau-al-ij + 0.5*bt-ij - 0.5*ga-ij - iv-tau-de-ij);

when STILS-TMQ2 => -- formules (4.4) et (4.5)

 Me(i,j):= Me(i,j) + wd *
 ((1.0/15.0) * tau-al-ij + (2.0/ 3.0) * bt-ij +
 (-2.0/ 3.0) * ga-ij + (-8.0/ 3.0) * iv-tau-de-ij);

 Re(i,j):= Re(i,j) + wd *
 ((8.0/15.0) * tau-al-ij + (16.0/ 3.0) * iv-tau-de-ij);

 Me3(i,j):= Me3(i,j) + wd *
 ((-1.0/30.0) * tau-al-ij +
 (-1.0/ 6.0) * bt-ij +
 (1.0/ 6.0) * ga-ij +
 (1.0/ 3.0) * iv-tau-de-ij);

 Me4(i,j):= Me4(i,j) + wd *
 ((2.0/15.0) * tau-al-ij +
 0.5 * bt-ij +
 0.5 * ga-ij +
 (7.0/ 3.0) * iv-tau-de-ij);

when STILS-TMQ3 => -- formules (4.12) .. (4.14)

 Me(i,j):= Me(i,j) + wd *
 ((33.0/ 560.0) * tau-al-ij + (57.0/ 80.0) * bt-ij +
 (-57.0/ 80.0) * ga-ij + (-189.0/ 40.0) * iv-tau-de-ij);

 Re(i,j):= Re(i,j) + wd *
 ((27.0/ 70.0) * tau-al-ij + (54.0/ 5.0) * iv-tau-de-ij);

 Me3(i,j):= Me3(i,j) + wd *
 ((-27.0/ 560.0) * tau-al-ij + (-81.0/ 80.0) * bt-ij +
 (81.0/ 80.0) * ga-ij + (-297.0/ 40.0) * iv-tau-de-ij);

 Me4(i,j):= Me4(i,j) + wd *
 ((-3.0/ 140.0) * tau-al-ij + (3.0/ 10.0) * bt-ij +

```

```

 (-3.0/ 10.0) * ga_ij + (27.0/ 20.0) * iv_tau_de_ij);

Me5(i,j):= Me5(i,j) + wd *
 ((19.0/1680.0) * tau_al_ij + (7.0/ 80.0) * bt_ij +
 (-7.0/ 80.0) * ga_ij + (-13.0/ 40.0) * iv_tau_de_ij);

Me6(i,j):= Me6(i,j) + wd *
 ((8.0/ 105.0) * tau_al_ij + 0.5 * bt_ij +
 0.5 * ga_ij + (37.0/ 10.0) * iv_tau_de_ij);

when STILS_TDQ1 =>

 -- A, 11
 Me(i,j) := Me(i,j) + wd * (al_ij);
 Re(i,j) := Re(i,j) + wd * (-tau * bt_ij - de_ij);
 -- A, 12
 Me3(i,j) := Me3(i,j) + wd * (de_ij);
 Me4(i,j) := Me4(i,j) + wd * ((-tau/6.0) * de_ij);

 -- B, 11
 Me6(i,j) := Me6(i,j) + wd * (tau * bt_ij - de_ij);
 -- B, 12
 Me8(i,j) := Me8(i,j) + wd * ((2.0*tau/3.0) * de_ij);
 Me9(i,j) := Me9(i,j) + wd * ((tau/6.0) * de_ij);

when Galer_TDQ1 =>
 Me(i,j):= Me(i,j) + wd * de_ij;
 Re(i,j):= Re(i,j) + wd * ((1.0/3.0) * tau*bt_ij + 0.5*de_ij);
 Me3(i,j):= Me3(i,j) + wd * ((1.0/6.0) * tau*bt_ij + 0.5*de_ij);
 Me4(i,j):= Me4(i,j) + wd * ((1.0/6.0) * tau*bt_ij - 0.5*de_ij);

when Galer_TMQ1 => -- formule (3.3)

 --"M=A" (implicite/c^k+1), "R=B" (explicite/c^k)
 Me(i,j):= Me(i,j) + wd * ((1.0/3.0) * tau*bt_ij + 0.5*de_ij);
 Re(i,j):= Re(i,j) + wd * ((-1.0/6.0) * tau*bt_ij + 0.5*de_ij);

when Strea_TMQ1 => -- formule (3.1)

 --"M=A" (implicite/c^k+1), "R=B" (explicite/c^k)
 -- contrib. streamline (multiplie par delt1) = STILS!
 Me(i,j):= Me(i,j) + wd *
 ((1.0/3.0) * tau*bt_ij + 0.5*de_ij + delt1 *
 ((1.0/3.0)*tau_al_ij + 0.5*bt_ij + 0.5*ga_ij + iv_tau_de_ij)
);
 Re(i,j):= Re(i,j) + wd *
 ((-1.0/6.0) * tau*bt_ij + 0.5*de_ij + delt1 *
 ((-1.0/6.0)*tau_al_ij - 0.5*bt_ij + 0.5*ga_ij + iv_tau_de_ij)
);

when others =>
 raise cas_imprevu; -- Space-time-method devrait etre couvert
end case;

end; -- declare

else -- Not space-time -> Galerkin, SD, SD-DC, SILS !

 -- spspm = (K grad psi_j | grad psi_i) : terme de diffusion

```

```

-- spv = (grad psi-j | V) : terme de transport
Re(i,j):= Re(i,j) + wd * (spspm + psi_i_corr * spv);
if not calc_second then
 -- m_ij=(f psi-j, psi-i)
 Me(i,j):= Me(i,j) + wd * fact_mij * psi(j) * psi_i_corr;
end if;

end if;

end loop; -- j

if calc_second and then equa_el(i) > 0 then
 -- Memorisation des forces exterieures dans le vecteur ce
 -- ** ce = contribution de l'element dans le vecteur c
 -- (=ck ou ckp1) **
 ce(i):= ce(i) - psi(i)*q;
end if;
end loop; -- i
exit when nsd < 3; -- il n'y a pas de dim 3
end loop; -- i3
exit when nsd < 2; -- il n'y a pas de dim 2
end loop; -- i2
end loop; -- i1
end Ematr;

```

## C.2 Assemblage des matrices globales

On rappelle que, quelque soit la méthode imaginée pour résoudre une EDP linéaire par éléments finis, on aboutit à un système linéaire de la forme  $Ax = Bd$ . Voici la partie du programme qui assemble les matrices « élémentaires » dans les matrices « globales »  $A$  et  $B$ . A noter que pour l'assemblage des matrices multi-blocs (pour les dimensions temporelles 2 ou plus), on a programmé l'algorithme d'assemblage de manière *générique* dans la procédure `g_Assma_Multibloc`, en faisant abstraction du type de stockage pour les matrices globales (format creux, format bande). Ceci permet d'avoir à créer et maintenir qu'une copie de l'algorithme. La concrétisation pour chaque format est faite en une simple déclaration, (sous-procédures `Assma_Multibloc_creux` et `Assma_Multibloc_bande` dans `Assma_Multibloc`.

---

```
--| Assembler les matrices M et R locales dans la matrice globale A ou B |--
```

---

```

procedure Assma_bande
 (MB : in out Band_matrix;
 Me,Re: el_matrix;
 equa_el: el_iarray;
 multi_M, multi_R: FT_real; -- multiplicateurs de M et R
 eq: genre_equation) is

 ii,jj: integer;

 begin
 for j in 1..iel loop
 jj:= equa_el(j);
 if jj > 0 then -- noeud libre; jj=num. de l'equation

```

```

 for i in 1..iel loop
 ii:= equa_el(i); -- ii=num. de l'equation
 if ii > 0 then
 Add(MB, ii,jj, multi_M * Me(i,j) + multi_R * Re(i,j));
 end if;
 end loop; -- i
 end if;
end loop; -- j
end Assma_bande;

procedure Assma_creux
(MC : in out CRS_matrix;
 Me,Re: el_matrix;
 equa_el: el_iarray;
 multi_M, multi_R: FT_real -- multiplicateurs de M et R
) is

 ii,jj: integer;

begin
 for j in 1..iel loop
 jj:= equa_el(j);
 if jj > 0 then -- noeud libre; jj=num. de l'equation
 for i in 1..iel loop
 ii:= equa_el(i); -- ii=num. de l'equation
 if ii > 0 then
 Add(MC, ii,jj, multi_M * Me(i,j) + multi_R * Re(i,j));
 end if;
 end loop; -- i
 end if;
 end loop; -- j

end Assma_creux;

procedure Assma
(nom_matr: character; Me,Re: el_matrix;
 equa_el: el_iarray;
 multi_M, multi_R: FT_real; -- multiplicateurs de M et R
 eq: genre_equation) is

begin
 case format_matrices is

 when bande =>
 case nom_matr is
 when 'A' => Assma_bande(AB.all, Me,Re,equa_el, multi_M, multi_R, eq);
 when 'B' => Assma_bande(BB.all, Me,Re,equa_el, multi_M, multi_R, eq);
 when others => null;
 end case;

 when creux =>
 case nom_matr is
 when 'A' => Assma_creux(AC.all, Me,Re,equa_el, multi_M, multi_R);
 when 'B' => Assma_creux(BC.all, Me,Re,equa_el, multi_M, multi_R);
 when others => null;
 end case;

 end case;
end Assma;

```

```

-- 13-Jan-2000: STILS-TMQ2+. On assemble les matrices dans le systeme
-- multiple.

-- En mode <<grands blocs>> on aura
--
-- Q2:
-- T
-- [R M]
-- A = [M M4] B = [-M 0]
-- [0 -M3]

-- 20-Jun-2000: generique (matrices creuses ET bande!)

generic
 type kind_of_matrix is private; -- (<>): precision pour Ada95
 type kind_of_matrix_a is access kind_of_matrix;

 A, B : kind_of_matrix_a;

 with procedure Add(A: in out kind_of_matrix; i,j: integer; value: FT_real);

procedure g_Assma_Multibloc (Me, Re, Me3, Me4, Me5, Me6,
 Me7, Me8, Me9, Me10, Me11, Me12 : el_matrix;
 equa_el : el_iarray);

procedure g_Assma_Multibloc (Me, Re, Me3, Me4, Me5, Me6,
 Me7, Me8, Me9, Me10, Me11, Me12 : el_matrix;
 equa_el : el_iarray) is
 ii, jj, N: integer;
 nbc: constant positive := multi_blocs(meth_trans);
 grands_blocs_avec_matrice_bande: exception;
begin
 if format_matrices = bande and config_blocs = grands_blocs then
 raise grands_blocs_avec_matrice_bande;
 end if;

 case config_blocs is
 when grands_blocs => N := nb_equa(trans); -- bloc suivant
 when petits_blocs => N := 1; -- case suivante
 end case;

 for j in 1..iel loop
 jj := equa_el(j);
 if jj > 0 then -- noeud libre; jj=num. de l'equation
 case config_blocs is
 when grands_blocs => null;
 when petits_blocs => jj := (jj-1) * nbc + 1; -- 1,2,.. vers 1,3,..
 end case;
 for i in 1..iel loop
 ii := equa_el(i); -- ii=num. de l'equation
 if ii > 0 then
 case config_blocs is
 when grands_blocs => null;
 when petits_blocs => ii := (ii-1) * nbc + 1; -- 1,2,.. vers 1,3,..
 end case;

 case meth_trans is

```

**when** STILS.TMQ2 => -- Système (4.6)

```
Add(A.all, ii, jj, Re(i,j)); -- R -- ligne 1
Add(A.all, ii, N+jj, Me(j,i)); -- Mt
Add(A.all, N+ii, jj, Me(i,j)); -- M -- ligne 2
Add(A.all, N+ii, N+jj, Me4(i,j)); -- M4

Add(B.all, ii, jj, -Me(i,j)); -- -M
Add(B.all, N+ii, N+jj, -Me3(i,j)); -- -M3
```

**when** STILS.TMQ3 => -- Système (4.15)

```
Add(A.all, ii, jj, Re(i,j)); -- R -- ligne 1
Add(A.all, ii, N+jj, Me3(i,j)); -- M3
Add(A.all, ii, 2*N+jj, Me4(i,j)); -- M4

Add(A.all, N+ii, jj, Me3(j,i)); -- M3t -- ligne 2
Add(A.all, N+ii, N+jj, Re(i,j)); -- R
Add(A.all, N+ii, 2*N+jj, Me(j,i)); -- Mt

Add(A.all, 2*N+ii, jj, Me4(j,i)); -- M4t -- ligne 3
Add(A.all, 2*N+ii, N+jj, Me(i,j)); -- M
Add(A.all, 2*N+ii, 2*N+jj, Me6(i,j)); -- M7

Add(B.all, ii, jj, -Me(i,j)); -- -M
Add(B.all, N+ii, N+jj, -Me4(j,i)); -- -M4t
Add(B.all, 2*N+ii, 2*N+jj, -Me5(i,j)); -- -M5
```

**when** STILS.TDQ1 =>

```
Add(A.all, ii, jj, Me(i,j)); -- ligne 1
Add(A.all, ii, N+jj, Re(i,j));
Add(A.all, N+ii, jj, Me3(i,j)); -- ligne 2
Add(A.all, N+ii, N+jj, Me4(i,j));
if ii==jj then
 Add(A.all, 2*N+ii, 2*N+jj, 1.0); -- ligne 3
end if;

Add(B.all, ii, jj, Me5(i,j));
Add(B.all, ii, N+jj, Me6(i,j));
Add(B.all, ii, 2*N+jj, Me7(i,j));
Add(B.all, N+ii, jj, Me3(i,j));
Add(B.all, N+ii, N+jj, Me8(i,j));
Add(B.all, N+ii, 2*N+jj, Me9(i,j));
```

**when** Galer.TDQ1 =>

```
Add(A.all, ii, jj, Re(i,j)); -- R -- ligne 1
Add(A.all, ii, N+jj, Me3(i,j)); -- M3
Add(A.all, N+ii, jj, Me4(i,j)); -- M4 -- ligne 2
Add(A.all, N+ii, N+jj, Re(i,j)); -- R

Add(B.all, ii, jj, Me(i,j)); -- M
```

**when others** => **raise** cas\_imprevu;

**end case**; -- meth\_trans

**end if**;

```

 end loop; -- i
 end if;
end loop; -- j

end g_Assma_Multibloc;

procedure Assma_Multibloc (Me, Re, Me3, Me4, Me5, Me6 : el_matrix;
 equa_el : el_iarray) is

 procedure Assma_Multibloc_creux is
 new g_Assma_Multibloc(FTRSM.CRS_Matrix, FTRSM.p_CRS_Matrix,
 A2C, B2C, FTRSM.Add);

 procedure Assma_Multibloc_bande is
 new g_Assma_Multibloc(FTRBM.Band_matrix, FTRBM.p_Band_matrix,
 A2B, B2B, FTRBM.Add);

begin
 case format_matrices is
 when bande => Assma_Multibloc_bande (Me, Re, Me3, Me4, Me5, Me6, equa_el);
 when creux => Assma_Multibloc_creux (Me, Re, Me3, Me4, Me5, Me6, equa_el);
 end case;
end Assma_Multibloc;

```

### C.3 Remplissage des matrices globales

Voici la procedure principale consacrée au remplissage des matrices  $A$  et  $B$ .

```

procedure Remplir_matrices(eq: genre_equation; stationn: boolean) is
 fl, -- force aux noeuds (inutilise ici)
 be: el_array; -- second membre elementaire
 mem_el: positive;
 nee: el_iarray; -- vecteurs elem. des no d'eq.
 Me, Re, Me3, Me4, Me5, Me6: el_matrix;

begin
 case format_matrices is
 when bande =>
 for j in 1..nb_noeuds loop
 for ib in AB'range(2) loop AB(j, ib):= 0.0; BB(j, ib):= 0.0; end loop;
 end loop;
 when creux =>
 for i in AC.val'range loop AC.val(i):= 0.0; BC.val(i):= 0.0; end loop;
 if A2C /= null then
 for i in A2C.val'range loop
 A2C.val(i):= 0.0;
 B2C.val(i):= 0.0;
 end loop;
 end if;
 end case;

 -- Les integrales pouvant etre non nulles sont celles de paires de fonctions
 -- de base correspondant a 2 noeuds ayant au moins un element en commun;
 -- il suffit donc de calculer sur chaque element les integrales des
 -- paires de fonctions de cet element; au total, on a les integrales
 -- calculees sur le domaine.

```

```

elements:
 for n in 1..nb_elem loop
 mem_el:= n; -- en cas d'exception
 for i in 1..iel loop
 nee(i):= ia(eq)(elem(i,n));
 end loop;
 -- Matrices de masse et de rigidite elementaires: Me, Re
 Ematr(Me,Re,Me3,Me4,Me5,Me6,
 be, nee, nnde(n), xne(n), fl, calc_second=> false, eq=> eq);
 -- Assemblage des matrices totales
 if stationn then -- cas stationnaire (pour CI)
 Assma('A', Me,Re,nee, 0.0, 1.0, eq);

 elseif eq=trans and meth_trans = STILS_TM then
 Assma('A', Me,Re,nee, 1.0, 0.0, trans); -- Ici A="M"
 Assma('B', Me,Re,nee, 0.0, 1.0, trans); -- B="R"

 elseif eq=trans and STILS_multi(meth_trans) then

 Assma_TQ2(Me,Re,Me3,Me4,Me5,Me6, nee);

 else
 case methode_temps_EF_effective is
 when Schema_th =>
 Assma('A', Me,Re,nee, iv_tau, theta, eq);
 Assma('B', Me,Re,nee, iv_tau, theta-1.0, eq);
 when BDF2 =>
 Assma('A', Me,Re,nee, 1.5*iv_tau/1.5, 1.0, eq);
 Assma('B', Me,Re,nee, iv_tau/1.5, 0.0, eq);
 end case;
 end if;
end loop elements;
exception
 when FEK.negative_jacobian =>
 Put("matr: orientation inversee sur l'elem:");
 Put(mem_el); raise; -- on re-leve l'exception
 when others =>
 Put(mem_el); raise; -- on re-leve l'exception
end Remplir_matrices;

```

## C.4 Paramètres pour l'ensemble du projet Flux-Transport

Tout — hormis STILS-Burgers non-linéarisé — est calculé par un même programme, avec des paramètres qui spécifient la méthode de résolution en temps (pour des méthodes d'éléments finis classiques), le type de matrice (matrice bande ou creuse), le type d'élément, et par là la dimension spatiale. Dès lors qu'un paramètre correspondait à un choix avantageux, indépendamment des problèmes à résoudre, nous l'avons fixé en constante — ce qu'un optimiseur de code comprend comme une commande de compilation conditionnelle. Au contraire, un paramètre comme le type d'élément apparaît dans le fichier de définition du problème et non dans le programme puisqu'il est destiné à changer de cas en cas.

---

----| Parametres pour le programme FT Flux-Transport |----

---

with IEEE\_numbers, Text\_IO, Sequential\_IO;

```
with FEFLOW, Sparse, FEK, Matrices, BandMatr;
with Min_Max, Add_Inc;
```

```
package FT_Param is
 use FEK, Text_IO;
```

---

```
----- Nombres a virgule flottante -----
```

---

```
 subtype FT_real is long_float; -- type flottant pour TOUS les calculs de FT
```

```
 -- IEEE pour les E-S binaires
```

```
 subtype IEEE_real is IEEE_numbers.IEEE_real;
```

---

```
----- Modules divers, pour l'exportation -----
```

---

```
 package MMI is new Min_Max(integer,">");
```

```
 package MMF is new Min_Max(FT_real,">");
```

```
 package ADF is new Add_Inc(FT_real,1.0,"+", "-");
```

```
 package IIO is new Integer_IO(integer);
```

```
 package FIO is new Float_IO(float);
```

```
 package FTRIO is new Float_IO(FT_real);
```

```
 package SFIO is new Sequential_IO(IEEE_real);
```

```
 -- Matrices bandes, pour FT_real
```

```
 package FTRBM is new BandMatr(FT_real, integer,
 Matrices.vector, Matrices.matrix);
```

```
 -- Matrices creuses, pour FT_real
```

```
 package FTRSM is new Sparse(FT_real, integer, Matrices.vector);
```

```
 type mode_velo is (vel_noeuds, vel_elems);
```

```
 package VLIO is new Enumeration_IO(mode_velo);
```

```
 -- vel_noeuds: on considere la vitesse en chaque noeud (lecture de donnees)
```

```
 -- vel_elems: on considere la vitesse au centre de chaque element (joli)
```

---

```
----- Identification des EDP -----
```

---

```
 type genre_equation is (flux, trans); -- sortes d'EDP
```

```
 nb_eq: natural; -- nombre d'EDP
```

```
 -- nb d'equa. lin. par EDP, <= nb_noeuds
```

```
 nb_equa: array(genre_equation) of natural;
```

---

```
----- Fichiers pour les solutions -----
```

---

```
 mode_seq: constant boolean:= false; -- true: sequentiel; false: texte
```

```
 f_h: Text_IO.file_type; -- charge en mode texte
```

```
 f_h_seq: SFIO.file_type; -- charge en mode seq.
```

```
 f_C: Text_IO.file_type; -- concentration en mode texte
```

```
 f_dtC: Text_IO.file_type; -- dc/dt (mth. hybr.) en mode texte
```

```
 f_C_seq: SFIO.file_type; -- concentration en mode seq.
```

```

f_part: Text_IO.file_type; -- particules en mode texte
f_vel: Text_IO.file_type; -- vitesse de filtration en mode texte
f_vel_seq: SFIO.file_type; -- vitesse de filtration en mode seq.
f_cnc_int: Text_IO.file_type; -- concentration integrale

f_sol: Text_IO.file_type; -- fichiers divers
f_cl: array(genre_equation) of Text_IO.file_type;

----- Noms de fichiers -----

nom_noeuds: constant string:= "node.dat";
nom_elem: constant string:= "elem.dat";
nom_temps: constant string:= "time.cfg";
nom_meth: constant string:= "meth.cfg";

nom_h: constant string:= "sol001.dat";
nom_cnc: string(1..10):= "solxxx.dat"; -- xxx=002..999
nom_dtc: string(1..10):= "dtxxx.dat"; -- xxx=002..999
nom_part: constant string:= "par002.dat";
nom_vel: constant string:= "vel001.dat";
nom_cnc_int: string(1..10):= "cncxxx.dat"; -- xxx=002..999

----- Identification des methodes pour resoudre l'equation de transport --

type t_methode_trans is
(
 C_Convection, -- Caracteristiques
 Galer_kine, -- EF normaux
 Galer_TMQ1, -- EF en espace-temps
 Galer_TDQ1, -- EF en espace-temps, discontinus en temps
 Streamline, -- ou SUPG - fonction-test w modifiee par terme (V|grad w)
 Strea_TMQ1, -- SUPG en espace-temps
 SUPG_DC, -- SUPG avec encore un terme (V//|grad w); V// est une proj.
 SILS, -- SILS : moindres-carres en espace seulement!
 STILS_TMQ1, -- STILS sur [t;t+tau] et approximations -> marche en temps
 STILS_TMQ2, -- STILS Q2 en temps
 STILS_TMQ3, -- STILS Q3 en temps
 STILS_TDQ1, -- STILS discontinu Q1
 STILS_TMQ1S2 -- STILS a deux pas sur [t-tau;t+tau]
);

type t_multi_blocs is array(t_methode_trans) of positive ;

multi_blocs : constant t_multi_blocs := t_multi_blocs'

(Galer_TDQ1 | STILS_TMQ2 => 2,
 STILS_TMQ3 => 3,
 STILS_TDQ1 => 3,
 others => 1);

type set_of_meth is array(t_methode_trans) of boolean ;

-- Methodes espace-temps (sinon: semi-discretise)

Space_time_method: constant set_of_meth:=
 set_of_meth'(Galer_TMQ1 | Strea_TMQ1 |

```

```

 STILS-TMQ1 |
 STILS-TMQ2 | STILS-TMQ3 |
 Galer-TDQ1 | STILS-TDQ1 |
 STILS-TMQ1S2 => true,
 others => false);

-- Methodes multi-blocs

Space-time-multi: constant set_of_meth:=
 set_of_meth'(STILS-TMQ2 | STILS-TMQ3 |
 Galer-TDQ1 | STILS-TDQ1 |
 others => true,
 => false);

Two-step-STILS: constant set_of_meth:=
 set_of_meth'(STILS-TMQ1S2 => true,
 others => false);

-- Methode augmentee (hybride): on affiche aussi $dc/dt = p$

Augmented: constant set_of_meth:=
 set_of_meth'(STILS-TDQ1 => true,
 others => false);

----- Parametres des matrices -----

-- Ameliorations a faire [13-dec-2000] :
-- - faire dependre la symetrie du type de methode (equation de transport)
-- - desolidariser completement les matrices pour l'equation de flux
-- et celle de transport.
-- Ex. pour plug-flow, il faudrait
-- - le format bande, asymetrique pour l'equation de flux
-- (CG meme symetrique ne converge pas)
-- - le format creux, symetrique pour l'equation de transport

type format_matrice is (bande, creux);
format_matrices: constant format_matrice:= creux; -- Tout sous un format

-- Pour les methodes espace-temps d'ordre m en temps: faire
-- m blocs de taille N (grands blocs) ou N blocs de taille m (petits blocs)

type t_config_blocs is (petits_blocs, grands_blocs);
config_blocs: constant t_config_blocs:= petits_blocs;

lda, -- demi-largeur de bande des matrices bande
nnz_max: natural; -- nombre max. de compos. non-nulles des matr. creuses

-- Symetrie des matrices:

Matr_sym: constant boolean:= false; -- Supposer A,B symetriques, ou non

mult_lda: constant array(boolean) of positive:= (true=> 1, false=> 3);
-- Essai du 1.IX.1998: 688 elements, 7155 noeuds (fissu3), 72 pas d'1 heure
-- aucune difference (ascii!) entre les fichiers solution pour sym et asym !

-- Type de preconditionnement pour methode iterative
precond: constant FTRSM.t_precond:= FTRSM.diag;

```

---

----- Particules -----

---

```
-- facteur d'amplitude du nombre max approx de particule p.r. au type d'elem
ampl_partic: constant array(t_element) of positive:=
 (L2=> 1, Q4=> 7, Q9=> 6, B27=> 6);

-- nombre maximal d'elements voisins a un element (sens strict)
max_voisins: constant array(t_element) of positive:=
 (L2=> 2, Q4=> 8, Q9=> 8, B27=> 26);
```

---

----- Temps -----

---



---

```
-- Choix de methodes temporelles arbitraires (/= STILS...) pour --
-- la discretisation totale --

type t_methode_temps is (BDF2, Schema_th);
theta: constant:= 0.5; -- theta du schema eponyme

-- Methode numerique pour la discretisation EF en temps:
methode_temps_EF: constant t_methode_temps:= BDF2;
methode_temps_EF_effective: t_methode_temps; -- :=methode_temps_EF apres pas 2

-- Parametres lus dans time.cfg des 12.III.1998 :

nb_pas_t: natural; -- nombre de pas de temps (retenus)
pas_de_t: FT_real; -- duree d'un pas de temps [s]
 -- FEFLOW p. 5-6: pas de 1000 sec.

nb_pas_interm: natural; -- nombre de pas de temps intermediaires
 -- (non retenus dans la solution)
nb_pas_cc_max: natural; -- nombre de pas de temps intermediaires
 -- max. pour c-convection
vagues: Natural; -- nombre de vagues d'injection (c-c)

tau, -- τ pas de temps fin, = pas_de_t / (nb_pas_interm+1)
iv_tau, -- 1/tau
temps: FT_real;
```

---

----- Elements Finis -----

---

```
FEK_eps: constant:= 1.0e-8; -- tolerance pour FEK

-- Points tk et poids ak de Gauss pour l'integration en n=3 points
-- des fonctions de base (shape functions)
gauss: array(1..3) of
 FT_real:= (0.77459666924148, 0.0, -0.77459666924148);
-- t1 = -sqrt(3/5), t2 = 0, t3 = sqrt(3/5) racines de (1/2)(5x^3-3x)
poids: array(1..3) of
 FT_real:= (0.555555555555556,0.888888888888889,0.555555555555556);
-- ak = 8 / [(3 tk^2 - 1) (15 tk^2 - 3)]
```

```

nb_elem, -- nombre d'elements du maillage
nbfix, -- nombre de conditions aux limites fixes
nb_noeuds: natural; -- nombre de noeuds

neumann_pur: array(genre_equation) of boolean;
 -- faux=> il y a des conditions de dirichlet
 -- vrai=> la solution est definie *a constante pres*

type tbool is array(integer range <>) of boolean;
type pbool is access tbool;
neubor, elembor: pbool; -- noeud au bord ? elem touchant un bord ?

-- Supposer le flux h stationnaire ? non=> on ne recalc. pas sur [0,T]
flux_stationn: boolean:= False;
-- variable (24-aug-2000). mise sur True si : code_CI(flux)=key_is_steady

--| **** Ces parametres dependent du probleme ! **** |--

-- Ss: Specific storage / emmagasinement specifique
Ss: constant := FEFLOW.Theis.S;

omega : constant:= 1.0; -- Porosite 0.3
iv_omega: constant:= 1.0 / omega;

-- Resoudre l'eq de flux ?
-- non=> c'est un cas sans potentiel,
-- ou bien on espere que la bonne solution est deja la
resoudre_flux: boolean:= true;

-- Traitement des over- / undershootings

troncage_permis: constant boolean:= false;
pas_tronque: natural:= 0;
solmax, solmin: array(genre_equation) of FT_real;

-- Divers

cas_imprevu: exception;
debug : constant boolean:= false;
debug_CL : constant boolean:= false;
debug_c_convection: constant boolean:= false;
f_debug: Text_IO.file_type;

end FT_Param;

```

## C.5 STILS-Burgers

Ce qui suit est le listage complet du programme de résolution pour l'équation non-linéaire de Burgers par la méthode STILS-TMQ1, à savoir (6.6) qui équivaut au système (6.7). On retrouvera la définitions des fonctions *alpha*, *beta*, *gamma*, *epsilon* dans (6.8), de *fx* dans

(6.9), de  $Df_x$  dans (6.10). Bien entendu, les intégrales sont calculées à l'avance puisqu'on connaît les fonctions de base  $\psi_j$ .

```

--
-- STILS-Burgers, GdM 29-feb-2000 (comm. 21h47, fonct. 0h44 :-)
--

with Text_IO;
with Math_Lib;

use Text_IO;

procedure STILBurg is

 subtype real is long_float;
 package NEF is new Math_Lib(real); use NEF;
 package RIO is new Float_IO(real); use RIO;

 N: constant integer:= 7;
 h: constant real:= 1.0 / real(N+1);

 N_pas : constant:= 50;
 t_final: constant:= 0.25;

 compte_it_newt: natural;

 tau: constant:= t_final / N_pas;

 f: file_type;

 type vector is array(integer range <>) of real;

 u: vector(0..N+1); -- u_0, u_{N+1} valent 0 (CL)

 x,r: vector(1..N); -- solution d'essai et reste pour Newton

 -- Matrice tridiagonale pour Ax=d

 a: vector(1..N); -- diag
 b: vector(1..N-1); -- diag inf
 c: vector(1..N-1); -- diag sup

 d: vector(1..N); -- 2nd membre

 -- Sous-programmes
 -- Partie analyse

 -- integrales des fonctions de base

 function alpha(i,j: integer) return real is
 begin
 case abs(i-j) is
 when 0 => return 2.0 * h / 3.0;
 when 1 => return h / 6.0;
 when others => return 0.0;
 end case;
 end alpha;

 function beta(i,j,l: integer) return real is
 begin
 if i=j and j=l then
 return 0.0;

```

```

 elsif i=l and l=j-1 then
 return -1.0/3.0;
 elsif i=l and l=j+1 then
 return 1.0/3.0;
 elsif (j=l and l=i+1) or (i=j and j=l+1) then
 return -1.0/6.0;
 elsif (j=l and l=i-1) or (i=j and j=l-1) then
 return 1.0/6.0;
 else
 return 0.0;
 end if;
end beta;

function gamma(i,j,k: integer) return real is
begin
 if i=j and j=k then
 return 0.0;
 elsif i=j and j=k-1 then
 return 1.0/3.0;
 elsif i=j and j=k+1 then
 return -1.0/3.0;
 elsif (i=k and k=j+1) or (j=k and k=i+1) then
 return 1.0/6.0;
 elsif (i=k and k=j-1) or (j=k and k=i-1) then
 return -1.0/6.0;
 else
 return 0.0;
 end if;
end gamma;

function epsilon(i,j,k,l: integer) return real is
begin
 if (l=i and i=k and k=j-1) or (l=i and i=k and k=j+1) then
 return 1.0 / (3.0 * h);
 elsif (i=k and k=j-1 and j=l) or (k=l and l=i-1 and i=j) or
 (i=j and j=k-1 and k=l) or (i=k and k=j+1 and j=l) then
 return 1.0 / (6.0 * h);
 elsif (i=j and j=l and l=k+1) or (i=j and j=l and l=k-1) then
 return -2.0 / (3.0 * h);
 else
 return 0.0;
 end if;
end epsilon;

-- On cherche x(1)..x(N) tel que fx(i)=0 pour tous les i

function fx(i:integer) return real is
f: real:= 0.0;
b1: integer:= 1; -- borne inf des sommes
b2: integer:= N; -- borne sup des sommes
begin

 -- support des fonctions de base fait que
 -- les int. sont nuls pour |i-j|>1, |i-k|>1, etc.
 if b1 < i-1 then b1:= i-1; end if;
 if b2 > i+1 then b2:= i+1; end if;

 for j in b1..b2 loop
 f:= f + (x(j)-u(j)) * alpha(i,j) / tau;
 end loop;
end fx;

```

```

end loop;

for j in b1..b2 loop
 for l in b1..b2 loop
 f:= f + (x(j)-u(j)) * (u(l)+2.0*x(l)) * beta(i,j,l) / 6.0;
 end loop;
end loop;

for j in b1..b2 loop
 for k in b1..b2 loop
 f:= f + (u(j)*(2.0*u(k)+x(k)) + x(j)*(u(k)+2.0*x(k))) *
 gamma(i,j,k) / 6.0;
 end loop;
end loop;

for j in b1..b2 loop
 for k in b1..b2 loop
 for l in b1..b2 loop
 f:= f + (
 u(j)*(3.0*u(k)*u(l) + 2.0*x(k)*u(l) +
 2.0*u(k)+x(l) + 3.0*x(k)*x(l))
 +
 x(j)*(u(k)*(2.0*u(l)+3.0*x(l)) + 3.0*x(k)*(u(l)+4.0*x(l)))
) * epsilon(i,j,k,l) * tau / 60.0;
 end loop;
 end loop;
end loop;

return f;

end fx;

function Dfx(i,m: integer) return real is
 f: real;
 b1: integer:= 1; -- borne inf des sommes
 b2: integer:= N; -- borne sup des sommes
begin
 -- support des fonctions de base fait que
 -- les int. sont nuls pour |i-j|>1, |i-k|>1, etc.
 if b1 < i-1 then b1:= i-1; end if;
 if b2 > i+1 then b2:= i+1; end if;

 if abs(i-m) > 1 then
 return 0.0;
 else
 f:= alpha(i,m) / tau;

 for j in b1..b2 loop
 f:= f +
 (x(j) / 3.0) * (beta(i,m,j) + beta(i,j,m) +
 gamma(i,m,j) + gamma(i,j,m))
 +
 (u(j) / 6.0) * (beta(i,m,j) - 2.0 * beta(i,j,m) +
 gamma(i,m,j) + gamma(i,j,m));
 end loop;
 end if;
end function Dfx;

```

```

 for j in b1..b2 loop
 for k in b1..b2 loop
 f:= f + (tau/60.0) *
 (
 2.0*u(j)*u(k)*(epsilon(i,j,m,k)+epsilon(i,j,k,m)+
 epsilon(i,m,j,k))
 +
 3.0*u(j)*x(k)*(epsilon(i,j,k,m)+epsilon(i,j,m,k)+
 epsilon(i,k,j,m)+epsilon(i,m,j,k)+
 epsilon(i,k,m,j)+epsilon(i,m,k,j))
 +
 12.0*x(j)*x(k)*(epsilon(i,m,j,k)+epsilon(i,j,k,m)+
 epsilon(i,j,m,k))
);
 end loop;
 end loop;

 return f;
 end if;

end Dfx;

-- Partie alg. lin.

-- Decomposition LR tridiagonale universelle
-- Avant a,b,c pour A; apres: a=m,b=l pour L; c=r pour R

procedure LR(a,b,c: in out vector) is
 bad_dims: exception;
begin
 if a'first/=1 or b'first/=1 or c'first/=1 or
 b'last/=c'last or b'last/=a'last-1 then
 raise bad_dims;
 end if;
 for i in b'range loop
 c(i):= c(i) / a(i);
 a(i+1):= a(i+1) - c(i)*b(i);
 end loop;
end LR;

procedure Solve_LR(m,l,r,d: in vector; x: in out vector) is
 y: vector(d'range);
 n: constant integer:= a'last;
 bad_dims: exception;
begin
 if m'first/=1 or l'first/=1 or r'first/=1 or
 d'first/=1 or x'first/=1 or
 l'last/=n-1 or r'last/=n-1 or d'last/=n or x'last/=n then
 raise bad_dims;
 end if;

 -- Ly=d :
 y(1):= d(1) / m(1);
 for i in 2 .. y'last loop
 y(i):= (d(i) - l(i-1) * y(i-1)) / m(i);
 end loop;

 -- Rx=y :

```

```

 x(n):= y(n);
 for i in reverse 1 .. n-1 loop
 x(i):= y(i) - r(i) * x(i+1);
 end loop;
end Solve_LR;

function Norme(x:vector) return real is
 n: real:= 0.0;
begin
 for i in x'range loop
 n:= n + x(i)*x(i);
 end loop;
 return Sqrt(n);
end Norme;

begin
 for j in 0..N+1 loop u(j):= sin(2.0*3.1415926*real(j)*h); end loop;

 Create(f, name=> "ustils.dat");

 for i in 0..N+1 loop Put(f,u(i),3,3,0); end loop; New_Line(f);

 temps: for k in 1..N_pas loop

 Put(integer'image(k)); Put(' ');

 x:= u(1..N); -- vecteur initial Newton

 compte_it_newt:= 1;

 newton: loop

 -- remplir matrice
 for i in 1..N loop a(i):= Dfx(i ,i); end loop;
 for i in 1..N-1 loop b(i):= Dfx(i+1,i); end loop;
 for i in 1..N-1 loop c(i):= Dfx(i ,i+1); end loop;

 -- 2nd membre
 for i in 1..N loop d(i):= -fx(i); end loop;

 LR(a,b,c); -- decomposition
 Solve_LR(a,b,c,d, r); -- resolution

 for i in x'range loop x(i):= x(i) + r(i); end loop;

 exit when Norme(r) < 0.00001;

 compte_it_newt:= compte_it_newt + 1;
 Put(' ');

 end loop newton;

 New_Line;

 u(1..N):= x;

 for i in 0..N+1 loop Put(f,u(i),3,3,0); end loop; New_Line(f);

 end loop temps;

```

```
 Close(f);
end STILBurg;
```

# Bibliographie

- [1] Y. AMIRAT, D. BRESCH, J. LEMOINE ET J. SIMON, *Démonstration constructive d'un théorème de G. de Rham*, Rend. Sem. Mat. Univ. Padova, 102 (1999), pp. 341–365.
- [2] P. AZÉRAD, *Analyse des équations de Navier-Stokes en bassin peu profond et de l'équation de transport*, Thèse, Université de Neuchâtel, 1996.
- [3] P. AZÉRAD ET J. POUSIN, *Inégalité courbe pour le traitement variationnel de l'équation de transport*, C. R. Acad. Sci Paris, 322 (1996), pp. 721–727.
- [4] J. BARANGER ET D. SANDRI, *Quelques propriétés de la méthode de Galerkin discontinue*, C. R. Acad. Sci Paris, 322 (1996), pp. 785–788.
- [5] J. BARRÈRE, O. GIPOULOUX ET S. WHITAKER, *On the closure problem for Darcy's law*, Transport in porous media, 7 (1992), pp. 209–222.
- [6] O. BESSON, *Communication personnelle*, (2001).
- [7] M. BOSSY ET D. TALAY, *Vitesse de convergence d'un algorithme particulière stochastique pour l'équation de Burgers*, C. R. Acad. Sci Paris, 320 (1995), pp. 1129–1134.
- [8] H. BREZIS, *Analyse fonctionnelle*, Masson, 1983.
- [9] F. BREZZI ET M. FORTIN, *Mixed and hybrid finite element methods*, Springer-Verlag, 1991.
- [10] P. CIARLET, *The finite element method for elliptic problems*, North-Holland Elsevier, 1978.
- [11] B. COCKBURN ET C.-W. SHU, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2440–2463.
- [12] R. COLLIN, *Bandwidth reduction by automatic renumbering*, International Journal for Numerical Methods in Engineering, 6 (1973), pp. 345–356.
- [13] B. DACOROGNA, *Direct methods in the calculus of variations*, Springer-Verlag, 1989.
- [14] ———, *Introduction au calcul des variations*, PPUR, 1992.
- [15] H. DARCY, *Les fontaines publiques de la ville de Dijon*, Dalmont, Paris, 1856.
- [16] R. DAUTRAY ET J. L. LIONS, *Analyse mathématique et calcul numérique*, vol. 9, Masson, 1988.
- [17] G. DE MARSILY, *Hydrogéologie quantitative*, Masson, 1981.
- [18] G. DE MONTMOLLIN, *STILS method for pure convection equation : Time-marching approach and numerical comparisons*, ECCOMAS 2000 Congress, Barcelona, (2000).
- [19] G. DE RHAM, *Variétés différentiables*, Hermann, 1960.
- [20] J. DENY ET J.-L. LIONS, *Les espaces du type Beppo Levi*, Ann. Inst. Fourier, 5 (1954), pp. 305–370.
- [21] H.-J. G. DIERSCH, *FEFLOW - Physical basis of modeling*, WASY, Institute for Water Resources Planning and Systems Research, Berlin, 1996.
- [22] I. EKELAND ET R. TEMAM, *Analyse convexe et problèmes variationnels*, Dunod, 1974.
- [23] E. GODLEWSKI ET P. A. RAVIART, *Hyperbolic systems of conservation laws*, SMAI - Ellipses, 1990.
- [24] ———, *Numerical approximation of hyperbolic systems of conservation laws*, Springer-Verlag, 1996.

- [25] E. HAIRER, S. NØRSETT ET G. WANNER, *Solving Ordinary Differential Equations I*, Springer-Verlag, 1987.
- [26] P. HANSBO, *The characteristic streamline diffusion method for convection-diffusion problems*, Comput. Meths Appl. Mech. Eng., 96 (1992), pp. 239–253.
- [27] T. J. R. HUGHES, *The Finite Element Method*, Prentice-Hall, 1987.
- [28] T. J. R. HUGHES, L. P. FRANCA ET G. M. HUBERT, *A new finite element formulation for computational fluid dynamics : VIII. The Galerkin/Least-Squares method for advective-diffusive equations*, Comput. Meth. Appl. Mech. Eng., 73 (1989), pp. 173–189.
- [29] T. J. R. HUGHES, M. MALLETT ET A. MIZUKAMI, *A new finite element formulation for computational fluid dynamics : II. Beyond SUPG*, Comput. Meth. Appl. Mech. Eng., 54 (1986), pp. 341–355.
- [30] P. S. HUYAKORN ET G. F. PINDER, *Computational methods in subsurface flow*, Academic Press, 1983.
- [31] F. JOHN, *Partial Differential Equations*, Springer-Verlag, 1978.
- [32] C. JOHNSON, *Numerical solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, 1987.
- [33] C. JOHNSON, A. SZEPESSY ET P. HANSBO, *On the convergence of shock-capturing streamline diffusion finite element methods for hyperbolic conservation laws*, European Congress on Computational Methods in Applied Sciences and Engineering, 54 (1990), pp. 107–129.
- [34] O. KOLDITZ, *FEFLOW - Benchmarks and examples*, WASY, Institute for Water Resources Planning and Systems Research, Berlin, 1996.
- [35] D. KRÖNER, *Numerical schemes for conservation laws*, Wiley - Teubner, 1997.
- [36] B. LAPEYRE, E. PARDOUX ET R. SENTIS, *Méthodes de Monte-Carlo pour les équations de transport et de diffusion*, Springer, 1998.
- [37] A. LATTO ET E. TENENBAUM, *Compactly supported wavelets and the numerical solution of Burgers' equation*, C. R. Acad. Sci Paris, 311 (1990), pp. 903–909.
- [38] P. D. LAX, *Hyperbolic systems of conservation laws and the mathematical theory of shock waves*, SIAM, 1973.
- [39] P. D. LAX ET B. WENDROFF, *On the stability of difference schemes*, Comm. on pure and appl. Math., 15 (1962), pp. 363–371.
- [40] P. LESAIN ET P. A. RAVIART, *On a finite element method for solving the neutron transport equation*, Math. Research Center, Madison, 33 (1974), pp. 89–123.
- [41] R. J. LEVEQUE, *Numerical Methods for Conservation Laws*, Birkhäuser, 1991.
- [42] J.-L. LIONS, *Quelques méthodes de résolution des problèmes aux limites non linéaires*, Gauthier-Villars, 1969.
- [43] E. MAGENES ET G. STAMPACCHIA, *I problemi al contorno per le equazioni differenziali di tipo ellittico*, Ann. Scuola Norm. Sup. Pisa, 12 (1958), pp. 247–357. 27.
- [44] J. NEČAS, *Equations aux dérivées partielles*, Presses Univ. Montréal, 1965.
- [45] H. NGUYEN ET J. REYNEN, *A space-time least squares finite element scheme for advection-diffusion equations*, Comput. Meths. Appl. Mech. Engrg., 42 (1984), pp. 331–342.
- [46] J. T. ODEN, I. BABUŠKA ET C. E. BAUMANN, *A discontinuous hp finite element method for diffusion problems*, J. Comput. Phys., 146 (1998), pp. 495–519.
- [47] P. PERROCHET, *Méthodes de simulation opérationnelles aux éléments finis pour la résolution de problèmes de transferts dominés par la convection. Applications aux pollutions des eaux souterraines.*, Thèse N° 1046, École Polytechnique Fédérale de Lausanne, 1992.
- [48] P. PERROCHET ET P. AZÉRAD, *Space-time integrated least-squares : Solving a pure advection equation with a pure diffusion operator*, J. Comput. Phys., 117 (1995), pp. 183–193.

- [49] G. PINDER ET E. O. FRIND, *Application of Galerkin's procedure to aquifer analysis*, Water Resources Research, 8 (1972), pp. 108–120.
- [50] P. A. RAVIART ET J. M. THOMAS, *Introduction à l'analyse numérique des équations aux dérivées partielles*, Masson, 1992.
- [51] L. SCHWARTZ, *Théorie des distributions*, Hermann, 1966.
- [52] J. SIMON, *Démonstration constructive d'un théorème de G. de Rham*, C. R. Acad. Sci Paris, 316 (1993), pp. 1167–1172.
- [53] G. D. SMITH, *Numerical solution of partial differential equations : finite difference methods*, Oxford University Press, 1978.
- [54] P. K. SMOLARKIEWICZ, *The multi-dimensional Crowley advection scheme*, Monthly Weather Review, 110 (1982).
- [55] A. STANFORTH, J. CÔTÉ ET J. PUDYKIEWICZ, *Comments on "Smolarkiewicz's deformational flow."*, Monthly Weather Review, 115 (1987), pp. 894–900.
- [56] G. STRANG, *Introduction to applied mathematics*, Wellesley-Cambridge Press, 1986.
- [57] L. TARTAR, *Topics in nonlinear analysis*, Publications mathématiques d'Orsay, 1978.
- [58] R. TEMAM, *Navier-Stokes equations*, North Holland, 1984.
- [59] C. V. THEIS, *The relation between the lowering of the piezometric surface and the rate and duration of discharge of a well using groundwater storage*, Trans. American Geophysical Union, 16th Annual meeting, (1935), pp. 519–524.
- [60] R. S. VARGA, *Matrix iterative analysis*, Prentice-Hall, 1965.
- [61] VO-KHAC KHOAN, *Distributions, analyse de Fourier, opérateurs aux dérivées partielles*, Vuibert, 1970.
- [62] F. D. VUYST, *Un nouveau schéma numérique pour la résolution des équations d'Euler compressibles*, C. R. Acad. Sci Paris, 314 (1992), pp. 577–582.
- [63] S. WHITAKER, *A theoretical derivation of Darcy's law*, Transport in porous media, 1 (1986), pp. 3–25.
- [64] *Encyclopedic dictionary of Mathematics*, 1977.