Fonctionnalité d'un environnement orienté objet pour le développement de code éléments finis

Dominique Eyheramendy — Thomas Zimmermann

Laboratoire de mécanique des structures et milieux continus
Ecole Polytechnique Fédérale de Lausanne
DGC-A2, Ecublens
CH-1015 Lausanne, Suisse

Dominique.Eyheramendy@lsc.dgc.epfl.ch

RÉSUMÉ. *On se propose au travers de cette communication de présenter les aspects d'interface et de fonctionnalité d'un environnement mathématique symbolique permettant d'une part la dérivation de formulation d'éléments finis et d'autre part la programmation automatique dans un code numérique. On illustrera l'approche sur deux formulations pour l'élasticité linéaire en milieu incompressible et sur une formulation stabilisée pour le problème de Navier-Stokes.*

ABSTRACT. *Our purpose in this paper is to introduce aspects of userís and developer's graphical interface associated with operational aspects in an object-oriented environment for symbolic derivation and automatic programming of finite elements. The approach is illustrated first on the example of two formulations for the linear elasticity and second on the example of a stabilized formulation for the Navier-Stokes problem.*

MOTS-CLÉS : *formulations variationnelles, méthode éléments finis, élasticité linéaire pour milieu incompressible, Navier-Stokes, programmation orientée objet, calcul symbolique, calcul numérique.*

KEY WORDS : *variational formulations, finite element method, incompressible linear elasticity, Navier-Stokes, object-oriented programming, symbolic computation, numerical computation.*

1. Introduction

Les principes de la modélisation orientée objet appliquée à la dérivation de formulations éléments finis et à la génération automatique de code numérique ont été posés dans [ZIM 96, EYH 96a, EYH 98]. Le modèle hiérarchique permettant une approche hybride symbolique/numérique illustré en Figure 1, nommé FEM_Theory et proposé dans [EYH 96a (figure 1 p. 279)] est étudié à partir des manipulations mathématiques réalisées aux différentes étapes de la dérivation ; l'implantation automatique dans le code numérique objet FEM_Object est décrite dans [EYH 98]. En complément à [EYH 96b et 97c], on se propose d'aborder la dérivation d'un problème aux conditions de bord du point de vue d'un utilisateur voulant définir le nouveau type d'outil nécessaire à la dérivation du problème particulier. On va donc ainsi s'intéresser au niveau le plus élevé de l'élaboration de l'environnement, c'est-à-dire le plus proche du mécanicien et de son langage. On part du principe que l'on évolue dans un environnement dans lequel existent des objets représentant les différentes étapes de la dérivation d'un problème classique d'éléments finis : forme forte du problème (forme différentielle), forme variationnelle, forme faible, forme discrète ou semi-discrète. Ces objets sont déterminés de façon intuitive et s'appuient sur une étude approfondie de la démarche qui consiste à élaborer un modèle éléments finis. Il en résulte une certaine incertitude quant au choix de ces objets, qu'une étude étendue du genre de celle que nous proposons ici permet de lever, partiellement tout au moins. On va étudier la liste des opérations, que l'on espère générale, que l'on souhaite appliquer lors d'une dérivation (intégration par parties, dicrétisation du domaine,...).

Pour cela, dans un premier temps, une description de la gestion graphique dans FEM_theory est donnée (utilisation et développement). Puis, à partir de deux exemples de résolution d'un problème d'élasticité linéaire pour un milieu incompressible, et d'une formulation non-linéaire pour le problème de Navier-Stokes, les fonctions spécifiques liées à ces formulations sont décrites.

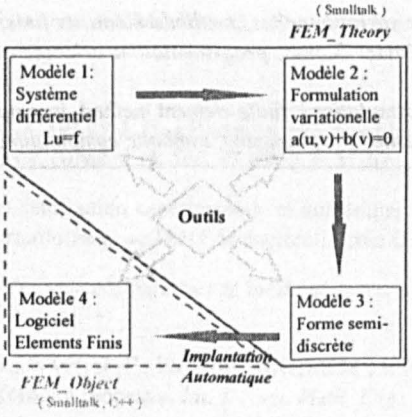

Figure 1. *Schéma général de l'approche hybride symbolique/numérique*

2. Gestion graphique dans un environnement orienté objet pour la dérivation symbolique de problèmes aux éléments finis

2.1. *Principe*

Les objectifs pour l'environnement graphique de développement symbolique sont d'une part d'avoir une vue des objets de la dérivation et, d'autre part de pouvoir leur envoyer des messages. Ainsi un système fenêtré simple a été développé. Celui-ci s'intègre très naturellement aux objets manipulés.

L'objet graphique principal (voir figure 2), fenêtre graphique instance de la classe **FEMTheoryMainView,** hérite de la classe de Smalltalk **Window** (voir [VIS 96a] et [WIN 95a] pour les outils graphiques utilisés) tout le comportement lié à la gestion de la partie graphique. La classe **FEMTheoryMainView** possède un attribut **currentObject** qui est l'objet courant manipulé par l'utilisateur. Un système de bouton poussoir permet l'envoi de messages à cet objet courant. Ces messages engendrent les opérations que l'utilisateur veut réaliser sur le problème qu'il veut résoudre. La résolution d'un problème commence donc à partir d'une instance de **FEMTheoryMainView**, celle-ci permettant l'instanciation d'une nouvelle formulation.

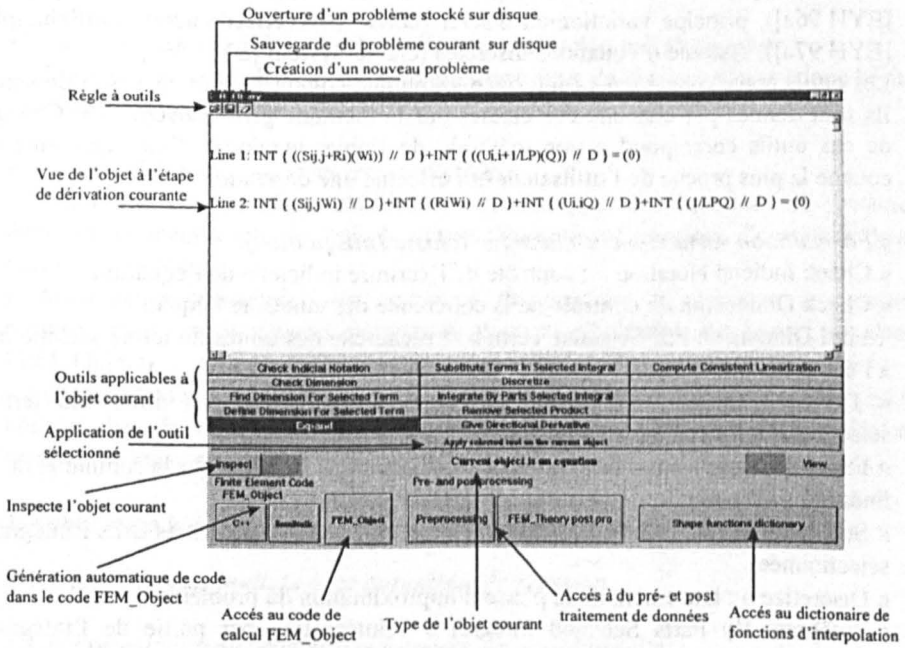

Figure 2. *Description de la fonctionnalité de l'écran de FEM_Theory*

2.2. *Description de l'interface*

L'utilisateur communique avec la formulation manipulée via l'interface graphique (voir la description sur la figure 2). Le choix des messages envoyés se fait par le biais de la barre de sélection d'outil, l'envoi étant contrôlé par le bouton d'envoi de message. L'affichage des outils disponibles pour l'objet courant est géré par l'interface (affichage des outils dans la barre de sélection d'outils), mais seul l'objet courant connaît sa liste d'outils (méthode *giveArrayOfTools*). En ce qui concerne l'affichage, chaque objet courant est capable de recomposer la chaîne de caractères qui le représente (à partir des objets qui le composent, voir [EYH 96b] pour les notations utilisées), l'affichage étant géré par la fenêtre principale ; celle-ci donne également le type de l'objet courant (figure 2).

2.3. *Les objets manipulés dans l'interface*

L'utilisateur peut avoir à manipuler un certain nombre de modèles de formulations dans l'interface (figure 2). Ils peuvent être de cinq types :
- pour le problème continu : formulation variationnelle classique (classe **IntEquation** [EYH 96a]), principe variationnel (classe **VariationalPrinciple** [EYH 97a])
- pour le problème discret : équation discrète (classe **DiscretizedEquation** [EYH 96a]), principe variationnel discret (classe **DiscretizedVariationalPrinciple** [EYH 97a]), système d'équations discrètes (classe **System** [EYH 96a])

Les outils apparaissant dans l'interface graphique (figure 2) sont décrits ci-dessous. Ils sont donnés par chacune des classes par la méthode *giveArrayOfTools*. Chacun de ces outils correspond à une méthode de l'objet manipulé. Cela représente la couche la plus proche de l'utilisateur qui effectue une dérivation.

• *Formulation variationnelle classique (classe **IntEquation**)*
« Check Indicial Notation » : contrôle de l'écriture indicielle de l'équation
« Check Dimension » : contrôle de la cohérence des unités de l'équation
« Find Dimension For Selected Term » : recherche des unités du terme sélectionné à l'écran
« Define Dimension For Selected Term » : définition des unités du terme sélectionné à l'écran dans l'équation
« Expand » : application des règles de distributivité du produit sur la somme et de la linéarité de l'intégrale
« Substitute Terms In Selected Integral » : remplacement de termes dans l'intégrale sélectionnée
« Discretize » : lancement de la phase d'approximation du problème
« Integrate By Parts Selected Integral » : intégration par partie de l'intégrale sélectionnée et application du théorème de Gauss sur ce résultat
« Remove Selected Product » : permet de supprimer un terme dans la formulation

« Compute Consistent Linearization » : calcule la forme linéarisée consistante de la formulation variationnelle (exception faite pour l'instant des lois de comportement non linéaires)

• *Principe variationnel (classe **VariationalPrinciple**)*

Les outils de cette classe sont les mêmes que ceux de la super-classe (donc hérités de **IntEquation**). Mais un outil est rajouté à la liste.

« Minimize » : permet de miniser la fonctionnelle par rapport à ces inconnues

• *Equation discrète (classe **DiscretizedEquation**)*

« Transpose » : permet de transposer l'équation discrète

« Invoke Linear Independence » : applique la propriété d'indépendance linéaire de l'équation par rapport aux termes virtuels

« Shape Function Replacing » : remplacement des fonctions d'interpolation par leur expression en repère local (accès à un dictionnaire de fonctions d'interpolation)

« Rename » : permet un changement de notation du terme discret sélectionné

« Remove Selected Product » : permet de supprimer le terme sélectionné dans la formulation

« Add A Perturbation Term » : permet de rajouter des termes dans la formulation discrète du problème (type Galerkin/moindres carrés par exemple, voir partie 3.2)

•*Principe variationnel discret (classe **DiscretizedVariationalPrinciple**)*

Les outils de cette classe sont les mêmes que ceux de la super-classe (donc hérité de **DiscretizedEquation**).

•*Système d'équations discrètes (classe **System**)*

Les outils de cette classe sont les mêmes que ceux des équations du système, donc ici les mêmes que ceux de la classe **DiscretizedEquation**. Cependant deux outils sont ajoutés.

« Assemble » : assemble les équations du système et regroupe les termes correspondant aux inconnues (nécessaire pour la génération de code C++ dans FEM_Object)

« Reorder Elemental Contributions » : permet de modifier la position des inconnues nodales dans le vecteur des inconnues (nécessaire pour rendre la forme discrète compatible avec le code numérique FEM_Object)

3. Application à un problème d'élasticité linéaire en milieu incompressible

3.1. *Formulation pénalisée avec paramètre de pression*

3.1.1. *Formulation du problème et formulation variationnelle*

La formulation pénalisée proposée fait intervenir une variable auxiliaire p, paramètre de pression. Celle-ci est décrite dans [HUG 87], les mêmes notations sont adoptées ici.

Etant données les fonctions f, g et F, trouver (u, p) déplacement et paramètre de pression, (u, p) respectant les conditions de régularité nécessaires sur le domaine Ω ($\Omega \subset \Re^{n_{sd}}$, n_{sd} est la dimension de l'espace, $\partial\Omega$ bord de Ω défini tel que $\partial\Omega = \partial_1\Omega \cup \partial_2\Omega$), tels que :

$\sigma_{ij,j} + f_i = 0$ dans Ω avec les conditions de bord suivantes :

$\sigma_{ij} n_j = F_i$ sur $\partial_2\Omega$

$u_{i,i} + \frac{p}{\lambda} = 0$ dans Ω

$u_i = g_i$ sur $\partial_1\Omega$

On définit $\mathcal{W}$ et $\mathcal{S}$, respectivement les espaces solution et virtuel pour les déplacements, et $\mathcal{P}$ l'espace solution et virtuel des paramètres de pressions ($\mathcal{W}$, $\mathcal{S}$ et $\mathcal{P}$ sont définis avec les conditions de régularité et de bords classiques pour les fonctions), une formulation variationnelle correspondant à ce problème s'écrit :
Etant donné f, trouver $(u,p) \in \mathcal{S} \times \mathcal{P}$ tels que $\forall (w,q) \in \mathcal{W} \times \mathcal{P}$:

$$\int_\Omega (\sigma_{ij,j} + f_i) w_i \, dv + \int_\Omega (u_{i,i} + \frac{p}{\lambda}) q \, dv = 0$$

3.1.2. *Fonctionnalité liée à une formulation pénalisée d'un milieu élastique linéaire incompressible*

La dérivation de cette formulation est décrite complètement dans [EYH 98]. Les outils et fonctions décrits dans [EYH 96a] pour un problème d'élasticité en dynamique permettent à l'utilisateur de générer la forme symbolique discrète pour le problème. Il est cependant nécessaire d'ajouter au niveau de la génération de code la possibilité de faire de l'intégration numérique réduite sur le terme de pénalisation. Ainsi, lors de l'opération de codage automatique, une fenêtre supplémentaire proposera un schéma d'intégration (Gauss) pour chacune des matrices élémentaires de la forme discrète. Du point de vue programmation, seules deux fenêtres de dialogue offrant le choix d'intégration numérique sont nécessaires ; de plus, la gestion de l'attribut **numericalIntegrationScheme** (type de schéma d'intégration) des classes **DiscretizationMatrix** (forme discrète élémentaire) et **Integral** (intégrale) est légèrement modifiée.

3.2. *Formulation stabilisée de type Galerkin/moindres carrés*

3.2.1. *Formulation variationnelle stabilisée*

On se propose d'évaluer une formulation stabilisée décrite dans [FRA 87] pour le problème de Stokes. Cette formulation est également capable de représenter un milieu solide incompressible.
Partant de la formulation variationnelle classique pour un milieu incompressible donnée par :
Etant donnée la fonction f, trouver $(u,p) \in \mathcal{S} \times \mathcal{P}$ tels que $\forall (w,q) \in \mathcal{W} \times \mathcal{P}$:

$$\int_\Omega (\sigma_{ij,j} + f_i) w_i \, dv + \int_\Omega u_{i,i} q \, dv = 0$$

La forme approximée de Galerkin du problème est donnée par :
Soit les sous-espaces $\mathcal{S}^h$, $\mathcal{W}^h$ et $\mathcal{P}^h$, respectivement approximation de $\mathcal{S}$, $\mathcal{W}$ et $\mathcal{P}$. Le nouveau problème s'écrit alors :
Soit f, trouver $(u^h, p^h) \in (\mathcal{S}^h \times \mathcal{P}^h)$ tels que $\forall\, (w^h, q^h) \in (\mathcal{W}^h \times \mathcal{P}^h)$:

$$R((u^h, p^h);(w^h, q^h)) = -\int_\Omega C_{ijkl}\varepsilon_{kl}(u^h)\varepsilon_{ij}(w^h)dv + \int_\Omega p_h w^h_{i,i}dv + \int_\Omega u^h_{i,i}q^h dv - \int_\Omega f_i w^h_i dv = 0$$

La formulation stabilisée proposée s'obtient en ajoutant à la forme de Galerkin précédente des termes de moindres carrés (voir [FRA 87]) pondérés par deux coefficients (paramètres de stabilisation) dépendant du maillage et des caractéristiques du milieu :
Soit f, trouver $(u^h, p^h) \in (\mathcal{S}^h \times \mathcal{P}^h)$ tels que $\forall\, (w^h, q^h) \in (\mathcal{W}^h \times \mathcal{P}^h)$:

$$R((u^h, p^h);(w^h, q^h)) + \sum_{\Omega^e \in \Omega_h} \left[\int_{\Omega^e} (C_{ijkl}\varepsilon_{kl,j}(u) - p_{,i} - f_i) D_1 (C_{ijkl}\varepsilon_{kl,j}(w) - q_{,i})\, dv - \int_{\Omega^e} u_{i,i}\ D_2\ w_{i,i}\ dv \right] = 0$$

où D_1 et D_2 sont les paramètres de stabilisation.

3.2.2. *Fonctionnalité liée à une formulation stabilisée d'un milieu élastique linéaire incompressible*

Par rapport aux outils et fonctions décrits dans [EYH 96a] seule une fenêtre de dialogue pour les termes discrets doit être programmée. Bien entendu, l'outil permettant d'introduire ces termes doit être ajouté à la liste des outils des objets discrets, **DiscretizedEquation** et **DiscretizedVariationalPrinciple**. Ce type de développement ne nécessite que quelques heures grâce à l'utilisation d'outils de développement graphiques dans l'environnement Smalltalk (développement d'interface graphique WindowBuilder Pro [WIN 96a]).

3.3. *Etude comparative des résultats numériques*

Les nouveaux éléments sont testés sur le problème classique de la cavité remplie d'un matériau incompressible, décrit figure 3. L'environnement doit permettre cette comparaison rapide de formulation pour un même problème.

Les paramètres constitutifs et numériques choisis sont : $\mu = 1$ and $\lambda = 10^7$.

L'élément Q1/Q0 est testé sur un maillage 12*12. Les champs de déplacement et pression sont montrés dans la figure 4. Une phase de post-traitement des résultats est nécessaire pour la pression, mais ici aucun artifice de lissage n'est utilisé. Comme ceci est connu, cet élément est peu performant pour l'évaluation des pressions, mais donne en revanche des résultats corrects pour les déplacements ; une phase de lissage des résultats en pression peut cependant être suffisante si l'on ne cherche pas à avoir des résultats de qualité en pression. Cette formulation laisse apparaître un phénomène de *checkerboard* (échiquier), problème lié à la stabilité de la formulation.

Cette formulation doit être comparée à la formulation proposée par Franca dans [FRA 87] et dérivée dans le paragraphe 0. L'élément est également testé sur le problème de la cavité. Cet élément Q1/Q1 est testé sur un maillage 17*17. Les paramètres constitutifs choisis ici sont : $\mu = 1$. La méthode pour le calcul des paramètres de stabilisation « D1 » et « D2 » est implantée manuellement ($D_1 = \frac{\delta_1 h^2}{2\mu}$ et $D_2 = 2\mu\delta_2$). Pour les paramètres de stabilisation, on a choisi $\delta_1 = 0.5$ et $\delta_2 = 0$. Ces résultats sont en accord avec ceux trouvés dans [FRA 87] ; une évaluation de la formulation peut être effectuée.

Une comparaison sommaire sur la forme générale entre les résultats obtenus ici (voir figure 4 et figure 5) est faite. Les deux formulations conduisent à des résultats comparables et acceptables en ce qui concerne les champs de déplacement. Par contre, comme cela est bien connu, l'élément Q1/Q0 a de faibles performances pour l'évaluation des champs de pression (noter qu'aucun lissage n'est réalisé dans le post-traitement des résultats). Un phénomène *d'échiquier* apparaît dans l'évaluation des pressions. L'élément Q1/Q1 présenté dans ce paragraphe donne de bons résultats en pression, semble un peu trop diffusif (résultats comparés avec des calculs réalisés ultérieurement, et avec ceux de [FRA 87]). Ceci provient vraisemblablement du fait que le maillage utilisé sur cet exemple numérique est trop grossier. On s'est contenté, dans cet exemple, de montrer la possibilité de rapidement évaluer qualitativement deux formulations pour le problème de Stokes en incompressible. La performance du code généré, bien que faible, permet tout de même de réaliser des tests intéressants.

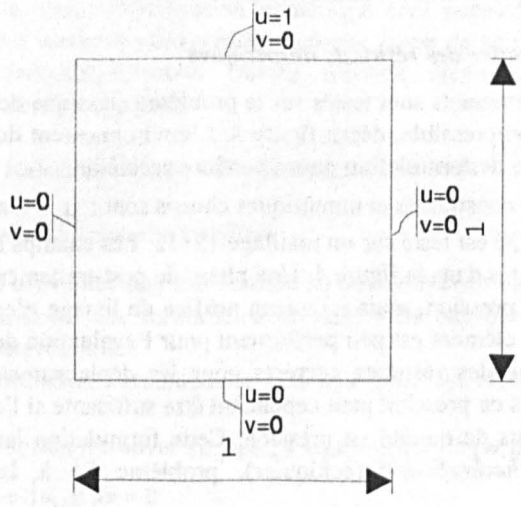

Figure 3. *Description du problème de la cavité*

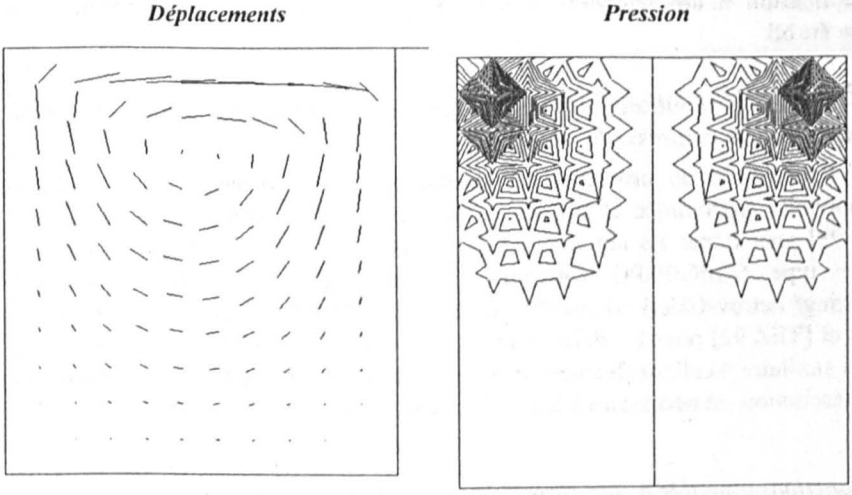

Figure 4. *Résultats numériques pour la formulation pénalisée (élément Q1/Q0)*

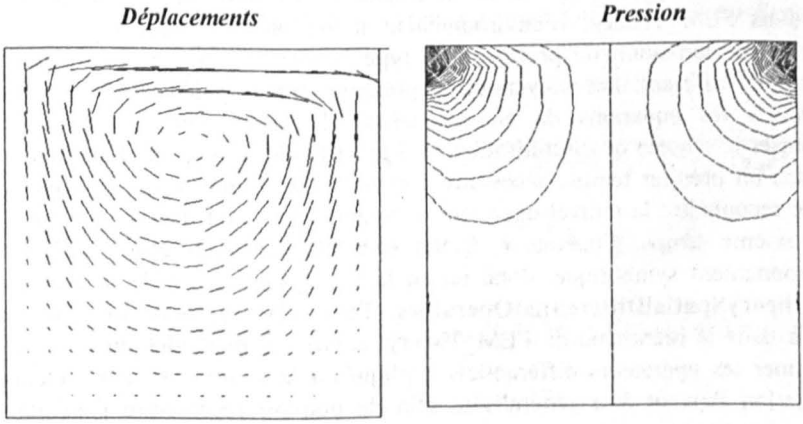

Figure 5. *Résultats numériques pour la formulation stabilisée (élément Q1/Q1)*

Remarque 1. Les applications montrées dans [EYH 97b] montrent la généricité de l'approche décrite dans ce paragraphe en ce qui concerne les formulations stabilisées par ajouts de termes de moindres carrés.
Remarque 2. Les mêmes formulations peuvent être utilisées pour la résolution d'écoulement de Stokes en mécanique des fluides (voir [HUG 87]).

4. Application à un problème non linéaire : problème de Navier-Stokes en régime établi

4.1. *Formulation stabilisée pour le problème de Navier-Stokes en régime établi pour un fluide incompressible*

La forme forte du problème est montrée dans la tableau 1, la formulation variationnelle approximée et stabilisée dans la tableau 2 (voir le paragraphe 3 et [EYH 98] pour toutes les notations utilisées). Le schéma de stabilisation choisi ici est de type SUPG/PSPG (Streamline Upwind/Petrov-Galerkin et Pressure-Stabilizing/ Petrov-Galerkin) suggéré dans [TEZ 92] (voir [EYH 97b] pour plus de détails et [TEZ 92] pour les définitions). La dérivation, menée à bien en 14 étapes, est très similaire à celles effectuées dans les paragraphes précédents. Noter que, ici, une linéarisation est nécessaire à la résolution approchée du problème.

4.2. *Fonctionnalité liée à une formulation stabilisée pour le problème de Navier-Stokes en régime établi pour un fluide incompressible*

Dans [EYH 97c], un schéma de linéarisation a été introduit en tant que nouvel outil dans FEM_Theory. L'environnement a été étendu naturellement afin de permettre la dérivation de problème de type Navier-Stokes. De nombreux types d'équations différentielles peuvent être représentés et manipulés dans FEM_Theory, y compris les équations de Navier-Stokes. Il est simplement nécessaire de généraliser le schéma de discrétisation pour la partie de convection des équations. Il est, dans un premier temps, nécessaire d'étendre les capacités de l'environnement afin de reconnaître le nouvel opérateur différentiel introduit dans l'équation. Dans un deuxième temps, l'opérateur discret correspondant doit être introduit dans l'environnement symbolique, donc ici en tant que sous-classe de la classe mère **FEMTheorySpatialDifferentialOperators**. Tout cela représente un changement minime dans la hiérarchie de FEM_Theory. Aussi, les méthodes du produit pour déterminer les opérateurs différentiels appliqués à la solution et à la fonction de pondération doivent être généralisées afin de pouvoir reconnaître l'opérateur de convection ; les formes élémentaires correspondant à ces opérateurs sont ajoutées ici pour un champ vectoriel.

Une nouvelle fonction est introduite au niveau de la forme continue du problème (forme variationnelle c'est-à-dire classe **IntEquation**), l'outil de linéarisation « Compute Consistent Linearisation » . On peut noter que la fonctionnalité de stabilisation utilisée dans le paragraphe précédent est suffisamment générale pour être utilisée ici.

4.3. *Résultats numériques*

Certains termes de la linéarisation sont négligés dans la contribution à la matrice tangente afin de simplifier les calculs numériques. L'opération qui consiste à omettre des termes dans la matrice tangente est réalisée sur la forme approximée. Un

Etant donné f, trouver u vitesse et p pression satisfaisant aux exigences de continuités suffisantes, telles que :

Ωin $R^{n_{sd}}$

$n_{sd} = 2$

Equation d'équilibre :

$\sigma_{ij,j} + f_i = \rho\, u_j u_{i,j}$ on Ω

Equation de continuité :

$u_{i,i} = 0$ on Ω

Conditions de bords :

$\sigma_{ij} n_j = F_i$ on $\partial_2\Omega$

$u_i = \bar{u}_i$ on $\partial_1\Omega$

avec $\partial\Omega = \partial_1\Omega \cup \partial_2\Omega$

Relation de comportement :

$\sigma_{ij} = -p\delta_{ij} + 2\mu\, \varepsilon_{ij}(u)$ on Ω

Loi cinématique :

$\varepsilon_{kl}(u) = \frac{1}{2}(u_{k,l} + u_{l,k})$ on Ω

Tableau 1. *Forme forte du problème pour un écoulement de Navier-Stokes en incompressible pour un régime établi*

Etant donné f, trouver $(u^h, p^h) \subset (\mathcal{S}^h \times \mathcal{P}^h)$ tel que pour tout $(w^h, q^h) \subset (\mathcal{W}^h \times \mathcal{P}^h)$ (notations définies paragraphe 3) :

$$\int_\Omega \rho\, u_j^h u_{i,j}^h dv - \int_\Omega 2\mu\varepsilon_{ij}(u^h)\varepsilon_{ij}(w^h)dv + \int_\Omega p_h w_{i,i}^h dv + \int_\Omega u_{i,i}^h q^h dv - \int_\Omega f_i w_i^h dv$$

$$+ \sum_{\Omega^e \in \Omega^h} \left[\int_{\Omega^e} (\rho\, u_j^h u_{i,j}^h - 2\mu\varepsilon_{ij,j}(u^h) + p_{,i}^h - f_i)\tau_{mom}\, (\rho\, u_j^h w_{i,j}^h - 2\mu\varepsilon_{ij,j}(w^h) + q_{,i}^h) dv \right] = 0$$

avec :

$$\tau_{mom} = \left(\left(\frac{2|u|}{h} \right)^2 + \left(\frac{4\mu}{h^2} \right)^2 \right)^{-1/2} \quad \text{(pour des détails voir [BEH 94])}$$

Tableau 2. *Formulation approximée et stabilisée pour le problème de Navier-Stokes en régime établi*

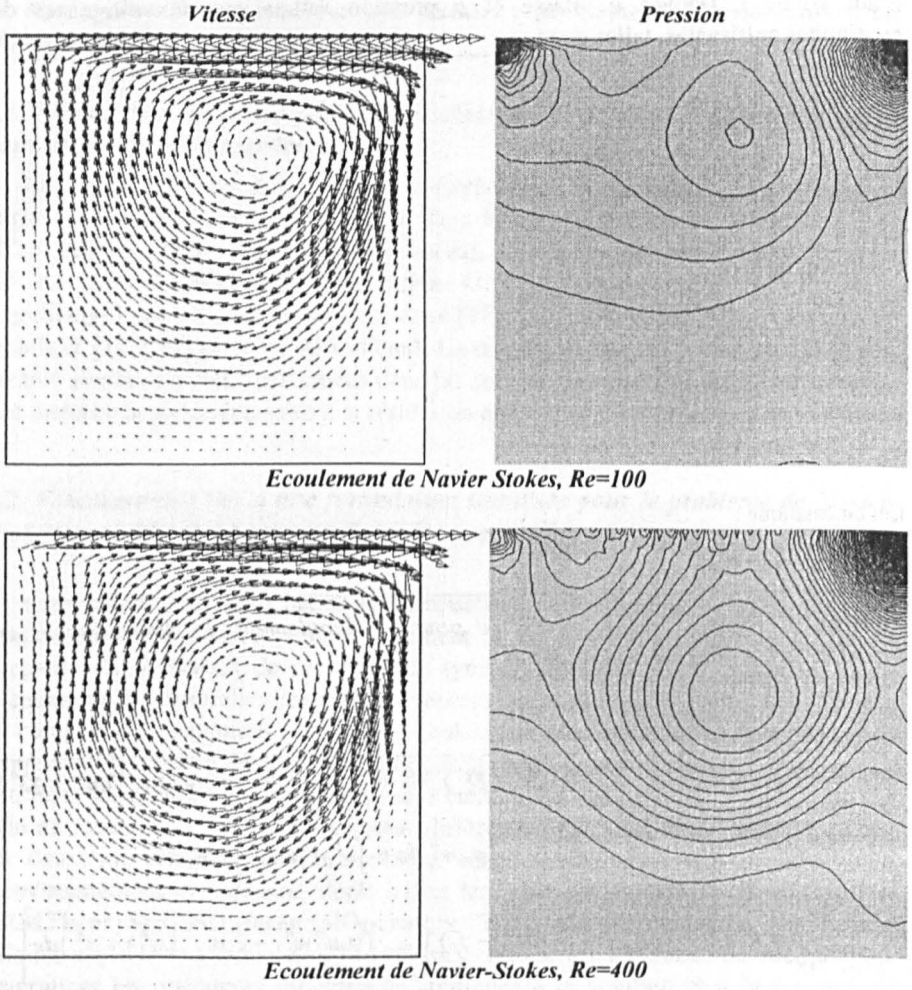

Figure 6. *Résultats numériques pour la formulation stabilisée pour le problème de Navier-Stokes*

élément classique de type quadrilatère est choisi. Les champs de vitesse et pression sont interpolés sur l'élément aux quatre nœuds de coin. Une intégration de Gauss à quatre points est adoptée. La forme du paramètre de stabilisation est proposée dans [BEH 94] dans laquelle on omet le terme dépendant du temps. Dans un souci de simplification, on omet également dans le terme de stabilisation la partie provenant de l'équation de continuité. Cela peut être fait pour les cas de bas Reynolds (voir les applications numériques pour le problème de Stokes au paragraphe 3). Ici, les résultats numériques ont été obtenus en utilisant un schéma itératif pour lequel une technique de *ramping* est adoptée, c'est-à-dire que le nombre de Reynolds est augmenté lentement depuis un écoulement de type Stokes. A chaque incrément du

nombre de Reynolds, la convergence est vérifiée. Les deux premières itérations sont de type Picard (la matrice de rigidité est utilisée comme matrice tangente). Cela permet d'assurer la convergence, si l'on est trop éloigné de la solution au Reynolds courant. Les itérations suivantes sont de type Newton (la matrice tangente obtenue lors de la dérivation est utilisée). On montre ainsi la flexibilité du code obtenu automatiquement. Les résultats numériques sur l'exemple classique de la cavité (problème décrit au paragraphe 3) pour un maillage de 1 024 éléments sont montrés dans la figure 6. Les résultats sont comparables à ceux trouvés chez d'autres auteurs (par exemple [GIA 82] et [SCH 83], ou [TEZ 92a] et [STO 97] pour des calculs similaires).

5. Conclusion

Dans cet article, les principes de programmation liés à l'interface graphique et à ses liens avec les objets de la dérivation symbolique de code éléments finis ont été décrits. La décentralisation d'une partie des tâches (génération de l'image de l'objet et gestion des outils) vers les entités manipulées par l'utilisateur, associée à l'utilisation d'outils de développement d'interface graphique très performants, ont permis de simplifier toute la gestion de l'environnement fenêtré. Au travers de la description complète de trois formulations relativement représentatives d'une classe de problèmes à caractères mixte et convectif, on a énuméré les méthodes nécessaires à l'élaboration de modèles éléments finis. Cette étape est fondamentale dans la détermination des *bons objets* pour le problème envisagé. Une démarche similaire à celle que nous proposons dans cet article, appliquée à un nouveau problème peut éventuellement remettre en cause, tout ou partie, le modèle hiérarchique proposé dans [EYH 96a] pour l'environnement de dérivation symbolique. On met ainsi l'accent sur l'aspect évolutif que peut avoir un environnement orienté objet pour l'élaboration de code éléments finis.

Remerciements

Cette étude est financée par le Fond National Suisse pour la Recherche Scientifique, subside n° 20-45697.95.

6. Bibliographie

[BEH 94] M. BEHR, T.E. TEZDUYAR, Finite element solution strategies for large-scale flow simulation, *Comput. Methods Appl. Mech. Engrg.*, 112 (1994) pp. 3-24.

[EYH 96a] D. EYHERAMENDY, Th. ZIMMERMANN, Object-oriented finite elements : II. A symbolic environment for automatic programming, *Comput. Methods Appl. Mech. Engrg.*, 132 (1996) pp. 277-304.

[EYH 96b] D. EYHERAMENDY, Th. ZIMMERMANN, Object-oriented Finite Element Programming : An interactive environment for symbolic derivations, Application to an Initial Boundary Value Problem, Advances in Engineering Software 27 (1996) 3-10.

[EYH 97a] D. EYHERAMENDY, Th. ZIMMERMANN Dérivations symboliques pour code éléments finis - Application à un problème d'élasticité, Actes du 3e Colloque national en calcul des structures de Giens, (1997).

[EYH 97b] D. EYHERAMENDY, Object-Oriented Finite Elements Programming : Symbolic derivation and automatic programming, PhD thesis report 1752, Ecole Polytechnique Fédérale de Lausanne, (1997).

[EYH 97c] D. EYHERAMENDY, Th. ZIMMERMANN, Intégration d'une approche variationnelle pour la méthode des éléments finis : Application à un problème de convection non-linéaire, Soumis à la *Revue Européenne des éléments finis* (numéro spécial), (1997).

[EYH 98] D. EYHERAMENDY, Th. ZIMMERMANN, Object-oriented finite elements : III. Theory and application of automatic programming, *Comput. Methods Appl. Mech. Engrg.*, 154 (1998) pp. 41-68.

[FRA 87] L. P. FRANCA, New mixed finite element methods, Ph.D. thesis report, Stanford University, 1987.

[GHI 82] U. GHIA, K.N. GHIA, C.T. SHIN, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, *J. Computational Physics*, vol. 48 (1982) 387-411.

[HUG 87] T. J. R. HUGHES, *The Finite Element Method*, Prentice Hall, (1987).

[SCH 83] R. SCHREIBER and H.B. KELLER, Driven cavity flows by efficient numerical techniques, *J. Computational Physics* , vol. 49 (1983) 310-333.

[SHA 88] F. SHAKIB, Finite Element analysis of the compressible Euler and Navier-Stokes equations, Ph.D. thesis report, Stanford University, 1988.

[STO 97] M. STORTI, N. NIGRO, S. INDELSOHN, Equal order interpolations: a unified approach to stabilize the incompressible and advective effects, *Comput. Methods Appl. Mech. Engrg.*, 143 (1997) pp. 317-331.

[TEZ 92a] T.E. TEZDUYAR, S. MITTAL, S.E. RAY, R. SHIH, Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements, *Comput. Methods Appl. Mech. Engrg.*, 95 (1992) pp. 221-242.

[TEZ 92b] T.E. TEZDUYAR, M. BEHR, J. LIOU, A new strategy for finite element computations involving moving boundaries and interfaces - The deforming-spatial-domain/space-time procedure : I. The concept and the preliminary numerical tests, *Comput. Methods Appl. Mech. Engrg.*, 94 (1992) pp. 339-351.

[VIS 95a] VisualSmalltalk Enterprise - 32 Bit Pure Object-Oriented Programming System, User's guide, ParkPlace Digitalk, (1995).

[WIN 96] WindowBuilder Pro/V 3.1, Tutorial and Reference Guide, ObjectShare Systems Inc., (1996).

[ZIM 96] Th. ZIMMERMANN, D. EYHERAMENDY, Object-oriented finite elements : I. Principles of symbolic derivation and automatic programming, *Comput. Methods Appl. Mech. Engrg.*, 132 (1996) pp. 277-304.