# Stabilized Finite Elements in Geomechanics

by

Stéphane Commend

Ing. civil dipl. EPFL (1994)

Submitted to the Laboratoire de mécanique des Structures et des milieux Continus
in partial fulfillment of the requirements for the degree of

Dr ès Sc. Techn.

at the

ECOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

April 2001



Signature of Author .....
Laboratoire de mécanique des Structures et des milieux Continus
April 2001

Accepted by .....
Dr. Thomas Zimmermann
PhD Supervisor

# Contents

# Summary

This work deals with stabilized finite element formulations in geomechanics. Many low-order elements exhibit a tendency to volumetric mesh locking at the incompressible elastic limit or in plastic computations. Literature proposes different remedies to overcome this deficiency: mixed displacement-pressure formulations, selective integration techniques - among which the so-called $\overline{\mathbf{B}}$ method, and enhanced assumed strains [12][51] are popular ones. The $\overline{\mathbf{B}}$ method is known to handle the incompressible case in both elastic and plastic regimes. The standard $\overline{\mathbf{B}}$ method described in [28] does not however predict correctly the dilatant plastic case, and such a formulation for simple elements like the linear triangle is not available. The enhanced assumed strain (EAS) method handles dilatant plastic flows, but again there is no EAS formulation for linear triangles.

Mixed displacement-pressure formulations will work correctly provided that the Babuška & Brezzi [1][5] conditions are fulfilled. These conditions put a restriction on the fields interpolating the displacement and the pressure. A simpler way to assess good behavior of an element is to have it undergo a patch test [61][64]. If the element fails to pass the test, spurious oscillations in the pressure field may happen, and displacements will be subject to kinematic constraints: the element locks. In particular, this is the case for the quadrilateral element with bilinear interpolation (Q4) and the linear triangle (T3) with equal order interpolation fields for both displacement and pressure, as also shown by the constraint count rule [28].

Using simple (bi-)linear elements is often desirable in order to reduce cost of computation and simplify mesh generation, mainly in three-dimensional cases. A simple formulation capable of handling all of incompressible elasticity as well as incompressible or dilatant plasticity would be therefore of great use. This formulation should also allow mixing different element types in the same mesh. In this work we try to establish such a formulation, based on what has been developped in the last 20 years mainly in the field of fluid mechanics.

In chapter 1, a state-of-the-art of stabilized methods applied to advection-diffusion, fluid and solid mechanics is presented, with particular emphasis on the two types of stabilization that will help us build our finite-element scheme: Galerkin Least-Squares (GLS) [16][30] and Finite Increment Calculus (FIC) [42][43] formulations. The second chapter is a recall of governing principles of mechanics, constitutive modelling, as well as standard finite-element techniques, for both elasticity and plasticity, from which our formulation will be derived. The $\overline{\mathbf{B}}$ method is also described. The third chapter is the kernel of this work: a mixed formulation applied to elasticity is derived, and the expected problems of the standard Galerkin formulation are illustrated on the driven cavity flow example. A stabilization scheme based on the GLS technique is proposed and validated. The discussion then focalizes on the weighting part of the stabilizing terms, including the stabilization factor $\tau = \frac{\alpha^e (h^e)^2}{2\mu}$ [27]. It is shown that neglecting some terms in this weighting

part (therefore getting a modified-GLS "SUPG-like" method) can simplify the derivation. A nonlinear formulation is then derived in order to handle plasticity. Linearization of the weighting and the residual part is carefully exposed. Then a FIC-scheme derivation is carried out, based on the original FIC method. It provides on the one hand a tentative physical justification for the nature of the stabilizing terms, and on the other hand a directional character is introduced in the formulation. Last but not least, the Laplacian pressure operator scheme (also called "$\mathbf{K}'_{pp}$ only") is introduced [44]. This formulation is obtained by adding to the original mixed formulation stabilizing terms built on the Laplacian of the pressure.

Three plane strain benchmark problems validate the approaches and are used to calibrate the stabilization factor: the thick cylinder test loaded by an internal pressure, a vertical cut stability analysis and finally the bearing capacity of a strip footing. Plastic models considered include von Mises and Drucker-Prager. It is shown that the modified-GLS and the Laplacian pressure operator scheme ("$\mathbf{K}'_{pp}$ only") methods handle the three problems correctly with $0.1 < \alpha^e < 1$. The advantages of the FIC-scheme method have yet to be proven, but in the majority of cases the results obtained are in good agreement with the two other methods. The fourth chapter deals with the numerical implementation of the proposed formulation. The code is written in C++; it extends a linear version of an object-oriented finite-element code developped at EPFL [14]. The basic features of object-oriented programming are recalled, then the hierarchy of classes is discussed, with particular emphasis on the main classes which handle the nonlinear computation and the stabilization process. The methods which make a stabilized element possible to compute its elemental contributions are shown. Finally conclusions are drawn, including recommendations for the choice of stabilization.

# Version abrégée

Ce travail traite des formulations stabilisées d'éléments finis en géomécanique. Les éléments finis d'ordre inférieur (fonctions d'interpolation linéaires) ont tendance à verrouiller en présence d'élasticité incompressible ou de plasticité. La littérature propose plusieurs remèdes pour surmonter ce problème: les formulations mixtes déplacement-pression, les techniques de sous-intégration (parmi lesquelles on trouve la méthode $\overline{\mathbf{B}}$) ainsi que les méthodes de déformations assumées (Enhanced Assumed Strains ou EAS, [12][51]) en font partie. La méthode $\overline{\mathbf{B}}$ traite de manière correcte le cas incompressible en régime élastique comme en régime plastique, mais le problème avec la méthode décrite par Hughes [28] est d'une part son incapacité à prédire correctement le cas de la plasticité dilatante, et d'autre part l'absence d'une telle formulation pour un élément simple comme le triangle linéaire. La méthode EAS traite pour sa part de manière satisfaisante les cas de régimes plastiques dilatants, mais à nouveau il n'existe pas de formulation EAS pour les triangles linéaires.

Les conditions mathématiques de Babuška & Brezzi [1][5] sont nécessaires pour que les formulations mixtes déplacement-pression fonctionnent de manière correcte. Ces conditions imposent une restriction sur l'ordre d'interpolation des champs de déplacement et de pression. Un moyen plus simple de s'assurer du bon comportement d'un élément fini est de lui faire passer un "patch-test" [61][64]. Si l'élément ne passe pas ce test, le champ de pression sera généralement sujet à des oscillations parasitaires et les déplacements seront soumis à des contraintes cinématiques entraînant le verrouillage de l'élément. C'est le cas en particulier pour le quadrilatère (Q4) bilinéaire et pour le triangle (T3) linéaire possédant une interpolation d'ordre égal pour les champs de déplacement et de pression. On utilise aussi parfois une approche heuristique, la règle du décompte des contraintes ("constraint count rule", [28]), permettant de déterminer l'aptitude d'un élément à se comporter de manière adéquate dans des situations incompressibles.

L'utilisation d'éléments d'ordre inférieur est toutefois souvent désirable afin de réduire le temps de calcul et de simplifier la génération des données, particulièrement en 3D. Une formulation permettant de résoudre les problèmes d'élasticité incompressible et d'élastoplasticité à l'aide d'éléments simples (linéaires) serait donc précieuse. Cette formulation devrait également permettre le mélange de différents types d'éléments dans le même maillage. On tente dans ce travail d'établir une telle formulation sur la base de travaux effectués durant les 20 dernières années, principalement dans le domaine de la mécanique des fluides.

Dans le premier chapitre, une brève revue de l'utilisation des méthodes stabilisées dans les domaines de l'advection-diffusion et de la mécanique numérique des fluides est effectuée. On s'attarde principalement sur deux approches qui vont nous aider à établir notre propre formulation stabilisée: l'approche de Galerkin aux moindre-carrés (Galerkin Least-Squares ou GLS, [16][30]) et l'approche de calcul aux incréments finis (Finite Increment Calculus ou FIC,

[42][43]). Le deuxième chapitre est consacré à un rappel des principes fondamentaux régissant la mécanique des milieux continus, de certaines lois constitutives (élastoplasticité) et de formulations d'éléments finis standards. La méthode $\overline{\mathbf{B}}$ est également introduite à ce niveau. Une formulation mixte appliquée à l'élasticité est dérivée dans le troisième chapitre, et les problèmes de la formulation standard sont illustrés sur un exemple d'écoulement dans une cavité (problème de Stokes ou d'élasticité incompressible). Une méthode de stabilisation basée sur la technique GLS est proposée puis validée. La discussion se porte ensuite sur la pondération des termes stabilisants, et plus particulièrement sur la facteur de stabilisation $\tau = \frac{\alpha^e (h^e)^2}{2\mu}$ [27]. On montre qu'on peut simplifier la dérivation de la formulation en négligeant une partie du terme de pondération (on parle alors de formulation GLS-modifiée ou SUPG) sans affecter la qualité des résultats obtenus. Une formulation non-linéaire est ensuite dérivée afin de traiter le cas de l'élastoplasticité. Le processus de linéarisation des termes de pondération et des résidus est expliqué en détail. On s'attache ensuite à la dérivation d'une formulation basée sur l'approche FIC (on l'appelle FIC-scheme). Cette formulation nous fournit d'une part une tentative de justification physique de la nature des termes stabilisants, et d'autre part elle introduit un caractère directionnel. Enfin, une formulation appelée Laplacien de la pression (Laplacian Pressure Operator Scheme ou LPOS, [44]) est introduite. Cette formulation est obtenue en ajoutant à la forme mixte matricielle originelle des termes émanant de la divergence de l'équation d'équilibre débouchant sur le Laplacien de la pression.

Trois exemples sont testés afin de valider les différentes approches présentées dans ce travail: le cas du cylindre épais chargé par une pression interne, l'analyse de stabilité d'un talus vertical et la capacité portante d'une fondation superficielle. Les modèles plastiques considérés sont von Mises et Drucker-Prager. On peut conclure que les formulation GLS-modifiée et LPOS fournissent des résultats équivalents et satisfaisants pour un choix du paramètre $\alpha^e$ tel que $0.1 \leq \alpha^e \leq 1$. Les avantages de la formulation directionnelle (FIC-scheme) sont encore à prouver, mais dans la majorité des cas les résultats obtenus à l'aide de cette formulation correspondent aux deux autres approches. Le quatrième chapitre traite de l'implémentation numérique de l'approche stabilisée. Le code, écrit en C++, est basé sur un code linéaire orienté-objet développé à l'EPFL [14]. Les principes de base de la programmation orientée-objet sont rappelés, puis la hiérarchie des classes est fournie et les classes principales sont décrites en détail. On énumère ensuite les méthodes permettant à un élément stabilisé de calculer et de fournir ses contributions élémentaires au système global. Enfin le chapitre 5 conclut cette étude et fournit des recommandations sur le type optimal de stabilisation.

# Acknowledgements

I would like to express my deepest thanks to the following people:

- Dr Thomas Zimmermann, my PhD supervisor, for... everything
- Prof. François Frey, director of the Laboratory of Structural and Continuum mechanics (LSC) at the EPFL where I have spent seven fruitful years
- Prof. T.J.R. Hughes, director of the Mechanics and Computation Division in the School of Engineering at Stanford, for his welcome during my stay in Stanford University. My Californian thanks also go to all the post-graduate students I have met there, among them Gonzalo Feijoo, Chandler Fulton, Fabio Guarnieri, Assad Oberai and finally to Mary Driscoll
- Dr Andrzej Truty, for introducing me to nonlinear finite element analysis
- The members of the jury, for accepting to review this work
- All the members of the LSC, past and present, for their friendship and for their support, among them David Alvarez, Olivier Bernard, Patricia Bomme, José Diaz, Dominique Eyheramendy, César Falla-Luque, Richard Frenette, Milan Jirašek, Krzyzstof Podleś, Blaise Rébora, Jean-Luc Sarf, Birgitte Seem, Marc-André Studer, Laurent Vernier and Wajd Zimmermann
- The Swiss National Science Foundation, for grant 21-49404.96
- The Swiss National Committee of Large Dams for their financial support

Finally a warmful thank you to my family, especially my parents, and to all of my friends. You know what you mean to me.

Lausanne, 2nd March 2001 Stéphane Commend

# Notations

The following symbols and abbreviations are used in this thesis:

| | |
|---|---|
| $\sigma_{ij}$ | stress tensor |
| $\delta_{ij}$ | Kronecker's symbol |
| $\boldsymbol{\sigma}$ | stress vector with components (in 3D): $\boldsymbol{\sigma} = \{\sigma_{11}, \sigma_{22}, \sigma_{12}, \sigma_{33}, \sigma_{13}, \sigma_{23}\}^T$ |
| $\mathbf{1}$ | Kronecker's vector (in 3D): $\mathbf{1} = \{1, 1, 0, 1, 0, 0\}^T$ |
| $\varepsilon_{ij}$ | total strain tensor |
| $\varepsilon^p_{ij}$ | plastic strain tensor |
| $\boldsymbol{\varepsilon}$ | total strain vector with components ordered as above |
| $\boldsymbol{\varepsilon}^p$ | plastic strain vector with components ordered as above |
| $u_i$ | displacement component |
| $\mathbf{u}$ | displacement vector |
| $p$ | mean pressure |
| $w_i$ | displacement weighting function component |
| $\mathbf{w}$ | displacement weighting function vector |
| $q$ | pressure weighting function |
| $D_{ijkl}$ | elastic constitutive tensor |
| $\overline{D}_{ijkl}$ | deviatoric projection of the elastic constitutive tensor |
| $\overline{\mathbf{D}}^{uu}, \overline{\mathbf{D}}^{up}, \overline{\mathbf{D}}^{pu}, \overline{\mathbf{D}}^{pp}$ | elastoplastic tangent constitutive submatrices |
| $\overline{\mathbf{K}}^{uu}, \overline{\mathbf{K}}^{up}, \overline{\mathbf{K}}^{pu}, \overline{\mathbf{K}}^{pp}$ | stiffness submatrices |
| $\mathbf{F}_u, \mathbf{F}_p$ | right-hand side subvectors |

| | |
|---|---|
| $N_i$ | displacement shape function |
| $\mathbf{N}$ | displacement shape function vector |
| $\widetilde{N}_i$ | pressure shape function |
| $\widetilde{\mathbf{N}}$ | pressure shape function vector |

| | |
|---|---|
| $(\cdot)^i$ | some quantity $(\cdot)$ at iteration $i$ |
| $(\cdot)_n$ | some quantity $(\cdot)$ at step $t_n$ |
| $\Delta(\cdot)$ | increment of some quantity $(\cdot)$ |
| $(\cdot)_{,i}$ | indicates derivation of $(\cdot)$ with respect to $x_i$ |
| $\overrightarrow{\nabla}(\cdot)$ | gradient of some quantity $(\cdot)$ |
| $(\cdot)_{(i,j)}$ | symmetric part of the $(\cdot)$ gradient $= \frac{(\cdot)_{i,j}+(\cdot)_{j,i}}{2}$ |
| $\lvert(\cdot)\rvert$ | absolute value of $(\cdot)$ |
| $(\cdot)^e$ | some quantity $(\cdot)$ at an element node |
| $(\cdot)^h$ | some quantity $(\cdot)$ approximated inside an element |
| $(\cdot)^{el}$ or $(\cdot)^e$ | elastic part of some quantity $(\cdot)$ |
| $(\cdot)^{pl}$ or $(\cdot)^p$ | plastic part of some quantity $(\cdot)$ |
| $(\cdot)^{tr}$ | elastic trial value of some quantity $(\cdot)$ |

# Chapter 1

# Introduction

## 1.1 Methodology

The aim of this work is to find a stable finite element formulation in the field of geomechanics. Standard finite element formulations exhibit pathologies such as spurious oscillations in the stress field or locking in the displacement field as soon as they deal with incompressible or dilatant media. Selective integration or mixed methods allow us to overcome some of these deficiencies, but while on the one hand the $\overline{\mathbf{B}}$ method described in [28] is restricted to incompressibility and particular elements, on the other hand interpolation fields for mixed displacement-pressure methods are restricted by a mathematical condition (Babuška & Brezzi [1][5]) and therefore convenient choices such as equal low-order interpolation fields for both unknowns are prevented. Stabilization methods were developped mainly in the past two decades, motivated at the early stage by the need of overcoming spurious oscillations in convection-dominated flows (first presented by Brooks & Hughes in 1982 [4]). The idea was to circumvent the Babuška & Brezzi condition by adding stability to the finite element scheme without upsetting consistency. One of these methods, called Galerkin Least-Squares (GLS) [16][30], consists in adding least-square terms based on the residual of the equilibrium equation to the original mixed formulation. Stabilizing terms can be put in the following form introducing the intrinsic time scale (commonly called stabilization parameter) $\tau$:

$$\sum_{e}^{n_{el}} \int_{\Omega^e} \tau \left(\mathcal{L}\mathbf{w}^h\right)^T \left(\mathcal{L}\mathbf{u}^h - f\right) d\Omega$$

$\left(\mathcal{L}\mathbf{u}^h + \mathbf{f}\right)$ is an abstract notation for the left-hand side of the differential equation governing the problem where $\mathcal{L}$ is a differential operator and $\mathbf{w}^h$ is the weighting function corresponding to $\mathbf{u}^h$. In this work we try to apply this idea to the geomechanics field. The development goes through the following steps:

- Extension of an exisiting linear object-oriented code to nonlinear finite element analysis.

The implementation of different plastic models (von Mises and Drucker-Prager), elements (Q4, T3) and formulations ($\overline{\mathbf{B}}$, mixed $\mathbf{u}$-$p$) has been carried out. Benchmarks confirm the shortcomings of standard methods and therefore a new stable formulation is sought.

- Development and application of appropriate GLS formulations to our benchmark problems. The full GLS stabilizing scheme has been tried first on the incompressible elasticity problem. Systematic tests have then shown that results of equal quality could be obtained neglecting the displacement part in the weighting term, therefore keeping only the gradient of the pressure (giving rise to a modified-GLS scheme, form-similar to the SUPG approach). This is confirmed on incompressible and dilatant plasticity. Another argument in favor of ignoring the displacement contribution in the weighting part arises from linearization difficulties in the plastic case. The nature of the so-called stabilization factor - a contribution in the stabilizing terms whose justification can be established by dimensional analysis - has also been scrutinized.

- In the context of linear computational fluid dynamics, stability of these enhanced finite elements can be proven through error analysis, while consistency is retained as the residual of the equilibrium equation is present in the stabilizing terms. In the nonlinear case, an error analysis of the same kind is very tedious, not to say impossible. Therefore, another scheme based on the Finite Increment Calculus (FIC) approach [42][43] was implemented in order to provide a tentative physical justification of the presence of stabilizing terms, emanating from equilibrium over a finite-dimensional balance domain with high-order Taylor approximations. We have brought up the similarities that this approach (FIC-scheme) shares with modified-GLS. It also introduces a directional character in our formulation that may be desirable in some situations like directional plasticity (multilaminate model) or finite strains.

- Another approach brought up by Brezzi & Pitkäranta [6] and used for the first time by Pastor et al [44] in the soil mechanics context stabilizes the pressure equation by adding the pressure Laplacian resulting from the divergence of the equilibrium equation (Laplacian pressure operator scheme (LPOS) also cited in [42]). This approach has also been implemented in order to compare results with modified-GLS and FIC-scheme approaches.

- Four different benchmarks have been carried out in order to validate the different approaches. First, the modified-GLS scheme has been calibrated on an incompressible elasticity (or Stokes) problem: the driven cavity flow. Then, three elastoplastic tests have been performed: the thick-cylinder loaded by an internal pressure, the stability analysis of a vertical cut and the bearing capacity of a strip footing. These benchmarks have shown that both the modified-GLS and the LPOS methods give the same results: they stabilize the pressure field and avoid volumetric locking even in the presence of different element types in the same mesh. The FIC-scheme approach almost fulfills the same expectations but sometimes has trouble to converge and a real need for its directional character has yet to be found. Finally, these benchmarks have also allowed us to calibrate the stabilization parameter $\tau$.

## 1.2 Governing Equations

For sake of clarity, let us review here the governing equations for three problems, namely the 1D advection-diffusion problem, the incompressible steady-state Navier-Stokes equations and the mixed form of nonlinear solid mechanics.

### 1.2.1 1D advection-diffusion problem

The 1D advection-diffusion governing differential equation can be written:

$$-u\nu\phi_{,x} + k\phi_{,xx} + f = 0 \tag{1.1}$$

where the comma denotes differentiation. In Equation 1.1, $u$ is a known velocity field, $\nu$ the advective parameter, $\phi$ the transported unknown (temperature for instance), $k$ is the conductivity and $f$ the source term.

### 1.2.2 Incompressible steady-state Navier-Stokes equations

The multi-dimensional incompressible Navier-Stokes equations of motion governing the mechanics of a Newtonian fluid can be expressed as:

$$-\rho u_j u_{i,j} + \sigma_{ij,j} + f_i = 0 \tag{1.2}$$
$$u_{i,i} = 0 \tag{1.3}$$

with the following constitutive equation expressing the stress $\sigma_{ij}$ in function of the velocity $u_i$ and the pressure $p$:

$$\sigma_{ij} = -p\delta_{ij} + 2\mu\varepsilon_{ij} = -p\delta_{ij} + 2\mu u_{(i,j)} \tag{1.4}$$

$\rho$ is the fluid's density, $\mu$ its viscosity, $\lambda$ the other Lamé parameter. Neglecting the convective term $\rho u_j u_{i,j}$ in Equation 1.2, we get the incompressible Stokes equations:

$$\sigma_{ij,j} + f_i = 0 \tag{1.5}$$
$$u_{i,i} = 0 \tag{1.6}$$

This set of equations is known to govern as well incompressible elasticity in the field of solid mechanics, where $u_i$ is now the displacement field and $\mu$ the shear modulus.

### 1.2.3 Mixed formulation for solid mechanics

If we consider a mixed approach (displacement-pressure) of solid mechanics in a general sense (plasticity included), the following set of equations can be used:

$$\sigma_{ij,j} + f_i = 0 \tag{1.7}$$
$$u^e_{i,i} - \frac{p}{K} = 0 \tag{1.8}$$

The first equation results from equilibrium, while the second is a constitutive equation for the pressure (therefore called pressure equation). This latter equation expresses the fact that the mean pressure is proportional to the volumetric part of the elastic strain [64]: $p = K\varepsilon^e_v = K u^e_{i,i}$, where $K$ is the elastic bulk modulus. The incremental constitutive equation linking $\Delta\sigma_{ij}$, $\Delta u_i$ and $\Delta p$ reads:

$$\Delta\sigma_{ij}(\mathbf{u}, p) = \overline{D}_{ijkl}\left[\Delta\varepsilon_{kl}(\mathbf{u}) - \Delta\varepsilon^p_{kl}(\mathbf{u}, p)\right] + \delta_{ij}\Delta p \tag{1.9}$$

with $\overline{\mathbf{D}}$ the deviatoric projection of the constitutive matrix (defined in chapter 3 in Equation 3.3).

## 1.3 Goal of Stabilization

In advection-diffusion or Navier-Stokes problems, there are two main sources of potential numerical instabilities. The first is due to the presence of the convective term, which can be the cause of spurious oscillations in the solution when it becomes predominant with respect to the diffusive term. The second source of numerical problems lies in the choice of interpolation for the unknown fields. It is well known that, for instance, equal-order interpolation for both the displacement and the pressure will produce strong oscillations in the pressure field. This prevents us from using standard (bi-)linear elements which are of good practical use, easy to generate with a preprocessor and cheap speaking of computational costs. In the context of nonlinear analysis of solids involving plasticity, both of these problems are present. The first of the above problems is difficult to notice at first sight, although the plastic part of the increment of deformation presents similarities with the convective term, but the second problem arises as soon as equal interpolation fields are chosen. Along with pressure oscillations, we denote a tendency to volumetric mesh locking either for incompressible elasticity or plasticity. Stabilization applied to soil plasticity should therefore:

- eliminate spurious oscillations in the pressure field
- overcome locking phenomena in all cases (incompressible elasticity, incompressible or dilatant plasticity)
- allow the use of low-order equal interpolation elements

Yet another point is that common techniques that have been developed cannot usually accomodate different types of elements in the same mesh, like for instance linear triangles and bilinear quads. We therefore also would like the stabilized approach to:

- allow the mixture of different element types in the same mesh

We will now make a review of stabilized methods applied to advection-diffusion, fluid mechanics and solid mechanics. A particular attention will be given to two approaches, namely the ones derived by Hughes and Oñate, as they are cornerstones for our later developments.

## 1.4 Review of Stabilized Methods

### 1.4.1 Origins of Stabilized Approaches

Numerical methods for solving problems governed by partial differential equations do not always converge to the desired result. Methods such as finite differences or standard finite elements exhibit an unstable character when certain conditions are fulfilled. This is the case for instance in convection-dominant advection-diffusion problems, or in the incompressible limit for fluid flows or elasticity. This is also the case by extension in nonlinear problems such as the one which is of interest here, namely elastoplasticity applied to geomechanics.

In the finite differences applied to advection-diffusion cases context, an artificial (or balancing) diffusion is often added to the original scheme for two reasons: counter-balance the overly anti-diffusive character of the finite differences stencil and fight numerical instabilities in convection-dominant cases. The presence of this balancing diffusion can also be established by upwinding the advective term, giving rise to a stable formulation that may lose its accuracy (becoming overly diffusive).

In 1982, Brooks and Hughes introduced a multi-dimensional generalization of optimal upwinding schemes in their paper [4] about Streamline Upwind/Petrov-Galerkin (SUPG) formulations. They added streamline-upwind contributions to the weighting part of the weak form of the problem that contributed to enhance the stability of the method without compromising its consistency as these contributions weight the residual of the governing differential equation and therefore vanish when the approximate solution is replaced by the actual one. The "streamline" term finds its justification in the fact that upwinding effects are only necessary in the direction of the flow and should not generate a spurious crosswind diffusion. They also extended the method to the incompressible Navier-Stokes equations, giving rise to the use of the application of such methods to mixed problems such as the one, involving displacements and pressures, governing equilibrium and continuity in the field of elastoplastic solid mechanics (Equations 1.7-1.9). Johnson et al [35] analyzed the SUPG method from the mathematical point of view in the context of the multi-dimensional advection-diffusion equation and established optimal convergence rates.

In 1984, Brezzi & Pitkäranta [6] introduced another kind of stabilization for the Stokes problem, based on changing the discrete incompressibility condition by adding a term of the form $\sum h_e^2 \int \overrightarrow{\nabla} p_h \overrightarrow{\nabla} q_h d\Omega$, where $h_e$ is a characteristic length of the element, $p_h$ the approximated pressure field and $q_h$ the corresponding pressure weighting function.

### 1.4.2 Mixed Problems Issues

Another source of instabilities in a mixed displacement-pressure formulation is the possible inappropriate combination of interpolation fields. Several techniques can be used to assess good behavior of a finite element. From a mathematical point of view, the Babuška & Brezzi (BB) conditions [1][5] are requirements which guarantee that the finite element solution exists, is unique and converges at optimal rate. These conditions are rather technical, and difficult to comprehend without strong mathematical background. A simpler procedure called constraints counts [28] permits to assess whether or not an element will lock. Another alternative to the fulfillment of BB conditions is the patch test [61][64]. Typically, a finite element based on equal low-order continuous interpolations for displacements and pressures does not pass this test and therefore produces locking in the displacement field and oscillations in the pressure field.

### 1.4.3 Stabilization in the CFD and Advection-Diffusion Fields

After the pioneering work done by Brooks & Hughes [4], a Hughes, Franca & Balestra [27] contribution on a stable Petrov-Galerkin formulation for the Stokes problem is of particular interest for us as governing equations match the ones of incompressible elasticity. In this paper, a stabilization factor of the form $\tau = \frac{\alpha^e (h^e)^2}{2\mu}$ is introduced and provides a model for our own stabilization factor. Tezduyar [57] calls this stabilization scheme pressure-stabilizing/Petrov-Galerkin (PSPG). In his paper he introduces a method to handle incompressible Navier-Stokes equations.

A generalization of the concept of stabilization has been made in the late eighties by Hughes, Franca et al [16][17][30], giving rise to a class of methods called Galerkin Least-Squares (GLS). The *least-squares* terminology is justified by the fact that the stabilizing term results from the derivative of a least-squares potential [31]. Various forms of this stabilization scheme have been applied to advection-diffusion and computational fluid dynamics [19][20] among which a modification of the GLS formulation has been proposed by Douglas & Wang [13] on the Stokes problem: a change of sign in a part of the weighting term of the stabilizing term produces better results for high-order interpolations. For linear interpolations however both methods coïncide. A comparison study between SUPG and GLS formulations has been carried out on the incompressible Navier-Stokes problem by Hannani et al [23]. Franca et al [18] have introduced a method called Galerkin Gradient Least-Squares (G$\overrightarrow{\nabla}$LS) obtained from adding to the Galerkin method a term obtained from a least-squares form of the gradient of the Euler-Lagrange equation.

Oñate [40] has introduced the concept of balance over a finite-dimension domain to justify the existence of stabilizing terms. He applied his method to advection-diffusion and fluid flow problems and showed the equivalence of his concept and methods such as SUPG and GLS. Later he generalized the notion of Finite Increment Calculus (FIC) procedure [43] on incompressible Navier-Stokes equations.

### 1.4.4 Stabilization Parameter $\tau$ Selection

In the context of incompressible Navier-Stokes equations, Franca & Frey [20] introduce the following form of the stabilizing factor:

$$\tau = \frac{h^e}{2|u|}\xi(\mathrm{Re}_e) \tag{1.10}$$

$$\mathrm{Re}_e = \frac{m|u|h^e}{4\mu} \tag{1.11}$$

$$\xi(\mathrm{Re}_e) = \begin{cases} \mathrm{Re}_e & \text{if } 0 \leq \mathrm{Re}_e < 1 \\ 1 & \text{if } \mathrm{Re}_e \geq 1 \end{cases} \tag{1.12}$$

$$m = \min\left\{\frac{1}{3}, 2C_k\right\} \tag{1.13}$$

For low Reynolds numbers $\mathrm{Re}_e$, $\tau$ reduces to $\frac{m(h^e)^2}{8\mu}$. Computing $C_k$ involves error estimates methods which can be carried out in linear cases (see for instance Harari & Hughes [24]) but would be rather difficult in nonlinear situations. A calibration of $\tau = \frac{\alpha^e(h^e)^2}{2\mu}$ (by analogy with [27]) through parameter $\alpha^e$ is therefore more appropriate to our elastoplastic problems.

Hughes [32] has developed a global methodology regrouping the concepts of multiscale models, bubble functions and stabilized methods through Green's functions giving rise to an expression for the stabilization factor. Several other authors have shown an equivalence between enriched finite elements with bubble functions and stabilized methods (see for instance Brezzi et al [7], Russo [48] or Zienkiewicz & Taylor [64]).

Finally in [40] and [43], Oñate gives an iterative scheme for the computation of the stabilization parameter based on a diminishing residual procedure.

### 1.4.5 Solid and Structural Mechanics Applications

Several structural problems have received attention in the stabilized methods research field. For instance, Timoshenko beams have been studied by Grosh & Pinsky [22] with the help of the GLS and the GGLS (Generalized GLS) methods, and Reissner-Mindlin Plate Theory has been scrutinized by Hughes & Franca [29].

Truty et al looked into the stabilization of mixed elastoplasticity [58][59] and two-phase consolidation problems [60]. Still in the soil plasticity context Pastor et al [44] stabilize the pressure equation by adding the pressure Laplacian (Laplacian pressure operator scheme (LPOS) also cited in [42]) resulting from the divergence of the equilibrium equation. Pastor et al [45] have introduced a fractional-step algorithm yielding a stable low-order formulation for problems in undrained soils. Similar time integration schemes - which yield stabilized forms when steady-state conditions are recovered - have been scrutinized by Zienkiewicz & Wu [62] and lately a general algorithm, the characteristic based split procedure, has been introduced in the CFD field (see [65]).

## 1.5 Galerkin Least-Squares (Hughes' Approach)

We will consider here the 1D advection-diffusion problem, governed by Equation 1.1. First, we will examine the classical finite differences method and compare it with the upwind method which cures the instability of the classical scheme in advection-dominated cases. The exact artifical diffusion method will be discussed, and give rise to a Petrov-Galerkin finite-element scheme. Finally the Galerkin Least-Squares (GLS) approach will be introduced as a generalization of the Petrov-Galerkin scheme. This development is based on [31].

**Remark 1** *if we set the advective parameter $\nu$ to be equal to 1, the exact solution of Equation 1.1 without source term ($f = 0$) is:*

$$\phi(x) = c_1 + c_2 \exp(P\frac{x}{L}) \tag{1.14}$$

*where $L$ is the length of the considered domain, $c_1$ and $c_2$ are constants defined by boundary conditions and $P = uL \,/\, k$ is the* ***Peclet number****.*

### 1.5.1 Finite Differences

#### Classical Scheme (Central Differences)

Consider a uniform mesh of $A_{\max}$ segments of size $h$ (see Figure 1-1). At node A, the differential operators present in Equation 1.1 are approximated by difference quotients given by:

$$u\phi_{,x}(x_A) \simeq u\frac{\phi_{A+1} - \phi_{A-1}}{2h} \tag{1.15}$$

$$k\phi_{,xx}(x_A) \simeq k\frac{\phi_{A+1} - 2\phi_A + \phi_{A-1}}{h^2} \tag{1.16}$$

and thus we now have to solve, for $1 \leq A \leq A_{\max} - 1$:

$$-u\frac{\phi_{A+1} - \phi_{A-1}}{2h} + k\frac{\phi_{A+1} - 2\phi_A + \phi_{A-1}}{h^2} + f_A = 0 \tag{1.17}$$

Unfortunately, for numerically advection-dominated cases, i.e. when the **element Peclet number** $\alpha$ defined by:

$$\alpha = \frac{uh}{2k} \tag{1.18}$$

is greater than 1, the solution oscillates (see Figure 1-2).

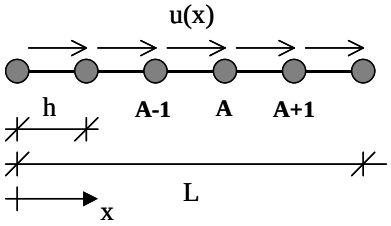



Figure 1-1: Uniform mesh

**Upwind Approximation**

Upwinding the advective term, i.e. expressing it with a difference quotient taken upwind of point $A$ (with respect to the velocity $u$ direction):

$$u\phi_{,x}(x_A) \simeq u\frac{\phi_A - \phi_{A-1}}{h} \tag{1.19}$$

leads to the stabilization of the numerical solution (see Figure 1-2), although we can notice that upwind differences are overly diffusive.

An interesting feature of the upwind scheme is that it can be written in the following form:

$$u\phi_{,x}(x_A) \simeq u\frac{\phi_A - \phi_{A-1}}{h} = u\frac{\phi_{A+1} - \phi_{A-1}}{2h} - \frac{uh}{2}\frac{\phi_{A+1} - 2\phi_A + \phi_{A-1}}{h^2} \tag{1.20}$$

Regrouping Equations 1.16 and 1.20 allows us to write the numerical problem as:

$$-u\frac{\phi_{A+1} - \phi_{A-1}}{2h} + \left(k + \frac{uh}{2}\right)\frac{\phi_{A+1} - 2\phi_A + \phi_{A-1}}{h^2} + f_A = 0 \tag{1.21}$$

and when we compare this form with Equation 1.17, we see that upwinding the advective term has introduced an artificial diffusion $\frac{uh}{2}$ into the central difference scheme.

**Exact Artificial Diffusion Method**

We wish now to solve the following finite difference scheme (without a source term):

$$-u\frac{\phi_{A+1} - \phi_{A-1}}{2h} + \left(k + \overline{k}\right)\frac{\phi_{A+1} - 2\phi_A + \phi_{A-1}}{h^2} = 0 \tag{1.22}$$

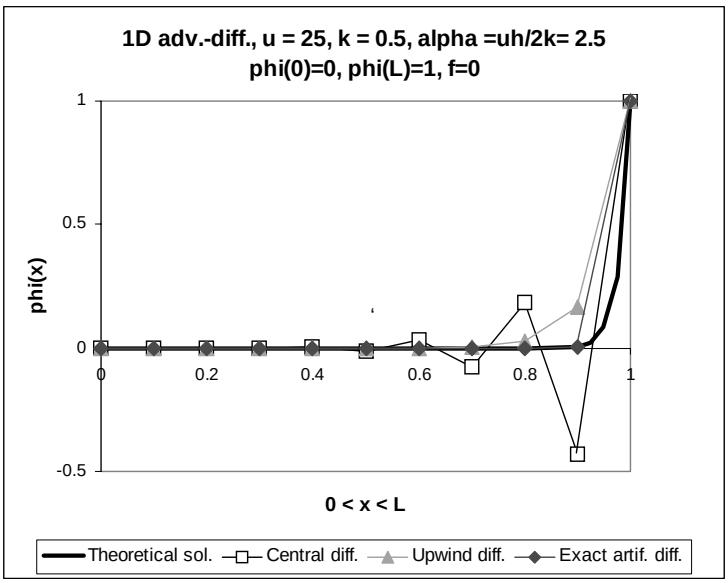



Figure 1-2: Central & upwind differences and exact artificial diffusion method for $\alpha = 2.5$

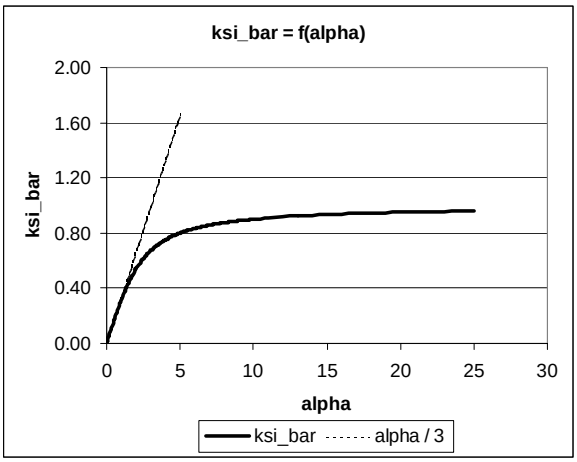



Figure 1-3: $\overline{\xi}$ plotted versus $\alpha$

where $\overline{k}$ is determined when approximating at best the exact solution (Equation 1.14) by:

$$\overline{k} = \frac{1}{2} uh\overline{\xi} \tag{1.23}$$

where we retrieve as in [9] ($\alpha$, the element Peclet number, is defined by Equation 1.18):

$$\overline{\xi} = \coth \alpha - \frac{1}{\alpha} \tag{1.24}$$

Figure 1-3 plots $\overline{\xi} = f(\alpha)$. We notice that $\lim_{\alpha \to 0} \overline{\xi} = \frac{\alpha}{3}$. When we apply this exact artificial diffusion scheme to our problem, we see that the exact solution is satisfied exactly at the nodes (see Figure 1-2).

**Remark 2** *central differences correspond to* $\overline{k} = 0 \Longleftrightarrow \overline{\xi} = 0$.

**Remark 3** *upwind differences correspond to* $\overline{k} = \frac{uh}{2} \Longleftrightarrow \overline{\xi} = 1$.

### 1.5.2 Petrov-Galerkin Method

We will now derive a variational method similar to the exact artificial diffusion method. Starting from the original differential equation 1.1, a weak form can be obtained by premultipying it by a weighting function $w$ and integrating over the domain:

$$\int_0^L w\left(-u\phi_{,x} + k\phi_{,xx} + f\right) dx = 0 \tag{1.25}$$

$w = 0$ at $x = 0$ and $x = L$ (as Dirichlet boundary conditions are assumed). Integrating the advective and the diffusive terms by parts leads to:

$$\int_0^L \left(w_{,x}u\phi - w_{,x}k\phi_{,x}\right) dx + \int_0^L wf dx = 0 \tag{1.26}$$

This form is unfortunately as ineffective as the central differences method in advection-dominated cases. In order to correct this fact, we consider adding to the left-hand side of Equation 1.26 the following term:

$$\sum_{A=1}^{A_{\max}} \int_{x_{A-1}}^{x_A} p\left(-u\phi_{,x} + k\phi_{,xx} + f\right) dx \tag{1.27}$$

where $p$ is a linear function of $w$. Integrating Equation 1.26 by parts and adding the terms defined by Equation 1.27 leads to:

$$\sum_{A=1}^{A_{\max}} \int_{x_{A-1}}^{x_A} (w+p)\left(-u\phi_{,x} + k\phi_{,xx} + f\right) dx + \sum_{A=1}^{A_{\max}-1} w(x_A)(k\phi_{,x}(x_A^+) - k\phi_{,x}(x_A^-)) = 0 \tag{1.28}$$

from which we can identify the following Euler-Lagrange equations:

- $-u\phi_{,x} + k\phi_{,xx} + f = 0$ in every element interior $]x_{A-1}, x_A[$
- $k\phi_{,x}(x_A^+) = k\phi_{,x}(x_A^-)$: continuity of flux across element interfaces

The whole point is to define $p$ based on the exact artificial diffusion method:

$$p = \tau u w_{,x} \tag{1.29}$$

where $\tau$, the **element intrinsic time scale**, is given by:

$$\tau = \frac{\overline{k}}{|u|^2} \tag{1.30}$$

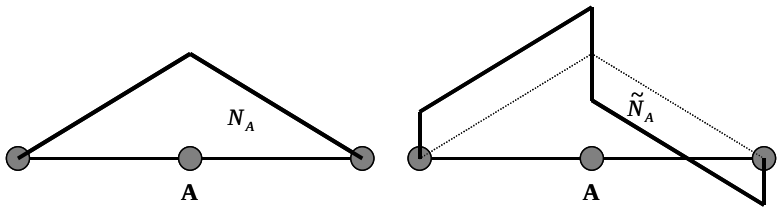



Figure 1-4: Galerkin and upwind Petrov-Galerkin weighting functions

**Remark 4** *in advection-dominated cases,* $\overline{\xi} \to 1$ *and therefore* $\tau = \frac{h}{2|u|}$.

So technically this Petrov-Galerkin method consists in adding to the original Galerkin scheme terms of the form:

$$\sum_{A=1}^{A_{\max}} \int_{x_{A-1}}^{x_A} \tau \mathcal{L}_{adv} w \left( \mathcal{L}\phi - f \right) dx \tag{1.31}$$

where:

$$\mathcal{L}_{adv} = u \frac{\partial}{\partial x} \tag{1.32}$$

This leads to discontinuous weighting functions $\widetilde{N}_A$ as illustrated in Figure 1-4. We also define for the advection-diffusion case:

$$\mathcal{L}_{diff} = -k \frac{\partial^2}{\partial x^2} \tag{1.33}$$

$$\mathcal{L} = \mathcal{L}_{adv} + \mathcal{L}_{diff} \tag{1.34}$$

### 1.5.3 Galerkin-Least Squares

Unfortunately, $\mathcal{L}$ cannot be always decomposed into an advective and a diffusive part. The Galerkin Least-Squares method is a generalization of the Petrov-Galerkin concept and consists in taking the entire $\mathcal{L}$ as part of the weighting term:

$$\sum_{A=1}^{A_{\max}} \int_{x_{A-1}}^{x_A} \tau \mathcal{L} w \left( \mathcal{L}\phi - f \right) dx \tag{1.35}$$

## 1.6 Finite Increment Calculus (FIC) Method (Oñate's Approach)

Oñate [42][43] has discussed a natural way to retrieve stabilization terms introducing the concept of balance over a finite domain. In contrary to what has been discussed before in the so-called "Hughes approach", where stabilizing terms were added after discretization in order to modify the original Galerkin scheme, the Finite Increment Calculus approach uses the standard Galerkin method applied to a **modified** differential equation. It will later be shown that both approaches applied to our problem provide the same Euler-Lagrange equations.

The two main reasons of having a close look at this formulation are the following: on the one hand it should help us justify "physically" the form of the stabilizing terms given in the GLS formulation, and on the other hand it should also introduce a directional character into the stabilizing terms.

In the following development we consider the one-dimensional advection-diffusion problem. A similar discussion could be made over a 2D or 3D domain or considering the Navier-Stokes equations. In section 3.3, we will derive our own governing equations (equilibrium and balance of mass) following the same guidelines.

### 1.6.1 Governing Equations

Consider the standard 1D advection-diffusion problem over a domain of length $l$ (see Figure 1-5) with boundary conditions $\phi = \overline{\phi}$ at $x = 0$ and $q = \overline{q}$ at $x = l$. $f(x)$ is a source term and $u(x)$ is the known velocity field which transports the unknown $\phi$. Figure 1-6 illustrates a finite-dimension balance domain $AB$ with $\overline{AB} = h$. Balance of fluxes between point $A$ and point $B$ reads:

$$\sum \text{Fluxes} = [\text{Fluxes at } A] - [\text{Fluxes at } B] + fh = 0 \tag{1.36}$$

Expressing the diffusive flow rate $q$ and the advective transport rate $[u\phi]$ at point $A$ with respect to their value at point $B$ with the help of Taylor expansions allows us to write:

$$q_A = q(x_B - h) = q(x_B) - h \left.\frac{dq}{dx}\right|_B + O(h^2) \tag{1.37}$$

$$[u\phi]_A = [u\phi](x_B - h) = [u\phi](x_B) - h \left.\frac{d[u\phi]}{dx}\right|_B + O(h^2) \tag{1.38}$$

Introducing Equations 1.37-1.38 into 1.36 noting that the position of $B$ is arbitrary, i.e. $x_B = x$ gives after simplification:

$$-\frac{d[u\phi]}{dx} - \frac{dq}{dx} + f = 0 \tag{1.39}$$

Introducing Fourier's law ($q = -k\frac{d\phi}{dx}$) and assuming $u$ to be constant:

$$-u\phi_{,x} + k\phi_{,xx} + f = 0 \tag{1.40}$$

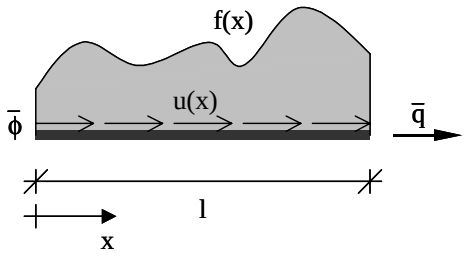



Figure 1-5: 1D advection-diffusion problem

and we recover the standard form of the 1D advection-diffusion problem, where $\nu$ the advective material parameter has been taken equal to 1.

### 1.6.2 Retrieving the Balancing (or Artificial) Diffusion Concept

Assume the advective term has an important variation which implies the extension of the Taylor's expansion up to the third term:

$$[u\phi]_A = [u\phi](x_B - h) = [u\phi](x_B) - h \left.\frac{d[u\phi]}{dx}\right|_B + \frac{h^2}{2} \left.\frac{d^2[u\phi]}{dx^2}\right|_B + O(h^3) \tag{1.41}$$

Introducing this new approximation into balance of fluxes (Equation 1.36) leads to:

$$-\frac{d[u\phi]}{dx} + \frac{h}{2}\frac{d^2[u\phi]}{dx^2} - \frac{dq}{dx} + f = 0 \tag{1.42}$$

and finally, using Fourier's law and assuming a constant velocity $u$:

$$-u\phi_{,x} + \left(k + \frac{uh}{2}\right)\phi_{,xx} + f = 0 \tag{1.43}$$

We therefore notice that accounting for a higher approximation of the advective term introduces naturally an artificial diffusion $\frac{uh}{2}$ in the governing equation which is function of the characteristic length $h$. It can be expressed as:

$$h = \alpha^e l^e \tag{1.44}$$

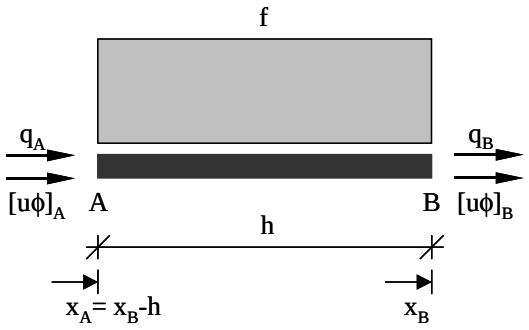



Figure 1-6: Balance domain

where $l^e$ is a characteristic element dimension (in 1D the element's length) and $\alpha^e$ a stabilization parameter to be determined.

### 1.6.3 Generalization of Stabilization

Assume now that the diffusive term has also to be expanded up to the third order and a linear variation of the source term $f$ over $AB$. We can therefore write the following expansions:

$$q_A = q(x_B - h) = q(x_B) - h \left.\frac{dq}{dx}\right|_B + \frac{h^2}{2} \left.\frac{d^2q}{dx^2}\right|_B + O(h^3) \tag{1.45}$$

$$f_A = f(x_B - h) = f(x_B) - h \left.\frac{df}{dx}\right|_B + O(h^2) \tag{1.46}$$

The balance of fluxes now reads (replacing $x_B$ by $x$):

$$q(x) + [u\phi](x) - q(x-h) - [u\phi](x-h) - \frac{1}{2}[f(x) + f(x-h)]h = 0 \tag{1.47}$$

and introducing Equations 1.41, 1.45 and 1.46 into 1.47 yields finally:

$$r - \frac{h}{2} r_{,x} = 0 \tag{1.48}$$

with $r$ the residual of the advection-diffusion equation defined by:

$$r = -u\phi_{,x} + k\phi_{,xx} + f \tag{1.49}$$

**Remark 5** *clearly, at the limit when $h \rightarrow 0$, Equation 1.48 gives the standard advection-diffusion equation.*

**Remark 6** *the term $-\frac{h}{2}r_{,x}$ introduces stabilization at the level of the differential equation. Therefore, a standard finite-element method such as Galerkin can be used.*

**Remark 7** *the directional character of this method is obvious in the multidimensional case where Equation 1.48 becomes $r - \frac{1}{2}\mathbf{h}^T\overrightarrow{\nabla}r = 0$. In 3D, $\mathbf{h} = \begin{bmatrix} h_x & h_y & h_z \end{bmatrix}$.*

### 1.6.4 Stabilized Neumann Boundary Condition

Writing the balance equation at a boundary point (see Figure 1-7), where this time the length of the domain has been taken as $h$ / 2 and $f$ is supposed to be constant over the domain:

$$\overline{q} - q(x_A) - [u\phi](x_A) - \frac{h}{2}f = 0 \tag{1.50}$$

with $x_A = x_B - \frac{h}{2}$, we retrieve, after expressing $(\cdot)_A$ values in function of $(\cdot)_B$ values through second-order Taylor expansions:

$$-[u\phi] + k\frac{d\phi}{dx} + \overline{q} - \frac{h}{2}r = 0 \tag{1.51}$$

where $r$ is expressed by Equation 1.49.

**Remark 8** *the need to take $h$ / 2 as domain length will become clear when we derive the weak form and compare it with the Petrov-Galerkin method.*

### 1.6.5 Discussion, Equivalence with Petrov-Galerkin Method

We obtain a weak form by premultiplying the stabilized differential Equation 1.48 by a weighting function $w$ and integrating over the domain (here from 0 to $l$):

$$\int_0^l w\left(r - \frac{h}{2}r_{,x}\right)dx = 0 \tag{1.52}$$

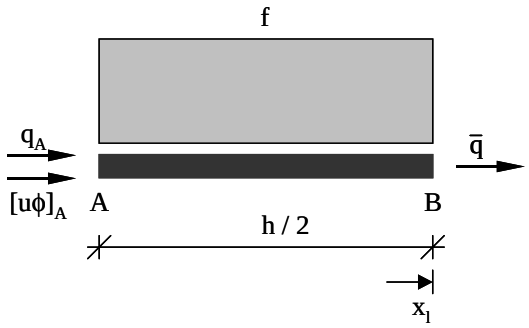



Figure 1-7: Balance domain next to a Neumann boundary condition

Integrating the second term by parts yields:

$$\int_0^l wr\,dx + \frac{h}{2}\int_0^l w_{,x} r\,dx - \left[\frac{h}{2}wr\right]_0^l = 0 \tag{1.53}$$

Regrouping the first two terms and noting that $w = 0$ at $x = 0$ (Dirichlet boundary):

$$\int_0^l \left(w + \frac{h}{2}w_{,x}\right) r\,dx - \left.\frac{h}{2}wr\right|_{x=l} = 0 \tag{1.54}$$

we introduce the stabilized Neumann boundary condition (Equation 1.51):

$$\int_0^l \left(w + \frac{h}{2}w_{,x}\right) r\,dx - \left. w\left(-u\phi + k\phi_{,x} + \overline{q}\right)\right|_{x=l} = 0 \tag{1.55}$$

or, introducing the sum over all elements and assuming $w$ and $\phi$ to be globally $C^0$ but suffering from slope discontinuities at element boundaries (therefore yielding jump terms):

$$\sum_{A=1}^{A_{\max}} \int_{x_{A-1}}^{x_A} \widetilde{w} r\,dx + \sum_{A=1}^{A_{\max}-1} \widetilde{w}(x_A)(k\phi_{,x}(x_A^+) - k\phi_{,x}(x_A^-)) - \left. w\left(-u\phi + k\phi_{,x} + \overline{q}\right)\right|_{x=l} = 0 \tag{1.56}$$

where $\widetilde{w} = \left(w + \frac{h}{2}w_{,x}\right)$. We indentify the following **Euler-Lagrange equations**:

- $r = -u\phi_{,x} + k\phi_{,xx} + f = 0$ in every element interior $]x_{A-1}, x_A[$

- $k\phi_{,x}(x_A^+) = k\phi_{,x}(x_A^-)$: continuity of flux across element interfaces
- $-u\phi + k\phi_{,x} + \overline{q} = 0$ at $x = l$

**Remark 9** *the first two Euler-Lagrange correspond to the ones we have derived in the Petrov-Galerkin case.*

**Remark 10** *hadn't we stabilized the Neumann boundary condition, we would not retrieve the classical boundary condition at $x = l$, but instead $-u\phi + k\phi_{,x} + \overline{q} - \frac{h}{2}r = 0$.*

**Remark 11** *the term weighting the residual takes the form $w + \frac{h}{2}w_{,x}$. This is similar to the one we have derived in the Petrov-Galerkin case (see Equation 1.28): $w + p$ with $p = \tau u w_{,x}$ and $\tau = \frac{h}{2|u|}$.*

# Chapter 2

# Preliminaries

In this chapter, different basic concepts are introduced (see Figure 2-1). First, we speak of principles governing continuum mechanics. Then, a brief introduction on elastoplasticity is made (constitutive modelling), and finally the standard finite element displacement formulation is applied to elasticity and elastoplasticity.

## 2.1 Governing Equations of Continuum Mechanics

### 2.1.1 Introduction

In the following section, the ingredients that we need to establish the boundary value problem that we will solve using the finite element method are presented. First, the physical concepts of stresses and strains are introduced. Then, fundamental balance laws are recalled [3][36], namely:

- conservation of mass (yields continuity)
- balance of linear momentum (yields equilibrium)
- balance of angular momentum (yields symmetry of the stress tensor)

Two basic theorems are also discussed: the divergence theorem and the transport theorem. Finally the energy balance laws will not be considered here as we will deal with isothermic problems.

### 2.1.2 Concept of Stress

External forces applied to a body induce internal forces which can be described by the concept of stress tensors. These stresses in turn induce strains, and the body moves and/or deforms.

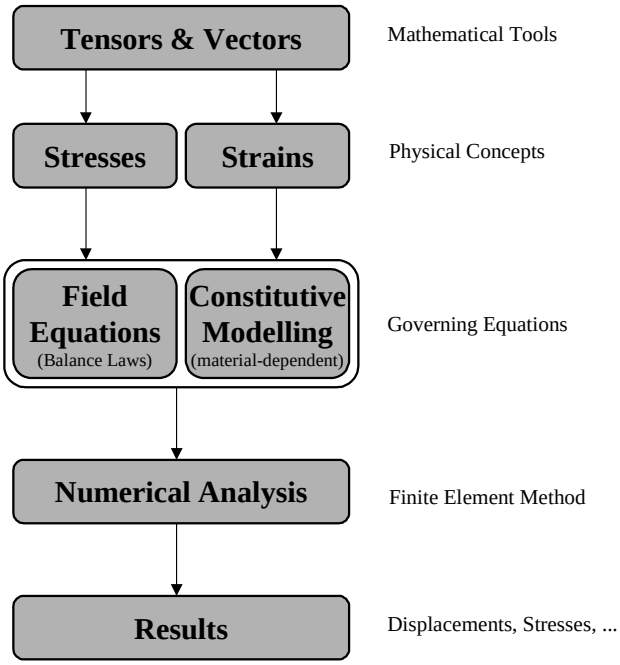



Figure 2-1: The global picture

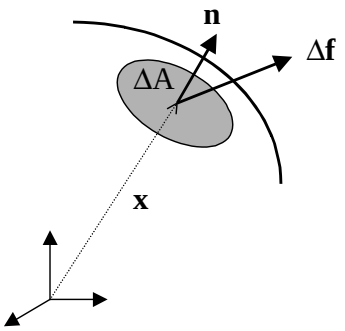



Figure 2-2: Concept of traction vector

Looking at Figure 2-2, we can define a traction vector $\mathbf{t}$ as:

$$\mathbf{t}(\mathbf{x}) = \lim_{\Delta A \to 0} \frac{\mathbf{\Delta f}}{\Delta A} \tag{2.1}$$

Cauchy postulates that the traction vector $\mathbf{t}$ remains unchanged for any surface having at $\mathbf{x}$ the same normal $\mathbf{n}$, which allows us to express $\mathbf{t}$ in function of $\mathbf{n}$, introducing the Cauchy stress tensor $\boldsymbol{\sigma}$:

$$\mathbf{t}(\mathbf{n}) = \boldsymbol{\sigma} \cdot \mathbf{n} \tag{2.2}$$

with:

$$\boldsymbol{\sigma} = \sigma_{pq} \left( \mathbf{e}_p \otimes \mathbf{e}_q \right) \tag{2.3}$$

where $\mathbf{e}_p$ is the $p$-th Euclidean basis vector. In a three-dimensional space, $\boldsymbol{\sigma}$ will have nine components:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \tag{2.4}$$

We will see later on that the balance of angular momentum allows us to show that $\boldsymbol{\sigma} = \boldsymbol{\sigma}^T$ and therefore that $\boldsymbol{\sigma}$ has only six independent components.

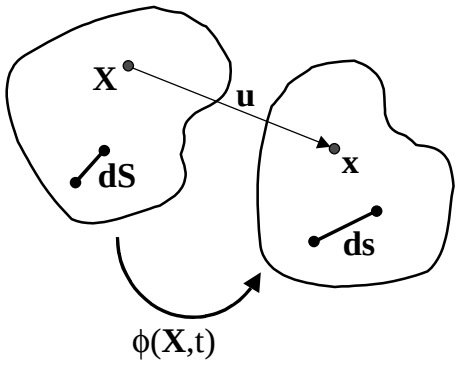



Figure 2-3: Reference and current configurations

### 2.1.3 Concept of Strain

#### Change of Configuration

To quantify the motion of a body, it is useful to represent it under two different configurations (see Figure 2-3): a **reference configuration**, where a given material point is indicated by $\mathbf{X}$, and a **current configuration**, where the same material point is designated by $\mathbf{x}$. Both configurations are related through the definition of the **motion** $\boldsymbol{\phi}$:

$$\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t) \overset{\text{notation}}{=} \mathbf{x}(\mathbf{X}, t) \tag{2.5}$$

The **displacement vector u** is defined by:

$$\mathbf{u}(\mathbf{X}, t) = \mathbf{x}(\mathbf{X}, t) - \mathbf{X} \tag{2.6}$$

#### Deformation Gradient

The **deformation gradient F** is a tensor given by:

$$\mathbf{F} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \overset{\text{notation}}{=} \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \tag{2.7}$$

or, in component notation:

$$F_{iI} = \frac{\partial x_i}{\partial X_I} \tag{2.8}$$

The determinant of $\mathbf{F}$ is called the **Jacobian determinant**:

$$J = \det(\mathbf{F}) \tag{2.9}$$

**Green Strain Tensor**

To quantify the deformation of a body, it is usual to study the variation of the squared-distance between two neighbouring points. If we take a look at Figure 2-3 and we express the value of $dS^2$ in the reference configuration and $ds^2$ which corresponds to $dS^2$ in the current configuration with the help of the Pythagorean Theorem:

$$dS^2 = dX_K dX_K = dX_I dX_J \delta_{IJ} \tag{2.10}$$
$$ds^2 = dx_k dx_k \tag{2.11}$$

Chain rule implies that:

$$dx_k = \frac{\partial x_k}{\partial X_I} dX_I \tag{2.12}$$

and therefore, introducing Equation 2.12 into Equation 2.11:

$$ds^2 = \frac{\partial x_k}{\partial X_I}\frac{\partial x_k}{\partial X_J} dX_I dX_J \tag{2.13}$$

The difference between the two values $ds^2 - dS^2$ is a function of the deformation only. It does **not** depend on the rigid displacement of the body. Therefore we write:

$$ds^2 - dS^2 = \left(\frac{\partial x_k}{\partial X_I}\frac{\partial x_k}{\partial X_J} - \delta_{IJ}\right) dX_I dX_J \tag{2.14}$$

In Equation 2.14, we identify the expression inside the parentheses with $2E_{IJ}$ and we call $\mathbf{E}$ the Green strain tensor. Equation 2.8 shows that we can express $\mathbf{E}$ as:

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right) \tag{2.15}$$

It is more common to express $\mathbf{E}$ in function of the displacement field $\mathbf{u}$. Noting that Equation 2.6 implies:

$$\frac{\partial x_k}{\partial X_I} = \frac{\partial u_k}{\partial X_I} + \frac{\partial X_k}{\partial X_I} = \frac{\partial u_k}{\partial X_I} + \delta_{kI} \tag{2.16}$$

We finally get, as a final expression for the Green strain tensor components $E_{IJ}$:

$$E_{IJ} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_J} + \frac{\partial u_j}{\partial X_I} + \frac{\partial u_k}{\partial X_I}\frac{\partial u_k}{\partial X_J}\right) \tag{2.17}$$

**Small Strains**

A linear approximation to small strains can be made if we make the assumption that for any $dS$, the rotation and the dilatation are small enough. In this case, the reference and the current configuration are unified. We also neglect the second-order terms in Equation 2.17 and this allows us to write the components of the infinitesimal strain tensor $\varepsilon_{ij}$ as:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \overset{\text{def.}}{=} u_{(i,j)} \tag{2.18}$$

### 2.1.4 Divergence Theorem

The divergence (or Green's) theorem allows us to express a surface integral into a volume integral (or vice versa). Assume we have a continuous vector-valued function $\mathbf{F}$. Then, assuming that $\mathbf{n}$ is the unit outward normal to $V$, and $S = \partial V$:

$$\int_S (\mathbf{n} \cdot \mathbf{F})\, dS = \int_V div(\mathbf{F}) dV \tag{2.19}$$

### 2.1.5 Conservation of Mass

The total mass $m(t)$ of a body $B$ at time $t$ can be expressed as:

$$m(t) = \int_V \rho dV \tag{2.20}$$

The value of the spatial derivative of $m(t)$ indicates the amount of mass that flows into a control volume $V$ (see Figure 2-4):

$$\frac{\partial m(t)}{\partial t} \equiv \int_V \frac{\partial \rho}{\partial t} dV + \int_S (\rho \mathbf{v} \cdot \mathbf{n})\, dS = 0 \tag{2.21}$$

Applying the divergence theorem allows us to write:

$$\int_S (\rho \mathbf{v} \cdot \mathbf{n})\, dS = \int_V div\,(\rho \mathbf{v})\, dV \tag{2.22}$$

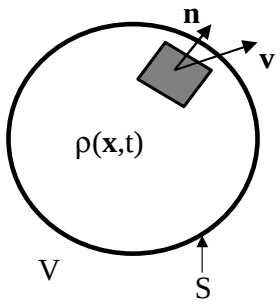



Figure 2-4: Control volume $V$

And combining Equations 2.21 and 2.22 leads to:

$$\int_V \left[\frac{\partial \rho}{\partial t} + div\,(\rho \mathbf{v})\right] dV = 0 \tag{2.23}$$

Argueing that $V$ is arbitrary, we retrieve the continuity equation in its local form:

$$\frac{\partial \rho}{\partial t} + div\,(\rho \mathbf{v}) = 0 \tag{2.24}$$

We can also express this equation writing:

$$div\,(\rho \mathbf{v}) = \frac{\partial}{\partial x_i}(\rho v_i) = \rho_{,i} v_i + \rho v_{i,i} = \mathbf{v} \cdot \overrightarrow{\nabla} \rho + \rho div\,(\mathbf{v}) \tag{2.25}$$

Introducing 2.25 into 2.24 we retrieve:

$$\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \overrightarrow{\nabla} \rho + \rho div\,(\mathbf{v}) = 0 \tag{2.26}$$

or, using the definition of the material time derivative:

$$\frac{d(\bullet)}{dt} = \frac{\partial (\bullet)}{\partial t} + \mathbf{v} \cdot \overrightarrow{\nabla} (\bullet) \tag{2.27}$$

we get the material form of the continuity equation:

$$\frac{d\rho}{dt} + \rho div\left(\mathbf{v}\right) = 0 \tag{2.28}$$

If the material is incompressible, $\frac{d\rho}{dt}$ vanishes and we retrieve the well-known incompressibility condition:

$$div(\mathbf{v}) = v_{i,i} = 0 \tag{2.29}$$

### 2.1.6 Reynolds Transport Theorem

Reynolds transport theorem expresses the material time derivative of a volume integral, where the control surface isn't fixed in space, but instead is defined as the boundary of a given **mass system**. For instance, consider the following expression:

$$\frac{d}{dt}\int_V \rho\mathcal{A}dV \tag{2.30}$$

where $\mathcal{A}$ may be a scalar, vectorial or tensorial continuous function (physical quantity per unit mass). If we express the above quantity as the sum of the time derivative of $\mathcal{A}$ in $V$ and the surface flux:

$$\begin{aligned}
\frac{d}{dt}\int_V \rho\mathcal{A}dV &= \int_V \frac{\partial}{\partial t}\left(\rho\mathcal{A}\right)dV + \int_S \left(\rho\mathcal{A}\right)\mathbf{v}\cdot\mathbf{n}dS \\
&= \int_V \left[\frac{\partial\left(\rho\mathcal{A}\right)}{\partial t} + div(\rho\mathcal{A}\mathbf{v})\right]dV \\
&= \int_V \left[\rho\frac{\partial\mathcal{A}}{\partial t} + \mathcal{A}\frac{\partial\rho}{\partial t} + \mathcal{A}\cdot div(\rho\mathbf{v}) + \rho\mathbf{v}\cdot\overrightarrow{\bigtriangledown}\mathcal{A}\right]dV \\
&= \int_V \mathcal{A}\left[\frac{\partial\rho}{\partial t} + div(\rho\mathbf{v})\right]dV + \int_V \rho\left[\frac{\partial\mathcal{A}}{\partial t} + \mathbf{v}\cdot\overrightarrow{\bigtriangledown}\mathcal{A}\right]dV
\end{aligned} \tag{2.31}$$

The first term vanishes because of the conservation of mass (Equation 2.24) and we finally get:

$$\frac{d}{dt}\int_V \rho\mathcal{A}dV = \int_V \rho\left[\frac{\partial\mathcal{A}}{\partial t} + \mathbf{v}\cdot\overrightarrow{\bigtriangledown}\mathcal{A}\right]dV \tag{2.32}$$

Introducing Equation 2.27, we get the simple result:

$$\frac{d}{dt}\int_V \rho\mathcal{A}dV = \int_V \rho\frac{d\mathcal{A}}{dt}dV \tag{2.33}$$

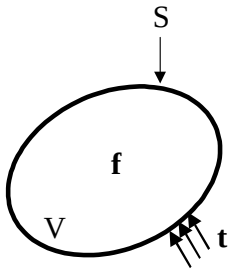



Figure 2-5: Equilibrium of a body $V$

### 2.1.7 Balance of Linear Momentum

The balance of linear momentum expresses the fact that the total forces (body and traction) acting on a given body $V$ (see Figure 2-5) are in equilibrium with the rate of change of linear momentum:

$$\int_V \mathbf{f} dV + \int_S \mathbf{t} dS = \frac{d}{dt} \int_V \rho \mathbf{v} dV \tag{2.34}$$

The notion of stress tensor $\boldsymbol{\sigma}$, introduced by Cauchy, and the divergence theorem allows us to write:

$$\int_S \mathbf{t} dS = \int_S (\boldsymbol{\sigma} \cdot \mathbf{n}) \, dS = \int_V div(\boldsymbol{\sigma}) dV \tag{2.35}$$

The right-hand side of Equation 2.34 can be written (due to Equation 2.33):

$$\frac{d}{dt} \int_V \rho \mathbf{v} dV = \int_V \rho \frac{d\mathbf{v}}{dt} dV \tag{2.36}$$

We finally get:

$$\int_V \left[ div(\boldsymbol{\sigma}) + \mathbf{f} - \rho \frac{d\mathbf{v}}{dt} \right] dV = 0 \tag{2.37}$$

And therefore, argueing that the volume $V$ is arbitrary:

$$div(\boldsymbol{\sigma}) + \mathbf{f} = \rho \frac{d\mathbf{v}}{dt} \tag{2.38}$$

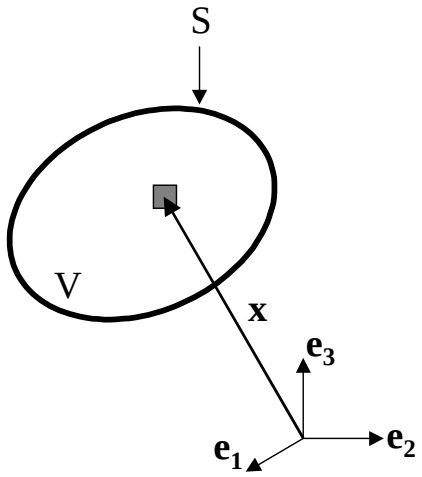



Figure 2-6: Balance of angular momentum of a body $V$

In the static case, we can write Equation 2.38 in indicial notation as:

$$\sigma_{ij,j} + f_i = 0 \tag{2.39}$$

### 2.1.8 Balance of Angular Momentum

At any point $\mathbf{x}$ of a given body $V$ (see Figure 2-6), the balance of angular momentum (or moment of momentum) allows us to write:

$$\int_V (\mathbf{f} \times \mathbf{x})\, dV + \int_S (\mathbf{t} \times \mathbf{x})\, dS = \frac{d}{dt} \int_V (\rho \mathbf{v} \times \mathbf{x})\, dV \tag{2.40}$$

If we expand and consider the $\mathbf{e}_1$ term ($\mathbf{e}_1$ being the first Euclidean basis vector), this leads to:

$$\int_V (f_2 x_3 - f_3 x_2)\, dV + \int_S (t_2 x_3 - t_3 x_2)\, dS = \frac{d}{dt} \int_V \rho\, (v_2 x_3 - v_3 x_2)\, dV \tag{2.41}$$

The right-hand side of Equation 2.41 can be written (thanks to the transport's theorem):

$$\begin{aligned}\frac{d}{dt}\int_V \rho\left(v_2x_3 - v_3x_2\right)dV &= \int_V \rho\left(v_2v_3 + \dot{v}_2x_3 - v_3v_2 - \dot{v}_3x_2\right)dV \\ &= \int_V \rho\left(\dot{v}_2x_3 - \dot{v}_3x_2\right)dV \end{aligned} \tag{2.42}$$

The second term of the left-hand side of Equation 2.41 can in turn be expanded as (using Cauchy's stress tensor principle and the divergence theorem):

$$\begin{aligned}\int_S \left(t_2x_3 - t_3x_2\right)dS &= \int_S \left(n_i\sigma_{i2}x_3 - n_i\sigma_{i3}x_2\right)dS \\ &= \int_V \left(\frac{d\left(\sigma_{i2}x_3\right)}{dx_i} - \frac{d\left(\sigma_{i3}x_2\right)}{dx_i}\right)dV \\ &= \int_V \left(\sigma_{i2}\delta_{i3} + x_3\frac{d\sigma_{i2}}{dx_i} - \sigma_{i3}\delta_{i2} - x_2\frac{d\sigma_{i3}}{dx_i}\right)dV \\ &= \int_V \left(\sigma_{32} + x_3\frac{d\sigma_{i2}}{dx_i} - \sigma_{23} - x_2\frac{d\sigma_{i3}}{dx_i}\right)dV \end{aligned} \tag{2.43}$$

Collecting terms, we can rewrite Equation 2.41 in the following form:

$$\begin{aligned}\int_V \left(\sigma_{32} - \sigma_{23}\right)dV &= -\int_V x_3\left(\frac{d\sigma_{i2}}{dx_i} + f_2 - \rho\dot{v}_2\right)dV \\ &\quad -\int_V x_2\left(\frac{d\sigma_{i3}}{dx_i} + f_3 - \rho\dot{v}_3\right)dV \end{aligned} \tag{2.44}$$

Both terms on the right-hand side vanish due to balance of linear momentum (equilibrium) in the $\mathbf{e}_2$ and $\mathbf{e}_3$ directions, which means that:

$$\int_V \left(\sigma_{32} - \sigma_{23}\right)dV = 0 \tag{2.45}$$

$V$ is an arbitrary volume. This leads to:

$$\sigma_{32} = \sigma_{23} \tag{2.46}$$

And considering similarly the $\mathbf{e}_2$ and $\mathbf{e}_3$ terms of Equation 2.40, we finally obtain the symmetry of the stress tensor:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^T \tag{2.47}$$

## 2.2 Constitutive Modelling

### 2.2.1 Introduction

Constitutive equations relate static and kinematic variables, i.e. in our case stresses and strains. Unlike balance laws, these equations are material-dependent. In this chapter the following assumptions are made:

- the material is isotropic and isothermal
- the strains are small, which allows us to use $\varepsilon_{ij}$ instead of $E_{IJ}$

### 2.2.2 Linear Elasticity

The generalized Hooke's law writes:

$$\sigma_{ij} = D_{ijkl}\varepsilon_{kl} \tag{2.48}$$

In the linear isotropic case, $D_{ijkl}$ coefficients are function of two parameters. Usually, Young modulus $E$ and Poisson ratio $\nu$ are used. In the three-dimensional case, symmetry of the stress and strain tensors allow us to write a $6 \times 6$ matricial representation of Equation 2.48 as:

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \tag{2.49}$$

where the stress vector $\boldsymbol{\sigma}$ is defined by:

$$\boldsymbol{\sigma} = \left\{ \begin{array}{c} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ \sigma_{33} \\ \sigma_{13} \\ \sigma_{23} \end{array} \right\} \tag{2.50}$$

The strain vector $\boldsymbol{\varepsilon}$ reads:

$$\boldsymbol{\varepsilon} = \left\{ \begin{array}{c} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \\ \varepsilon_{33} \\ 2\varepsilon_{13} \\ 2\varepsilon_{23} \end{array} \right\} \tag{2.51}$$

and $\mathbf{D}$ can be expressed as:

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \left[ \begin{array}{cccccc} (1-\nu) & \nu & 0 & \nu & 0 & 0 \\ \nu & (1-\nu) & 0 & \nu & 0 & 0 \\ 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 & 0 \\ \nu & \nu & 0 & (1-\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{array} \right] \tag{2.52}$$

Alternatively, Lamé's coefficients $\lambda$ and $\mu$ (the shear modulus) can be used:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{2.53}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{2.54}$$

They yield a simple expression for $\sigma_{ij}$:

$$\sigma_{ij} = \lambda \varepsilon_{kk} \delta_{ij} + 2\mu \varepsilon_{ij} \tag{2.55}$$

and the expression for $\mathbf{D}$ writes:

$$\mathbf{D} = \begin{bmatrix} \lambda+2\mu & \lambda & 0 & \lambda & 0 & 0 \\ \lambda & \lambda+2\mu & 0 & \lambda & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 \\ \lambda & \lambda & 0 & \lambda+2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{2.56}$$

**Remark 12** *the factor 2 introduced in Equation 2.51 derives from the fact that* $\sigma_{12} = \mu\gamma_{12}$*, whereas* $\varepsilon_{12} = \frac{1}{2}\gamma_{12}$ *(*$\gamma_{12}$ *is the engineer shear strain). This also simplifies the expression of* $\mathbf{D}$*. The same applies to* $(\bullet)_{13}$ *and* $(\bullet)_{23}$*quantities.*

### 2.2.3 Plasticity

#### Basic Ingredients

When elastic behavior is considered, deformation is reversible. Removal of the applied loads will result in the total recovery of the induced deformation. The limit of application of elasticity, defined by the yield condition $f(\boldsymbol{\sigma}) = 0$, depends on the considered material as well as the amount of deformation.

Plasticity implies that a part of the deformation is irreversible: elasto-plastic straining will leave a permanent strain (called the plastic strain) after stress removal (as can be seen on Figure 2-7). Plastic deformation is path dependent: there is no one-to-one correspondence between stress and strain during plastic deformation. It follows that the constitutive equations for plastic deformation must be expressed in differential equations or incremental form, even for rate-insensitive plasticity.

The total increment of strain $\boldsymbol{\Delta\varepsilon}$ will be decomposed into an elastic part $\boldsymbol{\Delta\varepsilon}^e$ and a plastic part $\boldsymbol{\Delta\varepsilon}^p$:

$$\boldsymbol{\Delta\varepsilon} = \boldsymbol{\Delta\varepsilon}^e + \boldsymbol{\Delta\varepsilon}^p \tag{2.57}$$

The relationship between the elastic strain increment and the stress increment is governed by:

$$\Delta\sigma_{ij} = D_{ijkl}\Delta\varepsilon^e_{kl} \tag{2.58}$$

Equation 2.58 can also be written in the following manner, thanks to Equation 2.57:

$$\Delta\sigma_{ij} = D_{ijkl}\left(\Delta\varepsilon_{kl} - \Delta\varepsilon^p_{kl}\right) \tag{2.59}$$

The general plasticity framework exhibits the following features [34]:

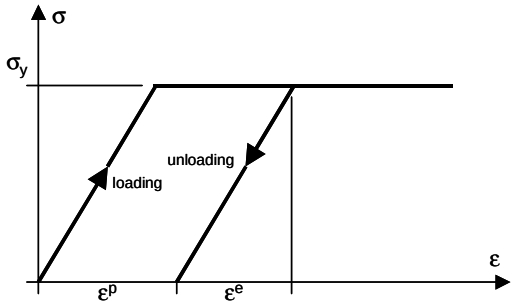



Figure 2-7: 1D $\sigma$-$\varepsilon$ curve

- the **additive decomposition** of the strain increment tensor (see Equation 2.57).
- a yield condition (or **yield function**) in stress space. This condition defines the stress states for which the material exhibits plastic flow, the initiation of the plastic process. For stress values below that surface in stress space, deformation is linear elastic. The yield condition can be expressed in its general form by:

$$f(\boldsymbol{\sigma}) = 0 \tag{2.60}$$

  The yield condition provides only one additional equation, while the new unknown $\boldsymbol{\Delta\varepsilon}^p$ has (in 3D) six independent components. Therefore, an additional rule governing plastic flow must be provided. This flow rule can be deduced from experimental observations, but from the theoretical point of view it is derived from the postulate of maximum plastic dissipation which expresses that among all stress states $\boldsymbol{\sigma}^*$ satisfying the yield condition, the power $\boldsymbol{\sigma}^* : \boldsymbol{\Delta\varepsilon}^p$ is maximized by the actual stress $\boldsymbol{\sigma}$. This condition holds if the direction of the plastic strain increment is normal to a convex yield surface.

- the **flow rule**, which expresses the direction of the plastic strain rate, can be written:

$$\boldsymbol{\Delta\varepsilon}^p = \Delta\gamma \frac{\partial f}{\partial \boldsymbol{\sigma}} \tag{2.61}$$

  $\Delta\gamma$, the plastic multiplier, is a scalar controlling the magnitude of the plastic strain increment. Unfortunately the condition that this rule is associated with the yield function does not hold for every case, especially for pressure-sensitive materials like soils or rocks. A more general expression therefore holds, depending on $g(\boldsymbol{\sigma})$, called the plastic potential:

$$\boldsymbol{\Delta\varepsilon}^p = \Delta\gamma \frac{\partial g}{\partial \boldsymbol{\sigma}} \stackrel{\text{def.}}{=} \Delta\gamma\ \mathbf{r} \tag{2.62}$$

- a **hardening law**, defining the evolution of the yield surface in time. For perfect plasticity, which is considered here, the yield surface remains constant and this ingredient is not necessary.

- the loading / unloading criteria: in the elastic regime the yield function must remain negative and the plastic multiplier $\Delta\gamma$ must be equal to zero, while in the plastic regime the yield function must be equal to zero and $\Delta\gamma$ must be positive. The Kuhn-Tucker conditions express this fact:

$$\begin{aligned} \Delta\gamma \; f(\boldsymbol{\sigma}) &= 0 && (2.63) \\ \Delta\gamma &\geq 0 && (2.64) \\ f(\boldsymbol{\sigma}) &\leq 0 && (2.65) \end{aligned}$$

- during plastic flow, the stress remains on the yield surface, and so the yield function remains equal to zero for a certain period of time. Therefore, another condition, called **consistency** requires that the derivative of the yield function vanishes whenever $\Delta\gamma$ is positive:

$$\Delta\gamma \; \dot{f}(\boldsymbol{\sigma}) = 0 \qquad (2.66)$$

This last ingredient allows us to determine the elastoplastic material stiffness matrix $\mathbf{D}^{ep}$ which is used in the computation of the tangential stiffness matrix in finite element computations. The derivation of $\mathbf{D}^{ep}$ will be carried out in section 2.3.2 when linearization is discussed.

An elastic-perfectly plastic model is totally defined by its yield function and its plastic potential. The two following sections present examples of plastic models widely used in numerical simulation.

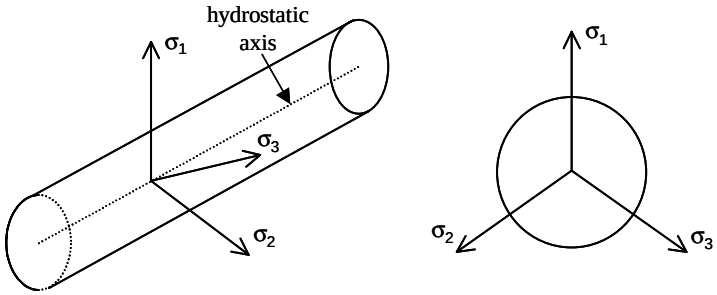



Figure 2-8: Von Misès yield surface: a) 3D representation - b) Deviatoric plane section

**von Mises Criterion**

This criterion assumes that plastic yielding will occur when the second invariant $J_2$ of the deviatoric stress tensor $\mathbf{s}$ reaches a critical value $k^2$, which is a material property. This criterion applies mainly for metals. It is expressed by:

$$f(\boldsymbol{\sigma}) = \sqrt{J_2} - k = 0 \tag{2.67}$$

where $J_2$, the second invariant of the deviatoric stress tensor $\mathbf{s}$ can be calculated by:

$$J_2 = \frac{1}{2} s_{ij} s_{ij} \tag{2.68}$$

and $s_{ij}$ is defined by:

$$s_{ij} = \sigma_{ij} - \frac{1}{3}\sigma_{kk}\delta_{ij} \tag{2.69}$$

A representation of this criterion in the three-dimensional stress space and in the deviatoric plane is given in Figure 2-8.

Assuming an associative flow rule (probably the only meaningful rule for von Mises plasticity), i.e. $g = f$, we can determine the flow vector as follows:

$$\mathbf{r}(\boldsymbol{\sigma}) = \frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{1}{2\sqrt{J_2}} s_{ij} \tag{2.70}$$

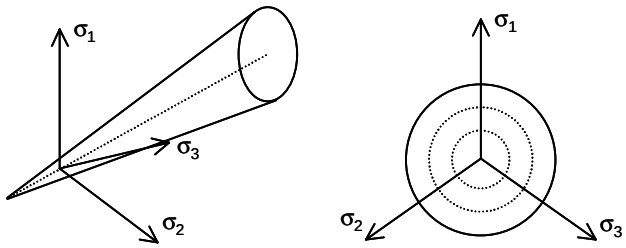



Figure 2-9: Drucker-Prager yield surface: a) 3D representation - b) Deviatoric plane section

**Drucker Prager Criterion**

This criterion adds a hydrostatic stress term to the von Mises criterion and therefore is pressure sensitive. This criterion applies for porous or brittle materials, including soils or rocks:

$$f(\boldsymbol{\sigma}) = a_\phi I_1 + \sqrt{J_2} - k = 0 \tag{2.71}$$

$a_\phi$ is a new parameter, and $I_1$ is the first invariant of the stress tensor $\boldsymbol{\sigma}$:

$$I_1 = \sigma_{kk} \tag{2.72}$$

The plastic potential $g$ is defined similarly as:

$$g(\boldsymbol{\sigma}) = a_\psi I_1 + \sqrt{J_2} \tag{2.73}$$

Drucker-Prager's yield criterion is represented in Figure 2-9.

An analogy can be made between Drucker-Prager's yield surface and the more commonly used Mohr-Coulomb yield criterion. A correspondance links the angle of friction $\phi$ and the cohesion $c$ with Drucker-Prager's parameters $a_\phi$ and $k$ through a size-adjustment [67].

### 2.2.4 Rate Independent Perfect Plasticity

We summarize here the statement of the nonlinear elastoplastic problem. We introduce some algorithmic concepts that will be developed later in this work, but it might be interesting to have a global picture at hand.

| | **Equations** | **3D Plastic case** | **3D Elastic case** |
|---|---|---|---|
| (i) | $\mathrm{div}(\boldsymbol{\sigma}) + \mathbf{f} = \mathbf{0}$ | 3 eq. & 6 $\dot{\boldsymbol{\sigma}}$ | 3 eq. & 6 $\dot{\boldsymbol{\sigma}}$ |
| (ii) | $\dot{\boldsymbol{\sigma}} = \mathbf{D}(\dot{\boldsymbol{\varepsilon}} - \dot{\boldsymbol{\varepsilon}}^p)$ | 6 eq. & 6 $\dot{\boldsymbol{\varepsilon}}$ + 6 $\dot{\boldsymbol{\varepsilon}}^p$ | 6 eq. & 6 $\dot{\boldsymbol{\varepsilon}}$; $\dot{\boldsymbol{\varepsilon}}^p = \mathbf{0}$ |
| (iii) | $\dot{\boldsymbol{\sigma}}^{tr} = \mathbf{D}\dot{\boldsymbol{\varepsilon}}$ | - | - |
| (iv) | $\dot{\boldsymbol{\sigma}}^p = -\mathbf{D}\dot{\boldsymbol{\varepsilon}}^p$ | - | - |
| (v) | $\dot{\boldsymbol{\varepsilon}} = \dot{\boldsymbol{\varepsilon}}^e + \dot{\boldsymbol{\varepsilon}}^p$ | 6 eq. & 6 $\dot{\boldsymbol{\varepsilon}}^e$ | 6 eq. & 6 $\dot{\boldsymbol{\varepsilon}}^e$ |
| (vi) | $\dot{\boldsymbol{\varepsilon}} = \frac{1}{2}\left(\overrightarrow{\nabla}\dot{\mathbf{u}} + \overrightarrow{\nabla}^T\dot{\mathbf{u}}\right)$ | 6 eq. & 3 $\dot{\mathbf{u}}$ | 6 eq. & 3 $\dot{\mathbf{u}}$ |
| (vii) | $\dot{\boldsymbol{\varepsilon}}^p = \dot{\gamma}\frac{\partial g}{\partial \boldsymbol{\sigma}}$, given $g(\boldsymbol{\sigma})$ | 6 eq. & 1 $\dot{\gamma}$ | - |
| (viii) | $f(\boldsymbol{\sigma})\begin{cases} = 0 & \text{if } \textit{plastic} \text{ process} \\ < 0 & \text{if } \textit{elastic} \text{ process} \end{cases}$ | 1 eq. | 1 eq. & 1 $f$ |
| | **Total** | 28 eq. & 28 unkn. | 22 eq. & 22 unkn. |

The resolution procedure is based on a discretized weak form in terms of displacement increments: given the displacement $\mathbf{u}_n$, the stress $\boldsymbol{\sigma}_n$, the strain $\boldsymbol{\varepsilon}_n$ and the plastic strain $\boldsymbol{\varepsilon}_n^p$ at $t_n$, find $\mathbf{u}_{n+1}$ and associated variables at $t_{n+1}$, where $t$ is a fictitious time.

The problem has to be solved iteratively as we deal with a nonlinear problem. At iteration $i+1$, strain increments result from (vi), as $\boldsymbol{\Delta\varepsilon}^i_{n+1} = \frac{1}{2}\left(\overrightarrow{\nabla}\boldsymbol{\Delta}\mathbf{u}^i_{n+1} + \overrightarrow{\nabla}^T\boldsymbol{\Delta}\mathbf{u}^i_{n+1}\right)$. A trial elastic stress is computed from (iii) as:

$$\boldsymbol{\sigma}^{tr,i}_{n+1} = \boldsymbol{\sigma}_n + \Delta\boldsymbol{\sigma}^{tr,i}_{n+1} = \boldsymbol{\sigma}_n + \mathbf{D}\boldsymbol{\Delta\varepsilon}^i_{n+1} \tag{2.74}$$

Elastic or plastic behaviour associated with $\boldsymbol{\sigma}^{tr,i}_{n+1}$ is identified through (viii). In the plastic case, $\Delta\gamma$ is then computed from $f(\boldsymbol{\sigma}_{n+1}) = 0$ (see Figure 2-10):

$$f(\boldsymbol{\sigma}_{n+1}) = f(\boldsymbol{\sigma}^{tr,i}_{n+1}) + \left.\frac{\partial f}{\partial \boldsymbol{\sigma}}\right|^T_{\boldsymbol{\sigma}^{tr,i}_{n+1}} \boldsymbol{\Delta\sigma}^p = 0 \tag{2.75}$$

where $\boldsymbol{\Delta\sigma}^p = -\mathbf{D}\Delta\gamma\frac{\partial g}{\partial \boldsymbol{\sigma}}$ (from (iv) & (vii)). This leads to:

$$\Delta\gamma = \frac{f(\boldsymbol{\sigma}^{tr}_{n+1})}{\frac{\partial f}{\partial \boldsymbol{\sigma}}^T\mathbf{D}\frac{\partial g}{\partial \boldsymbol{\sigma}}} \tag{2.76}$$

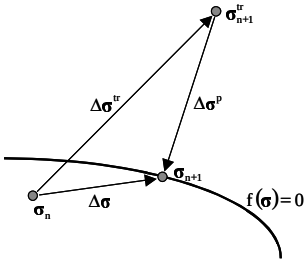



Figure 2-10: Illustration of $f(\boldsymbol{\sigma}_{n+1}) = 0$

Then, $\boldsymbol{\Delta\varepsilon}^{p,i}_{n+1}$, $\boldsymbol{\Delta\sigma}^{i}_{n+1}$ and finally $\boldsymbol{\sigma}^{i}_{n+1} = \boldsymbol{\sigma}_n + \boldsymbol{\Delta\sigma}^{i}_{n+1}$ result, which allows us to express $N(\mathbf{u}^{i}_{n+1})$ as:

$$N(\mathbf{u}^{i}_{n+1}) = \int_{\Omega} \mathbf{B}^T \boldsymbol{\sigma}^{i}_{n+1} d\Omega \tag{2.77}$$

The system of equations to be resolved takes the form:

$$\frac{\partial N(\mathbf{u}^{i}_{n+1})}{\partial \mathbf{u}} \boldsymbol{\Delta u} = \mathbf{F}_{n+1} - N(\mathbf{u}^{i}_{n+1}) \tag{2.78}$$

where $\mathbf{F}_{n+1}$ represents the known applied forces on the system at step $n+1$. We can therefore compute a new displacement increment $\boldsymbol{\Delta u}$ through Equation 2.78. The new accumulated increment of displacement $\boldsymbol{\Delta u}^{i+1}_{n+1}$ then reads:

$$\boldsymbol{\Delta u}^{i+1}_{n+1} = \boldsymbol{\Delta u}^{i}_{n+1} + \boldsymbol{\Delta u} \tag{2.79}$$

and we repeat the iterative process until convergence occurs, i.e. until the right-hand side of Equation 2.78 is less than some prescribed tolerance.

**Remark 13** *unloading phenomena (i.e. stresses which have reached the yield surface and then unload inside this surface) are handled without problem by all the formulations described in this work.*

**Remark 14** *the former development illustrates the standard displacement formulation. In the mixed case, pressure $p$ is introduced as a supplementary unknown field, $\dot{p}$ appears in the the constitutive equation expressing $\dot{\boldsymbol{\sigma}}$, and a new equation expressing continuity completes the formulation. Details will be given in chapter 3.*

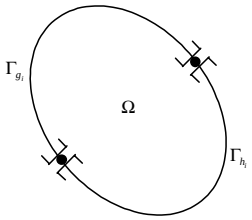



Figure 2-11: Problem statement

## 2.3 Standard Galerkin Finite Element Formulation

### 2.3.1 Elastic Case

#### Strong Form of the Boundary Value Problem

In this section, we will consider the standard displacement finite element method applied to small strains linear isotropic elasticity. Consider a body $\Omega$, its boundary $\partial\Omega = \Gamma = \overline{\Gamma_{g_i} \cup \Gamma_{h_i}}$ with $\Gamma_{g_i} \cap \Gamma_{h_i} = \emptyset$, with $i = 1, ..., n_{sd}$, where $n_{sd}$ stands for the number of spatial dimensions (see Figure 2-11).

We start with the following strong form $(S)$ of the boundary value problem: given $f_i : \Omega \longrightarrow \mathbb{R}$, $g_i : \Gamma_{g_i} \longrightarrow \mathbb{R}$ and $h_i : \Gamma_{h_i} \longrightarrow \mathbb{R}$, we wish to find $u_i : \overline{\Omega} \longrightarrow \mathbb{R}$ such that:

$$\sigma_{ij,j} + f_i = 0 \qquad \text{in } \Omega \tag{2.80}$$

$$u_i = g_i \qquad \text{on } \Gamma_{g_i} \tag{2.81}$$

$$\sigma_{ij} n_j = h_i \qquad \text{on } \Gamma_{h_i} \tag{2.82}$$

with the following constitutive relation:

$$\sigma_{ij}(\mathbf{u}) = D_{ijkl}\varepsilon_{kl}(\mathbf{u}) = D_{ijkl}u_{(k,l)} \tag{2.83}$$

Equation 2.80 represents the static equilibrium condition and derives from the governing principle expressing the conservation of linear momentum. Equation 2.81 is an essential boundary condition on the displacement field which must be enforced on the $\Gamma_{g_i}$ part of the boundary, while Equation 2.82 is a natural boundary condition appearing on $\Gamma_{h_i}$.

**Weak Form**

We begin with the definition of the following spaces:

$$\mathcal{S}_i = \{u_i \in H^1(\Omega) \mid u_i = g_i \text{ on } \Gamma_{g_i}\} \quad (2.84)$$
$$\mathcal{V}_i = \{w_i \in H^1(\Omega) \mid w_i = 0 \text{ on } \Gamma_{g_i}\} \quad (2.85)$$

$\mathcal{S}_i$ represents the trial solution space and $\mathcal{V}_i$ the weighting function space. We premultiply Equation 2.80 by a weighting function $w_i$ and integrate over the domain:

$$\int_\Omega w_i \left(\sigma_{ij,j} + f_i\right) d\Omega = 0 \quad (2.86)$$

Integrating the first expression by parts, we get:

$$-\int_\Omega w_{i,j}\sigma_{ij}d\Omega + \int_\Gamma w_i\sigma_{ij}n_j d\Gamma + \int_\Omega w_i f_i d\Omega = 0 \quad (2.87)$$

As $\sigma_{ij}$ is a symmetric tensor (thanks to the conservation of angular momentum), we can replace $w_{i,j}$ by $w_{(i,j)}$ in the first term of the left-hand side. The boundary term is decomposed in two parts corresponding to the two types of boundary conditions, essential and natural:

$$-\int_\Omega w_{(i,j)}\sigma_{ij}d\Omega + \sum_{i=1}^{n_{sd}}\left(\int_{\Gamma_{g_i}} w_i\sigma_{ij}n_j d\Gamma + \int_{\Gamma_{h_i}} w_i\sigma_{ij}n_j d\Gamma\right) + \int_\Omega w_i f_i d\Omega = 0 \quad (2.88)$$

The first part of the boundary term will now vanish as $w_i = 0$ on $\Gamma_{g_i}$ ($w_i \in \mathcal{V}_i$). The natural boundary condition (Equation 2.82) is introduced in the second boundary term and this finally leads to:

$$\int_\Omega w_{(i,j)}\sigma_{ij}d\Omega = \sum_{i=1}^{n_{sd}}\int_{\Gamma_{h_i}} w_i h_i d\Gamma + \int_\Omega w_i f_i d\Omega \quad (2.89)$$

We can now express the weak formulation $(W)$ of the problem as follows: for $i = 1, ..., n_{sd}$, given $f_i$, $g_i$ and $h_i$ as before (in $(S)$), we wish to find $u_i \in \mathcal{S}_i$ such that, for all $w_i \in \mathcal{V}_i$, the following scalar equation holds:

$$\int_\Omega w_{(i,j)}D_{ijkl}u_{(k,l)}d\Omega = \int_\Omega w_i f_i d\Omega + \sum_{i=1}^{n_{sd}}\int_{\Gamma_{h_i}} w_i h_i d\Gamma \quad (2.90)$$

In Equation 2.90, the constitutive equation 2.83 has been introduced.

Equation 2.90 can also be expressed in abstract form:

$$a(\mathbf{w},\mathbf{u}) = (\mathbf{w},\mathbf{f}) + (\mathbf{w},\mathbf{h})_\Gamma \tag{2.91}$$

**Galerkin Form**

The Galerkin ($G$) counterpart of the weak formulation ($W$) is obtained by introducing finite-dimensional subspaces of $\mathcal{S}_i$ and $\mathcal{V}_i$:

$$\mathcal{S}_i^h = \{u_i^h \mid u_i^h \simeq g_i \text{ on } \Gamma_{g_i}\} \tag{2.92}$$
$$\mathcal{V}_i^h = \{w_i^h \mid w_i^h \simeq 0 \text{ on } \Gamma_{g_i}\} \tag{2.93}$$

In order to separate the unknowns from the known boundary quantities specified on $\Gamma_{g_i}$, it is useful to decompose the displacement trial solution $u_i^h$ in the following way:

$$u_i^h = v_i^h + g_i^h \tag{2.94}$$

The $^h$ superscript denotes the approximation due to the discretization. $v_i^h$ lies in the weighting function space $\mathcal{V}_i^h$ while $g_i^h$ results in the approximate satisfaction of $u_i = g_i$ on $\Gamma_{g_i}$. ($G$) can therefore be stated in abstract notation as: given $f_i$, $g_i$ and $h_i$ as before, we wish to find $u_i^h = \left(v_i^h + g_i^h\right) \in \mathcal{S}_i^h$ such that, for all $w_i^h \in \mathcal{V}_i^h$ , the following scalar equation holds:

$$a\left(\mathbf{w}^h,\mathbf{v}^h\right) = \left(\mathbf{w}^h,\mathbf{f}\right) + \left(\mathbf{w}^h,\mathbf{h}\right)_\Gamma - a\left(\mathbf{w}^h,\mathbf{g}^h\right) \tag{2.95}$$

**Vector Notation** In order to simplify the construction of the matrix formulation, the terms of Equation 2.95 are redefined as follows in vector notation:

$$a\left(\mathbf{w}^h,\mathbf{v}^h\right) = \int_\Omega \boldsymbol{\varepsilon}\left(\mathbf{w}^h\right)^T \mathbf{D}\boldsymbol{\varepsilon}\left(\mathbf{v}^h\right) d\Omega \tag{2.96}$$
$$\left(\mathbf{w}^h,\mathbf{f}\right) = \int_\Omega \left(\mathbf{w}^h\right)^T \mathbf{f} d\Omega \tag{2.97}$$
$$\left(\mathbf{w}^h,\mathbf{h}\right)_\Gamma = \int_{\Gamma_h} \left(\mathbf{w}^h\right)^T \mathbf{h} d\Gamma \tag{2.98}$$
$$a\left(\mathbf{w}^h,\mathbf{g}^h\right) = \int_\Omega \boldsymbol{\varepsilon}\left(\mathbf{w}^h\right)^T \mathbf{D}\boldsymbol{\varepsilon}\left(\mathbf{g}^h\right) d\Omega \tag{2.99}$$

**Matrix Form**

**At the Global Level** Specifying the approximation on the displacement field by introducing the interpolation (or shape) functions $N_A(\mathbf{x})$ that will give a basis for $\mathbf{v}^h$, $\mathbf{g}^h$ and $\mathbf{w}^h$:

$$v_i^h(\mathbf{x}) = \sum_{A \in \eta - \eta_{g_i}} N_A(\mathbf{x})\, d_{iA} \tag{2.100}$$

$$g_i^h(\mathbf{x}) = \sum_{A \in \eta_{g_i}} N_A(\mathbf{x})\, g_{iA} \tag{2.101}$$

$$w_i^h(\mathbf{x}) = \sum_{A \in \eta - \eta_{g_i}} N_A(\mathbf{x})\, c_{iA} \tag{2.102}$$

where $\eta$ denotes the set of global displacement node numbers, and $\eta_{g_i} \subset \eta$ the nodes belonging to the essential boundary condition, i.e. $\Gamma_{g_i}$. $d_{iA}$ is the unknown displacement at node $A \in \eta - \eta_{g_i}$, on degree of freedom $i$. $g_{iA}$ is the known displacement at node $A \in \eta_{g_i}$, on degree of freedom $i$. Finally $c_{iA}$ is some virtual displacement at node $A \in \eta - \eta_{g_i}$, on degree of freedom $i$. Let $\mathbf{e}_i$ denote the $i$-th Euclidean basis vector for $\mathbb{R}^{n_{sd}}$. Then:

$$\mathbf{v}^h = v_i^h \mathbf{e}_i \tag{2.103}$$

$$\mathbf{g}^h = g_i^h \mathbf{e}_i \tag{2.104}$$

$$\mathbf{w}^h = w_i^h \mathbf{e}_i \tag{2.105}$$

Introducing Equations 2.100-2.105 into the Galerkin $(G)$ form (Equation 2.95) leads to the matrix form $(M)$ of the problem. Invoking arbitrariness of $\mathbf{w}^h$, one gets:

$$[\mathbf{K}]\{\mathbf{d}\} = \{\mathbf{F}\} \tag{2.106}$$

where the left hand-side matrix is derived from the vector notation of $(G)$ (Equation 2.96) and reads:

$$\mathbf{K} = \int_\Omega \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega \tag{2.107}$$

$\mathbf{d}$ is the unknown displacement vector, regrouping all $d_{iA}$. $\mathbf{B}$ is the strain-displacement operator, defined by:

$$\boldsymbol{\varepsilon}\left(\mathbf{v}^h\right) = \mathbf{B}\mathbf{d} \tag{2.108}$$

Finally, the right-hand side is explicitely given by:

$$\mathbf{F} = \int_\Omega \mathbf{N}^T \mathbf{f} d\Omega + \int_{\Gamma_h} \mathbf{N}^T \mathbf{h} d\Gamma - \mathbf{K}\mathbf{g} \tag{2.109}$$

**At the Element Level** The following elemental contributions result from Equations 2.107-2.109. First, the strain-displacement relation described in Equation 2.108 can be redefined at the element level:

$$\boldsymbol{\varepsilon}\left(\mathbf{v}_e^h\right) = \sum_{a=1}^{n_{en}} \mathbf{B}_a^e \mathbf{d}_a^e \tag{2.110}$$

where $\mathbf{v}_e^h$ is the approximated displacement field inside the $e$-th element, $\mathbf{d}_a^e$ is the nodal displacement vector associated with the $e$-th element and $n_{en}$ is the number of displacement nodes of the considered element. $\mathbf{B}_a^e$ has the following structure in the three-dimensional case ($n_{sd} = 3$):

$$\mathbf{B}_a^e = \begin{bmatrix} N_{a,1} & 0 & 0 \\ 0 & N_{a,2} & 0 \\ N_{a,2} & N_{a,1} & 0 \\ 0 & 0 & N_{a,3} \\ N_{a,3} & 0 & N_{a,1} \\ 0 & N_{a,3} & N_{a,2} \end{bmatrix} \tag{2.111}$$

where $N_{a,i} = \frac{\partial N_a}{\partial x_i}$. Introducing the following notation for the displacement equations' numbering:

$$p = n_{sd}(a-1) + i \tag{2.112}$$
$$q = n_{sd}(b-1) + j \tag{2.113}$$

$1 \leq p, q \leq n_{sd} n_{en}$. $1 \leq i, j \leq n_{sd}$, $1 \leq a, b \leq n_{en}$, we can specify the definition of the elemental contributions. In the following, the symbol $\underset{a=1}{\overset{n_{el}}{A}}$ denotes the assembly operation, and $n_{el}$ is the number of elements.

- $\mathbf{K} = \underset{a=1}{\overset{n_{el}}{A}} \mathbf{k}^e$, and $\mathbf{k}^e = \left[k_{pq}^e\right]$ with $k_{pq}^e = \mathbf{e}_i^T \int_{\Omega^e} \mathbf{B}_a^{eT} \mathbf{D} \mathbf{B}_b^e d\Omega \mathbf{e}_j$
- $\mathbf{F} = \underset{a=1}{\overset{n_{el}}{A}} \mathbf{f}^e$, and $\mathbf{f}^e = \left[f_p^e\right]$ with $f_p^e = \int_{\Omega^e} N_a^e f_i d\Omega + \int_{\Gamma_{h_i}^e} N_a^e h_i d\Omega - k_{pq}^e g_q^e$, and $g_q^e = g_j\left(\mathbf{x}_b^e\right)$ if and only if $\mathbf{x}_b^e \in \boldsymbol{\Gamma}_g$

### 2.3.2 Plastic Case

#### Introduction

Following the guidelines established in the elastic case, we can now apply the standard finite element formulation to elastoplasticity. The same hypotheses still hold, as well as the vector notations defined before. As an additional hypothesis, the material will be considered to be

perfectly elasto-plastic (**no hardening**). As we deal with plasticity, which means that the stress-strain state of the material is history- or path-dependent, we can't use the constitutive equation 2.83 in the integral form as we did when we dealt with linear elasticity. Instead, we have to express it in the rate or incremental form (both forms are equivalent as long as we speak of time-independent processes, which is the case here). The incremental constitutive equation reads:

$$\boldsymbol{\Delta\sigma}(\mathbf{u}) = \mathbf{D}\left[\boldsymbol{\Delta\varepsilon}(\mathbf{u}) - \boldsymbol{\Delta\varepsilon}^{p}(\mathbf{u})\right] \tag{2.114}$$

In the preceding equation, the **additive decomposition of the strain field** into an elastic and a plastic contribution has been used in order to replace $\boldsymbol{\Delta\varepsilon}^{e}(\mathbf{u})$ by $\boldsymbol{\Delta\varepsilon}(\mathbf{u}) - \boldsymbol{\Delta\varepsilon}^{p}(\mathbf{u})$:

$$\boldsymbol{\Delta\varepsilon}(\mathbf{u}) = \boldsymbol{\Delta\varepsilon}^{e}(\mathbf{u}) + \boldsymbol{\Delta\varepsilon}^{p}(\mathbf{u}) \tag{2.115}$$

Recalling the three other basic ingredients of plasticity, i.e. the definition of a **yield function**:

$$f(\boldsymbol{\sigma}) = 0 \tag{2.116}$$

the introduction of a **flow rule**:

$$\boldsymbol{\Delta\varepsilon}^{p}(\mathbf{u}) = \Delta\gamma \frac{\partial g}{\partial \boldsymbol{\sigma}} = \Delta\gamma\ \mathbf{r} \tag{2.117}$$

where $\Delta\gamma$ is the scalar plastic multiplier and $g$ is a function called the plastic potential, and finally **consistency** implying:

$$\Delta\gamma\ \dot{f}(\boldsymbol{\sigma}) = 0 \tag{2.118}$$

we can express the elastoplastic problem in the following form:

**Strong Form**

Given as before $f_i : \Omega \longrightarrow \mathbb{R}$, $g_i : \Gamma_{g_i} \longrightarrow \mathbb{R}$ and $h_i : \Gamma_{h_i} \longrightarrow \mathbb{R}$, we wish to find $u_i : \overline{\Omega} \longrightarrow \mathbb{R}$ such that:

$$\sigma_{ij,j} + f_i = 0 \qquad \text{in } \Omega \tag{2.119}$$

$$u_i = g_i \qquad \text{on } \Gamma_{g_i} \tag{2.120}$$

$$\sigma_{ij} n_j = h_i \qquad \text{on } \Gamma_{h_i} \tag{2.121}$$

The evolution of the stress tensor $\boldsymbol{\sigma}$ is related with the unknown displacement field $\mathbf{u}$ through the constitutive relation 2.114.

**Weak Form**

Premultiplying Equation 2.119 by a weighting function $w_i \in \mathcal{V}_i$ and integrating over the domain:

$$\int_\Omega w_i \left(\sigma_{ij,j} + f_i\right) d\Omega = 0 \tag{2.122}$$

Integrating the first expression by parts, using the symmetry of $\sigma_{ij}$ and the fact that $w_i = 0$ on $\Gamma_{g_i}$ leads to:

$$\int_\Omega w_{(i,j)} \sigma_{ij}\left(\mathbf{u}\right) d\Omega = \sum_{i=1}^{n_{sd}} \int_{\Gamma_{h_i}} w_i h_i d\Gamma + \int_\Omega w_i f_i d\Omega \tag{2.123}$$

And the weak form can be expressed as: given $f_i$, $g_i$ and $h_i$ as before, we wish to find $u_i \in \mathcal{S}_i$ such that, for all $w_i \in \mathcal{V}_i$, Equation 2.123 holds.

**Galerkin Form**

Introducing the finite-dimension spaces $\mathcal{S}_i^h \subset \mathcal{S}_i$ and $\mathcal{V}_i^h \subset \mathcal{V}_i$, we can express the Galerkin form of the problem as: given $f_i$, $g_i$ and $h_i$ as before, we wish to find $u_i^h = \left(v_i^h + g_i^h\right) \in \mathcal{S}_i^h$ (with $v_i^h \in \mathcal{V}_i^h$) such that, for all $w_i^h \in \mathcal{V}_i^h$, the following scalar equation holds:

$$\int_\Omega w_{(i,j)}^h \sigma_{ij}\left(\mathbf{u}^h\right) d\Omega = \sum_{i=1}^{n_{sd}} \int_{\Gamma_{h_i}} w_i^h h_i d\Gamma + \int_\Omega w_i^h f_i d\Omega \tag{2.124}$$

where $\sigma_{ij}$ is computed through the constitutive equation 2.114.

If we compare this form with the Galerkin form that we have derived in the case of elasticity, we see that we have dropped the decomposition of $u_i^h$ into $v_i^h$ and $g_i^h$. This has been done for the sake of simplicity. We will see in the matrix formulation described next what this implies when we introduce the global displacement vector $\mathbf{u}$.

Introducing the vector notation, we can rewrite Equation 2.124 as:

$$\int_\Omega \boldsymbol{\varepsilon}\left(\mathbf{w}^h\right)^T \boldsymbol{\sigma}\left(\mathbf{u}^h\right) d\Omega = \int_{\Gamma_h} \left(\mathbf{w}^h\right)^T \mathbf{h} d\Gamma + \int_\Omega \left(\mathbf{w}^h\right)^T \mathbf{f} d\Omega \tag{2.125}$$

**Matrix Form**

We now specify the approximations of the displacement field as follows:

$$u_i^h(\mathbf{x}) = \sum_{A\in\eta} N_A(\mathbf{x})\, u_{iA} \tag{2.126}$$

$$w_i^h(\mathbf{x}) = \sum_{A\in\eta-\eta_{g_i}} N_A(\mathbf{x})\, c_{iA} \tag{2.127}$$

And now comes the following assumption, taking into account the fact that $u_{iA}$ has to contain the prescribed displacements on $\eta_{g_i}$:

$$u_{iA} = \begin{array}{ll} \nearrow & \text{unknown displacement} \quad \text{if } A \in \eta - \eta_{g_i} \\ \searrow & g_{iA} \qquad\qquad\qquad\quad \text{if } A \in \eta_{g_i} \end{array} \tag{2.128}$$

Introducing Equations 2.126-2.127 into the Galerkin form, and invoking arbitrariness of $\mathbf{w}^h$, one gets:

$$\int_\Omega \mathbf{B}^T\boldsymbol{\sigma}\left(\mathbf{u}^h\right) d\Omega = \int_{\Gamma_h} \mathbf{N}^T\mathbf{h}\, d\Gamma + \int_\Omega \mathbf{N}^T\mathbf{f}\, d\Omega \overset{\text{def.}}{=} \mathbf{F}_{ext,n+1} \tag{2.129}$$

As we deal with nonlinear elastoplasticity, a residual-driven iterative scheme will be necessary in order to converge towards the correct solution. The preceding equation must therefore be satisfied at step $(n+1)$, iteration $(i+1)$:

$$\int_\Omega \mathbf{B}^T\boldsymbol{\sigma}_{n+1}^{i+1} d\Omega = \mathbf{F}_{ext,n+1} \tag{2.130}$$

**Linearization of the Equilibrium Equation** Rewriting Equation 2.130 as:

$$\int_\Omega \mathbf{B}^T\boldsymbol{\Delta\sigma}_{n+1}^{i+1} d\Omega = \mathbf{F}_{ext,n+1} - \int_\Omega \mathbf{B}^T\boldsymbol{\sigma}_n d\Omega \tag{2.131}$$

where we have introduced:

$$\boldsymbol{\sigma}_{n+1}^{i+1} = \boldsymbol{\sigma}_n + \boldsymbol{\Delta\sigma}_{n+1}^{i+1} \tag{2.132}$$

We will now use the incremental constitutive equation (Equation 2.114) in order to express $\boldsymbol{\Delta\sigma}_{n+1}^{i+1}$ in function of $\boldsymbol{\Delta\varepsilon}_{n+1}^{i+1}$ (dropping the indexes in order to simplify notation):

$$\boldsymbol{\Delta\sigma}\left(\mathbf{u}^h\right) = \mathbf{D}\left[\boldsymbol{\Delta\varepsilon}\left(\mathbf{u}^h\right) - \boldsymbol{\Delta\varepsilon}^p\left(\mathbf{u}^h\right)\right] \tag{2.133}$$

If we are in plastic regime, we can apply consistency as $\dot{f}(\boldsymbol{\sigma}) = 0$ (plastic process $\Rightarrow \Delta\gamma > 0$). This leads to:

$$\dot{f}(\boldsymbol{\sigma}) = \frac{\partial f}{\partial \boldsymbol{\sigma}}^T \boldsymbol{\Delta\sigma} \overset{\text{def.}}{=} \mathbf{a}^T \boldsymbol{\Delta\sigma} = 0 \tag{2.134}$$

and multiplying Equation 2.133 by $\mathbf{a}^T$ we can conclude:

$$\mathbf{a}^T \mathbf{D} \left[\boldsymbol{\Delta\varepsilon} - \boldsymbol{\Delta\varepsilon}^p\right] = 0 \tag{2.135}$$

Introducing the flow rule (Equation 2.117):

$$\mathbf{a}^T \mathbf{D} \boldsymbol{\Delta\varepsilon} - \mathbf{a}^T \mathbf{D} \Delta\gamma \mathbf{r} = 0 \tag{2.136}$$

we can get an explicit expression for the plastic multiplier $\Delta\gamma$:

$$\Delta\gamma = \frac{\mathbf{a}^T \mathbf{D} \boldsymbol{\Delta\varepsilon}}{\mathbf{a}^T \mathbf{D} \mathbf{r}} \tag{2.137}$$

Reintroducing Equation 2.137 into Equation 2.133 leads to:

$$\boldsymbol{\Delta\sigma} = \mathbf{D} \left[\boldsymbol{\Delta\varepsilon} - \frac{\mathbf{a}^T \mathbf{D} \boldsymbol{\Delta\varepsilon}}{\mathbf{a}^T \mathbf{D} \mathbf{r}} \mathbf{r}\right] \tag{2.138}$$

and rearranging terms yields:

$$\boldsymbol{\Delta\sigma} = \left[\mathbf{D} - \frac{(\mathbf{D}\mathbf{r})(\mathbf{D}\mathbf{a})^T}{\mathbf{a}^T \mathbf{D} \mathbf{r}}\right] \boldsymbol{\Delta\varepsilon} \overset{\text{def.}}{=} \mathbf{D}^{ep} \boldsymbol{\Delta\varepsilon} \tag{2.139}$$

that we can express in compact form at iteration $i+1$, step $n+1$:

$$\boldsymbol{\Delta\sigma}_{n+1}^{i+1} = \mathbf{D}_{n+1}^{ep,i+1} \boldsymbol{\Delta\varepsilon}_{n+1}^{i+1} \tag{2.140}$$

or:

$$\boldsymbol{\Delta\sigma}_{n+1}^{i+1} = \mathbf{D}_{n+1}^{ep,i+1} \mathbf{B} \boldsymbol{\Delta u}_{n+1}^{i+1} \tag{2.141}$$

During the iterative process, the accumulated value of the displacement is updated as follows:

$$\boldsymbol{\Delta u}_{n+1}^{i+1} = \boldsymbol{\Delta u}_{n+1}^{i} + \boldsymbol{\Delta u} \tag{2.142}$$

**Remark 15** *the expression for $\Delta\gamma$ given in Equation 2.137 helps us with the building of the tangent operator $\mathbf{D}_{n+1}^{ep,i+1}$, but it should be noted that the actual computation of $\Delta\gamma$ is made during*

*the stress return algorithm defined by Equation 2.76.*

**Matrix Form (Incremental Nonlinear Version)** We are now ready to define the incremental matrix form ($M_{n+1}^{i+1}$) of our problem that we will later import into the iterative nonlinear scheme. Rewriting Equation 2.130 using the definition of $\mathbf{\Delta\sigma}_{n+1}^{i+1}$ (Equation 2.140) we get:

$$[\mathbf{K}]\,\{\mathbf{\Delta u}\} = \{\mathbf{F}\} \tag{2.143}$$

with the left-hand side:

$$\mathbf{K} = \int_{\Omega} \mathbf{B}^T \mathbf{D}^{ep} \mathbf{B} d\Omega \tag{2.144}$$

and the right-hand side:

$$\mathbf{F} = \mathbf{F}_{ext,n+1} - \int_{\Omega} \mathbf{B}^T \boldsymbol{\sigma}_{n+1}^{i} d\Omega \tag{2.145}$$

In the preceding development, the definition of the accumulated value of $\mathbf{\Delta u}_{n+1}^{i+1}$ (given in Equation 2.142) has been introduced in order to isolate $\mathbf{\Delta u}$ in the left-hand side, and to compute:

$$\boldsymbol{\sigma}_{n+1}^{i} = \boldsymbol{\sigma}_n + \mathbf{\Delta\sigma}_{n+1}^{i} \tag{2.146}$$

The nonlinear strategy used to solve the problem can finally be summarized in the following manner:

- Step initialisation: $n = 0$
- 1. Compute $\mathbf{F}_{ext,n+1}$
- Iterative loop initiatisation: $i = 0$
- $\mathbf{u}_{n+1}^{0} = \mathbf{u}_n$ and $\mathbf{\Delta u}_{n+1}^{0} = \mathbf{0}$
- 2. Convergence test: $\left\|\mathrm{RHS}(M_{n+1}^{i+1})\right\| < TOL$ ?
- YES $\Rightarrow n++$, goto 1.
- Solve $(M_{n+1}^{i+1})$ (Equation 2.143)
- $\mathbf{u}_{n+1}^{i+1} = \mathbf{u}_{n+1}^{i} + \mathbf{\Delta u}$ and $\mathbf{\Delta u}_{n+1}^{i+1} = \mathbf{\Delta u}_{n+1}^{i} + \mathbf{\Delta u}$
- $i++$, goto 2.

### 2.3.3 Quality of the Standard Displacement-Based Finite Element Solution

#### Introduction

The best approximation property [28] stipulates that the Galerkin method gives the best solution in $\mathbf{S}^h$, i.e. the numerical solution $\mathbf{u}^h$ will be the closest to $\mathbf{u}$, the exact solution, in the following sense:

$$a\left(\mathbf{e},\mathbf{e}\right) \leq a\left(\mathbf{U}^h-\mathbf{u},\mathbf{U}^h-\mathbf{u}\right) \tag{2.147}$$

In Equation 2.147, $\mathbf{e}=\mathbf{u}^h-\mathbf{u}$ is the error in the finite element approximation while $\mathbf{U}^h$ represents any member of $\mathbf{S}^h$. This property guarantees good behavior of the finite element scheme, except in the case where **no** function in $\mathbf{S}^h$ is a good approximation to $\mathbf{u}$. This happens for instance in the case of incompressible elasticity or, by extension, for incompressible (or dilatant) plastic flow. Popular methods to overcome problematic behavior of standard finite elements include mixed formulations involving a new unknown field (pressure), or selective integration techniques. The former method will be treated in depth in the remainder of this work, while the latter is addressed now.

#### Illustration of a Fundamental Difficulty

The kinematic condition of incompressibility reads:

$$u_{i,i}=0 \tag{2.148}$$

When dealing with mixed displacement-pressure formulations, it is satisfied in the Galerkin sense when:

$$\left(q^h, div\ \mathbf{u}^h\right)=\sum_{e=1}^{n_{el}}\int_{\Omega^e} q^h div\ \mathbf{u}^h d\Omega=0 \tag{2.149}$$

where $q^h$ is a pressure weighting function (we will come back to such mixed formulations later in this work). Suppose we deal with linear displacement-constant pressure triangular elements. This implies that $q^h$ is constant over each element and therefore:

$$\int_{\Omega^e} div\ \mathbf{u}^h d\Omega=0 \qquad 1\leq e\leq n_{el} \tag{2.150}$$

What does this mean? It states the fact that the area of every element has to remain constant. As a result, node **N** (see Figure 2-12) is required to move horizontally in order to satisfy this kinematic constraint in the lower triangular element, and required to move vertically as a result of the kinematic constraint in the upper triangular element; therefore node **N** can't move ($\mathbf{u}^h=0$) and the reasoning can be extended to a $n\times n$ mesh (see Figure 2-13). This phenomenon is referred to as volumetric mesh **locking**.

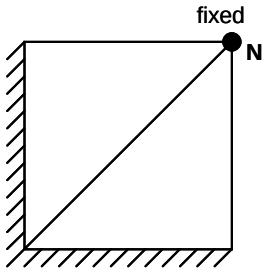



Figure 2-12: Illustration of locking

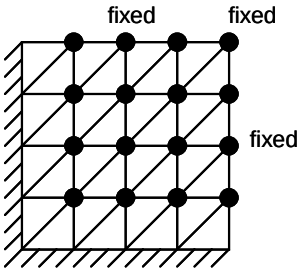



Figure 2-13: $n \times n$ mesh

**One Remedy: the $\overline{\mathbf{B}}$ Approach**

**Development** The $\overline{\mathbf{B}}$ approach proposed in [28] is a strain-projection method: a small change is introduced in the standard strain-displacement matrix $\mathbf{B}$ (which was defined in Equations 2.110 and 2.111) in order to underintegrate the volumetric components of the deformation (selective integration), and therefore to release the constraint of incompressibility by lowering the number of points where it has to be satisfied.

The approach indeed introduces a modification of the dilatational contribution to the standard $\mathbf{B}$ matrix, $\mathbf{B}_{dil}$, which is underintegrated or averaged over the element. The new strain-displacement relation then reads:

$$\varepsilon\left(\mathbf{u}^h\right) = \overline{\mathbf{B}}\mathbf{d} \tag{2.151}$$

with $\overline{\mathbf{B}}$ defined as:

$$\overline{\mathbf{B}} = \mathbf{B}_{dev} + \overline{\mathbf{B}}_{dil} \tag{2.152}$$

On the one hand, $\mathbf{B}_{dev}$ results from the volumetric-deviatoric split of the original $\mathbf{B}$ matrix:

$$\mathbf{B}_{dev} = \mathbf{B} - \mathbf{B}_{dil} \tag{2.153}$$

and on the other hand, $\mathbf{B}_{dil} = [\mathbf{B}^1_{dil}, ..., \mathbf{B}^a_{dil}, ...]$, $1 < a < n_{en}$, where $n_{en}$ is the number of nodes of the element. $\mathbf{B}^a_{dil}$ is defined by:

$$\mathbf{B}^a_{dil} = \frac{1}{3}\begin{bmatrix} N_{a,1} & N_{a,2} & N_{a,3} \\ N_{a,1} & N_{a,2} & N_{a,3} \\ 0 & 0 & 0 \\ N_{a,1} & N_{a,2} & N_{a,3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.154}$$

Combining all the former equations leads to $\overline{\mathbf{B}} = [\overline{\mathbf{B}^1}, ..., \overline{\mathbf{B}^a}, ...]$ with:

$$\overline{\mathbf{B}^a} = \begin{bmatrix} N_{a,1} + B_4 & B_6 & B_8 \\ B_4 & N_{a,2} + B_6 & B_8 \\ N_{a,2} & N_{a,1} & 0 \\ B_4 & B_6 & N_{a,3} + B_8 \\ N_{a,3} & 0 & N_{a,1} \\ 0 & N_{a,3} & N_{a,2} \end{bmatrix} \tag{2.155}$$

where:

$$B_4 = \frac{1}{3}\left(\overline{B}_1 - N_{a,1}\right) \tag{2.156}$$

$$B_6 = \frac{1}{3}\left(\overline{B}_2 - N_{a,2}\right) \tag{2.157}$$

$$B_8 = \frac{1}{3}\left(\overline{B}_3 - N_{a,3}\right) \tag{2.158}$$

Finally the method reduces to the appropriate definition of $\overline{B}_1$, $\overline{B}_2$ and $\overline{B}_3$:

$$\overline{B}_i(\boldsymbol{\xi}) = \sum_{\tilde{a}=1}^{\tilde{n}_{int}} \widetilde{N}_{\tilde{a}}(\boldsymbol{\xi}) B_{i\tilde{a}} \tag{2.159}$$

**Underintegration Approach** In this approach, one takes:

$$B_{i\tilde{a}} = B_i(\boldsymbol{\xi}_{\tilde{a}}) \tag{2.160}$$

For instance, the value at the center of the element can be used to compute the dilatational contribution for the bilinear quadrilateral element (which "normal" integration rule is $2 \times 2$):

$$\overline{B}_i(\boldsymbol{\xi}) = B_i(\mathbf{0}) \tag{2.161}$$

**Mean-Dilatation Formulation** In this case:

$$\overline{B}_i(\boldsymbol{\xi}) = \frac{\int_{\Omega^e} B_i \, d\Omega}{\int_{\Omega^e} d\Omega} \tag{2.162}$$

**Limitations of the Present $\overline{\mathbf{B}}$ Approach** This approach has been shown to overcome volumetric locking in the incompressible case for both elastic and plastic regimes. Two shortcomings remain nevertheless and speak for the building of a more general formulation:

- there is no $\overline{\mathbf{B}}$ formulation for the 2D three-node triangular element (as its "normal" integration rule reduces to 1 Gauss point)
- the efficiency of this method in the case of dilatant plastic flow, discussed in [12], is doubtful, as we will see later in the benchmarks section

This justifies the search for a formulation allowing us to overcome these problems: the stabilized mixed formulation, described in the next section.

# Chapter 3

# Stabilized Methods

## 3.1 A Mixed Approach to Elasticity

### 3.1.1 Preliminaries

In this section, we consider a problem with the following assumptions: the strains are small and the material is supposed to be elastic and isotropic. As we have seen in the previous section, some continuum mechanics problems cannot be solved in a suitable fashion with the help of the standard finite element method, i.e. with the displacement formulation. This is the case in particular for problems involving incompressible or nearly incompressible behavior, either in the elastic or in the plastic range. We have already described a method, the $\overline{\mathbf{B}}$ approach [28], capable of circumventing this shortcoming in certain cases. In the following section, we will examine now a mixed displacement-pressure formulation capable of handling the incompressible limit, and that will still work in the compressible case.

The main problem in the application of the standard displacement formulation lies in the determination of the mean stress, or pressure, which is related to the volumetric part of the strain. The trick is to introduce a new unknown representing explicitly the pressure field $p$, for which we will solve the problem in the same manner as we already do for the displacement field $\mathbf{u}$. We separate this pressure from the total stress field $\boldsymbol{\sigma}$ writing the following constitutive equation in vector notations:

$$\boldsymbol{\sigma} = \overline{\mathbf{D}}\boldsymbol{\varepsilon}(\mathbf{u}) + \mathbf{1}p \tag{3.1}$$

In Equation 3.1, $\boldsymbol{\sigma}$ is the total stress column-vector:

$$\boldsymbol{\sigma} = \left\{ \begin{array}{c} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ \sigma_{33} \\ \sigma_{13} \\ \sigma_{23} \end{array} \right\} \tag{3.2}$$

$\overline{\mathbf{D}}$ is the deviatoric projection of the elasticity matrix $\mathbf{D}$:

$$\overline{\mathbf{D}} = \mathbf{D}\left(\mathbf{I} - \frac{1}{3}\mathbf{1}\mathbf{1}^T\right) \tag{3.3}$$

where $\mathbf{D}$ can be expressed in function of the Lamé parameters $\lambda$ and $\mu$:

$$\mathbf{D} = \left[ \begin{array}{cccccc} \lambda + 2\mu & \lambda & 0 & \lambda & 0 & 0 \\ \lambda & \lambda + 2\mu & 0 & \lambda & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 \\ \lambda & \lambda & 0 & \lambda + 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{array} \right] \tag{3.4}$$

and the following relations express the Lamé parameters with respect to Young modulus $E$ and Poisson ratio $\nu$:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{3.5}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{3.6}$$

**1** is the vectorial representation of the Kronecker delta $\delta_{ij}$:

$$\mathbf{1} = \left\{ \begin{array}{c} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{array} \right\} \tag{3.7}$$

and **I** is the $6 \times 6$ identity matrix. Finally, the strain vector $\boldsymbol{\varepsilon}(\mathbf{u})$ in Equation 3.1 writes:

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \left\{ \begin{array}{c} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \\ u_{3,3} \\ u_{1,3} + u_{3,1} \\ u_{2,3} + u_{3,2} \end{array} \right\} \tag{3.8}$$

where the comma denotes differentiation in the following sense:

$$u_{i,j} = \frac{\partial u_i}{\partial x_j} \tag{3.9}$$

### 3.1.2 The Mixed Elastic Boundary Value Problem

#### Strong Form

Consider a body $\Omega$, its boundary $\partial\Omega = \Gamma = \overline{\Gamma_{g_i} \cup \Gamma_{h_i}}$ where $\Gamma_{g_i} \cap \Gamma_{h_i} = \emptyset$ with $i = 1, ..., n_{sd}$, where $n_{sd}$ stands for the number of spatial dimensions. As we have introduced a new unknown, the pressure field $p$, we need an additional equation so that our problem can be solved. So we start with the following strong form $(S)$ of the mixed boundary value problem: given $f_i : \Omega \longrightarrow$

$\mathbb{R}$, $g_i : \Gamma_{g_i} \longrightarrow \mathbb{R}$ and $h_i : \Gamma_{h_i} \longrightarrow \mathbb{R}$, we wish to find $u_i : \overline{\Omega} \longrightarrow \mathbb{R}$ and $p : \overline{\Omega} \longrightarrow \mathbb{R}$ such that:

$$\begin{aligned} \sigma_{ij,j} + f_i &= 0 \qquad \text{in } \Omega & (3.10)\\ u_{i,i} - \frac{p}{K} &= 0 \qquad \text{in } \Omega & (3.11)\\ u_i &= g_i \qquad \text{on } \Gamma_{g_i} & (3.12)\\ \sigma_{ij} n_j &= h_i \qquad \text{on } \Gamma_{h_i} & (3.13) \end{aligned}$$

with the linear elastic constitutive relation $\sigma_{ij}(\mathbf{u}) = D_{ijkl} u_{(k,l)}$.

Equation 3.10 represents the static equilibrium condition and derives from conservation of linear momentum. Equation 3.11 can be viewed as an additional constitutive equation for the pressure field $p$. $K$, the bulk modulus, is expressed by:

$$K = \lambda + \frac{2}{3}\mu = \frac{E}{3(1-2\nu)} \tag{3.14}$$

As $\nu$ tends to 0.5 and reaches the incompressible limit, $K$ tends to infinity. At the limit, the second term of the left-hand side of Equation 3.11 will vanish and this equation will become the incompressibility condition, deriving from the balance of mass governing principle:

$$div\ (\mathbf{u}) = u_{i,i} = 0 \tag{3.15}$$

Finally Equation 3.12 is an essential boundary condition on the displacement field which must be enforced on the $\Gamma_{g_i}$ part of the boundary, while Equation 3.13 is a natural boundary condition appearing on $\Gamma_{h_i}$.

**Weak Form**

The weak formulation of the mixed problem is similar to the weak formulation of the standard displacement formulation, except we need to introduce a term which implies the satisfaction of Equation 3.11. We begin with the definition of the following spaces:

$$\begin{aligned} \mathcal{S}_i &= \{u_i \in H^1(\Omega) \mid u_i = g_i \text{ on } \Gamma_{g_i}\} & (3.16)\\ \mathcal{V}_i &= \{w_i \in H^1(\Omega) \mid w_i = 0 \text{ on } \Gamma_{g_i}\} & (3.17)\\ \mathcal{P} &= \{p \in L^2(\Omega)\} & (3.18) \end{aligned}$$

$\mathcal{S}_i$ represents the trial solution space, $\mathcal{V}_i$ the weighting function space and $\mathcal{P}$ the pressure space. As there are no explicit boundary conditions on the pressure field, there is no need for two pressure spaces: both the pressure solution and the pressure weighting function will lie in $\mathcal{P}$.

On the one hand, similarly to what has been done in chapter 2, we retrieve the following

weak form of the equilibrium equation (see Equation 2.89):

$$\int_\Omega w_{(i,j)}\sigma_{ij}d\Omega = \sum_{i=1}^{n_{sd}}\int_{\Gamma_{h_i}} w_i h_i d\Gamma + \int_\Omega w_i f_i d\Omega \tag{3.19}$$

On the other hand, we premultiply Equation 3.11 by the weighting function $q$ and integrate over the domain:

$$\int_\Omega q\left(u_{i,i} - \frac{p}{K}\right)d\Omega = 0 \tag{3.20}$$

And we can express the weak formulation $(W)$ of the problem as follows: given $f_i$, $g_i$ and $h_i$ as before (in $(S)$), we wish to find $u_i \in \mathcal{S}_i$ and $p \in \mathcal{P}$ such that, for all $w_i \in \mathcal{V}_i$ and $q \in \mathcal{P}$, the following scalar equations hold:

$$\int_\Omega w_{(i,j)}\overline{D}_{ijkl}u_{(k,l)}d\Omega + \int_\Omega w_{(i,j)}\delta_{ij}p d\Omega = \int_\Omega w_i f_i d\Omega + \sum_{i=1}^{n_{sd}}\int_{\Gamma_{h_i}} w_i h_i d\Gamma \tag{3.21}$$

$$\int_\Omega q u_{i,i}d\Omega - \int_\Omega q\frac{p}{K}d\Omega = 0 \tag{3.22}$$

In Equation 3.21, the tensorial counterpart of the deviatoric-volumetric split expressed by Equation 3.1 has been introduced:

$$\sigma_{ij} = \overline{D}_{ijkl}\varepsilon_{kl} + \delta_{ij}p = \overline{D}_{ijkl}u_{(k,l)} + \delta_{ij}p \tag{3.23}$$

Equations 3.21 and 3.22 can be expressed in abstract form:

$$\overline{a}\left(\mathbf{w},\mathbf{u}\right) + \left(div\ \left(\mathbf{w}\right),p\right) = \left(\mathbf{w},\mathbf{f}\right) + \left(\mathbf{w},\mathbf{h}\right)_\Gamma \tag{3.24}$$

$$\left(q, div\ \left(\mathbf{u}\right)\right) - \left(q,\frac{p}{K}\right) = 0 \tag{3.25}$$

In the transformation from Equation 3.21 to Equation 3.24, the following identity has been used:

$$\int_\Omega w_{(i,j)}\delta_{ij}p d\Omega = \int_\Omega w_{(i,i)}p d\Omega = \int_\Omega w_{i,i}p d\Omega \tag{3.26}$$

### Galerkin Form

We now introduce finite-dimensional subspaces of $\mathcal{S}_i$ , $\mathcal{V}_i$, and $\mathcal{P}$ (respectively $\mathcal{S}_i^h$, $\mathcal{V}_i^h$ and $\mathcal{P}^h$). $(G)$ can therefore be stated in abstract notation as: given $f_i$, $g_i$ and $h_i$ as before, we wish to find $u_i^h = \left(v_i^h + g_i^h\right) \in \mathcal{S}_i^h$ (with $v_i^h \in \mathcal{V}_i^h$) and $p^h \in \mathcal{P}^h$ such that, for all $w_i^h \in \mathcal{V}_i^h$ and $q^h \in \mathcal{P}^h$,

the following scalar equations hold:

$$\overline{a}\left(\mathbf{w}^h,\mathbf{v}^h\right)+\left(div\ \left(\mathbf{w}^h\right),p^h\right) = \left(\mathbf{w}^h,\mathbf{f}\right)+\left(\mathbf{w}^h,\mathbf{h}\right)_\Gamma-\overline{a}\left(\mathbf{w}^h,\mathbf{g}^h\right) \quad (3.27)$$

$$\left(q^h,div\ \left(\mathbf{v}^h\right)\right)-\left(q^h,\frac{p^h}{K}\right) = -\left(q^h,div\ \left(\mathbf{g}^h\right)\right) \quad (3.28)$$

**Vector Notation** In order to simplify the construction of the matrix formulation, the terms of Equations 3.27 and 3.28 are redefined as follows in vector notation:

$$\overline{a}\left(\mathbf{w}^h,\mathbf{v}^h\right) = \int_\Omega \boldsymbol{\varepsilon}\left(\mathbf{w}^h\right)^T\overline{\mathbf{D}}\boldsymbol{\varepsilon}\left(\mathbf{v}^h\right)d\Omega \quad (3.29)$$

$$\left(div\ \left(\mathbf{w}^h\right),p^h\right) = \int_\Omega \boldsymbol{\varepsilon}\left(\mathbf{w}^h\right)^T\mathbf{1}p^h d\Omega \quad (3.30)$$

$$\left(\mathbf{w}^h,\mathbf{f}\right) = \int_\Omega \left(\mathbf{w}^h\right)^T\mathbf{f}\,d\Omega \quad (3.31)$$

$$\left(\mathbf{w}^h,\mathbf{h}\right)_\Gamma = \int_{\Gamma_h} \left(\mathbf{w}^h\right)^T\mathbf{h}d\Gamma \quad (3.32)$$

$$\overline{a}\left(\mathbf{w}^h,\mathbf{g}^h\right) = \int_\Omega \boldsymbol{\varepsilon}\left(\mathbf{w}^h\right)^T\overline{\mathbf{D}}\boldsymbol{\varepsilon}\left(\mathbf{g}^h\right)d\Omega \quad (3.33)$$

$$\left(q^h,div\ \left(\mathbf{v}^h\right)\right) = \int_\Omega q^h\mathbf{1}^T\boldsymbol{\varepsilon}\left(\mathbf{v}^h\right)d\Omega \quad (3.34)$$

$$\left(q^h,\frac{p^h}{K}\right) = \int_\Omega q^h\frac{p^h}{K}d\Omega \quad (3.35)$$

$$\left(q^h,div\ \left(\mathbf{g}^h\right)\right) = \int_\Omega q^h\mathbf{1}^T\boldsymbol{\varepsilon}\left(\mathbf{g}^h\right)d\Omega \quad (3.36)$$

### Matrix Form

**At the Global Level** On the one hand, we introduce displacement shape functions $N_A(\mathbf{x})$ as in chapter 2 (see Equations 2.100-2.102).

On the other hand, the approximation of the pressure field involves another set of interpolation functions $\widetilde{N}_{\widetilde{A}}(\mathbf{x})$ (which could be equal to $N_A(\mathbf{x})$ but do not have to), and therefore we can define explicitely $p^h$ and $q^h$:

$$p^h(\mathbf{x}) = \sum_{\widetilde{A}\in\widetilde{\eta}}\widetilde{N}_{\widetilde{A}}(\mathbf{x})\,p_{\widetilde{A}} \quad (3.37)$$

$$q^h(\mathbf{x}) = \sum_{\widetilde{A}\in\widetilde{\eta}}\widetilde{N}_{\widetilde{A}}(\mathbf{x})\,q_{\widetilde{A}} \quad (3.38)$$

where $\widetilde{\eta}$ denotes the set of global pressure node numbers. $p_{\widetilde{A}}$ is the unknown pressure at node $\widetilde{A}\in\widetilde{\eta}$, and $q_{\widetilde{A}}$ is some virtual pressure at node $\widetilde{A}\in\widetilde{\eta}$.

Introducing shape functions into the Galerkin $(G)$ form (Equations 3.27 and 3.28) leads to the matrix form $(M)$ of the problem. As $(G)$ must hold for any $\mathbf{w}^h$ and $q^h$, we get:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ \mathbf{K}_{pu} & \mathbf{K}_{pp} \end{bmatrix} \begin{Bmatrix} \mathbf{d} \\ \mathbf{p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_u \\ \mathbf{F}_p \end{Bmatrix} \tag{3.39}$$

where the left hand-side matrix coefficients are derived from the vector notation of $(G)$ (Equations 3.29-3.36) and read:

$$\mathbf{K}_{uu} = \int_\Omega \mathbf{B}^T \overline{\mathbf{D}} \mathbf{B} d\Omega \tag{3.40}$$

$$\mathbf{K}_{up} = \int_\Omega \mathbf{B}^T \mathbf{1} \widetilde{\mathbf{N}} d\Omega \tag{3.41}$$

$$\mathbf{K}_{pu} = \int_\Omega \widetilde{\mathbf{N}}^T \mathbf{1}^T \mathbf{B} d\Omega \tag{3.42}$$

$$\mathbf{K}_{pp} = -\frac{1}{K} \int_\Omega \widetilde{\mathbf{N}}^T \widetilde{\mathbf{N}} d\Omega \tag{3.43}$$

$\mathbf{d}$ is the unknown displacement vector, regrouping all $d_{iA}$. $\mathbf{p}$ is the unknown pressure vector, composed by all $p_{\widetilde{A}}$. $\widetilde{\mathbf{N}}$ is a vector containing the pressure interpolation functions $\widetilde{N}_{\widetilde{A}}(\mathbf{x})$ and $\mathbf{B}$ is the usual strain-displacement operator, defined by:

$$\boldsymbol{\varepsilon}\left(\mathbf{u}^h\right) = \mathbf{B}\mathbf{d} \tag{3.44}$$

Finally, the right-hand side is explicitely given by:

$$\mathbf{F}_u = \int_\Omega \mathbf{N}^T \mathbf{f} d\Omega + \int_{\Gamma_h} \mathbf{N}^T \mathbf{h} d\Gamma - \mathbf{K}_{uu}\mathbf{g} \tag{3.45}$$

$$\mathbf{F}_p = -\mathbf{K}_{pu}\mathbf{g} \tag{3.46}$$

**At the Element Level** The following elemental contributions result from Equations 3.40-3.46. For the displacement part, Equations 2.110-2.113 still hold. For the pressure equations' numbering, $1 \leq \widetilde{a}, \widetilde{b} \leq \widetilde{n}_{en}$ ($\widetilde{n}_{en}$ is the number of pressure nodes of the considered element). In the following, the symbol $\underset{a=1}{\overset{n_{el}}{A}}$ denotes the assembly operation, and $n_{el}$ is the number of elements.

- $\mathbf{K}_{uu} = \underset{a=1}{\overset{n_{el}}{A}} \mathbf{k}^e_{uu}$, and $\mathbf{k}^e_{uu} = \left[k^e_{uu,pq}\right]$ with $k^e_{uu,pq} = \mathbf{e}_i^T \int_{\Omega^e} \mathbf{B}_a^{eT} \overline{\mathbf{D}} \mathbf{B}_b^e d\Omega \mathbf{e}_j$
- $\mathbf{K}_{up} = \underset{a=1}{\overset{n_{el}}{A}} \mathbf{k}^e_{up}$, and $\mathbf{k}^e_{up} = \left[k^e_{up,p\widetilde{b}}\right]$ with $k^e_{uu,p\widetilde{b}} = \mathbf{e}_i^T \int_{\Omega^e} \mathbf{B}_a^{eT} \mathbf{1} \widetilde{N}^e_{\widetilde{b}} d\Omega$
- $\mathbf{K}_{pu} = \underset{a=1}{\overset{n_{el}}{A}} \mathbf{k}^e_{pu}$, and $\mathbf{k}^e_{pu} = \left[k^e_{pu,\widetilde{a}q}\right]$ with $k^e_{pu,\widetilde{a}q} = \int_{\Omega^e} \widetilde{N}^e_{\widetilde{a}} \mathbf{1}^T \mathbf{B}_b^e d\Omega \mathbf{e}_j$

- $\mathbf{K}_{pp} = \overset{n_{el}}{\underset{a=1}{A}} \mathbf{k}^e_{pp}$, and $\mathbf{k}^e_{pp} = \left[ k^e_{pp,\tilde{a}\tilde{b}} \right]$ with $k^e_{pu,\tilde{a}\tilde{b}} = -\frac{1}{K} \int_{\Omega^e} \widetilde{N}^e_{\tilde{a}} \widetilde{N}^e_{\tilde{b}} d\Omega$

- $\mathbf{F}_u = \overset{n_{el}}{\underset{a=1}{A}} \mathbf{f}^e_u$, and $\mathbf{f}^e_u = \left[ f^e_{u,p} \right]$ with $f^e_{u,p} = \int_{\Omega^e} N^e_a f_i d\Omega + \int_{\Gamma^e_{h_i}} N^e_a h_i d\Omega - k^e_{uu,pq} g^e_q$, and $g^e_q = g_j \left( \mathbf{x}^e_b \right)$ if and only if $\mathbf{x}^e_b \in \mathbf{\Gamma}_g$

- $\mathbf{F}_p = \overset{n_{el}}{\underset{a=1}{A}} \mathbf{f}^e_p$, and $\mathbf{f}^e_p = \left[ f^e_{p,\tilde{a}} \right]$ with $f^e_{p,\tilde{a}} = -k^e_{pu,\tilde{a}q} g^e_q$

**Positive Definiteness of the "Stiffness" Matrix (when $K \neq \infty$)** Contrary to the standard displacement finite element formulation, where it can be proved that the left-hand side (or stiffness) matrix is positive-definite and symmetric, the left-hand side of the matrix form ($M$) (Equation 3.39) is symmetric, but not positive-definite. In fact, in the incompressible case, the value of $K$ tends to infinity. In the slightly compressible case, the $\mathbf{K}_{pp}$ term will be negative as:

$$\mathbf{K}_{pp} = -\frac{1}{K} \int_{\Omega} \widetilde{\mathbf{N}}^T \widetilde{\mathbf{N}} d\Omega \tag{3.47}$$

Multiplying the second line of the matrix equation 3.39 by $-1$, symmetry will be lost, but we will show that the new $\overline{\mathbf{K}}$ is positive-definite:

$$\overline{\mathbf{K}} = \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ -\mathbf{K}_{pu} & -\mathbf{K}_{pp} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ -\mathbf{K}^T_{up} & -\mathbf{K}_{pp} \end{bmatrix} \tag{3.48}$$

where $\mathbf{K}_{pu} = \mathbf{K}^T_{up}$ (see Equations 3.41-3.42).

Taking the practical slightly compressible case, we establish the positive-definiteness of $\overline{\mathbf{K}}$ in the following way. Suppose we define a vector of the form:

$$\overline{\mathbf{c}} = \begin{Bmatrix} \overline{\mathbf{d}} \\ \overline{\mathbf{p}} \end{Bmatrix} \tag{3.49}$$

Positive-definiteness requires that $\overline{\mathbf{c}}^T\overline{\mathbf{K}}\overline{\mathbf{c}} \geq 0$ for any $\overline{\mathbf{c}}$ and that $\overline{\mathbf{c}}^T\overline{\mathbf{K}}\overline{\mathbf{c}} = 0$ implies $\overline{\mathbf{c}} = 0$.

$$\begin{aligned}
\overline{\mathbf{c}}^T\overline{\mathbf{K}}\overline{\mathbf{c}} &= \left\{ \begin{array}{c} \overline{\mathbf{d}} \\ \overline{\mathbf{p}} \end{array} \right\}^T \left[ \begin{array}{cc} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ -\mathbf{K}_{up}^T & -\mathbf{K}_{pp} \end{array} \right] \left\{ \begin{array}{c} \overline{\mathbf{d}} \\ \overline{\mathbf{p}} \end{array} \right\} \\
&= \left\{ \begin{array}{cc} \overline{\mathbf{d}}^T\mathbf{K}_{uu} - \overline{\mathbf{p}}^T\mathbf{K}_{up}^T & \overline{\mathbf{d}}^T\mathbf{K}_{up} - \overline{\mathbf{p}}^T\mathbf{K}_{pp} \end{array} \right\} \left\{ \begin{array}{c} \overline{\mathbf{d}} \\ \overline{\mathbf{p}} \end{array} \right\} \\
&= \overline{\mathbf{d}}^T\mathbf{K}_{uu}\overline{\mathbf{d}} - \overline{\mathbf{p}}^T\mathbf{K}_{up}^T\overline{\mathbf{d}} + \overline{\mathbf{d}}^T\mathbf{K}_{up}\overline{\mathbf{p}} - \overline{\mathbf{p}}^T\mathbf{K}_{pp}\overline{\mathbf{p}} \\
&= \overline{\mathbf{d}}^T\mathbf{K}_{uu}\overline{\mathbf{d}} - \overline{\mathbf{p}}^T\mathbf{K}_{up}^T\overline{\mathbf{d}} + \left(\overline{\mathbf{d}}^T\mathbf{K}_{up}\overline{\mathbf{p}}\right)^T - \overline{\mathbf{p}}^T\mathbf{K}_{pp}\overline{\mathbf{p}} \\
&= \overline{\mathbf{d}}^T\mathbf{K}_{uu}\overline{\mathbf{d}} - \overline{\mathbf{p}}^T\mathbf{K}_{up}^T\overline{\mathbf{d}} + \overline{\mathbf{p}}^T\mathbf{K}_{up}^T\overline{\mathbf{d}} - \overline{\mathbf{p}}^T\mathbf{K}_{pp}\overline{\mathbf{p}} \\
&= \overline{\mathbf{d}}^T\mathbf{K}_{uu}\overline{\mathbf{d}} - \overline{\mathbf{p}}^T\mathbf{K}_{pp}\overline{\mathbf{p}}
\end{aligned} \tag{3.50}$$

In the preceding development, the fact that $\overline{\mathbf{d}}^T\mathbf{K}_{up}\overline{\mathbf{p}} = \left(\overline{\mathbf{d}}^T\mathbf{K}_{up}\overline{\mathbf{p}}\right)^T$ as $\overline{\mathbf{d}}^T\mathbf{K}_{up}\overline{\mathbf{p}}$ is a scalar quantity (both $\overline{\mathbf{d}}$ and $\overline{\mathbf{p}}$ are column-vectors) has been used.

If we make the further assumption that both $\mathbf{K}_{uu}$ and $-\mathbf{K}_{pp}$ are positive-definite matrices, we can conclude:

$$\begin{aligned}
\overline{\mathbf{c}}^T\overline{\mathbf{K}}\overline{\mathbf{c}} &= \overline{\mathbf{d}}^T\mathbf{K}_{uu}\overline{\mathbf{d}} - \overline{\mathbf{p}}^T\mathbf{K}_{pp}\overline{\mathbf{p}} \geq 0 \\
\overline{\mathbf{c}}^T\overline{\mathbf{K}}\overline{\mathbf{c}} &= 0 \text{ implies } \overline{\mathbf{c}} = 0
\end{aligned} \tag{3.51}$$

but $\mathbf{K}_{uu}$ is positive-definite as it has been derived from the bilinear form $\overline{a}\left(\mathbf{w}^h, \mathbf{v}^h\right)$ and $-\mathbf{K}_{pp}$ is positive-definite if $K > 0$, which is always the case (as long as $K \neq \infty$). This concludes our proof and now we can write the following modified matrix form $(M^*)$, which will be the starting point for the stabilized formulation:

$$\left[ \begin{array}{cc} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ -\mathbf{K}_{up}^T & -\mathbf{K}_{pp} \end{array} \right] \left\{ \begin{array}{c} \mathbf{d} \\ \mathbf{p} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{F}_u \\ -\mathbf{F}_p \end{array} \right\} \tag{3.52}$$

### 3.1.3 Galerkin Least-Squares (GLS) Method

#### Introduction

It has been known for some time now that a number of mixed finite elements will fail to converge depending on the respective interpolations of the displacement and the pressure field, as indicated by the Babuśka & Brezzi stability condition. For instance, a typical four nodes quad with bilinear interpolation for both pressures and displacements will fail to converge and exhibit a tendency to mesh locking in the incompressible case, as well as an oscillatory pressure field. Therefore,

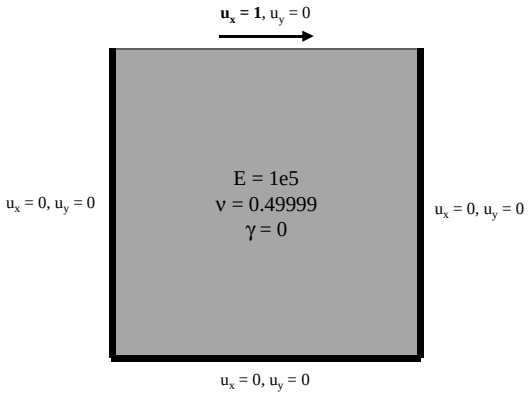



Figure 3-1: Cavity flow problem: boundary conditions

two improvement strategies can be distinguished:

- use higher-order finite elements, with a different interpolation for displacements and pressures
- use low-order finite elements (possibly with the same interpolation), but a stabilization technique is necessary in order to obtain convergence

### The Matrix Form Revisited

**Motivation** Applying the mixed formulation described until now (Equation 3.52) on an "incompressible" elasticity problem ($K = 1.67 \cdot 10^9\ KN/m^2$) inspired by the Stokes driven cavity flow (see Figure 3-1), we make the observation that the pressure field exhibits strong oscillations (see Figure 3-2). We have used here a $30 \times 30$ bilinear quads mesh. We have already seen that the $\overline{\mathbf{B}}$ approach with a displacement formulation helps to prevent both volumetric locking and spurious oscillations in the pressure field in the case of Stokes flow and incompressible elasticity, but we aim here at a more general methodology which would also allow us to mix different types of low-order elements (like quads and triangles) in the same mesh, as well as avoid locking in the case of dilatant plasticity.

The stabilized approach consists in adding terms to Equation 3.52 that will improve upon the stability of the standard Galerkin formulation without compromising its consistency.

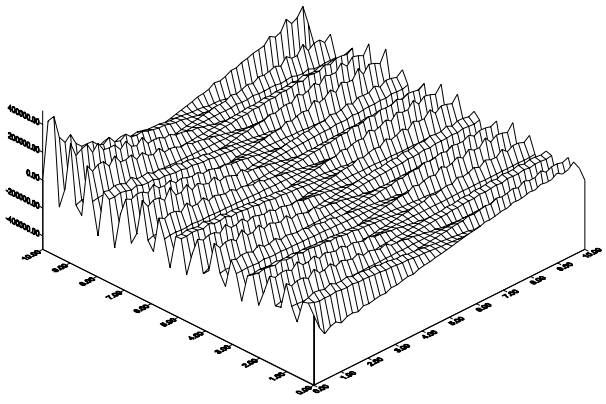


Figure 3-2: Pressure elevation, mixed formulation, no stabilization

**Stabilized Approach**

Following the methodology described in section 1.5 in the context of one-dimensional advection-diffusion, we will add some terms to the Galerkin form $(G)$ (Equations 3.27 and 3.28) in order to enhance its stability. In our case, by analogy with the GLS method, these terms take the form:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbb{L}\left(\mathbf{w}^h, q^h\right)^T \boldsymbol{\tau}\left(\mathcal{L}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right) d\Omega \tag{3.53}$$

or, in short notation, with $\Omega' = \bigcup \Omega^e$:

$$\left(\mathbb{L}\left(\mathbf{w}^h, q^h\right), \boldsymbol{\tau}\left(\mathcal{L}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right)\right)_{\Omega'} \tag{3.54}$$

In the preceding equation, $\left(\mathcal{L}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right)$ is an abstract notation for the left-hand side of the differential equilibrium equation (Equation 3.10), and $\mathbb{L}$ is a differential operator. For the Galerkin Least-Squares approach, $\mathbb{L} = +\mathcal{L}$.

How does this method retain consistency? The answer lies in the fact that as the added terms are based on a residual method, replacing $\mathbf{u}^h$ and $p^h$ by their exact counterparts $\mathbf{u}$ and $p$ in $\left(\mathcal{L}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right)$ would make these terms vanish identically.

Finally, if we introduce the following differential operator $\mathbf{L}$ (in the general three-dimensional

case, $n_{sd} = 3$):

$$\mathbf{L} = \begin{bmatrix} \dfrac{\partial}{\partial x_1} & 0 & 0 \\ 0 & \dfrac{\partial}{\partial x_2} & 0 \\ \dfrac{\partial}{\partial x_2} & \dfrac{\partial}{\partial x_1} & 0 \\ 0 & 0 & \dfrac{\partial}{\partial x_3} \\ \dfrac{\partial}{\partial x_3} & 0 & \dfrac{\partial}{\partial x_1} \\ 0 & \dfrac{\partial}{\partial x_3} & \dfrac{\partial}{\partial x_2} \end{bmatrix} \tag{3.55}$$

we can rewrite Equation 3.54 as:

$$\left(\mathbf{L}^T\boldsymbol{\sigma}\left(\mathbf{w}^h, q^h\right), \boldsymbol{\tau}\left(\mathbf{L}^T\boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right)\right)_{\Omega'} \tag{3.56}$$

**Remark 16** *the added Least-Squares terms are based on the equilibrium equation only, not on the second equation (constitutive equation expressing continuity). However the volumetric-deviatoric split of the stress tensor (see Equation 3.1) introduces naturally a coupling between the displacement and the pressure terms, i.e. stabilizing contributions will be present in both the displacement and the pressure equations (Equation 3.52).*

**Remark 17** *the vector notation introduces a matrix of stabilization factors* $\boldsymbol{\tau}$*. This matrix takes the form:*

$$\boldsymbol{\tau} = \tau\mathbf{I} \tag{3.57}$$

*where* $\mathbf{I}$ *is the* $n_{sd} \times n_{sd}$ *identity matrix. We can therefore speak of a scalar stabilization factor* $\tau$*, which definition is at the heart of the method.*

### Back to the Matrix Form

If we express stabilization terms of Equation 3.56 in vector notation with respect to the basis that we chose earlier in order to derive the matrix form, we come up with the following expression:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \left(\mathbf{L}^T\overline{\mathbf{D}}\mathbf{B}\mathbf{c} + \mathbf{L}^T\mathbf{1}\widetilde{\mathbf{N}}\mathbf{q}\right)^T \boldsymbol{\tau}\left(\mathbf{L}^T\overline{\mathbf{D}}\mathbf{B}\mathbf{d} + \mathbf{L}^T\mathbf{1}\widetilde{\mathbf{N}}\mathbf{p} + \mathbf{f}\right) d\Omega \tag{3.58}$$

Or, in shorter form:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \left(\mathbf{G}_u\mathbf{c} + \mathbf{G}_p\mathbf{q}\right)^T \boldsymbol{\tau}\left(\mathbf{G}_u\mathbf{d} + \mathbf{G}_p\mathbf{p} + \mathbf{f}\right) d\Omega \tag{3.59}$$

Adding these terms to Equation 3.52, and argueing that the added terms must hold for any $\mathbf{c}$ and $\mathbf{q}$ (i.e. $\mathbf{w}^h$ and $q^h$), we come up with the stabilized matrix form $(M')$ which will have the form:

$$\begin{bmatrix} \mathbf{K}_{uu} + \mathbf{K}'_{uu} & \mathbf{K}_{up} + \mathbf{K}'_{up} \\ -\mathbf{K}_{up}^T + \mathbf{K}'_{pu} & -\mathbf{K}_{pp} + \mathbf{K}'_{pp} \end{bmatrix} \begin{Bmatrix} \mathbf{d} \\ \mathbf{p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_u + \mathbf{F}'_u \\ -\mathbf{F}_p + \mathbf{F}'_p \end{Bmatrix} \tag{3.60}$$

where definitions 3.40-3.46 still hold, and:

$$\mathbf{K}'_{uu} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_u^T \boldsymbol{\tau} \mathbf{G}_u d\Omega \tag{3.61}$$

$$\mathbf{K}'_{up} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_u^T \boldsymbol{\tau} \mathbf{G}_p d\Omega \tag{3.62}$$

$$\mathbf{K}'_{pu} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_p^T \boldsymbol{\tau} \mathbf{G}_u d\Omega \tag{3.63}$$

$$\mathbf{K}'_{pp} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_p^T \boldsymbol{\tau} \mathbf{G}_p d\Omega \tag{3.64}$$

$$\mathbf{F}'_u = -\sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_u^T \boldsymbol{\tau} \mathbf{f} d\Omega \tag{3.65}$$

$$\mathbf{F}'_p = -\sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_p^T \boldsymbol{\tau} \mathbf{f} d\Omega \tag{3.66}$$

**Computation of $\mathbf{G}_u$**

Contrary to the standard finite element formulation, where the order of differentiation of the interpolation functions is limited to one, the introduction of the matrix term $\mathbf{G}_u$:

$$\mathbf{G}_u = \mathbf{L}^T \overline{\mathbf{D}} \mathbf{B} \tag{3.67}$$

involves second-order derivatives of the shape functions $N_a$. If we explicit Equation 3.67, we get:

$$\mathbf{G}_u = \begin{bmatrix} \mathbf{G}_u^1 & \dots & \mathbf{G}_u^{n_{en}} \end{bmatrix} \tag{3.68}$$

where:

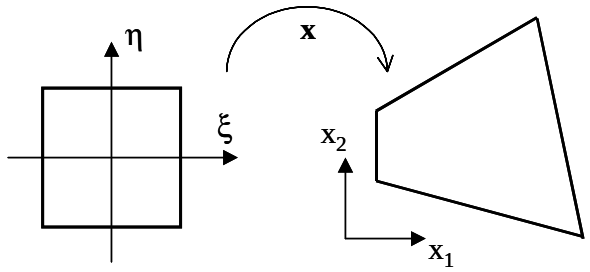



Figure 3-3: Parent domain (left) and global coordinates (right)

$$\mathbf{G}_u^a = \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_3} & 0 \\ 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} & 0 & 0 & \frac{\partial}{\partial x_3} \\ 0 & 0 & 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} \end{bmatrix} \overline{\mathbf{D}}_{6\times 6} \begin{bmatrix} \frac{\partial N_a}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial N_a}{\partial x_2} & 0 \\ \frac{\partial N_a}{\partial x_2} & \frac{\partial N_a}{\partial x_1} & 0 \\ 0 & 0 & \frac{\partial N_a}{\partial x_3} \\ \frac{\partial N_a}{\partial x_3} & 0 & \frac{\partial N_a}{\partial x_1} \\ 0 & \frac{\partial N_a}{\partial x_3} & \frac{\partial N_a}{\partial x_2} \end{bmatrix} \tag{3.69}$$

and we see that terms of the form $\frac{\partial^2 N_a}{\partial x_i \partial x_j}$ will appear. Introducing the parent domain $\square$ (see Figure 3-3) with its natural coordinates system $(\xi;\eta)$, we begin with the chain rule:

$$\frac{\partial^2 N_a}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_j}\left[\frac{\partial N_a}{\partial \xi_k}\frac{\partial \xi_k}{\partial x_i}\right] \tag{3.70}$$

and expanding the derivative of a product:

$$\frac{\partial}{\partial x_j}\left[\frac{\partial N_a}{\partial \xi_k}\frac{\partial \xi_k}{\partial x_i}\right] = \frac{\partial}{\partial x_j}\left[\frac{\partial N_a}{\partial \xi_k}\right]\cdot\frac{\partial \xi_k}{\partial x_i} + \frac{\partial N_a}{\partial \xi_k}\cdot\frac{\partial}{\partial x_j}\left[\frac{\partial \xi_k}{\partial x_i}\right] \tag{3.71}$$

The first term in the right-hand side of Equation 3.71 can be expanded in turn as:

$$\frac{\partial}{\partial x_j}\left[\frac{\partial N_a}{\partial \xi_k}\right] \cdot \frac{\partial \xi_k}{\partial x_i} = \frac{\partial^2 N_a}{\partial \xi_k \partial \xi_l}\frac{\partial \xi_l}{\partial x_j}\frac{\partial \xi_k}{\partial x_i} \tag{3.72}$$

and every term can be computed explicitely as the shape functions $N_a$ are directly function of $(\xi, \eta)$ and if we recall that $\frac{\partial \xi_k}{\partial x_i} = \xi_{k,i}$ can be computed (in three dimensions) via:

$$\boldsymbol{\xi}_{,\mathbf{x}} = \mathbf{x}_{,\boldsymbol{\xi}}^{-1} = \begin{bmatrix} x_{,\xi} & x_{,\eta} & x_{,\zeta} \\ y_{,\xi} & y_{,\eta} & y_{,\zeta} \\ z_{,\xi} & z_{,\eta} & z_{,\zeta} \end{bmatrix}^{-1} \tag{3.73}$$

The second term in the right-hand side of Equation 3.71 can itself be expanded as:

$$\frac{\partial N_a}{\partial \xi_k} \cdot \frac{\partial}{\partial x_j}\left[\frac{\partial \xi_k}{\partial x_i}\right] = \frac{\partial N_a}{\partial \xi_k}\frac{\partial^2 \xi_k}{\partial x_i \partial x_j} \tag{3.74}$$

and the only difficult term to compute reduces to $\frac{\partial^2 \xi_k}{\partial x_i \partial x_j}$. Starting with the derivative of a Kronecker delta $\delta_{pq}$, we can write:

$$\frac{\partial}{\partial x_j}(\delta_{pq}) = \frac{\partial}{\partial x_j}\left(\xi_{p,k}^{-1}\xi_{k,q}\right) = 0 \tag{3.75}$$

Expanding the derivative of the product:

$$-\frac{\partial \xi_{p,k}^{-1}}{\partial x_j}\xi_{k,q} = \xi_{p,k}^{-1}\frac{\partial \xi_{k,q}}{\partial x_j} \tag{3.76}$$

and multiplying Equation 3.76 by $\xi_{r,p}$ on the left leads to:

$$-\xi_{r,p}\frac{\partial \xi_{p,k}^{-1}}{\partial x_j}\xi_{k,q} = \xi_{r,p}\xi_{p,k}^{-1}\frac{\partial \xi_{k,q}}{\partial x_j} = \delta_{rk}\frac{\partial \xi_{k,q}}{\partial x_j} \tag{3.77}$$

Replacing index $q$ by $i$ and setting $r = k$ (so that $\delta_{rk} = 1$):

$$\frac{\partial \xi_{k,i}}{\partial x_j} = -\xi_{k,p}\frac{\partial \xi_{p,k}^{-1}}{\partial x_j}\xi_{k,i} \tag{3.78}$$

and using the fact that:

$$\frac{\partial \xi_{p,k}^{-1}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\frac{\partial x_p}{\partial \xi_k}\right] = \frac{\partial^2 x_p}{\partial \xi_k \partial \xi_m}\frac{\partial \xi_m}{\partial x_j} \tag{3.79}$$

Introducing Equation 3.79 into 3.78 and then back into Equations 3.74 and 3.70 finally yields:

$$\frac{\partial^2 N_a}{\partial x_i \partial x_j} = \frac{\partial^2 N_a}{\partial \xi_k \partial \xi_l}\frac{\partial \xi_l}{\partial x_j}\frac{\partial \xi_k}{\partial x_i} - \underbrace{\frac{\partial N_a}{\partial \xi_k}\frac{\partial \xi_k}{\partial x_p}}_{\frac{\partial N_a}{\partial x_p}}\frac{\partial^2 x_p}{\partial \xi_k \partial \xi_m}\frac{\partial \xi_m}{\partial x_j}\frac{\partial \xi_k}{\partial x_i} \tag{3.80}$$

The method computing these shape-functions second-order derivatives for the bilinear quad is illustrated in figure 3-4.

### Definition of the Stabilization Factor $\tau$

**Basic Scalar Form** We have already seen how to derive this stabilization factor in the case of the one-dimensional advection-diffusion equation (corresponding to the intrinsic time scale), but what can we say about this factor in the case we are interested in right now?

First, a dimensional analysis can give us hints on the nature of $\tau$. In the derivation of the standard Galerkin weak form (Equation 2.86), the equilibrium equation is premultiplied by $w_i$, a weighting function which has the nature of a virtual displacement, with units $[m]$:

$$\int_\Omega w_i \left(\sigma_{ij,j} + f_i\right) d\Omega = 0 \tag{3.81}$$

As we recall (Equation 3.53), stabilizing terms take the form:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \left(\mathbb{L}\left(\mathbf{w}^h, q^h\right)\right)^T \boldsymbol{\tau}\left(\mathcal{L}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right) d\Omega \tag{3.82}$$

And as we can identify $\left(\mathcal{L}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right) = \left(\sigma_{ij,j} + f_i\right)$, we can conclude that the dimensions of $w_i$ should match the dimension of $\mathbb{L}\left(\mathbf{w}^h, q^h\right) \cdot \tau \mathbf{I}$. For the Galerkin Least-Squares approach, $\mathbb{L} = +\mathcal{L}$, which implies that $\mathbb{L}\left(\mathbf{w}^h, q^h\right)$ has the dimensions of $\sigma_{ij,j}$ or $f_i$, i.e. $[\frac{kN}{m^3}]$. Therefore:

$$[m] = [\frac{kN}{m^3}] \cdot [\frac{m^4}{kN}] = [\frac{kN}{m^3}] \cdot [\frac{m^2}{\frac{kN}{m^2}}] \tag{3.83}$$

```
FloatMatrix*  Quad_UP_Stab :: giveD2NDX2 (int a, GaussPoint* gp)
{
               FloatArray* Coord = gp -> giveCoordinates() ;
               FloatMatrix* jacGP = this -> giveJacobianMatrix() -> EvaluatedAt(Coord) ;
               FloatMatrix* inv   = jacGP -> GiveInverse();
               FloatMatrix* answer = new FloatMatrix(2,2);
               FloatMatrix* D2NDKsi2 = this->giveD2NDKsi2(a);
               FloatArray* DNDX = this->giveDNDX(a, gp);

               for (int i=1; i<=2; i++) {
                  for (int j=1; j<=2; j++) {
                     double result = 0;
                     for (int l=1; l<=2; l++) {
                        FloatMatrix* D2XDKsi2 = this->giveD2XDKsi2(l);
                        for (int k=1; k<=2; k++) {
                           result += inv->at(l,j)*inv->at(k,i)*D2NDKsi2->at(k,l);
                           for (int m=1; m<=2; m++) {
                              result -= DNDX->at(l)*inv->at(m,j)*inv->at(k,i)*D2XDKsi2->at(k,m);
                           }
                        }
                        delete D2XDKsi2;
                     }
                     answer->at(i,j) = result;
                  }
               }

               delete jacGP; delete inv; delete D2NDKsi2; delete DNDX;
               return answer;
}
```

Figure 3-4: Method `Quad_UP_Stab::giveD2NDX2()`

And we can conclude:

$$\text{Units}(\tau) = [\frac{m^2}{\frac{kN}{m^2}}] \tag{3.84}$$

Similarly to what has been done in the fluid mechanics field, argueing that the equation that govern Stokes flow are the same that govern incompressible elasticity, we define $\tau$ as being (see [27]):

$$\tau = \frac{\alpha^e (h^e)^2}{2\mu} \tag{3.85}$$

$\alpha^e$ is a dimensionless scalar factor which has to be calibrated. $h^e$ is a characteristic length of the element - in our case the square root of the element surface - and $\mu$ is the element's material shear modulus.

**Taking Into Account Element Distorsion** The scalar form of the stabilization factor $\tau$ that has been derived in the preceding paragraph defines the element size as being the square root of the element area. Speaking of quadrilateral elements, this would mean that a 10 $m \times 10$ $m$ quad would have the same $\tau$ that a 25 $m \times 4$ $m$ rectangle. Another form of $\tau$ lets it be function of the covariant metric tensor $\mathbf{g}$, taking into account the element distorsion [53]:

$$\tau = C\frac{1}{\mu}\frac{1}{\sqrt{g_{ij}g_{ij}}} \tag{3.86}$$

where the covariant metric tensor $g_{ij}$ is defined by:

$$g_{ij} = \frac{\partial \xi_\alpha}{\partial x_i}\frac{\partial \xi_\alpha}{\partial x_j} \text{ (sum on } \alpha) \tag{3.87}$$

$C$ is a constant that can be determined by identifying Equations 3.85 and 3.86. For instance, in the case of quadrilateral elements, taking a square with a length of $h^e$ leads us to:

$$C = 2\sqrt{2}\alpha^e \tag{3.88}$$

as:

$$\frac{\partial \xi_\alpha}{\partial x_i} = \frac{2}{h^e} \tag{3.89}$$

Numerical experiments have shown no significant improvement so far when using this type of stabilization factor when we compare the results of this formulation with the basic scalar scheme.

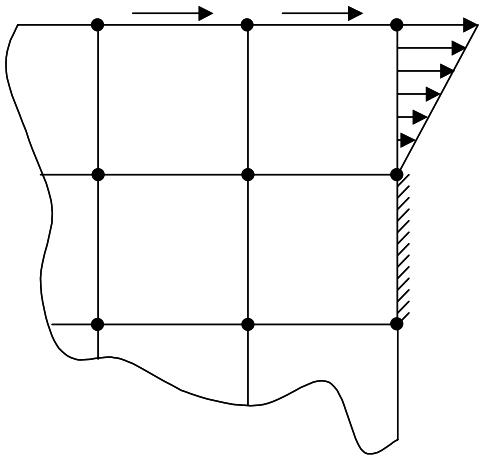


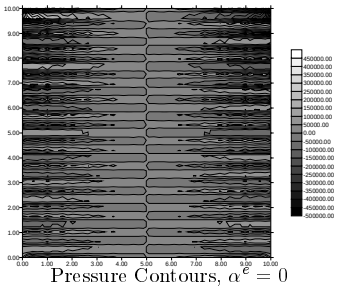
Figure 3-5: Corner boundary

### 3.1.4 Example: Cavity Flow

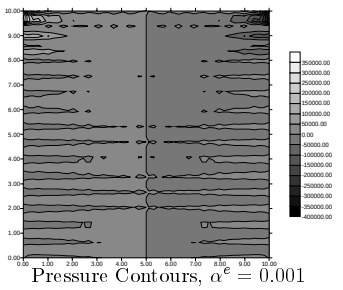
**Influence of $\alpha^e$**

The performance of the formulation is now tested on the cavity flow problem. Figure 3-1 illustrates the geometry, material properties and boundary conditions of the problem. Figure 3-5 defines the top corners boundary conditions.

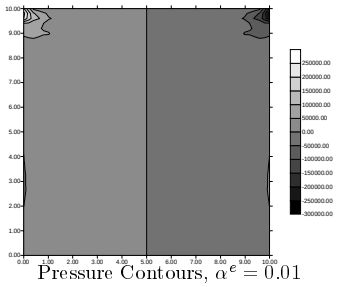
The element that we test here is the 4-node quad, with both the pressure and the displacement fields approximated with the usual bilinear shape functions. A $30 \times 30$ mesh is used and we compare here the solution for different values of the parameter $\alpha^e$. In this example, all the elements are square, which means that there is no difference between the two types of stabilization factors described before. Pressure contours are illustrated next.



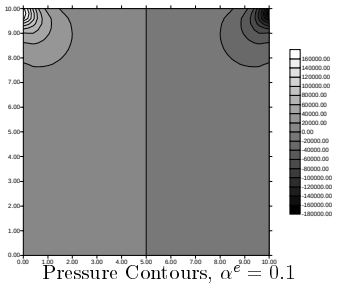
Pressure Contours, $\alpha^e = 0$



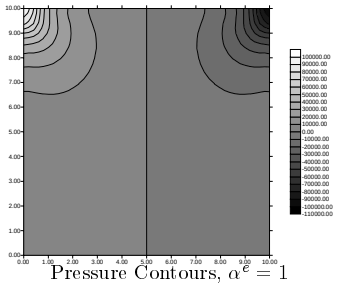
Pressure Contours, $\alpha^e = 0.001$



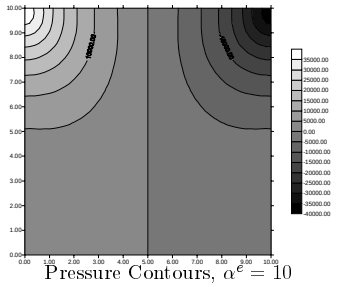
Pressure Contours, $\alpha^e = 0.01$



Pressure Contours, $\alpha^e = 0.1$



Pressure Contours, $\alpha^e = 1$



Pressure Contours, $\alpha^e = 10$

If we examine these results, we can conclude that the pressure field is very sensitive to the

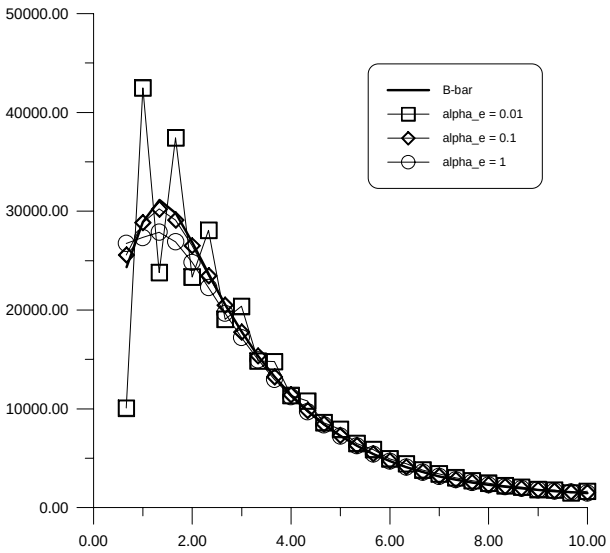


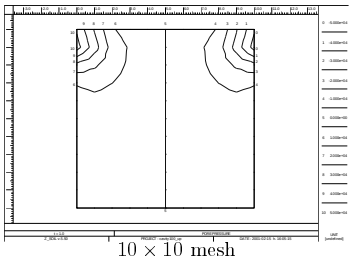

Figure 3-6: Comparison of Horizontal Stresses

$\alpha^e$ parameter. It seems that a value located between 0.1 and 1 gives the best results. $\alpha^e < 0.1$ induces oscillations in the pressure field (the stabilization is not strong enough), while if $\alpha^e$ is too large, the stabilization will be too strong and the pressure field will become too smooth, missing for instance the large values it should have at the corners. We will now look at the stress distribution along a vertical line ($x = 1$) and compare the solutions with the reference solution given by the $\overline{\mathbf{B}}$-method. Looking at Figure 3-6, $\alpha^e = 0.1$ is the parameter which matches the $\overline{\mathbf{B}}$-method at best.

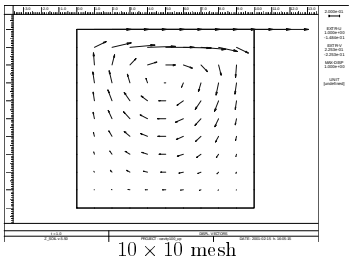
### $h$-Convergence

The stabilization parameter value $\alpha^e = 0.1$ is now applied to three different Q4 meshes ($10 \times 10$, $30 \times 30$ and $50 \times 50$) in order to show that the solution is mesh-independent. The pressure and the displacement fields are reproduced next for these three cases. We conclude that both fields are approximated in a correct way for the three different discretizations.

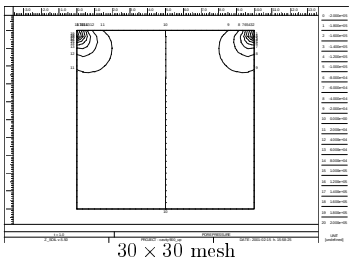
**Pressure field** **Displacement field**



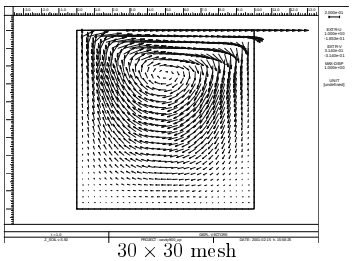
$10 \times 10$ mesh



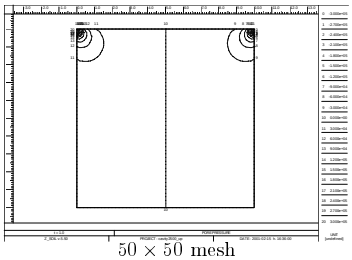
$10 \times 10$ mesh



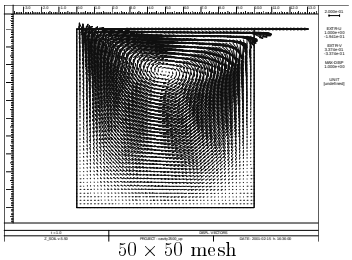
$30 \times 30$ mesh



$30 \times 30$ mesh



$50 \times 50$ mesh



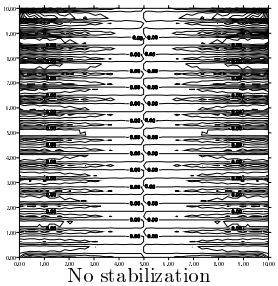
$50 \times 50$ mesh

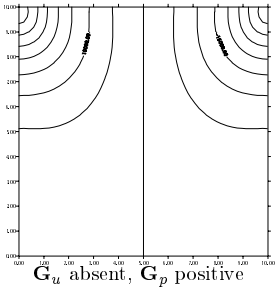
**Influence of $\mathbf{G}_u$**

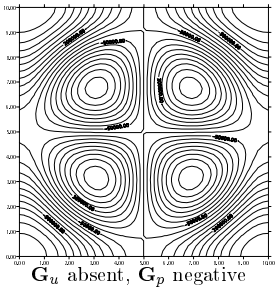
**Numerical Considerations** We have seen in a previous section that the computation of the $\mathbf{G}_u$ term in the weighting part of the stabilizing term can be tedious. In fact, we will even see later in the section dedicated to the plastic case that it can even be impossible to know what this matrix exactly is due to linearization problems. Considering this we will make the same cavity flow test in six different cases, depending on the presence or absence of $\mathbf{G}_u$ and $\mathbf{G}_p$ in the

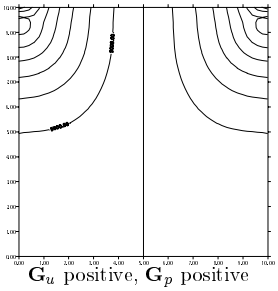
weighting part of stabilizing terms (see Equation 3.59):

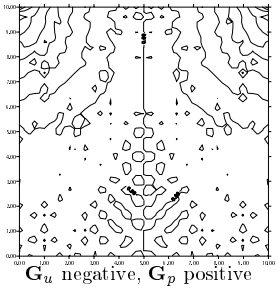
$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \left(\pm \mathbf{G}_u \mathbf{c} \pm \mathbf{G}_p \mathbf{q}\right)^T \boldsymbol{\tau} \left(\mathbf{G}_u \mathbf{d} + \mathbf{G}_p \mathbf{p} + \mathbf{f}\right) d\Omega \tag{3.90}$$

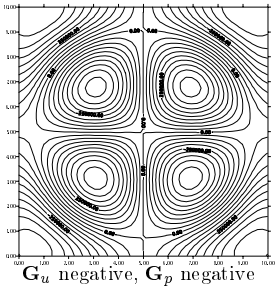

No stabilization


$\mathbf{G}_u$ absent, $\mathbf{G}_p$ positive


$\mathbf{G}_u$ absent, $\mathbf{G}_p$ negative


$\mathbf{G}_u$ positive, $\mathbf{G}_p$ positive


$\mathbf{G}_u$ negative, $\mathbf{G}_p$ positive


$\mathbf{G}_u$ negative, $\mathbf{G}_p$ negative

It follows from the observation of the results that the two only meaningful results are obtained in case 2 and case 4. Between these two cases, the addition of the $\mathbf{G}_u$ term doesn't seem to have a very big influence on the solution (and if it has, it is more a destabilizing effect than anything else, looking at the corners).

**Remark 18** *here $\alpha^e = 10$ in order to accentuate differences between the formulations.*

**Formal Justification** The absence of the $\mathbf{G}_u$ term in the weighting part can be justified in the following sense. Consider the weak form of the equilibrium equation in the mixed formulation without stabilization:

$$\int_\Omega w_{(i,j)}\sigma_{ij}(\mathbf{u},p)d\Omega = \int_{\Gamma_h} w_i h_i d\Gamma + \int_\Omega w_i f_i d\Omega \tag{3.91}$$

Integrating by parts the first term:

$$-\int_\Omega w_i\sigma_{ij,j}d\Omega + \int_{\Gamma_h} w_i\sigma_{ij}n_j d\Gamma = \int_{\Gamma_h} w_i h_i d\Gamma + \int_\Omega w_i f_i d\Omega \tag{3.92}$$

If $w_i$, $u_i$ and $p$ are $C^\infty$ all over the domain $\Omega$, Equation 3.92 can be rewritten:

$$\int_\Omega w_i\left(\sigma_{ij,j} + f_i\right)d\Omega - \int_{\Gamma_h} w_i\left(\sigma_{ij}n_j - h_i\right)d\Gamma = 0 \tag{3.93}$$

But if $w_i$, $u_i$ and $p$ are $C^0$ globally, but may suffer slope discontinuities at element boundaries, we have to write:

$$\sum_{e=1}^{n_{el}}\int_{\Omega^e} w_i\left(\sigma_{ij,j} + f_i\right)d\Omega - \int_{\Gamma_h} w_i\left(\sigma_{ij}n_j - h_i\right)d\Gamma - \int_{\Gamma_{int}} w_i\left\langle\sigma_{ij}n_j\right\rangle d\Gamma = 0 \tag{3.94}$$

In the preceding equation, $\langle\cdot\rangle$ denotes the jump across element boundaries, and $\Gamma_{int} = \sum_{e=1}^{n_{el}}\Gamma_e - \Gamma$. Equation 3.94 yields the following **Euler-Lagrange equations**:

- $\sigma_{ij,j} + f_i = 0$ inside every element $\Omega^e$
- $\sigma_{ij}n_j - h_i = 0$ on the $\Gamma_h$ boundary
- $\langle\sigma_{ij}n_j\rangle = 0$: continuity condition of the stress across element boundaries

If we can show that our stabilized formulations yield the same Euler-Lagrange equations, then we will know that we are solving the correct problem. Stabilization is obtained by adding

terms of the form (see Equation 3.53):

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbb{L}\left(\mathbf{w}, q\right)^T \boldsymbol{\tau}\left(\mathcal{L}\left(\mathbf{u}, p\right) + \mathbf{f}\right) d\Omega \tag{3.95}$$

and we know that the residual part of these terms corresponds to:

$$\mathcal{L}\left(\mathbf{u}, p\right) + \mathbf{f} = \sigma_{ij,j} + f_i \tag{3.96}$$

Therefore we can regroup these terms with the first term in Equation 3.94, whatever $\mathbb{L}$ is! We will get:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \widetilde{w_i}\left(\sigma_{ij,j} + f_i\right) d\Omega - \int_{\Gamma_h} w_i\left(\sigma_{ij} n_j - h_i\right) d\Gamma - \int_{\Gamma_{int}} w_i \left\langle \sigma_{ij} n_j \right\rangle d\Gamma = 0 \tag{3.97}$$

and the only difference with Equation 3.94 lies in the definition of the weighting function of the first term:

- $\mathbf{G}_u$ present: $\widetilde{w_i} = w_i + \tau \mathcal{L}_i\left(\mathbf{w}, q\right)$
- $\mathbf{G}_u$ absent: $\widetilde{w_i} = w_i + \tau \mathbb{L}_i\left(q\right) = w_i + \tau \overrightarrow{\bigtriangledown} q$

In both cases Euler-Lagrange equations correspond to the ones defined earlier.

In order to be exhaustive, the continuity equation should also be mentioned. Its weak from reads:

$$\int_{\Omega} q \left(u_{i,i} - \frac{p}{K}\right) d\Omega = 0 \tag{3.98}$$

Summing over elements:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} q \left(u_{i,i} - \frac{p}{K}\right) d\Omega = 0 \tag{3.99}$$

and it can be readily seen that the Euler-Lagrange equation corresponding to Equation 3.99 is:

- $u_{i,i} - \dfrac{p}{K} = 0$ inside every element $\Omega^e$

### Mixing Elements in the Same Mesh

Considering the same cavity flow problem (with $\alpha^e = 0.1$), the stabilized solution obtained when mixing bilinear quads and linear triangles is shown not to suffer from highly oscillatory patterns, such as the mixture of $\overline{\mathbf{B}}$ quadrilaterals and triangles (see Figures 3-7 - 3-9).

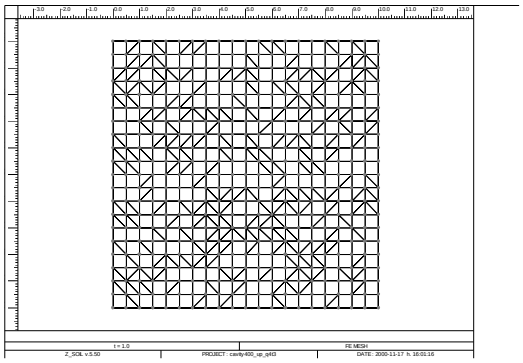


Figure 3-7: Cavity flow: Q4 + T3 mesh

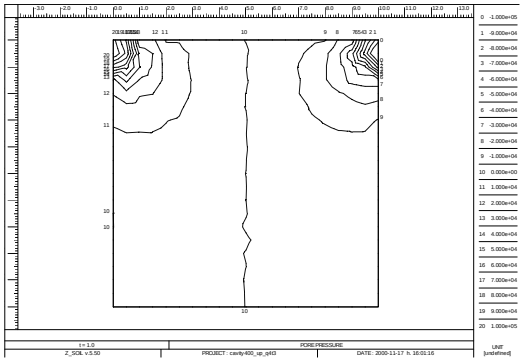


Figure 3-8: Stabilized Q4 and T3 pressure field

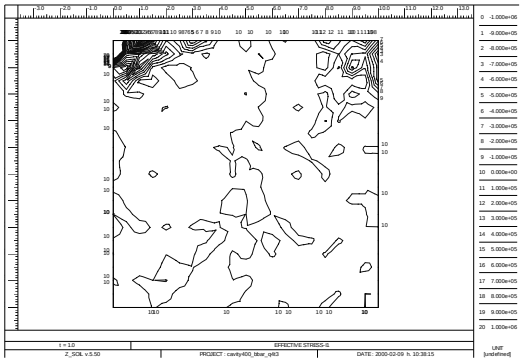


Figure 3-9: Q4 ($\overline{\mathbf{B}}$) and T3 pressure field

## 3.2 Extension to Plasticity

### 3.2.1 Preliminaries

Following the guidelines established in the elastic case, we can now apply the mixed approach to plasticity. The same hypotheses still hold, as well as the vector notations defined before. As an additional assumption, the material will be considered to be perfectly elasto-plastic (**no hardening**). As we deal with plasticity, which means that the stress-strain state of the material is history- or path-dependent, we can't use the constitutive equation in the integral form as we did when we dealt with linear elasticity. Instead, we have to express it in the rate or incremental form. The incremental constitutive equation then reads:

$$\boldsymbol{\Delta\sigma} = \overline{\mathbf{D}}\left[\boldsymbol{\Delta\varepsilon}\left(\mathbf{u}\right) - \boldsymbol{\Delta\varepsilon}^{p}\left(\mathbf{u},p\right)\right] + \mathbf{1}\Delta p \tag{3.100}$$

In the preceding equation, the **additive decomposition of the strain field** into an elastic and a plastic contribution has been used:

$$\boldsymbol{\Delta\varepsilon}\left(\mathbf{u}\right) = \boldsymbol{\Delta\varepsilon}^{e}\left(\mathbf{u}\right) + \boldsymbol{\Delta\varepsilon}^{p}\left(\mathbf{u},p\right) \tag{3.101}$$

As a recall, additional ingredients of plasticity needed here are: the definition of a **yield function**:

$$f\left(\boldsymbol{\sigma}\right) = 0 \tag{3.102}$$

the introduction of a **flow rule**:

$$\boldsymbol{\Delta\varepsilon}^p(\mathbf{u},p) = \Delta\gamma\frac{\partial g}{\partial\boldsymbol{\sigma}} = \Delta\gamma\ \mathbf{r} \tag{3.103}$$

where $\Delta\gamma$ is the scalar plastic multiplier and $g$ is a function called the plastic potential, and finally **consistency** implying:

$$\Delta\gamma\ \dot{f}(\boldsymbol{\sigma}) = 0 \tag{3.104}$$

### 3.2.2 The Mixed Plastic Boundary Value Problem

#### Strong Form

Given as before $f_i : \Omega \longrightarrow \mathbb{R}$, $g_i : \Gamma_{g_i} \longrightarrow \mathbb{R}$ and $h_i : \Gamma_{h_i} \longrightarrow \mathbb{R}$, we wish to find $u_i : \overline{\Omega} \longrightarrow \mathbb{R}$ and $p : \overline{\Omega} \longrightarrow \mathbb{R}$ such that:

$$\begin{aligned}
\sigma_{ij,j} + f_i &= 0 && \text{in } \Omega && (3.105)\\
u^e_{i,i} - \frac{p}{K} &= 0 && \text{in } \Omega && (3.106)\\
u_i &= g_i && \text{on } \Gamma_{g_i} && (3.107)\\
\sigma_{ij}n_j &= h_i && \text{on } \Gamma_{h_i} && (3.108)
\end{aligned}$$

where $\sigma_{ij}(\mathbf{u},p)$ is computed incrementally with the help of Equation 3.100.

#### Weak Form

Premultiplying Equation 3.105 by a weighting function $w_i \in \mathcal{V}_i$ and integrating over the domain:

$$\int_\Omega w_i\left(\sigma_{ij,j} + f_i\right)d\Omega = 0 \tag{3.109}$$

Integrating the first expression by parts, using the symmetry of $\sigma_{ij}$, Equation 3.108 and the fact that $w_i = 0$ on $\Gamma_{g_i}$ leads to:

$$\int_\Omega w_{(i,j)}\sigma_{ij}(\mathbf{u},p)\,d\Omega = \sum_{i=1}^{n_{sd}}\int_{\Gamma_{h_i}} w_i h_i d\Gamma + \int_\Omega w_i f_i d\Omega \tag{3.110}$$

On the other hand, we premultiply Equation 3.106 by a weighting function $q \in \mathcal{P}$ and integrate over the domain:

$$\int_\Omega q\left(u^e_{i,i} - \frac{p}{K}\right)d\Omega = 0 \tag{3.111}$$

And the weak form can be expressed as: given $f_i$, $g_i$ and $h_i$ as before, we wish to find $u_i \in \mathcal{S}_i$ and $p \in \mathcal{P}$ such that, for all $w_i \in \mathcal{V}_i$ and $q \in \mathcal{P}$, Equations 3.110 and 3.111 hold.

### Galerkin Form

Introducing the finite-dimensional subspaces $\mathcal{S}_i^h \subset \mathcal{S}_i$, $\mathcal{V}_i^h \subset \mathcal{V}_i$ and $\mathcal{P}^h \subset \mathcal{P}$, we can express the Galerkin form of the problem as: given $f_i$, $g_i$ and $h_i$ as before, we wish to find $u_i^h = (v_i^h + g_i^h) \in \mathcal{S}_i^h$ (with $v_i^h \in \mathcal{V}_i^h$) and $p^h \in \mathcal{P}^h$ such that, for all $w_i^h \in \mathcal{V}_i^h$ and $q^h \in \mathcal{P}^h$, the following scalar equations hold:

$$\int_\Omega w^h_{(i,j)} \sigma_{ij} \left( \mathbf{u}^h, p^h \right) d\Omega = \sum_{i=1}^{n_{sd}} \int_{\Gamma_{h_i}} w_i^h h_i d\Gamma + \int_\Omega w_i^h f_i d\Omega \tag{3.112}$$

$$\int_\Omega q \left( u^{e,h}_{i,i} - \frac{p^h}{K} \right) d\Omega = 0 \tag{3.113}$$

Introducing the vector notation, we can rewrite Equations 3.112 and 3.113 as:

$$\int_\Omega \boldsymbol{\varepsilon} \left( \mathbf{w}^h \right)^T \boldsymbol{\sigma} \left( \mathbf{u}^h, p^h \right) d\Omega = \int_{\Gamma_h} \left( \mathbf{w}^h \right)^T \mathbf{h} d\Gamma + \int_\Omega \left( \mathbf{w}^h \right)^T \mathbf{f} d\Omega \tag{3.114}$$

$$\int_\Omega q^h \mathbf{1}^T \boldsymbol{\varepsilon}^e \left( \mathbf{u}^h \right) d\Omega - \int_\Omega q^h \frac{p^h}{K} d\Omega = 0 \tag{3.115}$$

### Matrix Form

We now specify the approximations of the displacement and pressure fields as in the elastic case. Introducing two sets of shape functions $N_A(\mathbf{x})$ and $\widetilde{N}_{\widetilde{A}}(\mathbf{x})$ into the Galerkin form, and argueing that the virtual displacement and pressure fields can be eliminated invoking arbitrariness of $c_{iA}$ and $q_{\widetilde{A}}$ (i.e. $\mathbf{w}^h$ and $q^h$), we can write the following set of equations:

$$\int_\Omega \mathbf{B}^T \boldsymbol{\sigma} \left( \mathbf{u}^h, p^h \right) d\Omega = \int_{\Gamma_h} \mathbf{N}^T \mathbf{h} d\Gamma + \int_\Omega \mathbf{N}^T \mathbf{f} d\Omega \tag{3.116}$$

$$\int_\Omega \widetilde{\mathbf{N}}^T \mathbf{1}^T \boldsymbol{\varepsilon}^e \left( \mathbf{u}^h \right) d\Omega - \int_\Omega \widetilde{\mathbf{N}}^T \frac{p^h}{K} d\Omega = 0 \tag{3.117}$$

As we deal with nonlinear elasto-plasticity, a residual-driven iterative scheme will be necessary in order to converge towards the correct solution. The two preceding equations must be therefore satisfied at step $(n+1)$, iteration $(i+1)$:

$$\int_\Omega \mathbf{B}^T \boldsymbol{\sigma}^{i+1}_{n+1} d\Omega = \mathbf{F}_{ext,n+1} \tag{3.118}$$

$$\int_\Omega \widetilde{\mathbf{N}}^T \mathbf{1}^T \boldsymbol{\varepsilon}^{e,i+1}_{n+1} d\Omega - \int_\Omega \widetilde{\mathbf{N}}^T \frac{p^{h,i+1}_{n+1}}{K} d\Omega = 0 \tag{3.119}$$

**Linearization of the Equilibrium Equation** Rewriting Equation 3.118 as:

$$\int_{\Omega} \mathbf{B}^T \boldsymbol{\Delta\sigma}_{n+1}^{i+1} d\Omega = \mathbf{F}_{ext,n+1} - \int_{\Omega} \mathbf{B}^T \boldsymbol{\sigma}_n d\Omega \tag{3.120}$$

where we have introduced:

$$\boldsymbol{\sigma}_{n+1}^{i+1} = \boldsymbol{\sigma}_n + \boldsymbol{\Delta\sigma}_{n+1}^{i+1} \tag{3.121}$$

From Equation 3.121, it is clear that $\boldsymbol{\Delta\sigma}_{n+1}^{i+1}$ is an accumulated value. We will now use the incremental constitutive equation (Equation 3.100) in order to express $\boldsymbol{\Delta\sigma}_{n+1}^{i+1}$ in function of $\boldsymbol{\Delta\varepsilon}_{n+1}^{i+1}$ and $\Delta p_{n+1}^{i+1}$ (dropping the indexes in order to simplify notation):

$$\boldsymbol{\Delta\sigma} = \overline{\mathbf{D}} \left[ \boldsymbol{\Delta\varepsilon} \left( \mathbf{u}^h \right) - \boldsymbol{\Delta\varepsilon}^p \left( \mathbf{u}^h, p^h \right) \right] + \mathbf{1} \Delta p^h \tag{3.122}$$

Applying consistency, $\dot{f}(\boldsymbol{\sigma}) = 0$ (making the hypothesis that we are in a plastic process $\Rightarrow \Delta\gamma > 0$), leads to:

$$\dot{f}(\boldsymbol{\sigma}) = \frac{\partial f}{\partial \boldsymbol{\sigma}}^T \boldsymbol{\Delta\sigma} \overset{\text{def.}}{=} \mathbf{a}^T \boldsymbol{\Delta\sigma} = 0 \tag{3.123}$$

and multiplying Equation 3.122 by $\mathbf{a}^T$ we can conclude:

$$\mathbf{a}^T \overline{\mathbf{D}} \left[ \boldsymbol{\Delta\varepsilon} - \boldsymbol{\Delta\varepsilon}^p \right] + \mathbf{a}^T \mathbf{1} \Delta p^h = 0 \tag{3.124}$$

Introducing the flow rule (Equation 3.103):

$$\mathbf{a}^T \overline{\mathbf{D}} \boldsymbol{\Delta\varepsilon} - \mathbf{a}^T \overline{\mathbf{D}} \Delta\gamma \mathbf{r} + \mathbf{a}^T \mathbf{1} \Delta p^h = 0 \tag{3.125}$$

we get an explicit expression for the plastic multiplier $\Delta\gamma$:

$$\Delta\gamma = \frac{\mathbf{a}^T \overline{\mathbf{D}} \boldsymbol{\Delta\varepsilon} + \mathbf{a}^T \mathbf{1} \Delta p^h}{\mathbf{a}^T \overline{\mathbf{D}} \mathbf{r}} \tag{3.126}$$

Reintroducing Equation 3.126 into Equation 3.122 leads to:

$$\boldsymbol{\Delta\sigma} = \overline{\mathbf{D}} \left[ \boldsymbol{\Delta\varepsilon} - \frac{\mathbf{a}^T \overline{\mathbf{D}} \boldsymbol{\Delta\varepsilon}}{\mathbf{a}^T \overline{\mathbf{D}} \mathbf{r}} \mathbf{r} - \frac{\mathbf{a}^T \mathbf{1} \Delta p^h}{\mathbf{a}^T \overline{\mathbf{D}} \mathbf{r}} \mathbf{r} \right] + \mathbf{1} \Delta p^h \tag{3.127}$$

and rearranging terms yields:

$$\boldsymbol{\Delta\sigma} = \left[\overline{\mathbf{D}} - \frac{\left(\overline{\mathbf{D}}\mathbf{r}\right)\left(\overline{\mathbf{D}}\mathbf{a}\right)^T}{\mathbf{a}^T\overline{\mathbf{D}}\mathbf{r}}\right]\boldsymbol{\Delta\varepsilon} + \left[\mathbf{1} - \frac{\left(\overline{\mathbf{D}}\mathbf{r}\right)\left(\mathbf{a}^T\mathbf{1}\right)}{\mathbf{a}^T\overline{\mathbf{D}}\mathbf{r}}\right]\Delta p^h \tag{3.128}$$

that we can express in the compact form at iteration $i+1$, step $n+1$:

$$\boldsymbol{\Delta\sigma}_{n+1}^{i+1} = \overline{\mathbf{D}}^{uu}\boldsymbol{\Delta\varepsilon}_{n+1}^{i+1} + \overline{\mathbf{D}}^{up}\Delta p_{n+1}^{h,i+1} \tag{3.129}$$

or:

$$\boldsymbol{\Delta\sigma}_{n+1}^{i+1} = \overline{\mathbf{D}}^{uu}\mathbf{B}\boldsymbol{\Delta}\mathbf{u}_{n+1}^{i+1} + \overline{\mathbf{D}}^{up}\widetilde{\mathbf{N}}\boldsymbol{\Delta}\mathbf{p}_{n+1}^{i+1} \tag{3.130}$$

During the iterative process, the accumulated values of the displacement and the pressure are updated as follows:

$$\boldsymbol{\Delta}\mathbf{u}_{n+1}^{i+1} = \boldsymbol{\Delta}\mathbf{u}_{n+1}^{i} + \boldsymbol{\Delta}\mathbf{u} \tag{3.131}$$

$$\boldsymbol{\Delta}\mathbf{p}_{n+1}^{i+1} = \boldsymbol{\Delta}\mathbf{p}_{n+1}^{i} + \boldsymbol{\Delta}\mathbf{p} \tag{3.132}$$

**Remark 19** $\boldsymbol{\Delta}\mathbf{u}$ *and* $\boldsymbol{\Delta}\mathbf{p}$ *will be the solutions of the future matrix system.*

**Remark 20** *in the case of J2-Plasticity (von Mises),* $\overline{\mathbf{D}}^{up}$ *reduces to* $\mathbf{1}$ *as* $\mathbf{a}$ *is orthogonal to* $\mathbf{1}$ *and therefore* $\mathbf{a}^T\mathbf{1} = 0$.

**Remark 21** $\Delta p_{n+1}^{h,i+1}$ *can be viewed as the mean pressure of the stress increment* $\boldsymbol{\Delta\sigma}_{n+1}^{i+1}$ *if and only if the step is converged. During the iterations, it does not always coincide with the hydrostatic part of the hydrostatic-deviatoric split of the stress increment tensor.*

**Remark 22** *the expression for* $\Delta\gamma$ *given in Equation 3.126 helps us with the building of the tangent operators, but it should be noted that the actual computation of* $\Delta\gamma$ *is made during the stress return algorithm defined by Equation 2.76.*

**Linearization of the Continuity Equation** Following the guidelines that we established for the linearization of the equilibrium equation, we can rewrite Equation 3.119 in the form:

$$\int_\Omega \widetilde{\mathbf{N}}^T \Delta R_{\theta,n+1}^{i+1} d\Omega = -\int_\Omega \widetilde{\mathbf{N}}^T R_{\theta,n} d\Omega \tag{3.133}$$

where $R_{\theta,n}$ is defined by:

$$R_{\theta,n} = \mathbf{1}^T\left(\boldsymbol{\varepsilon}_n - \boldsymbol{\varepsilon}_n^p\right) - \frac{p_n^h}{K} \tag{3.134}$$

and $\Delta R_{\theta,n+1}^{i+1}$:

$$\Delta R_{\theta,n+1}^{i+1} = \mathbf{1}^T \left( \boldsymbol{\Delta\varepsilon}_{n+1}^{i+1} - \boldsymbol{\Delta\varepsilon}_{n+1}^{p,i+1} \right) - \frac{\Delta p_{n+1}^{h,i+1}}{K} \tag{3.135}$$

Introducing the flow rule, one gets:

$$\Delta R_{\theta,n+1}^{i+1} = \mathbf{1}^T \left( \boldsymbol{\Delta\varepsilon}_{n+1}^{i+1} - \Delta\gamma \mathbf{r} \right) - \frac{\Delta p_{n+1}^{h,i+1}}{K} \tag{3.136}$$

using now the computed value of $\Delta\gamma$ (Equation 3.126), dropping the indexes:

$$\Delta R_{\theta} = \mathbf{1}^T \left( \boldsymbol{\Delta\varepsilon} - \frac{\mathbf{a}^T \overline{\mathbf{D}} \boldsymbol{\Delta\varepsilon}}{\mathbf{a}^T \overline{\mathbf{D}} \mathbf{r}} \mathbf{r} - \frac{\mathbf{a}^T \mathbf{1} \Delta p^h}{\mathbf{a}^T \overline{\mathbf{D}} \mathbf{r}} \mathbf{r} \right) - \frac{\Delta p^h}{K} \tag{3.137}$$

and finally:

$$\Delta R_{\theta} = \left[ \mathbf{1}^T - \frac{\left(\mathbf{1}^T \mathbf{r}\right) \left(\overline{\mathbf{D}} \mathbf{a}^T\right)}{\mathbf{a}^T \overline{\mathbf{D}} \mathbf{r}} \right] \boldsymbol{\Delta\varepsilon} + \left[ -\frac{1}{K} - \frac{\left(\mathbf{1}^T \mathbf{r}\right) \left(\mathbf{a}^T \mathbf{1}\right)}{\mathbf{a}^T \overline{\mathbf{D}} \mathbf{r}} \right] \Delta p^h \tag{3.138}$$

We rewrite Equation 3.138 in a more compact form, reintroducing the indexes, and using $\mathbf{B}$ and $\widetilde{\mathbf{N}}$ in order to express $\Delta R_{\theta} = f\left(\boldsymbol{\Delta u}, \boldsymbol{\Delta p}\right)$ (nodal values):

$$\Delta R_{\theta,n+1}^{i+1} = \overline{\mathbf{D}}^{pu} \mathbf{B} \boldsymbol{\Delta u}_{n+1}^{i+1} + \overline{\mathbf{D}}^{pp} \widetilde{\mathbf{N}} \boldsymbol{\Delta p}_{n+1}^{i+1} \tag{3.139}$$

**Remark 23** *in the case of J2-Plasticity (von Mises),* $\overline{\mathbf{D}}^{pu}$ *reduces to* $\mathbf{1}^T$ *and* $\overline{\mathbf{D}}^{pp}$ *reduces to* $-\frac{1}{K}$ *as* $\mathbf{r}$ *is orthogonal to* $\mathbf{1}$ *and therefore* $\mathbf{1}^T \mathbf{r} = 0$.

**Matrix Form (Incremental Nonlinear Version)** We are now ready to define the incremental matrix form ($M_{n+1}^{i+1}$) of our problem that we will later use in the nonlinear iterative scheme. Regrouping Equations 3.120 and 3.133 using the definitions of $\boldsymbol{\Delta\sigma}_{n+1}^{i+1}$ (Equation 3.130) and $\Delta R_{\theta,n+1}^{i+1}$ (Equation 3.139), we get:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ \mathbf{K}_{pu} & \mathbf{K}_{pp} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\Delta u} \\ \boldsymbol{\Delta p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_u \\ \mathbf{F}_p \end{Bmatrix} \tag{3.140}$$

with the left-hand side:

$$\mathbf{K}_{uu} = \int_\Omega \mathbf{B}^T \overline{\mathbf{D}}^{uu} \mathbf{B} d\Omega \tag{3.141}$$

$$\mathbf{K}_{up} = \int_\Omega \mathbf{B}^T \overline{\mathbf{D}}^{up} \widetilde{\mathbf{N}} d\Omega \tag{3.142}$$

$$\mathbf{K}_{pu} = \int_\Omega \widetilde{\mathbf{N}}^T \mathbf{D}^{pu} \mathbf{B} d\Omega \tag{3.143}$$

$$\mathbf{K}_{pp} = \int_\Omega \widetilde{\mathbf{N}}^T \overline{\mathbf{D}}^{pp} \widetilde{\mathbf{N}} d\Omega \tag{3.144}$$

and the right-hand side:

$$\mathbf{F}_u = \mathbf{F}_{ext,n+1} - \int_\Omega \mathbf{B}^T \boldsymbol{\sigma}^i_{n+1} d\Omega \tag{3.145}$$

$$\mathbf{F}_p = -\int_\Omega \widetilde{\mathbf{N}}^T R^i_{\theta,n+1} d\Omega \tag{3.146}$$

In the preceding development, the definition of the accumulated values (given in Equations 3.131-3.132) has been introduced, in order to isolate $\mathbf{\Delta u}$ and $\mathbf{\Delta p}$ in the left-hand side, and to compute:

$$\boldsymbol{\sigma}^i_{n+1} = \boldsymbol{\sigma}_n + \mathbf{\Delta}\boldsymbol{\sigma}^i_{n+1} \tag{3.147}$$

$$R^i_{\theta,n+1} = R_{\theta,n} + \Delta R^i_{\theta,n+1} \tag{3.148}$$

A positive definite version ($\overline{M}^{i+1}_{n+1}$) of this incremental matrix equation is obtained by multipying its second line by $-1$ (see the development of the elastic case, Equation 3.48):

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ -\mathbf{K}_{pu} & -\mathbf{K}_{pp} \end{bmatrix} \begin{Bmatrix} \mathbf{\Delta u} \\ \mathbf{\Delta p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_u \\ -\mathbf{F}_p \end{Bmatrix} \tag{3.149}$$

### 3.2.3 Stabilization

Similarly to what has been done in the elastic case, we will now add terms to the matrix Equation 3.149 in order to enhance its stability. These terms take the form:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \left( \mathbf{L}^T \boldsymbol{\sigma} \left( \mathbf{w}^h, q^h \right) \right)^T \boldsymbol{\tau} \left( \mathbf{L}^T \boldsymbol{\sigma} \left( \mathbf{u}^h, p^h \right) + \mathbf{f} \right) d\Omega \tag{3.150}$$

where $\boldsymbol{\tau}$ is a stabilization factor matrix, with $\boldsymbol{\tau} = \tau \mathbf{I}$. The nature of $\tau$ has been described in

the part concerning elasticity. It is assumed that the same kind of stabilization factor can be used in the plastic case. The weighting part and the residual part of Equation 3.150 will now be addressed separately.

### Weighting Part

In the elasto-plastic case, the weighting part of the stabilizing terms (Equation 3.150) can be written as:

$$\mathbf{L}^T\boldsymbol{\sigma}\left(\mathbf{w}^h, q^h\right) = \mathbf{L}^T\left(\overline{\mathbf{D}}\left(\boldsymbol{\varepsilon}(\mathbf{w}^h) - \boldsymbol{\varepsilon}^p(\mathbf{w}^h, q^h)\right) + \mathbf{1}q^h\right) \tag{3.151}$$

As we have already seen in the treatment of the elastic case, the displacement part of the weighting term does not really help stabilizing the problem. Keeping this in mind, we will drop the plastic contribution, avoiding therefore linearization, and rewrite:

$$\mathbf{L}^T\boldsymbol{\sigma}\left(\mathbf{w}^h, q^h\right) \simeq \mathbf{L}^T\overline{\mathbf{D}}\mathbf{B}\mathbf{c} + \mathbf{L}^T\mathbf{1}\widetilde{\mathbf{N}}\mathbf{q} \stackrel{\text{def.}}{=} \mathbf{G}_u^{el}\mathbf{c} + \mathbf{G}_p^{el}\mathbf{q} \tag{3.152}$$

**Remark 24** *the $\mathbf{G}_u^{el}$ term will vanish for linear triangles. As we have seen in the elastic case, we can also drop it for quadrilateral elements without compromising the consistency of the scheme.*

**Remark 25** *the $\mathbf{G}_p^{el}$ calculation leads to (for $n_{sd} = 3$):*

$$\begin{aligned}
\mathbf{G}_p^{el} &= \mathbf{L}^T\mathbf{1}\widetilde{\mathbf{N}} = \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_3} & 0 \\ 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} & 0 & 0 & \frac{\partial}{\partial x_3} \\ 0 & 0 & 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \widetilde{\mathbf{N}} \\
&= \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{bmatrix} \widetilde{\mathbf{N}} = \overrightarrow{\nabla}\widetilde{\mathbf{N}}
\end{aligned} \tag{3.153}$$

### Residual Part

Expressing the residual part of the stabilizing term at step $(n+1)$, iteration $(i+1)$:

$$\mathbf{L}^T\boldsymbol{\sigma}_{n+1}^{i+1}\left(\mathbf{u}^h, p^h\right) + \mathbf{f} = \mathbf{L}^T\left(\boldsymbol{\sigma}_n + \Delta\boldsymbol{\sigma}_{n+1}^{i+1}\right) + \mathbf{f} \stackrel{\text{def.}}{=} \mathbf{R}_{\sigma,n} + \boldsymbol{\Delta}\mathbf{R}_{\sigma,n+1}^{i+1} + \mathbf{f} \tag{3.154}$$

where the incremental constitutive equation (Equation 3.122) can be used in order to derive an expression for $\Delta\mathbf{R}_{\sigma,n+1}^{i+1}$:

$$\boldsymbol{\Delta}\mathbf{R}_{\sigma,n+1}^{i+1} = \mathbf{L}^T\left(\overline{\mathbf{D}}\left[\boldsymbol{\Delta}\boldsymbol{\varepsilon}_{n+1}^{i+1} - \boldsymbol{\Delta}\boldsymbol{\varepsilon}_{n+1}^{p,i+1}\right] + \mathbf{1}\Delta p_{n+1}^{h,i+1}\right) \tag{3.155}$$

Following the linearization of the equilibrium equation, we recall:

$$\boldsymbol{\Delta}\boldsymbol{\sigma}_{n+1}^{i+1} = \overline{\mathbf{D}}^{uu}\mathbf{B}\boldsymbol{\Delta}\mathbf{u}_{n+1}^{i+1} + \overline{\mathbf{D}}^{up}\widetilde{\mathbf{N}}\boldsymbol{\Delta}\mathbf{p}_{n+1}^{i+1} \tag{3.156}$$

and therefore:

$$\boldsymbol{\Delta}\mathbf{R}_{\sigma,n+1}^{i+1} = \mathbf{L}^T\overline{\mathbf{D}}^{uu}\mathbf{B}\boldsymbol{\Delta}\mathbf{u}_{n+1}^{i+1} + \mathbf{L}^T\overline{\mathbf{D}}^{up}\widetilde{\mathbf{N}}\boldsymbol{\Delta}\mathbf{p}_{n+1}^{i+1} \overset{\text{def.}}{=} \mathbf{G}_u^{ep}\boldsymbol{\Delta}\mathbf{u}_{n+1}^{i+1} + \mathbf{G}_p^{ep}\boldsymbol{\Delta}\mathbf{p}_{n+1}^{i+1} \tag{3.157}$$

While $\mathbf{R}_{\sigma,n}$ can be written as:

$$\mathbf{R}_{\sigma,n} = \mathbf{G}_u^{ep}\mathbf{u}_n + \mathbf{G}_p^{ep}\mathbf{p}_n \tag{3.158}$$

**Back to the Matrix Form**

Collecting the weighting and the residual part of the stabilizing term leads to (using the accumulated values definitions of Equations 3.131-3.132):

$$\sum_{e=1}^{n_{el}}\int_{\Omega^e}\left(\mathbf{G}_u^{el}\mathbf{c} + \mathbf{G}_p^{el}\mathbf{q}\right)^T\boldsymbol{\tau}\left(\mathbf{R}_{\sigma,n+1}^{i} + \mathbf{G}_u^{ep}\boldsymbol{\Delta}\mathbf{u} + \mathbf{G}_p^{ep}\boldsymbol{\Delta}\mathbf{p} + \mathbf{f}\right)d\Omega \tag{3.159}$$

If we add these terms to the matrix Equation 3.149 ($\overline{M}_{n+1}^{i+1}$) we will finally end up with the following system of equations ($\overline{M'}_{n+1}^{i+1}$)($\mathbf{c}$ and $\mathbf{q}$ (nodal values) will disappear as usual):

$$\begin{bmatrix} \mathbf{K}_{uu} + \mathbf{K}'_{uu} & \mathbf{K}_{up} + \mathbf{K}'_{up} \\ -\mathbf{K}_{pu} + \mathbf{K}'_{pu} & -\mathbf{K}_{pp} + \mathbf{K}'_{pp} \end{bmatrix}\begin{Bmatrix} \boldsymbol{\Delta}\mathbf{u} \\ \boldsymbol{\Delta}\mathbf{p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_u + \mathbf{F}'_u \\ -\mathbf{F}_p + \mathbf{F}'_p \end{Bmatrix} \tag{3.160}$$

where:

$$\mathbf{K}'_{uu} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_u^{el,T} \boldsymbol{\tau} \mathbf{G}_u^{ep} d\Omega \tag{3.161}$$

$$\mathbf{K}'_{up} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_u^{el,T} \boldsymbol{\tau} \mathbf{G}_p^{ep} d\Omega \tag{3.162}$$

$$\mathbf{K}'_{pu} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_p^{el,T} \boldsymbol{\tau} \mathbf{G}_u^{ep} d\Omega \tag{3.163}$$

$$\mathbf{K}'_{pp} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_p^{el,T} \boldsymbol{\tau} \mathbf{G}_p^{ep} d\Omega \tag{3.164}$$

$$\mathbf{F}'_u = -\sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_u^{el,T} \boldsymbol{\tau} \left( \mathbf{R}^i_{\sigma,n+1} + \mathbf{f} \right) d\Omega \tag{3.165}$$

$$\mathbf{F}'_p = -\sum_{e=1}^{n_{el}} \int_{\Omega^e} \mathbf{G}_p^{el,T} \boldsymbol{\tau} \left( \mathbf{R}^i_{\sigma,n+1} + \mathbf{f} \right) d\Omega \tag{3.166}$$

The nonlinear strategy used to solve the problem can be summarized in the following manner:

- Step initialisation: $n = 0$
- 1. Compute $\mathbf{F}_{ext,n+1}$
- Iterative loop initiatisation: $i = 0$
- $\mathbf{u}^0_{n+1} = \mathbf{u}_n$ and $\boldsymbol{\Delta}\mathbf{u}^0_{n+1} = \mathbf{0}$
- $\mathbf{p}^0_{n+1} = \mathbf{p}_n$ and $\boldsymbol{\Delta}\mathbf{p}^0_{n+1} = \mathbf{0}$
- 2. Convergence test: $\left\| \mathrm{RHS}_u(\overline{M'}^{i+1}_{n+1}) \right\| < TOL_u$ **and** $\left\| \mathrm{RHS}_p(\overline{M'}^{i+1}_{n+1}) \right\| < TOL_p$ ?
- YES $\Rightarrow n++$, goto 1.
- Solve $(\overline{M'}^{i+1}_{n+1})$ (Equation 3.160)
- $\mathbf{u}^{i+1}_{n+1} = \mathbf{u}^i_{n+1} + \boldsymbol{\Delta}\mathbf{u}$ and $\boldsymbol{\Delta}\mathbf{u}^{i+1}_{n+1} = \boldsymbol{\Delta}\mathbf{u}^i_{n+1} + \boldsymbol{\Delta}\mathbf{u}$
- $\mathbf{p}^{i+1}_{n+1} = \mathbf{p}^i_{n+1} + \boldsymbol{\Delta}\mathbf{p}$ and $\boldsymbol{\Delta}\mathbf{p}^{i+1}_{n+1} = \boldsymbol{\Delta}\mathbf{p}^i_{n+1} + \boldsymbol{\Delta}\mathbf{p}$
- $i++$, goto 2.

## 3.3 Finite Increment Calculus Formulation

### 3.3.1 Derivation of the Stabilized Equations

Oñate [42] has introduced a natural way to retrieve stabilization terms by introducing higher-order approximations over a finite-dimensional domain in the governing equations (in our case, equilibrium and balance of mass). We will derive first the set of stabilized equations corresponding to equilibrium over a two-dimensional domain. Considering the domain shown in figure 3-10 ($n_{sd} = 2$ has been assumed here for the sake of simplicity), balance of horizontal forces can be expressed as:

$$\begin{aligned} &\frac{h_y}{2}\left[\sigma_{11}(A)+\sigma_{11}(B)\right]-\frac{h_y}{2}\left[\sigma_{11}(C)+\sigma_{11}(D)\right] \\ &+\frac{h_x}{2}\left[\tau_{21}(A)+\tau_{21}(C)\right]-\frac{h_x}{2}\left[\tau_{21}(B)+\tau_{21}(D)\right]+h_x h_y f_1=0 \end{aligned} \tag{3.167}$$

while balance of vertical forces writes:

$$\begin{aligned} &\frac{h_x}{2}\left[\sigma_{22}(A)+\sigma_{22}(C)\right]-\frac{h_x}{2}\left[\sigma_{22}(B)+\sigma_{22}(D)\right] \\ &+\frac{h_y}{2}\left[\tau_{12}(A)+\tau_{12}(B)\right]-\frac{h_y}{2}\left[\tau_{12}(C)+\tau_{12}(D)\right]+h_x h_y f_2=0 \end{aligned} \tag{3.168}$$

where $f_1$ and $f_2$ are body forces acting per unit area and $\tau_{21}$ and $\tau_{21}$ are shear stresses. Now, the key point is to express every quantity $\bullet$ at position $(B)$, $(C)$ and $(D)$ with respect to the same quantity at position $(A)$ using the following Taylor series expansions:

$$\bullet(B) = \bullet(A)-h_y\frac{\partial\bullet}{\partial y}+\frac{h_y^2}{2}\frac{\partial^2\bullet}{\partial y^2}+O(h_y^3) \tag{3.169}$$

$$\bullet(C) = \bullet(A)-h_x\frac{\partial\bullet}{\partial x}+\frac{h_x^2}{2}\frac{\partial^2\bullet}{\partial x^2}+O(h_x^3) \tag{3.170}$$

$$\begin{aligned} \bullet(D) = &\bullet(A)-h_x\frac{\partial\bullet}{\partial x}-h_y\frac{\partial\bullet}{\partial y}+\frac{h_x^2}{2}\frac{\partial^2\bullet}{\partial x^2} \\ &+\frac{h_y^2}{2}\frac{\partial^2\bullet}{\partial y^2}+h_x h_y\frac{\partial^2\bullet}{\partial x\partial y}+O(h_x^3,h_y^3) \end{aligned} \tag{3.171}$$

Introducing Equations 3.169-3.171 into Equations 3.167-3.168 leads to, after simplification:

$$r_{m_i}-\frac{1}{2}\mathbf{h}^T\overrightarrow{\nabla}r_{m_i}=0 \tag{3.172}$$

with:

$$r_{m_i}=\sigma_{ij,j}+f_i \tag{3.173}$$

and (in the general three-dimensional case, with $n_{sd} = 3$):

$$\mathbf{h} = \left\{ \begin{array}{c} h_x \\ h_y \\ h_z \end{array} \right\} \text{ and } \overrightarrow{\bigtriangledown} = \left\{ \begin{array}{c} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{array} \right\} \tag{3.174}$$

**Remark 26** *had we neglected the second-order terms in the Taylor series, we would then retrieve the standard form of equilibrium, namely* $r_{m_i} = 0$. *Taking these second-order terms into account in Equations 3.169-3.171 introduces naturally stabilizing terms.*

**Remark 27** *this approach also allows us to introduce a directional character in our formulation through* $\mathbf{h}$.

The same derivation can be made on the balance of mass equation. This leads to the following stabilized form:

$$r_d - \frac{1}{2}\mathbf{h}^T \overrightarrow{\bigtriangledown} r_d = 0 \tag{3.175}$$

with:

$$r_d = u^e_{i,i} - \frac{p}{K} \tag{3.176}$$

### 3.3.2 (In-)compressible Elasticity (or Stokes Flow)

#### Introducing the First Equation into the Second

A particular stabilized formulation which combines the experience gained from the different derivations made in the preceding sections and the directional character of the incremental formulation is now derived. Following the method explained in [43], we proceed as follows: starting from the stabilized set of Equations 3.172 and 3.175:

$$\begin{aligned} r_{m_i} - \frac{1}{2}\mathbf{h}^T \overrightarrow{\bigtriangledown} r_{m_i} &= 0 \qquad (3.177) \\ r_d - \frac{1}{2}\mathbf{h}^T \overrightarrow{\bigtriangledown} r_d &= 0 \qquad (3.178) \end{aligned}$$

Introducing the elastic constitutive equation:

$$\begin{aligned} \sigma_{ij} &= s_{ij} + \delta_{ij} p \\ &= 2\mu \left[ \varepsilon_{ij} - \frac{1}{3} u_{k,k} \delta_{ij} \right] + \delta_{ij} p \end{aligned} \tag{3.179}$$

where $p$ is the **mean pressure** and recalling the balance of mass residual in the following

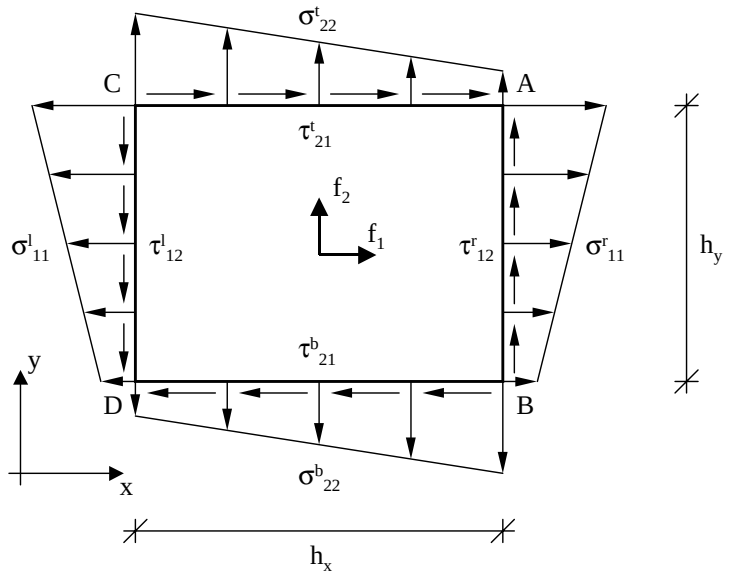



Figure 3-10: Equilibrium domain

form:

$$r_d = u_{i,i} - \frac{p}{K} \tag{3.180}$$

We will now introduce the constitutive equation 3.179 into the stabilized equilibrium equation 3.177:

$$\frac{\partial}{\partial x_j}\left[2\mu\left[\varepsilon_{ij} - \frac{1}{3}u_{k,k}\delta_{ij}\right] + \delta_{ij}p\right] + f_i - \frac{1}{2}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} = 0 \tag{3.181}$$

where in $h_{m_r}$ the $m$ index has been introduced for the sake of generality (possibility to introduce two different domains for mass and equilibrium balance). Rewriting Equation 3.181 assuming that $\mu$ is homogeneous in space leads to:

$$\frac{\partial}{\partial x_i}\left[\frac{2\mu}{3}u_{k,k} - p\right] = 2\mu\frac{\partial \varepsilon_{ij}}{\partial x_j} + f_i - \frac{1}{2}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.182}$$

or, dividing both sides by $\frac{2\mu}{3}$:

$$\frac{\partial}{\partial x_i}\left[u_{k,k} - \frac{3}{2\mu}p\right] = 3\frac{\partial \varepsilon_{ij}}{\partial x_j} + \frac{3}{2\mu}f_i - \frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.183}$$

The term in the left-hand side bracket of Equation 3.183 is form-identical to the balance of mass residual defined in Equation 3.180: only the factor multiplicating the pressure unknown $p$ differs. One can rewrite Equation 3.183 as:

$$\frac{\partial}{\partial x_i}\left[u_{k,k} - \frac{1}{K}p\right] = \frac{3K - 2\mu}{2\mu K}\frac{\partial p}{\partial x_i} + 3\frac{\partial \varepsilon_{ij}}{\partial x_j} + \frac{3}{2\mu}f_i - \frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.184}$$

Identifying the left-hand side bracket with $r_d$:

$$\frac{\partial r_d}{\partial x_i} = \frac{3K - 2\mu}{2\mu K}\frac{\partial p}{\partial x_i} + 3\frac{\partial \varepsilon_{ij}}{\partial x_j} + \frac{3}{2\mu}f_i - \frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.185}$$

or, introducing the notation $\left[\frac{3K-2\mu}{2\mu K}\frac{\partial p}{\partial x_i} + 3\frac{\partial \varepsilon_{ij}}{\partial x_j} + \frac{3}{2\mu}f_i\right] = [..]_1$:

$$\frac{\partial r_d}{\partial x_i} = [..]_1 - \frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.186}$$

Introducing this into Equation 3.178 leads to:

$$r_d + \frac{3h_{d_i}h_{m_r}}{8\mu}\frac{\partial r_{m_i}}{\partial x_r} - \frac{1}{2}h_{d_i}[..]_1 = 0 \tag{3.187}$$

**Remark 28** *The $[..]_1$ notation stands for "multiplied by a first-order in h term".*

### Weak Form

On the one hand, the weak treatment of the equilibrium equation is the standard one, i.e. multiplication by a displacement weighting function $w_i \in \mathcal{V}_i$ and integration over the domain by parts, using boundary conditions to get finally:

$$\int_\Omega w_{(i,j)} \sigma_{ij} d\Omega = \int_{\Gamma_h} w_i h_i d\Gamma + \int_\Omega w_i f_i d\Omega \tag{3.188}$$

On the other hand, we premultiply Equation 3.187 by a pressure weighting function $q \in \mathcal{P}$ and we write:

$$\int_\Omega q \left[ r_d + \frac{3 h_{d_i} h_{m_r}}{8\mu} \frac{\partial r_{m_i}}{\partial x_r} - \frac{1}{2} h_{d_i} [..]_1 \right] d\Omega = 0 \tag{3.189}$$

Expanding and integrating by parts the stabilizing term allows us to write:

$$\int_\Omega q \left[ r_d - \frac{1}{2} h_{d_i} [..]_1 \right] d\Omega - \int_\Omega \frac{\partial q}{\partial x_r} \frac{3 h_{d_i} h_{m_r}}{8\mu} r_{m_i} d\Omega + \int_\Gamma q n_r \frac{3 h_{d_i} h_{m_r}}{8\mu} r_{m_i} d\Gamma = 0 \tag{3.190}$$

The purpose of scrutinizing the FIC approach was to introduce a directional character in our stabilized formulation. Now we would like to identify a formulation which reduces to the same Euler-Lagrange equations as the modified-GLS version: we will call this formulation FIC-scheme. If we neglect boundary terms and $[..]_1$, this leads us to a weak statement which takes the form:

$$\int_\Omega q \left[ u_{k,k} - \frac{p}{K} \right] d\Omega \underline{- \int_\Omega \frac{\partial q}{\partial x_r} \frac{3 h_{d_i} h_{m_r}}{8\mu} \left[ \sigma_{ij,j} + f_i \right] d\Omega} = 0 \tag{3.191}$$

and the weak form is defined by: find $u_i \in \mathcal{S}_i$ and $p \in \mathcal{P}$ such that, for all $w_i \in \mathcal{V}_i$ and $q \in \mathcal{P}$, the two scalar equations 3.189 and 3.191 hold. The only difference with the standard mixed form of our problem is the underlined stabilizing term.

**Euler-Lagrange Equations** Combining Equation 3.191 with Equation 3.92 yields the following form, if we assume that all fields are globally $C^0$ but may suffer from slope discontinuities at element boundaries (therefore introducing jump terms):

$$\begin{aligned} &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \widetilde{w}_i \left( \sigma_{ij,j} + f_i \right) d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega^e} q \left[ u_{k,k} - \frac{p}{K} \right] d\Omega \\ &- \int_{\Gamma_h} w_i \left( \sigma_{ij} n_j - h_i \right) d\Gamma - \int_{\Gamma_{int}} w_i \left\langle \sigma_{ij} n_j \right\rangle d\Gamma = 0 \end{aligned} \tag{3.192}$$

where:

$$\widetilde{w}_i = w_i - \frac{\partial q}{\partial x_r}\frac{3h_{d_i}h_{m_r}}{8\mu} \tag{3.193}$$

Equation 3.192 yields the following four **Euler-Lagrange equations**, which can be compared and identified with the ones that we obtained earlier in the scalar case (see Equations 3.92-3.98):

- $\sigma_{ij,j} + f_i = 0$ inside every element $\Omega^e$
- $u_{i,i} - \frac{p}{K} = 0$ inside every element $\Omega^e$
- $\sigma_{ij}n_j - h_i = 0$ on the $\Gamma_h$ boundary
- $\langle \sigma_{ij}n_j \rangle = 0$: continuity condition of the stress across element boundaries

and we therefore conclude that this adopted formulation is form-similar to the modified-GLS approach as Euler-Lagrange equations coincide.

### Galerkin and Matrix Forms

The Galerkin ($G$) counterpart of the weak formulation ($W$) is obtained by introducing finite-dimensional subspaces, namely $\mathcal{S}_i^h \subset \mathcal{S}_i$, $\mathcal{V}_i^h \subset \mathcal{V}_i$ and $\mathcal{P}^h \subset \mathcal{P}$. The introduction of two sets of shape functions $N_A(\mathbf{x})$ and $\widetilde{N}_{\widetilde{A}}(\mathbf{x})$ leads to:

$$\int_\Omega \mathbf{B}^T\boldsymbol{\sigma}d\Omega = \mathbf{F}_{ext} \tag{3.194}$$

$$\int_\Omega \widetilde{\mathbf{N}}^T\mathbf{1}^T\mathbf{B}\mathbf{d}d\Omega - \frac{1}{\lambda}\int_\Omega \widetilde{\mathbf{N}}^T\widetilde{\mathbf{N}}\mathbf{p}d\Omega - \frac{3}{8\mu}\sum_{e=1}^{n_{el}}\int_{\Omega_e}\overrightarrow{\nabla}\widetilde{\mathbf{N}}^T\mathbf{h}_m\mathbf{h}_d^T\left[\mathbf{L}^T\boldsymbol{\sigma}+\mathbf{f}\right]d\Omega = 0 \tag{3.195}$$

The second equation should be multiplied by $-1$ in order to get a positive-definite form:

$$-\int_\Omega \widetilde{\mathbf{N}}^T\mathbf{1}^T\mathbf{B}\mathbf{d}d\Omega + \frac{1}{\lambda}\int_\Omega \widetilde{\mathbf{N}}^T\widetilde{\mathbf{N}}\mathbf{p}d\Omega + \frac{3}{8\mu}\sum_{e=1}^{n_{el}}\int_{\Omega_e}\overrightarrow{\nabla}\widetilde{\mathbf{N}}^T\mathbf{h}_m\mathbf{h}_d^T\left[\mathbf{L}^T\boldsymbol{\sigma}+\mathbf{f}\right]d\Omega = 0 \tag{3.196}$$

### A Systematic Comparison of FIC-Scheme and Modified-GLS

The modified-GLS scheme described in section 3.1 that was heuristically found to be the most stable is the one where the displacement contribution in the weighting part of the stabilizing term is neglected. This term therefore takes the form:

$$\sum_{e=1}^{n_{el}}\int_{\Omega_e}\mathbf{G}_p^{el,T}\boldsymbol{\tau}\left[\mathbf{L}^T\boldsymbol{\sigma}+\mathbf{f}\right]d\Omega = 0 \tag{3.197}$$

where the stabilization factors matrix $\boldsymbol{\tau}$ has the following structure in 3D:

$$\boldsymbol{\tau} = \tau \mathbf{I} = \frac{\alpha^e \left(h^e\right)^2}{2\mu} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.198}$$

The stabilizing term of Equation 3.195 and the one defined in Equation 3.197 are compared in the following table:

| | **FIC-scheme** | **modified-GLS** |
|---|---|---|
| sign of the term | $+$ | $+$ |
| weighting part | $\left(\overrightarrow{\bigtriangledown}\tilde{\mathbf{N}}\right)^T$ | $\left(\overrightarrow{\bigtriangledown}\tilde{\mathbf{N}}\right)^T$ |
| stabilization factor matrix $\boldsymbol{\tau}$ | $\frac{3}{8\mu}\begin{bmatrix} h_{mx}h_{dx} & h_{mx}h_{dy} & h_{mx}h_{dz} \\ h_{my}h_{dx} & h_{my}h_{dy} & h_{my}h_{dz} \\ h_{mz}h_{dx} & h_{mz}h_{dy} & h_{mz}h_{dz} \end{bmatrix}$ | $\frac{\alpha^e(h^e)^2}{2\mu}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ |
| residual part | $\left[\mathbf{L}^T\boldsymbol{\sigma} + \mathbf{f}\right]$ | $\left[\mathbf{L}^T\boldsymbol{\sigma} + \mathbf{f}\right]$ |

### 3.3.3 Plasticity

#### Introducing the First Equation into the Second

Starting from the stabilized set of Equations 3.172 and 3.175:

$$r_{m_i} - \frac{1}{2}\mathbf{h}^T\overrightarrow{\bigtriangledown} r_{m_i} = 0 \tag{3.199}$$

$$r_d - \frac{1}{2}\mathbf{h}^T\overrightarrow{\bigtriangledown} r_d = 0 \tag{3.200}$$

where $r_d$ takes the usual form, with $p$ the **mean pressure**:

$$r_d = u^e_{i,i} - \frac{p}{K} \tag{3.201}$$

and introducing the following nonlinear constitutive equation:

$$\sigma_{ij} = s_{ij} + \delta_{ij} p \tag{3.202}$$

with the deviatoric stress $s_{ij}$ defined as:

$$s_{ij} = 2\mu e^e_{ij} = 2\mu(e_{ij} - e^p_{ij}) \tag{3.203}$$

The deviatoric strain $e_{ij}$ is linked with $\varepsilon_{ij}$ through the following relation:

$$e_{ij} = \varepsilon_{ij} - \frac{1}{3}\delta_{ij}u_{r,r} \tag{3.204}$$

Introducing Equations 3.203 and 3.204 into Equation 3.202 allows us to write:

$$\sigma_{ij} = 2\mu(\varepsilon_{ij} - \frac{1}{3}\delta_{ij}u_{r,r}) - 2\mu e^p_{ij} + \delta_{ij}p \tag{3.205}$$

We will now introduce the constitutive equation 3.205 into the stabilized equilibrium equation 3.199, with $u_{r,r} = u^e_{r,r} + u^p_{r,r}$:

$$\frac{\partial}{\partial x_j}\left[2\mu(\varepsilon_{ij} - \frac{1}{3}\delta_{ij}u^e_{r,r}) - \frac{2\mu}{3}\delta_{ij}u^p_{r,r} - 2\mu e^p_{ij} + \delta_{ij}p\right] + f_i - \frac{1}{2}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} = 0 \tag{3.206}$$

Rearranging the terms leads to:

$$\frac{\partial}{\partial x_j}\left[\frac{2\mu}{3}\delta_{ij}u^e_{r,r} - \delta_{ij}p\right] = \frac{\partial}{\partial x_j}\left[2\mu(\varepsilon_{ij} - e^p_{ij} - \frac{2\mu}{3}\delta_{ij}u^p_{r,r})\right] + f_i - \frac{1}{2}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.207}$$

Dividing both sides by $\frac{2\mu}{3}$ gives:

$$\frac{\partial}{\partial x_i}\left[u^e_{r,r} - \frac{3}{2\mu}p\right] = 3\frac{\partial}{\partial x_j}\left[(\varepsilon_{ij} - e^p_{ij} - \frac{1}{3}\delta_{ij}u^p_{r,r})\right] + \frac{3}{2\mu}f_i - \frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.208}$$

The term in the left-hand side bracket of Equation 3.208 is form-identical to the balance of mass residual defined in Equation 3.201: only the factor multiplicating the pressure unknown $p$ differs. Similarly to what we have done in the elastic case, we can say that:

$$\frac{\partial}{\partial x_i}\left[u^e_{r,r} - \frac{1}{K}p\right] = \frac{3K - 2\mu}{2\mu K}\frac{\partial p}{\partial x_i} + 3\frac{\partial}{\partial x_j}\left[(\varepsilon_{ij} - e^p_{ij} - \frac{1}{3}\delta_{ij}u^p_{r,r})\right] + \frac{3}{2\mu}f_i - \frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.209}$$

Identifying the left-hand side bracket with $r_d$:

$$\frac{\partial r_d}{\partial x_i} = \frac{3K - 2\mu}{2\mu K}\frac{\partial p}{\partial x_i} + 3\frac{\partial}{\partial x_j}\left[(\varepsilon_{ij} - e^p_{ij} - \frac{1}{3}\delta_{ij}u^p_{r,r})\right] + \frac{3}{2\mu}f_i - \frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.210}$$

or, introducing the notation $\left[\frac{3K-2\mu}{2\mu K}\frac{\partial p}{\partial x_i}+3\frac{\partial}{\partial x_j}\left[(\varepsilon_{ij}-e^p_{ij}-\frac{1}{3}\delta_{ij}u^p_{r,r})\right]+\frac{3}{2\mu}f_i\right]=[..]^p_1$:

$$\frac{\partial r_d}{\partial x_i}=[..]^p_1-\frac{3}{4\mu}h_{m_r}\frac{\partial r_{m_i}}{\partial x_r} \tag{3.211}$$

and introducing Equation 3.211 into 3.200, we can write:

$$r_d+\frac{3h_{d_i}h_{m_r}}{8\mu}\frac{\partial r_{m_i}}{\partial x_r}-\frac{1}{2}h_{d_i}[..]^p_1=0 \tag{3.212}$$

**Weak Form**

On the one hand, the weak treatment of the equilibrium equation is the standard one, i.e. multiplication by a displacement weighting function $w_i\in\mathcal{V}_i$ and integration over the domain by parts, using boundary conditions to get finally:

$$\int_\Omega w_{(i,j)}\sigma_{ij}d\Omega=\int_{\Gamma_h}w_ih_id\Gamma+\int_\Omega w_if_id\Omega \tag{3.213}$$

On the other hand, we premultiply Equation 3.212 by a pressure weighting function $q\in\mathcal{P}$ and we write:

$$\int_\Omega q\left[r_d+\frac{3h_{d_i}h_{m_r}}{8\mu}\frac{\partial r_{m_i}}{\partial x_r}-\frac{1}{2}h_{d_i}[..]^p_1\right]d\Omega=0 \tag{3.214}$$

Expanding and integrating by parts the stabilizing term allows us to write:

$$\int_\Omega q\left[r_d-\frac{1}{2}h_{d_i}[..]^p_1\right]d\Omega-\int_\Omega\frac{\partial q}{\partial x_r}\frac{3h_{d_i}h_{m_r}}{8\mu}r_{m_i}d\Omega+\int_\Gamma qn_r\frac{3h_{d_i}h_{m_r}}{8\mu}r_{m_i}d\Gamma=0 \tag{3.215}$$

Invoking the same argument as in the linear case, we will neglect boundary terms and $[..]^p_1$ in order to retrieve a directional weak form which yields the same Euler-Lagrange equations as in the modified-GLS (scalar) case. This finally leads us to:

$$\int_\Omega q\left[u^e_{k,k}-\frac{p}{K}\right]d\Omega\underline{-\int_\Omega\frac{\partial q}{\partial x_r}\frac{3h_{d_i}h_{m_r}}{8\mu}\left[\sigma_{ij,j}+f_i\right]d\Omega}=0 \tag{3.216}$$

and the weak form is defined by: find $u_i\in\mathcal{S}_i$ and $p\in\mathcal{P}$ such that, for all $w_i\in\mathcal{V}_i$ and $q\in\mathcal{P}$, the two scalar equations 3.213 and 3.216 hold. The only difference with the standard mixed form of our problem is the underlined stabilizing term.

**Galerkin and Matrix Forms**

The introduction of finite-dimensional subspaces with bases defined by shape functions $N_A(\mathbf{x})$ and $\widetilde{N}_{\widetilde{A}}(\mathbf{x})$ allows us to write in vector notations, keeping in mind that as we deal with nonlinear elastoplasticity the equations must be satisfied at step $n+1$, iteration $i+1$:

$$\int_\Omega \mathbf{B}^T \boldsymbol{\sigma}_{n+1}^{i+1} d\Omega = \mathbf{F}_{ext,n+1} \tag{3.217}$$

$$\int_\Omega \widetilde{\mathbf{N}}^T \overline{\mathbf{D}}^{pu} \mathbf{B} \mathbf{d}_{n+1}^{i+1} d\Omega + \int_\Omega \widetilde{\mathbf{N}}^T \overline{\mathbf{D}}^{pp} \widetilde{\mathbf{N}} \mathbf{p}_{n+1}^{i+1} d\Omega - \frac{3}{8\mu} \sum_{e=1}^{n_{el}} \int_{\Omega_e} \overrightarrow{\nabla} \widetilde{\mathbf{N}}^T \mathbf{h}_m \mathbf{h}_d^T \left[ \mathbf{L}^T \boldsymbol{\sigma}_{n+1}^{i+1} + \mathbf{f} \right] d\Omega = 0 \tag{3.218}$$

Again the second equation should be multiplied by $-1$ in order to get a positive-definite form:

$$-\int_\Omega \widetilde{\mathbf{N}}^T \overline{\mathbf{D}}^{pu} \mathbf{B} \mathbf{d}_{n+1}^{i+1} d\Omega - \int_\Omega \widetilde{\mathbf{N}}^T \overline{\mathbf{D}}^{pp} \widetilde{\mathbf{N}} \mathbf{p}_{n+1}^{i+1} d\Omega + \frac{3}{8\mu} \sum_{e=1}^{n_{el}} \int_{\Omega_e} \overrightarrow{\nabla} \widetilde{\mathbf{N}}^T \mathbf{h}_m \mathbf{h}_d^T \left[ \mathbf{L}^T \boldsymbol{\sigma}_{n+1}^{i+1} + \mathbf{f} \right] d\Omega = 0 \tag{3.219}$$

**Linearization**

Introducing the following linearization:

$$\boldsymbol{\sigma}_{n+1}^{i+1} = \boldsymbol{\sigma}_n + \boldsymbol{\Delta\sigma}_{n+1}^{i+1} \tag{3.220}$$

$$\mathbf{u}_{n+1}^{i+1} = \mathbf{u}_n + \boldsymbol{\Delta} \mathbf{u}_{n+1}^{i+1} = \mathbf{u}_n + \boldsymbol{\Delta} \mathbf{u}_{n+1}^{i} + \boldsymbol{\Delta} \mathbf{u} \tag{3.221}$$

$$\mathbf{p}_{n+1}^{i+1} = \mathbf{p}_n + \boldsymbol{\Delta} \mathbf{p}_{n+1}^{i+1} = \mathbf{p}_n + \boldsymbol{\Delta} \mathbf{p}_{n+1}^{i} + \boldsymbol{\Delta} \mathbf{p} \tag{3.222}$$

and rewriting the constitutive equation for $\boldsymbol{\Delta\sigma}_{n+1}^{i+1}$ as:

$$\boldsymbol{\Delta\sigma}_{n+1}^{i+1} = \overline{\mathbf{D}}(\boldsymbol{\Delta\varepsilon}_{n+1}^{i+1} - \Delta\gamma \mathbf{r}) + \mathbf{1} \Delta p_{n+1}^{h,i+1} = \overline{\mathbf{D}}^{uu} \mathbf{B} \boldsymbol{\Delta} \mathbf{u}_{n+1}^{i+1} + \overline{\mathbf{D}}^{up} \widetilde{\mathbf{N}} \boldsymbol{\Delta} \mathbf{p}_{n+1}^{i+1} \tag{3.223}$$

leads to the following representation of the stabilized matrix system:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ -\mathbf{K}_{pu} + \mathbf{K}'_{pu} & -\mathbf{K}_{pp} + \mathbf{K}'_{pp} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\Delta} \mathbf{u} \\ \boldsymbol{\Delta} \mathbf{p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_u \\ -\mathbf{F}_p \end{Bmatrix} + \begin{Bmatrix} \mathbf{0} \\ \mathbf{F}'_p \end{Bmatrix} \tag{3.224}$$

with the left-hand side:

$$\mathbf{K}_{uu} = \int_\Omega \mathbf{B}^T \overline{\mathbf{D}}^{uu} \mathbf{B} d\Omega \tag{3.225}$$

$$\mathbf{K}_{up} = \int_\Omega \mathbf{B}^T \overline{\mathbf{D}}^{up} \widetilde{\mathbf{N}} d\Omega \tag{3.226}$$

$$\mathbf{K}_{pu} = \int_\Omega \widetilde{\mathbf{N}}^T \overline{\mathbf{D}}^{pu} \mathbf{B} d\Omega \tag{3.227}$$

$$\mathbf{K}_{pp} = \int_\Omega \widetilde{\mathbf{N}}^T \overline{\mathbf{D}}^{pp} \widetilde{\mathbf{N}} d\Omega \tag{3.228}$$

$$\mathbf{K}'_{pu} = \frac{3}{8\mu} \sum_{e=1}^{n_{el}} \int_{\Omega_e} \left( \overrightarrow{\nabla} \widetilde{\mathbf{N}} \right)^T \mathbf{h}_m \mathbf{h}_d^T \mathbf{L}^T \overline{\mathbf{D}}^{uu} \mathbf{B} d\Omega \tag{3.229}$$

$$\mathbf{K}'_{pp} = \frac{3}{8\mu} \sum_{e=1}^{n_{el}} \int_{\Omega_e} \left( \overrightarrow{\nabla} \widetilde{\mathbf{N}} \right)^T \mathbf{h}_m \mathbf{h}_d^T \mathbf{L}^T \overline{\mathbf{D}}^{up} \widetilde{\mathbf{N}} d\Omega \tag{3.230}$$

and the right-hand side:

$$\mathbf{F}_u = \mathbf{F}_{ext,n+1} - \int_\Omega \mathbf{B}^T \boldsymbol{\sigma}^i_{n+1} d\Omega \tag{3.231}$$

$$\mathbf{F}_p = -\int_\Omega \widetilde{\mathbf{N}}^T \left[ \mathbf{1}^T \left( \boldsymbol{\varepsilon}^i_{n+1} - \boldsymbol{\varepsilon}^{p,i}_{n+1} \right) - \frac{p^{h,i}_{n+1}}{K} \right] d\Omega \tag{3.232}$$

$$\mathbf{F}'_p = -\frac{3}{8\mu} \sum_{e=1}^{n_{el}} \int_{\Omega_e} \left( \overrightarrow{\nabla} \widetilde{\mathbf{N}} \right)^T \mathbf{h}_m \mathbf{h}_d^T \mathbf{f} d\Omega \tag{3.233}$$

**Remark 29** *All* $\overline{\mathbf{D}}$ *matrices are obtained through incremental consistency, they are defined in Equations 3.128-3.129 and 3.138-3.139.*

### 3.3.4 Definition of the Stabilization Parameters

#### Projection Procedure

Oñate [43] gives a procedure to compute $\mathbf{h}_m$ and $\mathbf{h}_d$ as functions of the velocities $\mathbf{u}$ for Navier-Stokes flow. Let's assume first that both stabilization parameters are the same, which implies:

$$\mathbf{h}_m = \mathbf{h}_d \tag{3.234}$$

The following form of $\mathbf{h}_m$ has been introduced by Oñate:

$$\mathbf{h}_m = h \frac{\mathbf{u}}{\|\mathbf{u}\|} \tag{3.235}$$

where $h$ is given by the maximal value of the projection of the velocity field $\mathbf{u}$ on the sides $\mathbf{l}_j$ of the element ($j = 1...n_{sides}$):

$$h = \max\left(\mathbf{l}_j^T \frac{\mathbf{u}}{\|\mathbf{u}\|}\right) \tag{3.236}$$

By analogy, we define in our case $\mathbf{h}_m$ with respect to the last converged accumulated increment of displacement $\boldsymbol{\Delta}\mathbf{u}_n$:

$$\mathbf{h}_m = h\frac{\boldsymbol{\Delta}\mathbf{u}_n}{\|\boldsymbol{\Delta}\mathbf{u}_n\|} \tag{3.237}$$

and:

$$h = \max\left(\mathbf{l}_j^T \frac{\boldsymbol{\Delta}\mathbf{u}_n}{\|\boldsymbol{\Delta}\mathbf{u}_n\|}\right) \tag{3.238}$$

**Remark 30** *the $\mathbf{h}_m\mathbf{h}_d^T$ in the FIC-scheme formulation induces off-diagonal terms in the stabilization factor matrix. Benchmark tests on the thick cylinder problem have shown that these terms infer coupling between the x and the y direction and therefore the residual in the x direction could be compensated by the residual in the y direction. On the one hand the sum of the two residuals could vanish, while on the other hand the individual residuals could be different than zero. In order to avoid this spurious phenomenon, it has been decided to modify the stabilization factor matrix as follows. The old version read:*

$$\boldsymbol{\tau} = \frac{3}{8\mu}\mathbf{h}_m\mathbf{h}_d^T = \frac{3}{8\mu}\begin{bmatrix} h_{mx}h_{dx} & h_{mx}h_{dy} & h_{mx}h_{dz} \\ h_{my}h_{dx} & h_{my}h_{dy} & h_{my}h_{dz} \\ h_{mz}h_{dx} & h_{mz}h_{dy} & h_{mz}h_{dz} \end{bmatrix} \tag{3.239}$$

*It has been replaced in all our following benchmarks by ($\mathbf{h}_m = \mathbf{h}_d$ by hypothesis):*

$$\boldsymbol{\tau} = \frac{3}{8\mu}\begin{bmatrix} h^2\frac{\Delta u_{x,n}^2}{\|\boldsymbol{\Delta}\mathbf{u}_n\|^2} & 0 & 0 \\ 0 & h^2\frac{\Delta u_{y,n}^2}{\|\boldsymbol{\Delta}\mathbf{u}_n\|^2} & 0 \\ 0 & 0 & h^2\frac{\Delta u_{z,n}^2}{\|\boldsymbol{\Delta}\mathbf{u}_n\|^2} \end{bmatrix} \tag{3.240}$$

*where $h$ is given by Equation 3.238. In the plane strain case ($n_{sd} = 2$), the last column and row of $\boldsymbol{\tau}$ has to be omitted.*

## 3.4 Laplacian Pressure Operator Scheme (LPOS or "$\mathbf{K}'_{pp}$ only")

Pastor & al [44] (see also Oñate [42] and Brezzi & Pitkäranta [6]) propose to add another term to the original mixed problem defined by Equation 3.140. Instead of weighting the residual of the equilibrium equation, they propose to add to the pressure equation the divergence of the equilibrium equation multiplied by a constant dependent on the mesh size.

Taking the divergence of the equilibrium equation leads to:

$$div(\sigma_{ij,j} + f_i) = 0 \tag{3.241}$$

Assuming that body forces are divergence-free and expressing $\sigma_{ij} = s_{ij} + \delta_{ij}p$:

$$div(s_{ij,j} + p_{,i}) = 0 \tag{3.242}$$

or, introducing the Laplacian operator $\nabla^2 = (\cdot)_{,ii}$:

$$div(s_{ij,j}) + \nabla^2 p = 0 \tag{3.243}$$

Introducing the continuity equation 3.180 into the computation of $div(s_{ij,j})$ leads to, after some algebra [44]:

$$div(s_{ij,j}) = \frac{4\mu}{3}\nabla^2\left(\frac{p}{K}\right) \tag{3.244}$$

Combining Equations 3.243 and 3.244 allows us to write:

$$div(s_{ij,j}) + \nabla^2 p = \frac{\lambda + 2\mu}{K}\nabla^2 p = 0 \tag{3.245}$$

where $K = \lambda + \frac{2}{3}\mu$ as usual. The divergence of the equilibrium equation therefore reduces to $\nabla^2 p = 0$, and the stabilized pressure equation becomes:

$$-(u^e_{i,i} - \frac{p}{K}) + \tau\nabla^2 p = 0 \tag{3.246}$$

**Remark 31** *the first negative sign is present in order to get a positive-definite system.*

For the constant, Pastor et al propose the following definition, based on Hughes & al works [27]:

$$\tau = \frac{\alpha^e (h^e)^2}{2\mu} \tag{3.247}$$

The incremental matrix system to be solved at each iteration therefore becomes:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ -\mathbf{K}_{pu} & -\mathbf{K}_{pp}+\mathbf{K}'_{pp} \end{bmatrix} \begin{Bmatrix} \mathbf{\Delta u} \\ \mathbf{\Delta p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_u \\ -\mathbf{F}_p+\mathbf{F}'_p \end{Bmatrix} \tag{3.248}$$

where $\mathbf{K}_{uu}$, $\mathbf{K}_{up}$, $\mathbf{K}_{pu}$, $\mathbf{K}_{pp}$, $\mathbf{F}_u$ and $\mathbf{F}_p$ have been defined by Equations 3.141-3.146 and:

$$\mathbf{K}'_{pp} = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau \left(\overrightarrow{\nabla}\widetilde{\mathbf{N}}\right)^T \left(\overrightarrow{\nabla}\widetilde{\mathbf{N}}\right) d\Omega \tag{3.249}$$

$$\mathbf{F}'_p = -\sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau \left(\overrightarrow{\nabla}\widetilde{\mathbf{N}}\right)^T \left(\overrightarrow{\nabla}\widetilde{\mathbf{N}}\mathbf{p}^i_{n+1}\right) d\Omega \tag{3.250}$$

## 3.5 Benchmarks

### 3.5.1 Introduction

In this section, three kinds of benchmarks are conducted: first, a thick cylinder test loaded by an internal pressure, then the stability analysis of a vertical cut and finally the bearing capacity of a superficial strip footing. Both incompressible and dilatant plastic flow models are used. Stabilized formulations are tested and a tentative calibration of the stabilization parameter $\alpha^e$ is done. The mixing of different types of elements in the same mesh is also addressed for every problem. Reference solutions result from theory or from other finite element techniques ($\overline{\mathbf{B}}$ or EAS quadrilaterals, "cross-diagonal pattern" triangles [39]).

**Remark 32** *as seen in the preceding sections, the stabilized formulations giving the most coherent results are the ones neglecting the* $\mathbf{G}_u$ *factor in the weighting part of the stabilizing terms (i.e. using the pressure gradient only as a weighting term). Only this kind of formulations is used in the following, see e.g. Equation 3.197.*

The following name convention will be used in order to distinguish between the three different kinds of stabilization:

- **scalar** - the **modified-GLS** derivation (see sections 3.1 & 3.2) with $\tau = \frac{\alpha^e (h^e)^2}{2\mu}$ [27]
- **FIC-scheme** - the formulation based on Finite Increment Calculus (see section 3.3) which introduces a **directional** aspect [43]
- **LPOS** - the Laplacian Pressure Operator Scheme, or "$\mathbf{K}'_{pp}$ only" (see section 3.4) [44]

Convergence is assumed when both Euclidean norms of the right-hand side of the linear system (displacement and pressure parts) are lower than 1% of the norm at the beginning of the step; divergence is assumed when one of the norms exceeds ten times the norm at the beginning of the step.

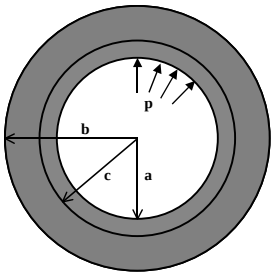



Figure 3-11: Thick cylinder test geometry

### 3.5.2 Thick Cylinder Test

#### Problem Statement and Analytical Solution

This experiment concerns a thick cylinder test, which has an analytical solution given by Hill in [25]. The internal and external radii of the cylinder are $a = 1.0$ and $b = 2.0\ m$, and we denote by $c$ the plastic radius. Young modulus $E = 21000\ kN/m^2$ and Poisson ratio $\nu = 0.49999$. The original solution [25] involves Tresca's yield criterion: $\sigma_\theta - \sigma_r = \sigma_y = 24\ kN/m^2$, where $\sigma_\theta$ is the circumferential stress, $\sigma_r$ the radial stress and $\sigma_y$ the yield stress (the longitudinal stress $\sigma_z$ doesn't play any role in the plastic process as $\nu \rightarrow 0.5 \Longrightarrow \sigma_z = \frac{1}{2}(\sigma_\theta + \sigma_r)$).

In the incompressible case, a corresponding von Mises criterion can be expressed as: $\sigma_\theta - \sigma_r = \sigma_{y'} = \frac{2\sigma_y}{\sqrt{3}}$, which corresponds to a value of the von Mises parameter $k = \frac{\sigma_y}{\sqrt{3}} = 13.8564\ kN/m^2$.

The internal pressure $p$ varies between $8\ kN/m$ and $20\ kN/m$, this latter value corresponding to the total plastification of the cylinder and its failure.

**Elastic Regime** The radial displacement is given by:

$$u_{el} = \frac{(1+\nu)p}{E\left(\frac{b^2}{a^2} - 1\right)}\left[(1-2\nu)\,r + \frac{b^2}{r}\right] \tag{3.251}$$

where $r$ is the radius at which the unknown value is computed. The radial and circumferential

stresses read:

$$\sigma_r = \frac{-p\left(\frac{b^2}{r^2}-1\right)}{\left(\frac{b^2}{a^2}-1\right)} \tag{3.252}$$

$$\sigma_\theta = \frac{p\left(\frac{b^2}{r^2}+1\right)}{\left(\frac{b^2}{a^2}-1\right)} \tag{3.253}$$

**Partly Plastic Cylinder** The plastic radius $c$ can be deduced from the following expression:

$$p = 2k\left[\ln\left(\frac{c}{a}\right)+\frac{1}{2}\left(1-\frac{c^2}{b^2}\right)\right] \tag{3.254}$$

The value of $c$ can then be used to compute the new radial displacement after partial plastification of the tube:

$$u_{pl} = \frac{(1+\nu)kc^2}{Eb^2}\left[(1-2\nu)\,r+\frac{b^2}{r}\right] \tag{3.255}$$

and the stresses in the elastic region ($c < r < b$) now read:

$$\sigma_r = \frac{-kc^2\left(\frac{b^2}{r^2}-1\right)}{b^2} \tag{3.256}$$

$$\sigma_\theta = \frac{kc^2\left(\frac{b^2}{r^2}+1\right)}{b^2} \tag{3.257}$$

while the stresses in the plastic region ($a < r < c$) are given by:

$$\sigma_r = 2k(-\frac{1}{2}-\ln\left(\frac{c}{r}\right)+\frac{c^2}{2b^2}) \tag{3.258}$$

$$\sigma_\theta = 2k(\frac{1}{2}-\ln\left(\frac{c}{r}\right)+\frac{c^2}{2b^2}) \tag{3.259}$$

### Quadrilateral Elements

**160 Q4 Mesh** We first consider a 160 Q4 elements mesh (see Figure 3-12). Due to symmetry, we only modelize a quarter of the cylinder. Interpolation fields are bilinear for both the pressure and the displacement, which requires stabilization. The theoretical solution found for the radial displacement (Equations 3.251 and 3.255), the radial stress (Equations 3.252 and 3.256) and the circumferential stress (Equations 3.253 and 3.257) are compared with the solution obtained using stabilized finite elements (both the scalar version and the directional version emanating from the FIC-scheme). Different values of the $\alpha^e$ parameter are tested in order to tune it.

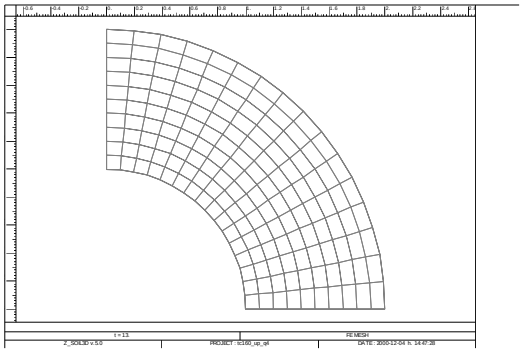


Figure 3-12: 160 Q4 mesh

Finally the $\overline{\mathbf{B}}$ solution (pure displacement formulation) is also plotted for comparison.

**Results Discussion** Looking at Figures 3-13 - 3-15 we notice that the $\overline{\mathbf{B}}$ solution matches the theoretical results well (except a small undershoot of the circumferential peak stress). This is not a surprise as the $\overline{\mathbf{B}}$ technique is known to overcome volumetric locking in incompressible situations [28], like here for a von Mises yield surface.

Speaking of scalar stabilized formulations, $\alpha^e = 0.01$ seems to be too small for stabilizing the stress field (oscillations are still noticeable). $\alpha^e = 1$ might be too much as the circumferential stress peak cannot be retrieved (Figure 3-14). A good value seems to be therefore $\alpha^e = 0.1 \rightarrow \frac{1}{6}$.

For the calculation involving the directional stabilization derived from the Finite Increment Calculus method, a good fitting is obtained with its stabilization factor defined as $\frac{3}{8\mu}\mathbf{h}\mathbf{h}^T$ i.e. $\alpha^e = 1$ (although no $\alpha^e$ is formally introduced in the original FIC method).

**Remark 33** *the poor approximation of stresses at the internal radius is a boundary effect due to the coarse discretization.*

**Additional Comparisons** We now compare one of the formulation giving the best results (i.e. the FIC-scheme with $\alpha^e = 1$) with a solution obtained using a denser mesh (see Figure 3-16) and with the formulation involving only the $\mathbf{K}'_{pp}$ term in the stabilization part (LPOS).

Figures 3-17 - 3-19 show that almost no difference can be noticed between the three analyses. The propagation of the yield radius is also in good agreement with the theoretical solution (Figure 3-20).

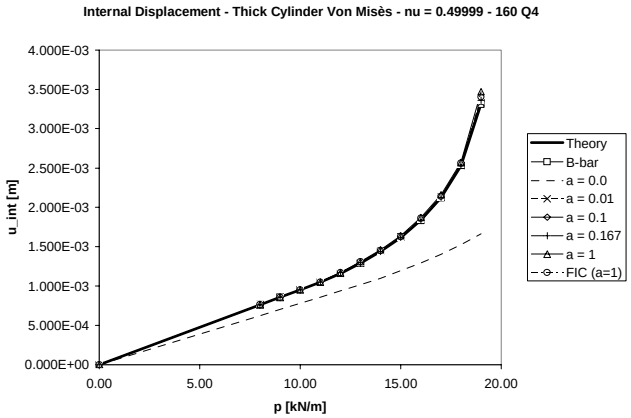



Figure 3-13: Radial displacement comparison

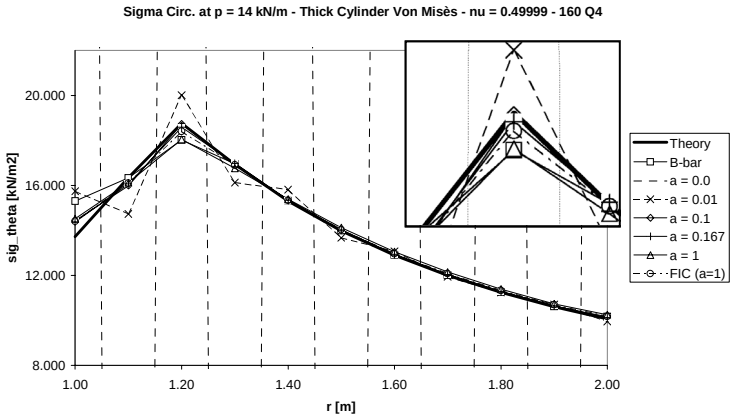



Figure 3-14: Radial stress comparison for $p = 14\ kN/m$

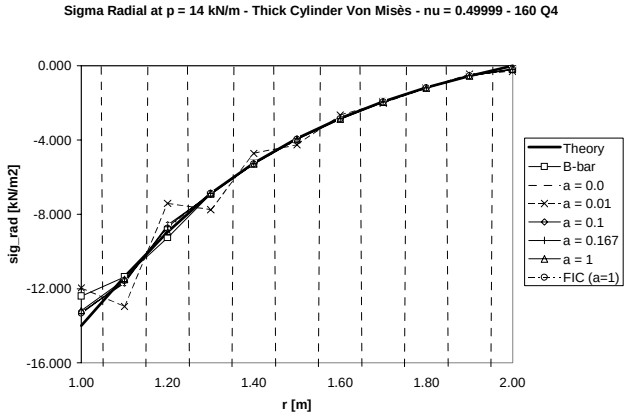



Figure 3-15: Radial stress comparison for $p = 14\ kN/m$

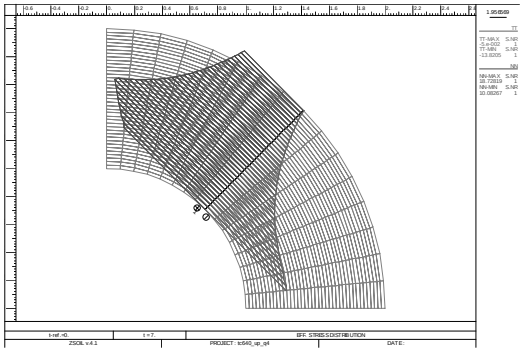


Figure 3-16: 640 Q4 mesh, radial and circ. stresses for $p = 14\ kN/m$

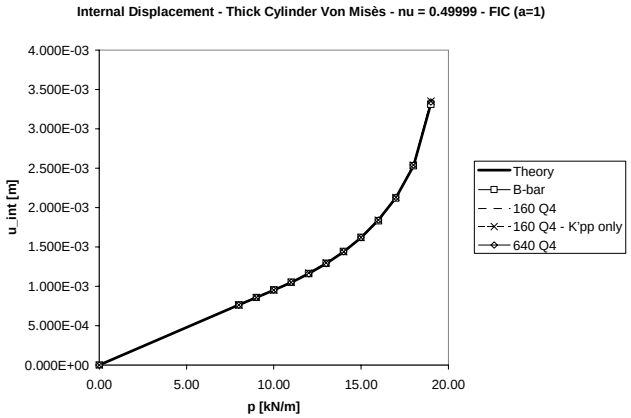



Figure 3-17: Radial displacement comparison

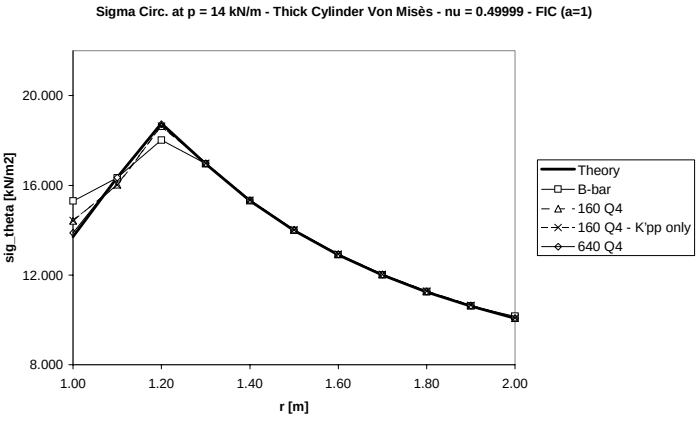



Figure 3-18: Circ. stress comparison for $p = 14\ kN/m$

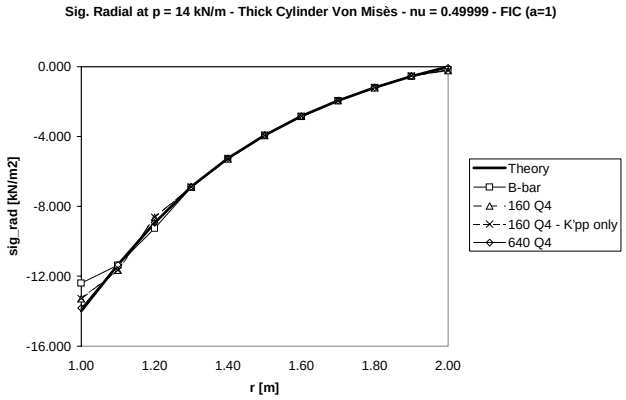



Figure 3-19: Radial stress comparison for $p = 14\ kN/m$

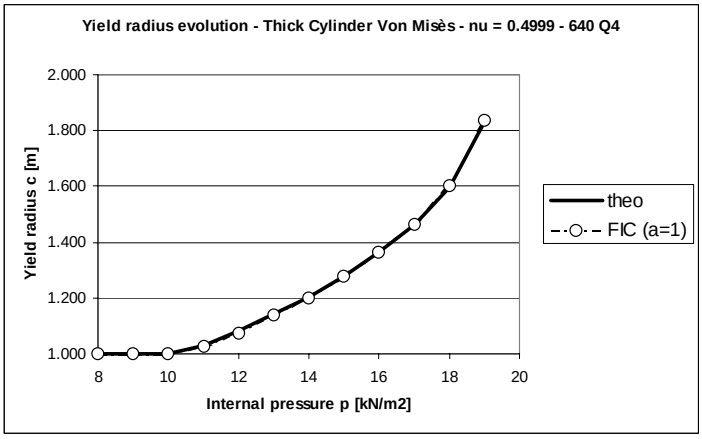



Figure 3-20: Evolution of yield radius

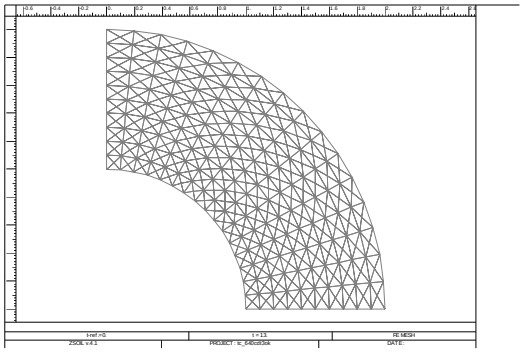


Figure 3-21: 640 "cross-diagonal pattern" T3 mesh

### Triangular Elements

**Cross-Diagonal Pattern** We consider now a 640 T3 elements mesh organized following the so-called cross-diagonal pattern (i.e. each quadrilateral (Q4) has been split into four triangles (T3), see Figure 3-21). The stabilized formulations are compared here with the standard displacement formulation which, in this pattern, is supposed to give acceptable results [39].

**Results Discussion** The $\alpha^e$ parameter has a very little effect in this case. Any value between $\alpha^e = 0.01$ and $\alpha^e = 1$ gives the same results. We can therefore use the same value than for the Q4 element, i.e. $\alpha^e = 0.1$. The standard displacement formulation is able to predict the correct radial displacement and the correct stress fields, as the cross-diagonal pattern is respected.

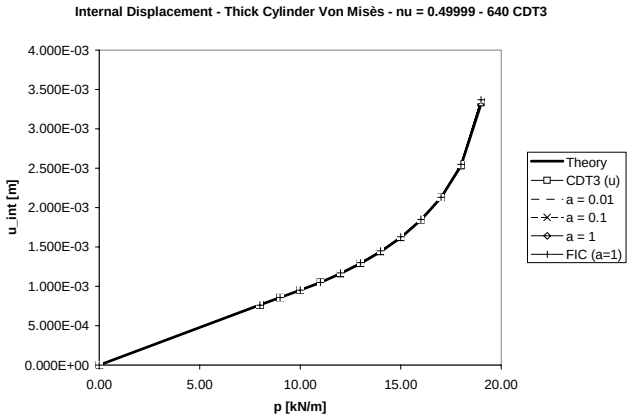



Figure 3-22: Radial displacement comparison

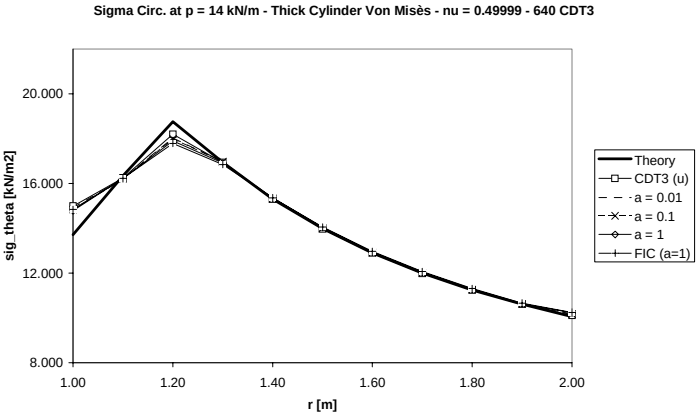



Figure 3-23: Circ. stress comparison for $p = 14\ kN/m$

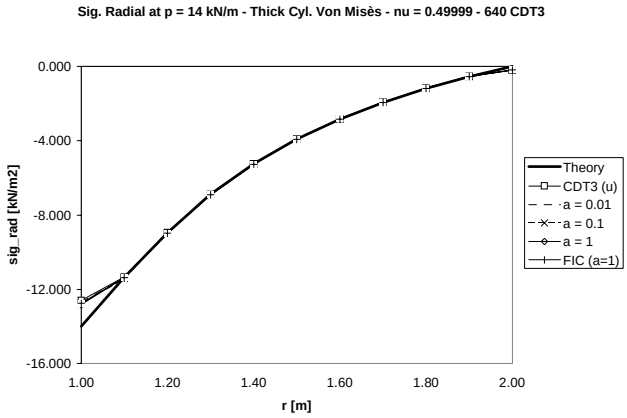



Figure 3-24: Radial stress comparison for $p = 14\ kN/m$

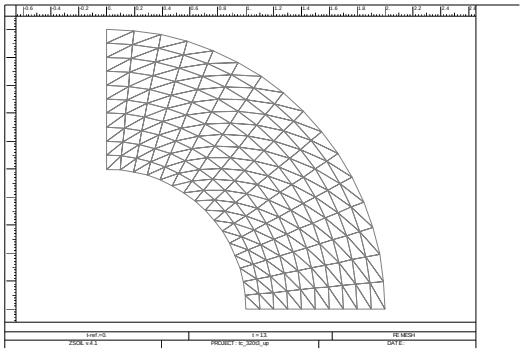


Figure 3-25: 320 T3 mesh

**Other Pattern of T3** Consider now a 320 T3 elements mesh (see Figure 3-25). The pure displacement formulation is compared with the stabilized scalar formulation, as well as with the FIC-scheme formulation.

**Results Discussion** As expected, the pure displacement formulation fails to behave correctly, while the stabilized case works. The $\alpha^e$ parameter has a little more effect in this case. Still, any value between $\alpha^e = 0.01$ and $\alpha^e = 1$ gives approximately the same results. Only if $\alpha^e$ is chosen really too small ($\alpha^e = 1e-10$), then the solution starts to oscillate. We can therefore use the same value as for the Q4 element, i.e. $\alpha^e = 0.1$.

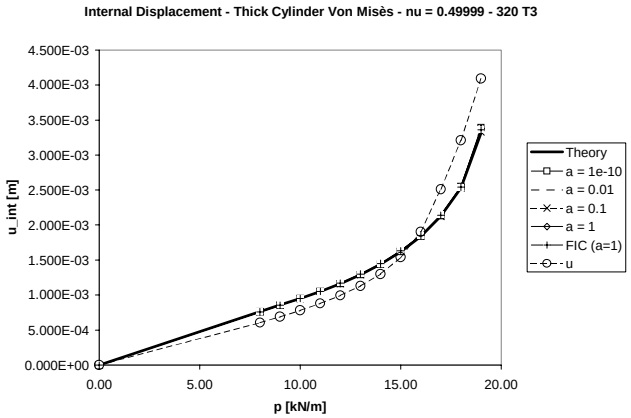



Figure 3-26: Radial displacement comparison

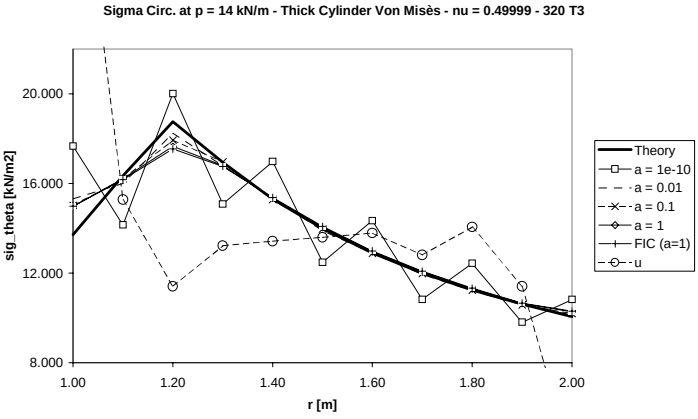



Figure 3-27: Circ. stress comparison for $p = 14\ kN/m$

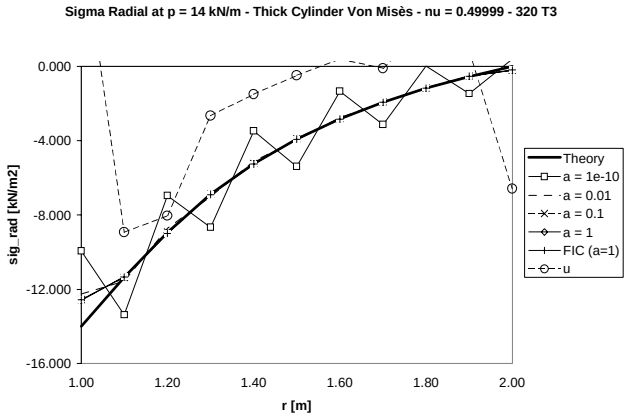



Figure 3-28: Radial stress comparison for $p = 14\ kN/m$

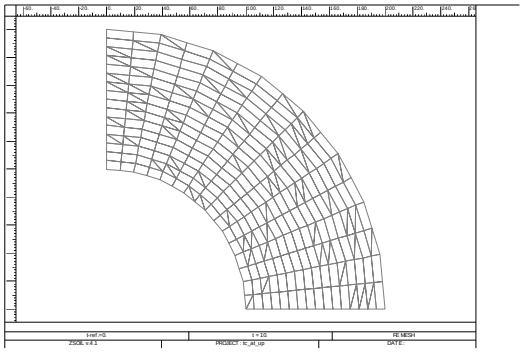


Figure 3-29: Mixed Q4 + T3 mesh

### Mixing Elements

Consider now the same thick cylinder test, but this time the mesh is composed by a mixture of Q4 and T3 (Figure 3-29).

**Uniformity of the Solution** The first criterion that we will scrutinize is the symmetry of the solution. Considering the $\overline{\mathbf{B}}$ solution ($\overline{\mathbf{B}}$ for Q4 elements, and standard T3 elements) given in Figure 3-30, we notice that the displacement for $p = 19\ kN/m$ (last converged step) is not uniform although the problem is symmetric, while the stabilized solution (with $\alpha^e = 0.1$ for both Q4 and T3) is almost uniform and the FIC-scheme solution is perfectly uniform (Figures 3-31 and 3-35).

**Results Discussion** Figures 3-32 - 3-34 show that the scalar version of the stabilized form give a good agreement with the theoretical results. The " $\mathbf{K}'_{pp}$ only" stabilization (LPOS) gives sensibly the same values, as well as the FIC-scheme version.

### Thick Cylinder Test Conclusions

For the scalar version, a value of $\alpha^e = 0.1 \rightarrow \frac{1}{6}$ seems to be reasonable. For the directional (FIC-scheme) version, $\alpha^e = 1$ looks OK. The mixture of elements has brought up the fact that uniformity of the solution is best taken care of by the directional formulation. Figure 3-36 summarizes the benchmarks that have been performed on the thick cylinder test.

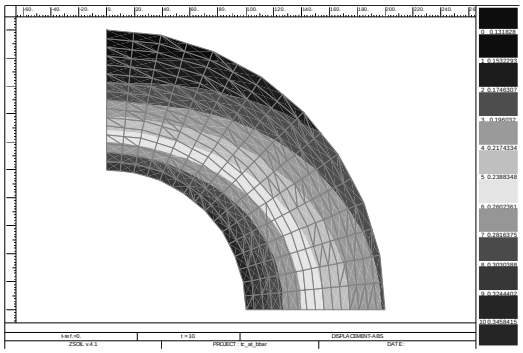


Figure 3-30: Displ. intensities at $p = 19\ kN/m$ Q4($\overline{\mathrm{B}}$) & T3

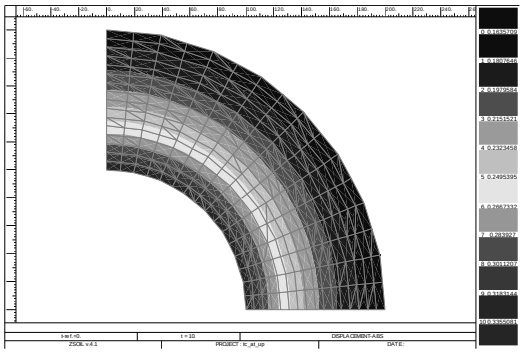


Figure 3-31: Displ. intensities at $p = 19\ kN/m$, $\alpha^e(\mathrm{Q4}) = \alpha^e(\mathrm{T3}) = 0.1$

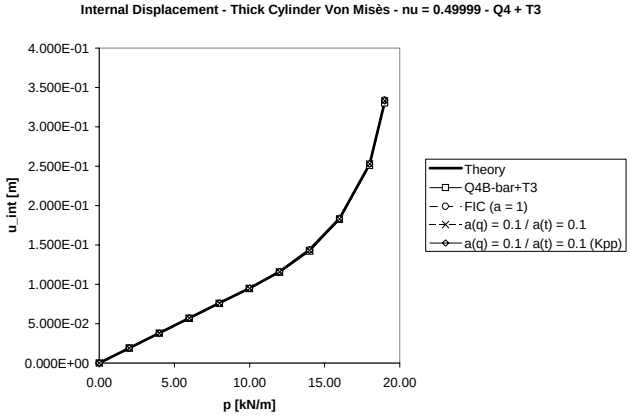



Figure 3-32: Radial displacement comparison

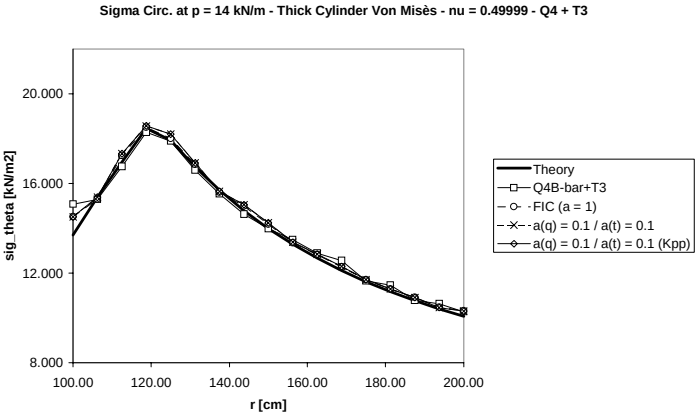



Figure 3-33: Circ. stress comparison for $p = 14\ kN/m$

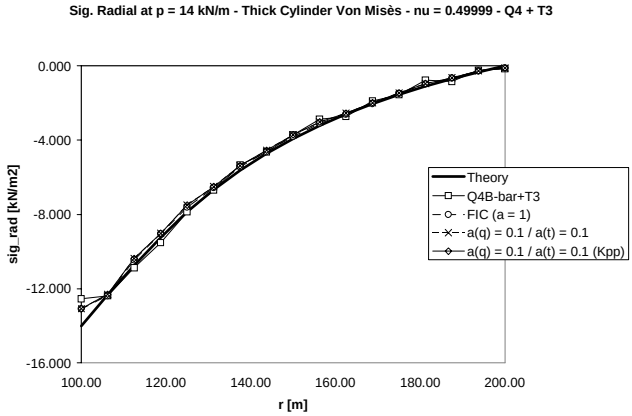



Figure 3-34: Radial stress comparison for $p = 14\ kN/m$

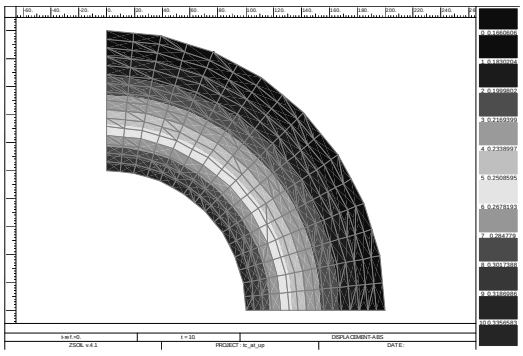


Figure 3-35: Displ. intensities at $p = 19\ kN/m$, FIC-scheme

| **Thick cylinder test** | Q4-$\overline{\mathbf{B}}$ T3-u | mod.-GLS $\alpha^e$ = **0.1** | FIC-scheme $\alpha^e$ = **1** | LPOS $\alpha^e$ = **0.1** |
|---|---|---|---|---|
| □ | OK | OK | OK | OK |
| △ | **FAILS** | OK | OK | OK |
| ⊠ | OK | OK | OK | OK |
| □+△ | **FAILS** (loss of sym.) | OK | OK | OK |

Figure 3-36: Recapitulation of thick cylinder test results for an incompressible material

**Dilatant Case**

It seemed interesting to make an analysis of the thick cylinder in a dilatant case. A Drucker-Prager criterion is used, with $c = 12\ kN/m^2$ and $\phi = \psi = 20^o$. All other characteristics of the material are taken equal to the von Mises case. The analysis is carried out on the 160 Q4 mesh (see Figure 3-12). Figures 3-37 - 3-39 show that all the stabilized versions (scalar with $\alpha^e = 0.1$, FIC-scheme and LPOS) give a good agreement with the reference solution (computed with EAS elements with Z_SOIL.PC [67]). Even The $\overline{\mathbf{B}}$ method gives results which can be considered acceptable (except a small propension to oscillate in the stress results). This can be explained by the fact that this example is not too constrained.

**Remark 34** *Drucker-Prager parameters are $a_\phi$, $a_\psi$ and $k$ (see section 2.2.3). When the Drucker-Prager criterion is defined with Mohr-Coulomb parameters ($c$, $\phi$ and $\psi$), a matching plane strain failure hypothesis leads to the following size-adjustment (see [67]):*

$$\begin{array}{llll} \textit{Deviatoric case } (\psi = 0^o) & k = c\ \cos\phi & a_\phi = \frac{\sin\phi}{3} & a_\psi = 0 \\ \textit{Associated case } (\psi = \phi) & k = \frac{3c}{\sqrt{9+12\tan^2\phi}} & a_\phi = \frac{\tan\phi}{\sqrt{9+12\tan^2\phi}} & a_\psi = a_\phi \end{array} \tag{3.260}$$

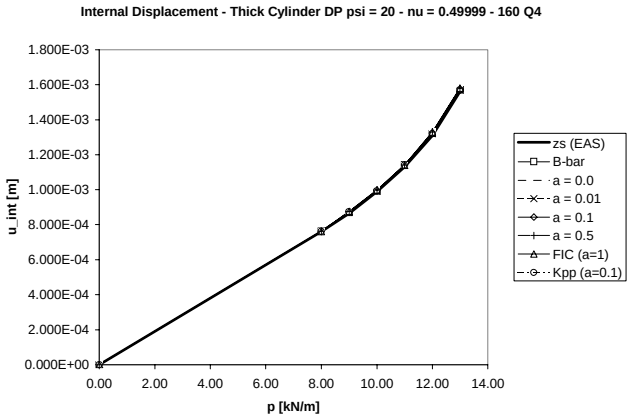



Figure 3-37: Radial displacement comparison (Drucker-Prager, $\psi = 20^o$)

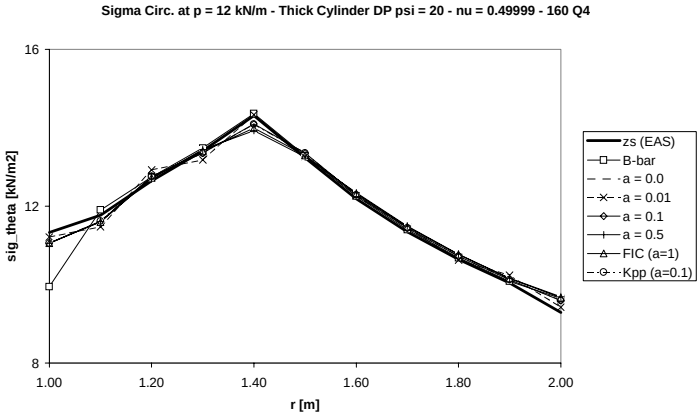



Figure 3-38: Circ. stress comparison for $p = 12\ kN/m$ (Drucker-Prager, $\psi = 20^o$)

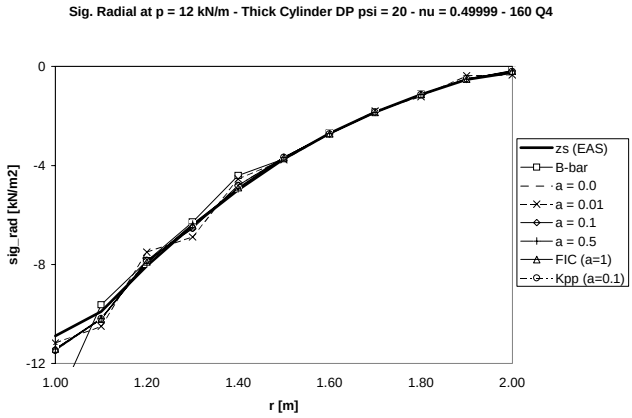



Figure 3-39: Radial stress comparison for $p = 12\ kN/m$ (Drucker-Prager, $\psi = 20^o$)

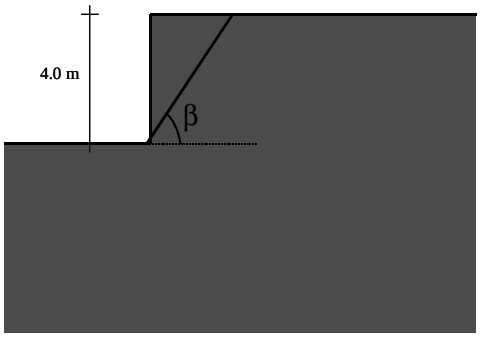



Figure 3-40: Vertical cut geometry

### 3.5.3 Vertical Cut Stability

#### Problem Geometry and Properties

The stability of a vertical cut is analysed now. The properties of the soil are as follows: $E = 10000\ kN/m^2$, $\nu = 0.4$, $\gamma = 20\ kN/m^3$. A Drucker-Prager criterion is used, with $c = 16\ kN/m^2$ and $\phi = 30^o$ (see remark in the previous section for the plane strain failure adjustment). The influence of the dilatancy angle $\psi$ on the solution will be taken into account, with $0^o \leq \psi \leq \phi$.

#### Theoretical Solution

Terzaghi [56] gives an estimation of the critical height of the vertical cut as:

$$H_c = \frac{N_s\ c}{\gamma} \tag{3.261}$$

where $N_s = N_s(\phi, \alpha)$ is a stability coefficient, which can be determined from $\phi$ and $\alpha$ - the angle of the slope (here $\alpha = 90^o$ and $N_s = 6.7$). Therefore, the safety factor reads:

$$SF = \frac{H_c}{H} = \frac{N_s\ c}{\gamma\ H} \tag{3.262}$$

In our case:

$$SF = \frac{6.7 \cdot 16}{20 \cdot 4} = 1.34 \tag{3.263}$$

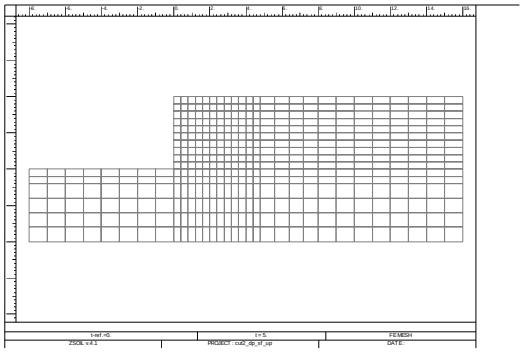


Figure 3-41: Q4 mesh

**Remark 35** *Chen [8] gives another estimation of the safety factor by:*

$$SF = \frac{H_c}{H} = \frac{c}{\gamma} \tan\left(45^o + \frac{\phi}{2}\right) = 1.385$$

The safety factor does not depend on the dilatancy angle, but the failure surface orientation does. Assuming a plane failure surface the most critical orientation is $\beta = 45^o + \frac{\psi}{2}$ [8] where $\beta$ is defined in Figure 3-40.

### Safety Analysis Algorithm

When the failure criterion is defined by parameters $c$ and $\phi$, the following safety analysis algorithm called $c$-$\phi$ **reduction** can be applied (see [67]):

- set a starting value for $SF_n$ (in most cases $SF_n = SF_0 = 1$)
- for each step, compute $c_n = \frac{c}{SF_n}$ and $\tan\phi_n = \frac{\tan\phi}{SF_n}$
- solve the boundary value problem
- $n \leftarrow n + 1$ until divergence occurs
- the last converged step gives the actual safety factor: $SF = SF_{n\ \text{(last converged)}}$

| | | Theory [Chen] | Computation |
|---|---|---|---|
| ψ = 0° | β | 45° | ~ 45° |
| | SF | 1.385 | 1.3 → 1.4 |
| ψ = 30° | β | 60° | ~ 60° |
| | SF | 1.385 | 1.3 → 1.4 |

Figure 3-42: Q4 mesh, scalar stabilization, $\alpha^e = 0.1$

**Quadrilateral Elements**

The mesh is illustrated in Figure 3-41. Two calculations have been performed, both with a value of $\alpha^e = 0.1$ on the scalar stabilized formulation. The first calculation is made with a deviatoric plastic flow ($\psi = 0^o$) and the second with an associated plastic flow ($\psi = 30^o$). Results are summarized in Figure 3-42, and the two failure surfaces (identified through displacement intensities) are illustrated in Figures 3-43 - 3-44. They correspond approximately to predictions with simplified theories (plane failure surfaces).

Computations were then repeated with "$\mathbf{K}'_{pp}$ only" (LPOS), FIC-scheme and $\overline{\mathbf{B}}$ formulations. The "$\mathbf{K}'_{pp}$ only" results match the standard scalar stabilization almost perfectly. The directional (FIC-scheme) formulation has trouble in the early stages of plastification and can't converge for $SF = 1.3$ ($\psi = 0^o$) and for $SF = 1.1$ ($\psi = 30^o$). The $\overline{\mathbf{B}}$ results are satisfactory in the case of incompressible flow ($\psi = 0^o$), but can't predict the correct failure in the dilatant one (numerical instabilities for $SF = 1.1$).

**Triangular Elements**

**Cross-Diagonal Pattern** As for the thick cylinder test, we examine here the performance of the standard triangular element (T3, displacement formulation), with a cross-diagonal pattern (see Figure 3-45 for the mesh). The deviatoric test ($\psi = 0^o$) gives satisfactory results (see Figure 3-46, $SF = 1.3 \rightarrow 1.4$ and $\beta = 45^o$), as well as the dilatant case (see Figure 3-47, $SF = 1.3 \rightarrow 1.4$ and $\beta \simeq 60^o$)

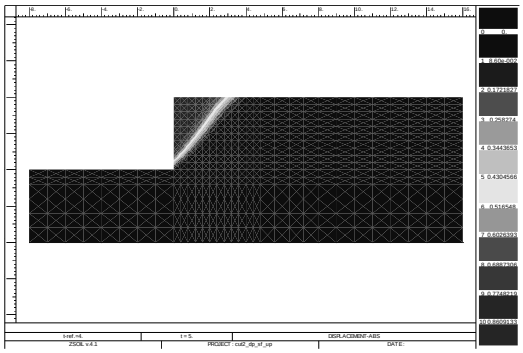


Figure 3-43: Displacement intensities for $\psi = 0^o$ (Q4)

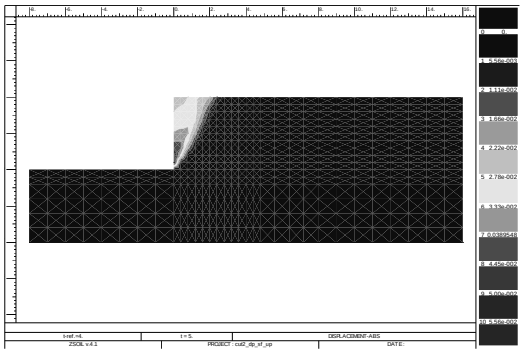


Figure 3-44: Displacement intensities for $\psi = 30^o$ (Q4)

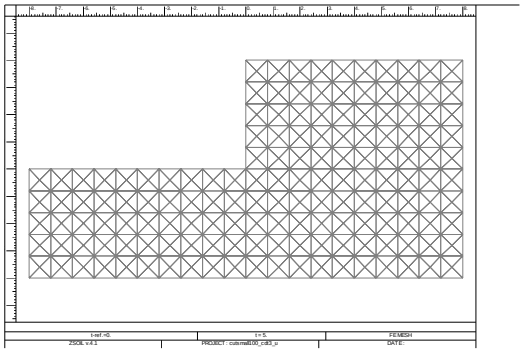


Figure 3-45: Cross-diagonal pattern T3 mesh

**Remark 36** *the orientation of the elements edges in the mesh coïncides with the failure surface in the deviatoric case ($\beta = 45^o$).*

**Other Pattern** As we examine the displacement formulation, non-convergence for $\psi = 30^o$ happens when we do not respect the cross-diagonal pattern of triangular elements (see Figure 3-48). In this case, stabilized mixed T3 elements (scalar formulation with $\alpha^e = 0.1$) are shown to give a satisfactory estimation of the safety factor ($SF = 1.2 \rightarrow 1.3$) as well as the correct failure surface in both the incompressible and the dilatant case (for the latter, see Figure 3-50).

### Mixing Elements

Consider now a mesh composed by a mixture of Q4 and T3 (Figure 3-49). We have seen before the difficulty of displacement formulations to estimate the safety factor in the dilatant case. The same problem happens here as the solution fails to converge for $SF = 1.1$ when we use a mixture of $\overline{\mathbf{B}}$ Q4 and standard T3. On the other hand when we consider a stabilized formulation for both types of elements ($\alpha^e = 0.1$), the correct safety factor ($SF = 1.3 \rightarrow 1.4$) as well as the expected failure surface (see Figure 3-51) are retrieved. The same conclusions can be drawn in the deviatoric case.

### Vertical Cut Stability Conclusions

For the scalar version, a value of $\alpha^e = 0.1$ (for both Q4 and T3) gives satisfactory results. Results are summarized in Figure 3-52.

**Remark 37** *the Laplacian pressure operator scheme ("$\mathbf{K}'_{pp}$ only") stabilization matches quasi perfectly the scalar stabilization.*

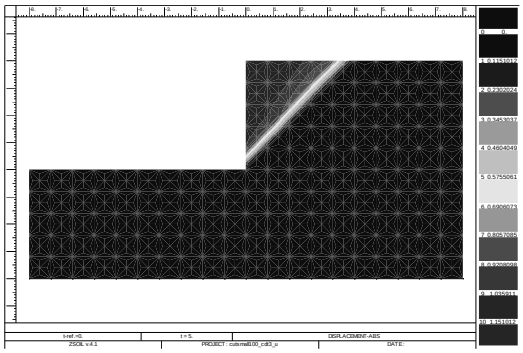


Figure 3-46: Displacement intensities for $\psi = 0^o$ ("cross-diagonal" T3)

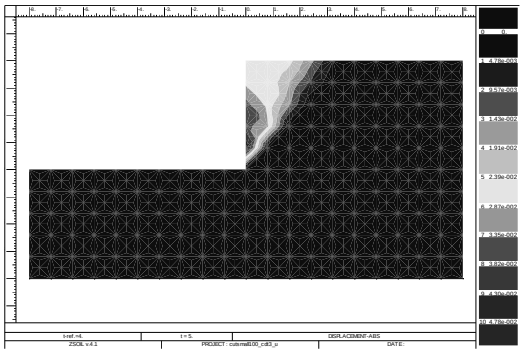


Figure 3-47: Displacement intensities for $\psi = 30^o$ ("cross-diagonal" T3)

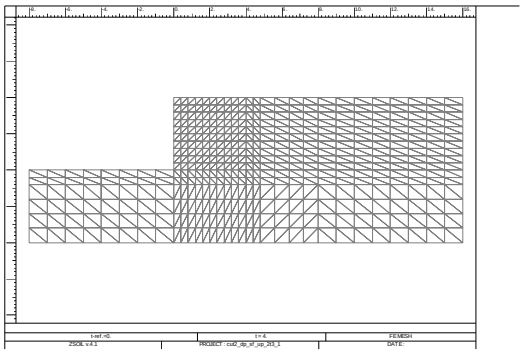


Figure 3-48: T3 mesh (other pattern)

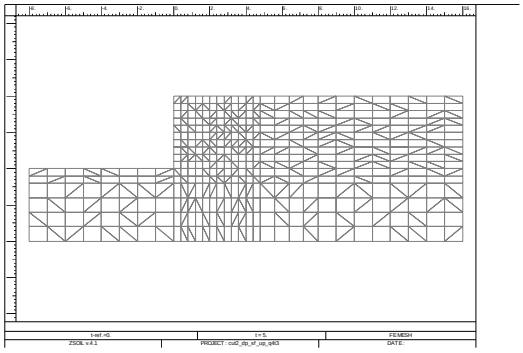


Figure 3-49: Q4 + T3 mesh

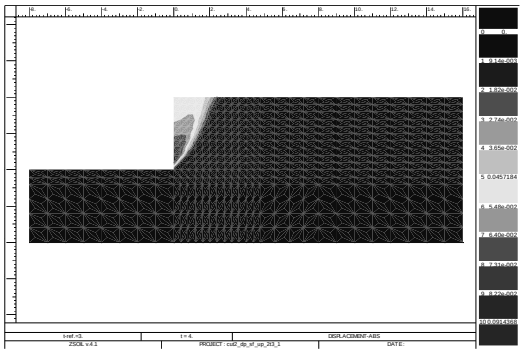


Figure 3-50: Displacement intensities for $\psi = 30^o$ (T3)

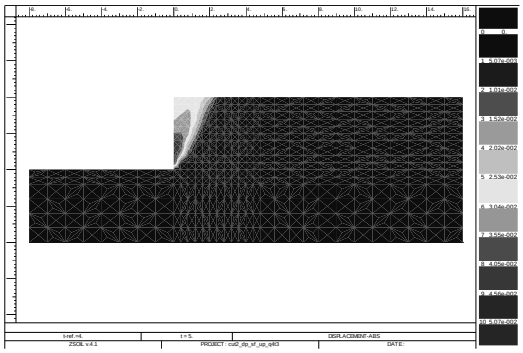


Figure 3-51: Displacement intensities for $\psi = 30^o$ (Q4 + T3)

| | $\psi = 0°$ | | $\psi = 30°$ | |
|---|---|---|---|---|
| **Cut stability analysis** | Q4-$\overline{B}$ T3-u | mod.-GLS $\alpha^e = 0.1$ | Q4-$\overline{B}$ T3-u | mod.-GLS $\alpha^e = 0.1$ |
| □ | OK (SF = 1.3→1.4) | OK (SF = 1.3→1.4) | **FAILS** | OK (SF = 1.3→1.4) |
| △ | **FAILS** | OK (SF = 1.2→1.3) | **FAILS** | OK (SF = 1.2→1.3) |
| ⊠ | OK (SF = 1.3→1.4) | OK (SF = 1.3→1.4) | OK (SF = 1.3→1.4) | OK (SF = 1.3→1.4) |
| □+△ | **FAILS** | OK (SF = 1.3→1.4) | **FAILS** | OK (SF = 1.3→1.4) |

Figure 3-52: Recapitulation of the vertical cut stability results

**Remark 38** *the directional (FIC-scheme) version has some trouble to converge in the early stages of the stability analysis in the deviatoric case (numerical instabilities at the onset of plastification). This difficulty to converge is even greater in the dilatant case.*

**Remark 39** *the under-estimation of the safety factor in the case of triangular elements can be corrected if we use a denser mesh.*

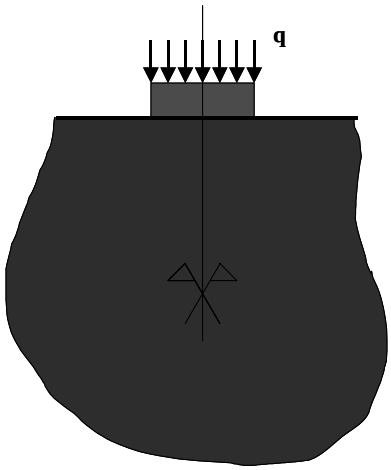



Figure 3-53: Footing geometry

### 3.5.4 Bearing Capacity of a Strip Footing

#### Problem Geometry and Properties

The problem of the bearing capacity of a superficial strip footing is described next. The properties of the soil are: $E = 3000\ kN/m^2$, $\nu = 0.38$, $\gamma = 0$. A Drucker-Prager criterion is used, with a size adjustment matching plane strain failure with the Mohr-Coulomb yield surface (see remark below and [8]). Cohesion $c = 1\ kN/m^2$, friction angle $\phi = 20°$, and dilatancy angle $\psi$ varies between $0°$ and $\phi$. The surface load on the footing is increased gradually: $q = 0 \rightarrow 20\ kN/m$. Rough contact between the soil and the rigid foundation is assumed (the foundation itself is modelized with an elastic material, $E = 1e6\ kN/m^2$, $\nu = 0.2$).

**Remark 40** *Drucker-Prager parameters are $a_\phi$, $a_\psi$ and $k$. When the Drucker-Prager criterion is defined with Mohr-Coulomb parameters ($c$, $\phi$ and $\psi$), a matching plane strain failure hypothesis leads to the following size-adjustment (see [67]):*

$$\begin{array}{llll} \textit{Deviatoric case } (\psi = 0^o) & k = c \cdot \cos\phi & a_\phi = \frac{\sin\phi}{3} & a_\psi = 0 \\ \textit{Associated case } (\psi = \phi) & k = \frac{3c}{\sqrt{9+12\tan^2\phi}} & a_\phi = \frac{\tan\phi}{\sqrt{9+12\tan^2\phi}} & a_\psi = a_\phi \end{array} \tag{3.264}$$

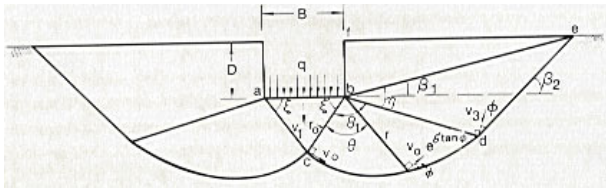



Figure 3-54: Assumed failure mechanism [8]

### Theoretical Solution

Terzaghi [56] gives the following expression, based on a Mohr-Coulomb yield surface, in order to calculate the ultimate load on a strip superficial foundation:

$$q_u = cN_c + \gamma t N_q + \frac{1}{2}\gamma B N_\gamma \tag{3.265}$$

where $t$ is the foundation depth, $B$ the foundation width and $N_c$, $N_q$, $N_\gamma$ are coefficients which are function of $\phi$. $\phi = 20°$ leads to $N_c = 18$, which yields in our case (as body loads are neglected, $\gamma = 0$):

$$q_u = 1 \cdot 18 = 18\ kN/m \tag{3.266}$$

The failure mechanism given by [8] is illustrated in figure 3-54 (in our superficial footing case, $D = 0$).

**Remark 41** *a refined solution by Matar [37] gives a theoretical failure load of* $q_u = 15.6\ kN/m$.

### Reference Solution

The reference solution for the load-displacement (vertical displacement under the footing) curves is obtained with enhanced assumed-strain elements (or EAS) which have been shown [12] to predict the correct failure load for both deviatoric and dilatant plastic flows. This computation has been performed with the Z_Soil [67] software and is identified by "**zs EAS**" in the following figures.

**Remark 42** *in the following load-displacement curves, only the converged points have been drawn and the diverged step has been represented by an arrow.*

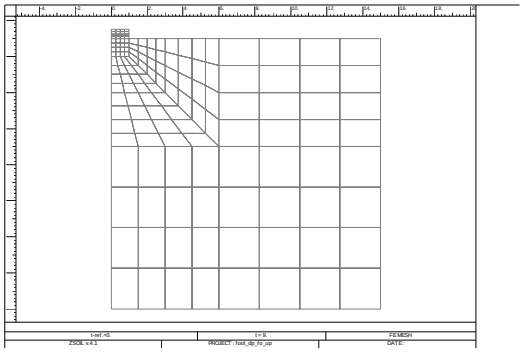


Figure 3-55: Q4 mesh

**Quadrilateral Elements**

The first mesh that we study is composed of quadrilaterals (Q4, Figure 3-55). Only half of the footing is modelized as we have a vertical symmetry plane. Two different plastic flows are examined: the deviatoric one ($\psi = 0°$) and the associated one ($\psi = \phi = 20°$).

In the incompressible case, the $\overline{\mathbf{B}}$ solution gives a correct failure load, while the scalar stabilized formulation gives satisfactory results for $\alpha^e = 0.1$ and even a better correspondance with the reference solution with $\alpha^e = 1$. The directional formulation (FIC-scheme) does not improve the results, but still converges to the same failure load ($q_u = 18\ kN/m$, last converged step). The " $\mathbf{K}'_{pp}$ only" (LPOS) stabilization also results almost to the same load-displacement curve (see Figure 3-58). The failure mechanism (identified from displacement intensities) is given on Figure 3-56 and can be compared with Figure 3-54.

In the dilatant case, the same kind of results are obtained, except this time the $\overline{\mathbf{B}}$ solution can't get to the failure load. Again the stabilized solution with $\alpha^e = 0.1 \rightarrow 1$ gives a satisfactory answer ($q_u = 18\ kN/m$, see Figure 3-59), although the failure load can't be as clearly identified as in the deviatoric case (even for the reference EAS solution). The failure mechanism is given in Figure 3-57.

**Triangular Elements**

The "cross-diagonal pattern" T3 (Figure 3-60) displacement-only solution gives a reasonably good approximation of the failure load in both the incompressible and the dilatant case ($q_u = 18\ kN/m$).

As for the second mesh (T3, Figure 3-61), the scalar stabilized formulation gives satisfactory results for $\alpha^e = 0.1$ and $\alpha^e = 1$. The failure mechanism is identified on Figure 3-62. The directional formulation (FIC-scheme) does not improve the results, and even has trouble to

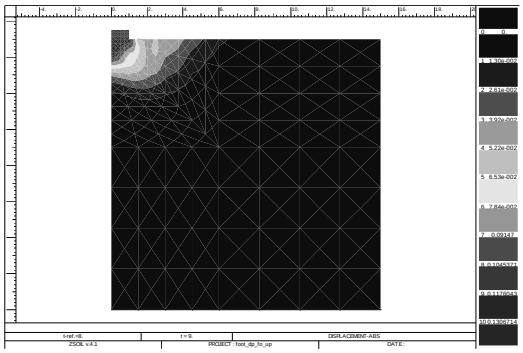


Figure 3-56: Displacement intensities , $\psi = 0^o$ (Q4, $\alpha^e = 1$)

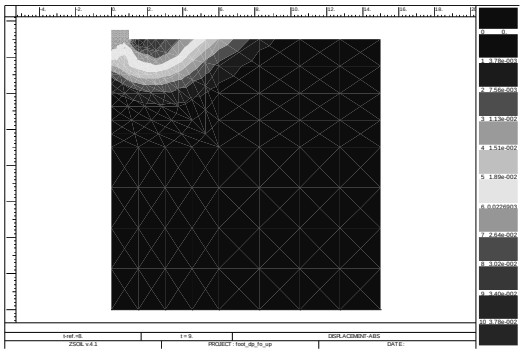


Figure 3-57: Displacement intensities , $\psi = 20^o$ (Q4, $\alpha^e = 1$)

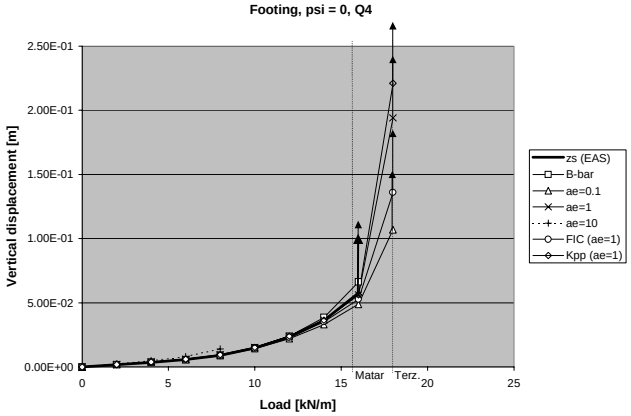



Figure 3-58: Load-displacement curve, $\psi = 0^o$ (Q4)

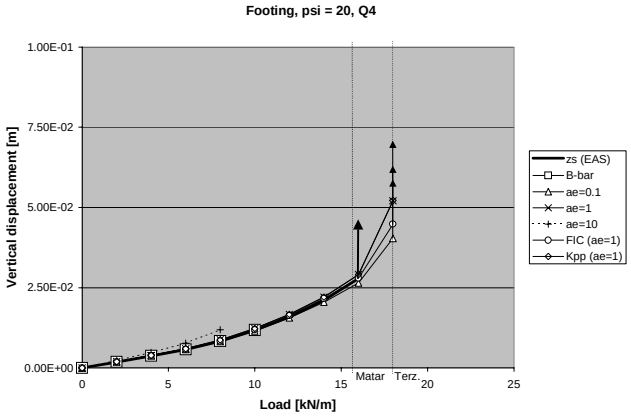



Figure 3-59: Load-displacement curve, $\psi = 20^o$ (Q4)

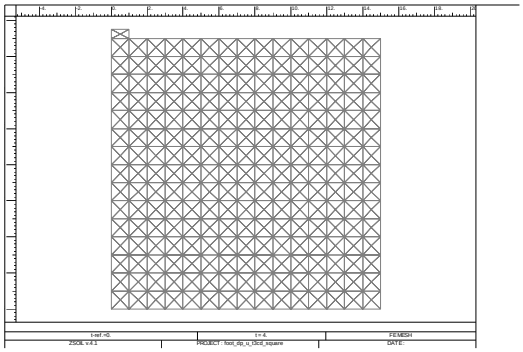


Figure 3-60: Cross-diagonal pattern T3 mesh

converge with $\alpha^e = 1$ (see Figure 3-63).

In the dilatant case, the same kind of results are obtained. Again the stabilized solution with $\alpha^e = 0.1 \rightarrow 1$ gives a satisfactory answer ($q_u = 18\ kN/m$, see Figure 3-65) and the failure mechanism is given in Figure 3-64. It can be noticed that the ultimate load can't be as clearly identified as in the incompressible case. The FIC-scheme formulation again has trouble to converge after $q = 10\ kN/m$.

### Mixing Elements

Considering now another mesh composed by a mixture of elements (Q4 + T3, Figure 3-66). In the incompressible case, the scalar stabilized formulation gives again good results for $\alpha^e = 0.1$. The directional formulation (FIC-scheme) does not improve the results, but converges to the same failure load ($q_u = 18\ kN/m$, see Figure 3-69). The failure mechanism is identified on Figure 3-67.

In the dilatant case, the same kind of results are obtained, the scalar stabilized solution with $\alpha^e = 0.1$ gives a satisfactory answer ($q_u = 18\ kN/m$, with the failure mechanism illustrated in Figure 3-68), but the FIC-scheme solution fails to identify clearly a divergence (see Figure 3-70).

### Footing Conclusions

The scalar stabilized formulation (or modified-GLS) is able to handle all cases (incompressible and dilatant, Q4, T3 and Q4+T3); the tuning of the stabilization parameter results in $\alpha^e = 0.1 \rightarrow 1$. The directional (FIC-scheme) formulation has more trouble to converge. Both the $\overline{\mathbf{B}}$ quadrilateral and the cross-diagonal triangles are useful in the incompressible case, but fail to converge or to predict the correct failure load in the dilatant case.

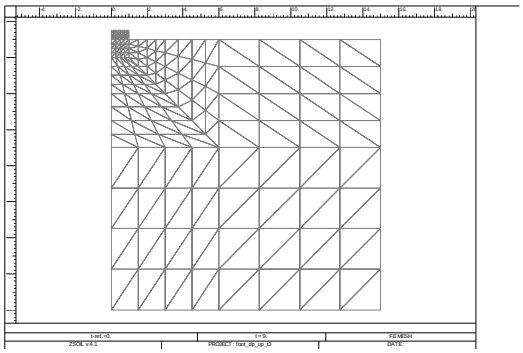


Figure 3-61: T3 mesh

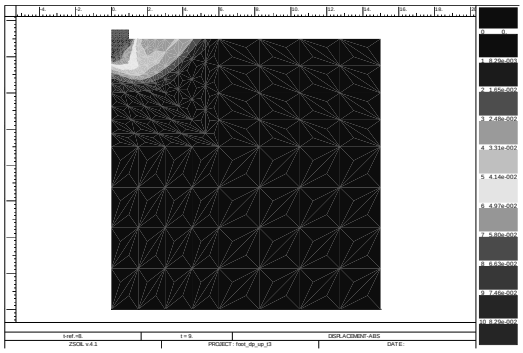


Figure 3-62: Displacement intensities , $\psi = 0^o$ (T3, $\alpha^e = 1$)

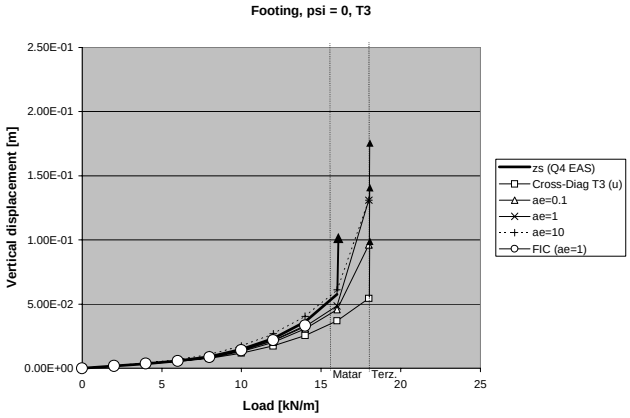



Figure 3-63: Load-displacement curve, $\psi = 0^o$ (T3)

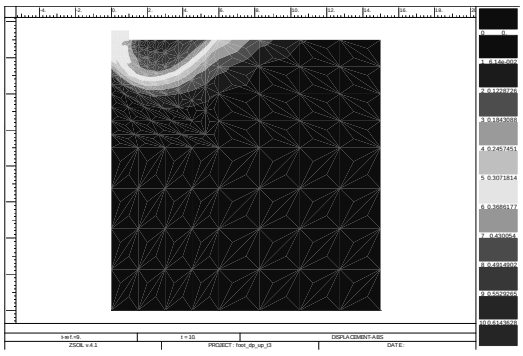


Figure 3-64: Displacement intensities , $\psi = 20^o$ (T3, $\alpha^e = 1$)

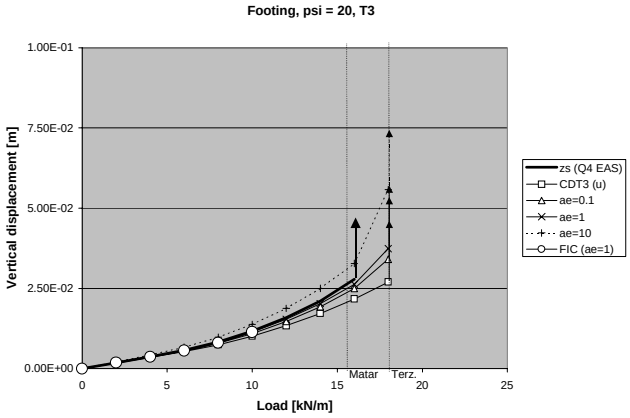



Figure 3-65: Load-displacement curve, $\psi = 20^o$ (T3)

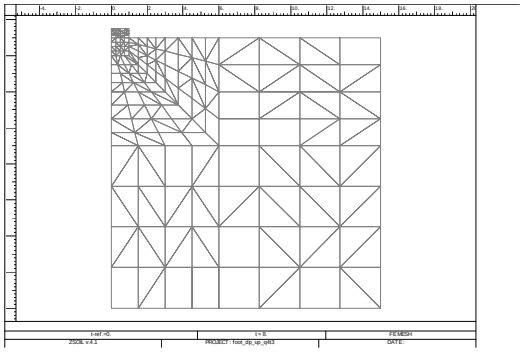


Figure 3-66: Q4 + T3 mesh

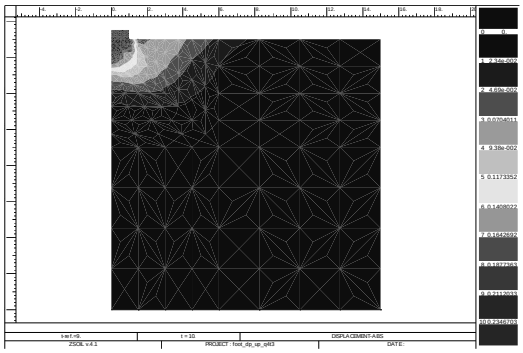


Figure 3-67: Displacement intensities , $\psi = 0^o$ (Q4 + T3, $\alpha^e = 0.1$)

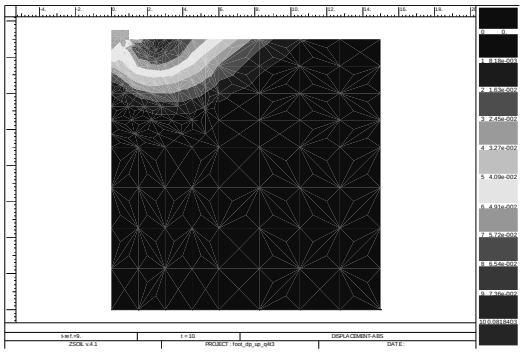


Figure 3-68: Displacement intensities , $\psi = 20^o$ (Q4 + T3, $\alpha^e = 0.1$)

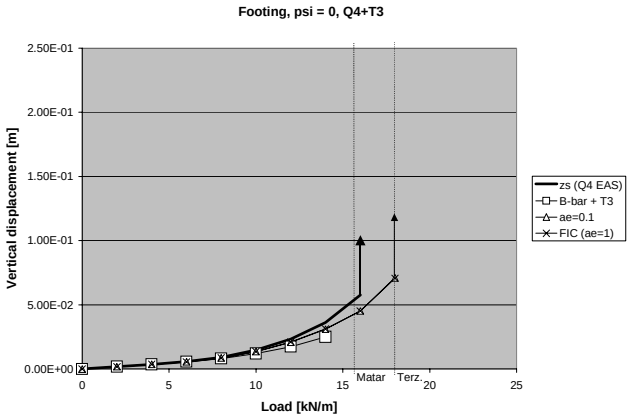



Figure 3-69: Load-displ. curve, $\psi = 0^o$ (Q4+T3)

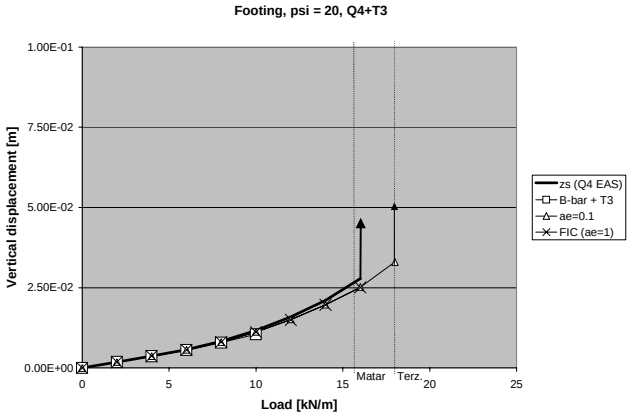



Figure 3-70: Load-displ. curve, $\psi = 20^o$ (Q4+T3)

| | ψ = 0° | | | ψ = 20° | | |
|---|---|---|---|---|---|---|
| **Footing bearing capacity** | Q4-$\overline{B}$ T3-u | mod.-GLS $\alpha^e$ = 0.1 | FIC-scheme $\alpha^e$ = 1 | Q4-$\overline{B}$ T3-u | mod.-GLS $\alpha^e$ = 0.1 | FIC-scheme $\alpha^e$ = 1 |
| □ | OK | OK | OK | **FAILS** | OK | OK |
| △ | **FAILS** | OK | Trouble to converge | **FAILS** | OK | Trouble to converge |
| ⊠ | OK | - | - | OK | - | - |
| □+△ | **FAILS** | OK | OK | **FAILS** | OK | Trouble to converge |

Figure 3-71: Recapitulation of the footing bearing capacity tests

Finally the failure mechanism is retrieved in every case for a failure load $q_u = 18\ kN/m$, matching the Terzaghi prediction but overshooting the EAS solution ($q_u = 16\ kN/m$) by about 10 %.

**Remark 43** *the failure mechanism in the dilatant case seems to mobilize more soil than in the incompressible case.*

Figure 3-71 summarizes the benchmarks that have been performed on the footing test.

**Remark 44** *the LPOS stabilization gives the same results as the scalar stabilization.*

### 3.5.5 Conclusions and Recommendations

Regrouping the conclusions drawn in the three benchmarks cases, we can observe that:

- the scalar formulation (modified-GLS) seems more robust than the directional one (the FIC-scheme has trouble to converge and sometimes oscillates)
- a good guess for the stabilization parameter is $0.1 \leq \alpha^e \leq 1$
- there is no noticeable difference between the scalar formulation and the " $\mathbf{K}'_{pp}$ only" stabilization described by Pastor [44] and Oñate (LPOS) [42]
- $\overline{\mathbf{B}}$ quads are useful in the incompressible case, but fail to predict the correct behaviour close to the ultimate load in the dilatant case

I would therefore recommend the following type of stabilizing terms (which have to be added to the mixed formulation). This stabilized formulation should be valid for both deviatoric and dilatant plastic flow as well as for incompressible elasticity, on any kind of mesh composed by (bi)linear elements (Q4, T3 or Q4+T3):

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \overrightarrow{\nabla} \bar{\mathbf{N}}^T \boldsymbol{\tau} \left( \mathbf{L}^T \boldsymbol{\sigma} \left( \mathbf{u}^h, p^h \right) + \mathbf{f} \right) d\Omega \tag{3.267}$$

with:

$$\boldsymbol{\tau} = \frac{\alpha^e \left(h^e\right)^2}{2\mu} \mathbf{I} \tag{3.268}$$

where $\mathbf{I}$ is the $n_{sd} \times n_{sd}$ identity matrix and:

$$0.1 \leq \alpha^e \leq 1 \tag{3.269}$$

**Remark 45** *in the last two benchmarks, i.e. the cut stability and the footing bearing capacity, a restart with a smaller time step could be performed in order to get a finer approximation of the theoretical solution. This has not been done here as our purpose was to run a great number of analyses to be able to compare formulations and calibrate the stabilization parameter.*

**Remark 46** *a number of cases have shown that the solution of these three benchmarks does not depend on the element size. Moreover, systematic comparison of four different element patches (quadrilaterals, triangles, mixture of Q4 and T3, cross-diagonal triangles) confirm this fact.*

# Chapter 4

# Implementation

In this section, the numerical implementation of the stabilized methods described in this work is reviewed. First, a brief recall of object-oriented techniques is done, then the code itself is described in four steps:

- hierarchy of classes
- flowchart illustrating how the code works
- in-depth description of the most important classes
- application to a stabilized quadrilateral element (main methods)

## 4.1 Introduction: Object-Oriented Aspects

Object-oriented programming (see e.g. [14] and references therein) has proven in recent years to be one of the easiest, fastest and most efficient ways to program robust scientific software. The basic components of the finite element method, like the node, the element, the material, can easily be fitted into an objects world, with their own behavior, state and identity. We review here the key features of object-oriented programming:

- **inheritance and polymorphism**: in the hierarchy of classes, every object is an instance of a class. A class is an abstract data type which can be considered as the mold of the object. Classes are organised within a hierarchy (class-subclass), which allows a subclass (say, `Quad_U`) to inherit the methods and variables from its superclass (say, `Element`). Polymorphism expresses the fact that two different classes will react differently (in their own manner) to the same message. For instance, the message `myElement→giveBMatrix()` will be interpreted differently by an object of the class `Quad_U` (defining quadrilateral elements) and an object of the class `Triangle_U` (defining triangular elements). The hierarchy of classes of the nonlinear finite element code which has been used to test the stabilized formulations described in this work is given in the next subsection. It provides

a fast overview of the entire software. The fact that the code can be described in such a compact way can be very valuable, when extensions are considered.

- **robustness and modularity: encapsulation of data**: an object is a device capable of performing predefined actions, such as storing information (in its variables), executing tasks (through its methods), or accessing other objects (by sending messages). Variables describe the state of the object, while methods define its behavior. Objects hide their variables from other components of the application. For instance, class `Element` does not have direct access to its Young Modulus. The Young Modulus is stored in class `Material`. The object has to send a message, like `myMaterial→giveYoungModulus()` to access it.

- **non-anticipation and state encapsulation**: non-anticipation expresses the fact that the content of a method should not rely on any assumption on the state of the variables. Strict obedience to non-anticipation will contribute significantly to code robustness.

- **efficiency**: as far as numerical performance is concerned, languages such as C++ have shown performances similar to Fortran. With respect to code development speed using object-oriented techniques, the programmer can maximize reusability of the software and «program like he thinks», which leads to faster prototyping.

## 4.2 Code Description

### 4.2.1 Preamble

The code has been written in C++, built on the original FEM_Object environment dealing with linear elasticity [14]. A first step consisted in extending the capabilities of the software to nonlinear analysis ([10], following [38]). Then mixed formulations and stabilized approaches have been added. In Figure 4-1, the internal circle represents the original FEM_Object code developped at the LSC, the intermediate deals with the extension to nonlinear analysis and finally the external circle regroups the developments necessary to handle stabilized formulations applied to soil plasticity.

**Remark 47** *the original FEM_Object deals with linear statics and dynamics, however the study of stabilized fomulations has been restricted to static analysis in this work. Therefore, in the following, only the static aspects of the code will be discussed.*

### 4.2.2 Hierarchy of Classes

As been said before, object-oriented programming organises classes (and therefore objects) hierarchically. Among other benefits, this hierarchy provides a fast overview of the code for a newcomer. Figure 4-2 illustrates the hierarchy of classes in our case. Bold classes will be scrutinized in the following subsection. A full description of the other classes, necessary to handle finite element calculations but not relevant to the discussion of stabilized methods can be found in [10].

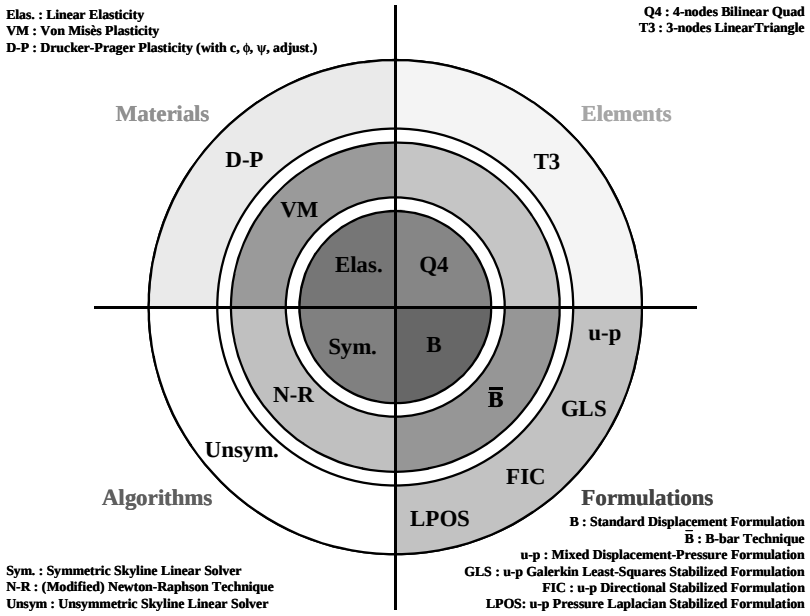



Figure 4-1: C++ code evolution

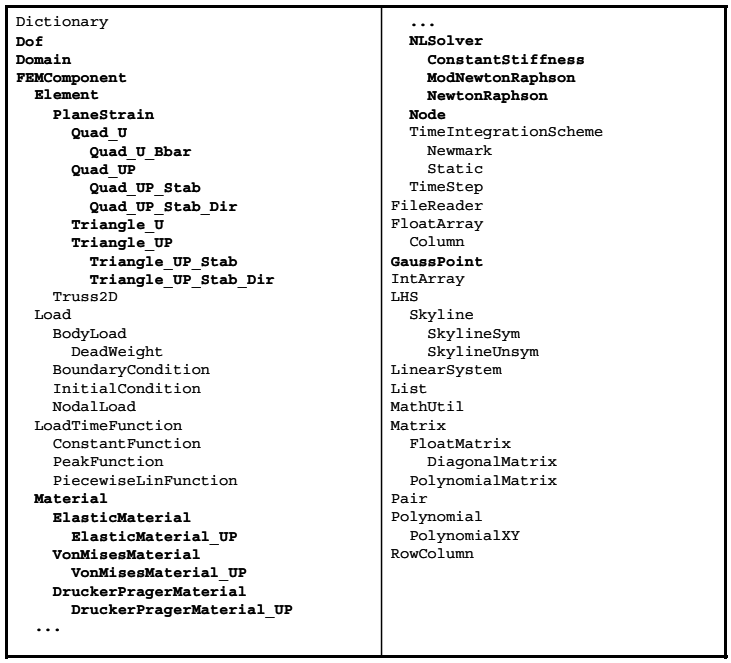



Figure 4-2: Hierarchy of Classes

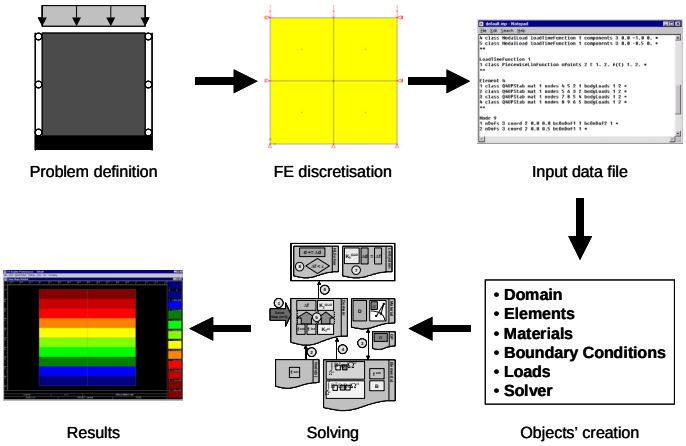



Figure 4-3: From problem to solution

### 4.2.3 How Does it Work?

#### Solving a Boundary Value Problem

Figure 4-3 illustrates the steps which lead from a boundary value problem definition to its solution. The first step is to define the problem that we want to solve: geometry, applied loads, materials, boundary conditions... Then, the finite element discretization is made, and an input file is created. This file contains all the necessary informations for the mechanical part of the software. Up to this point, there is no noticeable difference between object-oriented and procedural codes. The object-oriented code then creates objects from the data file: a global object called **Domain** which is a representation of the problem to be solved, and a number of other objects like **Nodes**, **Elements**, **Materials**, **Loads**, ... Then, the actual finite element problem can be solved through interactions between these objects (or classes). These interactions are described next. Finally results are stored in output files and a graphical postprocessor allows us to represent quantities such as displacements, stress intensities, ...

#### Interactions Between Classes

Say we have reached equilibrium at step $n$, and now we want to solve the problem for step $n+1$. As we deal with a nonlinear case, the process is iterative. Figure 4-4 illustrates the interactions between the objects which will lead us to the solution. The following steps can be identified:

1. A message is sent to class `Domain`, asking him to solve step $n+1$. `Domain` then asks its `Nodes` and `Elements` to compute their elemental contributions.
2. `Nodes` return their force vectors to `Domain`.
3. `Elements` interact with their `Material` and `GaussPoints` to compute a stress increment associated with the current accumulated increment of displacement. `Material` also computes a constitutive matrix.
4. `Elements` return a body force vector, an internal force vector and a tangent stiffness matrix to `Domain`.
5. `Domain` assembles all elemental contributions and forms the left-hand side and the right-hand side of the problem.
6. `Domain` asks the solver to take care of the computation.
7. `LinearSystem` solves the problem, which solution is a new displacement increment.
8. `NonlinearSolver` updates the value of the accumulated displacement increment and tests the convergence of the step, i.e. if internal and external forces are in equilibrium. If they are, the step is ended, otherwise a new iteration is necessary.

### 4.2.4 Main Classes Description

**Remark 48** *Appendix A regroups the list of attributes and methods for all the classes described in this section.*

#### Domain

The domain can be considered as the "main" object that contains all the problems' components; there is a single instance of `Domain` for each problem, and its principal tasks are:

- receiving the messages from the user and initiating the corresponding operations
- storing the components of the mesh: the list of nodes, elements, materials, loads, ...
- storing the non-linear solver type (class `NLSolver`) and the time-integration scheme
- providing objects with access to the data file

#### Components

This class, which is the superclass of classes `Element`, `Load`, `LoadTimeFunction`, `Material`, `NLSolver`, `Node`, `TimeIntegrationScheme` and `TimeStep`, regroups the attributes and methods which are common to all its subclasses (mainly access to the domain or to the input file).

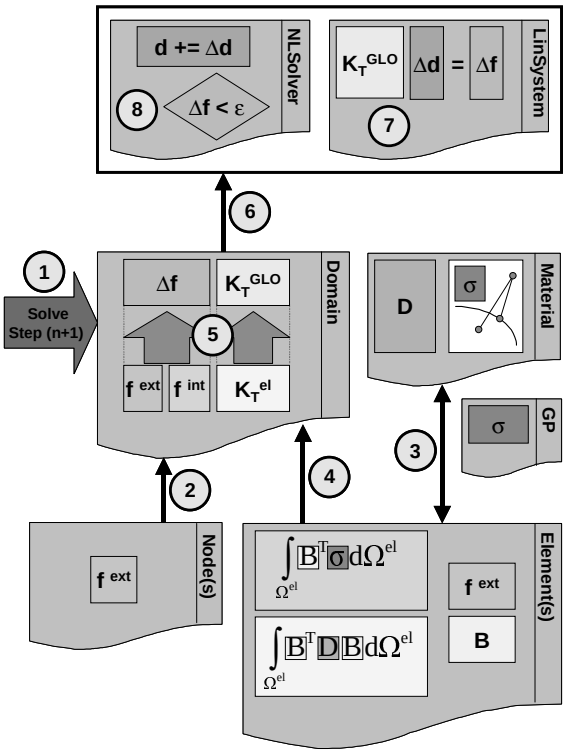



Figure 4-4: Interactions between classes

**Node**

A node is the attribute of one or more elements. It has the following four tasks:

- returning its coordinates
- managing (creating and storing) its degrees of freedom (class `Dof`)
- computing and assembling its nodal load vector (class `NodalLoad`)
- updating its attributes at the end of each step

**Degree of Freedom**

A degree of freedom is an attribute of a node. Its role is to relieve the node from the following tasks:

- managing the unknowns (like $u_i$ and, in the case of a mixed formulation, $p$)
- equation numbering
- checking the existence of a boundary condition if any, and storing its number

**Elements**

It regroups the attributes and the methods which are common to every element (which are instances of classes `Quad_U`, `Quad_U_BBar`, `Quad_UP`, `Quad_UP_Stab`, ...). Its main tasks are:

- calculating its stiffness matrix and its load vector
- giving its contributions to the left-hand side and the right-hand side of the system (through the assembly operation performed by class `Domain`)
- reading, storing and returning its data

Figures 4-5 - 4-6 schematically illustrate the two most important tasks of an element in a nonlinear code: forming its contribution to the left-hand and to the right-hand side (with the help of its Gauss points and material). We will show in the next section the methods which carry out these computations in the case of a stabilized direcional formulation.

Class `PlaneStrain` provides a generic superclass for plane strain elements (Q4 quadrilaterals and T3 triangles).

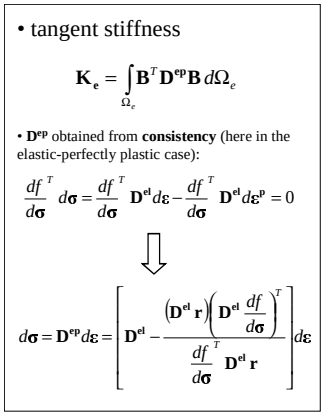



Figure 4-5: Elemental left-hand side

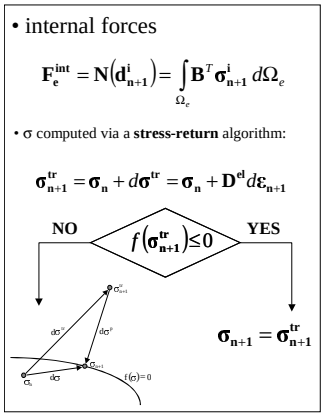



Figure 4-6: Elemental internal force vector (contribution to the right-hand side)

### Materials

Superclass `Material` was created in order to regroup common tasks for its subclasses. Usually, a material is an attribute of many elements of the mesh. The constitutive information is stored in this class. In this code, three kinds of materials are implemented:

- elastic
- elastic-perfectly plastic with a von Mises yield surface
- elastic-perfectly plastic with a Drucker-Prager yield surface

For each of these materials, two classes are implemented: a first one to handle the pure displacement formulation, and a subclass to implement the mixed displacement-pressure approach. Apart from returning their physical properties, they also perform two important tasks which help to return elemental contributions to the global system:

- computing the current stress state through a stress return algorithm
- building a (tangent) constitutive matrix

### Gauss Point

A Gauss point is an attribute of an element. Its task is to regroup the data which are specific to the Gauss point: the coordinates and the weight of the point in numerical integration, the strains, the stresses. Nonlinear analysis induces some special tasks for the Gauss point. It has to store the stress and strain state at the current iteration, but also remember the last converged stress state. It also manages the amplitude of the stress return (through $\Delta\gamma$), the state of the point (elastic or plastic), as well as the stress level.

### Nonlinear Solvers

Class `NLSolver`, which is the superclass of classes `ConstantStiffness`, `ModNewtonRaphson` and `NewtonRaphson`, implements a nonlinear solver. Its main task is to solve the nonlinear problem at each iteration and each step (see Figure 4-7 for a 1D illustration of the Newton-Raphson method). The convergence (or divergence) of the iterative process is also checked in this class. The type of left-hand side depends on the type of algorithm which is defined by one of its three following subclasses (see Figure 5-25 in Appendix A).

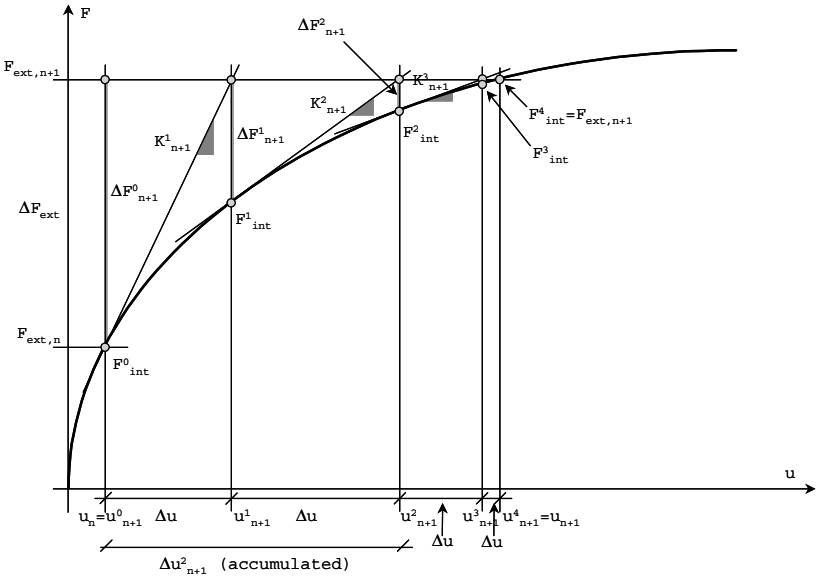



Figure 4-7: 1D illustration of the Newton-Raphson method

### 4.2.5 Building the LHS and the RHS for a Stabilized Q4 (FIC-scheme)

#### Motivation

It is interesting to take a close look at some of the methods which are directly in relation with the stabilized formulation that we have described in chapter 3. We will therefore examine methods which help to the construction of elemental contributions for the stabilized quad (directional (FIC-scheme) formulation). The listing of these methods can be found in Appendix A. The nonlinear solver handles the iterative solving process (see Figure 4-8 for the standard displacement case) with its `NLSolver::Solve()` method (see Figure 4-9). In this method we identify `domain→ComputeJacobian(dxacc)` which asks the domain to form its left-hand side regrouping the elemental contributions, and `domain→ComputeRHSAt(dxacc)` which asks the domain to build its right-hand side. In its turn, the domain asks all the elements which compose the mesh to build up their contributions to both sides of the system (through methods `Element::ComputeStaticLhsAt(stepN, dxacc)` and `Element::ComputeStaticRhsAt(stepN, dxacc)`).

#### Building the Elemental LHS

`Domain` triggers the LHS building with the message: `element→Compute StaticLhsAt(stepN, dxacc)`; the method illustrated in Figure 5-26 is therefore called. In this method, the first step is to extract the elemental displacement increment vector `dElem` from the global accumulated displacement increment `dxacc`. Then, method `Element::computeTangent StiffnessMatrix(dElem)` is activated. This method is virtual, which means that each subclass of class `Element` has to implement its own version. Figures 5-27 - 5-28 illustrate this method for class `Quad_UP_Stab_Dir`. A loop over Gauss points is made for numerical integration. In this loop, the strain-displacement matrix $\mathbf{B}$ and the constitutive operator $\mathbf{D}$ are computed ($\mathbf{D}$ corresponds here to $\overline{\mathbf{D}}^{uu}$ and is retrieved by the element's material). Then the terms which fill up the tangent stiffness matrix are built. `Material` also returns matrices $\overline{\mathbf{D}}^{up}$, $\overline{\mathbf{D}}^{pu}$ and $\overline{\mathbf{D}}^{pp}$, using consistency and depending on the type of plastic surface. Stabilizing terms are then added to the left-hand side. Different flags handle the cases if we include the displacement part in the weighting function or not, or if we use "$\mathbf{K}'_{pp}$ only" (LPOS) stabilization (`pastorFlag`). Usually, as we have seen before, only the pressure part ($\mathbf{G}_p$) is included in the weighting term. Computation of $\mathbf{G}_p$ is illustrated in Figure 5-29. The last ingredient is the computation of the stabilization factor, split in two methods (see Figures 5-30 - 5-31). In the first method the scalar part of the stabilization factor is built, and in the second directional aspects are taken into account, according to what has been derived in the theoretical section.

#### Building the Elemental RHS

The first method called by `Domain` is `Element::ComputeStaticRhsAt(stepN, dxacc)` (pictured in Figure 5-32). This method first handles imposed displacements, then builds up the right-hand side from the external and the internal force vector (illustrated in Figure 5-33), as well as the two different residuals ($R_\theta$ and $\mathbf{R}_\sigma$, see theoretical section). In the internal forces vector building, `Material` has to compute the current stress state via a stress-return algorithm (see Figure 5-34 for the method which handles the stress return in the von Mises case).

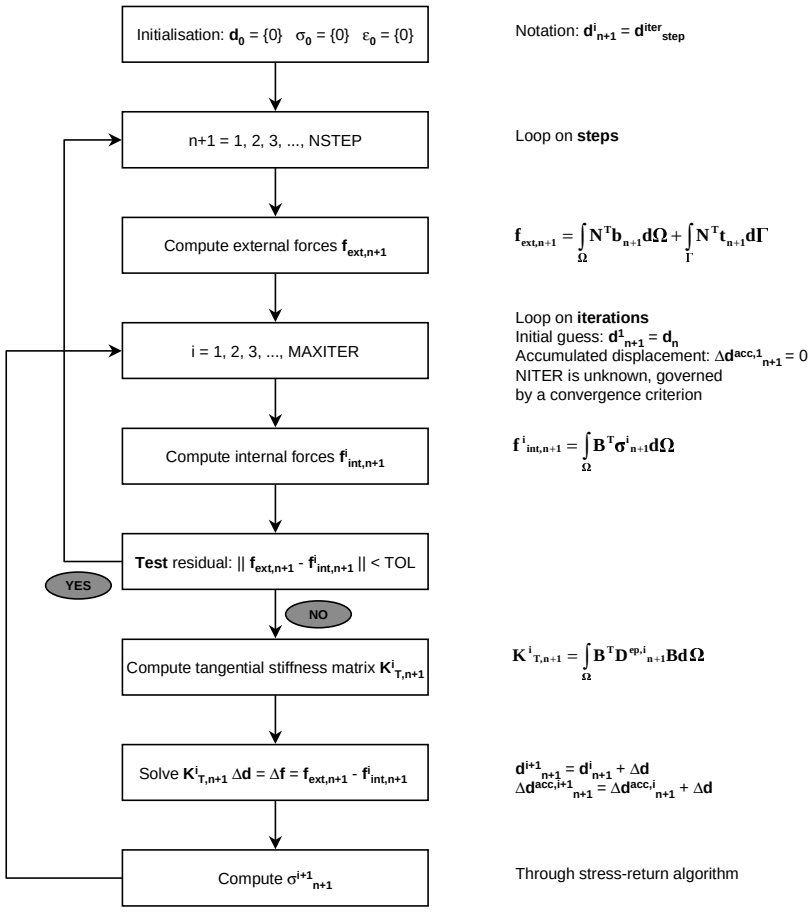



Figure 4-8: Global nonlinear algorithm

```
//------------------------------
FloatArray* NLSolver :: Solve ()
//------------------------------

{
Skyline      *jacobian;
FloatArray   *x,*y,*dx, *dxacc;
double       initialNorm, norm, tolerance, maxIterations;
const double PRECISION=1.e-9;
int          i,hasConverged=0;
tolerance = this->give('t');
maxIterations = this->give('n');
x = this->domain->GiveInitialGuess();
dxacc = new FloatArray(this -> domain -> giveNumberOfFreeDofs());
for (i=1; i<=maxIterations; i++) {
      this->currentIteration = i;
      y = this->domain->ComputeRHSAt(dxacc)->minus();
      jacobian = this->domain->ComputeJacobian(dxacc);
      this->linearSystem->setLHSTo(jacobian);
      this->linearSystem->setRHSTo(y);
      norm = y->giveNorm();
      if (i == 1) initialNorm = norm;
      if (initialNorm = 0.) initialNorm = 1.;
      if (norm/initialNorm<tolerance||norm<PRECISION) {
                  hasConverged = 1;
                  linearSystem->updateYourself();
                  break;
      }
      if (norm/initialNorm > 10) {
                  hasConverged = 0;
                  linearSystem->updateYourself();
                  break;
      }
      this->linearSystem->solveYourself();
      dx = this->linearSystem->giveSolutionArray();
      x->add(dx);
      dxacc->add(dx);
      linearSystem->updateYourself();
}
this->numberOfIterations = i;
this->convergenceStatus = hasConverged;
delete dxacc; delete jacobian; delete y;
return x;
}
```

Figure 4-9: Method `NLSolver::Solve()`

# Chapter 5

# Conclusion

This work has dealt with stabilized finite element formulations in geomechanics, i.e. incompressible and dilatant elastoplasticity in the field of continuum mechanics. Following the methodology used with success in computational fluid dynamics, we have derived a stable mixed finite element scheme by adding terms to the original matricial form of the problem. The need for such a formulation results from the fact that to our knowledge none of the methods present in the literature is able to handle elastoplasticity (in presence of incompressible or dilatant plastic flow) in a correct way for low-order – and therefore computationally cheap – elements like linear triangles. Another interesting point is that our stabilized formulation allows to mix different element types in the same mesh.

The first scheme that we have studied is the Galerkin Least-Squares formulation (used in CFD by Hughes et al [16][30]). In this scheme, stabilizing terms are derived from a least-squares potential, and consistency is preserved as the residual of the governing equilibrium equation is explicitely present in these terms. We have shown that neglecting the displacement term in the weighting part of the stabilizing terms – therefore keeping only the gradient of the pressure and getting a modified-GLS scheme form-like to the SUPG method [4] – doesn't affect results and simplifies the linearization process in the elastoplastic case. An important part of stabilized formulations is the definition of the stabilization factor $\tau$. Its nature has been discussed and the proposed scheme introduces $\tau$ similarly to what has been done in Stokes flow stabilization [27]. The second scheme, called the Finite Increment Calculus formulation (developed in the Navier-Stokes context by Oñate [42][43]), has provided a tentative physical justification of the modified-GLS form and also introduced a directional character – depending on the increment of displacement – in our formulation. We have established that under certain assumptions, an equivalence between the two approaches can be derived through corresponding Euler-Lagrange equations, giving rise to a FIC-scheme formulation. Finally the Laplacian Pressure Operator Scheme (introduced in soil plasticity by Pastor [44]) has also been implemented.

Following the results of four typical benchmarks, namely the Stokes driven cavity flow (incompressible elasticity), the thick cylinder test loaded by an internal pressure, the stability of a vertical cut and the bearing capacity of a strip footing in various elastoplastic situations (both incompressible and dilatant plastic flows), have shown the validity of the proposed approaches and has allowed us to calibrate the stabilization factor $\tau$. Both the modified-GLS and the LPOS

methods have given similar and satisfactory results, and while the FIC-scheme method seems to have more trouble to converge, it still reproduces the same results as the other formulations in most of the benchmarks. However the utility of a directional character in our formulation hasn't been discovered yet.

From the implementational point of view, the introduction of new elements, materials or even formulations has been made easy by the usage of an object-oriented approach.

To conclude I would recommend the use of one of the following two approaches:

- modified-GLS approach, where stabilizing terms take the form:

$$\sum_{e=1}^{n_{el}} \int_{\Omega^e} \left(\overrightarrow{\nabla}\widetilde{\mathbf{N}}\right)^T \frac{\alpha^e \left(h^e\right)^2}{2\mu} \mathbf{I} \left[\mathbf{L}^T \boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right) + \mathbf{f}\right] d\Omega \tag{5.1}$$

- LPOS, i.e. introduce stabilizing terms of the form:

$$\sum_{e=1}^{n_{el}} \frac{\alpha^e \left(h^e\right)^2}{2\mu} \int_{\Omega^e} \left(\overrightarrow{\nabla}\widetilde{\mathbf{N}}\right)^T \overrightarrow{\nabla}\widetilde{\mathbf{N}} d\Omega \tag{5.2}$$

Optimal values of $\alpha^e$ have been found in the lower range of the interval $[10^{-1}, 1]$.

The former approach has the advantage of being mathematically more consistent than the latter due to the presence of the residual in the stabilizing terms, but the Laplacian Pressure Operator Scheme yields a smaller computational cost.

Future developments could include the introduction of the multilaminate model (a material with preferential directions of plastification) or the derivation of stabilization for a finite strains model. In both cases the FIC-scheme approach (introducing a directional character in the formulation) may help if the two other approaches are inefficient. Another interesting point could be the search for a stabilization factor $\tau$ using an analytical solution that we have at hand (the thick cylinder test solution for instance) from which an exact stiffness matrix could be derived and then compared with a discrete one yielding from stabilized formulations.

# Appendix A: Classes and Methods

Class **Domain** inherits from : **-**

| Tasks | Attributes | Methods |
|---|---|---|
| 1) creation | - | *Domain()*<br>instanciateYourself() |
| 2) management of the problem's components | elementList<br>nodeList<br>materialList<br>loadList<br>loadTimeFunctionList<br>nlSolver<br>timeIntegrationScheme<br>numberOfElements<br>numberOfNodes<br>numberOfFreeDofs | giveElement(i)<br>giveNode(i)<br>giveMaterial(i)<br>giveLoad(i)<br>giveLoadTimeFunction(i)<br>giveNLSolver()<br>giveTimeIntegrationScheme()<br>giveNumberOfElements()<br>giveNumberOfNodes()<br>giveNumberOfFreeDofs() |
| 3) problem solving | unknownArray | giveUnknownArray()<br>solveYourself()<br>formTheSystemAt(aStep)<br>solveYourselfAt(aStep)<br>terminate(aStep) |
| 4) interactions with the nonlinear solver | - | giveInitialGuess()<br>givePastUnknownArray()<br>computeRHSAt(aΔd)<br>computeInternalForces(aΔd)<br>computeLoadVectorAt(aStep)<br>computeJacobian()<br>computeTangentStiffnessMatrix() |
| 5) input / output | dataFileName<br>inputStream<br>outputStream | giveDataFileName()<br>giveInputStream()<br>giveOutputStream()<br>readNumberOf(aComponent) |

Figure 5-1: Class `Domain`

Class **FEMComponent** inherits from : **-**

| Tasks | Attributes | Methods |
|---|---|---|
| 1) creation, access | number<br>domain | *FEMComponent (aDomain, aNumber)* |
| 2) reading in the input file | - | read(aChar)<br>readIfHas(aChar)<br>readInteger(aChar)<br>readNumberOf(aChar)<br>readString(aChar) |
| 3) identification | - | giveClassName(aChar)<br>giveKeyword(aChar) |

Figure 5-2: Class `FEMComponent`

Class **Node** inherits from : FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data | number<br>domain | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Node(aDomain, aNumber)*<br>instanciateYourself() |
| 2) positioning in space | coordinates | getCoordinates()<br>giveCoordinate(i) |
| 3) management of the degrees of freedom | dofArray<br>numberOfDofs | giveDof(i)<br>giveNumberOfDofs() |
| 4) management of the nodal load vector: | | |
| a) computation | loadArray | computeLoadVectorAt(aStep)<br>giveLoadArray() |
| b) assembly | locationArray | assembleYourLoadsAt(aStep)<br>giveLocationArray() |
| 5) output | - | printOutputAt(aStep, aFile)<br>printBinaryResults(aStep, aFile) |
| 6) internal handling | - | updateYourself()<br>giveClassName()<br>printYourself() |

Figure 5-3: Class `Node`

Class **Dof** inherits from : -

| Tasks | Attributes | Methods |
|---|---|---|
| 1) creation | node<br>number | *Dof(aNumber, aNode)*<br>giveNumber() |
| 2) unknowns handling | unknowns<br>pastUnknowns | computeStaticUnknown()<br>giveUnknown(aChar, aTimeStep)<br>givePastUnknown(aChar, aTimeStep)<br>getSolution() |
| 3) linear system building | equationNumber | giveEquationNumber() |
| 4) initial and boundary conditions | ic<br>bc | giveIC()<br>giveBC()<br>hasIC()<br>hasBC() |
| 5) output | - | printStaticOutputAt(aStep, aFile)<br>print (aStep, aFile) |
| 6) internal handliing | - | updateYourself()<br>printYourself() |

Figure 5-4: Class `Dof`

Class **Element** inherits from : FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data | number<br>domain | giveNumber()<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Element(aDomain, aNumber)*<br>typed()<br>ofType(anElementType)<br>instanciateYourself() |
| 2) attributes identification | nodeArray<br>numberOfNodes<br>material<br>bodyLoadArray<br>gaussPointArray<br>numberOfGaussPoints | giveNode(i)<br><br>giveMaterial()<br>giveBodyLoadArray() |
| 3) computation and assembly | stiffnessMatrix<br>constitutiveMatrix | giveStiffnessMatrix()<br>giveConstitutiveMatrix()<br>computeTangentStiffnessMatrix()<br>computeStaticLHSAt(aStep)<br>computeLoadVectorAt(aStep)<br>computeBcLoadVectorAt(aStep)<br>computeVectorOfPrescribed(aStep)<br>computeStaticRHSAt(aStep)<br>computeInternalForces(aΔd)<br>computeStrainIncrement(aGP, aΔd)<br>giveLocationArray()<br>... |
| 4) output | - | printOutputAt(aStep, aFile) |
| 5) internal handling | - | computeNumberOfDofs()<br>... |

Figure 5-5: Class `Element`

Class **PlaneStrain** inherits from : Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly | number<br>domain<br>... | giveNumber()<br>...<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *PlaneStrain(aDomain, aNumber)* |

Figure 5-6: Class `PlaneStrain`

Class **Quad_U** inherits from : PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>… | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Quad_U(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix()<br>computeNMatrixAt(aGP)<br>computeBMatrixAt(aGP)<br>computeCompactBMatrixAt(aXsiEtaPoint)<br>computeConstitutiveMatrix() |
| 3) numerical integration | jacobianMatrix | giveJacobianMatrix()<br>computeGaussPoints()<br>computeVolumeAround(aGP) |

Figure 5-7: Class `Quad_U`

Class **Quad_U_BBar** inherits from : Quad_U, PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation, access to data, computation & assembly, numerical integration, … | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Quad_U_BBar(aDomain, aNumber)* |
| 2) B-bar matrix handling | - | computeBMatrixAt(aGP)<br>giveBBarMatrix()<br>computeMeanBBar()<br>computeBBarAtCenter() |

Figure 5-8: Class `Quad_U_BBar`

Class **Quad_UP** inherits from : PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>… | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Quad_UP(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix()<br>computeNMatrixAt(aGP)<br>computeNpMatrixAt(aGP)<br>computeBMatrixAt(aGP)<br>computeCompactBMatrixAt(aXsiEtaPoint)<br>computeConstitutiveMatrix() |
| 3) numerical integration | jacobianMatrix | giveJacobianMatrix()<br>computeGaussPoints()<br>computeVolumeAround(aGP) |
| 4) residuals handling | - | computeResiduumTheta()<br>computeResiduumSigma(aΔd) |

Figure 5-9: Class `Quad_UP`

Class **Quad_UP_Stab** inherits from : Quad_UP, PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>numerical integration,<br>... | number<br>domain<br>... | giveNumber()<br>... |

| **Tasks** | **Attributes** | **Methods** |
|---|---|---|
| 1) creation | - | *Quad_UP_Stab(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix() |
| 3) stabilization | - | giveStabilizationFactor(aGP, aΔd)<br>giveElementSize()<br>giveGuMatrixAt(aGP, aD_bar)<br>giveGpMatrixAt(aGP, aD_bar)<br>giveD2NDX2 (anInt, aGP)<br>giveD2NDKsi2 (anInt)<br>giveDNDX (anInt, aGP)<br>giveD2XDKsi2 (anInt) |
| 4) residuals handling | - | computeResiduumSigma(aΔd) |

Figure 5-10: Class `Quad_UP_Stab`

Class **Quad_UP_Stab_Dir** inherits from : Quad_UP, PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>numerical integration<br>… | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Quad_UP_Stab_Dir(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix() |
| 3) stabilization | - | giveStabilizationFactor(aGP)<br>giveH_H_T(aGP, aΔd);<br>giveElementSize()<br>giveGuMatrixAt(aGP, aD_bar)<br>giveGpMatrixAt(aGP, aD_bar)<br>giveD2NDX2 (anInt, aGP)<br>giveD2NDKsi2 (anInt)<br>giveDNDX (anInt, aGP)<br>giveD2XDKsi2 (anInt) |
| 4) residuals handling | - | computeResiduumSigma(aΔd) |

Figure 5-11: Class `Quad_UP_Stab_Dir`

Class **Triangle_U** inherits from : PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>… | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Triangle_U(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix()<br>computeNMatrixAt(aGP)<br>computeBMatrixAt(aGP)<br>computeCompactBMatrixAtCenter()<br>computeConstitutiveMatrix() |
| 3) numerical integration | - | computeArea()<br>computeGaussPoints()<br>computeVolumeAround(aGP) |

Figure 5-12: Class `Triangle_U`

Class **Triangle_UP** inherits from : PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>… | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Triangle_UP(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix()<br>computeNMatrixAt(aGP)<br>computeNpMatrixAt(aGP)<br>computeBMatrixAt(aGP)<br>computeCompactBMatrixAtCenter()<br>computeConstitutiveMatrix() |
| 3) numerical integration | - | computeArea()<br>computeGaussPoints()<br>computeVolumeAround(aGP) |
| 4) residuals handling | - | computeResiduumTheta()<br>computeResiduumSigma(aΔd) |

Figure 5-13: Class `Triangle_UP`

Class **Triangle_UP_Stab** inherits from : Triangle_UP, PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>numerical integration,<br>… | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Triangle_UP_Stab(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix() |
| 3) stabilization | - | giveStabilizationFactor(aGP, aΔd)<br>giveGpMatrixAt(aGP, aD_bar) |
| 4) residuals handling | - | computeResiduumSigma(aΔd) |

Figure 5-14: Class `Triangle_UP_Stab`

Class **Triangle_UP_Stab_Dir** inherits from : Triangle_UP, PlaneStrain, Element, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation & assembly,<br>numerical integration,<br>… | number<br>domain<br>… | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Triangle_UP_Stab_Dir(aDomain, aNumber)* |
| 2) computation | - | computeTangentStiffnessMatrix() |
| 3) stabilization | - | giveStabilizationFactor(aGP)<br>giveH_H_T(aGP, aΔd)<br>giveGpMatrixAt(aGP, aD_bar) |
| 4) residuals handling | - | computeResiduumSigma(aΔd) |

Figure 5-15: Class `Triangle_UP_Stab_Dir`

Class **Material** inherits from : FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data | number<br>domain | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *Material(aDomain, aNumber)*<br>typed()<br>ofType(aMaterialType) |
| 2) attributes identification | propertyDictionnary | give(aProperty) |
| 3) internal handling | - | giveClassName()<br>giveKeyword()<br>printYourself() |

Figure 5-16: Class `Material`

Class **ElasticMaterial** inherits from : Material, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>attributes identification,<br>... | number<br>domain<br>propertyDictionary | giveNumber()<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *ElasticMaterial(aDomain, aNumber)*<br>instanciateYourself() |
| 2) computation | - | computeStress(aGP, anElem, aΔd)<br>computeConstitutiveMatrix(aGP, anElem) |
| 3) internal handling | - | giveClassName()<br>printYourself() |

Figure 5-17: Class `ElasticMaterial`

Class **ElasticMaterial_UP** inherits from : ElasticMaterial, Material, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>attributes identification,<br>Computation,<br>... | number<br>domain<br>propertyDictionary | giveNumber()<br>computeConstitutiveMatrix(aGP, anElem)<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *ElasticMaterial_UP(aDomain, aNumber)* |
| 2) computation | - | computeStress(aGP, anElem, aΔd)<br>computeDupBar(aGP, anElem)<br>computeDpuBar(aGP, anElem)<br>computeDppBar(aGP, anElem) |
| 3) internal handling | - | giveClassName()<br>printYourself() |

Figure 5-18: Class `ElasticMaterial_UP`

Class **VonMisesMaterial** inherits from : Material, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>attributes identification,<br>... | number<br>domain<br>propertyDictionary | giveNumber()<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *VonMisesMaterial(aDomain, aNumber)*<br>instanciateYourself() |
| 2) computation | - | computeStress(aGP, anElem, aΔd)<br>computeConstitutiveMatrix(aGP, anElem)<br>computeDFDSigma(aStressState)<br>computeYieldFunctionFor(aStressState)<br>computeStressLevelFor(aStressState) |
| 3) internal handling | - | giveClassName()<br>printYourself() |

Figure 5-19: Class `VonMisesMaterial`

Class **VonMisesMaterial_UP** inherits from : VonMisesMaterial, Material, FEMComponent

| Inherited Tasks | Inherited Attributes | Inherited Methods |
|---|---|---|
| creation,<br>access to data,<br>attributes identification,<br>Computation,<br>... | number<br>domain<br>propertyDictionary | giveNumber()<br>… |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *VonMisesMaterial_UP(aDomain, aNumber)* |
| 2) computation | - | computeStress(aGP, anElem, aΔd)<br>computeConstitutiveMatrix(aGP, anElem)<br>computeDupBar(aGP, anElem)<br>computeDpuBar(aGP, anElem)<br>computeDppBar(aGP, anElem) |
| 3) internal handling | - | giveClassName()<br>printYourself() |

Figure 5-20: Class `VonMisesMaterial_UP`

Class **DruckerPragerMaterial** inherits from : Material, FEMComponent

| Inherited Tasks | Inherited Attributes | Inherited Methods |
|---|---|---|
| creation,<br>access to data,<br>attributes identification,<br>... | number<br>domain<br>propertyDictionary | giveNumber()<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *DruckerPragerMaterial(aDomain, aNumber)*<br>instanciateYourself() |
| 2) computation | - | computeStress(aGP, anElem, aΔd)<br>computeConstitutiveMatrix(aGP, anElem)<br>computeDFDSigma(aStressState)<br>computeDQDSigma(aStressState)<br>computeYieldFunctionFor(aStressState)<br>computeStressLevelFor(aStressState) |
| 3) DP parameters | k<br>aphi<br>apsi | setDPParameters() |
| 3) internal handling | - | giveClassName()<br>printYourself() |

Figure 5-21: Class `DruckerPragerMaterial`

Class **DruckerPragerMaterial_UP** inherits from : DruckerPragerMaterial, Material, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>attributes identification,<br>Computation,<br>... | number<br>domain<br>propertyDictionary | giveNumber()<br>… |

| **Tasks** | **Attributes** | **Methods** |
|---|---|---|
| 1) creation | - | *DruckerPragerMaterial_UP(aDomain, aNumber)* |
| 2) computation | - | computeStress(aGP, anElem, aΔd)<br>computeConstitutiveMatrix(aGP, anElem)<br>computeDupBar(aGP, anElem)<br>computeDpuBar(aGP, anElem)<br>computeDppBar(aGP, anElem) |
| 3) internal handling | - | giveClassName()<br>printYourself() |

Figure 5-22: Class `DruckerPragerMaterial_UP`

Class **GaussPoint** inherits from : -

| Tasks | Attributes | Methods |
|---|---|---|
| 1) creation | number<br>element<br>coordinates<br>weight | *GaussPoint(aNumber, anElement, x, y, w)*<br>giveNumber()<br>giveCoordinates()<br>giveCoordinate(i)<br>giveWeight() |
| 2) stresses / strains handling | stressVector<br>previousStressVector<br>strainVector | giveStressVector()<br>givePreviousStressVector()<br>giveStrainVector()<br>letStressVectorBe(aStressState)<br>letPreviousStressVectorBe(aStressState)<br>letStrainVectorBe(aStrainState) |
| 3) stress return computation | deltaGamma<br>plasticCode<br>stressLevel<br>apexState | setDeltaGamma(aDeltaGamma)<br>giveDeltaGamma()<br>isPlastic()<br>givePlasticCode()<br>computeStressLevel()<br>giveStressLevel()<br>giveApexState()<br>isAtApex(aBoolean)<br>... |
| 4) residuals handling | rTheta<br>rSigma<br>previousRTheta<br>previousRSigma | giveRTheta()<br>giveRSigma()<br>setRTheta(aRTheta)<br>setRSigma(aRSigma)<br>... |
| 5) output | - | printOutput(aFile)<br>printBinaryResults(aFile) |
| 6) internal handling | - | updateYourself() |

Figure 5-23: Class `GaussPoint`

Class **NLSolver** inherits from : FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data | number<br>domain | giveNumber()<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *NLSolver(aDomain, aNumber)*<br>typed()<br>ofType(aNLSolverType) |
| 2) attributes identification | propertyDictionnary | give(aProperty) |
| 3) computation | linearSystem<br>maxIterations<br>numberOfIterations<br>currentIteration<br>tolerance<br>convergenceStatus<br>consistentDep | solve()<br>giveLinearSystem()<br>giveNumberOfIterations()<br>giveCurrentIteration()<br>giveConvergenceStatus()<br>giveConsistentDep() |
| 4) internal handling | - | updateYourself()<br>giveClassName()<br>giveKeyword()<br>printYourself() |

Figure 5-24: Class `NLSolver`

Class **NewtonRaphson** inherits from : NLSolver, FEMComponent

| **Inherited Tasks** | **Inherited Attributes** | **Inherited Methods** |
|---|---|---|
| creation,<br>access to data,<br>computation,<br>... | number<br>domain<br>linearSystem<br>... | giveNumber()<br>... |
| **Tasks** | **Attributes** | **Methods** |
| 1) creation | - | *NewtonRaphson(aDomain, aNumber)*<br>instanciateYourself() |
| 2) internal handling | - | giveClassName()<br>printYourself() |

*Remark*: Classes **ModNewtonRaphson** and **ConstantStiffness** also inherit from : NLSolver, FEMComponent

Figure 5-25: Class `NewtonRaphson`

```
//------------------------------------------------------------------------
FloatMatrix*  Element :: ComputeStaticLhsAt (TimeStep* stepN, FloatArray* dxacc)
//------------------------------------------------------------------------
{
FloatArray *dElem = dxacc->Extract(this->giveLocationArray());
return  this->computeTangentStiffnessMatrix(dElem) ;
delete dElem;
}
```

Figure 5-26: Method `Element::ComputeStaticLHSAt(stepN, dxacc)`

```
FloatMatrix*  Quad_UP_Stab_Dir :: computeTangentStiffnessMatrix (FloatArray* dElem)
{
Material *mat; GaussPoint *gp;
FloatMatrix *b,*duuBarb,*duuBar,*Np;
double dV; int i;

stiffnessMatrix = new FloatMatrix();
mat    = this->giveMaterial();
for (i=0; i<numberOfGaussPoints; i++) {
               gp = gaussPointArray[i];
               b  = this->ComputeBmatrixAt(gp);
               if (! strcmp(nlSolverClassName,"ConstantStiffness")) {
                              duuBar = this->giveConstitutiveMatrix()->GiveCopy();
               } else {
                              duuBar = mat->ComputeConstitutiveMatrix(gp,this);
               }
               dV = this->computeVolumeAround(gp);

               //STANDARD GALERKIN TERMS

               duuBarb = duuBar->Times(b);
               stiffnessMatrix->addProduct(b,duuBarb,dV);
               double dppBar = mat->ComputeDppBar(gp,this);
               double dppBarTimesV = dppBar * dV;
               Np = this->ComputeNPmatrixAt(gp);
               stiffnessMatrix->addProduct(Np,Np,multMup*dppBarTimesV);
               FloatMatrix *dpuBar,*dupBar,*dupBarNp,*dpuBarNp;
               dupBar = mat->ComputeDupBar(gp,this);
               dpuBar = mat->ComputeDpuBar(gp,this);
               dupBarNp = dupBar->Times(Np);
               dpuBarNp = dpuBar->Times(Np);
               stiffnessMatrix->addProduct(b,dupBarNp,dV);
               stiffnessMatrix->addProduct(dpuBarNp,b,multMup*dV);

               //GLS TERMS FOR STABILIZATION

               FEMOptions* femOptions;
               femOptions = this->domain->giveFEMOptions();
               double uFlag    = femOptions -> give('u') ;
               double pFlag    = femOptions -> give('p') ;
               double pastorFlag = femOptions -> give('z') ;
               FloatMatrix *GuEl, *GuPl;
               GuEl = this->giveGuMatrixAt(gp, this->giveConstitutiveMatrix()->GiveCopy());
               GuPl = this->giveGuMatrixAt(gp, duuBar);
               FloatMatrix *GpEl, *GpPl; FloatMatrix *one;
               one = new FloatMatrix(4,1) ;
               one->at(1,1) =  1.; one->at(2,1) =  1.;
               one->at(3,1) =  0.; one->at(4,1) =  1.;
               GpEl = this->giveGpMatrixAt(gp, one);
               GpPl = this->giveGpMatrixAt(gp, dupBar);
               delete one;

               //h * h_T matrix
               FloatMatrix* h_h_T = this -> giveH_H_T(gp, dElem);
               double tauTimesdV = this->giveStabilizationFactor(gp) * dV;
               ...
```

Figure 5-27: Method `Quad_UP_Stab_Dir::computeTangentStiffnessMatrix(dElem)` #1/2

```
        ...
        if (pastorFlag == 0) {
              //u-u slot
              if (uFlag == 1) {
                 stiffnessMatrix->addProduct(GuEl,h_h_T->Times(GuPl), multGu*tauTimesdV);
              } else if (uFlag == 2) {
                 stiffnessMatrix->addProduct(GuPl,h_h_T->Times(GuPl), multGu*tauTimesdV);
              }
              //u-p slot
              if (uFlag == 1) {
                 stiffnessMatrix->addProduct(GuEl,h_h_T->Times(GpPl), multGu*tauTimesdV);
              } else if (uFlag == 2) {
                 stiffnessMatrix->addProduct(GuPl,h_h_T->Times(GpPl), multGu*tauTimesdV);
              }
              //p-u slot
              if (pFlag == 1) {
                 stiffnessMatrix->addProduct(GpEl,h_h_T->Times(GuPl), multGLS*tauTimesdV);
              } else if (pFlag == 2) {
                 stiffnessMatrix->addProduct(GpPl,h_h_T->Times(GuPl), multGLS*tauTimesdV);
              }
           }

           //p-p slot
           if (pFlag == 1) {
              stiffnessMatrix->addProduct(GpEl,h_h_T->Times(GpPl), multGLS*tauTimesdV);
           } else if (pFlag == 2) {
              stiffnessMatrix->addProduct(GpPl,h_h_T->Times(GpPl), multGLS*tauTimesdV);
           }

           delete GuEl; delete GuPl; delete GpEl; delete GpPl; delete h_h_T;
           delete dupBar; delete dpuBar; delete dupBarNp; delete dpuBarNp;
           delete duuBar; delete duuBarb; delete Np; delete b;
}
return stiffnessMatrix;
}
```

Figure 5-28: Method `Quad_UP_Stab_Dir::computeTangentStiffnessMatrix(dElem) #2/2`

```
FloatMatrix*  Quad_UP_Stab_Dir :: giveGpMatrixAt (GaussPoint* gp, FloatMatrix* dupBar)
{
               FloatMatrix  *answer, *jacGP, *inv ;
               FloatArray   *Coord ;
               double       ksi,eta,n1Ksi,n2Ksi,n3Ksi,n4Ksi,n1Eta,n2Eta,n3Eta,n4Eta,ksiX,ksiY,etaX,etaY ;
               double          fourth = 0.25;
               double d1bar = dupBar->at(1,1); double d2bar = dupBar->at(2,1);
               double d3bar = dupBar->at(3,1); double d4bar = dupBar->at(4,1);
               ksi = gp -> giveCoordinate(1) ;
               eta = gp -> giveCoordinate(2) ;

               Coord = gp -> giveCoordinates() ;
               jacGP = this -> giveJacobianMatrix() -> EvaluatedAt(Coord) ;
               inv   = jacGP -> GiveInverse();
               ksiX = inv -> at(1,1); ksiY = inv -> at(1,2);
               etaX = inv -> at(2,1); etaY = inv -> at(2,2);

               // partial derivatives of the shape functions N_i, with respect to ksi and eta
               n1Ksi = (-1. + eta) * fourth; n2Ksi = ( 1. - eta) * fourth;
               n3Ksi = ( 1. + eta) * fourth; n4Ksi = (-1. - eta) * fourth;
               n1Eta = (-1. + ksi) * fourth; n2Eta = (-1. - ksi) * fourth;
               n3Eta = ( 1. + ksi) * fourth; n4Eta = ( 1. - ksi) * fourth;

               // construction of Gp
               answer = new FloatMatrix(2,12);
               answer -> at(1,3) = (d1bar*(n1Ksi*ksiX + n1Eta*etaX))+(d3bar*(n1Ksi*ksiY + n1Eta*etaY));
               answer -> at(2,3) = (d2bar*(n1Ksi*ksiY + n1Eta*etaY))+(d3bar*(n1Ksi*ksiX + n1Eta*etaX));
               answer -> at(1,6) = (d1bar*(n2Ksi*ksiX + n2Eta*etaX))+(d3bar*(n2Ksi*ksiY + n2Eta*etaY));
               answer -> at(2,6) = (d2bar*(n2Ksi*ksiY + n2Eta*etaY))+(d3bar*(n2Ksi*ksiX + n2Eta*etaX));
               answer -> at(1,9) = (d1bar*(n3Ksi*ksiX + n3Eta*etaX))+(d3bar*(n3Ksi*ksiY + n3Eta*etaY));
               answer -> at(2,9) = (d2bar*(n3Ksi*ksiY + n3Eta*etaY))+(d3bar*(n3Ksi*ksiX + n3Eta*etaX));
               answer -> at(1,12) = (d1bar*(n4Ksi*ksiX + n4Eta*etaX))+(d3bar*(n4Ksi*ksiY + n4Eta*etaY));
               answer -> at(2,12) = (d2bar*(n4Ksi*ksiY + n4Eta*etaY))+(d3bar*(n4Ksi*ksiX + n4Eta*etaX));

               delete jacGP; delete inv;
               return answer;
}
```

Figure 5-29: Method `Quad_UP_Stab_Dir::giveGpMatrixAt(gp, dupBar)`

```
double Quad_UP_Stab_Dir :: giveStabilizationFactor (GaussPoint* gp)
{
                double tau = 0.0;
                //mu
                Material *mat;
                double   e,nu;
                double   mu_el;
                mat   = this -> giveMaterial();
                e     = mat -> give('E');
                nu    = mat -> give('n');
                mu_el = e / (2.0+nu+nu);

                FEMOptions* femOptions = this->domain->giveFEMOptions();
                double delta1  = femOptions -> give('q') ;

                tau = delta1 * 3.0 / (8.0 * mu_el);
                return tau;
}
```

Figure 5-30: Method `Quad_UP_Stab_Dir::giveStabilizationFactor(gp)`

```
FloatMatrix*  Quad_UP_Stab_Dir :: giveH_H_T (GaussPoint* gp, FloatArray* dElem)
{
                FloatMatrix* h_h_T;

                //isotropic version [Hughes]
                double h = this->giveElementSize();
                FloatMatrix* h_h_T_iso = new FloatMatrix(2,2);
                h_h_T_iso -> at(1,1)    = h*h ;
                h_h_T_iso -> at(1,2)    = 0.0 ;
                h_h_T_iso -> at(2,1)    = 0.0 ;
                h_h_T_iso -> at(2,2)    = h*h ;
                //return h_h_T_iso;

                //directional version according to [Onate 2000], CMAME 182 p. 365
                FloatArray* d = (this->ComputeNmatrixAt(gp))->Times(dElem);
                FloatArray* dOnly = new FloatArray(2);
                dOnly -> at(1) = d -> at(1);
                dOnly -> at(2) = d -> at(2);
                delete d;
                double norm = dOnly->giveNorm();

                if (norm == 0) {
                                h_h_T = h_h_T_iso;
                } else {
                                dOnly->times(1.0/norm); //unit vector

                                //Computation of hs_max
                                ...

                                //directional DIAGONAL version
                                h_h_T = new FloatMatrix(2,2);
                                double d1 = dOnly->at(1);
                                double d2 = dOnly->at(2);
                                h_h_T -> at(1,1)    = hs_max*hs_max*d1*d1 ;
                                h_h_T -> at(1,2)    = 0.0 ;
                                h_h_T -> at(2,1)    = 0.0 ;
                                h_h_T -> at(2,2)    = hs_max*hs_max*d2*d2 ;
                }
                delete dOnly;
                //h_h_T->printYourself("h_h_T");
                return h_h_T;
}
```

Figure 5-31: Method `Quad_UP_Stab_Dir::giveH_H_T(gp, dElem)`

```
FloatArray*  Quad_UP :: ComputeStaticRhsAt (TimeStep* stepN, FloatArray* dxacc)
{
   FloatArray *fInternal,*fExternal,*dElem,*residuumTheta,*residuumSigma;
   dElem = dxacc->Extract(this->giveLocationArray());

   //add delta prescribed displacement vector - SC 29.04.99
   ...

   fInternal = this->ComputeInternalForces(dElemTot);
   fExternal = this->ComputeLoadVectorAt(stepN);
   residuumTheta = this->ComputeResiduumTheta();
   residuumSigma = this->ComputeResiduumSigma(dElemTot);

   FloatArray *answer;
   answer = new FloatArray();
   answer->plus(fExternal);
   answer->minus(fInternal);
   answer->minus(residuumTheta);
   answer->minus(residuumSigma);

   delete dElem; delete fExternal; delete fInternal;
   delete residuumTheta; delete residuumSigma;

   return answer;
}
```

Figure 5-32: Method `Quad_UP::ComputeStaticRHSAt(stepN, dxacc)`

```
FloatArray*  Element :: ComputeInternalForces (FloatArray* dElem)
{
   Material         *mat;
   GaussPoint       *gp;
   FloatMatrix      *b;
   FloatArray       *f;
   double           dV;
   int              i;

   mat   = this->giveMaterial();
   f     = new FloatArray();
   for (i=0; i<numberOfGaussPoints; i++) {
      gp    = gaussPointArray[i];
      b     = this->ComputeBmatrixAt(gp);
      mat -> ComputeStress(dElem,this,gp);
      dV    = this->computeVolumeAround(gp);
      f->plusProduct(b,gp->giveStressVector(),dV);
      delete b;
   }
   return f;
}
```

Figure 5-33: Method `Element::  ComputeInternalForces(dElem)`

```
void  VonMisesMaterial_UP :: ComputeStress (FloatArray* dxacc, Element* elem, GaussPoint* gp)
{
                FloatArray *sigmaTrial, *deltaEpsilon; FloatMatrix *Del;
                // as we are here in u-p, Del is in fact DelBar!!!
                Del = elem->giveConstitutiveMatrix()->GiveCopy();
                deltaEpsilon = elem->computeStrainIncrement(gp,dxacc);

                sigmaTrial = Del->Times(deltaEpsilon);
                FloatArray *one, *Np;
                one = (new FloatArray(4))->giveOneArray();
                Np = elem->ComputeNParrayAt(gp);
                double deltaP; deltaP = Np->transposedTimes(dxacc);
                FloatArray *oneDeltaP = one->Times(deltaP);
                sigmaTrial->add(oneDeltaP); delete oneDeltaP;

                double rTheta = 0.; Material *mat ;
                double   e,nu,k; mat   = elem -> giveMaterial() ;
                e  = mat -> give('E') ; nu  = mat -> give('n') ;
                k = e / (3. * (1. - 2. * nu));
                rTheta += gp->givePreviousRTheta();
                rTheta += one->transposedTimes(deltaEpsilon);
                rTheta -= deltaP / k;
                delete Np;

                sigmaTrial->add(gp->givePreviousStressVector());
                double fSigTr = this->computeYieldFunctionFor(sigmaTrial);

                if (fSigTr > 0) {
                                double deltaGamma;
                                FloatArray *dFDSigma = this->computeDFDSigma(sigmaTrial);
                                FloatArray *DeldFDSigma = Del->Times(dFDSigma);
                                deltaGamma = fSigTr / (dFDSigma->transposedTimes(DeldFDSigma));
                                sigmaTrial->minus((DeldFDSigma)->times(deltaGamma));
                                gp->isPlastic();
                                gp->setDeltaGamma(deltaGamma);
                                rTheta -= (one->transposedTimes(dFDSigma))*deltaGamma;
                                delete dFDSigma; delete DeldFDSigma;
                }
                gp->letStressVectorBe(sigmaTrial); //gp stores sigmaTrial (corrected)
                gp->setRTheta(rTheta); //gp stores also the residuum of the second equation
                delete Del; delete deltaEpsilon; delete one;
}
```

Figure 5-34: Method `VonMisesMaterialUP::ComputeStress(dxacc, elem, gp)`

# Appendix B: Vector Notation for the Plane Strain Case

In chapters 2 and 3, the vector notation has been applied to the general three-dimensional case, while all subsequent benchmarks have been conducted in the plane strain formulation. That is why we summarize here the definition of the different column-vectors and matrices that we have defined in this work, specialized to the plane strain case. In general, column-vectors can be obtained from the three-dimensional case by neglecting the last two rows (as $\varepsilon_{13} = \varepsilon_{23} = 0$). For operators like $\mathbf{B}$ or $\mathbf{L}$, the last column is also dropped (as $n_{sd} = 2$).

**Remark 49** *this $4 \times 4$ matricial representation is a generalization of the $3 \times 3$ matricial representation introduced in [28], where the $(\cdot)_{33}$ component has been introduced explicitely in order to facilitate future developments, like for instance an axisymmetic formulation.*

The stress vector $\boldsymbol{\sigma}$ is defined by:

$$\boldsymbol{\sigma} = \left\{ \begin{array}{c} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ \sigma_{33} \end{array} \right\} \tag{5.3}$$

The strain vector $\boldsymbol{\varepsilon}$ reads:

$$\boldsymbol{\varepsilon} = \left\{ \begin{array}{c} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \\ \varepsilon_{33} \end{array} \right\} = \left\{ \begin{array}{c} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \\ u_{3,3} \end{array} \right\} \tag{5.4}$$

and the $4 \times 4$ constitutive matrix $\mathbf{D}$ can be expressed as:

$$\mathbf{D} = \begin{bmatrix} \lambda+2\mu & \lambda & 0 & \lambda \\ \lambda & \lambda+2\mu & 0 & \lambda \\ 0 & 0 & \mu & 0 \\ \lambda & \lambda & 0 & \lambda+2\mu \end{bmatrix} \tag{5.5}$$

The elemental strain-displacement matrix $\mathbf{B}^e$ is defined as $\mathbf{B}^e = [\mathbf{B}^e_1, ..., \mathbf{B}^e_a, ...]$, with $1 \leq a \leq n_{en}$. In plane strain, $\mathbf{B}^e_a$ can be written:

$$\mathbf{B}^e_a = \begin{bmatrix} N_{a,1} & 0 \\ 0 & N_{a,2} \\ N_{a,2} & N_{a,1} \\ 0 & 0 \end{bmatrix} \tag{5.6}$$

while the differential operator $\mathbf{L}$ reads:

$$\mathbf{L} = \begin{bmatrix} \dfrac{\partial}{\partial x_1} & 0 \\ 0 & \dfrac{\partial}{\partial x_2} \\ \dfrac{\partial}{\partial x_2} & \dfrac{\partial}{\partial x_1} \\ 0 & 0 \end{bmatrix} \tag{5.7}$$

The vector representation $\mathbf{1}$ of the Kronecker delta $\delta_{ij}$ is now given by:

$$\mathbf{1} = \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{Bmatrix} \tag{5.8}$$

and the 2D version of the gradient operator $\overrightarrow{\nabla}$ reads:

$$\overrightarrow{\nabla} = \begin{bmatrix} \dfrac{\partial}{\partial x_1} \\ \dfrac{\partial}{\partial x_2} \end{bmatrix} \tag{5.9}$$

Finally the plane strain version of the $\overline{\mathbf{B}}$ method (see section 2.3.3 for the derivation) requires the following changes: $\mathbf{B}_{dil} = [\mathbf{B}^1_{dil}, ..., \mathbf{B}^a_{dil}, ...]$, $1 \leq a \leq n_{en}$, where $n_{en}$ is the number of nodes of the element. $\mathbf{B}^a_{dil}$ is defined by:

$$\mathbf{B}^a_{dil} = \frac{1}{3} \begin{bmatrix} N_{a,1} & N_{a,2} \\ N_{a,1} & N_{a,2} \\ 0 & 0 \\ N_{a,1} & N_{a,2} \end{bmatrix} \tag{5.10}$$

This leads to $\overline{\mathbf{B}} = [\overline{\mathbf{B}^1}, ..., \overline{\mathbf{B}^a}, ...]$ with:

$$\overline{\mathbf{B}^a} = \begin{bmatrix} N_{a,1} + B_4 & B_6 \\ B_4 & N_{a,2} + B_6 \\ N_{a,2} & N_{a,1} \\ B_4 & B_6 \end{bmatrix} \tag{5.11}$$

where:

$$B_4 = \frac{1}{3}\left(\overline{B}_1 - N_{a,1}\right) \text{ and } B_6 = \frac{1}{3}\left(\overline{B}_2 - N_{a,2}\right) \tag{5.12}$$

# Bibliography


[1] BABUSKA I. Error Bounds for Finite Element Method. *Numerical Mathematics* (16), 322-333. 1971

[2] BEHR M., FRANCA L.P., TEZDUYAR T.E. Stabilized Finite Element Methods for the Velocity-Pressure-Stress Formulation of Incompressible Flows. *Computer Methods in Applied Mechanics and Engineering* (104), 31-48. 1993

[3] BORJA R.I., Computational Geomechanics. *Stanford University Course Material*. 2000

[4] BROOKS A.N., HUGHES T.J.R. Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations. *Computer Methods in Applied Mechanics and Engineering (32), 199-259.* 1982

[5] BREZZI F. On the Existence, Uniqueness and Approximation of Saddle-Point Problems Arising from Lagrange Multipliers. *Revue Fr. d'Automatique Inf. Rech. Opér., Série Rouge Analyse Numérique R-2* (8), 129-151. 1974

[6] BREZZI F., PITKARANTA J. On the Stabilization of Finite Element Approximations of the Stokes Problem. *Efficient Solutions of Elliptic Problems, Notes on Numerical Fluid Mechanics, Vieweg-Wiesbaden* (10), 11-19. 1984

[7] BREZZI F., BRISTEAU M.O., FRANCA L.P., MALLET M., ROGE G. A Relationship between Stabilized Finite Element Methods and the Galerkin Method with Bubble Functions. *Computer Methods in Applied Mechanics and Engineering* (96), 117-129. 1992

[8] CHEN W.F., LIU X.L. Limit Analysis in Soil Mechanics. *Elsevier*. 1990

[9] CHRISTIE I., GRIFFITHS D.F., MITCHELL A.R., ZIENKIEWICZ O.C. Finite Element Methods for Second Order Differential Equations with Significant First Derivatives. *International Journal for Numerical Methods in Engineering (10), 1389-1396.* 1976

[10] COMMEND S. An Object-Oriented Approach to Nonlinear Finite Element Programming. *LSC-EPFL (Swiss Federal Institute of Technology) Internal Report 98/07*. 1998

[11] COMMEND S., TRUTY A., ZIMMERMANN T. Enhanced Finite Element Formulations for Nonlinear and Ultimate Load Analysis of Geomaterials, with Application to Large Dam Sites. *LSC-EPFL (Swiss Federal Institute of Technology) Internal Report 99/02*. 1999

[12] DE BORST R., GROEN A.E. Some Observations on Element Performance in Isochoric and Dilatant Plastic Flow. *International Journal for Numerical Methods in Engineering (38), 2887-2906.* 1995

[13] DOUGLAS J., WANG J. An Absolutely Stabilized Finite Element Method for the Stokes Problem. *Mathematical Computations* (52), 495-508. 1989

[14] DUBOIS-PELERIN Y., ZIMMERMANN T., BOMME P. Object-Oriented Finite Element Programming: Theory and C++ Implementation for FEM_Object C++ 001. *Elmepress International.* 1993

[15] EYHERAMENDY D. Object-Oriented Finite Element Programming: Symbolic Derivations and Automatic Programming. *EPFL PhD Dissertation # 1752.* 1997

[16] FRANCA L.P. New Mixed Finite Element Methods. *Stanford University PhD Dissertation.* 1987

[17] FRANCA L.P., HUGHES T.J.R. Two Classes of Mixed Finite Element Methods. *Computer Methods in Applied Mechanics and Engineering* (69), 89-129. 1988

[18] FRANCA L.P., DUTRA DO CARMO E.G. The Galerkin Gradient Least-Squares Method. *Computer Methods in Applied Mechanics and Engineering (74), 41-54.* 1989

[19] FRANCA L.P., FREY S.L., HUGHES T.J.R. Stabilized Finite Element Methods: I. Application to the Advective-Diffusive Model. *Computer Methods in Applied Mechanics and Engineering (95), 253-276.* 1992

[20] FRANCA L.P., FREY S.L. Stabilized Finite Element Methods: II. The Incompressible Navier-Stokes Equations. *Computer Methods in Applied Mechanics and Engineering* (99), 209-233. 1992

[21] FRANCA L.P., HUGHES T.J.R. Convergence Analyses of Galerkin Least-Squares Methods for Symmetric Advective-Diffusive Forms of the Stokes and Navier-Stokes Equations. *Computer Methods in Applied Mechanics and Engineering* (105), 285-298. 1993

[22] GROSH K., PINSKY P.M. Design of Galerkin Generalized Least-Squares Methods For Timoshenko Beams. *Computer Methods in Applied Mechanics and Engineering (132), 1-16.* 1996

[23] HANNANI S.K., STALINAS M., DUPONT P. Incompressible Navier-Stokes Computations with SUPG and GLS Formulations - A Comparison Study. *Computer Methods in Applied Mechanics and Engineering* (124), 153-170. 1995

[24] HARARI I., HUGHES T.J.R. What Are $C$ and $h$? : Inequalities for the Analysis and Design of Finite Element Methods. *Computer Methods in Applied Mechanics and Engineering* (97), 157-192. 1992

[25] HILL R. The Mathematical Theory of Plasticity. *Oxford Science Publications.* 1950

[26] HUGHES T.J.R., PISTER K.S. Consistent Linearization in Mechanics of Solids and Structures. *Computers & Structures (8), 391-397.* 1978

[27] HUGHES T.J.R., FRANCA L.P., BALESTRA M. A New Finite Element Formulation for Computational Fluid Dynamics: V. Circumventing the Babuska-Brezzi Condition: A Stable Petrov-Galerkin Formulation of the Stokes Problem Accomodating Equal-Order Interpolations. *Computer Methods in Applied Mechanics and Engineering (59), 85-99.* 1986

[28] HUGHES T.J.R. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. *Prentice-Hall.* 1987

[29] HUGHES T.J.R., FRANCA L.P. A Mixed Finite Element Formulation for Reissner-Mindlin Plate Theory: Uniform Convergence of all Higher-Order Spaces. *Computer Methods in Applied Mechanics and Engineering* (67), 223-240. 1988

[30] HUGHES T.J.R., FRANCA L.P., BALESTRA M. A New Finite Element Formulation for Computational Fluid Dynamics: VIII. The Galerkin/Least-Squares Method for Advective-Diffusive Equations. *Computer Methods in Applied Mechanics and Engineering* (73), 173-189. 1989

[31] HUGHES T.J.R., FRANCA L.P., CHALOT F., JOHAN Z. Stabilized Finite Element Methods in Fluid Mechanics. *Stanford University Course Material.* 1994

[32] HUGHES T.J.R. Multiscale Phenomena: Green's Functions, the Dirichlet-to-Neumann Formulation, Subgrid Scale Models, Bubbles and the Origins of Stabilized Methods. *Computer Methods in Applied Mechanics and Engineering (127), 387-401.* 1995

[33] HUGHES T.J.R., BELYTSCHKO T. Nonlinear Finite Element Analysis. *Course Notes.* 1997

[34] JIRASEK M., BAZANT Z.P. Inelastic Analysis of Structures. *John Wiley & Sons.* 2001 (to appear)

[35] JOHNSON C., NAVERT U., PITKARANTA J. Finite Element Methods for Linear Hyperbolic Problems. *Computer Methods in Applied Mechanics and Engineering (45).* 285-312. 1984

[36] MALVERN L.E. Introduction to the Mechanics of a Continuous Medium. *Prentice-Hall.* 1969

[37] MATAR M., SALENCON J. Capacité portante des semelles filantes. *Revue Fr. de Géotechnique (9)*, 51-76. 1979

[38] MENETREY P., ZIMMERMANN T. Object-Oriented Non-Linear Finite Element Analysis: Application to J2 Plasticity. *Computers & Structures (49-5), 767-777.* 1993

[39] NAGTEGAAL J.C., PARKS D.M., RICE J.R. On Numerically Accurate Finite Element Solutions in the Fully Plastic Range. *Computer Methods in Applied Mechanics and Engineering* (4), 153-177. 1974

[40] ONATE E. On the Stabilization of Numerical Solution for Advective-Diffusive Transport and Fluid Flow Problems. *Publication CIMNE # 81.* 1996

[41] ONATE E., GARCIA J., IDELSOHN S. Computation of the Stabilization Parameter for the Finite Element Solution of Advective-Diffusive Problems. *Publication CIMNE # 100.* 1997

[42] ONATE E. Derivation Of Stabilized Equations for Numerical Solution of Advective-Diffusive Transport and Fluid Flow Problems. *Computer Methods in Applied Mechanics and Engineering (151), 233-265.* 1998

[43] ONATE E. A Stabilized Finite Element Method for Incompressible Viscous Flows Using a Finite Increment Calculus Formulation. *Computer Methods in Applied Mechanics and Engineering (182), 355-370.* 2000

[44] PASTOR M., QUECEDO M., ZIENKIEWICZ O.C. A Mixed Displacement-Pressure Formulation for Numerical Analysis of Plastic Failure. *Computers & Structures (62-1), 13-23.* 1997

[45] PASTOR M., LI T., LIU X., ZIENKIEWICZ O.C. Stabilized Low-Order Finite Elements for Failure and Localization Problems in Undrained Soils and Foundations. *Computer Methods in Applied Mechanics and Engineering (174), 219-234.* 1999

[46] PINSKY P. A Finite Element Formulation for Elastoplasticity Based on a Three-Field Variational Equation. *Computer Methods in Applied Mechanics and Engineering (61), 41-60.* 1987

[47] QUARTERONI A., VALLI A. Numerical Approximation of Partial Differential Equations. *Springer.* 1991

[48] RUSSO A. Bubble Stabilization of Finite Element Methods for the Linearized Incompressible Navier-Stokes Equations. *Computer Methods in Applied Mechanics and Engineering* (132), 335-343. 1996

[49] SHAKIB F. A Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations. *Stanford University PhD Dissertation.* 1988

[50] SIMO J.C., KENNEDY J.G., TAYLOR R.L. Complementary Mixed Finite Element Formulations for Elastoplasticity. *Computer Methods in Applied Mechanics and Engineering (74), 177-206.* 1989

[51] SIMO J.C., RIFAI M.S. A Class of Mixed Assumed Strain Methods and the Method of Incompatible Modes. *International Journal for Numerical Methods in Engineering (29), 1595-1638.* 1990

[52] SIMO J.C., HUGHES T.J.R. Computational Inelasticity. *Springer.* 1998

[53] TAYLOR C.A., HUGHES T.J.R., ZARINS C.K. Finite Element Modeling of Blood Flow in Arteries. *Computer Methods in Applied Mechanics and Engineering (158), 155-196.* 1998

[54] TAYLOR R.L., SIMO J.C., ZIENKIEWICZ O.C., CHAN A.C.H. The Patch-Test - A Condition for Assessing FEM Convergence. *International Journal for Numerical Methods in Engineering (22), 39-62.* 1986

[55] TAYLOR R.L. A Mixed-Enhanced Formulation for Tetrahedral Finite Elements. *International Journal for Numerical Methods in Engineering (47), 205-227.* 2000

[56] TERZAGHI K. Mécanique théorique des sols. *Dunod, Paris.* 1951

[57] TEZDUYAR T.E. Stabilized Finite Element Formulations for Incompressible Flow Computations. *Advances in Applied Mechanics* (28), 1-44. 1992

[58] TRUTY A., ZIMMERMANN T. A Robust Formulation for FE-Analysis of Elasto-Plastic Media. *Proceedings of Numerical Models in Geomechanics NUMOG VI*, 1997

[59] TRUTY A., ZIMMERMANN T. Stabilized Mixed u-p Formulation for Elasto-Plastic Media. *Proceedings of Korbielow '98*. 1998

[60] TRUTY A. A Stabilized FE-Formulation for Fully Saturated Two-Phase Media. *LSC-EPFL (Swiss Federal Institute of Technology) Internal Report 99/01*. 1999

[61] ZIENKIEWICZ O.C., QU S., TAYLOR R.L., NAKAZAWA S. The Patch Test for Mixed Formulations. *International Journal for Numerical Methods in Engineering (23), 1873-1883*. 1986

[62] ZIENKIEWICZ O.C., WU J. Incompressibility Without Tears - How To Avoid Restrictions of Mixed Formulation. *International Journal for Numerical Methods in Engineering (32), 1189-1203*. 1991

[63] ZIENKIEWICZ O.C., CHAN A.H.C., PASTOR M., SCHREFLER B.A., SHIOMI T. Computational Geomechanics with Special Reference to Earthquake Engineering. *John Wiley & Sons*. 1999

[64] ZIENKIEWICZ O.C., TAYLOR R.L. The Finite Element Method, Vol. 1: The Basis. *Butterworth & Heinemann*. 2000

[65] ZIENKIEWICZ O.C., TAYLOR R.L. The Finite Element Method, Vol. 3: Fluid Dynamics. *Butterworth & Heinemann*. 2000

[66] ZIMMERMANN T., TRUTY A., COMMEND S. A Comparison of Finite Element Enhancements for Incompressible and Dilatant Behavior of Geomaterials. *Proceedings of CIMASI '98*. 1998

[67] Z_SOIL.PC User Manual. *Zace Services Ltd., Lausanne*. 1985-2001.

# Curriculum Vitae

**Stéphane Commend**, born on the 30th of July 1971.

Nationality: Swiss. Male, single

### Education

- 1997-2001 - PhD Candidate, Swiss Federal Institute of Technology, Lausanne EPFL. Research topic: Stabilized Finite Elements in Geomechanics
- 1989-1994 - Civil Engineer Diploma, Swiss Federal Institute of Technology
- 1986-1989 - Certificate of Secondary Education, CESSEV, La Tour-de-Peilz

### Experience

- 1994-2001 - Research engineer, Laboratory of Structural and Continuum mechanics, Swiss Federal Institute of Technology. Work in the development of a limit analysis software for underground structures in soils and rocks as software engineer. Teaching assistant for the dynamics and soil mechanics courses (numerical methods)
- 1999-2000 - Visiting scholar in the Mechanics and Computation Division (Prof. Thomas J.R. Hughes), School of Engineering, Stanford University, USA

### Skills

- Languages - French (native), English (fluent), German
- Computer skills - Object-oriented programming (**Java**, Smalltalk, C++), HTML, UNIX

### Publications

*Scientific Journals*

- Th. Zimmermann, P. Bomme, D. Eyheramendy, L. Vernier, S. Commend. **Aspects of an Object-Oriented Finite Element Environment**. *Computers and Structures 68*, p. *1-16*, 1998
- S. Commend, Th. Zimmermann. **Object-Oriented Nonlinear Finite Element Programming: a Primer**. *Accepted for publ. in Advances in Engineering Software*, 2001

*Internal Reports*

- S. Commend, A. Truty, Th. Zimmermann. **Enhanced finite element formulations for nonlinear and ultimate load analysis of geomaterials, with application to large dams sites**. *LSC Internal Report* 99/2, 1999

- S. Commend. **An Object-Oriented Approach to Nonlinear Finite Element Programming**. *LSC Internal Report* 98/7, 1998

- Th. Zimmermann, S. Commend, A. Truty, F. Frey. **Réhabilitation du barrage de la Maigrauge, note de calcul complémentaire**. *LSC Internal Report* 97/6, 1997

*Conference Proceedings*

- Th. Zimmermann, S. Commend, A. Truty, J.-L. Sarf. **Numerical Simulations for Dam Constructions**. *Proc. 10th IACMAG Conf., Tucson, Arizona*, 2001

- Th. Zimmermann, A.Truty, J.-L. Sarf, S. Commend. **Recent Advances in Geotechnical Engineering Software**. *Proc. Congress on Advances in Computer Methods in Geotechnical and Geoenvironmental Engineering (S. Yufin Ed., Balkema), Moscow*, 2000

- Th. Zimmermann, P. Bomme, S. Commend. **Un concept d'objet intelligent pour applications scientifiques**. *4ème Colloque National en Calcul des Structures, Giens*, 1999

- Th. Zimmermann, A. Truty, S. Commend. **A Comparison Of Finite Element Enhancements For Incompressible And Dilatant Behavior Of Geomaterials**. *Proceedings of CIMASI*, 1998

- Th. Zimmermann, D. Eyheramendy, P. Bomme, S. Commend, R.S. Arruda. **Object-Oriented Finite Element Programming: Languages, Symbolic Derivations, Reasoning Capabilities**. *Proceedings of NAFEMS*, 1997

- A. Truty, Th. Zimmermann, S. Commend, A. Urbanski, Y. Li. **Numerical Simulation of Stability and Failure in Elastoplastic Layered Media**. *Proceedings of IACMAG, Wuhan*, 1997

- Th. Zimmermann, S. Commend, P. Roelfstra. **Numerical Simulation Of Underground Works**. *Proceedings of "Journées d'études internationales, AFTES", Chambéry*, 1996

- Th. Zimmermann, P. Bomme, D. Eyheramendy, L. Vernier, S. Commend. **Object Oriented Finite Element Techniques: Towards a General Purpose Environment**. *Proceedings of Civil-Comp '95, Cambridge*, 1995

- Th. Zimmermann, S. Commend, A. Truty, A. Urbanski. **Numerical Simulation Of Underground Openings**. *Proceedings of the 10th European-Conference on Soil Mechanics and Foundation Engineering, Mamaia (Romania)*, 1995

- Th. Zimmermann, Ph. Menétrey, S. Commend. **Numerical Analysis of the Bearing Capacity of Superficial Foundations**. *Proceedings of the Int. Conf. on Computational Methods in Structural and Geotechnical Engineering, Hong-Kong*, 1994

- L. Vernier, S. Commend, Th. Zimmermann. **Object Oriented Programming and Expert Systems Applied to Building Civil Engineering CAL**. *Proceedings of the Int. Conf. on Computer Aided Learning and Instruction in Science and Engineering, Paris*, 1994