REPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère De L'enseignement Supérieur
Et De La Recherche Scientifique

UNIVERSITÉ ECHAHID HAMMA LAKHDAR
D'EL-OUED
FACULTÉ DE LA SCIENCES EXACTES

MÉMOIRE DE FIN D'ÉTUDE

MASTER ACADÉMIQUE

Domaine : Mathématiques et informatique

Filière : Mathématiques

Option : Mathématiques fondamentales

Thème

Résolution d'équations aux dérivées partielles avec FreeFem++.

Présenté par : ALIA Maroua
NISSE Sara

Soutenu publiquement le :08/06/2023, devant le jury composé de :

MEDEKHEL Hamza	Président	MCB Univ. El-Oued
MOUMEN BEKKOUCHE M.	Encadreur	MCB Univ. El-Oued
DOUDI Nadjet	Examinateur	MCB Univ. El-Oued

Promotion : 2022/2023

Dédicaces

A nos chers parents, pour tous leurs sacrifices, leur amour, leur tendance, leur soutien et leurs prières tout au long de nos études.

A nos chers frères et soeurs, pour leur appui et leur encouragement.

A nos chers professeurs qui nous ont accordés les meilleures connaissances.

A nos amis qui nous ont soutenus dans les moments difficiles.

Que ce travail soit l'accomplissement de vos voeux tant allégués, et le fruit de votre soutien infaillible.

Merci d'être toujours lá pour nous.

Remerciments

Avant tout nous remercions ***ALLAH*** *qui nous a orienté au chemin du savoir et aux portes de la science.*

Un remerciement particulier à notre encadreur ***Dr. MOUMEN BEKKOUCHE Mohammed*** *pour la pertinence de ses remarques et sa patience pendant la réalisation de ce travail de recherche.*

Nous remercions également ***Dr. MEDEKHEL Hamza*** *pour avoir accepté de présider.*
Notre jury ***Dr. DOUDI Nadjet*** *pour avoir accepté dexaminer notre travail.*

Nous remercions aussi le corps professoral et administratif de Faculté Des Sciences Exactes pour la richesse et la qualité de leur enseignement, qui déploient de grandes efforts pour assurer à leurs étudiants une formation actualisée.

Nous tenons à remercier toutes les personnes qui ont participé de prés ou de loin à la réalisation de ce travail.

Résumé

FreeFem++ est un logiciel (freeware) écrit en C++, porté sous Windows, Linux et MacOs. Il permet de créer des maillages ainsi que de résoudre des équations aux dérivées partielles par la méthode des éléments finis. L'objectif de ce travail est de trouver des solutions à classes des équations aux dérivées partielles stationnaires et instationnaires en utilisant FreeFem++. On expliquant la méthode de résolution est illustrée à travers des exemples des problèmes variés. On aborde également la recherche de la formulation variationnelle des problèmes, ainsi que les principales commandes utilisées pour écrire le programme de résolution.

Mots clés : FreeFem++, formulation variationnelle, équations aux dérivées partielles, problèmes, stationnaires, instationnaires.

Abstract

FreeFem++ is a software (freeware) written in C++, ported to Windows, Linux and macOS. It makes it possible to create meshes as well as to solve partial differential equations by the finite element method. The objective of this work is to find solutions to classes of stationary and unstationary partial differential equations using FreeFem++. The method of solving is explained and illustrated through examples of various problems. We also discuss the research of the variational formulation of the problems, as well as the main commands used to write the solving program.

Key words: FreeFem++, variational formulation, partial differential equations, problems, stationary, unstationary.

ملخص

فر يفام++ برنامج (مجاني) مكتوب بلغة C++ ، يعمل على ويندوز ، لينكس وماك. يسمح بإنشاء شبكات وكذلك حل المعادلات التفاضلية الجزئية باعتماد طر يقة العناصر المنتهية. الهدف من هذا العمل هو إيجاد حلول لبعض أصناف المعادلات التفاضلية الجزئية المستقرة والتطور ية باستخدام فر يفام++. وقد تم شرح طر يقة الحل وتوضيحها من خلال أمثلة لمشاكل مختلفة. كما تناول البحث الصيغة التغايرية لهذه المسائل ، وكذلك الأوامر الرئيسية المستخدمة لكتابة برنامج الحل .

الكلمات المفتاحية: فريفام++، الصيغة التغايرية، معادلات تفاضلية جزئية، مشاكل، مستقرة، غير مستقرة.

Notations

Dans tout ce qui suit, nous utiliserons les notations suivantes :

EDP	Equations différentielle partielles.
C.L.	Conditions aux limites.
$\mathbb{N}$	L'ensemble des nombres naturels.
$\mathbb{R}$	L'ensemble des nombres réels.
Ω	Un ouvert.
$\partial\Omega$ où Γ	La frontière de Ω.
$\frac{\partial u}{\partial x}$	Dérivée partielle en espace.
$\frac{\partial u}{\partial t}$	Dérivée partielle en temps.
∇u	Gradient en espace : $\nabla u = (\frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_N})^T$.
$divu$	Divergence d'un vecteur $u = (u_1, ..., u_N)^T$: $divu = \sum_{i=1}^{N} \frac{\partial u_i}{\partial x_i}$.
Δu	Laplacien de u : $\Delta u = div(\nabla u) = \sum_{i=1}^{N} \frac{\partial^2 u}{\partial x_i^2}$.
dx	Mesure de Lebesgue de dimension N.
$d\sigma$	Mesure de Lebesgue sur le bord $\partial\Omega$.

Table des matières

Table des figures

Introduction

Les équations aux dérivées partielles représentent une importante outil dans toutes les filières des sciences d'ingérieuries et physiques, où les plus des relations et les lois reliant entre les variables des phénomènes physiques ou d'ingérieurie qui se reflettent sous forme des équations différentielles.

Pour bien comprendre ces dernniers problèmes il est indispensable de résoudre ces équations, du fait que dans plusieurs ces leurs solutions s'avèrent trés difficiles et compliquées. Les savant et les chercheurs ont trouvé plus facile d'utiliser ces programmes pour résoudre ce gendre de problèmes.

FreeFem++ est l'un desprograme qui facilite la résolution de ce gendre de problème en réduisant tout le programe ou il est posé sur la formulation variationnelle du problème.
Ce dernier outil de programation c'est devolopper comme suite :

- 1987 MacFem/PCFem les ancêtres (O. Pironneau en Pascal) payant.
- 1992 FreeFem réécriture de C++ (P1,P0 un maillage) O. Pironneau, D. Bernardi, F. Hecht , C. Prudhomme (adaptation Maillage, bamg).
- 1996 FreeFem+ réécriture de C++ (P1,P0 plusieurs maillages) O. Pironneau, D. Bernardi, F. Hecht (algèbre de fonction).
- 1998 FreeFem++ réécriture avec autre noyau élément fini, et un autre langage utilisateur ; F. Hecht, O. Pironneau, K.Ohtsuka.
- 1999 FreeFem 3d (S. Del Pino) , Une première version de freefem en 3d avec des méthodes de domaine fictif.
- 2008 FreeFem++ v3 réécriture du noyau élément fini pour prendre en compte les cas multidimensionnels : 1d,2d,3d

Dans ce mémoire on va exprimer comment résoudre les équations aux dérivées partielles par FreeFem++. notre travail se présente en trois chapitres :

Dans le premier chapitre on va présenter comment installer FreeFem++, certaines de ses propriétés et le principe de se travail. Aussi, trois méthodes classiques pour la résolution des EDP (méthode des éléments finis, méthode des différences finis, méthode de Galarkin). Enfin, on va présenter des généralités sur EDP et ses classifications.

Dans le deuxième chapitre on va présenter des problèmes stationnaires, les étapes de la solution par Freefem++ et on va étudier le problème de Poisson en détail en se basant sur la classification des EDP (C.L. de Dirichlet "homogène", C.L. de Neuman, C.L. mixtes, C.L. de Fourier). Aussi, on va présenter quelques notes et instructions liées aux commandes de FreeFem++, ainsi que quelques erreurs à éviter lors de la création du code.

Dans le troisième chapitre on va présenté des problèmes instationnaires, et on va expliquer les méthodes de résoudre avec des conditions aux limites différents.

Chapitre 1

Présentation à FreeFem++

Dans ce chapitre, on va présenter une introduction au logiciel FreeFem++, on va expliquer sa définition, et commant l'installe, et ses principes de résoudre.
On va expliquer les commandes importante utilisées de manière répétée pour résoudre les équations aux dérivées partielles.

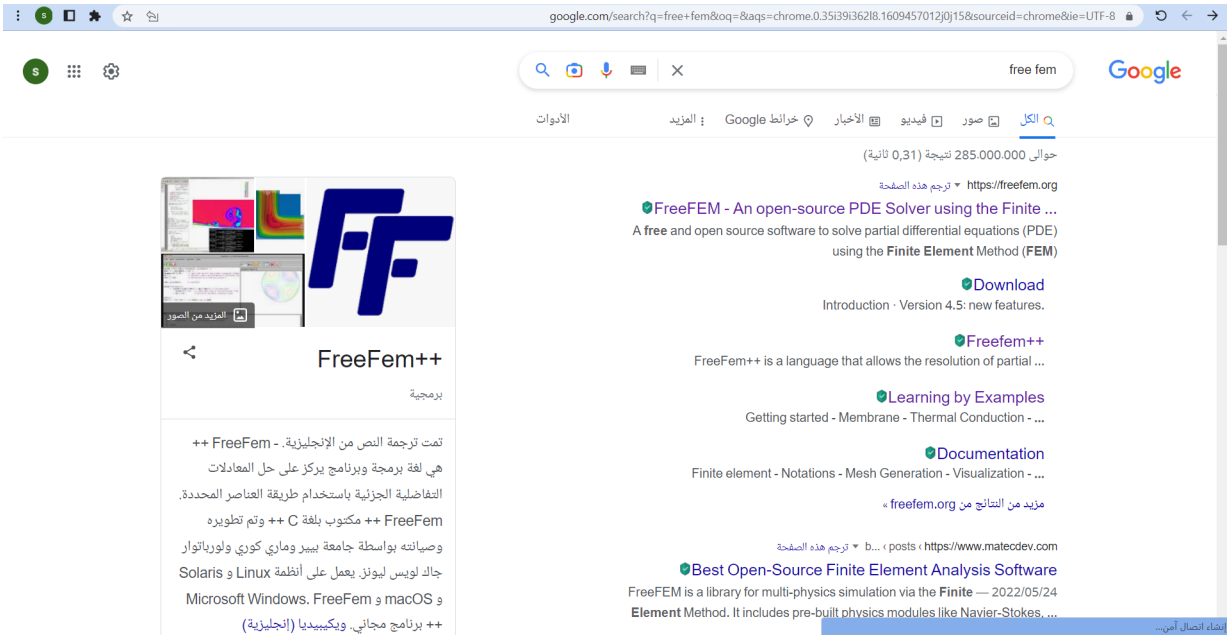
De plus, on va présenter les concepts généraux liés aux équations aux dérivées partielles et à la formulation variationnelle, tout en soulignant les méthodes les plus importantes utilisées pour les résoudre.

1.1 FreeFem++

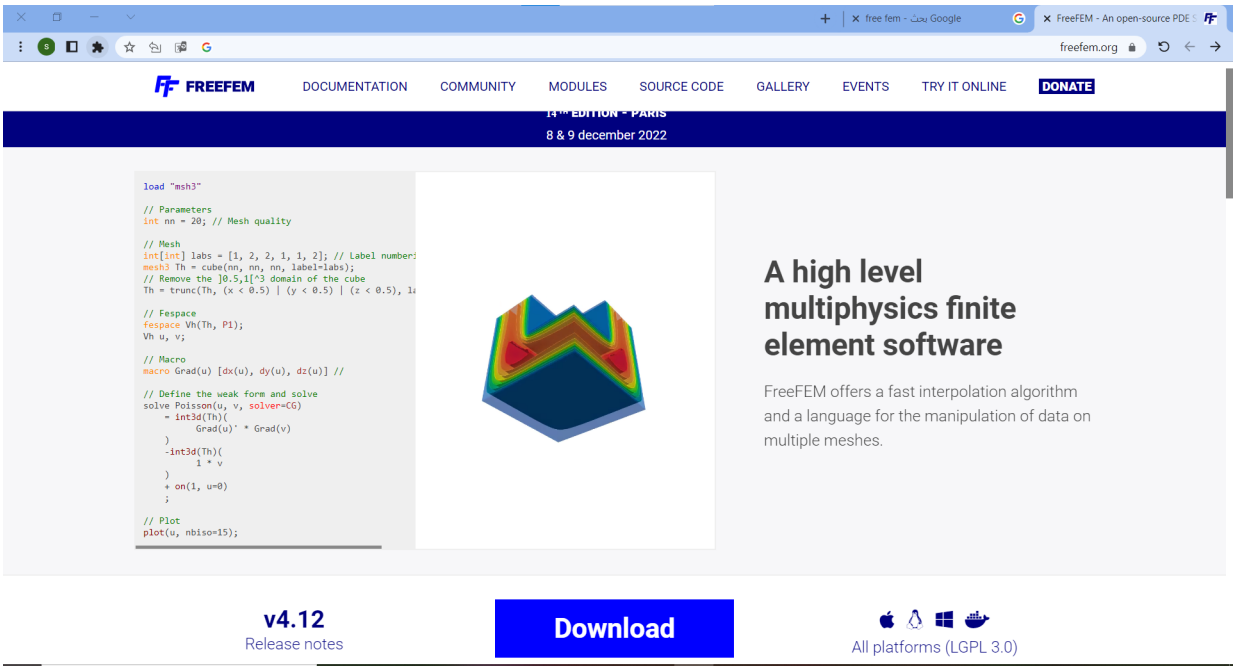
Définition 1.1. *FreeFem++ est un logiciel (freeware) écrit en C++, porté sous Windows, Linux et MacOs. Il s'exécute en ligne de commande. Il permet de créer des maillages ainsi que de résoudre des équations aux dérivées partielles par la méthode des éléments finis.*

1.1.1 Comment l'installer

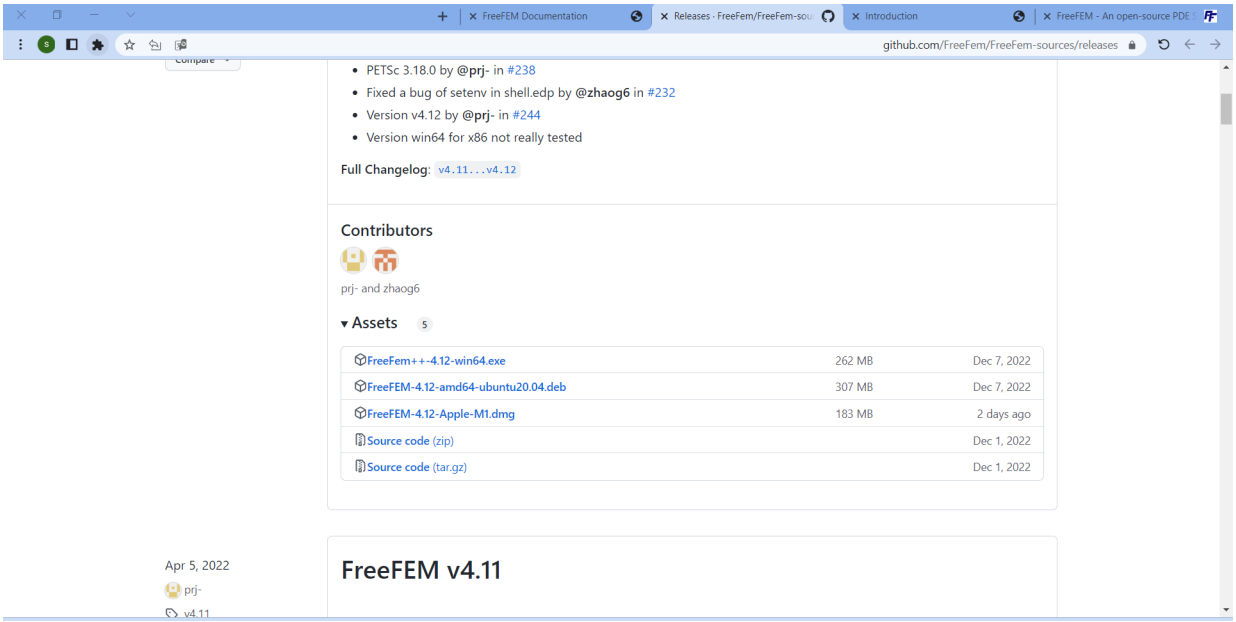
1. Ouvrir le lien :http://www.freefem.org/f++, où la fenêtre suivante apparaît



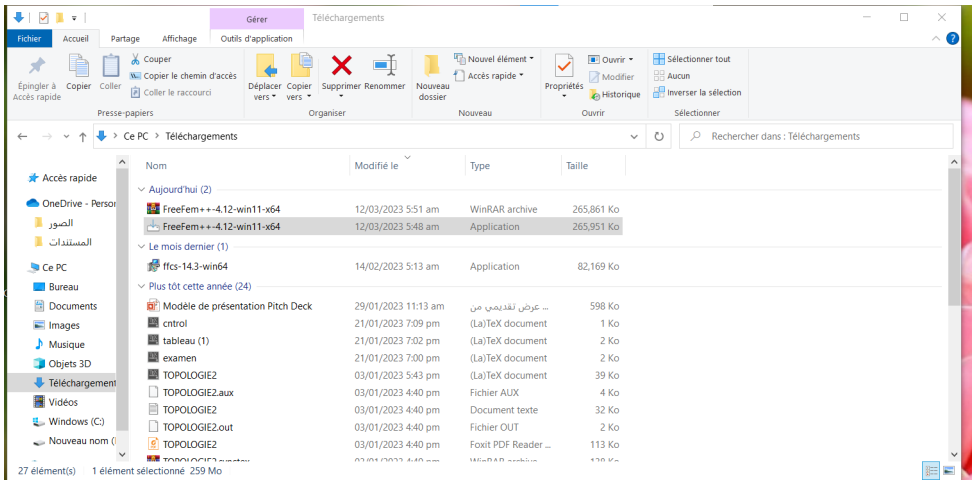
2. Choisir la premiére option parmis les résultats de la recherche et cliquer "Download"



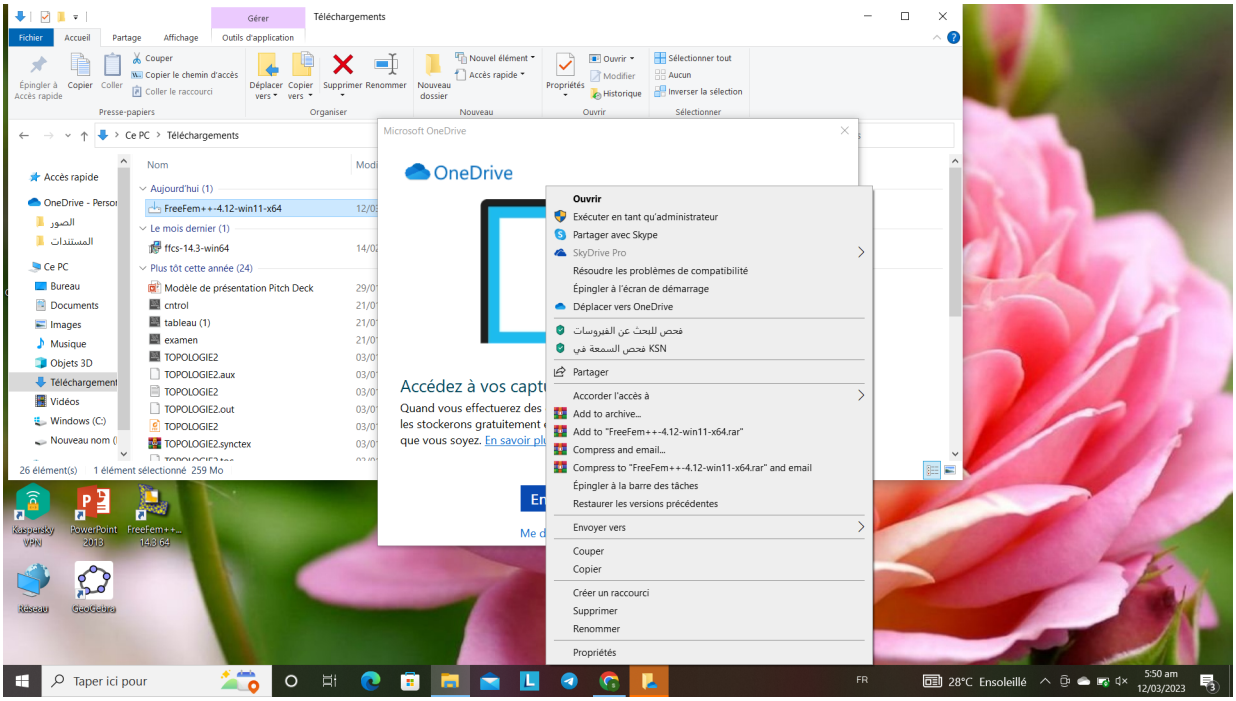
3. Choisir le premier choix pour Windows 64 bits

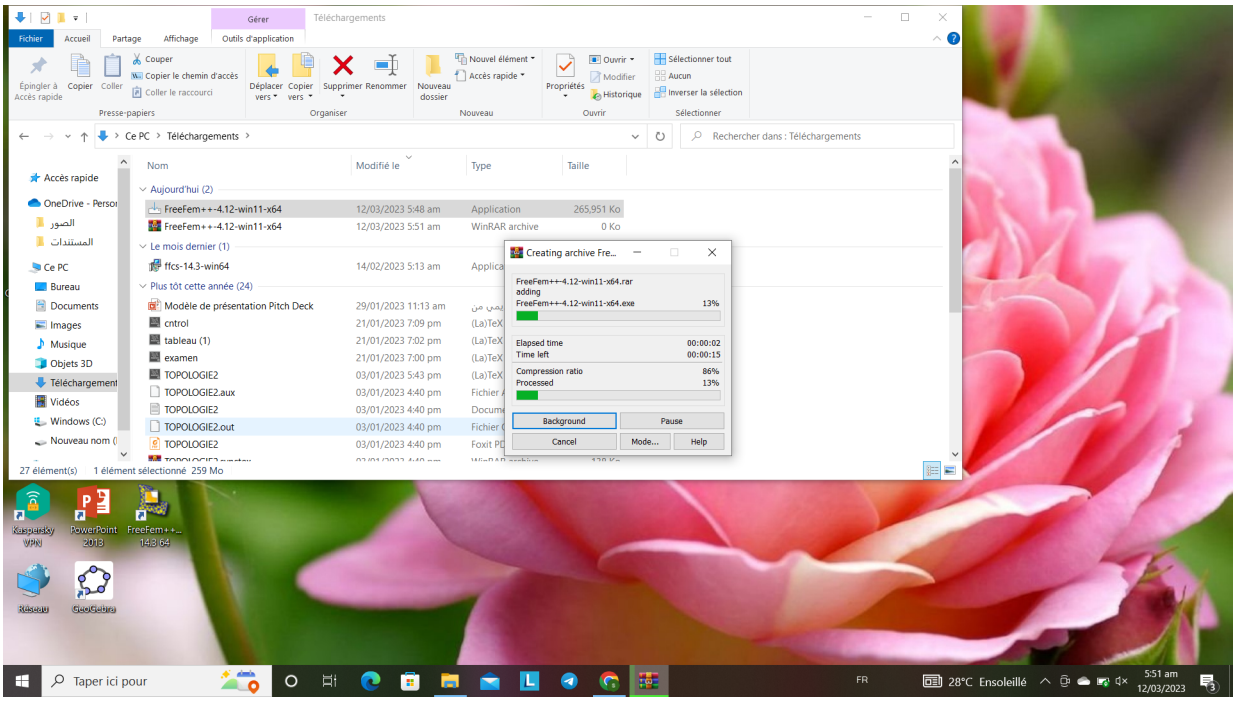


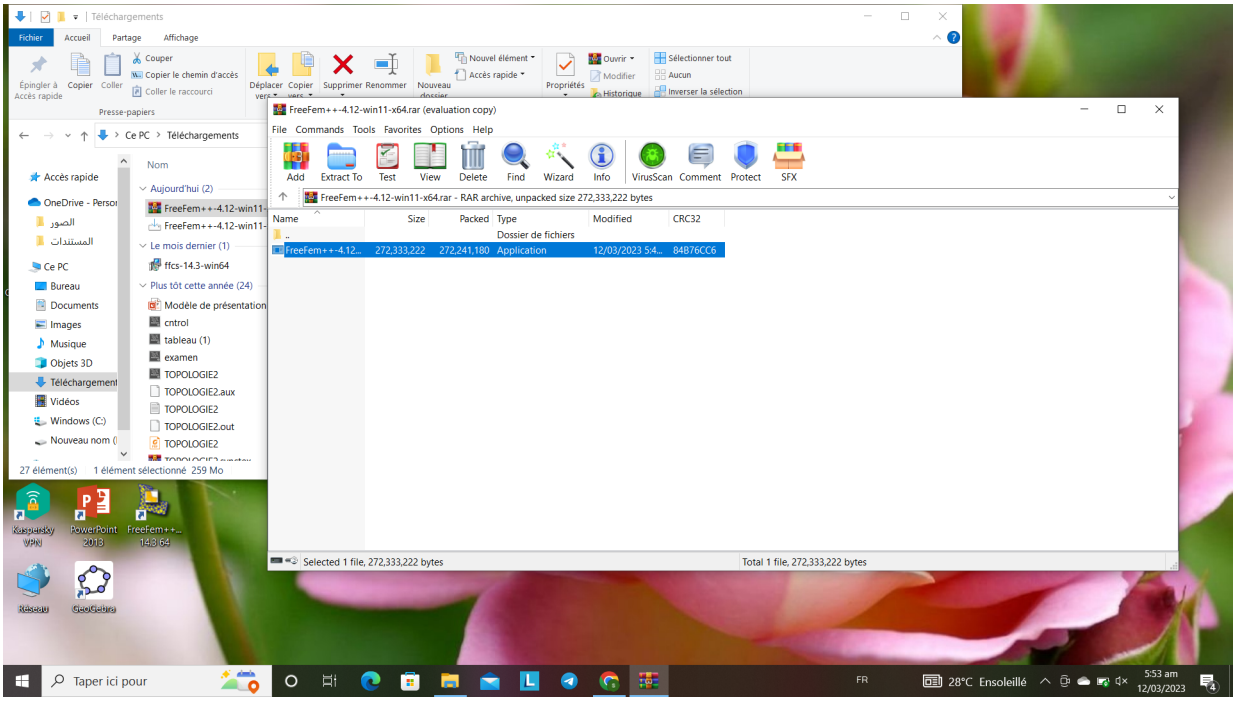
4. Aller à "Téléchargement"

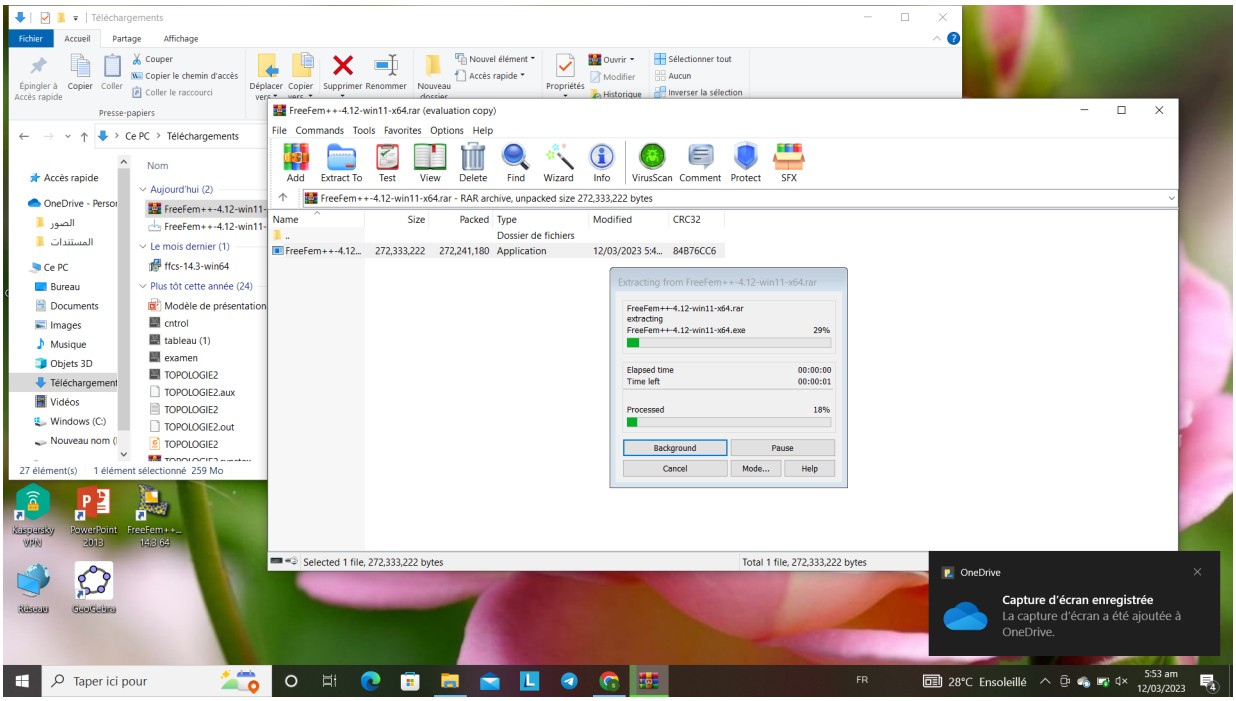


5. Ouvrir leur emballer

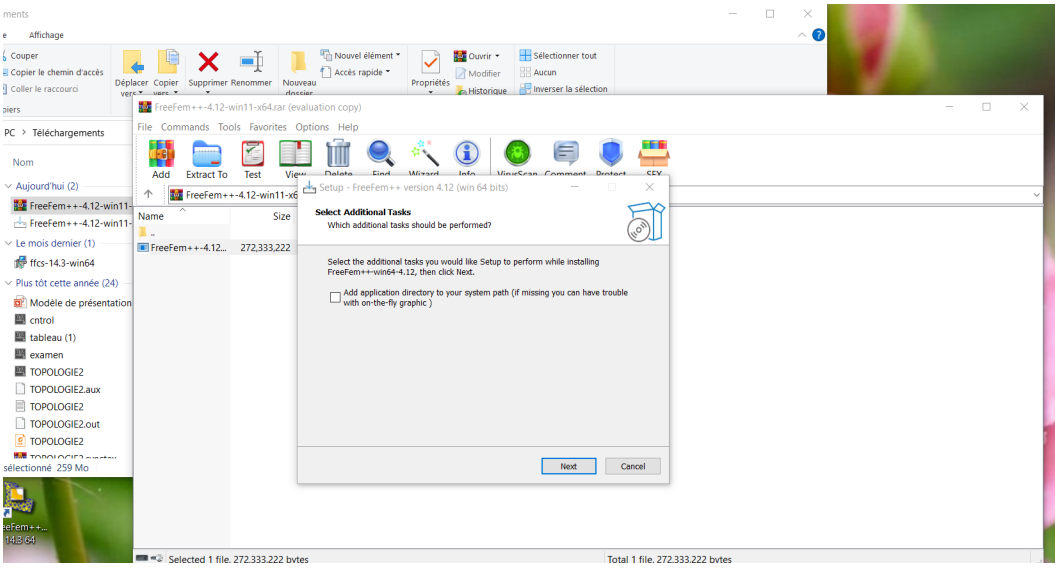



Téléchargements
Creating archive Fre...
FreeFem++-4.12-win11-x64.rar
adding
FreeFem++-4.12-win11-x64.exe 13%
Elapsed time 00:00:02
Time left 00:00:15
Compression ratio 86%
Processed 13%
Background
Pause
Cancel
Mode...
Help
27 élément(s) 1 élément sélectionné 259 Mo
Taper ici pour
28°C Ensoleillé
5:51 am
12/03/2023

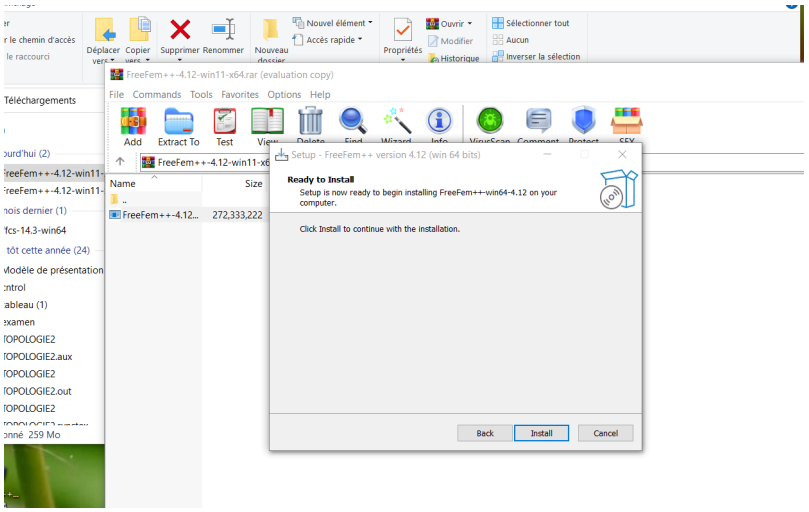

FreeFem++-4.12-win11-x64.rar (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan Comment Protect SFX
FreeFem++-4.12-win11-x64.rar - RAR archive, unpacked size 272,333,222 bytes
Name Size Packed Type Modified CRC32
Dossier de fichiers
FreeFem++-4.12... 272,333,222 272,241,180 Application 12/03/2023 5:4... 84B76CC6
Selected 1 file, 272,333,222 bytes
Total 1 file, 272,333,222 bytes
Taper ici pour
28°C Ensoleillé
5:53 am
12/03/2023

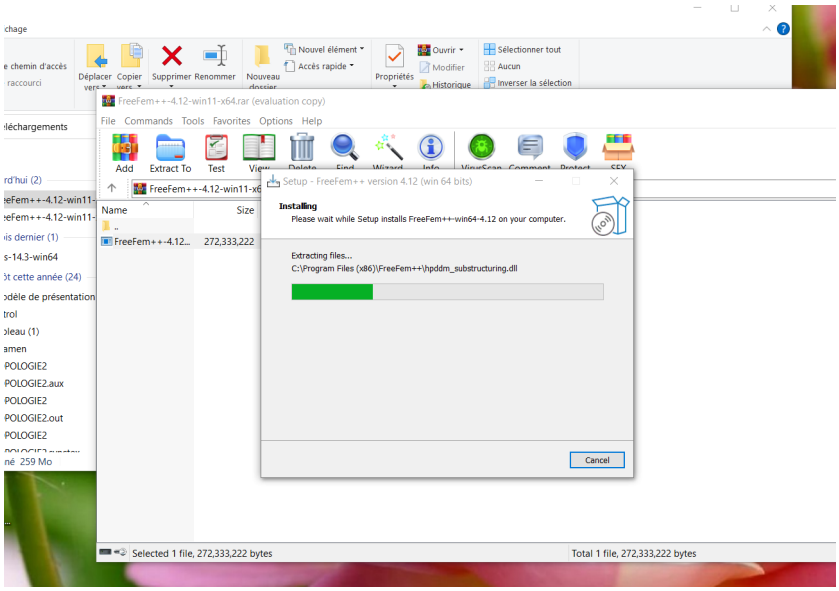


6. Cliquer sur "Next"

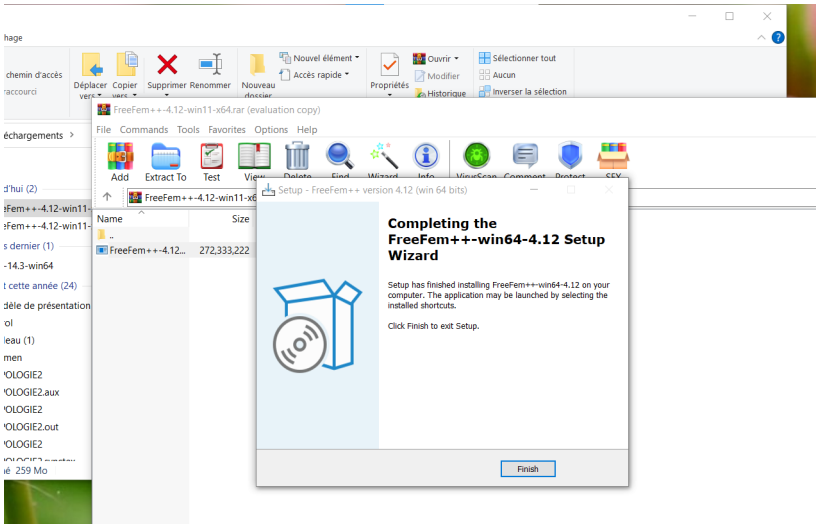


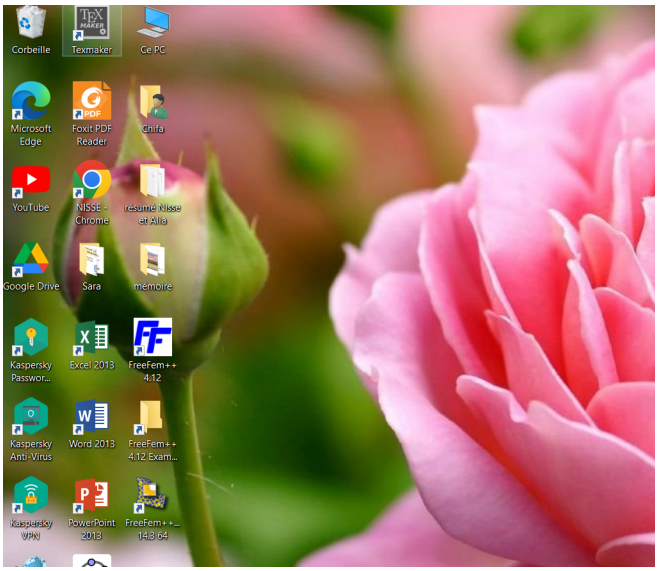
7. Cliquer sur "Install"





8. Cliquer sur "Finish"





Remarque 1.1. Pour installer FreeFem++ à 32 bits, on choisis la copie Windows 32 bits dans le même lien en dessus.

1.2 Quelques méthodes de résoudre de l'EDP

1.2.1 Principe de la méthode des éléments finis

L'approche générale de la méthode des éléments finis est la suivante. Nous avons une EDP à résoudre sur un domaine Ω. On écrit la formulation variationnelle de cette EDP, puis on obtient un problème du type :

$$Trouver \quad u \in V \quad tel\ que \quad a(u,v) = L(v), \forall v \in V \tag{1.1}$$

On essaie ensuite d'approximer u par une fonction linéaire continue par morceaux. A cette fin nous introduisons un maillage sur l'intervalle Ω constitué de n sous-intervalles, et l'espace correspondant $V_h \subset V$ de toutes les linéaires continues par morceaux. Puisqu'il s'agit de fonctions qui s'annulent au extrémités de Ω, alors le problème approximatif est :

$$Trouver \quad u_h \in V_h \quad tel\ que \quad a(u_h, v_h) = L(v_h), \forall v_h \in V_h \tag{1.2}$$

L'espace V_h est de dimension finie, donc le problème 1.2 peut en fait être refondu comme un système linéaire, comme nous le montrons maintenant. Notons N la dimension de V_h et soit $(\varphi_1, ..., \varphi_N)$ une base de V_h. On développe $u_h \in V_h$ sur la base de V_h comme :

$$u_h = \sum_{i=1}^{N} u_i \varphi_i$$

En considérant φ_j comme fonction test, on voit que le problème 1.2 est équivalent à :

$$Trouver \quad u_1, ..., u_N \quad telque \quad \sum_{i=1}^{N} u_i a(\varphi_i, \varphi_j) = L(\varphi_j), \quad \forall j = 1, ..., N \tag{1.3}$$

On introduit ainsi la matrice $A \in \mathbb{R}^{N \times N}$ définie par :

$$A_{ij} = a(\varphi_i, \varphi_j)$$

et le vecteur $b \in \mathbb{R}^N$ défini par :

$$F_j = L(\varphi_j)$$

et reformuler le problème en dessus en tant que système linéaire :

$$\begin{pmatrix} a(\varphi_1,\varphi_1)\cdots a(\varphi_1,\varphi_n) \\ \vdots \qquad\qquad \vdots \\ a(\varphi_1,\varphi_n)\cdots a(\varphi_n,\varphi_n) \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} l(\varphi_1) \\ \vdots \\ l(\varphi_n) \end{pmatrix} \tag{1.4}$$

Sous forme matricielle on écrit ceci :

$$Au = F$$

La matrice A est appellée matrice de rigidité, en référence aux problèmes de mécanique où elle a été introduite pour la première fois. Cette matrice a des propriétés qui proviennent directement des propriétés satisfaites par la forme bilinéaire a, et b comme vecteur de charge.

1.2.2 Principe de la méthode des différences finies

Le principe de la méthode des différences finies consiste à construire un maillage (un ensemble de points x_i pour $N = 1, N = 2$) sur lequel on va appliquer la méthode des différences finies, puis notre problème continue (P_c) devient un problème discret (P_h), la solution numérique de (P_h) revient à résoudre un systeme de type $A_h u_h = b$.

D'abord, on opte à la discrétisation du maillage comme suite :
Dans le cas $N = 1$ **:** $\Omega =]a, b[\,, \partial\Omega = \{a, b\}$ le maillage.

$$x_0 = a \quad x_{S+1} = b \quad x_i = a + ih, \quad 1 \le i \le S, \quad h = \frac{b-a}{S+1}$$

Dans le cas $N = 2$ **:** $\Omega =]a, b[\times]c, d[$ un maillage.

$$x_i = a + ih \quad 1 \le i \le S, \quad h = \frac{b-a}{S+1}$$

$$y_i = c + jk \quad 1 \le j \le T \quad k = \frac{d-c}{T+1}$$

$h = x_{i+1} - x_i$ et $k = y_{j+1} - y_j$ sont les pas de la discrétisation.

Ensuite, on trouve les expressions de la première et seconde dérivées en utilisant des éditions limitées d'Euler pour la formule d'approximation du $\dfrac{\partial u}{\partial x}$ et $\dfrac{\partial^2 u}{\partial x^2}$

Enfin, on étudie la convergence, la consistance et la stabilité du schéma

Proposition 1.1. *On dit que le problème est converge si et sellement s'il est consiste et stable.*

Remarque 1.2. On étudie la consistance et la stabilité pour la norme $\|.\|_\infty$ de notre problème précédent.

1.2.3 Principe de la méthode de Galerkin

La méthode de Galerkin est une méthode très générale et très robuste. Lideé de la méthode est la suivante :
Partant d'un problème posé dans un espace de dimension infinie, on procède dabord à une approximation dans une suite croissante de sous-espaces de dimension finie, on va résoudre ensuite le problème approché, ce qui est en général plus facile que de résoudre directement en dimension infinie.
Enfin, on passe dune façon a dune autre à la limite quand on fait tendre la dimension des espaces dapproximation vers linfini pour construire une solution du problème de départ. Il convient de noter que, outre son intérêt théorique, la méthode de Galerkin fournit également un procédé constructif dapproximation.[10]

1.3 Formulation variationnelle

1.3.1 Les espaces de Sobolev

Soit Ω un ouvert dans $\mathbb{R}^N$.

. L'espace $H^1(\Omega)$:

$$H^1(\Omega) = \{u \in L^2(\Omega), tq \quad \frac{\partial u}{\partial x_i} \in L^2(\Omega) \quad \forall i = 1, ..., N\}$$

On munit $H^1(\Omega)$ du produit scalaire

$$\langle u, v\rangle_{H^1(\Omega)} = \int_\Omega u(x)v(x)dx + \int_\Omega \nabla u(x)\nabla v(x)dx \quad \forall u, v \in H^1(\Omega)$$

. L'espace $H_0^1(\Omega)$:

$$H_0^1(\Omega) = \left\{u \in H^1(\Omega) : \quad \forall \varepsilon > 0, \exists \varphi \in \mathcal{D}(\Omega) \text{tel que} \quad \|u - \varphi\|_{H^1(\Omega)} < \varepsilon\right\}$$

On munit $H_0^1(\Omega)$ du produit scalaire

$$\langle u, v\rangle_{H_0^1(\Omega)} = \int_\Omega \nabla u(x)\nabla v(x)dx \quad \forall u, v \in H_0^1(\Omega)$$

. L'espace $H^m(\Omega)$:

$$H^m(\Omega) = \left\{u \in L^2(\Omega) : \quad D^\alpha u \in L^2(\Omega), \quad \forall \alpha \in \mathbb{N}^N, avec|\alpha| \leq m\right\}$$

On munit $H^m(\Omega)$ du produit scalaire

$$\langle u, v\rangle_{H^m(\Omega)} = \sum_{|\alpha|\leq m} \int_\Omega D^\alpha u D^\alpha v dx \quad \forall u, v \in H^m(\Omega)$$

où D^α est la dérivation au sens des distributions d'ordre α.

Remarque 1.3. $H_0^1(\Omega)$ est sous-espace de $H^1(\Omega)$ des fonctions qui s'annulent sur la frontière de Ω.

1.3.2 Formule de Green

Théorème 1.1. *Soit Ω un ouvert borné dans $\mathbb{R}^N$ de classe C^1, et $\nu(x) = (\nu_1, ..., \nu_N)$ le vecteur normale unitaire extérieur à Ω.*
Alors,$\forall u, v \in H^1(\Omega)$ on a :

$$\int_\Omega \frac{\partial u}{\partial x_i} v dx = \int_\Gamma u_{|\Gamma} v_{|\Gamma} \nu_i d\sigma - \int_\Omega u \frac{\partial v}{\partial x_i} dx \quad \forall i = 1, ..., N$$

En particulier, si $u \in H^2(\Omega)$et $v \in H^1(\Omega)$ on a :

$$\int_\Omega v \Delta u dx = \int_\Gamma \frac{\partial u}{\partial \nu}_{|\Gamma} v_{|\Gamma} d\sigma - \int_\Omega \nabla u \nabla v dx$$

Si de plus, u et $v \in H^2(\Omega)$, on a :

$$\int_\Omega (v\Delta u - u\Delta v) dx = \int_\Gamma \left(\frac{\partial u}{\partial \nu}_{|\Gamma} v_{|\Gamma} - u_{|\Gamma} \frac{\partial v}{\partial \nu}_{|\Gamma} \right) d\sigma$$

1.3.3 Formulation variationnelle

La formulation variationnelle également connue sous le nom de formulation faible permet de trouver de manière simple et rapide la solution à des phénomènes ou problèmes modélisés à travers des EDP, ceux-ci lorsqu'ils sont analysés avec les techniques ou la théorie classique des EDP, il est très complexe de trouver une solution qui satisfaisant la dite équation.
Considérons le problème suivant :

$$(P) \begin{cases} Pu = f, \text{sur} \Omega \in \mathbb{R}^N \\ C(u)|_\Gamma = g \end{cases}$$

telque P : opérateur différentiel
$C(u)|_\Gamma$: conditions au bord de $\Omega(\Gamma)$
Pour passer au problème au sens faible :

$$(P_v) \begin{cases} u \in V \\ a(u, v) = L(v), \forall v \in V \end{cases}$$

On on va le transformer à la formulation variationnelle

1. Multiplier l'équation du problème (P)pour une fonction test, intégrer par parties et utiliser les conditions aux bords appropriées .
2. Identifier l'espace de Hilbert V, la forme bilinéaire $a(.,.)$ et la forme linéaire$L$.
3. Vérifier, si possible, les hypothèses du théoreme de Lax-Milgram alors il éxiste une unique solution faible du problème (P).

Théorème 1.2 (Lax-Milgram)**.** *Soit V un espace de Hilbert : $(V, <.,.>)$.*
Suposons que : $a(.,.) : V \times V \longrightarrow \mathbb{R}$ est une forme bilinéaire, et $L : V \longrightarrow \mathbb{R}$ une forme linéaire, telles que :

1. *$a(.,.)$ est continue : $\exists c > 0$, tel que $\forall a(u, v) \in V$, on a $|a(u, v)| \leq c\|u\|.\|v\|$*
2. *$a(.,.)$ est coercive : $\exists \alpha > 0$, tel que $\forall u \in V$, on a $a(u, u) \geq \alpha\|u\|^2$*
3. *L est continue (donc bornée) : $\exists M > 0$, tel que $\forall u \in V : |L(u)| \leq M\|u\|$*

Alors, il existe un unique $u \in V$ tel que $\forall u \in V$, on a $a(u, v) = L(v)$.

1.4 Comment l'utiliser

1.4.1 Principe de fonctionnement de FreeFem++

1. Génère des maillages.
2. Construire automatiquement la matrice M (appelée Matrice Masse/ Rigide).
3. Construire automatiquement un deuxiéme membre.
4. Utiliser plongeurs solveurs linéaires (intégrés).
5. Travailler sur des problemes 2D ou 3D.
6. Générer des graphiques/textes/fichiers.

1.4.2 Commandes de base

1. Déclaration de variable.
2. Définir le domaine.
3. Définir le maillage.

1.4.3 Quelques commandes

. **A propos des maillages :**
`mesh` = type pour ... mailles ;
`square` = pour créer des maillages structurés ;
`savemesh` = enregistrer un maillage dans un fichier ;
`readmesh` = lire le maillage précédemment sauvegardé ;
`border` = type pour les courbes paramétrées ;
`buildmesh` = construire un maillage avec des frontières paramétrées ;
`adaptmesh` = raffiner un maillage .

. **A propos des formulations variationnelles :**
`fespace` = type pour les espace d'éléments finis ;
`problem` = type pour un problème variationnel ;
`P1,P2,P3` = éléments de Lagrange ;
`int2d` = intégration sur un domaine ;
`int1d` = intégration sur une partie de la frontière ;
`on` = pour un poser des conditions aux limites ;
`func` = type pour une fonction de x et y ;
`int` = type pour les entiers ;
`plot` = just deviner ;
`include` =pour inclure un script dans un autre ;
`ofstream` = pour diriger la sortie vers un fichier ;
`ifstream` = pour obtenir l'entrée d'un fichier ;

. **Divers :**
`macro` = pour définir une macro, se termine par un commentaire ;
`for` = pour faire une boucle ;
$[P1, P1]$ = produit des espaces Éléments Finis ;
`P0` = Éléments Finis ;
`movemesh` = déplacer un maillage ;

`init` = construire la matrice ou non ;
`real` = type pour les réels ;
`fill` = remplir entre les isolignes ;
`cmm` = commentaires pourle tracé ;
`wait` = attendre avant le prochain tracé ;
`u[]` =si u est un élément fini, renvoie les valeurs de u à chaque degré de libreté .

Remarque 1.4. Toutes les commandes sont finis par point vérgule(;).

1.4.4 Comment l'utiliser

1. Utiliser un éditeur de texte pour écrire votre script.
2. L'enregistrer à l'extention de fichier **.edp(toto.edp)**.
3. L'exécuter en cliquant sous dessus Windows (aussi MacOs), ou en exécutant la commande FreeFem++toto.edp sous Linux.

Remarque 1.5. Comme la plupart des logiciels, une documentation complète et de nombreux exemples sont inclus dans FreeFem++.

1.4.5 Liste des types des élément finis

[`P0`] : élément fini discontinu constant par morceaux ($2d$, $3d$, surface $3d$) , les degrés de liberté sont les valeurs de l'élément barycentre.

$$\mathbb{P}_h^0 = \{v \in L^2(\Omega) | \forall k \in \mathcal{T}_h, \exists \alpha_k \in \mathbb{R} : v_{|k} = \alpha_k\}$$

[`P1`] : élément fini continu linéaire par morceaux ($2d$, $3d$, surface $3d$) , les degrés de liberté sont des valeurs sommets.

$$\mathbb{P}_h^1 = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_1\}$$

[`P1dc`] : élément fini discontinu linéaire par morceaux ($2d$, $3d$ avec charge `"Element_ P1dc"`).

$$\mathbb{P}_{dc|h}^1 = \{v \in L^2(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_1\}$$

[`P1b`] : élément fini continu linéaire par morceaux plus bulle $(2d, 3d)$.
cas de $2D$:

$$\mathbb{P}_{b|h}^1 = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_1 \oplus Span\{\lambda_0^k \lambda_1^k \lambda_2^k\}\}$$

cas de $3D$:

$$\mathbb{P}_{b|h}^1 = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_1 \oplus Span\{\lambda_0^k \lambda_1^k \lambda_2^k \lambda_3^k\}\}$$

Où $\lambda_i^k, i = 0, ..., d$ sont des fonction de coordonnées barycentriques $d+1$ d'élément K (triangle ou tétraédre) .
[`P1b1,P1b13d`] : élément fini continu linéaire par morceaux plus bulle linéaire (avec charge `"Element _ P1b12d, 3d"`.La bulle est construite en divisant le K, un barycentre en sous-élément $d+1$. (besoin de charger `"Element_ P1b1"`).
[`P2,P2`] : élément fini continu P_2 par morceaux (2d, 3d, surface 3d) .

$$\mathbb{P}_h^2 = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_2\}$$

[`P2b,P2b3d`] :

cas de $2D$:

$$\mathbb{P}^2_{b|h} = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_2 \oplus Span\{\lambda_0^k \lambda_1^k \lambda_2^k\}\}$$

cas de $3D$:

$$\mathbb{P}^2_{b|h} = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_2 \oplus Span\{\lambda_0^k \lambda_1^k \lambda_2^k \lambda_3^k\}\}$$

[P2dc] : élément fini discontinu P_2 par morceaux $(2d)$.

$$\mathbb{P}^2_{dc|h} = \{v \in L^2(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_2\}$$

[P2h] : continu homogène quadratique (sans $P1$) .
[P3] : élément fini continu $P3$ par morceaux $(2d)$. (nécessite la charge `"Element_ P3"`).

$$\mathbb{P}^3_h = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_3\}$$

[P3dc] : élément fini discontinu P_3 par morceaux $(2d)$ (nécessite la charge `"Element_ P3dc"`) .

$$\mathbb{P}^3_{dc|h} = \{v \in L^2(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_3\}$$

[P4] : élément fini continu $P4$ par morceaux $(2d)$ (nécessite la charge`"Element_ P4"`).

$$\mathbb{P}^4_h = \{v \in H^1(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_4\}$$

[P4dc] : élément fini discontinu P_4 par morceaux $(2d)$ (nécessite la charge `"Element_ P4dc"`).

$$\mathbb{P}^4_{dc|h} = \{v \in L^2(\Omega) | \forall k \in \mathcal{T}_h, v_{|k} \in P_4\}.$$

Où :

- P_1 est l'ensemble des polynômes dans $\mathbb{R}^2$ de degré ≤ 1 .
- P_2 est l'ensemble des polynômes dans $\mathbb{R}^2$ de degré ≤ 2 .
- P_3 est l'ensemble des polynômes dans $\mathbb{R}^2$ de degré ≤ 3 .
- P_4 est l'ensemble des polynômes dans $\mathbb{R}^2$ de degré ≤ 4.

1.5 Généralités sur les équations aux dérivées partielles

Définition 1.2. *Une EDP est une relation entre l'inconnue u sur$\Omega \subset \mathbb{R}^N$ et les variables $x_1, x_2, ..., x_N$ et les dérivées partielles de u.*
La formme générale d'une EDP peut s'écrire :

$$F\left(u, \frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_N}, \frac{\partial^2 u}{\partial x_1^2}, ..., \frac{\partial^m u}{\partial x_N^m}\right)(x) = h(x) \tag{1.5}$$

Chaque fonction vérifie la relation 1.5 s'appelle une solution à 1.5.

Exemple 1.1. $\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial^2 u}{\partial y^2} = 0 \qquad \Omega \subset \mathbb{R}^2$
$u(x, y) = (x + y)$ est une solution à l'équation si-dessus.

Définition 1.3. L'ordre d'une EDP c'est l'ordre le plus élevé de la dérivée partielle de la relation 1.5.

Exemple 1.2. $\dfrac{\partial u}{\partial x} - \dfrac{\partial^2 u}{\partial x \partial y} = 1$ l'ordre= 2

$\left(\dfrac{\partial u}{\partial x}\right)^2 - \dfrac{\partial u}{\partial x} = x\sin(x-y)$ l'ordre= 1

Définition 1.4. *On dit qu'une EDP est **linéaire** si et sellement si l'équation peut s'écrire comme : $L(u) = f(x)$ et L une opérateur linéaire par raport à u et ses dérivées partielles.*

Exemple 1.3. 1. $\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial u}{\partial y} + u - 6xy = 0$ $L(u) = \dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial u}{\partial y} + u$

cette EDP est linéaire, $\forall u, v, \forall \alpha, \beta \in \mathbb{R}$:

$$\begin{aligned} L(\alpha u + \beta v) &= \alpha \frac{\partial^2 u}{\partial x^2} + \beta \frac{\partial^2 v}{\partial x^2} - \left(\alpha \frac{\partial u}{\partial y} + \beta \frac{\partial v}{\partial y}\right) + \alpha u + \beta v \\ &= \alpha \left(\frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial y} + u\right) + \beta \left(\frac{\partial^2 v}{\partial x^2} - \frac{\partial v}{\partial y} + v\right) \\ &= \alpha L(u) + \beta L(v) \end{aligned}$$

2. $u\dfrac{\partial^2 u}{\partial x^2} = 1 - x$ (non linéaire).

Définition 1.5 (Conditions aux limites)**.** .

- *Une condition de **Dirichlet** est une condition où on impose la valeur de u sur $\partial\Omega$ (le bord)(condition du type 1).*
- *Une condition de **Neumann** est une condition où on impose la valeur de la dérivée normale deu sur $\partial\Omega$ (le bord) c'est-a-dire la valeur de $\frac{\partial u}{\partial x}$sur $\partial\Omega$ (condition du type 2).*
- *Une condition de **Fourier Robien** est une condition où on impose une relation entre les conditions de Dirichlet et Neuman (condition du type 3)*

1.5.1 EDP du second ordre dans $\mathbb{R}^N$

Une EDP linéaire du second ordre dans $\mathbb{R}^N$ s'écrit sous la forme générale suivante :

$$\sum_{i,j=1}^{N} a_{ij}(X)\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{N} f_i(X)\frac{\partial u}{\partial x_i} + g(X) = h(X)$$

$X = (x_1, x_2, ..., x_N) \in \mathbb{R}^N$ où $A = (a_{ij})_{1\le i,j\le N}$ est par convention est **symétrique**.
Par exemple dans $\mathbb{R}^2$

$$a_{11}\frac{\partial^2 u}{\partial x^2} + 2a_{12}\frac{\partial^2 u}{\partial x \partial y} + a_{22}\frac{\partial^2 u}{\partial y^2} + f_1\frac{\partial u}{\partial x} + f_2\frac{\partial u}{\partial y} + gu = h$$

En générale :

$$a(x,y)\frac{\partial^2 u}{\partial x^2} + 2b(x,y)\frac{\partial^2 u}{\partial x \partial y} - c(x,y)\frac{\partial^2 u}{\partial y^2} + d(x,y)\frac{\partial u}{\partial x} + e(x,y)\frac{\partial u}{\partial y} + g(x,y)u = h(x,y)$$

$A = \begin{pmatrix} a(x,y) & b(x,y) \\ b(x,y) & c(x,y) \end{pmatrix}$ la matrice symétrique.

1.5.2 Classification d'une EDP linéaire dans $\mathbb{R}^N$

(Dans $\mathbb{R}^N$)

- EDP est **elliptique** si et sellement si toutes les valeurs propres de A sont non nulles et de même signe.
- EDP est **hyperbolique** si et sellement si toutes les valeurs propres de A sont non nulles et de même signe sauf une valeur de signe opposé.
- EDP est **parabolique** si et sellement si toutes les valeurs propres de A sont non nulles et de même signe et une valeur est nulle.

Cas particulier : (pour $N = 2$)

- Si $b^2 - ac < 0$, l'EDP est elliptique.
- Si $b^2 - ac > 0$, l'EDP est hyperbolique.
- Si $b^2 - ac = 0$, l'EDP est parabolique.

Exemple 1.4. $X = (x, t)$

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} - c^2 \Delta u = 0 & X \in \mathbb{R} \times \mathbb{R}_+^* \\ \dfrac{\partial u}{\partial t}(x, 0) = f(x) \\ u(x, t) = f(x) \end{cases}$$

$-c^2 \dfrac{\partial^2 u}{\partial x^2} + (1 - c^2) \dfrac{\partial^2 u}{\partial t^2} = 0$

- Si $c > 1$ notre EDP elliptique.
- Si $c < 1$ notre EDP hyperbolique.
- Si $c \in \{-1, 0, 1\}$ l'EDP parabolique.

Chapitre 2

Problèmes stationnaires

Dans ce chapitre on va exprimer comment résoudre les équations et les problèmes différenttielles stationnaires par FreeFem++.
Nous avons donné aussi quelques exemples d'EDP avec des conditions différentes.

On va exprimer pour trouver la solution d'EDP par FreeFem++, on va définir la frontière du domaine d'étude du problème par des courbes, aprés on chiosissant une discritisation de courbe qui détermine la frontière. Cette frontière, celle-ci décrit une surface en maille la pour obtenir un maillage approprié.

2.1 Les étapes du solution par FreeFem++

2.1.1 Construction du maillage

Définition 2.1 (La frontière)**.** *celle-ci est faite de réunion de courbes fournies sous forme paramétrique.*

(a) **Définition paramétrique de chaque portion de la frontière :** Une courbe parametrique étant définie par exemple comme : $[t_i, t_f] \ni t \longmapsto \begin{cases} x = f(t) \\ y = g(t) \end{cases}$

(b) **Discrétisation des portions et leurs réunions :** On détérmine des plusieurs points sur chaque courbe et on réuni ces portions de courbes pour définir la frontière du domaine. Ainsi, on peut par exemple dessiner la frontière du domaine ci-dessus par le bout de code.

2.1.2 Maillage de la surface délimitée

On maille la surface decrite par la frontière. Ceci suppose évidemment que la frontière décrite précédemment doit étre une courbe fermé orientée de sorte que la surface délimitée se trouve à gauche lorsquon parcourt le courbe.
Aprés, pour définir l'espace d'élément finis on étulise le maillage qui la construit, et on chiosis le type d'élément de Lagrange pour priciser l'espace .
Dans l'étape suivante nous avons entré la formulation variationnelle du problème par la commande "`int2d` " pour intégre dans le dommaine et le "`int1d` " pour intégrer dans la frontière, et on ajoute les conditions initiales.

2.2 Étude problème de Poisson

C'est l'équation aux dérrivées partielles elliptique du second ordre suivante : c'est une distribution générallement donnée.

Remarque 2.1. L'inclusion des conditions aux limites sous la forme variationnelle dans le code FreeFem++ diffère selon la classification des EDP, et se présente comme suite(généralement) :

. Une forme scalaire "on"(pour Dirichlet)

```
1 on(label, la condition)
```

. Une forme linéaire sur Γ(pour Neuman dans 2D)

```
1 -int1d(le maillage)(la condition)
```

ou

```
1 -int1d(le maillage,label)(la codition)
```

. Une forme bilinéaire sur Γ(pour Robin dans 2D)

```
1 int1d(le maillage)(la condition)
```

ou

```
1 int1d(le maillage,label)(la codition)
```

Comme on va voir plus tard.

2.2.1 Conditions aux limites de Dirichlet "homogène"

$$\begin{cases} -\Delta u = f \quad \text{sur} \quad \Omega \\ u = 0 \quad \text{sur} \quad \partial\Omega \end{cases} \tag{2.1}$$

On va trouver $u \in H_0^1(\Omega)$ tel que $a(u,v) = l(v), \forall v \in H_0^1(\Omega)$
$a(u,v)$ est une forme bilinéaire continue et coercive sur $H_0^1(\Omega) \times H_0^1(\Omega)$
et $L(v)$ est une forme linéaire continue sur $H_0^1(\Omega)$
on a :
$-\Delta u = f$, soit $v \in H_0^1(\Omega)$
Nous multiplions les deux cotés par v aprés nous intégrons sur Ω

$$-\Delta u \cdot v = f \cdot v$$

$$\text{alors,} \int_\Omega -\Delta u \cdot v dx = \int_\Omega f \cdot v \; dx$$

maintenant nous avons utiliser la formule de Green, alors on a :

$$\int_\Omega -\Delta u \cdot v dx = \int_\Omega \nabla u \nabla v dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} v \partial\sigma$$

tel que $\sigma \in \partial\Omega$ et $\overrightarrow{n}$: c'est le vecteur normale unitaire exterieur à Ω

$$\text{c'implique} \int_\Omega \nabla u \nabla v dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} v \partial\sigma = \int_\Omega f \cdot v \ dx$$

Car, $u = 0$ sur $\partial\Omega$ alors, $\int_{\partial\Omega} \frac{\partial u}{\partial n} v \partial\sigma = 0$

$$\text{donc,} \int_\Omega \nabla u \nabla v dx = \int_\Omega f \cdot v \ dx$$

Prend $a(u,v) = \int_\Omega \nabla u \nabla v dx$ et $l(v) = \int_\Omega f \cdot v \ dx$

Exemple 2.1. Considérons le problème suivant :

$$\begin{cases} -\Delta u = f \ \text{ sur } \Omega =]0,1[\times]0,1[\\ u = 0 \ \text{ sur } \partial\Omega = \{0,1\} \times [0,1] \cup [0,1] \times \{0,1\} \end{cases} \tag{2.2}$$

tel que $f = cos(x)y$.
La formulation variationnelle de ce problème est :

$$\int_\Omega \nabla u \nabla v dx = \int_\Omega f \cdot v \ dx$$

On va résoudre se problème par FreeFem++ :

1. on va construire le maillage :
   Comme le bord est un carré on définie la frontière par "`square`", et on choisis le nombre de noeuds sur la frontière(on tend à faire la discrétisation de la frontière 10) pour obtenir le maillage .
   (ligne 1)
2. On définis l'espace d'élément finis :
   On va prendre les éléments de cet éspace comme des polynôme de Lagrange de dégrée ≤ 1 .
3. On définit le problème, et on l'égalise sous sa formulation variotionnelle .
4. On ajoute les conditions aux limites, dans ce cas : condition de Dirichlet .
5. Pour résoudre le problème il suffit décrire son nom suivi de point-virgule.

Le code de ce problème

```
1 mesh Sh= square(10,10) ;    // génération du maillage de surface
2 fespace Vh(Sh,P1) ;     // espace de P1 élément fini (On utilise le maille du
     carre et l'élément fini P1)
3 Vh u,v ;     // u et v des élément dans Vh
4 func f=cos(x)*y ;     // f est la fonction du variables x et y
5 problem Poisson(u,v)=    // Définition de problème
6 int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))     // form bilinéaire
7 -int2d(Sh)(f*v)    // form linéaire
8 +on(1,2,3,4,u=0) ;     // Conditions de Dirichlet
```

```
9  Poisson ;      // Solution de problème de Poisson
10 plot(Sh) ;     //Dessin du maillage
11 plot(u)  ;     // Dessin de la solution
```

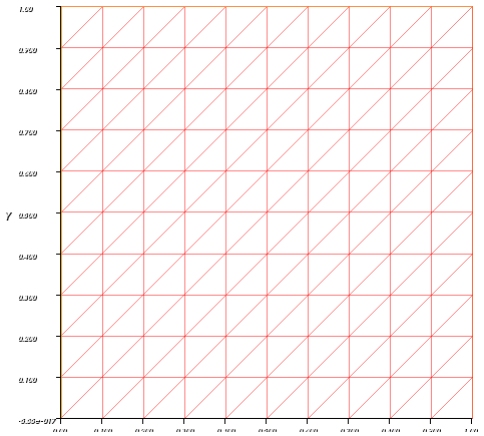


FIGURE 2.1 – Maillage

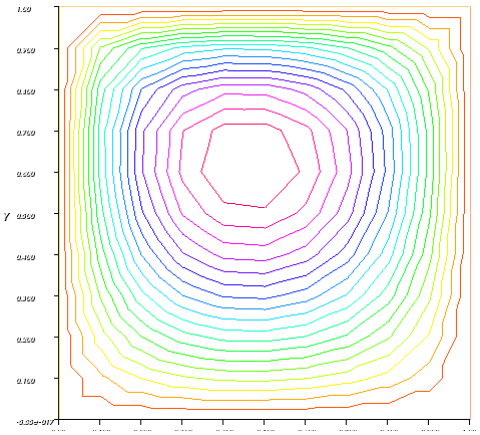


FIGURE 2.2 – Solution en utilisant `P1`

Remarque 2.2. Quand on utilise les polynômes de Lagrange de dégré plus grand, on obtient une solution plus exacte. Par exemple on change `P1` par `P2` dans la ligne 2 on obtient la solution comme suite

```
1 fespace Vh(Sh,P2) ;     // espace de P2 élément fini
```

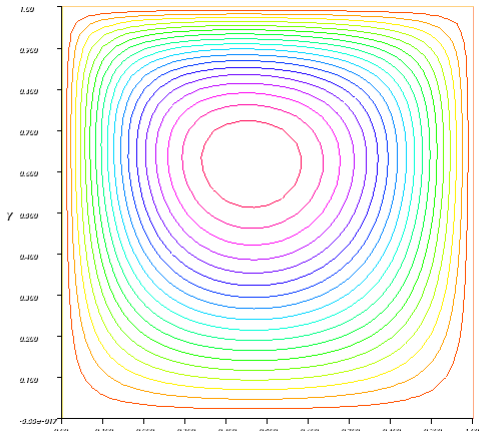


FIGURE 2.3 – Solution en utilisant P2

Exemple 2.2. Dans le même problème quand on change la surface d'étude par le disque unité, alors on va définir le bord par une courbe paramétrique. Autrement dit, on va écrire l'équation de la frontière de disque.
Pour préciser le bord et construire le maillage dans Freefem++ il faut :

1. définir le courbes et leur donner leurs noms .
2. préciser le domaine de t.
3. écris x et y en fonction de t .
4. définir le bord par la commande "`border`" comme suivant :

```
1 border nom de courbe (t= t0 , T)  { x= x(t); y= y(t);};
```

Pour construire le maillage on utilise la commande "`buildmesh`", on écrit :

```
1 mesh nom de maillage = buildmesh (nom de courbe (z) );
```

tel que z est le nombre de noeuds sur le bord.

Le code de ce problème

```
1  border h(t=0,2*pi) { x=cos(t); y=sin(t);};    //le bord de domaine
2  mesh Th=buildmesh(h(30));     // maillage de 30 points dans le bord
3  fespace Vh(Th,P1) ;     // espace de P1 élément fini
4  Vh u,v ;     //u et v des élément dans Vh
5  func f=cos(x)*y ;     // f est la fonction du variables x et y
6  problem Poisson(u,v)=     // Définition de problème
7  int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))     // form bilinéaire
8  -int2d(Th)(f*v)     // form linéaire
9  +on(h,u=0) ;     // Conditions de Dirichlet
10 Poisson ;     // Solution de problème de Poisson
11 plot(Th);     // Dessin de maillage
12 plot(u) ;     // Dessin de la solution
```

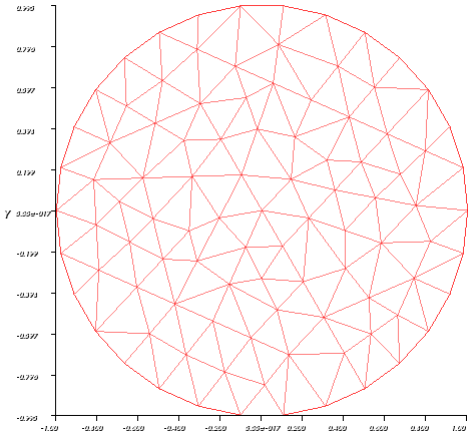


FIGURE 2.4 – Maillage avec 30 points dans le bord

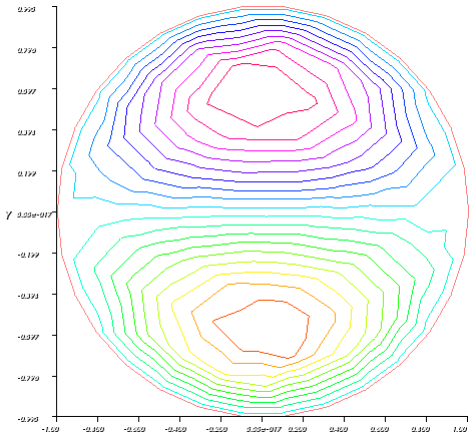


FIGURE 2.5 – Solution avec 30 points dans le bord

Remarque 2.3. Quand on maille à plus de points, on obtient une solution plus précise.

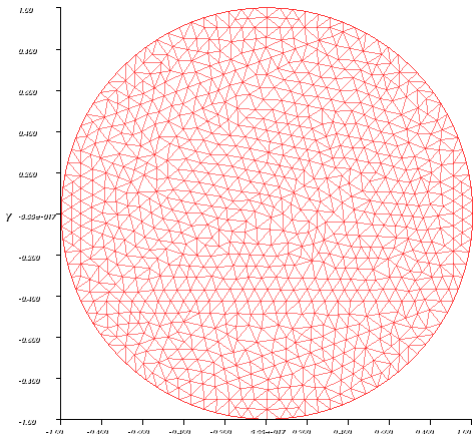


FIGURE 2.6 – Maillage avec 100 points dans le bord

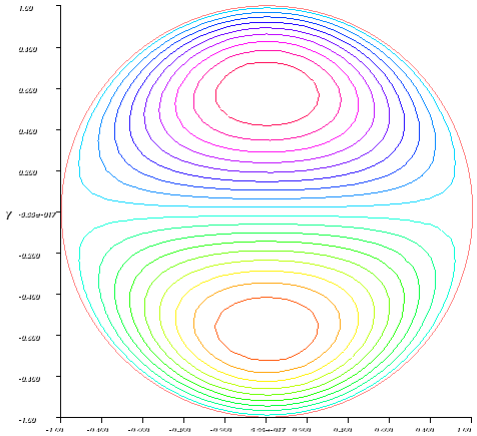


FIGURE 2.7 – Solution avec 100 points dans le bord

Exemple 2.3. Par la même méthode on pricise la frontière. Mais on définit chaque bord individuellement.
Pour construire le maillage on utilise la commande "`buildmesh`" et on fait la somme des bords avec le nombre de noeuds dans chaque bord.
La division de la frontière n'a pas nécessaire régulière.

Le code de ce problème

```
1  // Maillage
2   border a0(t=0., 1.){ x=5.; y=1.+2.*t;};
3   border a1(t=0., 1.){ x=5.-2.*t; y=3.;};
4   border a2(t=0., 1.){ x=3.-2.*t; y=3.-2.*t;};
5   border a3(t=0., 1.){ x=1.-t; y=1.;};
6   border a4(t=0., 1.){ x=0.; y=1.-t;};
7   border a5(t=0., 1.){ x=t; y=0.;};
8   border a6(t=0., 1.){ x=1.+4.*t; y=t;};
9   mesh Sh = buildmesh(a0(20) + a1(20) + a2(20)
10  + a3(20) + a4(20) + a5(20) + a6(20));
11 fespace Vh(Sh,P1) ;     // espace de P1 élément fini
12 Vh u,v ;     // u et v des éléments dans Vh
13 func f=cos(x)*y ;     // f est la fonction du variables x et y
14 problem Poisson(u,v)=     // Définition de problème
15 int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))     // form bilinéaire
16 -int2d(Sh)(f*v)     // form linéaire
17 +on(a0,a1,a2,a3,a4,a5,a6,u=0);     // Conditions de Dirichlet
18 Poisson ;     // Solution de problème de Poisson
19 plot(Sh);     // Dessin de maillage
20 plot(u) ;     // Dessin de la solution
```

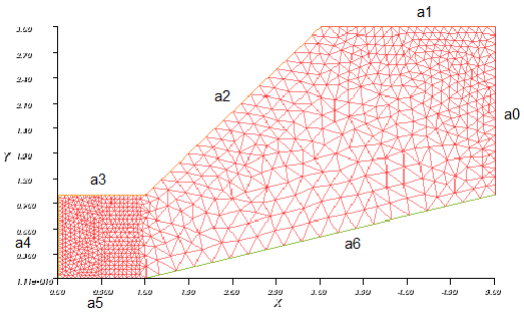


FIGURE 2.8 – Maillage

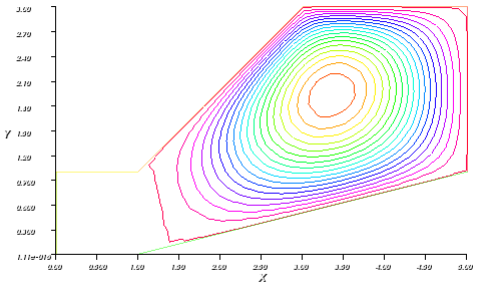


FIGURE 2.9 – Solution

Remarque 2.4. La condition aux limites de Dirichlet est imposée sur toute la frontière, si la condition nétait pas le même sur chaque frontière, on aurait pu remplacer la ligne 17 par :

```
1  +on(a0,u=4)+on(a1,u=7)+on(a2,u=2)+on(a3,u=5)+on(a4,u=1)+on(a5,u=3)
2  +on(a6,u=6);    // condition de Dirichlet
```

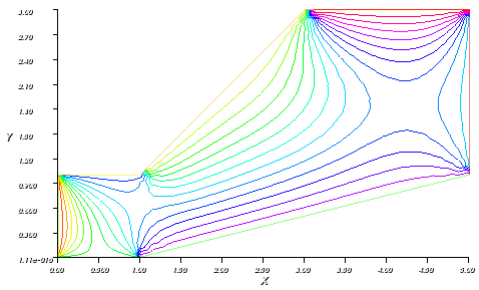


FIGURE 2.10 – Solution avec maillage irrégulier

2.2.2 Conditions aux limites de Neumann

On considère le problème suivant :

$$\begin{cases} -\Delta u + u = f \quad dans \Omega \\ \dfrac{\partial u}{\partial n} = 0 \quad \text{sur } \partial\Omega \end{cases} \tag{2.3}$$

Pour trouver la formulation variationnelle on multiplie léquation 2.3 par une fonction test régulière v et on intègre par parties en admettant que la solution $u$est suffisamment régulière, on obtient

$$-\int_\Omega \Delta u v dx + \int_\Omega uv dx = \int_\Omega fv dx$$

d'où, en utilisant la formule de Green

$$-\int_{\partial\Omega} \frac{\partial u}{\partial n} v d\sigma + \int_\Omega \nabla u.\nabla v dx + \int_\Omega uv dx = \int_\Omega fv dx$$

ce qui fait, en tenant compte de la condition au bord

$$\int_\Omega \nabla u.\nabla v dx + \int_\Omega uv dx = \int_\Omega fv dx.$$

où

$$a(u,v) = \int_\Omega \nabla u.\nabla v dx + \int_\Omega uv dx \quad et \quad l(v) = \int_\Omega fv dx.$$

Théorème 2.1. [8]*Soit Ω un ouvert et l un élément du dual de $H^1(\Omega)$, alors il existe une solution unique au problème*

$$\begin{cases} Trouver \quad u \in H^1(\Omega) \quad tel \quad \forall v \in H^1(\Omega) \\ \int_\Omega \nabla u.\nabla v(x) dx + \int_\Omega uv(x) dx = l(v). \end{cases} \tag{2.4}$$

De plus, si Ω est de classe C^2 avec $\partial\Omega$ borné et si $f \in L^2(\Omega)$, alors $u \in H^2(\Omega)$ et vérifie

$$-\Delta u + u = f \quad p.p \quad dans\Omega$$

et

$$\frac{\partial u}{\partial n} = 0 \quad sur\partial\Omega$$

au sens des traces.

Exemple 2.4. On a le problème suivant :

$$\begin{cases} -\Delta u = f \quad \text{sur } \Omega \\ \dfrac{\partial u}{\partial n} = g \quad \text{sur } \partial\Omega \end{cases} \tag{2.5}$$

la formulation variationnelle est

$$-\int_{\partial\Omega} \frac{\partial u}{\partial n} v d\sigma + \int_\Omega \nabla u.\nabla v dx = \int_\Omega fv dx$$

$$\text{c'implique} - \int_{\partial\Omega} gv d\sigma + \int_\Omega \nabla u.\nabla v dx = \int_\Omega fv dx$$

tel que : $f = cos(x)y \quad g = 1.$

Le code de ce problème

```
1  // omega est un disque avec deux trou
2  int Neumann=1;
3  border C0(t=0, 2*pi){ x=5*cos(t); y=5*sin(t);label=Neumann;};
4  border C1(t=0, 2*pi){ x=2+0.3*cos(t); y=3*sin(t);label=Neumann;};
5  border C2(t=0, 2*pi){ x=-2+0.3*cos(t); y=3*sin(t);label=Neumann;};
6  mesh Th = buildmesh(C0(60) +C1(-50) + C2(-50));
7  plot(Th);
8  fespace Vh(Th,P1);
9  Vh u,v;
10 func f=cos(x)*y;  func g=1.;
11 problem Poisson(u,v)=
12 int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))
13 -int1d(Th,Neumann)(g*v)     //Forme linéaire sur ∂Ω pour Neumann
14 -int2d(Th)(f*v);
15 Poisson;
16 plot(u);
```

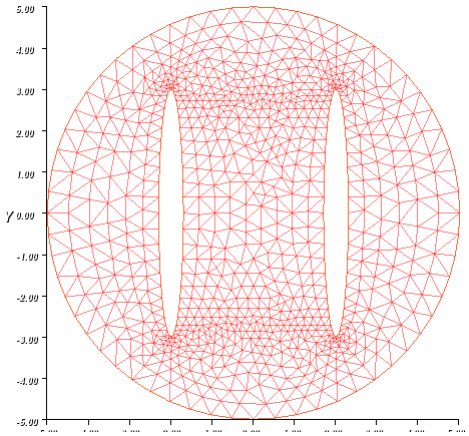


FIGURE 2.11 – Maillage

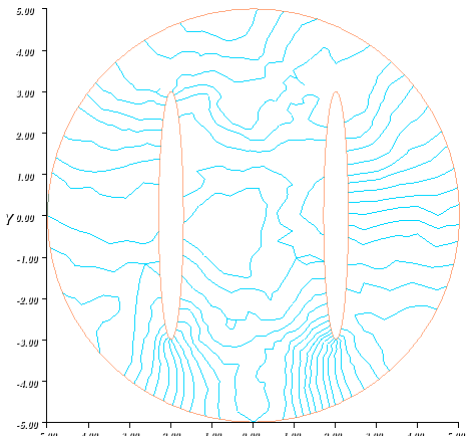


FIGURE 2.12 – Solution en utilisant P1

Remarque 2.5. .

1. Les éléments finis `P3` ne sont pas disponibles par défaut. Il faut les charger d'abord (en fait, vous pouvez même définir vos propres éléments) par la commande `"load"` au début de code et on change `P1` par `P3` dans la ligne 6 comme suite :

```
1  load "Element_P3";
2  .......
3  fespace Vh(Th,P3);
```

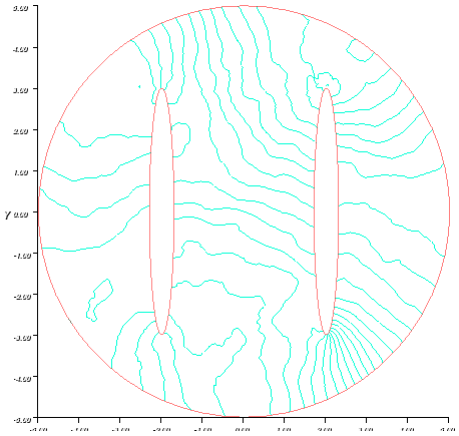


FIGURE 2.13 – Solution en utilisant `P3`

2. L'élément fini `P4` est le plus grand élément pouvant être utilisé dans Freefem++.

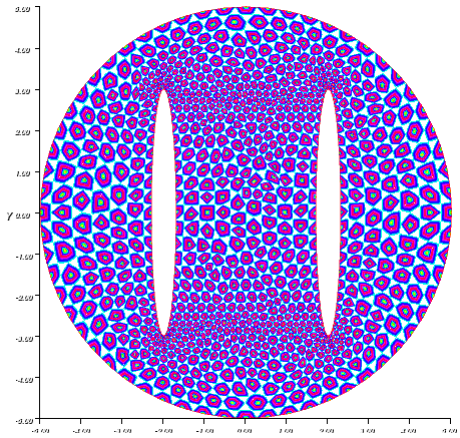


FIGURE 2.14 – Solution en utilisant `P4`

3. Si on charge `P5`, le code n'éxecute pas et Freefem++ affiche l'erreur suivant :

```
-- FreeFem++ v  3.260003 (date jeu. 16 janv. 2014 15:52:04)
 Load: lg_fem lg_mesh lg_mesh3 eigenvalue
    1 : load "Element_P5"
load error : Element_P5
            fail :
list  prefix: 'C:\Users\windows7\FreeFem++-cs-14.3\Contents\Windows\' " list  suffix : " , '.dll'

 Error line number 1, in file C:\Users\windows7\AppData\Local\Temp\.ffcs9836.edp, before  token Element_P5
Error load
  current line = 1
Compile error : Error load
          line number :1, Element_P5
error Compile error : Error load
          line number :1, Element_P5
 code = 1 mpirank: 0
FreeFem++ returned error 1
```

FIGURE 2.15 – Affichage d'erreur

2.2.3 Conditions aux limites mixtes

Freefem++ nous permet d'inclure les conditions aux limites de Dirchlet et de Neumann ensemble, comme dans les exemples suivants :

Exemple 2.5. On considère le problème suivant :

$$\begin{cases} -\Delta u = f \quad \text{sur } \Omega \\ \dfrac{\partial u}{\partial n} = g \quad sur \Gamma_N \\ u = u_D \quad \text{sur } \Gamma_D \end{cases} \tag{2.6}$$

La formulation variationnelle est :

$$-\int_{\Gamma_D} u_D v d\sigma + \int_{\Gamma_N} g v d\sigma + \int_{\Omega} \nabla u . \nabla v dx = \int_{\Omega} f v dx.$$

tel que : $f = cos(x)y \quad g = 1 \quad u_D = x$.
On choisit un domaine L-forme.

Le code de ce problème

```
1  int Dirichlet=1,Neumann=2 ;    // Pour label définition
2  border a(t=0,1.0){x=t ; y=0 ; label=Neumann ;} ;
3  border b(t=0,0.5){x=1 ; y=t ; label=Neumann ;} ;
4  border c(t=0,0.5){x=1-t ; y=0.5 ;label=Neumann ;} ;
5  border d(t=0.5,1){x=0.5 ; y=t ; label=Dirichlet ;} ;
6  border e(t=0.5,1){x=1-t ; y=1 ; label=Dirichlet ;} ;
7  border h(t=0.0,1){x=0 ; y=1-t ;label=Dirichlet ;} ;
8  plot(a(6) + b(4) + c(4) +d(4) + e(4) + f(6),wait=1) ;
9  mesh Sh =buildmesh(a(6) + b(4) + c(4) +d(4) + e(4) + f(6)) ;
10 fespace Vh(Sh,P1) ;
11 Vh u,v ;
12 func f=cos(x)*y ;func ud=x ; func g=1. ;
13 problem Poisson(u,v)=
14 int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))
15 +int1d(Sh,Neumann)(g*v)
16 -int2d(Sh)(f*v)
17 +on(Dirichlet,u=ud) ;     // u=ud on label=Dirichlet=1
18 Poisson ;
19 plot(Sh) ;
20 plot(u) ;
```

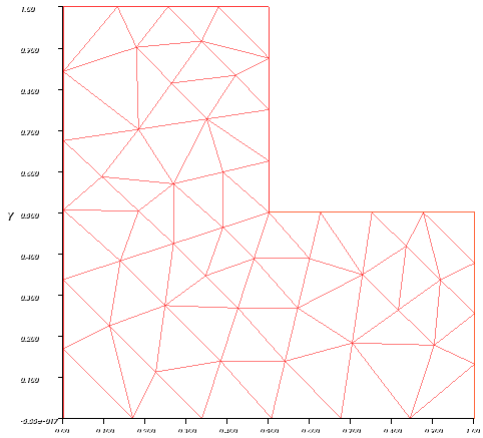


FIGURE 2.16 – Maillage

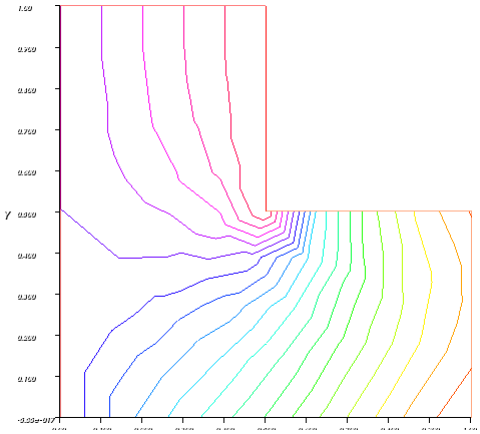


FIGURE 2.17 – Solution avec C.L. mixtes

Remarque 2.6. .

1. Si les bordures ne sont pas fermées (ici l'erreur dans le bord b)

```
1  int Dirichlet=1,Neumann=2;    // For label définition
2  border a(t=0,1.0){x=t; y=0; label=Neumann;};
3  border b(t=0,0.4){x=1; y=t; label=Neumann;};
4  border c(t=0,0.5){x=1-t; y=0.5;label=Neumann;};
5  border d(t=0.5,1){x=0.5; y=t; label=Dirichlet;};
6  border e(t=0.5,1){x=1-t; y=1; label=Dirichlet;};
7  border h(t=0.0,1){x=0; y=1-t;label=Dirichlet;};
8  plot(a(6) + b(4) + c(4) +d(4) + e(4) + f(6),wait=1);
9  mesh Sh =buildmesh(a(6) + b(4) + c(4) +d(4) + e(4) + f(6));
10 fespace Vh(Sh,P1);
11 Vh u,v;
12 func f=cos(x)*y;func ud=x; func g=1.;
13 problem Poisson(u,v)=
14 int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))
15 +int1d(Sh,Neumann)(g*v)
16 -int2d(Sh)(f*v)
17 +on(Dirichlet,u=ud);     // u=ud on label=Dirichlet=1
18 Poisson;
19 plot(Sh);
```

```
20 plot(u);
```

le maillage et la solution n'apparaissent pas.

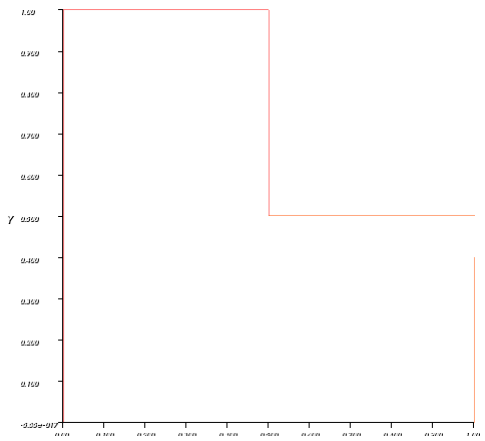


FIGURE 2.18 – Solution avec bordures ne sont pas fermés

Freefem++ affiche l'erreur suivante :

```
error : 56 0 56
Error: The boundary is not close => All triangles are outside
 Fatal error in the meshgenerator 888
  current line = 2
Meshing error: Bamg
 number : 888,
 catch Err bamg
Meshing error: Bamg
 number : 888,
 err code 4 ,  mpirank 0
FreeFem++ returned error 4
```

FIGURE 2.19 – Affichage d'erreur

2. Si les bordures ne sont pas continues et ne conservent pas la même direction (ici l'erreur dans le bord b)

```
1  int Dirichlet=1,Neumann=2;    // For label définition
2  border a(t=0,1.0){x=t; y=0; label=Neumann;};
3  border b(t=0.5,0){x=1; y=t; label=Neumann;};
4  border c(t=0,0.5){x=1-t; y=0.5;label=Neumann;};
5  border d(t=0.5,1){x=0.5; y=t; label=Dirichlet;};
6  border e(t=0.5,1){x=1-t; y=1; label=Dirichlet;};
7  border h(t=0.0,1){x=0; y=1-t;label=Dirichlet;};
8  plot(a(6) + b(4) + c(4) +d(4) + e(4) + f(6),wait=1);
9  mesh Sh =buildmesh(a(6) + b(4) + c(4) +d(4) + e(4) + f(6));
10 fespace Vh(Sh,P1);
11 Vh u,v;
12 func f=cos(x)*y;func ud=x; func g=1.;
13 problem Poisson(u,v)=
14 int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))
15 +int1d(Sh,Neumann)(g*v)
16 -int2d(Sh)(f*v)
17 +on(Dirichlet,u=ud);     // u=ud on label=Dirichlet=1
18 Poisson;
19 plot(Sh);
20 plot(u);
```

le maillage et la solution n'apparaissent pas.

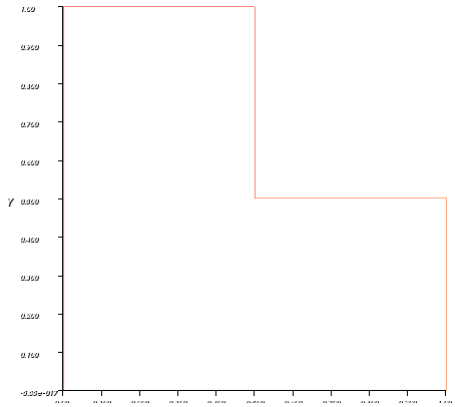


FIGURE 2.20 – Affichage de Freefem

Freefem++ affiche l'erreur suivante :

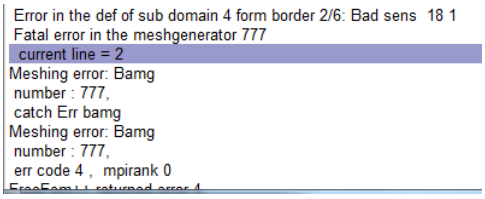


FIGURE 2.21 – Affichage d'erreur

Exemple 2.6. On considère le problème suivant :

$$\begin{cases} -\Delta u = f \quad dans\Omega \\ u = g \quad \text{sur } \Gamma_1 \\ \dfrac{\partial u}{\partial n} = h \quad \text{sur } \partial\Omega|\Gamma_1 \end{cases} \tag{2.7}$$

La formulation variationnelle est :

$$-\int_{\Gamma_1} gvd\sigma + \int_{\partial\Omega|\Gamma_1} hvd\sigma + \int_{\Omega} \nabla u.\nabla v dx = \int_{\Omega} fv dx.$$

Pour se problème, on choisit un domaine rectangulaire avec un trou. Le bord Γ_1 jouera le rôle de la frontière intérieure. La fonction f de sorte que lon connaisse la solution exacte du problème afin de calculer lerreur L^2. On choisira donc $f = 5\pi^2 sin2\pi x sin\pi y$ de sorte que la solution est $u = sin2\pi x sin\pi y$ et on choisit $g = h = 0$.

Le code de ce problème

```
1 // Définition des maillages
2 border Gammab( t=0,1){ x=2*t;y=0 ;label=1 ;};
3 border Gammad( t =0 ,1){ x=2 ;y=t  ; label = 2 ;};
4 border Gammah( t =0 ,1){ x=2*(1-t )  ; y=1 ; label = 3 ;};
5 border Gammag( t =0 ,1){ x=0 ;y=1-t  ; label = 4 ;};
```

```
6  border Gammai( t =0,2* pi ){ x=1+0.5*cos ( t ) ; y=0.5+0.3* sin ( t ) ; label ...
       = 5 ;} ;
7  mesh Th=buildmesh(Gammab( 40 )+Gammad( 20 )+Gammah( 40 )+Gammag( 20 ) ) ;
8  mesh Mh=buildmesh(Gammab( 40 )+Gammad( 20 )+Gammah( 40 )+Gammag( 20 )+
9  Gammai(-30) ) ;
10 plot (Th, ps="maillage . ps") ;
11 plot (Mh, ps="maillagetrou . ps ") ;
12 // Définition des espaces dapproximation
13 fespace Vh(Th, P1 ) ;
14 fespace Wh(Mh, P1 ) ;
15 // Définition des données du problème
16 func f =5*sin ( 2* pi *x ) * sin ( pi *y ) ;
17 func g=0 ;
18 func h=0 ;
19 func u=sin ( 2* pi *x ) * sin ( pi *y ) ;
20 Vh uh , vh , ue ;
21 Wh uuh , vvh ;
22 // Définition du problème variationnel sur Th
23 problem P( uh , vh ) = int2d (Th) (dx( uh ) *dx( vh )+dy( uh ) *dy( vh ) )
24 - int2d (Th) ( f *vh )
25 + on( 1 , 2 , 3 , 4 , uh=0) ;
26 // Définition du problème variationnel sur Mh
27 problem Q( uuh , vvh ) = int2d (Mh) (dx( uuh ) *dx( vvh )+dy( uuh ) *dy( vvh ) )
28 - int2d (Mh) ( f *vvh )
29 - int1d (Mh, 1 , 2 , 3 , 4 ) ( h*vvh )
30 + on( 5 , uuh=g ) ;
31 // Résolution des problèmes variationnels
32 P ;
33 Q ;
34 // Calcule derreur L^2 pour le problème P
35 ue=u ;
36 real erreur=int2d (Th) ( ( ue-uh ) *( ue-uh ) ) ;
37 string legende ="erreur = "+erreur ;
38 // Affichage des solutions
39 plot ( uh ,cmm=legende , wait=true , ps="lapdirichlet . ps ") ;
40 plot ( uuh , wait=true , fill =1,value=true , ps="lapneumanntrou . ps ") ;
```

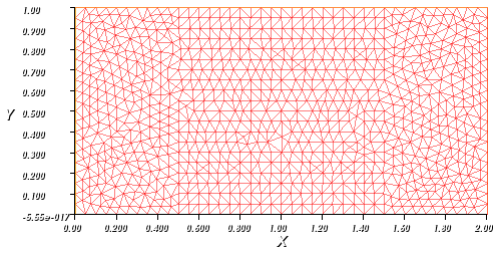


FIGURE 2.22 – Maillage Th

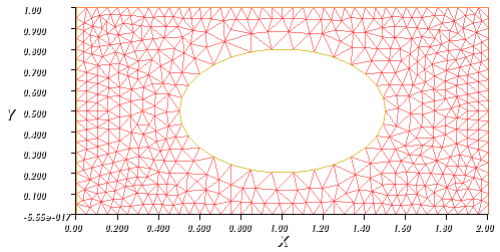
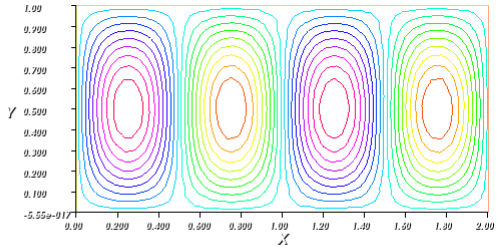


FIGURE 2.23 – Maillage `Mh`



```
erreur = 0.396508
```

FIGURE 2.24 – Calculer l'erreur pour problème `P`

Remarque 2.7. .

1. L'erreur est calculée et écrite dans la console.
2. Les résultats sont comparés à la solution exacte et représentés en termes de contours dans 2.6.

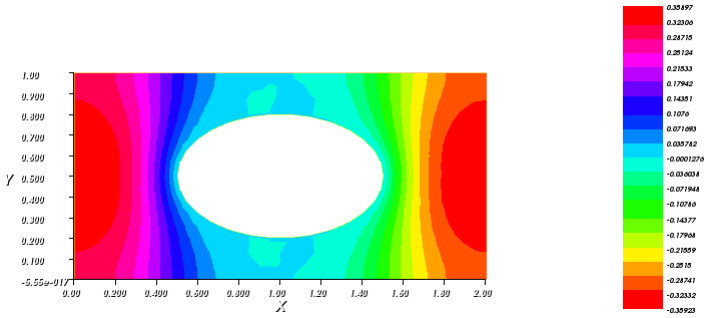


FIGURE 2.25 – Solutions

Remarque 2.8. .

1. Pour donner une référence (par exemple `label = 1`) à chaque courbe afin didentifier cette portion de la frontiere lors de lintroduction des conditions aux limites.
2. Pour remplir entre les isolignes, nous utilisons la commande "`fill=1`" avec "`plot`".
3. Afin d'afficher l'échelle à barres à côté de la solution, nous utilisons la commande "`value=true`" avec "`plot`".

2.2.4 Conditions aux limites de Robin

On considère le problème suivant :

$$\begin{cases} -\Delta u = f \quad \text{sur } \Omega \\ \dfrac{\partial u}{\partial n} + \alpha u = g \quad \text{sur } \partial\Omega \end{cases} \tag{2.8}$$

On suppose que l'ouvert Ω est borné, connexe et régulier. On multiplie l'équation par une fonction test v et on intègre sur Ω. Ce qui donne ici

$$-\int_\Omega \Delta u v dx = \int_\Omega f v dx$$

d'où, en utilisant toujours la formule de Green

$$-\int_{\partial\Omega} \frac{\partial u}{\partial n} v d\sigma + \int_\Omega \nabla u.\nabla v dx = \int_\Omega f v dx$$

ce qui fait, en tenant compte de la condition au bord

$$\int_\Omega \nabla u.\nabla v(x) dx + \alpha \int_{\partial\Omega} uv d\sigma = \int_\Omega f v dx + \int_{\partial\Omega} g v d\sigma$$

où

$$a(u,v) = \int_\Omega \nabla u.\nabla v dx + \alpha \int_{\partial\Omega} uv d\sigma \quad et \quad l(v) = \int_\Omega f v dx + \int_{\partial\Omega} g v d\sigma.$$

Dans l'exemple ci-dessous, on va considère l'equation de poisson sur $3D$ et avec conditions aux limites de Robin .

Il y a des différences manières de créer un maillage $3D$ dans FreeFem++. On utilise la commande "`buildlayers`" dans lequel un domaine $3D$ est généré à partir d'un maillage de $2D$, et aprés on défnit le domaine de troisième dimension, la commande écrire comme suite :

Il faut définir ce type de mesh par `load "msh3"` au début et définit la par

```
1  mesh3 (nom de mesh)=buildlayers(mesh de 2D,N,zbound=[a,b]);   //Maillage 3D
```

tel que :- N est le nombre de parties de domaine qu'on va définir.

- [a,b] est le domaine de troisième dimension.

Exemple 2.7.

$$\begin{cases} -\Delta u = f, \quad \text{sur } \Omega_{3D} = [0,1]^2 \times [0,1] \\ \dfrac{\partial u}{\partial n} = \alpha(u - u_e) \ \text{ sur le deuxième façe} \\ \dfrac{\partial u}{\partial n} = 0 \ \text{ sur le reste de la frontière} \end{cases} \tag{2.9}$$

La formulation variationnelle est :

$$\int_\Omega \nabla u.\nabla v dx dy dz + \int_{façe2} \alpha uv d\Gamma - \int_{façe2} \alpha u_e v d\Gamma - \int_\Omega f v dx dy dz = 0 \quad \forall v.$$

tel que $f = 1000 \cdot \chi_{\{0.4\leq x\leq 0.6\}} \cdot \chi_{\{0.4\leq y\leq 0.6\}} \cdot \chi_{\{0.4\leq z\leq 0.6\}}$

Le code de ce problème

```
1  //Équation de Poisson sur le cube unité 3D
2  load "msh3"     // Nous avons besoin de msh3 en 3D
3  mesh Th2=square(10,10);    //Maillage 2D
4  int ncapas=10;     // Nombre de couches horizontales sur l'axe z
5  real zmin=0, zmax=1;
6  mesh3 Th=buildlayers(Th2,ncapas,zbound=[zmin,zmax]);     //Maillage 3D
7  real alpha=0.25, ue=25;
8  func f=1000.*(x>0.4)*(x<0.6)*(y>0.4)*(y<0.6)*(z>0.4)*(z<0.6);
9  fespace Vh(Th,P13d);     //P1 éléments finis de Lagrange en 3D
10 Vh u,v;     // u,v appartiennent à Vh
11 // Et maintenant nous utilisons des macros pour simplifier les expressions
12 macro grad(u) [dx(u),dy(u),dz(u)]     //MOE
13 solve lapace3d(u,v)=int3d(Th)(grad(u)'*grad(v))
14 +int2d(Th,2)(alpha*u*v)
15 -int3d(Th)(f*v)
16 -int2d(Th,2)(alpha*ue*v);
17 plot(u);
```

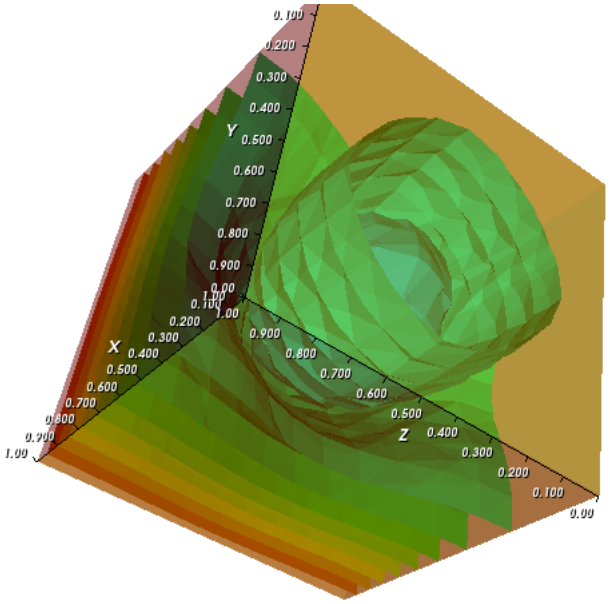


FIGURE 2.26 – Solution avec C.L. de Robin (3D)

Chapitre 3

Problèmes instationnaires

Dans ce chapitre on va présenter les problèmes instationnaires, et on va exprimer comment les résoudre par FreeFem++. Par donner des exemples du problèmes d'évolution et on va utilise différents schémas pour la solution .

3.1 Problèmes d'évolution

FreeFem++ résout également des problèmes d'évolution tels que l'équation de la chaleur. On considère le problème modèle comme équation de la chaleur. On se donne une fonction u_0 définie sur Ω qui exprime la condition initiale et une fonction $f(t,x)$ (source de chaleur) définie sur $[0,T]\times\Omega$ et il s'agit de trouver une fonction $u(t,x)$ solution de

$$\begin{cases} \dfrac{\partial u}{\partial t} - \Delta u = f & \text{sur }]0,T[\times\Omega \\ u = 0 & \text{sur }]0,T[\times\partial\Omega \\ u(0,x) = u_0(x) & \text{sur } \Omega \end{cases} \tag{3.1}$$

Pour le problème dévolution 3.1, et lors de la transformation sous la forme variationnelle :
On multiplions léquation par une fonction test $v \in H_0^1(\Omega)$ qui ne dépend pas du temps et intégrons sur Ω :

$$\int_\Omega \frac{\partial u}{\partial t}(t,x)v(x)dx - \int_\Omega \Delta u(t,x)v(x)dx = \int_\Omega f(t,x)v(x)dx.$$

En utilisant la formule de Green et le fait que (au moins formellement)

$$\int_\Omega \frac{\partial u}{\partial t}(t,x)v(x)dx = \frac{d}{dt}\int_\Omega u(t,x)v(x)dx.$$

On obtient

$$\frac{d}{dt}\int_\Omega u(t,x)v(x)dx + \frac{d}{dt}\int_\Omega \nabla_x u(t,x).\nabla v(x)dx = \int_\Omega f(t,x)v(x)dx.$$

En fin, on pose le schéma implicite ou explicite du problème.

3.1.1 Schéma implicite

Exemple 3.1. trouver u solution de

$$\begin{cases} \partial_t u - \Delta u = f & \text{dans } \Omega\times]0,T[\\ u(x,0) = u_0 & \text{dans } \Omega \\ u_{|\Gamma} = g \end{cases} \tag{3.2}$$

où u_0, f, g, T sont des données.
On lapproche à un schéma d'Euler implicite :$u^0 = u_0$
pour $n = 0, N$ avec $\delta t = T/N$ faire

$$\begin{cases} u^{n+1} - u^n - \delta t \Delta u^{n+1} = \delta t f \\ u^{n+1}_{|\Gamma} = g \quad \text{sur } \Omega \end{cases} \tag{3.3}$$

La formulation variationnelle :

$$\int_\Omega (u^{n+1} v + \delta t \nabla u^{n+1} . \nabla v) = \int_\Omega (u^n v + \delta t f v).$$

tel que : $f = 1$ $g = 0$ $u_0 = 0$.

Le code de ce problème

```
int inside = 2 ;     // bordure intérieure
int outside = 1 ;     // bordure extérieure
border a(t=1,2){x=t ;y=0 ;label=outside ;} ;
border b(t=0,1){x=2 ;y=t ;label=outside ;} ;
border c(t=2,0){x=t ;y=1 ;label=outside ;} ;
border d(t=1,0){x = 1-t ; y = t ;label=inside ;} ;
border e(t=0, pi/2){ x= cos(t) ; y = sin(t) ;label=inside ;} ;
border e1(t=pi/2, 2*pi){ x= cos(t) ; y = sin(t) ;label=outside ;} ;
int n=4 ;
mesh th = buildmesh( a(5*n) + b(5*n) + c(10*n) + d(5*n)) ;
mesh Th = buildmesh( e(5*n) + e1(25*n) ) ;
mesh Sh =th+Th;
plot(Sh,wait=1) ;
fespace Vh(Sh,P1) ;      // Espace P1 EF
Vh uh,vh,u1=0 ;     // fonction inconnue et test.
func f=1 ;     // fonction côté droit
func g=0 ;     // fonction de condition aux limites
real dt =0.01 ;
int i=0 ;
problem Poisson(uh,vh,init=i) =     // définition du problème
int2d(Sh)( uh*vh+dt*(dx(uh)*dx(vh) + dy(uh)*dy(vh)) )     // forme bilinéaire
- int2d(Sh)( (u1+dt*f)*vh )     // forme linéaire
+ on(1,2,3,4,uh=g) ;     // formulaire de conditions aux limites
for (i=0 ;i<10 ;i++)
{
Poisson ;     // résoudre le problème un+1 == uh
plot(uh,value=true,fill=1) ;
u1=uh ;     // poser un == u1
}
```

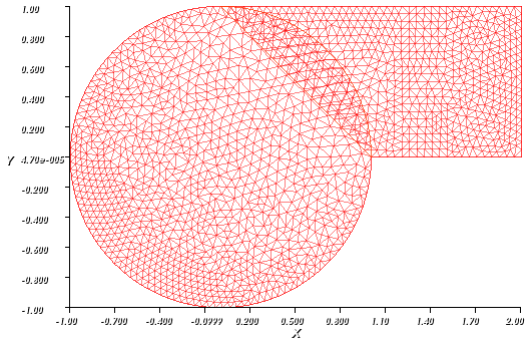


FIGURE 3.1 – Maillage

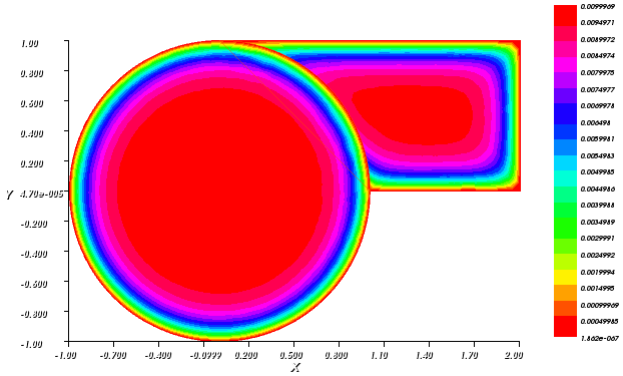


FIGURE 3.2 – Début de la solution

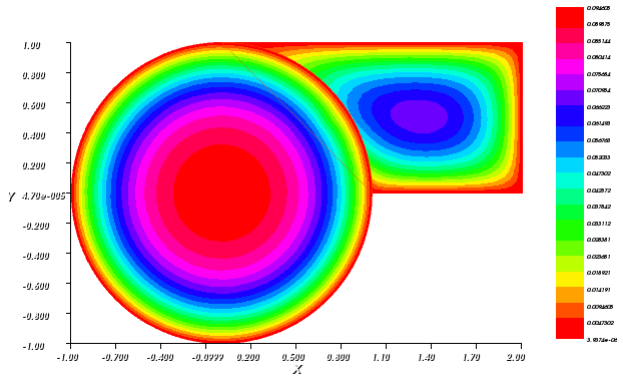


FIGURE 3.3 – Fin de la solution

Exemple 3.2. Dans ce qui suit, nous allons formuler une résolution d'une équation mise sous la forme suivante :

$$\begin{cases} \dfrac{\partial u(x,y,t)}{\partial t} - \alpha \triangle u(x,y,t) = f(x,y,t) & ((x,y),t) \in \Omega \times]0,T[\\ u(x,y,t) = 0 & ((x,y),t) \in \partial\Omega \times]0,T[\\ u(x,y,0) = u_0(x,y) & ((x,y),t) \in \Omega \end{cases} \tag{3.4}$$

Ceci revient à une formulation variationnelle :

$$\int_\Omega \frac{u^{n-1}v}{\triangle t} - \int_\Omega \frac{u^n v}{\triangle t} - \alpha \int_\Omega \nabla u \nabla v = \int_\Omega fv$$

On pose :

$$u_{ex}(x,y,t) = (x^2-4)*(y^2-4)*(t-9)$$

$$f(x,y,t) = x^2*y^2 - 2*t*x^2 - 2*t*y^2 + 14*x^2 + 14*y^2 + 16*t - 128$$

Maintenant on va présenter le code FreeFem du problème avec explication de chaque étape.

Le code de ce problème

```
1  // Domaine et maillage
2
3  real[int] edges = [-2,2,-2,2] ;      // Coins du domaine
4  int n = 50 ;      // Nombre d'éléments
5  mesh pave = ...
       square(n,n,[edges[0]+(edges[1]-edges[0])*x,edges[2]+(edges[3]-edges[2])*y]) ;
6    //Maillage rectangulaire
7  int BORDURE = 10 ;
8  int[int] labels = [1,BORDURE,2,BORDURE,3,BORDURE,4,BORDURE] ;
9  pave = change(pave,label=labels) ;
10 savemesh(pave,"pave.msh") ;
11 plot(pave,ps="pave.png") ;      // Graphique et exportation
```

Dans cette partie, le domaine rectangulaire est défini, ainsi que une généralisation de la bordure, et le nombre d'éléments du maillage. On exporte par la suite le maillage.

```
1  //-------------------------
2  // Constantes et espace éléments finis
3  //-------------------------
4  real t=0,T = 1.; // [0,T] Temps
5  int N = 50 ;      // Nombre d'éléments temps
6  real dt = T/N ;      // Pas de discrétisation
7  real alpha = 1 ;      // Conductivite thérmique
8  int i = 0 ;      // Index
9
10 fespace VH(pave,P2) ;
11 VH u,v,old,f,uex;
12
13 //-------------------------
14 // Condition initiales
15 //-------------------------
16 VH[int] uh(N+2) ;      // Vecteur regroupant les valeurs de uh
17 uh[0] = exp(x)*exp(y) ;      // Valeur initiale de uh a t=0
18 old = uh[0] ;
```

Ici on definit les constantes ainsi que la valeur de la solution à t=0.

```
1  //-------------------------
2  // Calcul et exportation
3  //-------------------------
4
5  // Formulation variationnelle du problème
6  problem thermic(u,v)=
7  int2d(pave)(dt*alpha*(dx(u)*dx(v)+dy(u)*dy(v)))
8  +int2d(pave)(u*v)
9  -int2d(pave)(old*v)
```

```
10 +on(BORDURE,u=0);
```

La formulation variationnelle traduite en syntaxe FreeFem++

```
1  // Erreurs
2  real[int] errL2(N+1);
3  real[int] errL2P(N+1);
4  real[int] errH1(N+1);
5  // Calcul en temps discrétise
6  for(i;i≤N;i++){
7      t=i*dt;     // Incrémentation du temps
8      f = x ^ 2 * y ^ 2 - 2 * t * x ^ 2 - 2 * t * y ^ 2 + 14 * x ^ 2 + 14 * y ^ ...
           2 + 16 * t - 128;
9       // Fonction second membre qui change en fonction de t
10     uex = (x ^ 2 - 4) * (y ^ 2 - 4) * (t - 9);     // Solution exacte évaluee avec
           Matlab auparavant
11     thermic;     // Résolution du problème
12     plot(u,wait=0,fill=1,cmm="t="+t,ps="snapshots/"+i+"_frame_t"+t+".png");
13         //Graphique et exportation
14     old = u;      // On remplace la valeur de U(n-1) par U(n)
15     uh[i+1] = old;      // (Facultatif)
16
17     // Evaluation des erreurs
18     errL2[i] = sqrt(int2d(pave)((u-uex)^2));
19     errH1[i] = sqrt(int2d(pave)((dx(u)-dx(uex))^2+(dy(u)-dy(uex))^2));
20     errL2P[i] = errL2[i]/sqrt(int2d(pave)((dx(u)+dy(u))^2));
21 }
```

Dans cette partie, on définit des tableaux qui vont contenir les valeurs des distances. Ainsi que la fonction second membre calculée avec Maple à partir d'une solution exacte aussi définie. Les deux fonctions ont été définies à l'intérieur de la boucle car elles sont dépendentes de t.

```
1  // Exportation des erreurs
2  // Entete de fichier en Markdown
3  {
4      ofstream f("evaluated_log.txt");
5      f << "| T Value | L2 Evaluated Difference | H1 Evaluated Difference
6       | L2 Evaluated Percentage   |"<< endl;
7      f << "|---------|-------------------------|-------------------------
8      |---------------------------|"<< endl;
9  };
10 for(int j=0;j≤N;j++){
11     {
12         ofstream f("evaluated_log.txt",append);
13         f << "|"+j*dt+"|"+errL2[j]+"|"+errH1[j]+"|"+errL2[j]/errL2P[j]+"|" <<endl;
14     };
15 }
```

On exporte les valeurs dans finchier texte sous forme de tableau en Markdown pour la facilité de sa mise en forme.

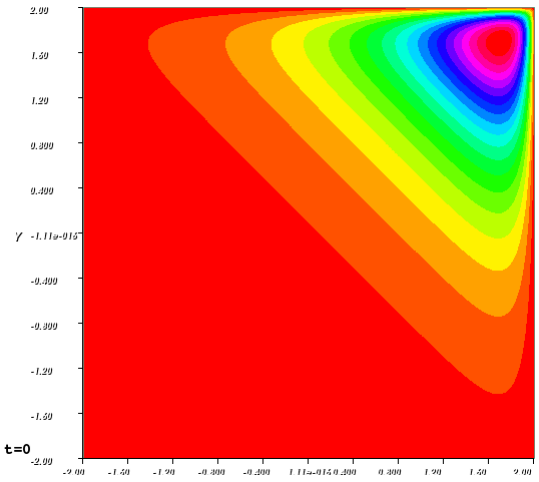


FIGURE 3.4 – Début de la solution

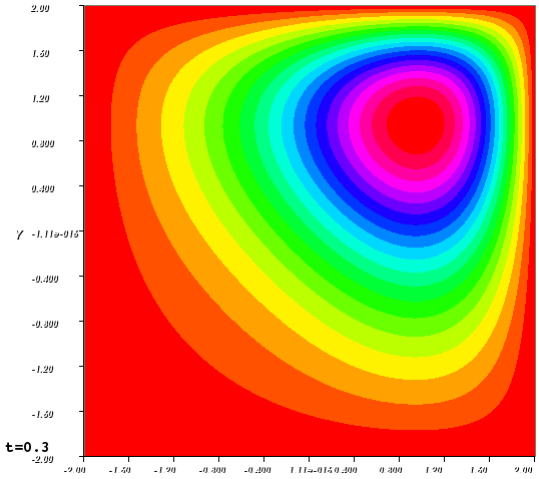


FIGURE 3.5 – Solution du moment t=0.3

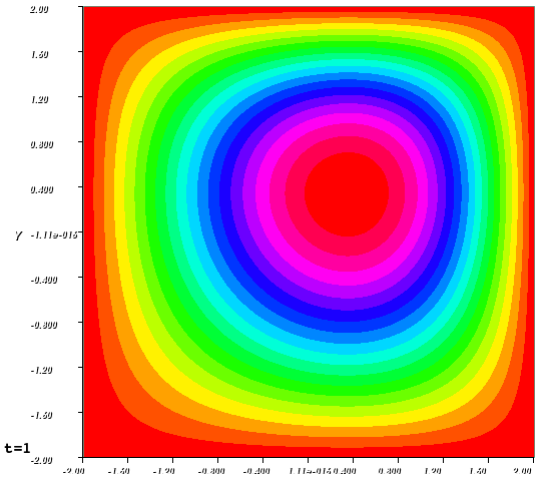


FIGURE 3.6 – Fin de la solution

Exemple 3.3. On souhaite déterminer lévolution dun solide élastique, solution du problème variationnel consistant à déterminer

$u \in C^1([0,T]\,;L^2(\Omega)^2) \cap C^0([0,T]\,;H^1(\Omega)^2)$ tel que pour tout $t \in]0,T[$ et tout $v \in H^1(\Omega)^2)$:

$$\int_\Omega \rho \frac{d^2u}{dt^2} v dx + \int_\Omega 2\mu e(u).e(v) + \lambda(div\ u)(div\ v) = \int_\Omega f.v \tag{3.5}$$

où f sont les forces appliquées, $\mu\ et\lambda$ les coefficients de Lamé du solide, ρ la masse volumique et $e()$ le tenseur symétrisé. Il faut éventuellement ajouter des conditions aux limites (temps et espace). On effectue une discrétisation de type différences finies en temps. On peut (par exemple) considéré le schéma implicite, centré dordre 2 consistant à déterminer

$$\begin{aligned}\int_\Omega \rho \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} v \\ +\theta \int_\Omega 2\mu e(u^{n+1}).e(v) + \lambda(div\ u^{n+1})(div\ v) \\ +(1-\theta) \int_\Omega 2\mu e(u^{n-1}).e(v) + \lambda(div\ u^{n-1})(div\ v) = \int_\Omega f.v.\end{aligned}$$

tel que : $f = 0$

Le code de ce problème

```
real Lx=5., Ly=1., dn=7., mu=10., lambda=0., rho=1. ;
real Dt=0.5, T=100., exa=0.3 ;
func f1=0. ;func f2=0. ;
mesh Sh=square(Lx*dn,Ly*dn,[Lx*x,Ly*y]) ;
fespace Vh(Sh,[P2,P2]) ;
macro e(u1,u2)
[dx(u1),(sqrt(2.)*dx(u2)+dy(u1))/2.,dy(u2)] //
macro div(u1,u2) (dx(u1)+dy(u2)) //
Vh [u1,u2],[v1,v2] ;
Vh [up1,up2],[upp1,upp2] ;
[upp1,upp2]=[0.,0.] ;     // Déplacement initial
[v1,v2]=[0.,-1] ;     // Vitesse initiale
[up1,up2]=[upp1+Dt*v1,upp2+Dt*v2] ;     // au temps Dt
real t=0 ;
real theta=0.6 ;     // theta >= 1/2 => cest stable
problem elasticity([u1,u2],[v1,v2],init=t,solver
=Cholesky)=
int2d(Sh)(rho*([u1,u2]'*[v1,v2])/(Dt)^2)
+int2d(Sh)(theta*(2.*mu*(e(u1,u2)'*e(v1,v2))
+lambda*div(u1,u2)*div(v1,v2)))
+int2d(Sh)((1.-theta)*(2.*mu*(e(upp1,upp2)'*e(v1,v2))
+lambda*div(upp1,upp2)*div(v1,v2)))
-int2d(Sh)(f1*v1+f2*v2)
-int2d(Sh)(rho*([2.*up1-upp1,2.*up2-upp2]'*[v1,v2])/(Dt)^2)
+on(4,u1=0,u2=0) ;
mesh Th=Sh ;
mesh Th0=Sh ;
fespace Ph(Th,P1) ;
fespace Wh(Sh,P1) ;
Ph sigma2=0 ; Wh sigma ;
while(t<T){
elasticity ;
sigma=(2.*mu*(e(u1,u2)'*e(u1,u2))+lambda*div(u1,u2)*div(u1,u2)) ;
```

```
34 Th=movemesh(Sh,[x+exa*u1,y+exa*u2]) ;
35 sigma2=0 ;sigma2[]=sigma[] ;
36 plot(sigma2,fill=1,wait=0,bb=[[0,-Ly],[Lx+Ly,2.*Ly]]);t+=Dt ; ...
       [upp1,upp2]=[up1,up2] ; [up1,up2]=[u1,u2] ;
37 }
```

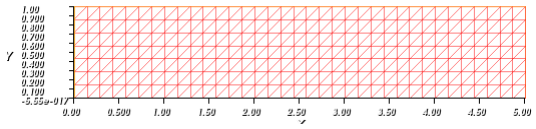


FIGURE 3.7 – Maillage

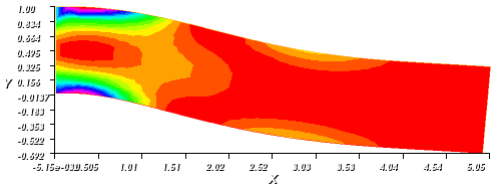


FIGURE 3.8 – Début de la solution

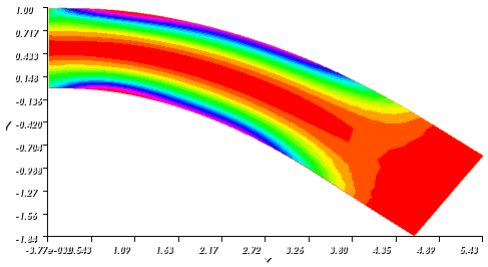


FIGURE 3.9 – Solution à moment facultatif

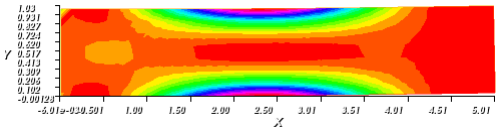


FIGURE 3.10 – Solution à moment facultatif

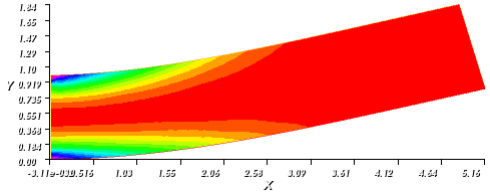


FIGURE 3.11 – Fin de la solution

Remarque 3.1. Quand le problème n'est pas stable le programme commance la solution mais il enregistre une erreur et s'arrête, par exemple si on change la ligne 15 dans le code au dessus comme ça

```
1 real theta=0.2 ;      // theta < 1/2 => nest pas stable
```

FreeFem++ donne la début de solution comme suite

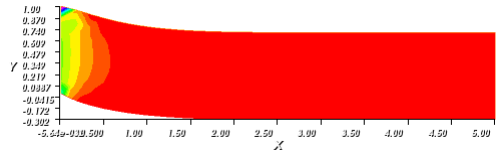


FIGURE 3.12 – Début de la solution

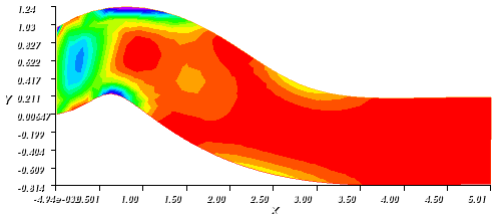


FIGURE 3.13 – Solution à moment facultatif

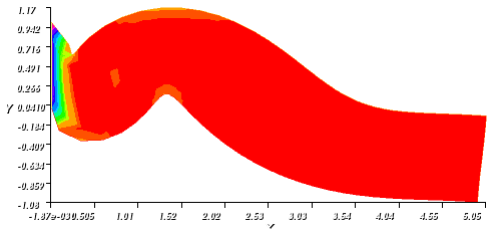


FIGURE 3.14 – Fin de la solution

et il affiche l'erreur suivante

```
429
        Error movemesh: 1 triangles was reverse  (=> no move)
u min -0.0344495 max 5.23032
v min -1.5598 max 1.16011
 current line = 35
Exec error : Error move mesh triangles was reverse
  -- number :1
Exec error : Error move mesh triangles was reverse
  -- number :1
 err code 7 ,  mpirank 0
FreeFem++ returned error 7
```

FIGURE 3.15 – Affichage d'erreur

Exemple 3.4. considérons le problème suivant :

$$\begin{cases} u_t - \nabla \cdot (k \nabla u) = 0, & \text{sur } \Omega \times [0,T] \\ u(x,y,0) = u_0(x,y) & \text{sur } \Omega \\ k\dfrac{\partial u}{\partial n} + \alpha(u - u_e) = 0 & \text{sur } \partial\Omega \times [0,T] \end{cases} \tag{3.6}$$

tel que : $u = u(x, y, t)$ représent la température à la position (x, y) et le temps t
$u_0(x, y)$ est l'initiale température. Pour simplifier on prend $\Omega = [0, 6] \times [0, 1]$

$$k(x, y) = \begin{cases} 1 \text{ sur } y < 0.5 \\ 0.2 \text{ si non} \end{cases} \tag{3.7}$$

Comme dans l'exemple précédent, la condition aux limites reflète la loi de refroidissement de Newton. Pour obtenir la formulation variationnelle de 3.6, on procède comme dans le cas stationnaire. Pour commencer, on multiplie l' EDP par une fonction test $v = v(x, y)$. Ensuite, on intégre sur Ω et on applique l'intégration par parties. On obtient :

$$\int_\Omega \frac{\partial u}{\partial t} v \; dxdy - \int_{\partial\Omega} k \frac{\partial u}{\partial n} v \; d\Gamma + \int_\Omega \nabla u \nabla v dxdy = 0$$

On pose les conditions aux limites, on obtient :

$$\int_\Omega \frac{\partial u}{\partial t} v \; dxdy - \int_{\partial\Omega} \alpha(u - u_e) v \; d\Gamma + \int_\Omega k \nabla u \nabla v \; dxdy = 0 \qquad \forall v \in H_0^1.$$

Aprés on va faire la discrétisation du temps par utiliser le schéma implicite d'Euler.
On note par : $dt = T/N$, avec $N \in \mathbb{N}$ fixe, et $u_n = u(t_n)$, avec $t_n = n \; dt$, $0 \leqslant n \leq N$.
Alors, on a : $\dfrac{\partial u}{\partial t} = \dfrac{u_n - u_{n-1}}{dt}$ En substituant cette expression dans (3.6)

$$\int_\Omega \frac{u_n - u_{n-1}}{dt} dxdy + \int_{\partial\Omega} \alpha(u_n - u_e) v \; d\Gamma + \int_\Omega k \nabla u_n \nabla v \; dxdy = 0 \quad \forall v \in H_0^1.$$

Qu'elle est la formulation discrète faible qu'on implimente dans FreeFem ++ suivant :

Le code de ce problème

```
1  //équation de chaleur transitoire
2  func u0=10+90*x/6;      //température initiale
3  func k=1.8*(y<0.5)+0.2;     // conductivité thermique
4  real ue=25;     //température extérieure
5  real alpha=0.25;
6  real T=0.7;     //temps finale
7  real dt=0.1;     //pas de temps
8  mesh Th=square(30,5,[6*x,y]);     // domaine
9  fespace Vh(Th,P1);     // Éléments finis de Lagrange P1
10 Vh u=u0,v,uold;
11 Vh[int] sol((T/dt)+1);
12 //formulation variationnelle. Schéma d'Euler implicite
13 problem calor(u,v)=int2d(Th)(u*v/dt+k*(dx(u)*dx(v)+dy(u)*dy(v)))
14 +int1d(Th,1,2,3,4)(alpha*u*v)
15 -int1d(Th,1,2,3,4)(alpha*ue*v)
16 -int2d(Th)(uold*v/dt);
17 real t = 0;
18 for (int m = 0; m ≤ 0.7/dt; m++){
19    // Update
20     t = t+dt;
21     uold=u;
22     // Solve
23     calor;
```

```
24      // Plot
25     plot(u,fill=1, wait=true);
26  }
```

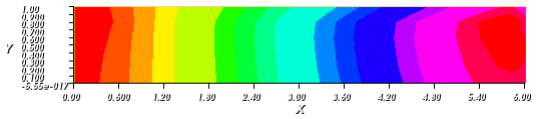


FIGURE 3.16 – Début de la solution

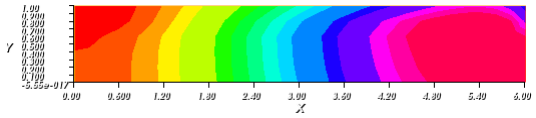


FIGURE 3.17 – Fin de la solution

3.1.2 Schéma d'euler explicite

Exemple 3.5. Considérons le probléme linéaire de Sobolev suivant

$$\begin{cases} u_t - \Delta u_t - \Delta u = f(t) & \text{sur } \Omega \times [0,T] \\ u = 0 & \text{sur } \partial\Omega \times [0,T] \\ u(0) = u_0 & \text{sur } H_0^1(\Omega) \end{cases} \tag{3.8}$$

la formulation variationnelle est

$$\int_\Omega u_t v \, dx - \int_\Omega \Delta u_t v dx - \int_\Omega \Delta u v \, dx = \int_\Omega f v \, dx$$

$$\text{c'implique,} \int_\Omega u_t v \, dx + \int_\Omega \nabla u_t \nabla v \, dx + \int_\Omega \nabla u \nabla v \, dx = \int_\Omega f v \, dx$$

$$\text{alors,} \frac{d}{dt}\int_\Omega uv + \frac{d}{dt}\int_\Omega \nabla u \nabla v \, dx + \int_\Omega \nabla u \nabla v \, dx = \int_\Omega f v \, dx$$

tel que $f = x*(1-x)*y*(1-y)*cos(1)+2*(sin(1)+cos(t))*(x-x*x)*(y-y*y)$
et $v \in H_0^1(\Omega)$

Le code de ce problème

```
1  int np=5;    //finesse du maillage
2  int[int] lab=[1,5,3,4];    //étiquetage des bords
3  mesh Th1 =square(2*np,2*np,[2*(x-1),2*(y-0.5)],label=lab);
4  border b1(t=-pi/2.,pi/2.){x=cos(t); y=sin(t); label=2;}
5  border b2(t=0,1){x=0.; y=1-2*t; label=5;}
6  mesh Th2 = buildmesh(b1(pi*np) + b2(2*np));
7  mesh Th = Th1 + Th2;
8  plot(Th,wait=1);
9  int n = 150;
10 fespace Vh(Th, P1) ;     //la définition de lespace Vh
```

```
Vh uh, vh, uhn ;
real t=0, T=1, dt=0.1, dt2=1, t1=1, erreur=0 ;
real [int] erreur2(10) ;
erreur2=0 ;
func f=x*(1-x)*y*(1-y)*cos(t1)+2*(sin(t1)+cos(t))*(x-x*x)*(y-y*y) ; ;     //second
    membre
func u=x*(1-x)*y*(1-y)*sin(t1) ;     //solution exacte
func dxu=(1-2*x)*y*(1-y)*sin(t1) ;  func dyu=x*(1-x)*(1-2*y)*sin(t1);     // le
    gradient de la solution exacte
problem Eulerexplicite(uh,vh, solver=LU)=     //la formulation variationnelle
int2d(Th)( (dx(uh)*dx(vh)+dy(uh)*dy(vh))+uh*vh )
-int2d(Th)( (dt*f+uhn)*vh)
-int2d(Th)((1-dt)*(dx(uhn)*dx(vh)+dy(uhn)*dy(vh)))
+on(1, 2, 3, 4, uh=0) ;     //on peut modifier la condition sur le bord
int i=0 ;
for(i ; i<10 ; i++){
dt=dt/2; t=dt; uhn=0; erreur=0 ;

Eulerexplicite ;
uhn=uh ;
}
i=0 ;
for(i ; i<10 ; i++){
dt2=dt2/2 ;
plot(uhn ,fill=1 ,value=true);
}
```

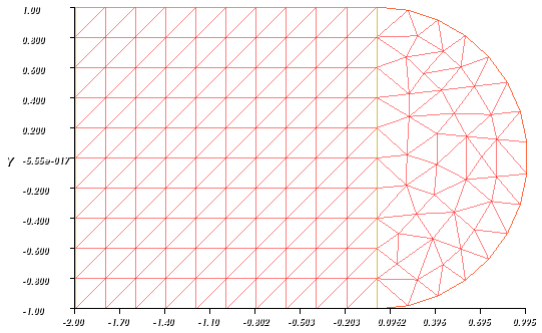


FIGURE 3.18 – Maillage

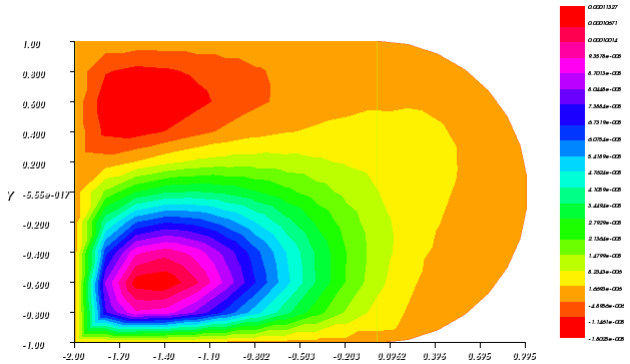


FIGURE 3.19 – Début de la solution

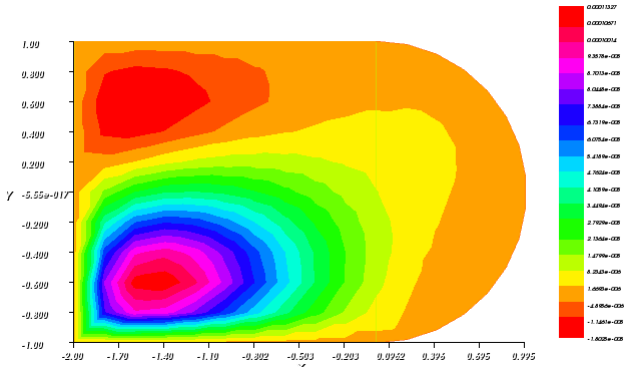


FIGURE 3.20 – Fin de la solution

3.1.3 Schéma de Rung-Kutta

Du même problème on applique le schéma de Rung-Kutta.

Exemple 3.6. .

Le code de ce problème

```
int np=5;      //finesse du maillage
int[int] lab=[1,5,3,4];      //étiquetage des bords
mesh Th1 = square(2*np,2*np,[2*(x-1),2*(y-0.5)],label=lab);
border b1(t=-pi/2.,pi/2.){x=cos(t); y=sin(t); label=2;}
border b2(t=0,1){x=0.; y=1-2*t; label=5;}
mesh Th2 = buildmesh(b1(pi*np) + b2(2*np));
mesh Th = Th1 + Th2;
plot(Th,wait=1);
 //inversion de centre (≠ 3,0), puis rotation
 //dangle pi/3
 //en écriture complexe
complex Z0 = -3,Z1=exp(1i*pi/3);
 //Affixe complexe du point courant :
func Z = x + (1i)*y;
func X = real(Z1*(Z0 + 1./(Z-Z0)));
func Y = imag(Z1*(Z0 + 1./(Z-Z0)));
Th = movemesh(Th,[X,Y]);
plot(Th,wait=1);

int n = 20;
real t=0, t1=1 , T=20, dt=0.1, erreur=0 ,erreur2=0 ;
fespace Vh (Th, P1) ;
Vh uh, vh, uhn, uh1, uhn1 ;
func g=0;
func f1=x*(1-x)*y*(1-y)*cos(t1)+2*(sin(t1)+cos(t))*(x-x*x)*(y-y*y) ;
func f2=x*(1-x)*y*(1-y)*cos(t1+dt)+2*(sin(t1+dt)+cos(t+dt))*(x-x*x)*(y-y*y) ;
func u=x*(1-x)*y*(1-y)*sin(t1) ;
func dxu=(1-2*x)*y*(1-y)*sin(t1) ;
func dyu=x*(1-x)*(1-2*y)*sin(t1) ;
problem Eulerexplicite(uh1,vh, solver=LU)=
int2d(Th)( (dx(uh1)*dx(vh)+dy(uh1)*dy(vh))+uh1*vh)
-int2d(Th)( (dt*f1+uhn)*vh)
-int2d(Th)((1-dt)*(dx(uhn)*dx(vh)+dy(uhn)*dy(vh)))
```

```
+on(1, 2, 3, 4,uh1=g);
problem RungeKutta(uh,vh, solver=LU)=
int2d(Th)( (dx(uh)*dx(vh)+dy(uh)*dy(vh))+uh*vh)
-int2d(Th)( (0.5*dt*f1+uhn)*vh)
-int2d(Th)((1-0.5*dt)*(dx(uhn)*dx(vh)+dy(uhn)*dy(vh)))
-int2d(Th)(0.5*dt*(f2*vh-(dx(uh1)*dx(vh)+dy(uh1)*dy(vh))))
+on(1, 2, 3, 4, uh=g) ;
t=dt; uhn=0;t1=0 ;
int i=0;
for(i ; i< T; i++){
Eulerexplicite ;
RungeKutta ;
uhn=uh ;
t1=t1+dt ;
erreur=max(erreur, sqrt(int2d(Th)(square(dxu-dx(uh))+square(dyu-dy(uh))))) ;
plot(uh, fill=1, value=true) ;
}
```

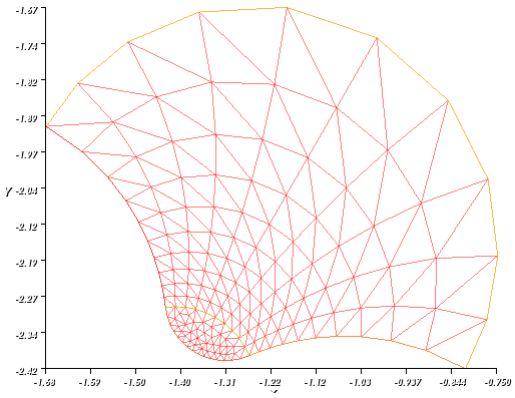


FIGURE 3.21 – Maillage

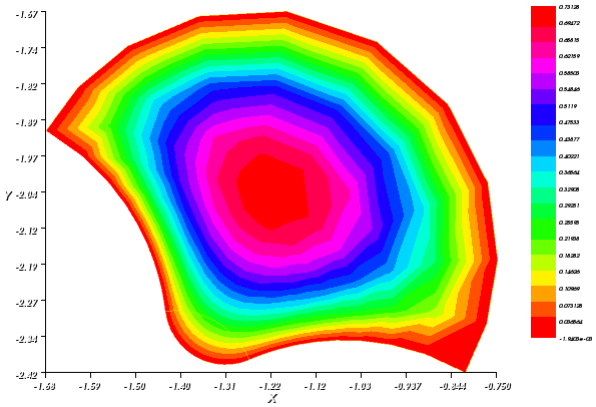


FIGURE 3.22 – Début de la solution

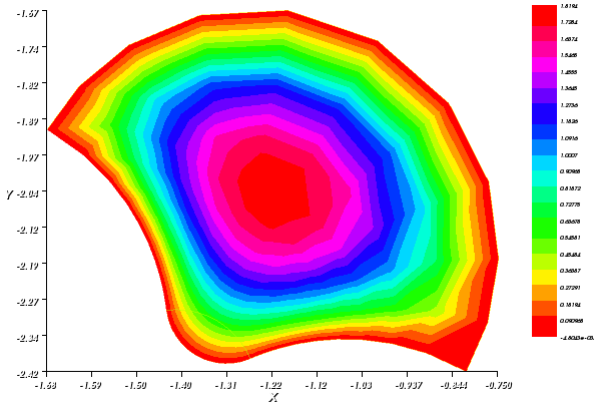


FIGURE 3.23 – Fin de la solution

3.1.4 Schéma implicite avec C.L de Robin

Exemple 3.7. Soit Ω un carré.

$$\begin{cases} \dfrac{\partial u}{\partial t} - \mu\Delta u = f & \text{sur } \Omega \times]0,T[\\ \mu\dfrac{\partial u}{\partial n} + \alpha(u - u_e) = 0 & \text{sur } \Gamma_2 \cup \Gamma_3 \\ u = u_e & \text{sur } \Gamma_1 \cup \Gamma_4 \\ u(x,y,0) = u_0(x) & \text{sur } \Omega \end{cases} \tag{3.9}$$

avec un coefficient de viscosité positif μ et des conditions aux limites de Neuman .
On résout 3.9 par méthode des éléments finis en espace et différences finies en temps.
Pour obtenir la formulation variationnelle, multiplier par la fonction de test v les deux côtés de l'équation et appliquer la formule de Green :

$$\int_\Omega \frac{\partial u}{\partial t} v dx + \mu \int_\Omega \nabla u \nabla v dx + \alpha \int_{\Gamma_2 \cup \Gamma_3} uv d\sigma = \int_\Omega f v dx + \alpha \int_{\Gamma_2 \cup \Gamma_3} u_e v d\sigma$$

$$v \in V = \{v \in H^1(\Omega)/v = 0 sur \Gamma_1 \cup \Gamma_4\}, u - u_e \in V$$

Schéma implicite :

$$\int_\Omega \frac{u^n - u^{n-1}}{dt} v dx + \mu \int_\Omega \nabla u^n \nabla v dx + \alpha \int_{\Gamma_2 \cup \Gamma_3} u^n v d\sigma = \int_\Omega f v dx + \alpha \int_{\Gamma_2 \cup \Gamma_3} u_e v d\sigma$$

$$v \in V = \{v \in H^1(\Omega)/v = 0 sur \Gamma_1 \cup \Gamma_4\}, u - u_e \in V, u^0 = u_0, n = 1, 2, ...$$

On pose

$$\alpha = 0.001, \mu = 1, u_e = 4, u_0 = 10,$$

$$f(x,y) = \begin{cases} 20xy & 0.6 \le x \le 0.8, 0.4 \le y \le 0.6 \\ 0 & ailleurs \end{cases}.$$

qu'on peut l'écrire : $f = 20 \cdot \chi_{\{0.6 \le x \le 0.8\}} \cdot \chi_{\{0.4 \le y \le 0.6\}}$

Le code de ce problème

```
// Equation de la chaleur, omega=[ 0,1] *[ 0,1]
// Schéma implicite
real Dx=0.02, Dy=0.02 ;     // declaration des variables
real t, ue = 4., mu = 1., alpha=.001, dt=0.1, Tf=10 ;
mesh Th=square(floor(1./Dx),floor(1./Dy)) ;     //definition du maillage
//mesh Th = square(n,m,[ x0+(x1-x0)*x,y0+(y1-y0)*y] ) ;
plot(Th,ps="square.jpg",cmm="maillage") ; // visualisation du maillage

// Définition de lespace déléments finis
fespace Vh(Th,P1) ;     // ensemble déléments finis
Vh uh,vh,uh0=10. ;
func f=20.*(.6≤x & x≤ .8)*(.4≤ y & y≤.6) ; // second membre

// Résolution du problème variationnel
problem chaleur(uh,vh) = int2d(Th)(uh*vh/dt) -
int2d(Th)(uh0*vh/dt) +
int2d(Th)((dx(uh)*dx(vh) + dy(uh)*dy(vh))*mu) -
int2d(Th)(f*vh)+ int1d(Th,2,3)(alpha*uh*vh) -
int1d(Th,2,3)(ue*vh*alpha) + on(1,4,uh=ue) ;
int kk=0 ;
for ( real t=0. ;t<Tf ;t+=dt)     // boucle du temps t
{
chaleur;
uh0=uh ;
if ( !(kk % 20))     // kk%20= le reste de la division kk/20
/ !(kk%20)=0 si le reste de la division nest pas nul, =1 si le reste est nul
// !(kk%20)= true or false
{plot(uh,cmm="t="+t+"[sec] ",dim=3,fill=true,value=true,wait=1) ; \
plot(uh,cmm="t="+t+"[sec]",dim=2,fill=true,value=true,wait=1) ;}
kk+=1 ;
}

// Visualisation des résultats numériques
cout<<"Taille.uh="<<uh[].n <<endl ;     // La taille du tableau
cout<<"Max.uh="<<uh[] .max<<" ; Min.uh="<<uh[] .min<<endl ;    // Max et min
cout<<"Norme.uh l1="<<uh[].l1<<endl ;     // Norme l1
cout<<"Norme.uh l2="<<uh[].l2<<endl ;     // Norme l2
cout<<"Norme.uh sup="$<<uh[]$.linfty$<<endl ;     // Norme sup
cout<<"Somme des termes="<<uh[].sum<<endl ;     // Somme
cout<<"uh[]="<<uh[]<< endl ;
```

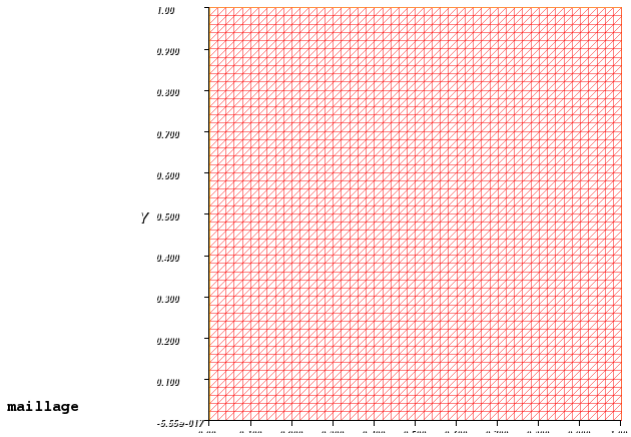


FIGURE 3.24 – Maillage

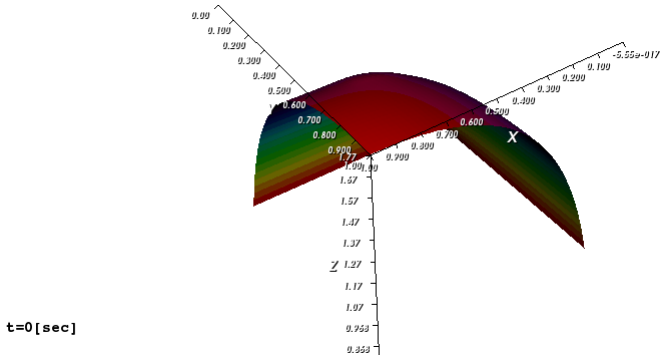


FIGURE 3.25 – Solution *uh*

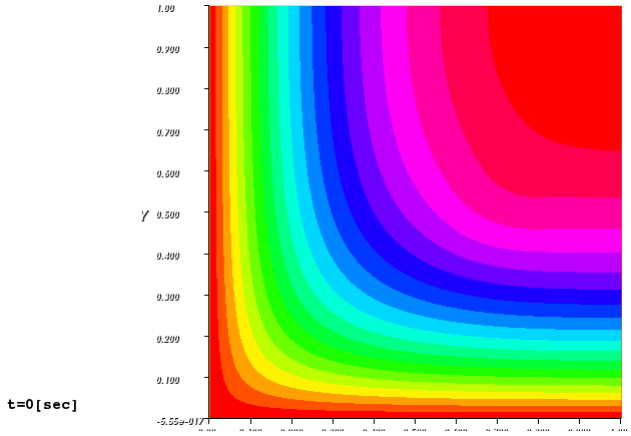


FIGURE 3.26 – Projection de la solution

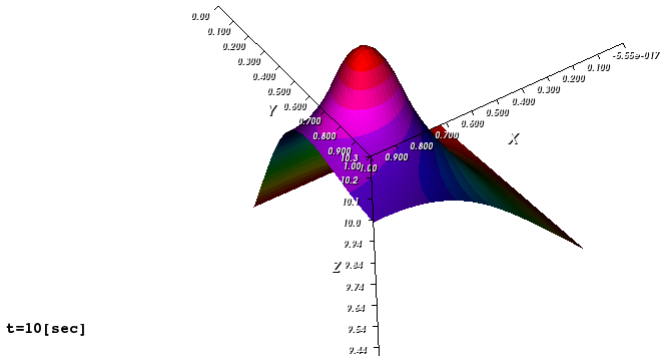


FIGURE 3.27 – Solution uh

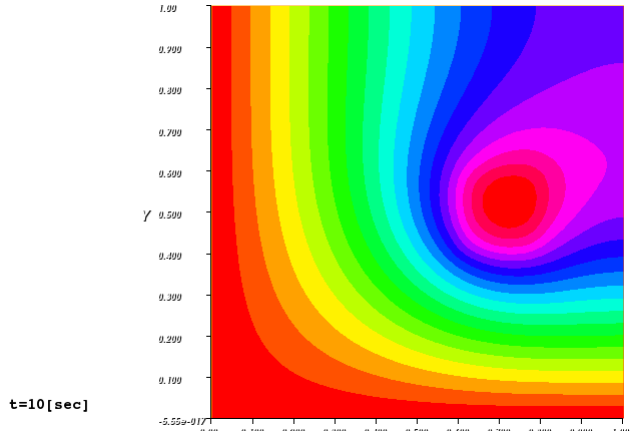


FIGURE 3.28 – Projection de la solution

Exemple 3.8. On prent Ω est un disque avec un trou.

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u = f \quad \text{sur } \Omega \times]0, T[\\ u = 20 \quad \text{sur } C_0 \\ u = 60 \quad \text{sur } C_1 \\ u(x, y, 0) = u_0(x) \quad \text{sur } \Omega \end{cases} \tag{3.10}$$

La formulation variationnelle

$$\int_\Omega u_t v - \int_\Omega \Delta u v = \int_\Omega f v$$

$$\text{Alors, } \int_\Omega u_t v \, dx - \int_{\partial\Omega} \nabla u v \, d\sigma + \int_\Omega \nabla u \nabla v \, dx = \int_\Omega f v dx$$

$$\text{C'implique } \int_\Omega \frac{u^n - u^{n-1}}{dt} v dx - \int_{\partial\Omega} \nabla u^n v \, d\sigma + \int_\Omega \nabla u^n \nabla v \, dx = \int_\Omega f v dx$$

tel que $f = 20 \cdot \chi_{\{6 \leq x \leq 8\}} \cdot \chi_{\{4 \leq y \leq 6\}}$
$v \in H^1(\Omega)$

Le code de ce problème

```
// Equation de la chaleur, omega=disque avec un trou
// Schéma implicite
//verbosity=0. ;
int C1=1, C2=2 ;
 border C0(t=0,2*pi){x=5*cos(t) ; y=5*sin(t) ;label=C2 ;};
 border C11(t=0,1){ x=-1+2*t; y=3 ; label=C1 ;};
 border C12(t=0,1){ x=1 ; y=3-6*t; label=C1 ;};
 border C13(t=0,1){ x=1-2*t; y=-3 ; label=C1 ;};
border C14(t=0,1){ x=-1 ; y=-3+6*t; label=C1 ;};
//plot(C0(50)+ C11(5)+C12(20)+C13(5)+C14(20)) ;
 mesh Th=buildmesh(C0(50)+ C11(5)+C12(20)+C13(5)+C14(20)) ;
plot(Th,ps="circle.jpg",cmm="maillage") ; // visualisation du maillage
fespace Vh(Th,P1) ;
Vh uh,vh,uh0=10. ;
real dt=0.1, Tf=10 ;
func f=20.*(6.≤x & x≤8.)*(4.≤y & y≤6.) ;
macro Grad(u)[dx(u),dy(u)]//
problem chaleur(uh,vh) = int2d(Th)(uh*vh/dt + Grad(uh)*Grad(vh))
- int2d(Th)(uh0*vh/dt + f*vh) +on(C1,uh=60)+on(C2,uh=20) ;
int kk=0 ;
for (real t=0. ;t<Tf ;t+=dt)
{
chaleur;
uh0=uh ;
if ( !(kk % 20))     // kk %20= le reste de la division kk/20
// !(kk%20)=0 si le reste de la division nest pas nul, =1 si le reste est nul
// !(kk%20)= true or false
{
plot(uh,cmm="t="+t+"[sec]",dim=3,fill=true,value=true,wait=1) ;
plot(uh,cmm="t="+t+"[sec]",dim=2,fill=true,value=true,wait=1) ;
}
kk+=1 ;
{
cout<<"uh[]="<<uh[]<<endl ;
cout<<"Taille.uh="<<uh[].n<<endl ;     // La taille du tableau
cout<<"Max.uh="<<uh[].max<<" ; Min.uh="<<uh[].min<<endl ;     // Max et min
cout<<"Norme.uh l1="<<uh[].l1<<endl ;     // Norme l1
cout<<"Norme.uh l2="<<uh[].l2<<endl ;     // Norme l2
cout<<"Norme.uh sup="<<uh[].linfty<<endl ;     // Norme sup
cout<<"Somme des termes="<<uh[].sum<<endl ;     // Somme
cout<<"uh[]="<<uh[]<<endl ;
```

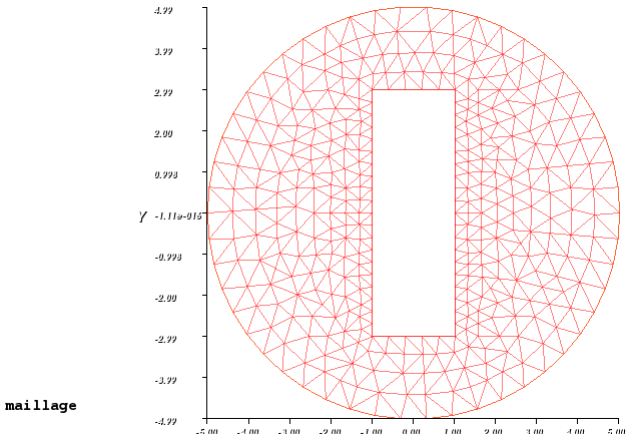


FIGURE 3.29 – Maillage

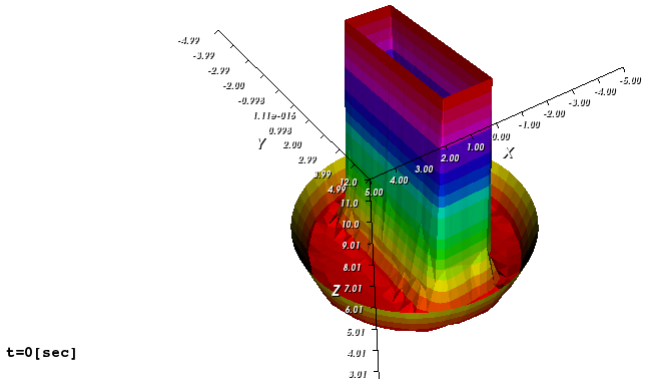


FIGURE 3.30 – Solution *uh*

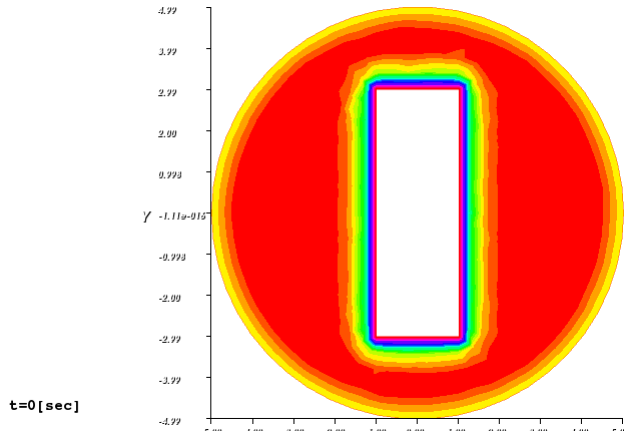


FIGURE 3.31 – Projection de la solution *uh*

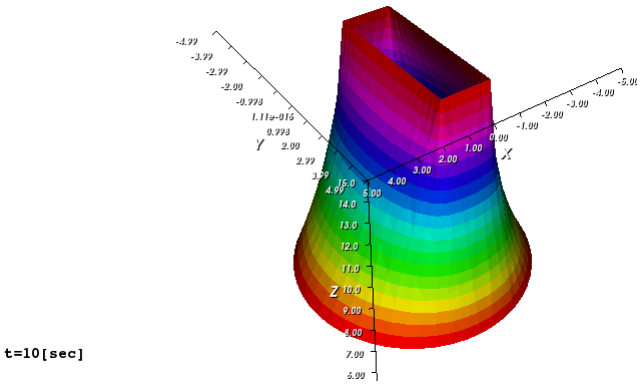


FIGURE 3.32 – Solution *uh*

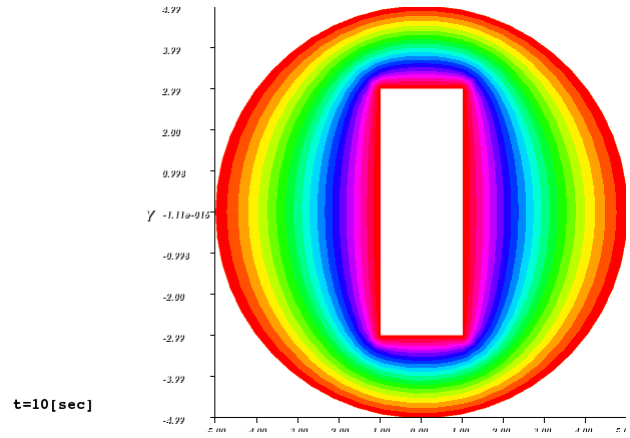


FIGURE 3.33 – Projection de la solution

Conclusion

Freefem++ est un logiciel en constante évolution car il est facile d'ajouter de nouveaux outils. Nous avons essayé d'illustrer cela avec les exemples.

Dans l'avenir, nous prévoyons d'inclure plus d'outils pour l'anisotropie tridimensionnelle différenciation adaptative et automatique des mailles par surcharge de l'opérateur, transparente intégration d'outils parallèles pour libérer l'utilisateur de la programmation de bas niveau.

Bibliographie

[1] Allaire, G. (2016). Analyse variationnelle des équations aux dérivées partielles : promotion 2014, année 2, deuxième semestre, période 3, MAT431. École polytechnique.

[2] Allaoua, M. Simulation numérique en FreeFem++ de quelques équations de la physique mathématiques.

[3] Auliac, S. (2014). Développement d'outils d'optimisation pour freefem++ (Doctoral dissertation, Paris 6).

[4] Bekkouche, F. (2018). Etude théorique et numérique des équations non-linéaires de Sobolev (Doctoral dissertation, Université de Valenciennes et du Hainaut-Cambresis).

[5] Font, R., & Peria, F. (2013). The Finite Element Method with FreeFem++ for beginners. Electronic Journal of Mathematics & Technology, 7(4).

[6] Frederic, H. (2022). FreeFEM Documentation. https://github.com/FreeFem/FreeFem-doc/raw/pdf/FreeFEM-documentation.pdf

[7] Hecht, F., & Pironneau, O. (2013). freefem+.

[8] Henrot, A.(2016). Polycopie de cours M1 Équations aux dérivées partielles.École des Mines de Nancy.

[9] Herbin, R. (2001). Polycopie de cours M1 Analyse numérique des EDP.

[10] Le Dret, H. (2010). Notes de cours M2−Équations aux dérivées partielles elliptiques. Université Pierre et Marie Curie, Paris, 4.

[11] Lucquin, B. (2004). Équation aux dérivées partielles et leurs approximations : niveau M1. Ellipses éd. Marketing.

[12] Maury, B. (2007). Méthode des éléments finis en élasticité : programme d'approfondissement, sciences de l'ingénieur, simulation et modélisation, innovation technologique : promotion 2005, année 3, période 1⋯ École polytechnique.

[13] Pantz, O. (2001). An Introduction to FreeFem++. École Polytechnique.