

Modélisation mult-échelle des phénomènes génériques d'imprégnation sous sollicitations mécaniques de mèches pour l'élaboration de composites à matrice organique.

Point avancement n°00 du 05/10/2022

Objectifs

Les procédés visés impliquent l'écoulement d'un fluide à travers un milieu poreux déformable qui mettent en jeu des phénomènes :

- mécaniques dus aux sollicitations de la mèche durant son imprégnation et à l'opération de compactage lors du bobinage pour le procédé d'enroulement, ou de l'application d'un rouleau presseur pour le procédé ATL/AFP ;
- convectifs liés à l'écoulement de la résine lors de la phase d'imprégnation des mèches, puis de la phase de compaction ;
- chimiques et thermiques dus à la polymérisation de la résine au cours du procédé.

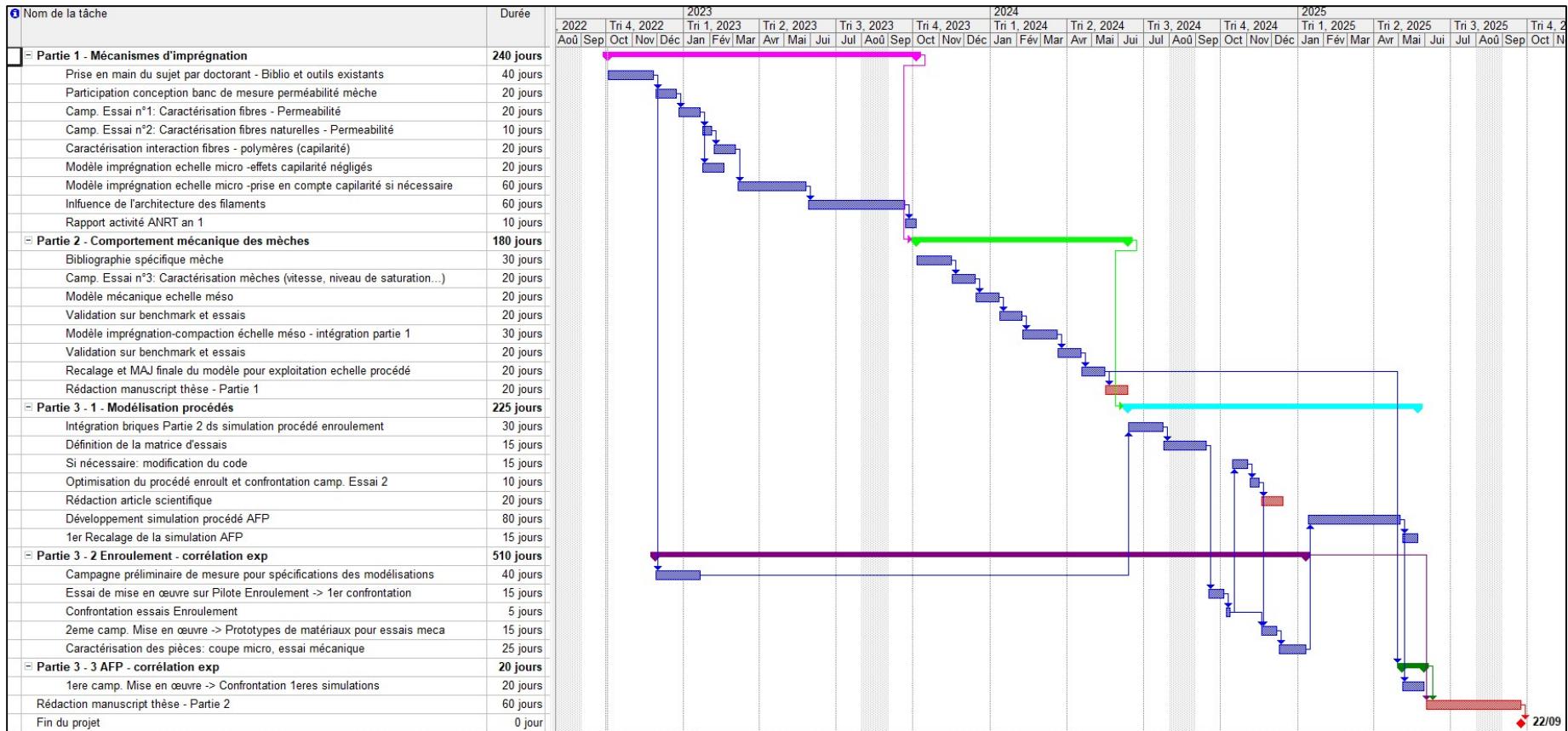
Rappel du sujet

L'objectif de cette thèse est de proposer et d'identifier un modèle permettant de décrire le couplage entre le comportement thermo mécanique de la mèche et sa saturation en polymère. Ce modèle permettra ainsi de prédire l'état de déformation et donc le taux de renfort pour un niveau de saturation donné.

Le travail est décomposé en 3 parties:

1. Décrire les mécanismes d'imprégnation à différentes échelles.
2. Identifier le comportement mécanique d'une mèche.
3. Intégrer les modèles méso déduits des étapes précédentes dans des modèles de simulation numérique des procédés d'enroulement filamentaire et de dépôse de mèches/bandes (ATL/AFP).

Planning prévisionnel : global



Planning prévisionnel : local



Pt 2

Pt 1

Bilan de la dernière réunion

VIDE.



Ordre du jour

- A. Mathématiques
- B. Physiques
- C. Outils et méthodes informatiques
- D. Modélisation à l'échelle macroscopique
- E. Modélisation à l'échelle mésoscopique
- F. Modélisation à l'échelle microscopique
- G. Actions à venir
- H. Bibliographie

A. Mathématiques

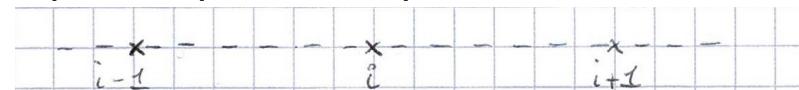
- Déploiement des trois grandes familles de méthodes numériques pour discréteriser des E.D.P classiques
- Traitement de maillages uniformes ou non
- Application à la diffusion, linéaire et non linéaire

A. Différences finies 1.D sur maillage uniforme

Trois schéma classiques pour la dérivée première d'une fonction

 Décentré amont	 Différences centrées	 Décentré aval
$u_i \approx \frac{u_i - u_{i-1}}{\Delta x}$	$u_i \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}$	$u_i \approx \frac{u_{i+1} - u_i}{\Delta x}$

Schéma du stencil trois points pour le laplacien

 stencil trois points du Laplacien
$u''_i \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$

A. Justification sur grille uniforme

On souhaite approcher la dérivée seconde u'' en les nœuds d'une discréttisation $\{x(i)\}$.

On choisit le nombre de points d'interpolation du schéma.

On remplace les grandeurs en chaque nœud $u(j)$ par son développement en série entière (développement de Taylor)

On résout le système linéaire associé, qui donne le schéma.

Illustration pour le stencil à trois points du laplacien :

$$\begin{aligned}
& u_i'' \sim \alpha u_{i-1} + \beta u_i + \gamma u_{i+1} \\
\Leftrightarrow & u_i'' \sim \alpha(u_i - \Delta x u_i' + \frac{\Delta x^2}{2} u_i'') + \beta u_i + \gamma(u_i + \Delta x u_i' + \frac{\Delta x^2}{2} u_i'') \\
\Leftrightarrow & u_i'' \sim (\alpha + \beta + \gamma) u_i + (\gamma - \alpha) \Delta x u_i' + (\alpha + \gamma) \frac{\Delta x^2}{2} u_i'' \\
\Leftrightarrow & \begin{cases} \alpha + \beta + \gamma = 0 \\ (\gamma - \alpha) \Delta x = 0 \\ (\alpha + \gamma) \frac{\Delta x^2}{2} = 1 \end{cases} \Rightarrow \begin{cases} \beta = -(\alpha + \gamma) \\ \gamma = \alpha \\ \alpha = \frac{1}{\Delta x^2} \end{cases} \Rightarrow \begin{cases} \alpha = \frac{1}{\Delta x^2} \\ \beta = -\frac{2}{\Delta x^2} \\ \gamma = \frac{1}{\Delta x^2} \end{cases} \\
\Leftrightarrow & u_i'' \sim \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}
\end{aligned}$$

Issu de l'identification des coefficients.

A. Validation de l'ordre de convergence pour la diffusion

Laplacien de Dirichlet

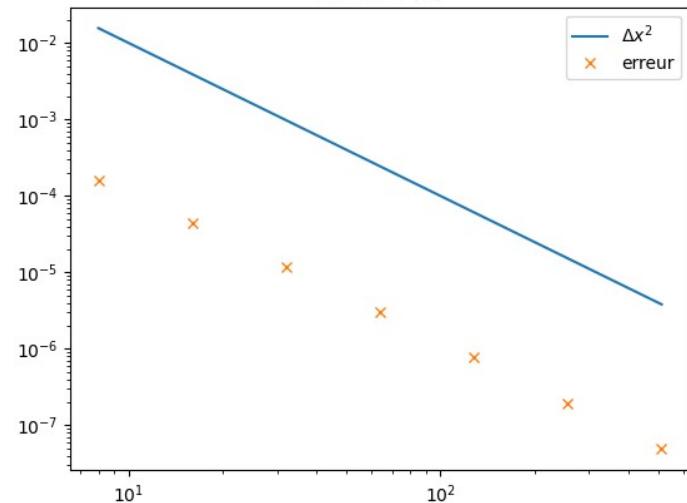
Soit à résoudre : $f: x \mapsto e^x \in \mathcal{E}^\infty([0,1]; \mathbb{R})$,

$$\begin{cases} \Delta u(x) = f(x) & (0, 1) \\ u(0) = 1 \\ u(1) = e \end{cases}$$

L'unique solution à ce problème est donnée par

$$u: [0, 1] \rightarrow \mathbb{R}; x \mapsto e^x$$

Erreur $L^2(\Omega)$



Laplacien mixte Dirichlet / Neumann

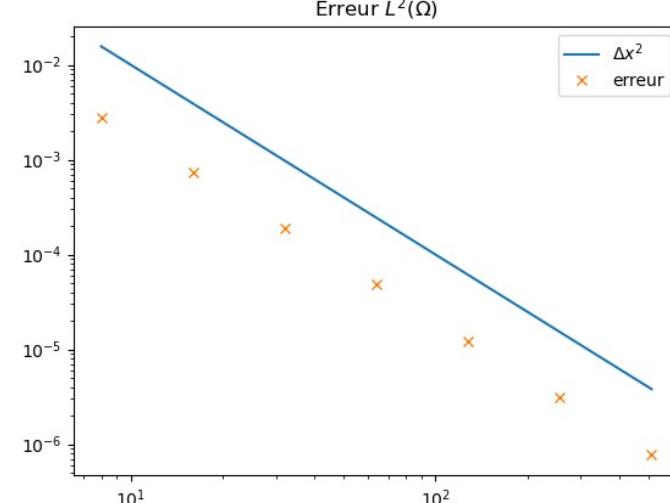
Soit à résoudre : $f: x \mapsto e^x \in \mathcal{E}^\infty([0,1]; \mathbb{R})$

$$\begin{cases} \Delta u(x) = f(x) & (0, 1) \\ u(0) = 1 \\ u'(1) = e \end{cases}$$

L'unique solution de ce problème est donnée par

$$u: [0, 1] \rightarrow \mathbb{R}; x \mapsto e^x$$

Erreur $L^2(\Omega)$



A. Schéma de diffusion 1.D sur maillage non uniforme

Identification des coefficients

$$\begin{aligned}
 u_i''' &= \alpha u_{i-1} + \beta u_i + \gamma u_{i+1} \quad \Delta x_- := x_i - x_{i-1}; \quad \Delta x_+ := x_{i+1} - x_i \\
 \Leftrightarrow u_i''' &= \alpha(u_i - \Delta x_- u_{i-1} + \frac{\Delta x_-^2}{2} u_i'') + \beta u_i + \gamma(u_i + \Delta x_+ u_{i+1} + \frac{\Delta x_+^2}{2} u_i'') \\
 \Leftrightarrow u_i''' &= (\alpha + \beta + \gamma) u_i + (\gamma \Delta x_+ - \alpha \Delta x_-) u_{i-1}'' + \frac{1}{2}(\alpha \Delta x_-^2 + \gamma \Delta x_+^2) u_i'' \\
 \Leftrightarrow \begin{cases} \alpha + \beta + \gamma = 0 \\ \gamma \Delta x_+ - \alpha \Delta x_- = 0 \\ \frac{1}{2}(\alpha \Delta x_-^2 + \gamma \Delta x_+^2) = 1 \end{cases} & \begin{array}{l} (1) \\ (2) \\ (3) \end{array} \quad \begin{array}{l} (2) \Leftrightarrow \gamma = \alpha \frac{\Delta x_-}{\Delta x_+} \\ (3) \Leftrightarrow \alpha \Delta x_-^2 + \alpha \Delta x_- \Delta x_+ = 2 \\ \Leftrightarrow \alpha \Delta x_- (\Delta x_- + \Delta x_+) = 2 \end{array} \\
 \Leftrightarrow \alpha &= \frac{2}{\Delta x_- (\Delta x_- + \Delta x_+)} \\
 (2) \Leftrightarrow \gamma &= \frac{2}{\Delta x_+ (\Delta x_- + \Delta x_+)} \\
 (1) \Leftrightarrow \beta &= -(\alpha + \gamma) \\
 \Leftrightarrow \beta &= -\left(\frac{2}{\Delta x_- (\Delta x_- + \Delta x_+)} + \frac{2}{\Delta x_+ (\Delta x_- + \Delta x_+)} \right) \\
 \text{d'où} \\
 u_i''' &\sim \frac{2}{\Delta x_- (\Delta x_- + \Delta x_+)} u_{i-1}'' + (-1) \left(\frac{2}{\Delta x_- (\Delta x_- + \Delta x_+)} + \frac{2}{\Delta x_+ (\Delta x_- + \Delta x_+)} \right) u_i'' + \frac{2}{\Delta x_+ (\Delta x_- + \Delta x_+)} u_{i+1}''
 \end{aligned}$$

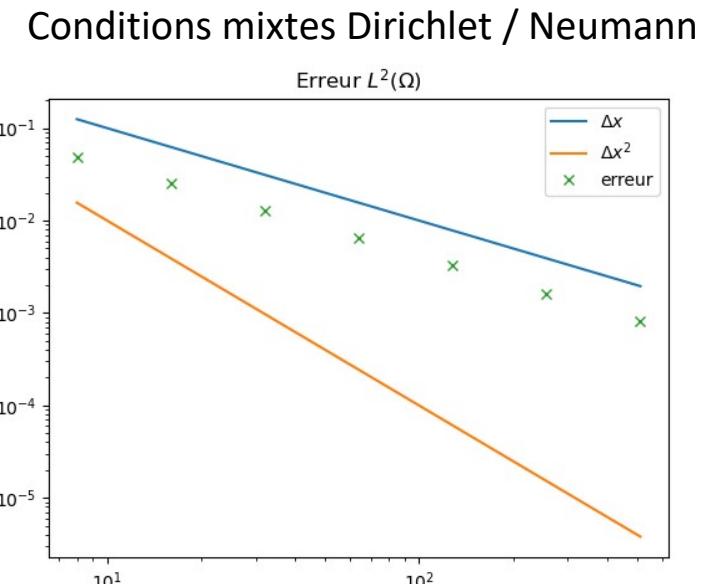
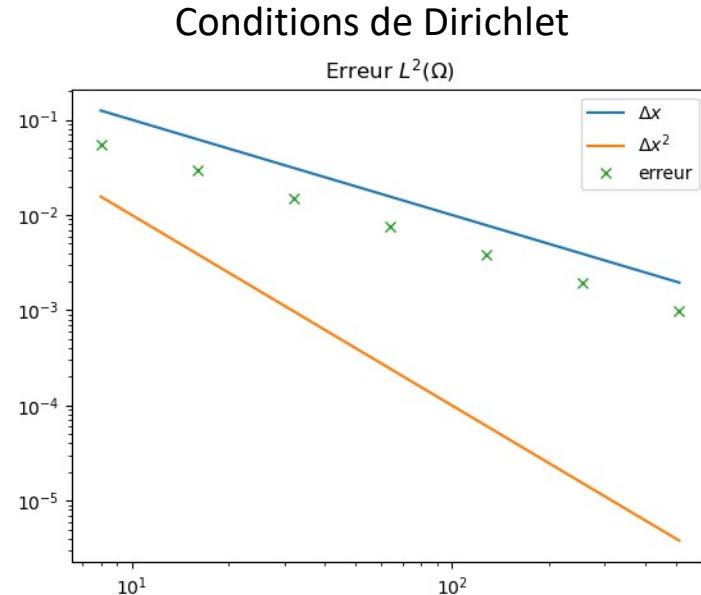
A. Illustration des ordres de convergence

Validation analytique sur le modèle de l'exponentielle, même problème que précédemment.

A gauche : conditions de Dirichlet aux bords,

A droite : conditions mixtes, Dirichlet à gauche et Neumann à droite

Le schéma proposé précédemment converge à l'ordre 1.



A. Problème d'optimisation pour l'équilibre mécanique

Soient $u: x \mapsto x^2 \in \mathcal{C}^\infty([0,1]; \mathbb{R})$

$Q: y \mapsto e^y \in \mathcal{C}^\infty(\mathbb{R}; \mathbb{R})$

On calcule :

$$\partial_x u : x \mapsto 2x$$

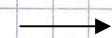
$$\partial_x Q(u) : x \mapsto 2x e^{x^2} \leftarrow = Q'(u) u'$$

on bien que :

$$\begin{aligned} \partial_x \{ Q(u) \partial_x u \} &= \{ \partial_x Q(u) \} \partial_x u + Q(u) \partial_{xx}^2 u \\ &= (2x)^2 e^{x^2} + 2e^{x^2} \\ &= 2e^{x^2}(2x^2 + 1) \\ &= 2Q(u)(2u + 1) \\ &\underbrace{\quad\quad\quad}_{f(u)} \end{aligned}$$

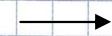
Donc u est solution du problème non linéaire :

Résidu continu

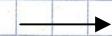


$$\left\{ \begin{array}{l} \partial_x (Q(u) \partial_x u) - f(u) = 0 \\ u(0) = 0 \\ u(1) = 1 \end{array} \right.$$

Dirichlet



Dirichlet



Déplacement convexe
non linéaire

Comportement convexe
non linéaire

Calcul du second membre :
quel effort donne la solution
souhaitée ?

Expression du résidu en
fonction de la solution, pour
des questions
d'implémentation

Problème statique
non linéaire 1.D

A. Panorama des méthodes de Newton

19. The basic Newton's method

The basic Newton's method replaces iteratively in the equation $\mathcal{F} = 0$ the function \mathcal{F} by its first-order expansion around the point $\{U^n, P^n\}$. In other words, the basic Newton's method solves iteratively the linear equation (Fig. 19.1)

$$\mathcal{F}(U^n, P^n, f, g) + \frac{D\mathcal{F}(U^n, P^n, f, g)}{D(U, P)}(U^{n+1} - U^n, P^{n+1} - P^n) = 0.$$

More precisely, the numerical solution of problem (18.1) by the basic Newton's method corresponds to the algorithm:

Step 0. Initialization.

With $\{U^0, P^0\}$ in $\mathbb{R}^{N_h} \times \mathbb{R}^{M_h}$ specified arbitrarily, calculate

$$R_i^0 = \mathcal{F}_i(U^0, P^0, f, g), \quad 1 \leq i \leq N_h + M_h.$$

Step 1. Iterative loop.

Then, for $n \geq 0$, with U^n, P^n, R^n known, compute $U^{n+1}, P^{n+1}, R^{n+1}$ by

(i) solving the linear system (S)

$$\frac{D\mathcal{F}(U^n, P^n, f, g)}{D(U, P)}(V^n, Q^n) = R^n \quad \text{in } \mathbb{R}^{N_h} \times \mathbb{R}^{M_h},$$

(ii) setting $U^{n+1} = U^n - V^n$ and $P^{n+1} = P^n - Q^n$,

(iii) computing $R_i^{n+1} = \mathcal{F}_i(U^{n+1}, P^{n+1}, f, g), 1 \leq i \leq N_h + M_h$.

The algorithm is stopped as soon as the error

$$\text{error} = a_1 \|R^{n+1}\| + a_2 \|V^n\| + a_3 \|Q^n\|$$

Référence [4], partie 2, chapitre
4, p 529 - 546

THEOREM 19.1. Let (U, P) be a solution of $\mathcal{F}(U, P) = 0$, with $D\mathcal{F}(U, P, f, g)/D(U, P)$ invertible and locally Lipschitz continuous. Then, if (U^0, P^0) is sufficiently close to (U, P) , we have

$$\|(U^{n+1}, P^{n+1}) - (U, P)\| \leq \beta \|(U^n, P^n) - (U, P)\|^2.$$

A. Diaporama des méthodes de Newton

20. Newton's method with incremental loading

We have just seen how critical a good initial guess was for ensuring the convergence of the Newton's method. Incremental loading is a computing strategy which provides such initial guesses. In incremental loading, instead of directly computing the final solution, we follow the differential curve

$$\mathcal{A}'(u) \frac{du}{ds} = \frac{df}{ds}, \\ u(0) = u^0,$$

introduced in Section 10 (Fig. 20.1). In our present notation, we have

$$u = (U, P), \quad \mathcal{A}'(u) = \frac{D\mathcal{F}}{D(U, P)}, \\ f(s) = \lambda(s)\{f, g\}, \quad \lambda(s) = s.$$

The construction of the solution curve is then done step by step, each step being solved by the following algorithm:

- (i) Predict the solution $u(\lambda^n + \Delta\lambda)$ by the explicit Euler scheme

$$u(\lambda^n + \Delta\lambda) \approx \tilde{u}^{n+1} = u(\lambda^n) + \Delta\lambda (\mathcal{A}'(u(\lambda^n)))^{-1} \frac{df}{ds}.$$

- (ii) Correct this first-order prediction by solving

$$\mathcal{A}(u(\lambda^n + \Delta\lambda)) = f(\lambda^n + \Delta\lambda)$$

by a Newton's method taking as initial guess the prediction \tilde{u}^{n+1} .

Such a guess \tilde{u}^{n+1} will indeed be close to the corresponding equilibrium solution if the load increment is not too large. As a consequence, the second step of the above algorithm will converge fast in most cases.

A. Diaporama des méthodes de Newton

NEWTON WITH INCREMENTAL LOADING AND LINE DAMPING: DETAILED ALGORITHM

Initialization

We start with $U^0 = 0$, P^0 equal to the hydrostatic pressure at rest, $\lambda = 0$ and with a given load increment $\Delta\lambda$. Then, we increase the value of λ by increments of $\Delta\lambda$ until we reach the value $\lambda = 1$. For each value of λ , starting with the solution (U^0, P^0) obtained at the previous increment, we compute the solution values of (U, P) by a Newton's procedure. More precisely:

Newton's procedure

Step 0. Initialization.

We compute the residual: $R^0 = \mathcal{F}(U^0, P^0, \lambda f, \lambda g)$.

Step 1. Iterative loop.

Then, for $n \geq 0$ and until satisfied, with U^n, P^n, R^n known, we compute $U^{n+1}, P^{n+1}, R^{n+1}$ by

- (i) computing and factoring the tangent operator (if needed):

$$K = \frac{\partial \mathcal{F}(U^n, P^n, \lambda f, \lambda g)}{\partial(U, P)},$$

- (ii) solving the linear system (S):

$$K(\Delta U, \Delta P) = -R^n,$$

- (iii) setting $(U^{n+1}, P^{n+1}) = (U^n, P^n) + (\Delta U, \Delta P)$,
- (iv) correcting this value by line damping,
- (v) and computing the residual: $R^{n+1} = \mathcal{F}(U^{n+1}, P^{n+1}, \lambda f, \lambda g)$.

A. Diaporama des méthodes de Newton

EULER-NEWTON ALGORITHM

Let $\{U^0, P^0, \lambda^0\}$ be given. Then, for $n = 0$ and as long as $\lambda^n \leq 1$,

- (i) compute the “velocity” at step n , that is the solution of

$$\frac{D\mathcal{F}(U^n, P^n, \lambda^n f, \lambda^n g)}{D(U, P, \lambda)}(\dot{U}^n, \dot{P}^n, \dot{\lambda}^n) = 0, \quad (21.5)$$

$$\|(\dot{U}^n, \dot{P}^n, \dot{\lambda}^n)\| = 1; \quad (21.6)$$

- (ii) set the solution at step $(n + 1)$ to the value of the Euler predictor

$$\{U_0^{n+1}, P_0^{n+1}, \lambda_0^{n+1}\} = \{U^n, P^n, \lambda^n\} + \Delta s \{(\dot{U}^n, \dot{P}^n, \dot{\lambda}^n)\};$$

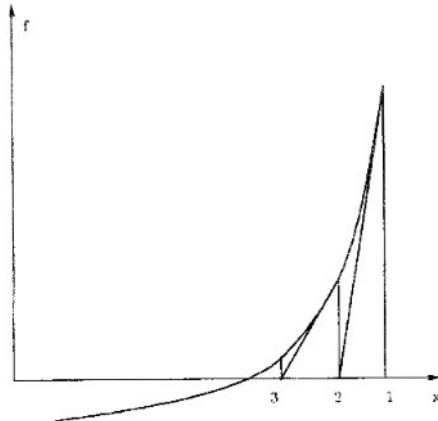
- (iii) project this solution back to the solution curve (21.1) by a Newton’s method (corrector step), that is, iteratively compute the solution of the nonlinear extended system

$$\mathcal{F}(U^{n+1}, P^{n+1}, \lambda^{n+1} f, \lambda^{n+1} g) = 0, \quad (21.7)$$

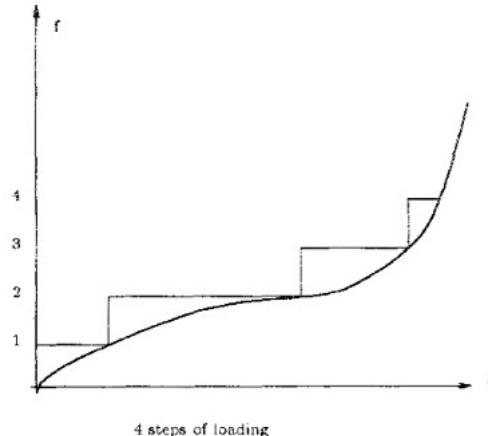
$$\begin{aligned} \dot{U}^n \cdot (U^{n+1} - U_0^{n+1}) + \dot{P}^n \cdot (P^{n+1} - P_0^{n+1}) \\ + \dot{\lambda}^n (\lambda^{n+1} - \lambda_0^{n+1}) = 0. \end{aligned} \quad (21.8)$$

A. Diaporama des méthodes de Newton

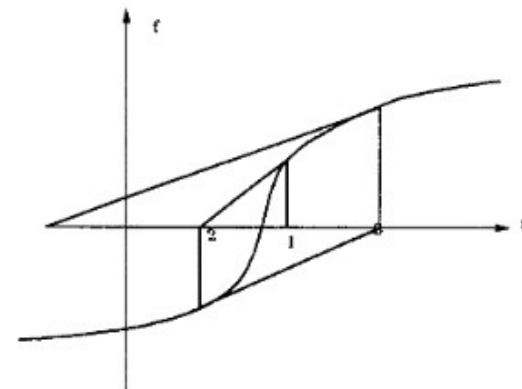
A. Diaporama des méthodes de Newton



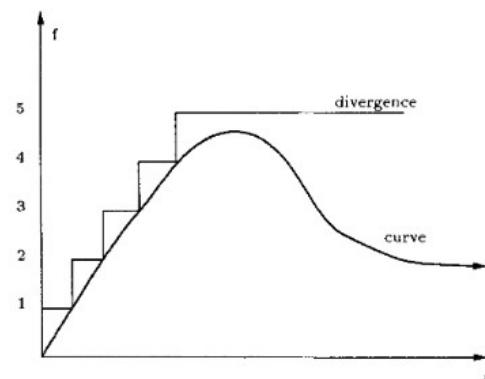
Newton standard



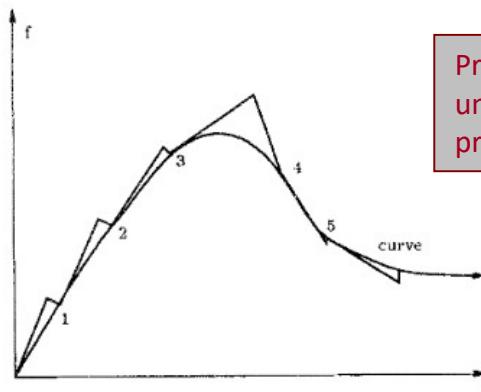
Incrément de chargement



Suite divergente



Pas de solution à l'étape de prédiction



Arclength continuation

Programmer les algorithmes à la main au moins une fois pour comparer avec les fonctions proposées dans le module Python/Scipy/Optimize

A. Les sous-espaces de Krylov

A comprendre, sans urgence. Sujet au mieux secondaire.

Krylov subspace methods

Krylov subspace methods are a family of algorithms for solving $\mathbf{Ax} = \mathbf{b}$ that search for an approximate solution from a Krylov subspace. They are iterative, as opposed to direct, algorithms and usually require a fast matrix-vector product for \mathbf{A} (and possibly \mathbf{A}^T). The most famous Krylov subspace methods are Arnoldi, Lanczos, conjugate gradient, BiCGSTAB, and GMRES. We do not have time to cover them all. Here, we will focus on the GMRES method.

Initial thoughts on Krylov subspaces

Let \mathbf{A} be an invertible matrix and suppose we want to solve $\mathbf{Ax} = \mathbf{b}$, but we only know \mathbf{A} via matrix-vector products, i.e., a function $\mathbf{v} \mapsto \mathbf{Av}$. Let's consider the sequence of vectors $\mathbf{b}, \mathbf{Ab}, \dots, \mathbf{A}^{n-1}\mathbf{b}, \mathbf{A}^n\mathbf{b}, \dots$, (not much else one can consider!). If we consider the first $n+1$, i.e., $\mathbf{b}, \mathbf{Ab}, \dots, \mathbf{A}^{n-1}\mathbf{b}, \mathbf{A}^n\mathbf{b}$, then we are guaranteed for these vectors to be linear dependent ($n+1$ vectors in n -dimensional space must be linear dependent). Therefore, there exist coefficients $\alpha_0, \dots, \alpha_n$ such that

$$\alpha_0\mathbf{b} + \alpha_1\mathbf{Ab} + \dots + \alpha_n\mathbf{A}^n\mathbf{b} = 0.$$

Let k be the smallest integer so that $\alpha_k \neq 0$. Then, since \mathbf{A}^{-1} exists:

$$\mathbf{A}^{-1}\mathbf{b} = -\frac{1}{\alpha_k}(\alpha_{k+1}\mathbf{b} + \dots + \alpha_n\mathbf{A}^{n-k-1}\mathbf{b}). \quad (\text{Weak Cayley-Hamilton theorem})$$

This shows that $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ can be computed by only matrix-vector products. This gives us an idea to search for "good" solutions from Krylov subspaces.

The Krylov subspace

Let \mathbf{A} be a matrix and \mathbf{c} a vector. The r th Krylov subspace, denoted by $\mathcal{K}_r(\mathbf{A}, \mathbf{c})$ is the vector space spanned by the vectors,
 $\mathbf{c}, \mathbf{Ac}, \dots, \mathbf{A}^{r-1}\mathbf{c}$.

Usually, but not necessarily, the vector \mathbf{c} is the right-hand side in a linear system $\mathbf{Ax} = \mathbf{b}$ so that $\mathbf{c} = \mathbf{b}$.

Suite dans la bibliographie...

A. Méthodes d'optimisation issues du théorème de Krylov

A compléter : gradients conjugués, GMRES, BiCG,
BiCGSTAB ... Encore moins urgent.

A. Schéma pour l'optimisation sur maillage uniforme (1/3)

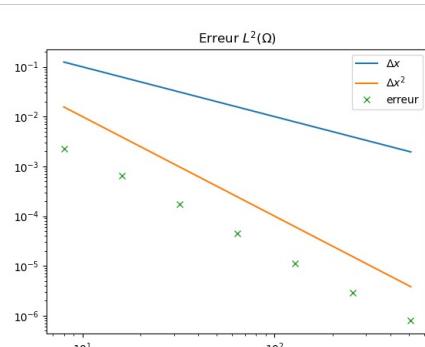
Schéma pour la diffusion non linéaire : sur maillage uniforme.

$$\partial_x(Q(u) \partial_x u)_i \sim \frac{\frac{1}{2}(Q(u_{i+1}) + Q(u_i))(u_{i+1} - u_i) - \frac{1}{2}(Q(u_i) + Q(u_{i-1}))(u_i - u_{i-1})}{\Delta x^2}$$

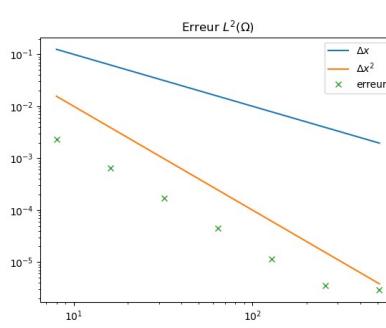
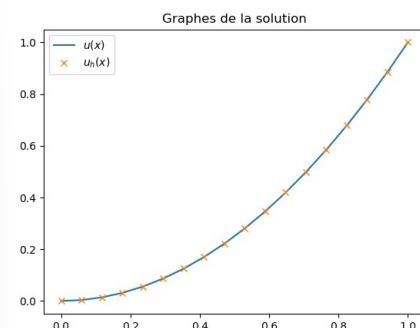
Validation du schéma et de l'implémentation avec `scipy.optimize.root(F, u0, method='krylov')`

Implémentation de la résolution du problème de diffusion non linéaire avec conditions de Dirichlet aux bords

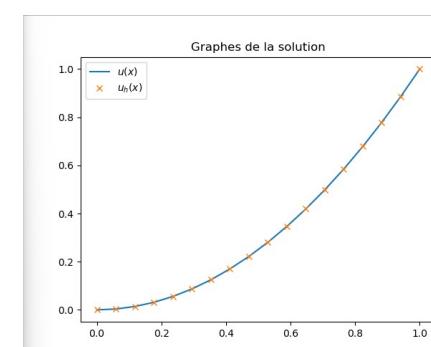
On observe une meilleure convergence pour l'initialisation à l'identité.



Initialisation à $u(0) = \text{Id}$



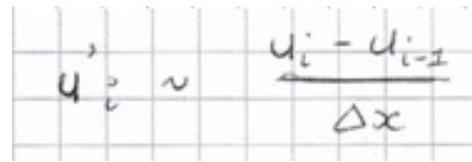
Initialisation à $u(0) = \text{Id}$



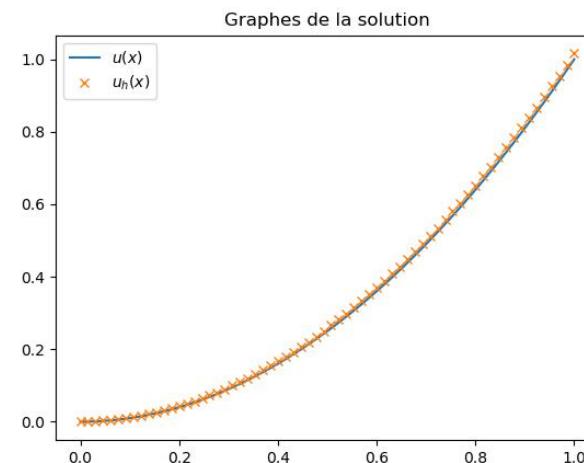
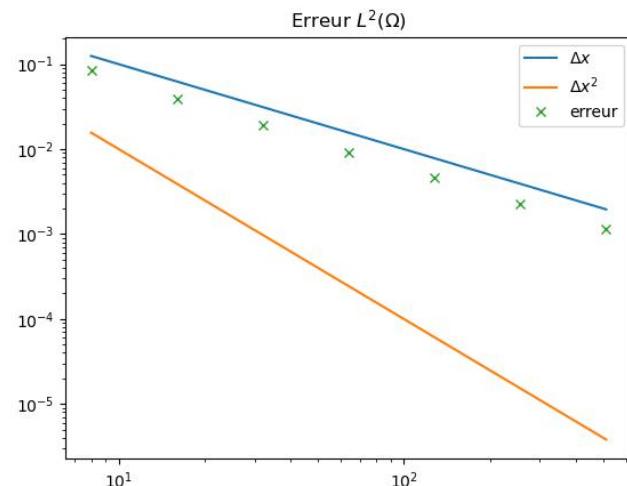
Initialisation à $u(0) = (x \rightarrow 0)$

A. Schéma pour l'optimisation sur maillage uniforme (2/3)

Implémentation du problème modèle avec conditions mixtes de Dirichlet / Neumann, on observe une convergence à l'ordre 1 pour le schéma décentré amont à deux points en $x = 1$.

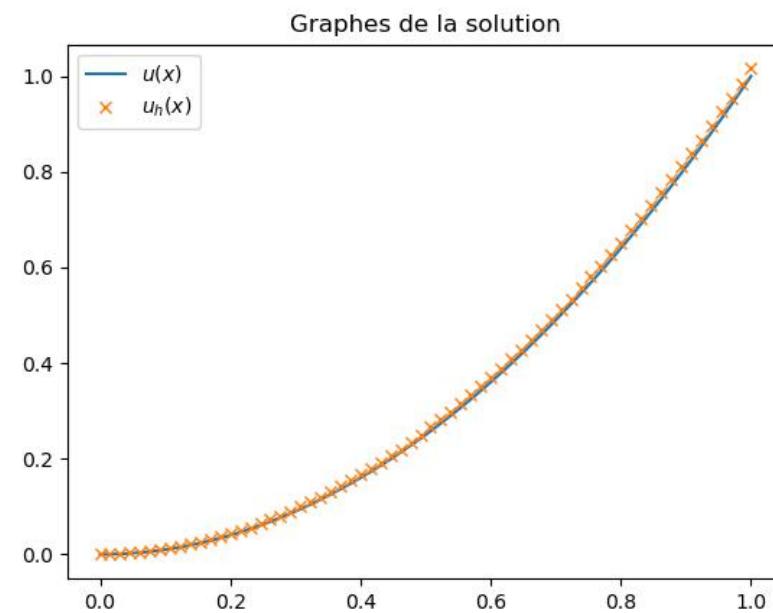
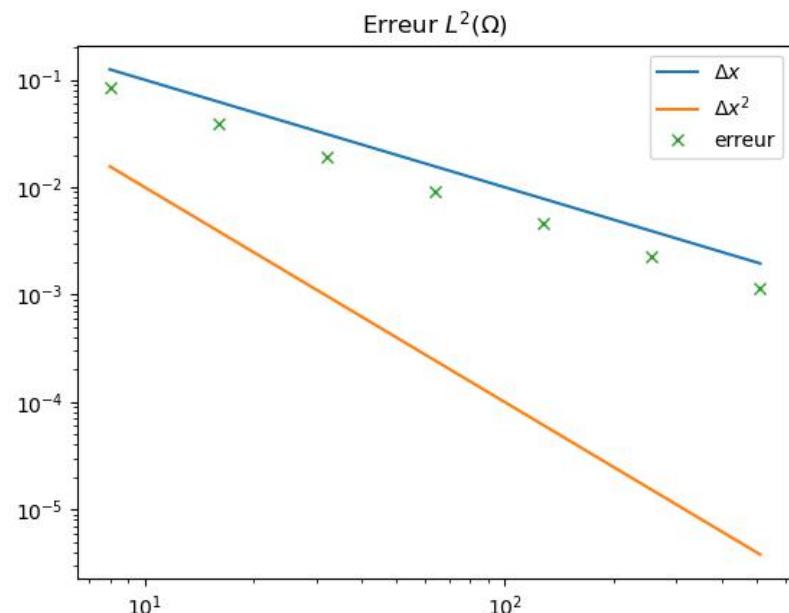


Initialisation à $u(0) = (x \rightarrow x)$ (Identité)



A. Schéma pour l'optimisation sur maillage uniforme (3/3)

Initialisation à $u(0) = (x \rightarrow 0)$ (Fonction nulle)



A. Justification et extension à des maillages non réguliers

Schéma pour la diffusion non linéaire sur maillage non uniforme

$$\partial_x(Q(u)\partial_x(u)) - f = 0 \Rightarrow \int_{x_{i-1/2}}^{x_{i+1/2}} \partial_x(Q(u)\partial_x(u)) - \int_{x_{i-1/2}}^{x_{i+1/2}} f = 0$$



Méthode de volumes finis

$$\left. \begin{aligned} x_{i+1/2} &\sim \frac{x_{i+1} + x_i}{2} \\ x_{i-1/2} &\sim \frac{x_i + x_{i-1}}{2} \end{aligned} \right\} \quad \boxed{\textcircled{2} \sim (x_{i+1/2} - x_{i-1/2}) f(x_i)}$$

$$\Rightarrow x_{i+1/2} - x_{i-1/2} \sim \frac{x_{i+1} - x_{i-1}}{2}$$

$$\textcircled{1}: \int_{x_{i-1/2}}^{x_{i+1/2}} \partial_x(Q(u)\partial_x(u)) \sim \left[Q(u) \partial_x u \right]_{x_{i-1/2}}^{x_{i+1/2}} \sim [Q(u(x_{i+1/2}))(\partial_x u)_{i+1/2} - Q(u(x_{i-1/2}))(\partial_x u)_{i-1/2}]$$

on approche Q par sa moyenne arithmétique :

$$Q_{i+1/2} \sim \frac{Q(u_{i+1}) + Q(u_i)}{2} ; \quad Q_{i-1/2} \sim \frac{Q(u_i) + Q(u_{i-1})}{2}$$

Interpolation polynomiale

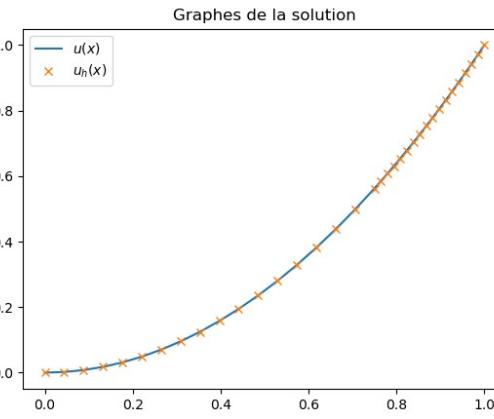
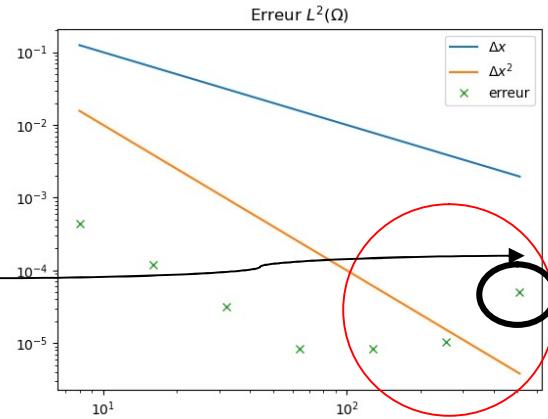
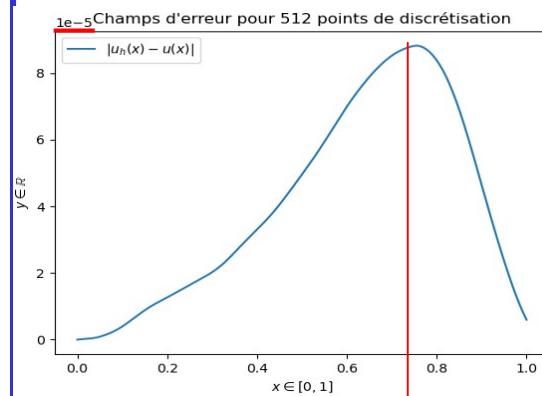
d'autre part :

$$; \quad u_{i+1/2} \sim \frac{u_{i+1} - u_i}{x_{i+1} - x_i} ; \quad u_{i-1/2} \sim \frac{u_i - u_{i-1}}{x_i - x_{i-1}}$$

Différences finies pour les flux

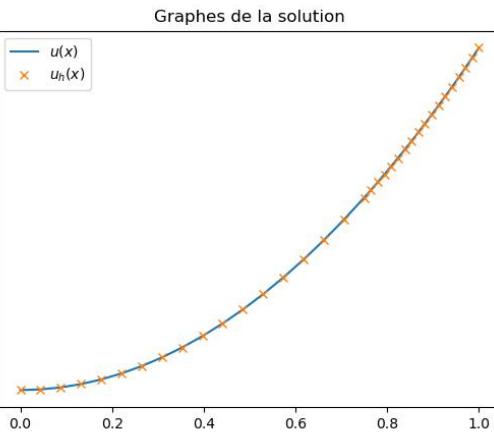
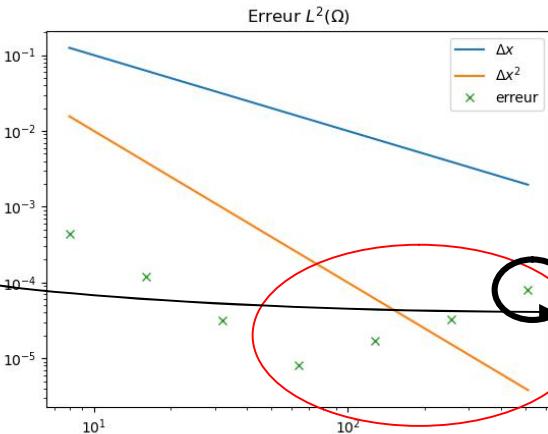
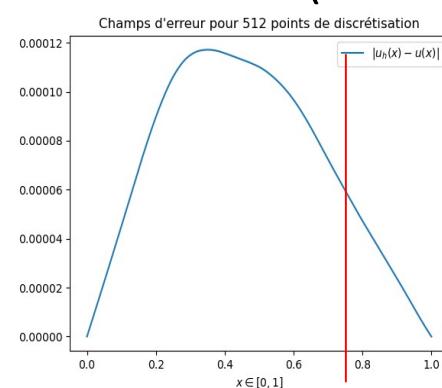
A. Convergence sur le problème modèle avec conditions de Dirichlet

Initialisation à $(x \rightarrow 0)$



~ 0.75 : abscisse de la frontière

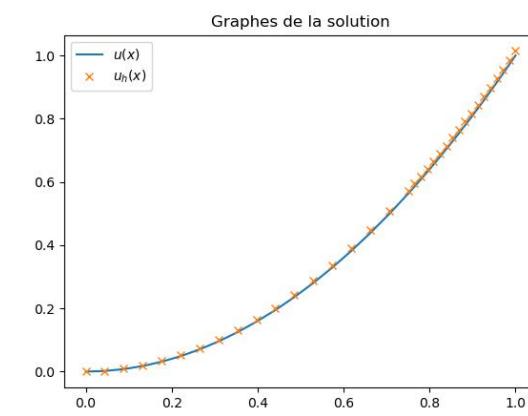
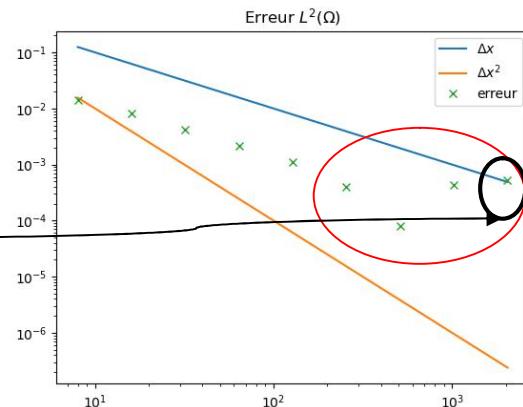
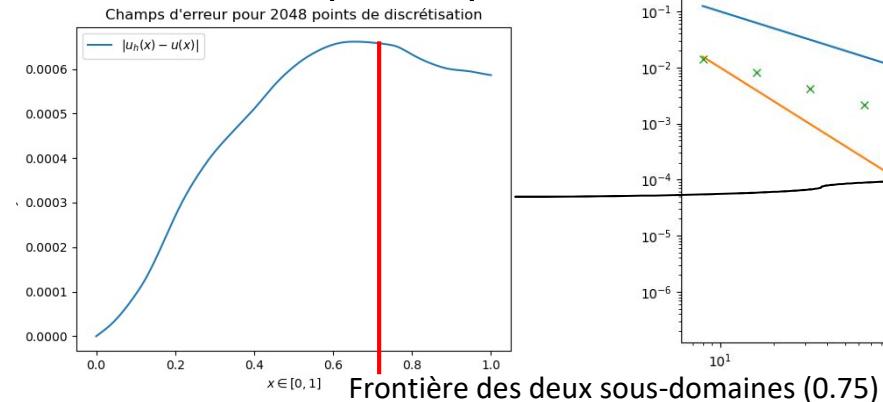
Initialisation à $(x \rightarrow x)$



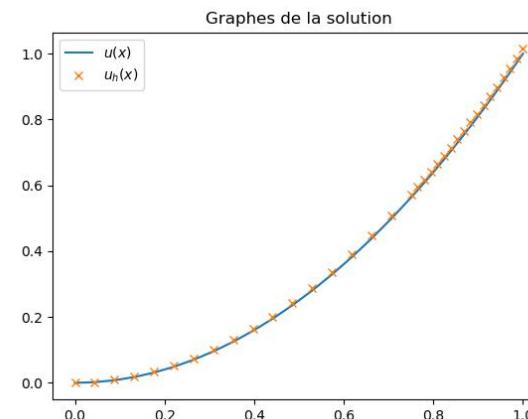
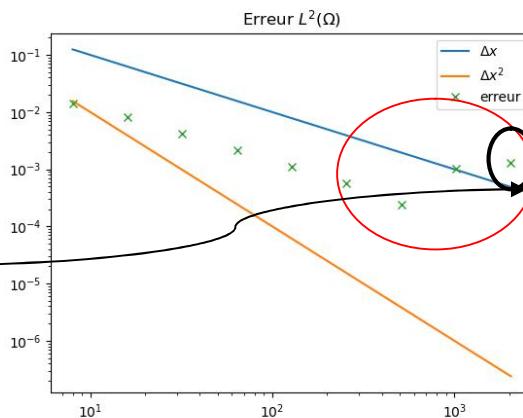
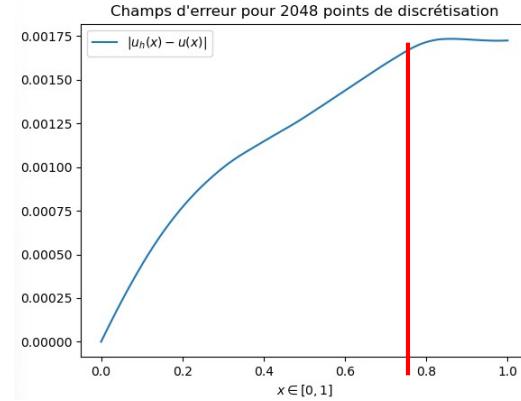
~ 0.75 : abscisse de la frontière

A. Convergence sur le problème modèle avec conditions mixtes

Initialisation à ($x \rightarrow 0$)



Initialisation à ($x \rightarrow x$)



A. Conclusion de la partie analyse numérique

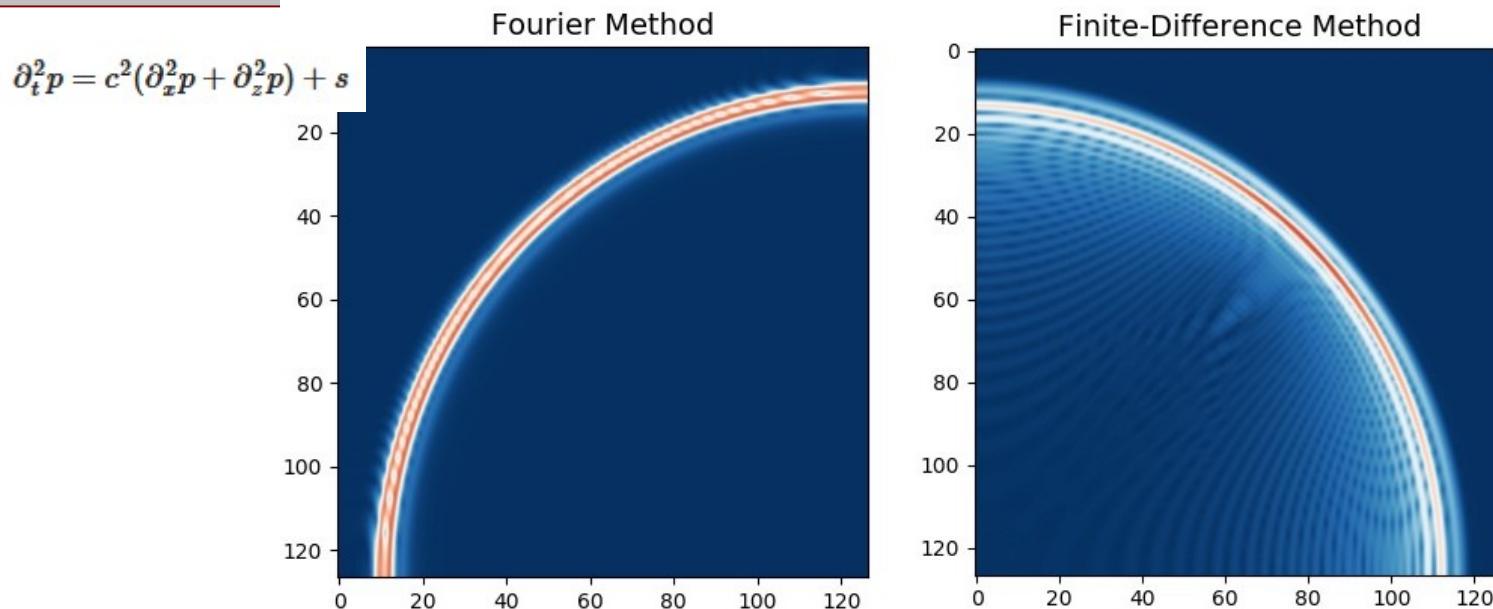
A terme :

Comparer les schéma issus des méthodes volumes finis, éléments finis et différences finies

Apprendre et maîtriser les méthodes numériques dans le formalisme *Lagrangien* : tout se passe sur le maillage de référence, très efficace en éléments finis

Reprendre les schéma pour les opérateurs différentiels axisymétriques !

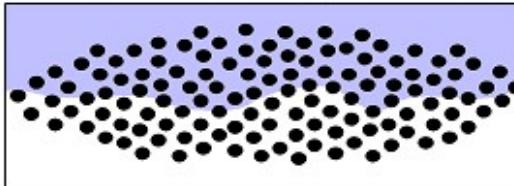
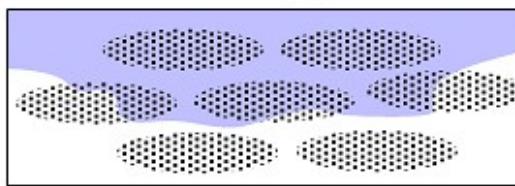
Attention à l'anisotropie des différences finies en contexte 2.D axisymétrique pour les phénomènes radiaux : quantifier l'erreur d'anisotropie ? **Illustration et démonstrateur pour l'équation d'onde** sur la page Github (dépôt de documentations et de code de Heiner Igel, Université de Munich) ; (.ipynb : IPython Notebook, se lit dans le navigateur comme une page HTML)



https://github.com/heinerigel/coursera/blob/master>Notebooks4Coursera/W5/W5_ps_fourier_acoustic_2d.ipynb

B. Physiques

B. Caractérisations des écoulements

Echelle	Microscopique	Mésoscopique	Macroscopique
Schéma			
Caractéristiques principales	Milieu triphasique : solide (fibres), liquide (résine), gaz (air)	2 milieux : mèche et espace intermèche 4 zones : liquide, air, mèche saturée, mèche insaturée	Milieu poreux homogénéisé
Grandeurs caractéristiques	Diamètre d'une fibre : ~1-10 um Espace intra-mèche (= inter-fibre) : ~1-10 um	Largeur mèche : ~ 1-10 mm Epaisseur mèche : ~ 0,1 mm Espace inter mèches : ?	Tenseur de perméabilité
Domaine d'étude	Quelques dizaines de fibres	Quelques mèches	Quelques cm à quelques mètres
Nombre caractéristique			
Reynolds			
Conséquences sur l'écoulement	Effets visqueux >> effets inertIELS		
Modèles d'écoulement / équations utilisées	Stokes	Stokes + Darcy (selon zone) ou : Brinkman (approche monolithique)	Milieu poreux : Darcy
Prise en compte capillarité	Terme de force à l'interface résine/air, dépend de la tension de surface, de la courbure, ... Possible prise en compte d'un angle statique à l'interface solide/liquide/gaz	Dans les mèches : pression capillaire (saut de pression à l'interface saturé/insaturé ?) A priori, canaux inter mèches trop grands pour qu'il y ait de la capillarité	Pression capillaire (saut de pression à l'interface saturé/insaturé ?)
Travaux dans biblio			

C. Outils et méthodes informatique

- Découverte de la suite Pack Office
- Manipulations en FreeFEM++
- Manipulations en Scilab et en Excel (production de rapports et modélisation)
- Introduction de Python et d'outils connexes dans l'écosystème de développement TENSYL
 - Anaconda (Miniconda)
 - V.S. Code (I.D.E)
 - *Matplotlib, Numpy, Scipy* (librairies de visualisation, d'analyse numérique et d'algorithmes en Python 3.x)
- Réflexion sur un modèle objet pour la simulation de procédés d'enroulement filamentaire de mèches humides
- Rédaction de scripts pour la validation de l'implémentation en Python 3.x des méthodes numériques impliquées dans le procédé.

C. Fonction *scipy.optimize.newton()*

scipy.optimize.newton

```
scipy.optimize.newton(func, x0, fprime=None, args=(), tol=1.48e-08, maxiter=50,
fprime2=None, x1=None, rtol=0.0, full_output=False, disp=True)
```

[\[source\]](#)

Find a zero of a real or complex function using the Newton-Raphson (or secant or Halley's) method.

Find a zero of the scalar-valued function *func* given a nearby scalar starting point *x0*. The Newton-Raphson method is used if the derivative *fprime* of *func* is provided, otherwise the secant method is used. If the second order derivative *fprime2* of *func* is also provided, then Halley's method is used.

If *x0* is a sequence with more than one item, *newton* returns an array: the zeros of the function from each (scalar) starting point in *x0*. In this case, *func* must be vectorized to return a sequence or array of the same shape as its first argument. If *fprime* (*fprime2*) is given, then its return must also have the same shape: each element is the first (second) derivative of *func* with respect to its only variable evaluated at each element of its first argument.

newton is for finding roots of a scalar-valued functions of a single variable. For problems involving several variables, see [root](#).

C. Fonction `scipy.optimize.root()` (1/2)

`scipy.optimize.root`

```
scipy.optimize.root(fun, x0, args=(), method='hybr', jac=None, tol=None, callback=None,  
options=None)
```

[\[source\]](#)

Find a root of a vector function.

Parameters: `fun` : *callable*

A vector function to find a root of.

`x0` : *ndarray*

Initial guess.

`args` : *tuple, optional*

Extra arguments passed to the objective function and its Jacobian.

`method` : *str, optional*

Type of solver. Should be one of

- 'hybr' ([see here](#))
- 'lm' ([see here](#))
- 'broyden1' ([see here](#))
- 'broyden2' ([see here](#))
- 'anderson' ([see here](#))
- 'linarmixing' ([see here](#))
- 'diagbroyden' ([see here](#))
- 'excitingmixing' ([see here](#))
- 'krylov' ([see here](#))
- 'df-sane' ([see here](#))

C. Fonction `scipy.optimize.root()` (2/2)

Notes

This section describes the available solvers that can be selected by the 'method' parameter. The default method is *hybr*.

Method *hybr* uses a modification of the Powell hybrid method as implemented in MINPACK [1].

Method *lm* solves the system of nonlinear equations in a least squares sense using a modification of the Levenberg-Marquardt algorithm as implemented in MINPACK [1].

Method *df-sane* is a derivative-free spectral method. [3]

Methods *broyden1*, *broyden2*, *anderson*, *linarmixing*, *diagbroyden*, *excitingmixing*, *krylov* are inexact Newton methods, with backtracking or full line searches [2]. Each method corresponds to a particular Jacobian approximations.

- Method *broyden1* uses Broyden's first Jacobian approximation, it is known as Broyden's good method.
- Method *broyden2* uses Broyden's second Jacobian approximation, it is known as Broyden's bad method.
- Method *anderson* uses (extended) Anderson mixing.
- Method *Krylov* uses Krylov approximation for inverse Jacobian. It is suitable for large-scale problem.
- Method *diagbroyden* uses diagonal Broyden Jacobian approximation.
- Method *linarmixing* uses a scalar Jacobian approximation.
- Method *excitingmixing* uses a tuned diagonal Jacobian approximation.

C. Fonction *scipy.optimize.root()*

Affichage des paramètres disponibles pour le solver *root()* avec la méthode *Krylov*

```
>>> import scipy.optimize
>>> scipy.optimize.show_options(solver="root", method="krylov")
Options
-----
nit : int, optional
    Number of iterations to make. If omitted (default), make as many
    as required to meet tolerances.
disp : bool, optional
    Print status to stdout on every iteration.
maxiter : int, optional
    Maximum number of iterations to make. If more are needed to
    meet convergence, `NoConvergence` is raised.
ftol : float, optional
    Relative tolerance for the residual. If omitted, not used.
fatol : float, optional
    Absolute tolerance (in max-norm) for the residual.
    If omitted, default is 6e-6.
xtol : float, optional
    Relative minimum step size. If omitted, not used.
xatol : float, optional
    Absolute minimum step size, as determined from the Jacobian
    approximation. If the step size is smaller than this, optimization
    is terminated as successful. If omitted, not used.
tol_norm : function(vector) -> scalar, optional
    Norm to use in convergence check. Default is the maximum norm.
line_search : {None, 'armijo' (default), 'wolfe'}, optional
    Which type of a line search to use to determine the step size in
    the direction given by the Jacobian approximation. Defaults to
    'armijo'.
jac_options : dict, optional
    Options for the respective Jacobian approximation.
```

```
jac_options : dict, optional
    Options for the respective Jacobian approximation.

rdiff : float, optional
    Relative step size to use in numerical differentiation.
method : {'lgmres', 'gmres', 'bicgstab', 'cgs', 'minres'} or function
    Krylov method to use to approximate the Jacobian.
    Can be a string, or a function implementing the same
    interface as the iterative solvers in
    `scipy.sparse.linalg`.

    The default is `scipy.sparse.linalg.lgmres`.
inner_M : LinearOperator or InverseJacobian
    Preconditioner for the inner Krylov iteration.
    Note that you can use also inverse Jacobians as (adaptive)
    preconditioners. For example,
    >>> jac = BroydenFirst()
    >>> kjac = KrylovJacobian(inner_M=jac.inverse).

    If the preconditioner has a method named 'update', it will
    be called as ``update(x, f)`` after each nonlinear step,
    with ``x`` giving the current point, and ``f`` the current
    function value.
inner_tol, inner_maxiter, ...
    Parameters to pass on to the "inner" Krylov solver.
    See `scipy.sparse.linalg.gmres` for details.
outer_k : int, optional
    Size of the subspace kept across LGMRES nonlinear
    iterations.

    See `scipy.sparse.linalg.lgmres` for details.
>>>
```

« Page 1 »

« Page 2 »

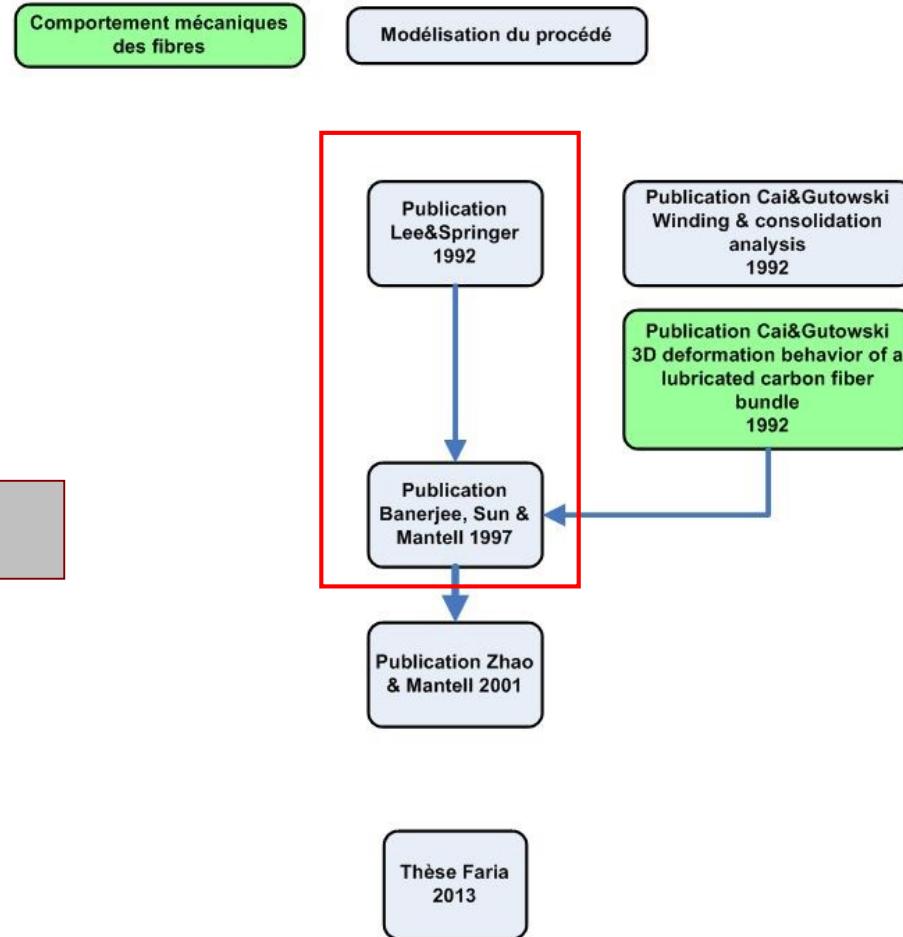
D. Modélisation à l'échelle macroscopique : contexte

Banerjee: dernière publication connue sur procédé enroulement.

Ajout /Springer-Lee:

- Modèle comportement méca mèche.
- Mélange résine.

Arnaud Gillet : ne pas utiliser Springer et Lee mais Farina (problème formel bien posé)



D. Modélisation à l'échelle macroscopique

- Procédé d'enroulement filamentaire de mèches humides.

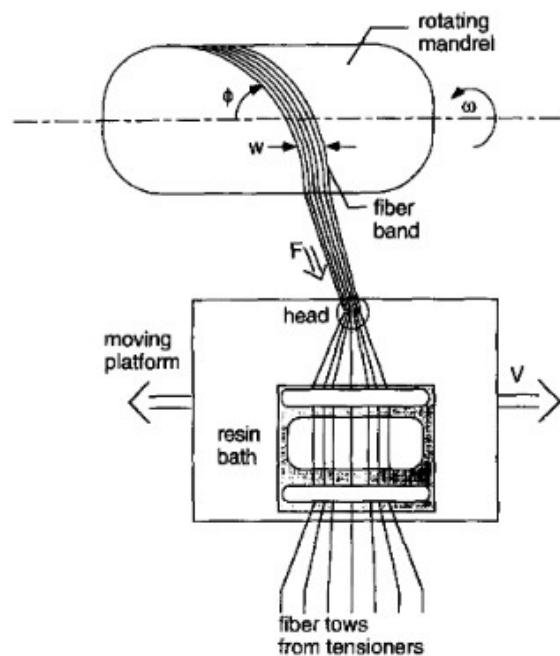


Figure 1 Schematic of the wet filament winding process

Extrait de Banerjee et al. [1]

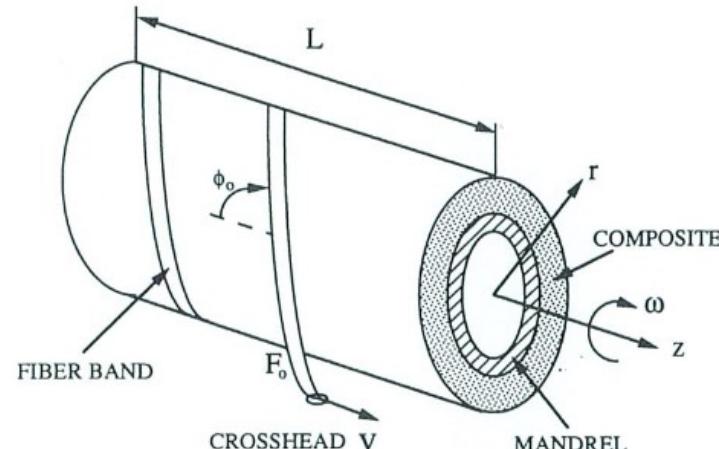


Figure 1. Description of the problem.

Extrait de Springer et Lee [2]

D. Objectifs de la modélisation macroscopique

Objectif : décrire la distribution des grandeurs :

- Température dans le cylindre et dans le mandrin
- Degré de cuisson dans le cylindre
- Viscosité dans le cylindre
- Position des fibres
- Tension des fibres
- Contraintes et déformations dans le cylindre et dans le mandrin
- Porosité dans le cylindre (ou le taux de fibres)

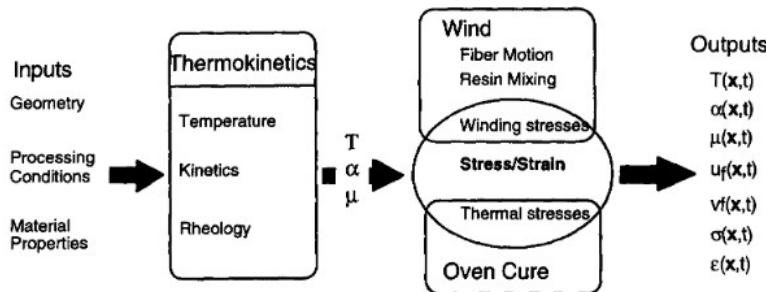


Figure 5 Schematic showing the inputs and outputs of the filament winding model and the relationships among submodels

D. L'apport de Banerjee & al.

L'idée est d'articuler des modèles physiques élémentaires pour obtenir un modèle d'écrasement d'un matériau élastique saturé en résine.

Points clefs :

- Considération de l'avancement de la réaction de durcissement et de ses conséquences sur les grandeurs physiques
- Prise en compte du mélange de résine et homogénéisation de ses propriétés dans un pli donné
- Considération de comportement mécanique *complexe* pour décrire l'équilibre mécanique des solides

D. Modélisation à l'échelle macroscopique

- Une mèche est passée dans un bain de résine avant d'être appliquée sur l'empilement grâce à une tête de dépose (c.f figures 1)
- La mèche est posée avec un certain angle phi, dépendant de la vitesse axiale de la tête et de la vitesse angulaire du mandrin
- Une tension F est appliquée à la mèche, laquelle implique une pression P sur l'empilement
- La pression P appuie sur l'empilement et le déforme
- Cette déformation induit une modification du taux de fibres
- Cette variation donne un écoulement de la résine vers la surface de l'empilement

D. Classification des enroulements par les durées d'écoulement de la résine dans un pli (1/2)

Durée d'écoulement de la résine à travers une mèche :

$$t_f = t^* \frac{4\kappa_z \mu h_0^2}{A_s r_d^2}$$

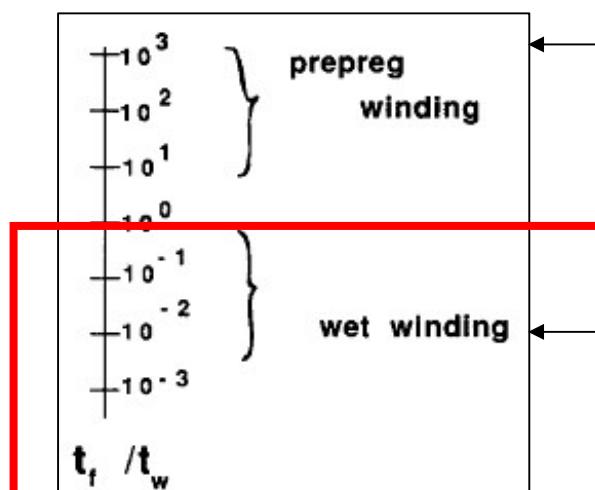
Constante de Kozeny Viscosité de la résine
Temps adimensionnel, chute significative de la pression Rayon d'une fibre
 $t^* = 0,5$ « Fiber bed spring constant »
Épaisseur de la mèche

(Cai et al., 1992)

D. Classification des enroulements par les durées d'écoulement de la résine dans un pli (2/2)

Mèche		AS4 3k	T700 12K
Epaisseur d'une couche	m	1.52E-04	1.20E-04
Viscosité	Pa.s	10	6.20E-01 620 mPa.s à 25°C selon FT
Nombre de mèches		65	
Temps d'enroulement	s	90	120 Durée 1 couche / 2
Rayon de fibre	m	3.50E-06	3.50E-06
Constante de Kozeny		0.2	0.2
Constante "Fiber spring"	Pa	4.14E+02	4.14E+02 Déterminée expérimentalement
Constante de temps d'écoulement	sec	36.7	1.4
Temps adimensionnel		0.204	0.006

REF: Données tvx 2016



Très peu d'écoulement.
Matériau : approximation linéaire élastique
Rigidité des fibres + pression de résine

Temps d'écoulement très faible devant temps d'enroulement.
L'état de déformation final est principalement lié aux fibres.
Pas de pression de résine ?

D. Le problème du placement des fibres

- Deux sources de mouvement des fibres :
 - La déformation des couches déjà en place lors de la pose de la couche courante (u_m , u_c)
 - Le mouvement de la couche courante dans la résine présente au sommet de l'empilement (u_f)

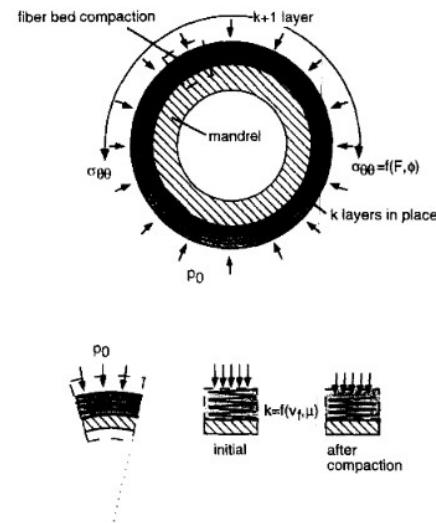


Figure 2 Compaction during filament winding. When the $k + 1$ layer is wound, the layers beneath are subjected to a compressive load

Extrait de Banerjee et al [1]

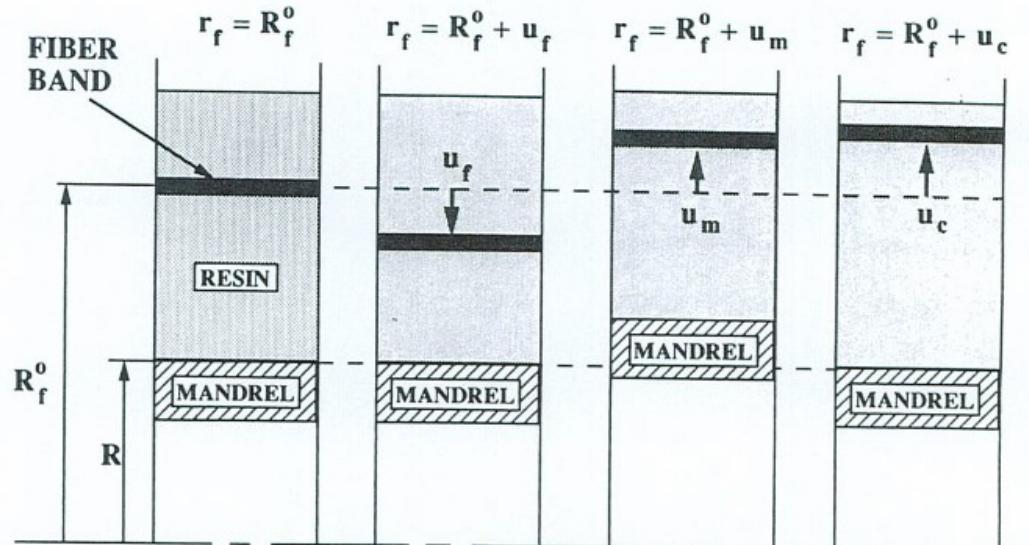


Figure 6. Illustration of the changes in fiber position relative to cylinder's axis from its position at the time of winding, due to the winding tension and the deformation of the mandrel and composite.

Extrait de Springer et Lee [2]

D. Déformation de l'empilement

- Déformations élastiques :
 - La tension de la mèche courante induit une pression sur l'empilement qui cause une déformation.
 - La réponse de l'empilement se calcule couche par couche: le comportement du pli dépend de l'état de la résine : durcie ou non, critère de viscosité
- Déformations chimiques :
 - Le pli est caractérisé par un facteur de retrait chimique (changement de volume en fonction de l'avancement du durcissement)
- Déformations thermiques :
 - Le pli est caractérisé par un facteur de dilatation thermique, et la température évolue du fait de la réaction chimique de la résine

D. Placement de la couche courante

Itération sur le temps d'enroulement

- La vitesse de la résine au travers de la couche courante est solution d'un problème de Darcy
- La vitesse de la couche courante dans le pli précédent est connue (changement de réf.)
- Le taux de fibres du pli précédent croît
- Le mélange est modifié
- La réaction chimique avance
- La température évolue

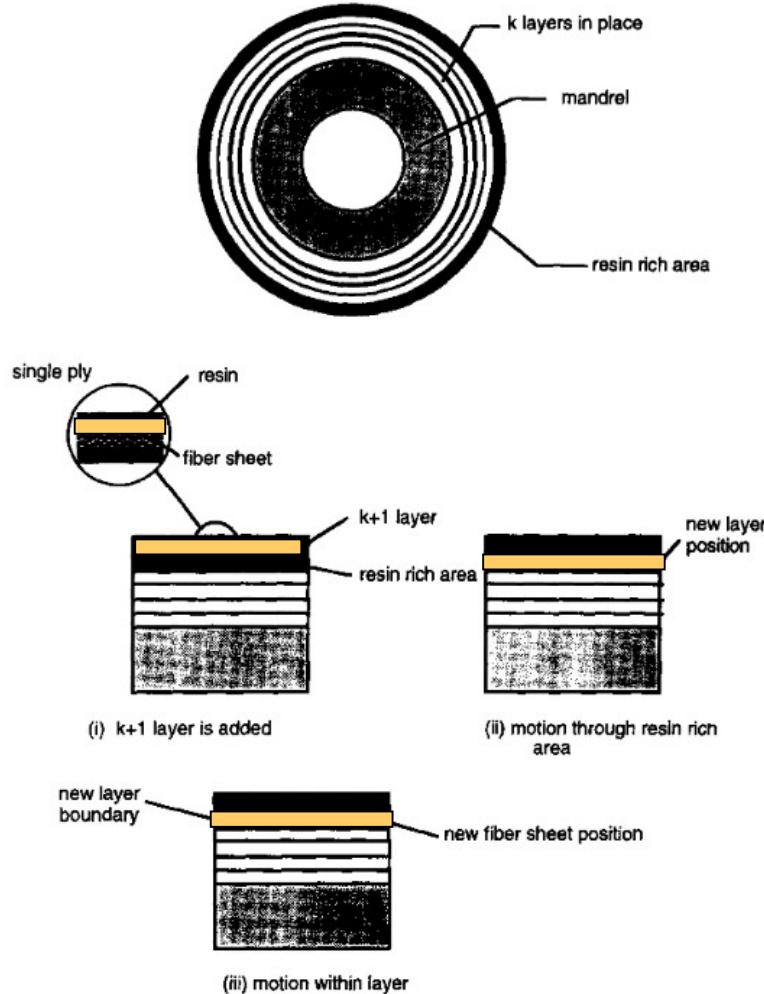
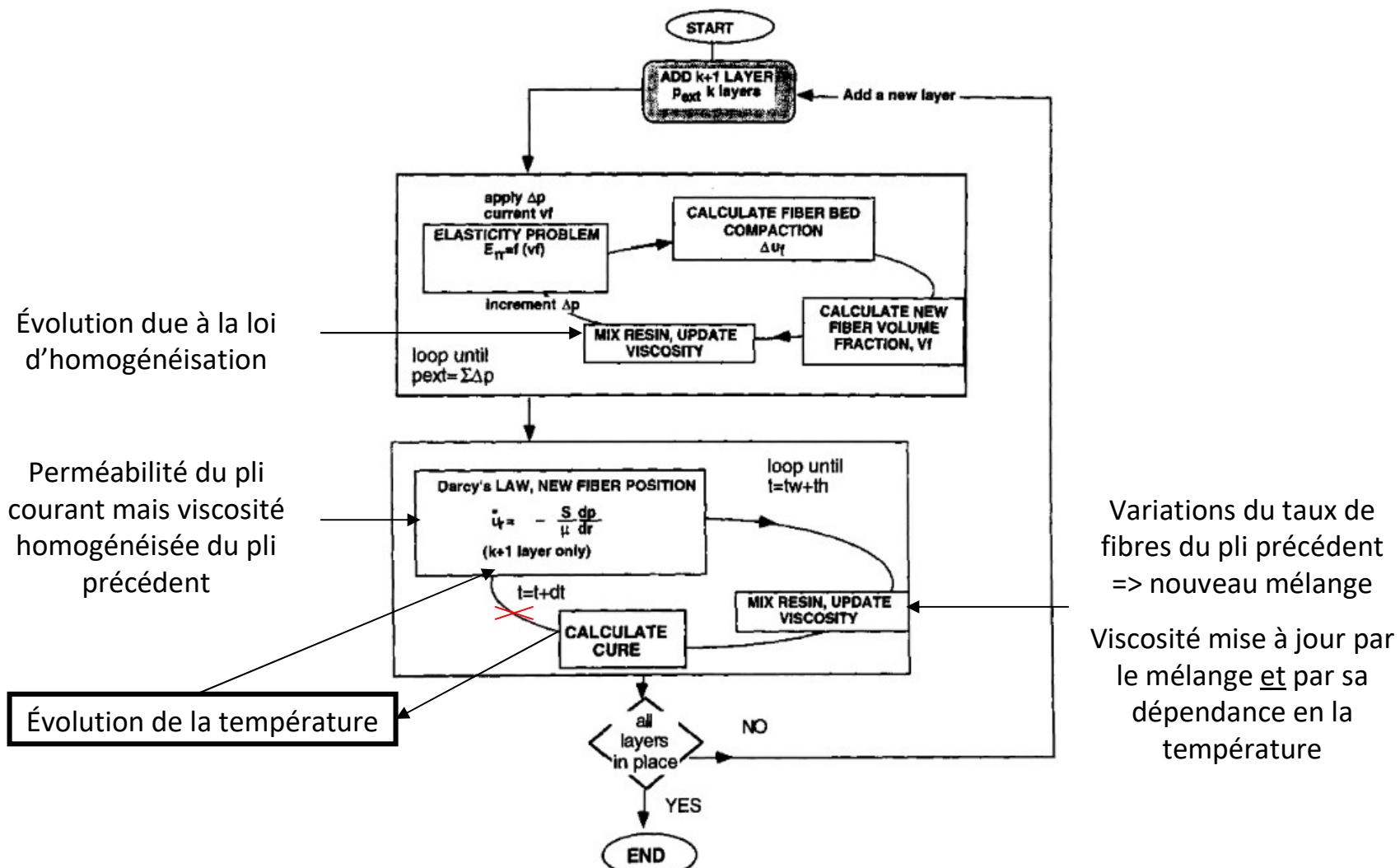


Figure 4 Fiber motion when the $k + 1$ layer is added: (i) Compaction from previous layers wound creates a resin rich region; (ii) the $k + 1$ layer moves through the resin rich region; and (iii) the layer then compacts

D. Algorithme de Banerjee et al [1]



E. Modélisation à l'échelle mésoscopique

Non traitée sur la période.

F. Modélisation à l'échelle microscopique

Non traitée sur la période.

G. Actions à venir

- Achever d'explorer le comportement de `scipy.optimize.root`, confronter à un algorithme de Newton implémenté à la main
 - Observer les champs d'erreurs pour la mauvaise convergence sur le maillage non uniforme
 - Valider l'implémentation du cas non uniforme sur maillage uniforme et maillage complètement irrégulier (lois polynomiales)
 - Écrire le problème tel que formalisé par Farina & al
 - Proposer un schéma numérique adapté et un algorithme de résolution
- PROCHAINE REUNION : VENDREDI 14/10, 10h : 12h

H. Bibliographie de la présentation

- [1] Banerjee, Model and experimental study of fiber motion in wet filament winding, J.C.M, 1998
- [2] Springer, Lee, Filament winding cylinders : I process model, J.C.M, 1990
- [3] Caï, Gutowski, Winding and consolidation analysis for cylindrical composite structures, 1992
- [4] P. Le Tallec, Numerical methods for non linear three-dimensional elasticity, Handbook of Numerical Analysis III (P.G Ciarlet, J. L. Lions), p. 465
- [5] Alex Townsend, Krylov subspaces, (<http://pi.math.cornell.edu/~web6140/TopTenAlgorithms/KrylovSubspace.html>)
- [6] Heiger Inel,