framatome

Parallélisation de calculs en fatigue

Auteur : Antoine Kempf

Superviseur entreprise : Benoît Serre

Superviseur académique : Mohab Safey El Din

Septembre 2024

Table des matières

1 Introduction **2**
- 1.1 Présentation de l'entreprise . . . 2
- 1.2 Le Centre Calcul Bourgogne . . . 2

2 État de l'art **4**
- 2.1 L'analyse de fatigue . . . 4
  - 2.1.1 Présentation . . . 4
  - 2.1.2 Notions de physique . . . 4
- 2.2 Le logiciel `ALLIANCE` . . . 7
  - 2.2.1 Présentation . . . 7
  - 2.2.2 Fonctionnement . . . 7
- 2.3 Contexte du stage . . . 9

3 Optimisation **10**
- 3.1 Modèle d'apprentissage . . . 10
  - 3.1.1 Contexte . . . 10
  - 3.1.2 Modèle . . . 10
  - 3.1.3 Résultats . . . 11
- 3.2 Approche théorique . . . 12
  - 3.2.1 Idée générale . . . 12
  - 3.2.2 Formulation . . . 14
  - 3.2.3 Preuve . . . 15
- 3.3 Approche structurelle . . . 16
  - 3.3.1 Parallélisation . . . 16
  - 3.3.2 Implémentation . . . 16

4 Résultats **18**
- 4.1 Approche théorique . . . 18
- 4.2 Approche structurelle . . . 20

5 Conclusion **22**

Chapitre 1

Introduction

1.1 Présentation de l'entreprise

Framatome est une entreprise française spécialisée dans la conception, la fabrication et la maintenance de composants pour les centrales nucléaires. Fondée en 1958 sous le nom de "Franco-Américaine de Constructions Atomiques", l'entreprise a joué un rôle central dans le développement de l'industrie nucléaire en France et dans le monde. Framatome est principalement impliqué dans trois domaines d'activités :

- Ingénierie et Services : Framatome fournit des services de conception d'ingénierie, de gestion de projets et de maintenance pour les réacteurs nucléaires. Ces services couvrent toutes les phases du cycle de vue d'une centrale, de la conception initiale à la déconstruction.
- Fabrication de Combustible : L'entreprise est l'un des principaux fabricants mondiaux de combustible nucléaire, fournissant divers assemblages de combustibles pour divers types de réacteurs à travers le monde.
- Équipements Nucléaires : Framatome conçoit et fabrique des équipements critiques pour les réacteurs nucléaires, tels que les générateurs de vapeur, les cuves de réacteur, les pressuriseurs, ainsi que les systèmes de contrôle-commande pour la sûreté nucléaire.

Framatome est une filiale du groupe EDF (Électricité de France), qui détient une part majoritaire de son capital. En tant que tel, Framatome bénéficie d'une position stratégique dans l'industrie nucléaire mondiale. L'entreprise a des installations et des bureaux dans plus de 20 pays et emploie environ 14 000 personnes.

L'innovation est au cœur des activités de Framatome. L'entreprise investit considérablement dans la recherche et le développement pour améliorer l'efficacité, la sécurité et la performance des technologies nucléaires. En parallèle, Framatome s'engage à respecter les normes environnementales les plus strictes et à contribuer à la transition énergétique vers des sources d'énergie décarbonées.

1.2 Le Centre Calcul Bourgogne

Le site de Saint-Marcel est l'une des principales installations industrielles de Framatome, située dans la région Bourgogne-Franche-Comté, près de Chalon-sur-Saône. Créé en 1975 en réponse à la montée en puissance du programme nucléaire français, le site est spécialisé dans la fabrication de gros composants mécaniques pour les réacteurs nucléaires, notamment les générateurs de vapeur, les cuves de réacteurs, les pressuriseurs et les tuyauteries primaires. Il couvre une surface de 40 hectares et dispose d'équipements de pointe pour la production, l'assemblage et le contrôle de qualité des composants pour 106 réacteurs nucléaires dans 11 pays. Saint-Marcel accueille également sur son site le Centre Technique de Framatome qui mène des activités de recherche et développement, ainsi que le très récent Centre Calcul Bourgogne (CCB), un centre d'expertise et de formation dédié au calcul mécanique.

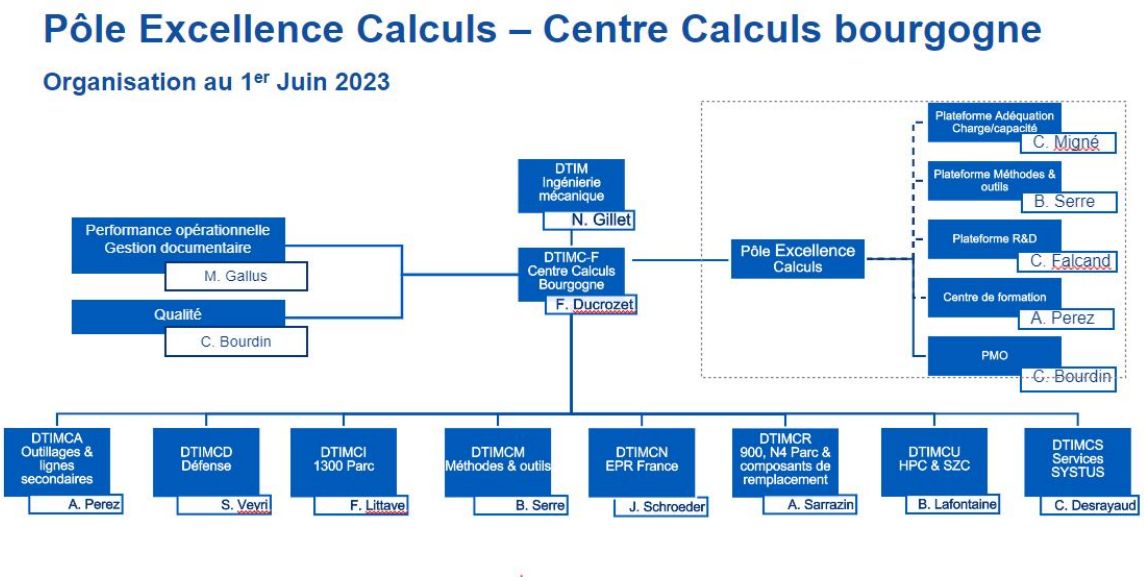

FIGURE 1.1 – Organigramme du Centre Calcul Bourgogne

Le département se divise en 8 sections travaillant chacune sur des projets différents et complémentaires. Par exemple, la section DTIMCU s'occupe des dossiers Hinkley Point et Sizewell. La section dans laquelle se déroule le stage est DTIMCM, Méthodes et Outils. Elle est dirigée par Benoît SERRE et ses objectifs sont les suivants :

— Réaliser les études d'analyse à la fatigue ou de rupture brutale.
— Développer les outils nécessaires à la réalisation des études en fatigue ou de rupture brutale (`ALLIANCE`, `DEFIS`, `DEFIX`, `CALORI`).
— Industrialiser l'utilisation de `code_aster` et de `SalomeMeca`.

Chapitre 2

État de l'art

2.1 L'analyse de fatigue

2.1.1 Présentation

Chez Framatome, et plus généralement dans le domaine du nucléaire, l'analyse de fatigue joue un rôle essentiel pour garantir la sécurité et la durabilité des composants critiques, tels que les cuves de réacteur, les tuyauteries et les systèmes de refroidissement. Ces structures sont soumises à des variations de température, de pression et d'efforts mécaniques qui induisent des cycles de contraintes répétés. Avec le temps, ces cycles peuvent provoquer des fissures microscopiques qui, si elles ne sont pas détectées et contrôlées, peuvent conduire à des lourdes défaillances. Dans le nucléaire, l'analyse de fatigue intègre des critères et des normes strictement encadrés pour assurer la fiabilité des installations. L'un des référentiels les plus utilisés dans ce domaine est le code RCC-M (Règles de Conception et de Construction des Matériels Mécaniques des îlots nucléaires). Ce code fournit des lignes directrices précises pour l'évaluation de la fatigue, notamment les critères de conception, les méthodes d'analyse et les facteurs de sécurité à appliquer. Comme dit précédemment, différents types d'efforts agissent sur les composants. La combinaison de ces efforts est donc cruciale pour déterminer la résistance réelle du matériau.

2.1.2 Notions de physique

Tenseur de contrainte

Le tenseur de contrainte est un outil mathématique utilisé en mécanique des milieux continus pour représenter l'état de contrainte en un point donné d'un matériau. Il s'agit d'une matrice 3×3 qui décrit les composantes des efforts internes (contraintes) appliqués sur une infinitésimale portion du matériau dans les trois directions de l'espace. Le tenseur de contrainte σ est exprimé sous la forme :

$$\sigma = \begin{pmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{pmatrix}$$

— Les termes $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$ sont les contraintes normales, qui représentent les forces agissant perpendiculairement aux faces élémentaires dans les directions x, y, z.
— Les termes $\tau_{xy}, \tau_{yz}, \tau_{xz}$ sont les contraintes de cisaillement, qui représentent les forces agissant parallèlement aux faces élémentaires.

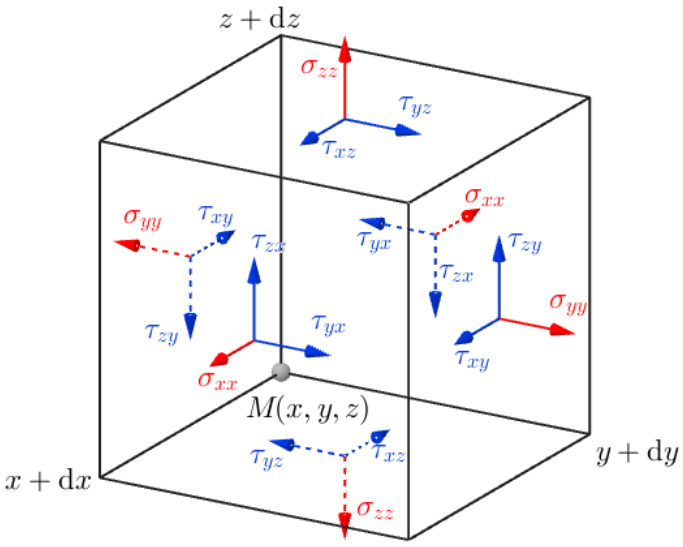

FIGURE 2.1 – Schéma représentant les composantes du tenseur des contraintes

Le tenseur est symétrique ($\tau_{xy} = \tau_{yx}$, etc.), ce qui réduit le nombre de termes indépendants à six. Le tenseur de contrainte permet de décrire l'état de contrainte complet en un point et est utilisé pour analyser la réponse de la structure sous diverses charges.

Critères de plasticité

Les critères de plasticité sont des outils utilisés en mécanique des matériaux pour déterminer à quel moment un matériau va commencer à se déformer de manière irréversible sous l'effet de contraintes appliquées. Ces critères permettent de prédire si un matériau reste dans son domaine élastique ou s'il entre dans une phase de déformation plastique. Parmi les critères les plus couramment utilisés, on retrouve les critères de Tresca et von Mises.

— Critère de Tresca : Le critère de Tresca, aussi appelé critère du cisaillement maximal, repose sur l'idée que la déformation plastique d'un matériau commence lorsque la contrainte de cisaillement maximale dans le matériau atteint une certaine valeur critique. Sa formule mathématique est la suivante :

$$\sigma_T = max(|\sigma_1 - \sigma_2|, |\sigma_2 - \sigma_3|, |\sigma_1 - \sigma_3|)$$

où $\sigma_1, \sigma_2, \sigma_3$ sont les contraintes principales (elles correspondent aux valeurs propres du tenseur de contrainte). Selon ce critère, la plasticité se produit lorsque la contrainte de cisaillement maximale σ_T égale la limite de cisaillement du matériau.

— Critère de von Mises : Le critère de von Mises, aussi appelé critère de l'énergie de distorsion, postule que la déformation plastique se produit lorsque l'énergie de distorsion due aux contraintes atteint une valeur critique. Ce critère est souvent préféré pour les matériaux ductiles, car il offre une meilleure prédiction de la déformation plastique dans des situations complexes de contrainte. Il est donné par l'expression :

$$\begin{aligned}\sigma_{VM} &= \sqrt{\frac{1}{2}[(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_1 - \sigma_3)^2]} \\ &= \sqrt{\frac{1}{2}[(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{xx} - \sigma_{zz})^2 + 6((\tau_{xy})^2 + (\tau_{yz})^2 + (\tau_{xz})^2)]}\end{aligned}$$

où $\sigma_1, \sigma_2, \sigma_3$ sont les contraintes principales. La condition de plasticité est atteinte lorsque la contrainte équivalente de von Mises σ_{VM} dépasse la limite d'élasticité du matériau.

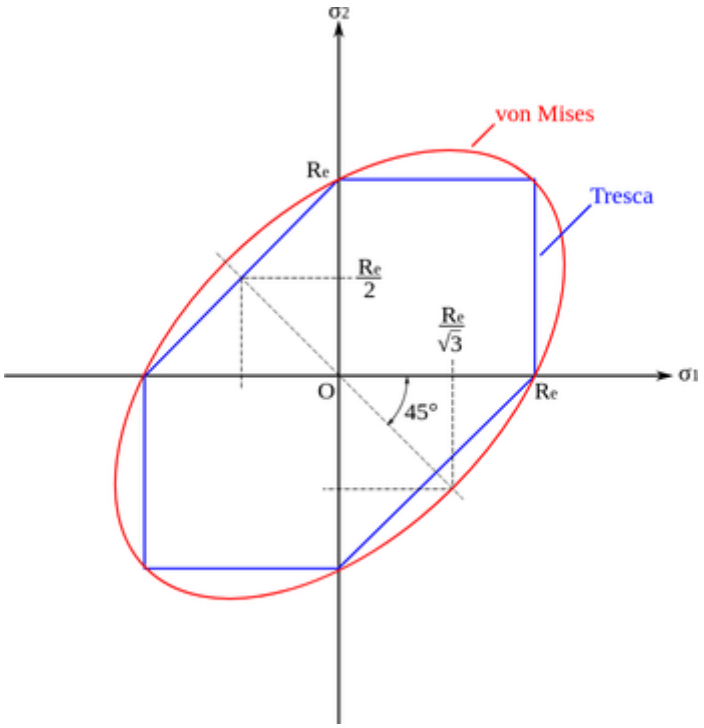

FIGURE 2.2 – Représentation des critères de Tresca et von Mises

2.2 Le logiciel ALLIANCE

2.2.1 Présentation

Le programme ALLIANCE est un programme développé par la section DTIMCM du CCB permettant de réaliser des analyses selon les codes RCC-M et ASME (American Society of Mechanical Engineers). Il permet en l'occurrence de réaliser les analyses de contraintes primaires, les analyses à 3 Sm et à la fatigue. La partie du programme sur laquelle nous allons nous concentrer est celle décrite dans le diagramme ci-dessous. Pour fonctionner, le programme se base sur deux éléments issus de résultat de calculs éléments finis : d'abord une mise en donnée décrivant le type d'analyse requise, les combinaisons de chargement à prendre en compte et la liste des coupes (section géométrique de la structure) à analyser. Puis, un fichier contenant, pour chaque coupe et différent chargement, les contraintes et températures en chaque point de la coupe.

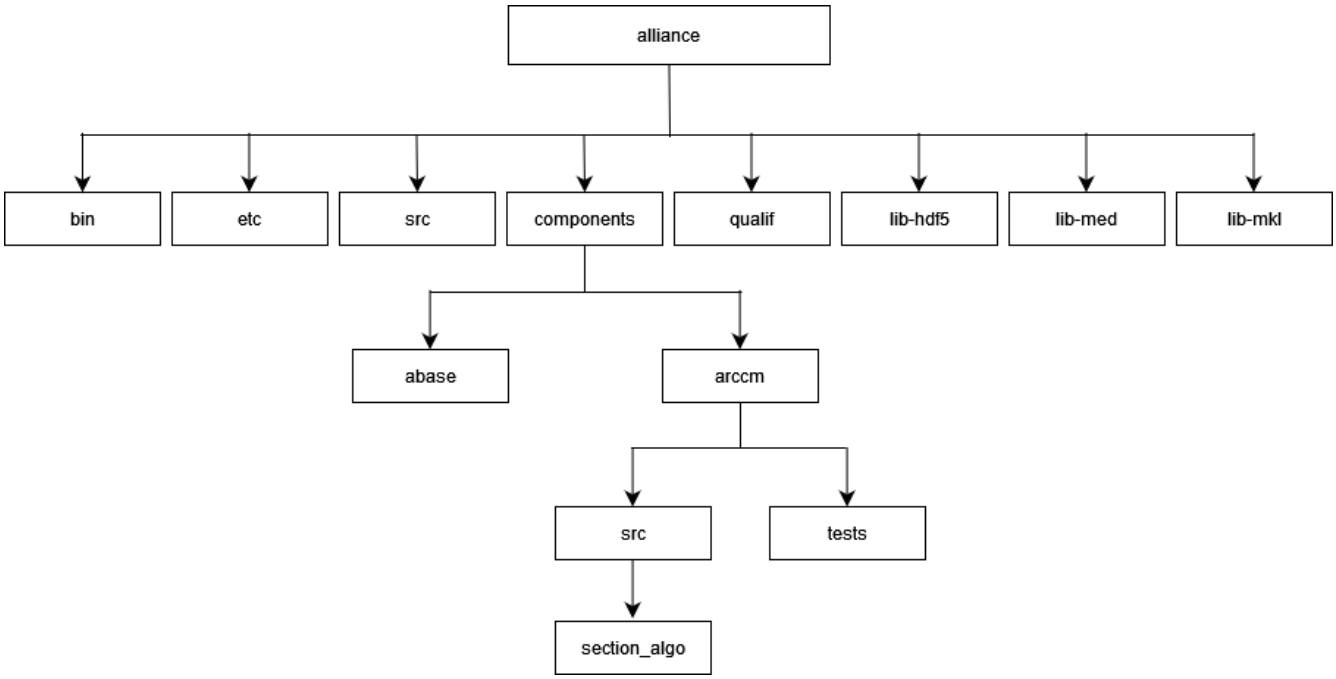

FIGURE 2.3 – Diagramme de l'arborescence du programme ALLIANCE

2.2.2 Fonctionnement

Sur la base de ces chargements, le programme (partie **section_algo**) détermine la contrainte totale associée aux différents points de la coupe géométrique. Dans le cas général, cette contrainte totale $\bar{\bar{\sigma}}_{tot}$ sera le cumul :

- d'une contrainte induite par un chargement unitaire de pression $\bar{\bar{\sigma}}_{pres}$, associé à un coefficient k_p,
- d'une contrainte thermoélastique $\bar{\bar{\sigma}}_{ther}$, associée à un coefficient k_h,
- de contraintes induites pour différents chargements d'efforts appliqués sur la structure. Chaque contrainte est notée $\bar{\bar{\sigma}}^i_{ef}$ et peut être associée à deux coefficients min et max notés c_i^{min} et c_i^{max} (on notera c_i^m dans le cas général)

Au final, la contrainte totale s'écrit de la manière suivante :

$$\bar{\bar{\sigma}}_{tot} = k_p \bar{\bar{\sigma}}_{press} + k_h \bar{\bar{\sigma}}_{ther} + \sum_{i=1}^{N} c_i^m \bar{\bar{\sigma}}^i_{ef}$$

L'analyse des contraintes primaires consiste à déterminer la contrainte maximale en maximisant les intensités des contraintes totales pour l'ensemble des combinaisons de chargements via la formule :

$$max\left[tresca\left(k_p\bar{\bar{\sigma}}_{press}+k_h\bar{\bar{\sigma}}_{ther}+\sum_{i=1}^{N}c_i^m\bar{\bar{\sigma}}_{ef}^i\right)\right]$$

La contrainte équivalente de Tresca est maximisée en tenant compte des coefficients appliqués sur les contraintes induites par les efforts en choisissant soit le coefficient maximal soit le coefficient minimal. Pour déterminer une amplitude maximale de contraintes, notée S_r, pour une liste de k instants de chargement, le programme `ALLIANCE` considère que les directions principales des contraintes peuvent varier au cours du temps. L'amplitude maximale de contraintes associée à 2 instants i, j est égale à :

$$\begin{aligned}S_r &= max\left[tresca\left(\bar{\bar{\sigma}}_{tot}(i)-\bar{\bar{\sigma}}_{tot}(j)\right)\right]\\ &= max\left[tresca\left(\left[k_p(i)-k_p(j)\right]\bar{\bar{\sigma}}_{press}+k_h(i)\bar{\bar{\sigma}}_{ther}(i)-k_h(j)\bar{\bar{\sigma}}_{ther}(j)+\sum_{p=1}^{N}\left[c_p^m(i)-c_p^m(j)\right]\bar{\bar{\sigma}}_{ef}^p\right)\right]\end{aligned}$$

Pour le calcul de S_r, le programme associe pour chaque contrainte induite par un effort soit le couple $< c_p^{max}(i); c_p^{min}(j) >$, soit le couple $< c_p^{min}(i); c_p^{max}(j) >$ afin de déterminer l'amplitude maximale. Il en résulte donc 2^N combinaisons possibles de coefficients associés aux efforts. Ainsi, pour l'ensemble des k instants de chargement, le nombre total de combinaisons est :

$$\frac{k(k-1)}{2}2^N$$

Voici le pseudo-code de la fonction chargé de calculer l'amplitude maximale de contrainte dans `ALLIANCE`.

Algorithm 1 calcul de l'amplitude de contrainte max - version originale

```
max = 0
comb_max = ()
comb_instants ← k(k − 1)/2
comb_tors ← 2^N
Pour c = 0 à comb_instants − 1 faire
    i, j = findIndex(c)
    Pour m = 0 à comb_tors − 1 faire
        comb = combinaison(i, j, m)
        val = tresca(comb)
        Si val > max alors
            max = val
            comb_max = comb
        Fin Si
    Fin Pour
Fin Pour
Renvoyer max, comb_max
```

On notera la présence de seulement deux boucles et non pas trois comme l'explication précédente pouvait le laisser penser. En effet, au lieu d'être composé de 2 boucles sur le nombre d'instants et une boucle sur $2^{\text{nb torseurs}}$, le nombre de boucles sur les instants est réduite à une seule et unique boucle sur $\frac{k(k-1)}{2}$. La motivation derrière cette modification est l'utilisation d'architecture distribuée. En effet, la structure de boucle seule rend la parallélisation plus simple qu'avec des boucles imbriquées.

Pour faire la correspondance entre l'unique indice c de cette boucle est les indices i, j identifiant les combinaisons et grâce auxquels on accédera aux données, il est nécessaire d'établir une règle. Pour cela, on part du postulat que les combinaisons sont symétriques, ainsi $i \in [0, k-1]$ et $j \in [i+1, k-1]$. Si on représente ces indices dans une matrice M avec i l'indice de la ligne et j l'indice de la colonne, on a la correspondance directe $c_{ij} = M_{ij}$.

$$M = \begin{pmatrix} n.a & 1 & 2 & 3 & \cdots & k-1 \\ n.a & n.a & k & k+1 & \cdots & \cdots \\ n.a & n.a & n.a & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ n.a & n.a & n.a & \cdots & n.a & \frac{k(k-1)}{2} \end{pmatrix} \tag{2.1}$$

Pour l'opération réciproque, il suffit de trouver la plus petite racine r du polynôme suivant : $i^2 - bi + 2c$ avec $b = 2k - 1$. On a $\Delta = b^2 - 8c$ donc $r = \frac{b-\sqrt{\Delta}}{2}$ et donc $i = \lceil r \rceil$. L'indice j est ensuite directement donné par la formule $j = c + p - \frac{1}{2}(i-1)(2k-i)$.

2.3 Contexte du stage

Comme indiqué précédemment, la complexité du temps de calcul de l'amplitude maximum de contrainte est de $O(2^N.k^2)$, ce qui correspond à une complexité exponentielle en N et quadratique en k. Il est ainsi naturel de se questionner sur les moyens d'optimiser ce programme tout en garantissant les résultats obtenus. Le but du stage est donc de proposer une architecture d'un programme en C++ optimisant l'actuel programme `ALLIANCE` écrit en Fortran et permettant une parallélisation OpenMP et MPI.

Chapitre 3

Optimisation

3.1 Modèle d'apprentissage

3.1.1 Contexte

Dans la quête d'optimisation du programme `ALLIANCE`, une première approche a été de se concentrer sur le calcul du critère de Tresca. En effet, calculer un Tresca est une opération très coûteuse et s'en abstenir serait synonyme de gains de temps significatifs. Pour cela, l'idée a été d'utiliser un modèle de prédiction basé sur du machine learning.

3.1.2 Modèle

Régression non linéaire

En machine learning, un modèle de régression non linéaire est une technique d'apprentissage permettant de modéliser des relations complexes et non linéaires entre des variables. Dans notre cas, nous essayons à travers une tel modèle de trouver une relation entre une matrice 3×3 symétrique et son critère Tresca.

Librairie scikit-learn

`scikit-learn` est une bibliothèque d'apprentissage automatique en open source et disponible sous licence BSD. Elle est écrite en Python et propose différents modèles d'apprentissage supervisés comme la classification et la régression. Pour notre étude, on utilisera une modèle de régression et plus particulièrement le modèle `RandomForestRegressor`.

`RandomForestRegressor` est une implémentation efficace et populaire de l'algorithme des forêts aléatoires pour les tâches de régression. Il s'appuie sur la combinaison de plusieurs arbres de décisions indépendants, créés à partir de sous-échantillons aléatoires des données d'entraînement. Chaque arbre est une structure hiérarchique de décisions basées sur les valeurs des caractéristiques d'entrées, et les prédictions finales du modèle sont obtenues en moyennant les prédictions de tout les arbres.

Implémentation

Pour utiliser ce modèle de manière optimale, il est d'abord nécessaire de générer des données d'entraînement et de test. Pour cela, on écrit la fonction `generateSym` qui génère un tableau de n matrices 3×3 symétriques stockées sous la forme de tableau de taille 6 dont les caractéristiques sont déterminées afin d'avoir un échantillon représentatif des cas observés en pratique. Dans l'ordre décroissant de la pondération : matrice aléatoire, matrice plane (xy, xz et yz), matrice diagonale, matrice hydrostatique (matrice scalaire) et enfin matrice cisaillements (matrice dont la diagonale est nulle). Les coefficients de la matrice sont générés aléatoirement de manière uniforme entre deux bornes opposées ($[-10000, 10000]$ dans notre cas). Une fois les données enregistrées, on entraîne le modèle à l'aide de la fonction `parallelTrain` de manière distribuée (parallélisation avec le module `Pool` de la librairie `multiprocessing`). Une fois l'apprentissage terminé, l'appel à `model.predict()` permet de tester la performance du modèle sur l'échantillon de test.

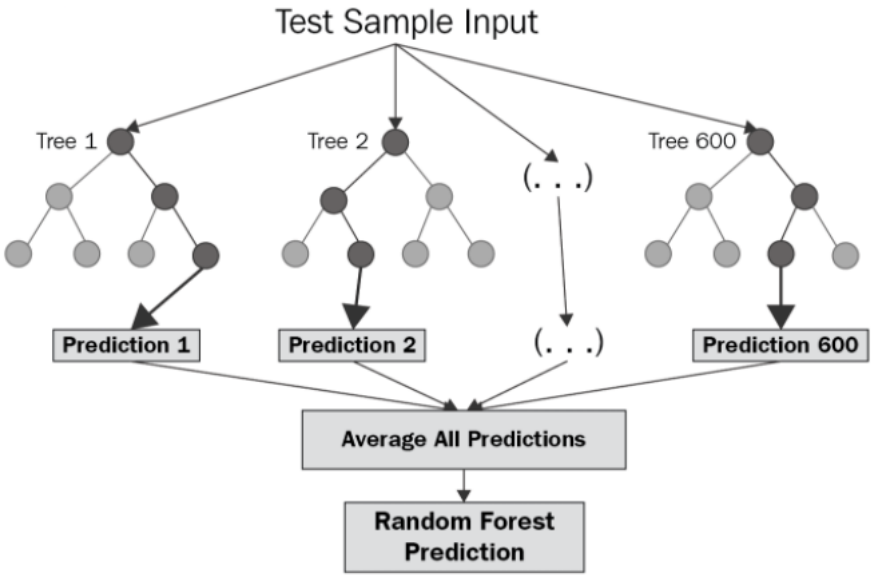

FIGURE 3.1 – Illustration du modèle `RandomForestRegressor`

```
def parallelTrain(X_train, y_train, model_save='model.joblib'):
 model = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1)
 model.fit(X_train, y_train)
 joblib.dump(model, model_save, compress=('gzip',3))
 size = asizeof.asizeof(model)
 print(f'Modèle sauvegardé sous: {model_save}, taille: {size} octets')
 return model


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = parallelTrain(X_train, y_train)
y_pred = model.predict(X_test)
```

3.1.3 Résultats

Pour évaluer un modèle de machine learning, `scikit-learn` propose plusieurs outils tels que la Mean Squared Error (MSE) et le R^2 Score.

La MSE mesure la moyenne des carrés des erreurs de prédictions, c'est-à-dire la différence entre les valeurs observées et les valeurs prédites. Avec n le nombre d'observations, y_i les valeurs réelles et $\hat{y}_i$ les valeurs prédites, sa formule est la suivante :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Une MSE plus faible indique que le modèle prédit les valeurs de manière plus précise. Étant donné qu'elle est basée sur les carrés des erreurs, les erreurs plus grandes sont pénalisées plus que les petites erreurs.

Le R^2 Score quant à lui mesure la proportion de la variance totale des données qui expliquée par le modèle. Il indique à quel point le modèle explique les variations des données. Avec $\bar{y}$ la moyenne des valeurs réelles, sa formule est la suivante :

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Un $R^2 = 1$ indique que le modèle explique parfaitement toute la variance des données. Un $R^2 = 0$ signifie que le modèle n'explique aucune des variations par rapport à la moyenne des valeurs. Un $R^2 < 0$ peut apparaître si le modèle est pire qu'un modèle naïf qui prédirait simplement la moyenne des valeurs cibles.

Le modèle d'apprentissage choisit est le suivant : avec $n = 1000000$ le nombre d'instances créées, on réalise l'apprentissage sur 80% de ces données et les tests sur les 20% restants. Le modèle de `RandomForestRegressor` est initialisé avec $n\_estimators = 100$ (correspond au nombre d'arbres dans la forêt). Les résultats sont alors les suivants :

```
Max Error: 7.867408455649333
Mean Squared Error: 1.408463652487627
R^2 Score: 0.9994768046986308
```

Les résultats à l'issue de l'apprentissage sont donc satisfaisants. Cependant, dans les faits, sa précision n'est pas aussi bonne lorsqu'il rencontre des nouvelles données. En effet, le problème de sur-apprentissage se pose du fait que le modèle est habitué à s'entraîner sur des données formatées et qu'il est compliqué d'avoir un échantillon représentant fidèlement les données rencontrées. De plus, le caractère non-linéaire et la plage de données d'entrées sont autant de facteurs qui rendent la prédiction du modèle lente, bien plus que le calcul du Tresca. L'approche par un modèle d'apprentissage n'a donc pas eu les résultats escomptés, et il a donc été nécessaire de trouver une autre approche au problème.

3.2 Approche théorique

3.2.1 Idée générale

Après avoir vu que l'approche par un modèle d'apprentissage n'était pas convaincante, un autre angle d'étude a été d'analyser le critère de Tresca d'un point de vue fonctionnel. Le calcul du critère de Tresca passant par un calcul de valeurs propres, il est donc très coûteux d'un point de vue algorithmique, et minimiser le nombre de Tresca calculés est une nécessité. Pour cela, l'idée a été de trouver un critère à la fois plus restrictif (c'est-à-dire qu'à un point donné de l'espace de contrainte, ce critère majore toujours le critère de Tresca) et plus rapide à calculer que le critère de Tresca permettant d'effectuer un tri dans les combinaisons. On commence pour cela à interpréter les critères en 3D dans l'espace des contraintes principales $(\sigma_1, \sigma_2, \sigma_3)$, il faut visualiser les surfaces de ruptures comme des volumes qui représentent des limites de plasticité du matériau. Dans la représentation 3.2, le critère de Tresca forme un prisme hexagonal droit centré sur l'axe hydrostatique($\sigma_1 = \sigma_2 = \sigma_3$) et le critère de von Mises forme un cylindre circonscrit au prisme hexagonal de Tresca.

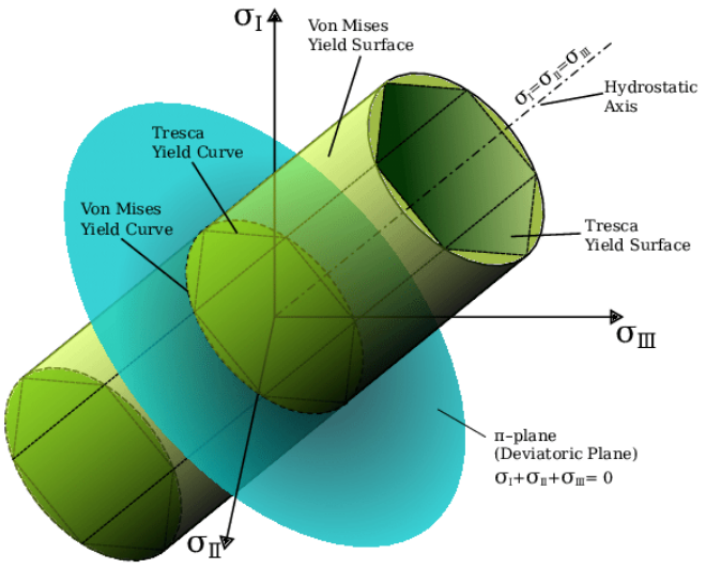

FIGURE 3.2 – Critères de von Mises et Tresca dans l'espace des contraintes principales

En réalisant une coupe de cet espace normal à l'axe hydrostatique, la géométrie des critères devient plus claire et facile à comparer.

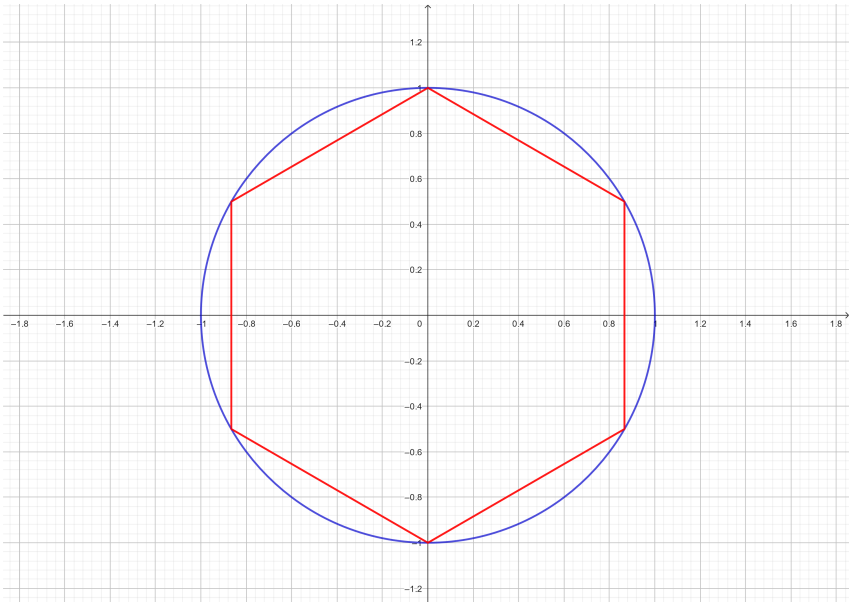

FIGURE 3.3 – Coupe du plan normal à la trisectrice représentant les critères de Tresca(rouge) et von Mises(bleu)

À l'aide de ce nouveau schéma, il est désormais plus simple d'identifier et de formaliser un critère comme défini dans le paragraphe précédent : on choisit le critère représenté par le cercle en vert inscrit dans l'hexagone de Tresca sur la figure 3.4. Ainsi, de la même manière que le critère de Tresca est plus restrictif que le critère de von Mises, notre nouveau critère sera plus restrictif que celui de Tresca.

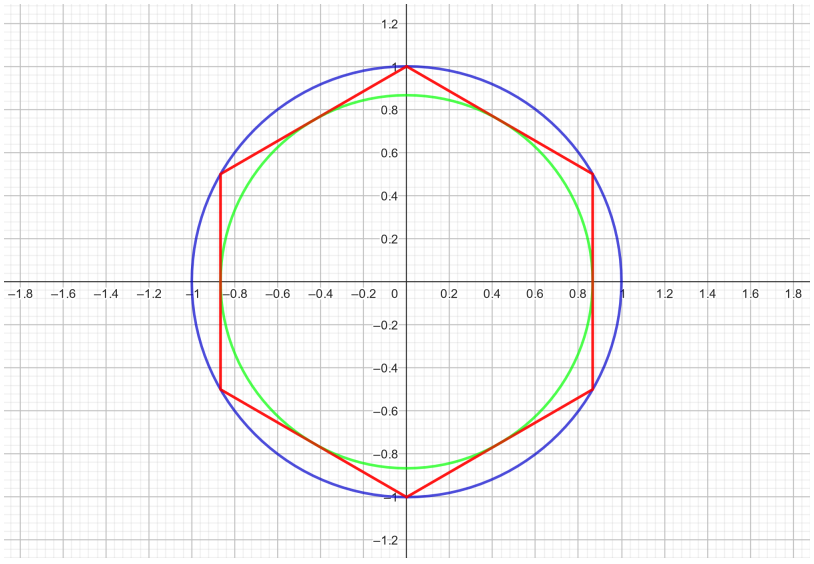

FIGURE 3.4 – Nouveau critère (vert) dans le plan normal à la trisectrice

D'un point de vue mathématique, cela signifie que l'on aura toujours :

$$\sigma_{VM} \leq \sigma_T \leq \sigma_{new}$$

3.2.2 Formulation

Pour trouver maintenant la formulation exacte de ce critère, on peut observer sur le schéma 3.4 que son tracé ressemble à une homothétie réduite du critère de von Mises. Pour trouver le facteur de réduction, qui correspond au rapport des rayons respectifs de chaque critère, le changement de repère nous facilite la démarche. En effet, en considérant le triangle rectangle suivant, on peut trouver de manière directe le rayon du nouveau critère.

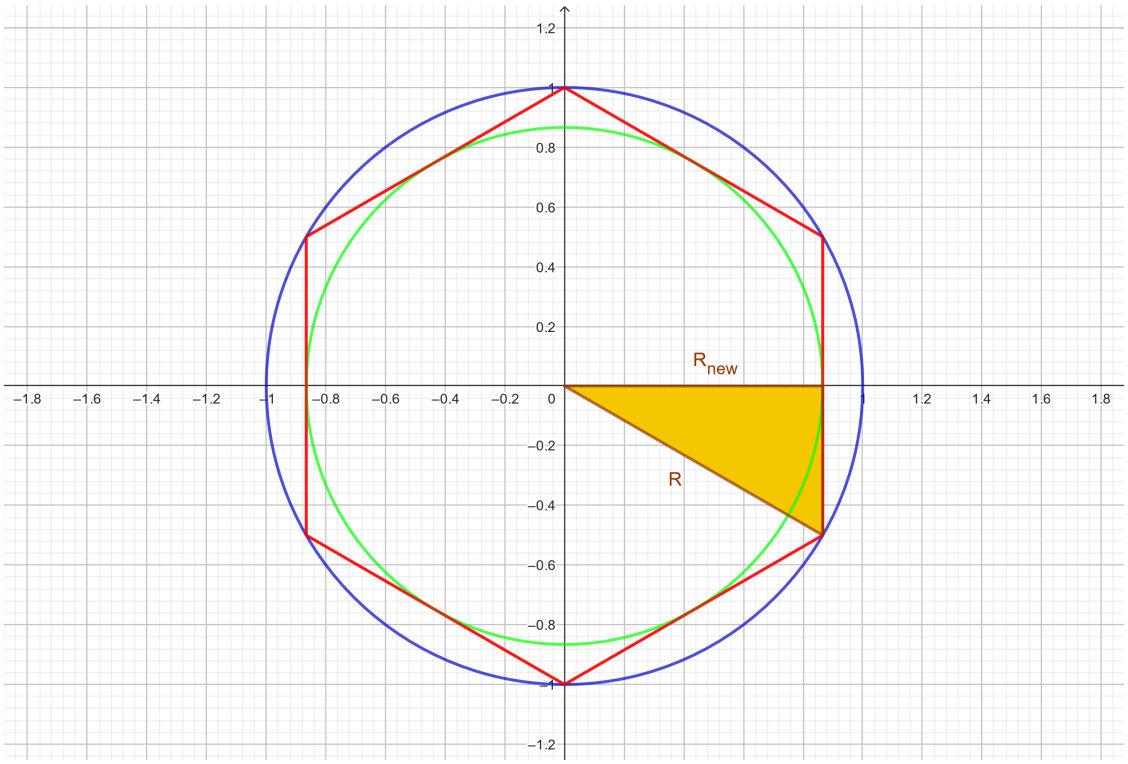

FIGURE 3.5 – Rayon des deux critères

Comme l'hexagone est régulier, on peut déterminer facilement les angles :

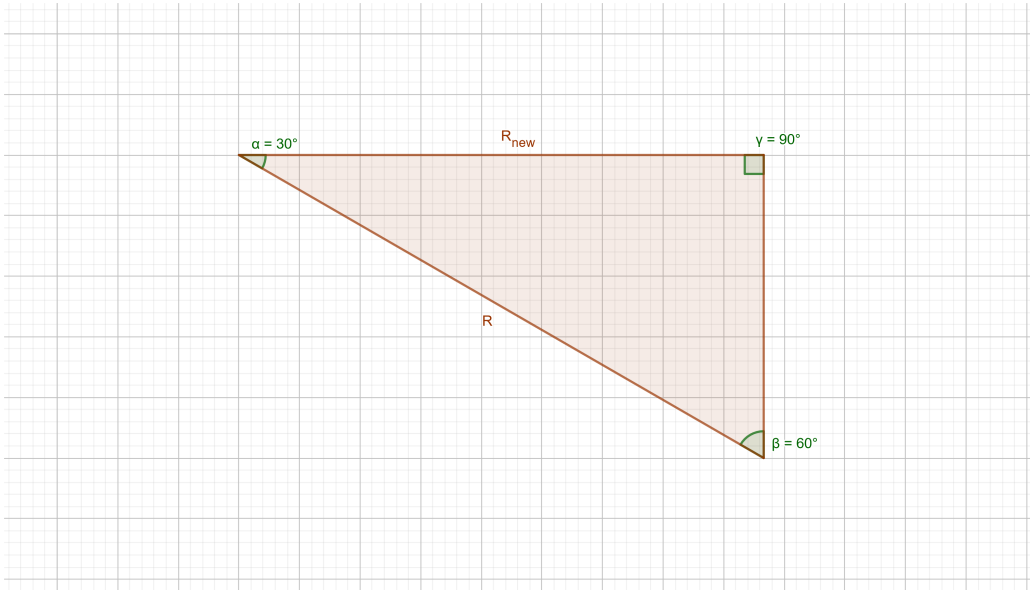

FIGURE 3.6 – Angles du triangle

La règle de trigonométrie donne donc directement :

$$sin(\beta) = \frac{R_{new}}{R} \Leftrightarrow R_{new} = Rsin(\beta)$$

or, on a $\beta = \frac{\pi}{3}$ donc $R_{new} = Rsin(\frac{2\pi}{3}) = \frac{\sqrt{3}}{2}R$. Ainsi, le facteur de réduction est $\frac{R_{new}}{R} = \frac{2}{\sqrt{3}}$. Et de cette manière, on a $\sigma_{new} = \frac{2}{\sqrt{3}}\sigma_{VM}$

3.2.3 Preuve

Démontrons à présent de manière formelle l'inégalité trouvée géométriquement :

$$\sigma_{VM} \leq \sigma_T \leq \frac{2}{\sqrt{3}}\sigma_{VM}$$

Démonstration. On commence par montrer la partie droite de l'inégalité, à savoir que $\sigma_T \leq \frac{2}{\sqrt{3}}\sigma_{VM}$. On considère pour cela la fonction σ_{VM} que l'on cherche à minimiser. On peut supposer sans perdre de généralité que $\sigma_1 \geq \sigma_2 \geq \sigma_3$.

$$\sigma_{VM} = \sqrt{\frac{1}{2}[(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_1 - \sigma_3)^2]} \text{ avec } \sigma_1 \geq \sigma_2 \geq \sigma_3$$

On utilise ensuite le changement de variable $x = \sigma_2 - \sigma_3$ et $T = \sigma_1 - \sigma_3$ (on remarque alors que $\sigma_T = |T|$), ce qui donne comme nouvelle expression :

$$\sigma_{VM} = \sqrt{\frac{1}{2}[x^2 + T^2 + (T - x)^2]} \text{ avec } 0 \leq |x| \leq |T|$$

Souhaitant étudier l'inégalité à une valeur de Tresca donnée, on s'intéresse à l'étude de la fonction à T fixé. Le point critique qui nous intéresse donc est celui associé à la dérivée partielle de la fonction par rapport à x.

$$\partial_x \sigma_{VM} = \frac{2x - T}{\sqrt{2[x^2 + T^2 + (T - x)^2]}}$$

Le point critique associé à $\partial_x \sigma_{VM}$ est alors la solution de $\partial_x \sigma_{VM} = 0$.

$$\partial_x \sigma_{VM} = 0 \Leftrightarrow \frac{2x - T}{\sqrt{2[x^2 + T^2 + (T - x)^2]}} = 0 \Leftrightarrow 2x - T = 0 \Leftrightarrow x = \frac{T}{2}$$

Le point critique est donc $x = \frac{T}{2}$, ce qui équivaut à $\sigma_2 - \sigma_3 = \frac{\sigma_1 - \sigma_3}{2}$ d'où $\sigma_2 = \frac{\sigma_1 + \sigma_3}{2}$. Vérifions maintenant qu'il s'agit d'un minimum en montrant que σ_{VM} est convexe, c'est-à-dire que $\partial^2_{xx}\sigma_{VM} \geq 0$.

$$\partial^2_{xx}\sigma_{VM} = \frac{3T^2}{4(x^2 - xT + T^2)^{\frac{3}{2}}} \geq 0$$

Il s'agit donc bien d'un minimum est le minimum de la fonction est donné par :

$$\sigma_{VMmin} = \sigma_{VM}(\frac{T}{2}, T) = \frac{1}{\sqrt{2}}\sqrt{\frac{T^2}{4} + T^2 + (T - \frac{T}{2})^2} = \frac{1}{2\sqrt{2}}\sqrt{T^2 + 4T^2 + T^2} = \frac{\sqrt{3}}{2}|T| = \frac{\sqrt{3}}{2}\sigma_T$$

Ainsi $\forall x, T$ on a $\sigma_{VMmin} = \frac{\sqrt{3}}{2}\sigma_T \leq \sigma_{VM}$, d'où $\sigma_T \leq \frac{2}{\sqrt{3}}\sigma_{VM}$.

Du caractère convexe de σ_{VM} et par symétrie de x et T on a également que le maximum de la fonction est atteint sur les bords du domaine, à savoir pour $\sigma_2 = \sigma_1$ où $\sigma_2 = \sigma_3$. Sans perdre de généralité, prenons $\sigma_2 = \sigma_3$ on a alors $x = 0$ et donc :

$$\sigma_{VMmax} = \sigma_{VM}(0, T) = \frac{1}{\sqrt{2}}\sqrt{2T^2} = |T| = \sigma_T$$

Ainsi, $\forall x, T$ on a $\sigma_{VMmax} = \sigma_T \geq \sigma_{VM}$, ce qui montre la partie gauche de l'inégalité et conclut la preuve. □

À l'aide de ce nouveau critère, on peut à présent utiliser une nouvelle version de l'algorithme qui calcule l'amplitude de contrainte maximum :

Algorithm 2 calcul de l'amplitude de contrainte max - version filtre

$max = 0$
$comb\_max = ()$
$comb\_instants \leftarrow k(k-1)/2$
$comb\_tors \leftarrow 2^N$
Pour $c = 0$ **à** $comb\_instants - 1$ **faire**
    $i, j = findIndex(c)$
    Pour $m = 0$ **à** $comb\_tors - 1$ **faire**
        $comb = combinaison(i, j, m)$
        $val = vonMises(comb)$
        Si $val < max$ **alors**
            continue
        Sinon alors
            $val = tresca(comb)$
            Si $val > max$ **alors**
                $max = val$
                $comb\_max = comb$
            Fin Si
        Fin Si
    Fin Pour
Fin Pour
Renvoyer $max, comb\_max$

3.3 Approche structurelle

3.3.1 Parallélisation

La parallélisation avec MPI(Message Passing Interface) et OpenMP(Open Multi-Processing) repose sur des modèles différents. MPI est conçu pour des systèmes à mémoire distribuée, où chaque processus fonctionne sur un nœud distinct avec sa propre mémoire, et les échanges entre processus se font par l'envoi de messages explicites, ce qui est particulièrement adapté aux clusters de calcul. En revanche, OpenMP est conçu pour des architectures à mémoire partagée, comme les systèmes multicœurs, où plusieurs threads (processus OpenMP) peuvent s'exécuter en parallèle sur des cœurs différents, mais partagent le même espace mémoire. MPI est plus complexe en raison de la nécessité de gérer la communication entre processus, mais il est adapté à des systèmes massivement parallèles, tandis qu'OpenMP simplifie la parallélisation en ajoutant des directives aux boucles et sections critiques dans le code, en exploitant efficacement les ressources de machines multiprocesseurs sans nécessiter de communication explicite entre les threads.

3.3.2 Implémentation

MPI se reposant sur l'échange de messages entre les processus, si un algorithme contient des boucles imbriquées, les processus pourraient avoir à s'échanger des données à chaque itération des différentes boucles. Cela peut augmenter considérablement le volume de communications, qui est souvent un goulot d'étranglement dans les systèmes distribués. L'idée dans cette approche structurelle est donc d'aplanir les boucles de l'algorithme. On se retrouve ainsi avec une boucle d sur $\frac{k(k-1)}{2}2^N$. Concernant la correspondance des indices, on identifie l'indice $m \in [0, 2^{nb\_tors} - 1]$ en calculant la division euclidienne de d par $\frac{k(k-1)}{2}$: le quotient donnera l'indice m et le reste donnera c, à partir duquel on trouvera i, j par le même raisonnement expliqué par la figure 2.1. L'algorithme obtenu après cette modification est le suivant :

Algorithm 3 calcul de l'amplitude de contrainte max - version filtre et aplanie

$max = 0$
$comb\_max = ()$
$comb\_instants \leftarrow k(k-1)/2$
$comb\_tors \leftarrow 2^N$
$comb\_total \leftarrow comb\_instants * comb\_tors$
Pour $d = 0$ **à** $comb\_total - 1$ **faire**
    $i, j, m = findIndex(d)$
    $comb = combinaison(i, j, m)$
    $val = vonMises(comb)$
    Si $val < max$ **alors**
        continue
    Sinon alors
        $val = tresca(comb)$
        Si $val > max$ **alors**
            $max = val$
            $comb\_max = comb$
        Fin Si
    Fin Si
Fin Pour
Renvoyer $max, comb\_max$

Chapitre 4

Résultats

Dans cette partie, nous allons présenter les résultats liés aux différentes optimisations évoquées dans le chapitre précédent. Pour mesurer l'amélioration des performances, nous prendrons comme référence une version C++ du code équivalent (dans une certaine mesure) en termes de structure et d'algorithmes à la version actuelle de `ALLIANCE` écrite en Fortran. En effet, de par le langage de programmation et les différentes valeurs calculées, il n'est pas pertinent de comparer mon implémentation à celle existante de `ALLIANCE`. Dans la suite, la mention de version originale fera donc référence à son équivalent C++.

4.1 Approche théorique

Pour évaluer l'impact de la version filtrée par l'homothétie du critère de von Mises, on compare les temps d'exécution de la version originale avec cette nouvelle version pour différents tests de taille variable.

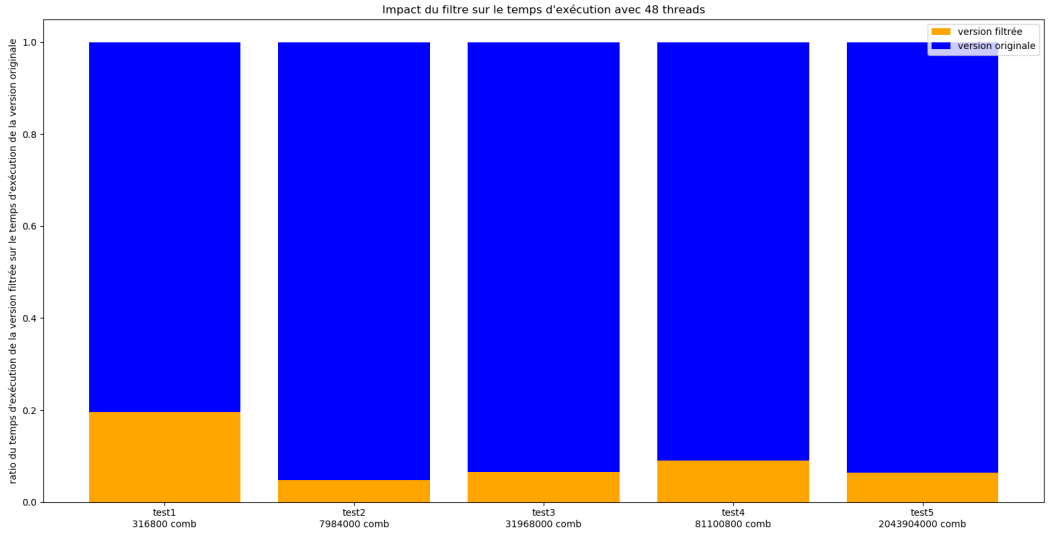

FIGURE 4.1 – Impact du filtre sur le temps d'exécution

On voit sur ce graphique de manière assez nette que le temps d'exécution est divisé par 5 dans le pire cas et près de 20 dans le meilleur cas, ce qui représente un gain en temps non négligeable.

On peut également se poser la question des facteurs qui rentrent en jeu dans cette amélioration de performance. Comme le montre le graphique ci-dessus, la taille du problème (nombre de combinaisons) n'est pas le seul élément à considérer. En effet, l'ordre dans lequel sont calculées les combinaisons importe également : le meilleur cas sera quand la première combinaison donne le maximum, ainsi seulement un Tresca sera calculé. Le pire cas quant à lui sera lorsque les combinaisons sont classées par Tresca croissant, ainsi, on sera obligé de calculer leur Tresca à chaque itération. Le graphique suivant montre le nombre de Tresca calculés par rapport au nombre de combinaisons totales.

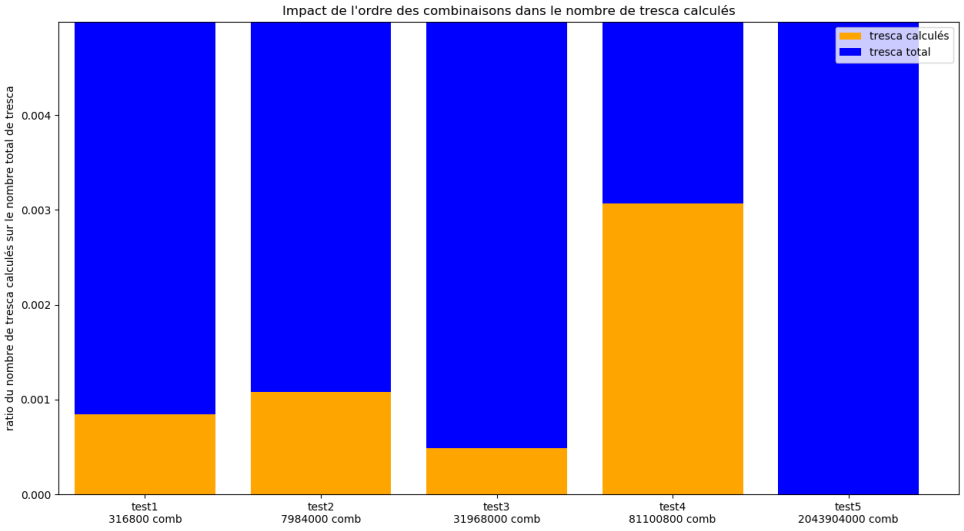

FIGURE 4.2 – Impact de l'ordre des combinaisons sur le nombre de Tresca calculés

On voit que le nombre de Tresca calculé est très faible dans tous nos cas tests. Ainsi, à moins que les combinaisons donnent des Tresca croissants, ce qui n'est presque jamais le cas en pratique, l'ordre des combinaisons n'a qu'une influence très faible sur le gain de temps.

Pour conclure sur cette partie, le filtre mis au point apparaît sans aucun doute comme une amélioration importante des performances qui plus est facile à implémenter dans la version actuelle de `ALLIANCE`. Pour la suite des expérimentations, on utilisera cette version de l'algorithme comme base.

4.2 Approche structurelle

Pour évaluer à présent l'impact sur les performances de la modification structurelle de l'algorithme ainsi que le choix d'une parallélisation MPI plutôt qu'OpenMP, on utilise un test de grande taille `06-CAT2_Z120P6_FATIGUE.DAT` qui comporte 2043904000 combinaisons. On trace ensuite les courbes d'accélération et d'efficacité de la parallélisation pour la version 2 boucles utilisant OpenMP et la version une boucle utilisant MPI.

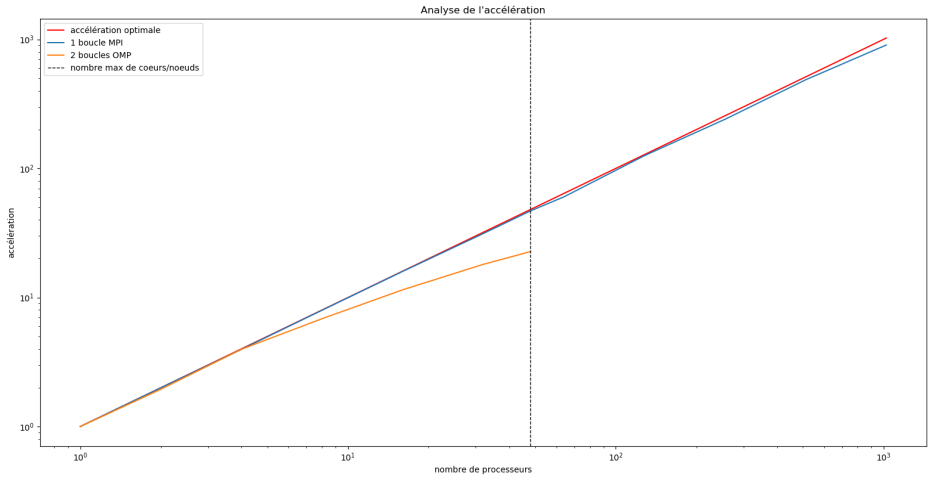

FIGURE 4.3 – Graphique représentant d'accélération de la parallélisation pour les deux implémentations

Sur ce graphique est représenté l'accélération de la parallélisation, à savoir $\frac{\text{temps}(1)}{\text{temps}(N)}$, pour les deux implémentations parallèles. En d'autres termes, cela mesure l'amélioration des performances obtenues grâce à la parallélisation. On représente par la courbe rouge une accélération optimale (quand l'accélération égale le nombre de processeurs). Concernant les deux implémentations, on observe qu'elles ont un comportement optimal pour peu de processeurs puis, tandis ce que la version MPI continue d'être proche de l'accélération optimale jusqu'à 1024 processeurs, la version OpenMP décroche assez rapidement aux alentours des 48 processeurs. Cela est dû à la synchronisation et aux communications liées à la mémoire partagée : à mesure que le nombre de threads augmente, cette surcharge devient plus importante, ce qui réduit l'efficacité globale.

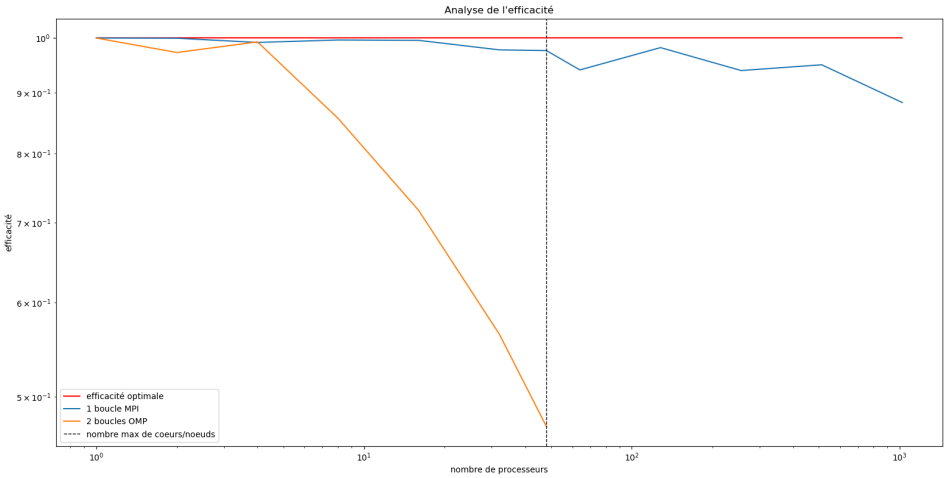

FIGURE 4.4 – Graphique représentant l'efficacité de la parallélisation pour les deux implémentations

Ce graphique représente l'efficacité de la parallélisation, donnée par $\frac{\text{temps(1)}}{N\text{temps(N)}}$, pour les deux implémentations. La courbe rouge représente l'efficacité théorique idéale, c'est-à-dire 1, signifiant qu'une augmentation du nombre de processeurs entraîne un gain de performance linéaire. La courbe orange représentant la version OpenMP montre une chute rapide de l'efficacité dès que le nombre de processeurs augmente. Cela indique que cette implémentation souffre d'une perte d'efficacité due à la synchronisation et la surcharge de communication. La courbe bleue, montrant l'efficacité de la version MPI, est plus stable quant à elle, indiquant une meilleure efficacité comparée à l'implémentation OpenMP même si elle commence à diminuer lorsque le nombre de processeurs augmente.

les analyses des graphiques d'accélération et d'efficacité nous poussent donc à la même conclusion : les deux implémentations parallèles ont des performances équivalentes lorsque le nombre de processus utilisé est relativement faible. Cependant, lorsque ce dernier augmente, la version MPI montre une meilleure scalabilité. Cela suggère que cette implémentation est mieux optimisée pour l'utilisation d'un grand nombre de processus.

Chapitre 5

Conclusion

Au terme de ce stage, il est essentiel de revenir sur les objectifs initiaux, les défis rencontrés, et les réussites obtenues. Ce stage a été l'occasion de mettre en pratique les connaissances théoriques acquises tout au long de mon cursus et de développer des compétences pratiques précieuses. Les expériences vécues ont non seulement enrichi ma compréhension du domaine, mais ont également mis en lumière les aspects clés du travail en entreprise. Pour reprendre chronologiquement le déroulé de mon stage, la première difficulté à laquelle j'ai été confronté fut la prise en main et la manipulation d'outils utilisés en mécaniques. En effet, venant d'un cursus mathématique et informatique, beaucoup de notions m'étaient totalement étrangères. La deuxième épreuve a été de comprendre et d'implémenter l'algorithme utilisé dans `ALLIANCE`. Le Fortran étant un langage de programmation que je ne maîtrise pas, la lecture du code source n'a pas suffi à ma compréhension, il a donc fallu m'adresser aux collègues de travail afin qu'ils me fournissent une explication adaptée. Une fois cette partie acquise, ma première approche au problème d'optimisation a été de concevoir un modèle d'apprentissage permettant de prédire les valeurs de Tresca de manière précise et plus rapide que le calcul en lui-même. Cependant, la variabilité des données et la complexité mathématique de ce critère ont rendu les résultats peu concluants. La deuxième approche a été de mettre au point un filtre permettant de limiter le nombre de Tresca calculés. Pour cela, l'utilisation d'un nouveau critère, semblable au critère de von Mises a été utilisé et a permis d'obtenir des gains très conséquents en termes de complexité algorithmique. L'avantage de cette optimisation est qu'elle est facilement réalisable, et peu d'ors et déjà être dans la version actuelle en Fortran du programme `ALLIANCE` avec l'assurance d'un gain industriel. Enfin, ma troisième et dernière approche a été de modifier de manière structurelle l'algorithme utilisé dans le programme, en privilégiant une parallélisation MPI plutôt qu'OpenMP. Pour rendre, cette parallélisation plus efficace et plus facilement réalisable, la structure de l'algorithme a été réduit à une seule boucle au lieu de deux. L'avantage d'une telle implémentation a été de pouvoir utiliser la parallélisation de manière plus conséquente, rendant la résolution de problèmes de grandes tailles beaucoup plus accessible. Concernant les pistes d'améliorations, d'un point de vue purement calculatoire, une idée serait de mettre au point une version utilisant des processeurs graphiques. En effet, la structure du programme étant SIMD(Simple Instruction Multiple Data), l'utilisation de GPU siérait parfaitement et pourrait de manière certaine proposer des gains encore plus conséquents. De manière générale, le but du stage était de proposer une architecture amorçant la refonte du programme `ALLIANCE` en C++. M'étant concentré sur une seule partie du programme, les possibilités d'améliorations restent importantes.

Bibliographie

[1] *American Society of Mechanical Engineers,Boiler and Pressure Vessel Code*. Edition 2023.

[2] *Règles de conception et de construction applicables aux materiles mécaniques des ilots nucléaires*. Edition 2022.

[3] OpenMP Architecture Review Board. *OpenMP API 5.2 Specification*, November 2021.

[4] Americo Cunha Jr, Yasar Yanik, Carlo Olivieri, and Samuel da Silva. Tresca vs. von Mises : Which failure criterion is more conservative in a probabilistic context ? *Journal of Applied Mechanics*, 91 :111008, 2024.

[5] Message Passing Interface Forum. *MPI : A Message-Passing Interface Standard Version 4.1*, November 2023.

[6] Framatome. Programme alliance - version 2.0 - analyse fonctionnelle. Technical report, Framatome, 2020. Courier Framatome DI-TC-2020.417 rév. E.

[7] Framatome. Testlaunch - utilitaire d'aide à la qualification. Technical report, Framatome, 2020. Courier Framatome EFFNC 2626 rév. I.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn : Machine learning in python. *Journal of Machine Learning Research*, 12(85) :2825–2830, 2011.

[9] Lagioia R. and Panteghini A. On the existence of a unique class of yield and failure criteria comprising tresca, von mises, Drucker–Prager,Mohr–Coulomb,Galileo–Rankine, Matsuoka–Nakai and Lade–Duncan. *Proc. Math. Phys. Eng. Sci.*, 472(2185) :20150713, January 2016.

[10] S. Timoshenko and J. N. Goodier. *Theory of Elasticity*. McGraw-Hill, 1959.