Titre: Title:	Déroulement de phase 3D grâce à la fermeture optimale des boucles de résidus
Auteur: Author:	El Mehdi Oudaoud
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Oudaoud, E. M. (2024). Déroulement de phase 3D grâce à la fermeture optimale des boucles de résidus [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/61865/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/61865/
Directeurs de recherche: Advisors:	Thibaut Vidal
Programme: Program:	Mathématiques appliquées

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Déroulement de phase 3D grâce à la fermeture optimale des boucles de résidus

EL MEHDI OUDAOUD
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Décembre 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Déroulement de phase 3D grâce à la fermeture optimale des boucles de résidus

présenté par **El Mehdi OUDAOUD**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Louis-Martin ROUSSEAU, président
Thibaut VIDAL, membre et directeur de recherche
Youssef DIOUANE, membre

DÉDICACE

À tous les amis que je me suis fait durant cette maîtrise,
vous me manquerez...

REMERCIEMENTS

Je tiens à exprimer ma profonde gratitude à mon directeur de recherche Thibaut Vidal, dont le soutien constant et les conseils précieux ont été déterminants tout au long de cette maîtrise. Grâce à son encadrement rigoureux, j'ai pu mener à bien ce travail, rédiger ce mémoire, et exposer de manière claire et fluide des concepts mathématiques parfois complexes. Je le remercie sincèrement pour le temps qu'il m'a accordé et pour son investissement dans ce projet.

Je souhaite également remercier M. Youssef Emine pour son aide précieuse durant toute ma maîtrise. Mes remerciements vont aussi à Mme Maria Bazzotte, dont les conseils avisés et le soutien dans les implémentations utilisant la résolution avec les problèmes de programmation linéaire en nombres entiers (MIP) ont été d'une grande valeur. Enfin, je tiens à exprimer ma reconnaissance à M. Chiekh Ahmed, pour avoir été une oreille attentive à qui j'ai souvent exposé mes idées et explications afin de valider leur cohérence.

Enfin, je tiens à remercier chaleureusement ma famille pour leur soutien moral inestimable tout au long de ma maîtrise. Je souhaite également exprimer ma gratitude envers les amis que j'ai rencontrés durant ces années à Montréal, une ville qui restera gravée dans ma mémoire pour toujours.

RÉSUMÉ

Le dépliement de phase en trois dimensions est un problème fondamental dans des domaines tels que l'imagerie médicale et la tomographie optique, où l'objectif est de reconstruire une phase absolue à partir de données enroulées dans l'intervalle $]-\pi, \pi]$. Deux approches principales sont généralement utilisées : des méthodes basées sur des informations locales, comme la cohérence physique, qui peuvent accumuler des erreurs, et des méthodes qui visent à identifier les zones problématiques à forte erreur et à les éviter efficacement pendant la reconstruction pour garantir la précision.

Cette étude présente une nouvelle approche pour traiter les zones problématiques qui, en trois dimensions, forment des *boucles de singularité* (ou *boucles fermées d'erreurs*). En appliquant la théorie des graphes pour modéliser le problème, notre méthode saisit efficacement les complexités topologiques des données, ce qui permet une détection précise de ces boucles. Nous implémentons un algorithme d'optimisation exact pour minimiser les *surfaces d'erreur* associées. Cela résout rigoureusement les boucles de singularité et garantit qu'aucun chemin d'intégration ne traverse des régions d'erreur, assurant ainsi une reconstruction robuste et cohérente de la phase.

Les résultats obtenus sur des ensembles de données synthétiques montrent que la méthodologie proposée permet des reconstructions précises et fiables, souvent comparables ou supérieures à celles des méthodes exactes conventionnelles, bien qu'au prix d'un temps de calcul élevé. Pour les données empiriques, l'approche conserve sa robustesse et sa cohérence en termes de qualité de reconstruction, mais reste exigeante en termes de calcul. Ce cadre offre une solution pratique pour relever les défis du déroulement de phase dans des applications exigeantes, telles que l'imagerie par résonance magnétique (IRM). En combinant la théorie des graphes et l'optimisation, ce travail contribue à améliorer la précision et la fiabilité de la reconstruction de phase en trois dimensions.

ABSTRACT

Three-dimensional phase unwrapping is a fundamental problem in fields such as medical imaging and optical tomography, where the goal is to reconstruct an absolute phase from wrapped data confined within the interval $]-\pi, \pi]$. Two main approaches are generally used: methods based on local information, such as physical coherence, which may accumulate errors, and methods aiming to identify problematic regions with high errors and efficiently avoid them during reconstruction to ensure accuracy.

This study introduces a new approach to address problematic regions that, in three dimensions, form *singularity loops* (or *closed error loops*). By applying graph theory to model the problem, our method effectively captures the topological complexities of the data, enabling precise detection of these loops. We implement an exact optimization algorithm to minimize the associated *error surfaces*. This rigorously resolves the singularity loops and ensures that no integration path crosses error regions, thereby achieving a robust and consistent phase reconstruction.

The results obtained on synthetic datasets demonstrate that the proposed methodology allows for accurate and reliable reconstructions, often comparable to or better than conventional exact methods, albeit at the cost of high computation times. For empirical data, the approach maintains its robustness and consistency in terms of reconstruction quality but remains computationally demanding. This framework provides a practical solution to address the challenges of phase unwrapping in demanding applications such as magnetic resonance imaging (MRI). By combining graph theory and optimization, this work contributes to improving the accuracy and reliability of three-dimensional phase reconstruction.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

IRM	Imagerie par Résonance Magnétique
PLNE	Problème Linéaire en Nombres Entiers
2D	Deux dimensions
3D	Trois dimensions
BFS	Parcous en Largeur
DFS	Parcours en Profondeur
LCA	Plus Petit Ancêtre Commun
OCT	Tomographie par cohérence optique
InSAR	Radar à Synthèse d'ouverture Interférométrique
$\oplus$	Différence Symétrique
mod	Modulo

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Le déroulement de phase est une technique fondamentale utilisée dans divers domaines scientifiques et techniques, tels que l'imagerie médicale, l'interférométrie radar et l'optique. Il permet de reconstruire une information de phase à partir de mesures qui sont limitées à une plage restreinte, typiquement entre $-\pi$ et π. Cette reconstruction est essentielle pour interpréter correctement les données et extraire des informations précises sur les phénomènes étudiés.

Mathématiquement, dans un problème multidimensionnel, où la phase est une fonction définie de $\mathbb{R}^D$ vers $\mathbb{R}$, avec $D = 1, 2$ ou 3, la phase enroulée Φ peut être exprimée à partir de la phase absolue Ψ selon la relation suivante :

$$\Phi = \Psi \mod 2\pi,$$

où l'opération de modulo ramène la phase dans l'intervalle $[-\pi, \pi]$.

Si on note *wrap* la fonction qui vient enrouler la phase Ψ entre $-\pi$ et π, et qui est définie comme suit :

$$\text{wrap}(\Psi) = (\Psi + \pi \mod 2\pi) - \pi$$

Ainsi, on peut dire que $\Phi = \text{wrap}(\Psi)$.

Le défi central du déroulement de phase consiste donc à reconstruire la phase absolue Ψ à partir de Φ. Cela nécessite de déterminer, pour chaque mesure de phase enroulée, le nombre de multiples de 2π qu'il faut ajouter ou soustraire pour obtenir une phase continue et cohérente sur l'ensemble des données.

1.1 Définitions et concepts de base

Nous introduisons la métrique d'erreur L_p, qui permet de quantifier la qualité de la reconstruction du signal de phase en dimensions $D = 1$, 2 ou 3. Cette métrique mesure la différence entre les gradients de la phase enroulée et de la phase déroulée, tout en tenant compte de la dimension spatiale du signal.

La métrique L_p est définie pour un signal de dimension D comme suit :

$$L_p = \left(\sum_{r=1}^{D} \sum_{\mathbf{i}} |\nabla_r \Phi(\mathbf{i}) - \nabla_r \Psi(\mathbf{i})|^p \right)^{\frac{1}{p}},$$

où :

- $\mathbf{i} = (i_1, i_2, \ldots, i_D)$ représente les indices de position dans l'espace discret de dimension D.
- r parcourt les D directions spatiales (par exemple, $r \in \{x, y, z\}$ pour $D = 3$).
- $\nabla_r \Phi(\mathbf{i})$ et $\nabla_r \Psi(\mathbf{i})$ désignent respectivement les gradients de la phase enroulée et déroulée le long de la direction r au point $\mathbf{i}$.

Le gradient dans la direction r est calculé comme la différence de phase entre le point $\mathbf{i}$ et son voisin immédiat dans cette direction :

$$\nabla_r \Phi(\mathbf{i}) = \Phi(\mathbf{i} + \mathbf{e}_r) - \Phi(\mathbf{i}),$$

où $\mathbf{e}_r$ est le vecteur unitaire dans la direction r.

Cette définition est générale et s'applique aux dimensions suivantes :

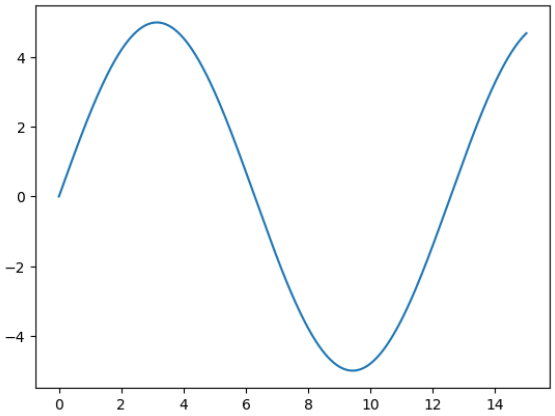
- **Dimension** $D = 1$: Le calcul est réalisé sur une seule direction spatiale, simplifiant ainsi l'expression de L_p.
- **Dimension** $D = 2$: Les gradients sont calculés selon les directions x et y, capturant les variations spatiales dans un plan.
- **Dimension** $D = 3$: Les trois directions x, y et z sont considérées, adaptées aux données volumétriques et aux applications tridimensionnelles.

Minimiser L_p revient à obtenir une phase déroulée Ψ dont les gradients se rapprochent au mieux de ceux de la phase enroulée Φ, tout en respectant la continuité spatiale dans la dimension spécifique étudiée. Cette approche assure une reconstruction fidèle et adaptée aux propriétés dimensionnelles du signal.

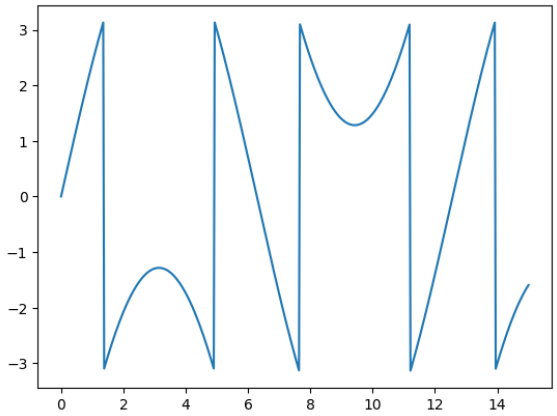
1.1.1 Le problème du déroulement de phase en une dimension

Pour l'exemple d'une seul dimension, prenons l'exemple simple d'un signal sinusoïdal, que l'on notera Ψ_1, représenté dans la Figure 1.1a. Ce signal est limité à l'intervalle $]-\pi, \pi]$ pour former le signal enroulé Φ_1 (Figure 1.1b). Le déroulement de phase consiste alors à retrouver Ψ_1 à partir de Φ_1.

De manière concrète, considérons un signal de phase enroulé Φ_1 discrétisé en N échantillons. Le critère d'Itoh [1] affirme que pour que le déroulement du signal soit exact et sans ambiguïté,



(a) Signal de phase original Ψ_1.



(b) Signal de phase enroulé Φ_1.

FIGURE 1.1 Effet de l'enroulement sur un signal sinusoïdal et unidimensionnel.

il faut que la variation de phase entre deux échantillons consécutifs soit inférieure à π.

c.-à-d.

$$|\Phi_1[n+1] - \Phi_1[n]| < \pi, \quad \forall n \in \{0, 1, \ldots, N-1\}.$$

Algorithm 1 Déroulement de phase en 1D avec le critère d'Itoh

Entrée : Signal de phase enroulée $\Phi_1[n]$, $n \in \{0, 1, \ldots, N-1\}$.

Sortie : Signal de phase original $\Psi_1[n]$.

Initialiser $\Psi_1[0] \leftarrow \Phi_1[0]$

for $n = 0$ à $N-2$ **do**

Calculer la différence enroulée :

$\Delta\Phi_1[n] \leftarrow \Phi_1[n+1] - \Phi_1[n]$.

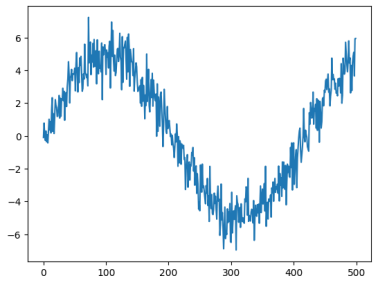
Corriger la discontinuité : $\Delta\Psi_1[n] \leftarrow \Delta\Phi_1[n] - 2\pi \cdot \text{round}\left(\frac{\Delta\Phi_1[n]}{2\pi}\right)$.

Calculer la phase absolue : $\Psi_1[n+1] \leftarrow \Psi_1[n] + \Delta\Psi_1[n]$.

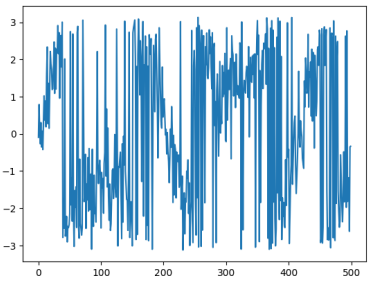
end for

return $\Psi_1[n]$

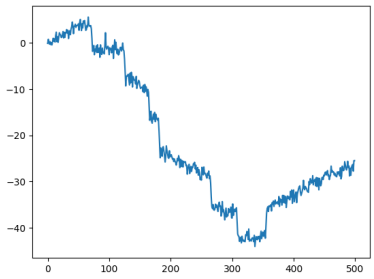
L'algorithme d'Itoh (Algorithme 1) est une méthode exacte de déroulement tant que la condition présentée ci-dessus est respectée. Or, dans les cas réels, la donnée de la phase interférométrique est souvent bruitée, et même sous-échantillonnée. Les Figures 1.2 et 1.3 illustrent respectivement comment le bruit et le sous-échantillonnage affectent le déroulement de phase.



(a) Signal original bruité Ψ'_1.

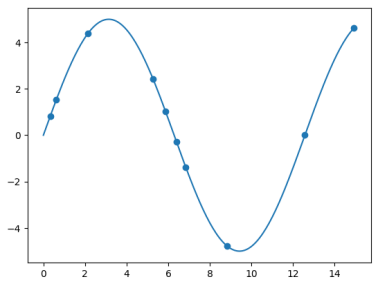


(b) Signal enroulé Φ'_1.

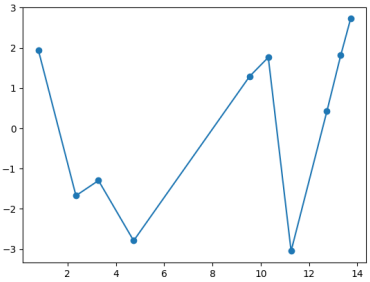


(c) Signal déroulé.

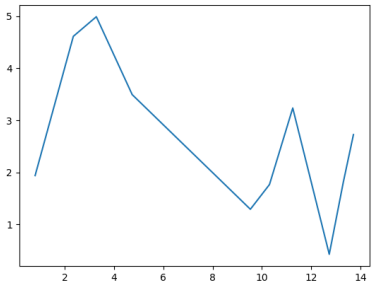
FIGURE 1.2 Les effets du bruits sur le déroulement.



(a) Signal original sous-échantillonné Ψ''_1.



(b) Signal enroulé Φ''_1.



(c) Signal déroulé.

FIGURE 1.3 Les effets du sous-échantillonnage du signal, sur le déroulement.

1.1.2 Extension au déroulement de phase en deux dimensions

Dépendance du chemin :

Pour un signal enroulé en trois dimensions, si le critère d'Itoh est respecté, à savoir que les différences de phase entre points adjacents sont inférieures à π, alors le chemin suivi pour dérouler le signal n'a pas d'importance. Cela garantit que le déroulement est **indépendant du chemin**, et que tous les chemins possibles aboutissent au même résultat.

Cependant, dans les situations réelles, les signaux sont souvent bruités et sous-échantillonnés. Ces deux facteurs peuvent violer le critère d'Itoh et entraîner des ambiguïtés dans le déroulement de phase. Cela conduit à des cas où deux chemins différents pour dérouler le signal peuvent produire des résultats divergents, un phénomène appelé **dépendance au chemin**.

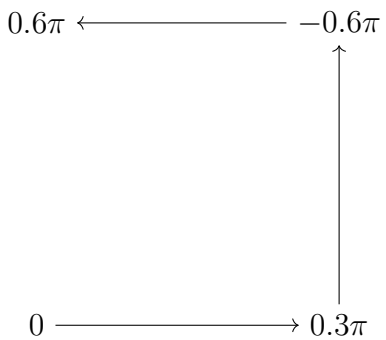
Ce phénomène est illustré dans la Figure 1.4, où deux chemins distincts peuvent être empruntés pour dérouler la phase dans une cellule carrée. Nous examinons deux cas possibles de déroulement de phase à l'intérieur de cette cellule. En appliquant l'algorithme d'Itoh sur les deux chemins, nous obtenons les résultats suivants :

— Pour le **chemin 1**, le déroulement de phase donne :

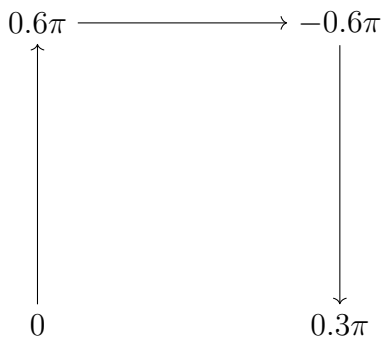
$$[0, 0.6\pi, -0.6\pi, -1.4\pi].$$

— Pour le **chemin 2**, le déroulement de phase donne :

$$[0, 0.6\pi, 1.4\pi, 2.3\pi].$$



(a) Déroulement via Chemin 1.



(b) Déroulement via Chemin 2.

FIGURE 1.4 Comparaison des deux chemins de déroulement dans une cellule carrée.

Les résidus :

Le phénomène de **dépendance de chemin** est ce qui va donner naissance à ce qu'on appelle dans la littérature du déroulement de phase, les **résidus** [2]. L'objectif est de reconstruire la phase absolue $\Psi(x, y)$ en intégrant les différences de phase déroulées $\Delta\Psi$ le long d'un chemin γ reliant un point de référence (x_0, y_0) au point (x, y) :

$$\Psi(x, y) = \Psi(x_0, y_0) + \int_{\gamma} \Delta\Psi \cdot \mathrm{dl}$$

Cependant, la présence de résidus perturbe ce processus. En effet un résidu est défini dans la littérature [3–5] comme une mesure de la discontinuité de la phase enroulée au sein d'une cellule élémentaire. Il est défini par la somme des différences de phase enroulées ($\Delta\Phi$) autour du périmètre de la cellule élémentaire 2×2, normalisée par 2π :

$$C = \frac{1}{2\pi} \sum_{\text{périmètre de la cellule}} \Delta\Phi.$$

Le résidu prend la valeur $+1$ (positif) si $C = +1$, et la valeur -1 (négatif) si $C = -1$ [3].

Mathématiquement, cela signifie que le champ de gradient de phase n'est pas conservatif. Pour un champ conservatif, l'intégrale du gradient de phase le long d'un chemin fermé est nulle :

$$\oint_{\mathcal{C}} \nabla\Psi \cdot \mathrm{dl} = 0$$

Or, si le chemin entoure un résidu, cette intégrale devient non nulle :

$$\oint_{\mathcal{C}} \nabla\Psi \cdot \mathrm{dl} = \pm 2\pi.$$

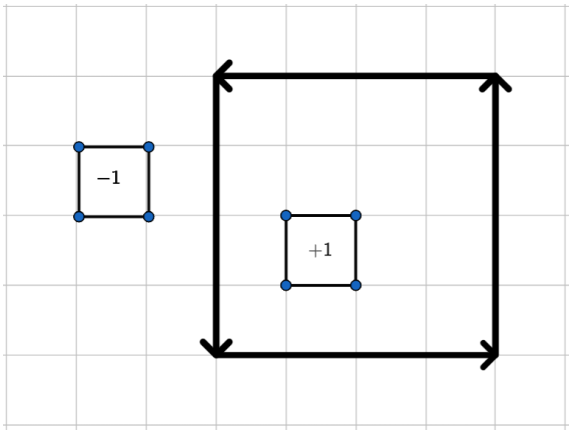
Dans l'exemple illustré à la Figure 1.5, le premier chemin fermé $\mathcal{C}_1$ entoure un résidu de valeur 2π. Cela implique que l'intégrale le long de ce chemin vérifie :

$$\oint_{\mathcal{C}_1} \nabla\Psi \cdot \mathrm{d}\mathbf{l} = +2\pi.$$

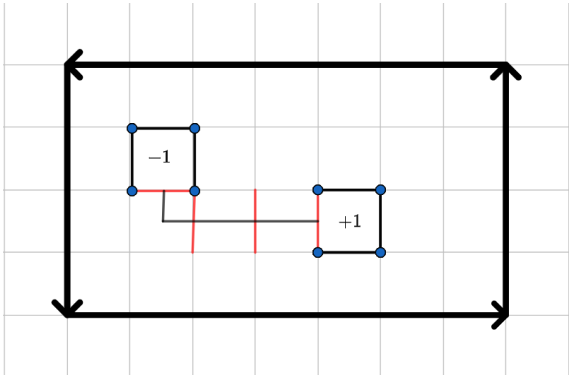
En revanche, pour le second chemin fermé $\mathcal{C}_2$, l'intégrale est nulle, car il inclut à la fois un résidu positif et un résidu négatif. On a donc :

$$\oint_{\mathcal{C}_2} \nabla\Psi \cdot \mathrm{d}\mathbf{l} = +2\pi - 2\pi = 0.$$

Ainsi, pour garantir que le problème de dépendance au chemin soit résolu en deux dimensions



(a) Exemple de chemin fermé C_1.



(b) Exemple de chemin fermé C_2.

FIGURE 1.5 Exemple de deux chemins fermées entourant des résidus.

(2D), il est nécessaire que chaque chemin d'intégration fermé entoure un nombre équilibré de résidus positifs et négatifs [6, 7].

Pour répondre à cette contrainte, on introduit des *branch cuts*, qui sont des lignes reliant les résidus positifs et négatifs de manière à équilibrer leurs contributions. Ces coupes de branche empêchent les chemins d'intégration de traverser les zones problématiques où la phase est incohérente.

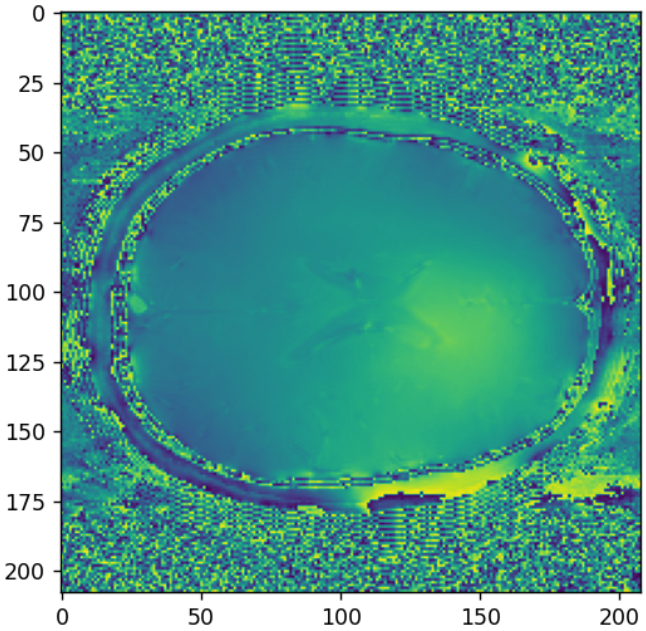
Une fois les *branch cuts* établies, tous les chemins d'intégration dans le signal produisent la même valeur de phase déroulée, quel que soit leur tracé. La littérature propose plusieurs méthodes pour relier efficacement les résidus, tout en minimisant les erreurs introduites dans le signal déroulé. Ces techniques visent à optimiser la disposition des *branch cuts* pour garantir une reconstruction précise et cohérente du champ de phase, qui va permettre de minimiser la métrique d'erreur L_p [7].

1.1.3 Le défi du déroulement de phase en trois dimensions :

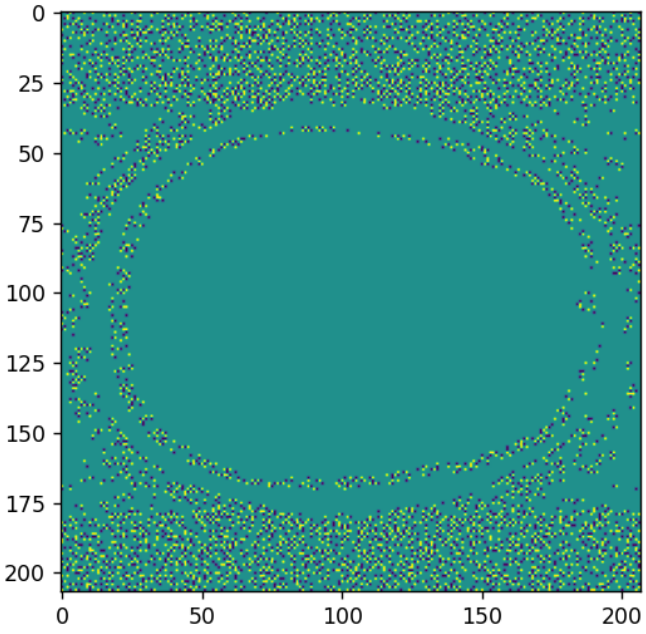
Dans la dimension $3D$, l'approche reste similaire à celle en $2D$. Pour le cas en trois dimensions, l'objectif est de reconstruire la phase absolue $\Psi(x, y, z)$ en intégrant les différences de phase déroulées $\Delta\Psi$ le long d'un chemin γ reliant un point de référence (x_0, y_0, z_0) au point (x, y, z) :

$$\Psi(x, y, z) = \Psi(x_0, y_0, z_0) + \int_{\gamma} \Delta\Psi \cdot \mathrm{d}\mathbf{l}'.$$

Ici, l'intégration est effectuée dans l'espace tridimensionnel, et $\mathrm{d}\mathbf{l}'$ représente un vecteur infi-



(a) Phase enroulée en deux dimension.



(b) les résidus positifs en jaune, et les résidues négatifs en bleu .

FIGURE 1.6 Exemples de résidus pour une phase enroulée en $2D$.

nitésimal le long du chemin γ, prenant en compte les variations de phase dans les directions x, y, et z.

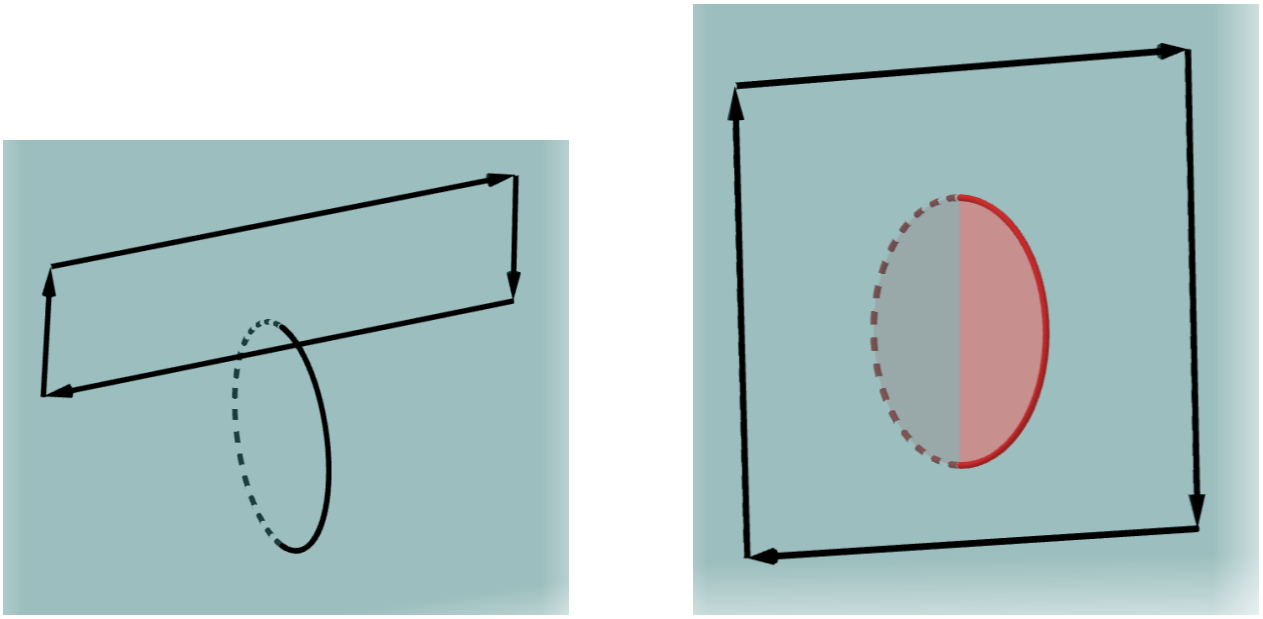
Et comme pour la dimension $2D$, **plusieurs chemins** sont possibles. Il est donc nécessaire que chaque chemin fermé dans le signal $3D$ entoure un nombre équilibré de résidus positifs et négatifs. Cependant, en $3D$, les résidus sont plus denses dans l'espace et peuvent apparaître sur les plans XY, YZ, et XZ.

Pour gérer cette complexité, les **boucles de singularités** ont été introduites dans le problème de déroulement en $3D$ [8]. Ces boucles relient les résidus de manière à ce que, si un chemin d'intégration fermé traverse une boucle, son intégrale ne soit plus nulle, ce qui introduit une **dépendance au chemin** [8, 9].

À titre d'exemple, la Figure 1.7 illustre ce phénomène. Le chemin fermé P_1 (Figure 1.7a) traverse l'intérieur d'la boucle de singularité, ce qui entraîne une intégrale non nulle $\oint_{\mathcal{P}_1} \nabla\Psi \cdot \mathrm{d}\mathbf{l} \neq 0$. En revanche, le chemin fermé P_2 (Figure 1.7b) contourne la boucle, ce qui garantit une intégrale nulle $\oint_{\mathcal{P}_2} \nabla\Psi \cdot \mathrm{d}\mathbf{l} = 0$.

En effet, pour garantir une intégration cohérente, il est nécessaire de bloquer tous les passages à travers les boucles de singularités en $3D$. Cela consiste à remplir ces boucles, empêchant tout chemin fermé de les traverser. Si toutes les boucles de singularités sont entièrement neutralisées et qu'aucun passage à travers elles n'est possible, alors chaque chemin d'intégration fermé en $3D$ sera nul. Par conséquent, la dépendance au chemin sera éliminée [9–11].

Il est important de mentionner qu'en trois dimensions, les surfaces utilisées pour fermer les boucles de singularités ne sont pas uniques. De chaque côté des surfaces de fermeture, le gradient déroulé peut s'écarter du gradient original du signal lorsqu'un gradient traverse ces surfaces. Par conséquent, la surface totale des régions de fermeture représente une borne supérieure sur l'erreur L_p décrite dans la section précédente. **Minimiser l'aire de ces surfaces** est donc essentiel pour réduire l'erreur globale lors du déroulement de phase.



(a) Exemple de chemin fermé P_1.

(b) Exemple de chemin fermé P_2.

FIGURE 1.7 Exemples de deux chemins fermés en trois dimensions.

1.1.4 Applications du déroulement de phase

Le déroulement de phase trouve des applications dans divers domaines :

- **Interférométrie radar** (InSAR), pour mesurer la topographie et les déformations de la surface terrestre [12, 13].
- **Imagerie médicale**, comme en résonance magnétique nucléaire (IRM) pour corriger les artefacts de phase [14, 15].
- **Optique adaptative**, pour analyser les fronts d'onde et améliorer la résolution optique [16, 17].
- **Tomographie par cohérence optique** (OCT), pour reconstruire des images en profondeur [16, 18]

1.2 Éléments de la problématique

Les défis de la problématique : Le déroulement de signal en trois dimensions (3D) constitue une problématique complexe qui dépasse largement le cadre des méthodes classiques développées pour les signaux en deux dimensions (2D). En effet, bien que les approches en 2D soient bien établies et largement utilisées, leur transposition à la 3D reste un véritable défi. Les signaux 3D présentent des propriétés géométriques et topologiques plus complexes,

notamment des singularités qui ne peuvent plus être réduites à de simples points mais qui prennent des formes plus élaborées, comme des boucles [8, 10]. Ces singularités rendent nécessaire une redéfinition complète des méthodes existantes, tant sur le plan conceptuel que pratique. Comment ces singularités doivent-elles être identifiées, et de quelle manière leur correction peut-elle être intégrée dans un processus algorithmique efficace, restent des questions centrales de cette recherche.

Un autre défi majeur réside dans la construction des chemins d'intégration permettant de reconstruire le signal initial sans compromettre sa cohérence. Dans le contexte 3D, ces chemins ne doivent pas traverser les zones marquées par des résidus, sous peine d'introduire des erreurs dans le signal reconstruit. Cependant, définir ces chemins d'intégration de manière robuste, en garantissant qu'ils contournent correctement les singularités, implique de concevoir des structures géométriques qui enveloppent ces zones critiques. Ces structures, appelées 'surfaces d'erreur', doivent être non seulement définies de manière rigoureuse mais également **optimisées** afin de minimiser leur coût computationnel tout en préservant l'intégrité du signal.

Enfin, il convient de souligner que ces défis sont exacerbés par les limitations des approches méthodologiques disponibles. Les méthodes classiques de déroulement en 2D, basées par exemple sur les *branch-cuts* [7], ont démontré leur efficacité pour les signaux bidimensionnels, mais leur extension directe à la 3D s'avère inadéquate. Dans la 3D, les singularités ne peuvent plus être simplement reliées par des lignes, mais nécessitent la définition de surfaces fermantes, un concept à la fois plus complexe et plus coûteux à mettre en œuvre. Les approches globales, qui tentent de minimiser les erreurs sur l'ensemble du signal [19], souffrent de limitations importantes en termes de complexité algorithmique et de sensibilité au bruit. De même, les méthodes régionales, qui progressent localement à travers le signal, ont du mal à gérer des volumes de données vastes et bruités, et leur efficacité dépend fortement de choix de paramètres souvent arbitraires.

Les limites des approches existantes : Les approches contemporaines présentent des faiblesses qui limitent leur applicabilité au déroulement de signaux 3D. Les méthodes globales, par exemple, reposent sur des calculs intensifs pour résoudre des problèmes d'optimisation complexes, mais elles sont souvent impraticables pour les grands volumes de données générés par les applications modernes, telles que l'imagerie médicale ou l'interférométrie radar [12, 13, 20]. Leur sensibilité aux discontinuités abruptes et au bruit aléatoire limite leur capacité à produire des résultats cohérents dans des environnements réels [16, 21, 22]. À l'opposé, les approches régionales, qui segmentent le signal en parties plus petites pour simplifier le processus, sont souvent incapables d'assurer une cohérence globale et risquent de produire

des résultats divergents lorsque les données sont fortement bruitées [23–25].

Par ailleurs, les méthodes récentes basées sur l'apprentissage profond, bien qu'innovantes, posent d'autres problèmes. Ces approches nécessitent des ensembles de données d'entraînement massifs et variés, et leur caractère de 'boîte noire' rend difficile leur interprétation [18, 20, 26, 27]. Cette opacité est particulièrement problématique dans des domaines critiques tels que la médecine [14, 15, 22] ou la géophysique [28, 29], où la transparence des décisions algorithmiques est essentielle. De plus, ces modèles se montrent souvent inadéquats lorsqu'ils sont confrontés à des données présentant des anomalies ou des distributions atypiques [8, 10, 30].

Ainsi, la combinaison de ces limitations souligne l'urgence de repenser les fondements mêmes des approches existantes pour les adapter aux exigences spécifiques du déroulement en 3D. Cela implique non seulement de surmonter les barrières techniques actuelles mais aussi de proposer des solutions qui soient robustes, efficaces et applicables à des scénarios concrets.

Une approche nécessairement innovante

Dans ce contexte, cette recherche vise à offrir une réponse nouvelle et complète à ces défis. Elle se concentre sur la redéfinition des concepts clés, tels que les résidus et les boucles de singularité, afin de les adapter aux spécificités de la 3D. L'objectif est également de développer des méthodes algorithmiques permettant de détecter et de corriger ces singularités de manière efficace, tout en garantissant que les chemins d'intégration restent cohérents. Enfin, en cherchant à minimiser les surfaces d'erreur à travers des approches géométriques et algorithmiques optimisées, cette recherche vise à établir une base théorique et pratique robuste pour répondre aux besoins de la 3D.

En définitive, cette recherche se positionne comme une tentative de surmonter les limites des approches actuelles en explorant des solutions innovantes, adaptées à la complexité et aux exigences uniques des signaux tridimensionnels.

Objectifs de la recherche

Les objectifs de recherche de ce mémoire sont de développer une méthode optimale pour le déroulement d'un signal d'interférométrie en trois dimensions (3D), en s'appuyant sur la construction et l'exploitation des boucles de singularités.

Une contribution clé de ce mémoire réside dans l'élaboration d'une solution indépendante du type de signal analysé. Qu'il s'agisse de données issues de l'Imagerie par Résonance Magnétique (IRM), de l'Interférométrie Radar à Synthèse d'Ouverture (InSAR) ou d'autres contextes, la méthode proposée se veut généralisable et adaptable. Pour atteindre cet objec-

tif, nous introduirons un cadre mathématique rigoureux pour la construction des boucles de singularité, en exploitant des outils avancés issus de la théorie des graphes.

Enfin, pour la première fois dans la littérature, ce travail proposera une optimisation des surfaces d'erreur en utilisant une **approche exacte**. Cette contribution permettra de minimiser ces surfaces de manière optimale, garantissant ainsi une solution théorique et pratique précise pour le problème du déroulement en 3D. Cette méthodologie est en effet bien ambitieuse, mais elle est nécessaire pour répondre aux défis posés par le déroulement en 3D et pour proposer une solution universelle et robuste à ce problème complexe.

1.3 Plan du mémoire

Le plan du mémoire sera donc le suivant :

1. Revue de la littérature sur les déroulement de signal $2D$ et $3D$.
2. Présentation des concepts de résidus et de boucles de singularités en $3D$, ainsi que les algorithmes de construction de ces boucles, avec une analyse de complexité.
3. Construction et minimisation des surfaces d'erreur en $3D$, en utilisant un programme linéaire en nombres entiers.
4. Résultats de la méthode sur les données synthétiques et critères d'évaluation.

CHAPITRE 2 REVUE DE LITTÉRATURE

Le déroulement de phase (*phase unwrapping*) est une étape essentielle dans de nombreuses applications d'imagerie telles que l'interférométrie radar à synthèse d'ouverture (InSAR), l'imagerie par résonance magnétique (IRM), la tomographie par cohérence optique (OCT) et l'holographie. Il s'agit de reconstruire la phase absolue d'un signal à partir de mesures limitées dans l'intervalle $[-\pi, \pi]$. Cependant, ce processus est complexe en raison des discontinuités, du bruit et des singularités présentes dans les données mesurées.

Dans cette section, nous présentons une revue détaillée des méthodes existantes pour le déroulement de phase en deux et trois dimensions, en mettant l'accent sur les défis associés et les solutions proposées dans la littérature.

2.1 Le déroulement de phase en deux dimensions

Méthodes basées sur les coupes de branche (*branch-cuts*)

La méthode des *branch-cuts* est parmi les premières méthodes de déroulement de phase à être proposées pour les signaux d'interférométrie en $2D$. Des chercheurs de l'Institut de technologie de Californie ont proposé en 1988 cette méthode qui commence par identifier les résidus de phase [3]. En effet, ces erreurs locales sont détectées de manière assez simple. La somme des différences de phase autour de ces pixels n'est pas nulle, ce qui indique la présence d'une singularité. Les résidus positifs et négatifs sont donc regroupés par des lignes que l'on appellera *branch-cuts*, formant ainsi des barrières qui empêchent l'intégration autour de ces singularités et ainsi éviter la propagation de l'erreur tout au long du signal reconstruit.

Cette méthode reste toujours parmi les plus utilisées dans le déroulement de phase, vu sa robustesse face au bruit dans le signal [7,31,32]. Le plus grand défi est de trouver les *branch-cuts* de manière optimale, ce qui nécessite de recourir à des heuristiques non triviales pour les identifier, ce qui peut accroître la complexité de l'algorithme de déroulement de phase. Des méthodes de détection optimales des *branch-cuts* ont été proposées, par exemple celle basée sur l'utilisation des forêts couvrantes équilibrées [7,33]. Afin de procéder à cette détection, les auteurs cherchent à minimiser les discontinuités dans le signal, en structurant les connexions entre les résidus de phase de manière à former des forêts équilibrées, ce qui permet de limiter les erreurs introduites par les singularités de phase. Leur méthode s'est en effet avérée bien efficace et peu sensible au bruit. Cependant, elle reste coûteuse en termes de calculs, ce qui la rend difficile à appliquer [33]. Trouver des forêts couvrantes équilibrées est un problème NP-

difficile, semblable à un problème généralisé de Steiner [34]. Donc la plus grande limitation de la recherche des *branch-cuts* est la complexité de l'algorithme de détection, qui nécessite généralement des heuristiques assez compliquées pour résoudre des instances du problème de grande taille et avec beaucoup de bruit .

Méthodes de minimisation globale :

Les méthodes de minimisation globale visent à reformuler le problème de déroulement de phase en une optimisation globale, ce qui permet de surmonter les difficultés causées par les discontinuités et le bruit dans le signal. Elles sont basées sur la minimisation de la norme L_p des différences entre les gradients de la phase déroulée et ceux de la phase mesurée (voir par exemple [35]). L'objectif est donc de trouver la phase déroulée qui minimise la valeur de l'erreur L_p présentée dans la section précédente. Les normes L_0 [7, 30, 36] et L_1 [30, 37] sont fréquemment utilisées. Le principal avantage de ces méthodes réside dans leur robustesse face au bruit et aux erreurs aléatoires [30, 36]. Cependant, en considérant l'image globalement, ces méthodes peuvent être sensibles aux erreurs locales.

Ce type de méthode propose des algorithmes avec des exigences computationnelles assez élevées, ce qui les rend difficiles à appliquer sur des données de grande taille [36]. Cependant, des travaux récents ont proposé des améliorations pour rendre ces méthodes plus efficaces [30]. Une étude publiée en 2017 a proposé le déroulement de phase pour des images de grande dimension, notamment dans le cadre de l'interférométrie radar à synthèse d'ouverture (InSAR), en utilisant également les normes L_0 et L_1 pour l'optimisation. Afin de mieux gérer les contraintes liées aux grandes images, une stratégie de division en plusieurs tuiles a été proposée, chaque région est traitée de manière locale, puis les résultats sont fusionnés pour obtenir une solution globale. Cette approche a permis de réduire la complexité de l'algorithme et de le rendre plus adapté aux données de grande taille.

Néanmoins, au-delà de la complexité de l'algorithme, ces méthodes peuvent être sensibles aux discontinuités de phase très abruptes et prononcées [30, 35]. Les méthodes basées sur la norme L_2 ont tendance à lisser les discontinuités, ce qui peut entraîner une perte de précision dans le déroulement.

Méthodes basées sur l'apprentissage profond :

L'avancée des techniques de l'apprentissage profond dans le domaines de vision par ordinateur et de traitement d'image a permis de proposer des méthodes de déroulement de phase en 2D basées sur l'apprentissage profond [20]. Ces méthodes exploitent la capacité des réseaux de neurones convolutifs *Convolutional Neural Networks (CNN)* [38] à apprendre des représentations hiérarchiques des données à partir de données volumineuses [26, 27].

L'un des travaux de recherche approche le problème en utilisant une fonction de cartographie directe entre des données à partir des phases enroulées vers les phases déroulées [19]. Ceci est rendu possible grâce à la grande taille de l'ensemble de données d'entraînement.

Ces méthodes ont normalement l'avantage de ne pas nécessiter de connaissances a priori sur les données. Cependant, elles nécessitent des ensembles de données variés et de grande taille pour que l'entrainement soit efficace. Ainsi, l'apprentissage automatique peut résulter en des modèles qui ne reconnaissent pas les discontinuités abruptes dans le signal, ou qui sont extrêmement sensibles au niveaux de bruit élevés.

De plus, l'effet boîte noire de ces méthodes ainsi que la difficulté de les interpréter, les rendent moins populaires dans les applications où la transparence et l'explicabilité des résultats sont essentielles. Les applications sensibles du déroulement, comme la médecine [14, 15] et la géophysique [28,29], nécessitent souvent des méthodes plus explicites et interprétables. Cette opacité des modèles d'apprentissage profond reste encore un obstacle majeur.

2.2 Le déroulement de phase en trois dimensions :

Le passage à la troisième dimension ajoute des défis supplémentaires au déroulement de phase, en raison de la complexité topologique accrue et du volume de données à traiter. Les singularités de phase se manifestent sous forme de boucles de résidus plutôt que de points isolés [8–10]. La gestion de ces boucles est cruciale pour éviter les erreurs globales dans le déroulement. Dans cette section, nous allons examiner les méthodes existantes pour le déroulement de phase en trois dimensions, en mettant l'accent sur leurs principes, mais aussi sur leurs avantages et leurs inconvénients.

Méthodes basées sur les *branch-cuts* en 3D :

La méthode des *branch-cuts* est assez bien établie en $2D$. Cependant, en $3D$, les singularités de phase ont des formes plus complexes, ce qui rend la détection des *branch-cuts* plus difficile [8, 10]. En effet, les singularités de phase en 3D se manifestent sous forme de boucles de résidus, ce qui nécessite des approches plus sophistiquées pour les détecter et ensuite les prendre en considération lors de la reconstruction de la phase.

Des chercheurs ont proposé une extension de la méthode des *branch-cuts* en 3D, en introduisant des surfaces fermantes qui ferment les boucles de singularités [8]. Le principe fondamental de la méthode de Huntley est de détecter les boucles de singularité, et ensuite de placer des surfaces fermantes autour de ces boucles pour empêcher la propagation de l'erreur. Les principales étapes de cette méthode sont :

1. **Détection des résidus :** Les résidus de phase sont identifiés en calculant la somme des différences de phase autour de chaque voxel. Les résidus positifs et négatifs sont regroupés pour former des boucles de singularités.
2. **Placement des surfaces fermantes :** les auteurs proposent de placer des boucles ombrées autour des boucles de singularité. Ces boucles-ci vont rétrécir sur elles-mêmes pour jusqu'à ce que leur aire soit nulle. Lors de processus, les boucles ombrées vont balayer ce que les auteurs vont considérer comme la surface *branch-cut* liée à la boucle de résidu. Les boucles ombrées seraient donc similaires à des élastiques qui se rétractent.
3. **Déroulement de phase :** La phase est déroulée en intégrant les différences de phase sur le réseau 3D, en évitant de traverser les surfaces fermantes.

Cette approche identifie les boucles de singularité en détectant des « coins » au sein des boucles, souvent en forme de « L » ou de « U » [10]. Ces coins sont ensuite comblés, renforçant ainsi la robustesse de la méthode initiale. Le remplissage des boucles utilise une technique de « boucles ombrées », ce qui facilite la gestion des singularités complexes [10]. Une des principales différences de cette méthode réside dans l'ajout de poids aux arêtes du réseau [10]. Au lieu d'attribuer systématiquement des poids nuls aux arêtes traversées par les coupures de branchement, cette approche applique des poids basés sur la qualité des données. Ainsi, les arêtes sont pondérées en fonction de leur fiabilité, permettant à l'algorithme de privilégier les chemins les plus fiables lors du déroulement de phase [10].

Méthodes basées sur le lissage de Fourier en 3D :

Un des auteurs qui ont poussé le domaine du déroulement de phase en 3D a présenté sa thèse de doctorat en 2007 [39], sur l'utilisation des transformées de Fourier pour le déroulement de phase en 3D. Les méthodes utilisant les transformées de Fourier abordent le problème de déroulement sous un angle différent. Ces méthodes ne cherchent pas forcément à détecter ni à éviter les zones de singularité lors de la reconstruction de la phase. La profilométrie par transformée de Fourier (FTP) est une méthode de déroulement de phase qui en projetant un motif de franges sinusoïdales sur la surface de l'objet et en capturant l'image déformée par une caméra [40]. La déformation de ces franges donne des informations sur la topographie de la surface. En appliquant donc une transformée de Fourier [41], le problème est donc passé de sa formulation spatiale à une formulation fréquentielle, ce qui permet de simplifier le processus de calcul.

Cependant, le plus grand désavantage des méthodes basées sur la profilométrie par transformée de Fourier est leur sensibilité au bruit, et aussi leur incapacité à gérer les discontinuités de phase abruptes. La profilométrie est efficace lorsque la donnée en question est lisse et régulière, mais elle peut échouer dans des cas plus complexes.

Méthodes d'expansion de région *Region expansion* en 3D :

Les méthodes d'expansion de régions en trois dimensions sont basées sur le principe de croissance régionale pour résoudre le problème du déroulement de phase. Au lieu de traiter chaque point individuellement ou de minimiser globalement sur tout le volume, ces approches identifient des zones cohérentes où la phase est fiable, puis étendent progressivement le déroulement de phase à travers ces régions, en se basant sur la continuité locale de la phase. Cela permet d'adapter le déroulement aux variations locales et de limiter l'impact du bruit et des discontinuités.

En 2002, Cusack a proposé une méthode d'expansion de régions qui minimise l'écart entre les gradients de la phase déroulée et ceux de la phase mesurée [42]. Cette approche débute par analyser les niveaux de bruit dans différentes régions de la phase, puis privilégie les zones les moins bruitées lors de l'intégration, afin de minimiser l'erreur globale. Cela permet de mieux gérer la propagation des erreurs en se concentrant sur les zones les plus fiables.

La méthode *PRELUDE* (*Phase Region Expanding Labeller for Unwrapping Discrete Estimates*) proposée par Jenkinson [22] est également une technique d'expansion de régions en trois dimensions. Elle démarre à partir de voxels fiables et étend le déroulement de phase aux voxels adjacents, en tenant compte de la continuité de la phase. *PRELUDE* est spécialement conçue pour les données d'IRM, prenant en compte les caractéristiques spécifiques de ces images. Cette méthode arrive à bien gérer les discontinuités, tout en étant peu sensible au bruit. Cependant, pour des données volumineuses, *PRELUDE* peut être lente, avec des temps de calcul pouvant atteindre plusieurs heures.

Pour améliorer l'efficacité, Karsa et Shmueli ont introduit la méthode *SEGUE* (*Speedy Region Growing Algorithm for Unwrapping Estimated Phase*) [43]. *SEGUE* (*Speedy Region Growing Algorithm for Unwrapping Estimated Phase*) améliore la méthode *PRELUDE* en utilisant un mécanisme de tri dynamique basé sur une file de priorité. Cette approche traite en priorité les voxels les plus fiables, en ajustant dynamiquement leur ordre selon des critères comme la fiabilité des gradients ou le rapport signal/bruit. Contrairement à *PRELUDE*, qui suit un ordre statique, *SEGUE* s'adapte à la qualité locale des données, réduisant ainsi la propagation des erreurs. De plus, *SEGUE* exploite le traitement parallèle pour unwrap plusieurs régions simultanément, ce qui diminue considérablement le temps de calcul. Ces améliorations rendent *SEGUE* bien plus rapide et efficace que *PRELUDE*, particulièrement pour les ensembles de données volumineux ou bruités.

Cheng *et al.* [23] ont proposé une nouvelle méthode d'expansion de régions en trois dimensions pour le déroulement de phase dans la cartographie de susceptibilité magnétique abdominale. Leur approche combine le partitionnement de régions et la modélisation polynomiale locale

pour gérer efficacement le bruit élevé et les changements rapides de phase. Les zones de phase sont initialement partitionnées en régions cohérentes, en excluant les voxels bruyants. Ensuite, la phase est déroulée séquentiellement en utilisant un modèle polynomial local, améliorant ainsi la précision du déroulement de phase dans des conditions difficiles.

Xu et Cumming [25] ont développé une méthode d'expansion de régions pour le déroulement de phase en interférométrie SAR. Leur algorithme identifie des régions de haute cohérence et les fait croître en vérifiant la fiabilité des connexions entre les pixels, ce qui permet de gérer efficacement les zones de faible cohérence.

Arevalillo-Herráez *et al.* [44] ont présenté un algorithme robuste de déroulement de phase en trois dimensions basé sur le *clustering*. Leur méthode identifie des régions cohérentes en regroupant les voxels avec des gradients de phase similaires, puis déroule la phase à l'intérieur de chaque cluster. Cela améliore la gestion des discontinuités et du bruit en traitant localement les variations de phase.

Enfin, la méthode *ROMEO* (*Rapid, Optimal Multi-Echo* phase unwrapping) proposée par Dymerska *et al.* [45] est une contribution significative dans ce domaine. *ROMEO* est spécialement conçue pour le déroulement de phase en trois dimensions dans les environnements d'imagerie par résonance magnétique (IRM). Elle intègre les propriétés physiques de l'IRM pour améliorer la précision et l'efficacité du déroulement.

La méthode *ROMEO* utilise une approche d'expansion de régions en attribuant des poids de « qualité » aux arêtes du graphe représentant la phase en 3D. Ces poids sont calculés en tenant compte de la cohérence spatiale, temporelle et de magnitude du signal, permettant à l'algorithme de privilégier les chemins les plus fiables lors du déroulement.

Pour la **cohérence spatiale**, le poids entre deux voxels adjacents i et j est calculé comme suit :

$$W_{(i,j)}^{\text{Spatiale}} = 1 - \left| \frac{\Omega(\Psi_i - \Psi_j)}{\pi} \right|,$$

où Ω est la fonction d'enroulement et Ψ_i la phase du voxel i. Ce poids est proche de 1 si les phases des voxels i et j sont similaires, indiquant une forte cohérence spatiale.

La **cohérence de magnitude** est évaluée par :

$$W_{(i,j)}^{\text{Magnitude}} = \left(\frac{\min(M_i, M_j)}{\max(M_i, M_j)} \right)^2,$$

où M_i est la magnitude du voxel i. Ce poids est proche de 1 si les magnitudes des voxels i et

j sont similaires, reflétant une forte cohérence de magnitude.

Les poids finaux attribués aux arêtes sont le produit des poids de cohérence spatiale et de magnitude, créant ainsi une matrice de poids de qualité Q. Pour réduire la mémoire et améliorer l'efficacité computationnelle, la matrice Q est transformée en une matrice de coûts C entiers entre 1 et 255 :

$$C = \max\left(\lfloor 255 \cdot (1 - Q) \rfloor, 1\right).$$

L'algorithme de *ROMEO* utilise ensuite l'algorithme de Prim [46] pour trouver l'arbre couvrant minimal (MST) basé sur les coûts C. La complexité de cet algorithme est $O(|E| \log |V|)$, où $|E|$ est le nombre d'arêtes et $|V|$ le nombre de nœuds (voxels), ce qui rend l'algorithme extrêmement efficace en termes de temps de calcul.

CHAPITRE 3 CRÉATION DES BOUCLES DE SINGULARITÉ

3.1 Concepts Élémentaires :

Le problème de déroulement de signal en 3D est un problème qui vise à reconstruire la phase d'un signal tridimensionnel à partir de ses valeurs de phase enroulées. Afin de formaliser ce problème de manière mathématiquement claire, nous associerons à chaque phase tridimensionnelle enroulée Φ deux graphes fondamentaux :

1. Un graphe de grille $G_{grid}(\Phi)$ qui représente la structure discrète du domaine tridimensionnel où le signal est défini. Ce graphe servira de support pour l'intégration des chemins lors du déroulement de phase, et il jouera un rôle crucial dans la formulation des contraintes qui interviennent dans les méthodes de résolution que nous développerons ultérieurement dans le chapitre 4.
2. Un graphe de faces $G_{faces}(\Phi)$ qui est construit à partir des résidus de phase associés à chaque face du graphe de grille $G_{grid}(\Phi)$. Ce graphe sera le support que l'on utilisera pour la construction des boucles de singularité, que l'on abordera dans la suite de ce chapitre.

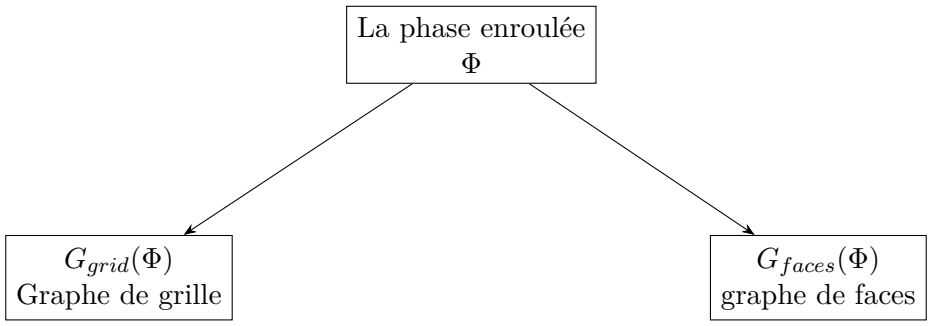


FIGURE 3.1 Schéma arborescent avec concepts associés.

Ces graphes sont les bases qui vont permettre de structurer le problème de déroulement d'une phase enroulée Φ en $3D$. Les définitions précises de ces graphes, ainsi que des concepts de **résidu**, **face** et **boucle de singularité** seront introduites par la suite pour clarifier leur rôle dans la résolution du problème du déroulement de phase en $3D$.

3.1.1 Graphe de grille :

Definition 3.1 (Graphe de grille)**.** *Pour une phase tridimensionnelle enroulée* Φ *de taille* $N \times L \times M$*, on associe le graphe de grille* $G_{grid}(\Phi) = (V_{grid}(\Phi), E_{grid}(\Phi))$ *où*

1. $V_{grid}(\Phi) := \{(x, y, z) \in \mathbb{N}^3 \ / \ x < N, y < L, z < M\}$ *est l'ensemble des sommets de la grille.*
2. $\forall (x, y, z), (x', y', z') \in V_{grid}(\Phi)$*,* $((x, y, z), (x', y', z')) \in E_{grid}(\Phi)$ *si et seulement si* $|x - x'| + |y - y'| + |z - z'| = 1$*.*

Après avoir introduit le graphe de grille, on introduit quelques sous concepts mathématiques nécessaires pour la suite.

Definition 3.2 (Face)**.** *Une face* F *dans le graphe de grille* $G_{grid}(\Phi)$ *est un cycle quadrilatéral constitué de quatre sommets connectés en une boucle, c'est-à-dire un sous-graphe* $F = (V_F, E_F)$ *tel que :*

— $|V_F| = 4$ *et* $|E_F| = 4$*,*

— *Chaque sommet* $v \in V_F$ *a un degré* $\deg_F(v) = 2$ *dans* F*,*

— *Les arêtes forment une séquence fermée connectant les sommets en ordre cyclique.*

Le barycentre de F *est le point* $\mathbf{c}_F = \dfrac{1}{4} \sum_{v \in V_F} v$*.*

Par exemple, sur le plan $x = i$, les sommets (i, j, k), $(i, j+1, k)$, $(i, j+1, k+1)$ et $(i, j, k+1)$ forment une face, et son barycentre est $(i, j + 0.5, k + 0.5)$.

Definition 3.3 (Cube)**.** *Un cube* C *dans un graphe de grille* $G_{grid}(\Phi)$ *est un sous-graphe de* $G_{grid}(\Phi)$*, que l'on notera* $C = (V_C, E_C)$ *tel que :*

$$\forall v \in E_C : deg(v) = 3$$

.

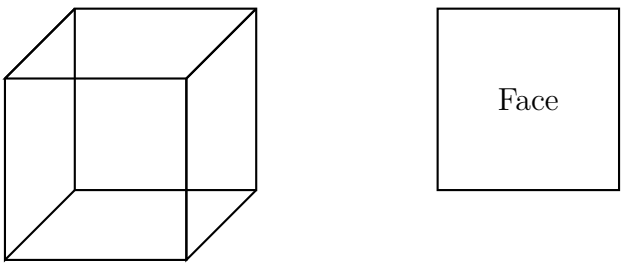


FIGURE 3.2 Exemple de cube et de face.

Pour un cube $C = (E_C, V_C)$, on dit que la face $F = (V_F, E_F)$ est une face de C, si F est un sous-graphe de C.

Definition 3.4 (Sens canonique de la face par rapport à un cube)**.** *Pour chaque face F d'un cube C, la normale de F est définie comme pointant* ***vers l'extérieur du cube*** *C. Le sens de parcours des sommets de la face est défini par la* ***règle de la main droite*** *[8] :*

— *Si le pouce de la main droite pointe dans la direction de la normale à F, alors les doigts suivent le sens de parcours des sommets.*

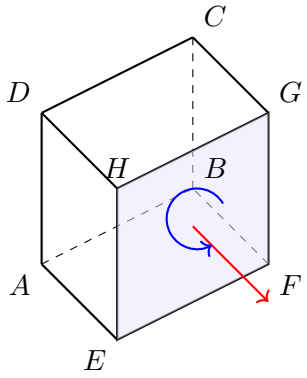


FIGURE 3.3 Cube vu en perspective avec une face surlignée, une normale sortante et le sens canonique défini par la règle de la main droite.

La Figure 3.3 illustre le sens canonique du parcours de la face $EFGH$ par rapport au cube $ABCDEFGH$. La normale de la face $EFGH$ pointe vers l'extérieur du cube, et donc le sens canonique de parcours de ses sommets selon la Définition 3.4 est $E \to F \to G \to H \to E$. Il est important de noter que le sens canonique pour une face donnée dépend du cube que l'on choisit comme référence. Pour l'exemple illustré dans la Figure 3.3, le sens canonique de la face $HCBG$ par rapport au cube $DFEACHGB$ est $C \to H \to G \to B \to C$, et son sens canonique par rapport au cube $ABCDEFGH$ est $H \to C \to B \to G \to H$.

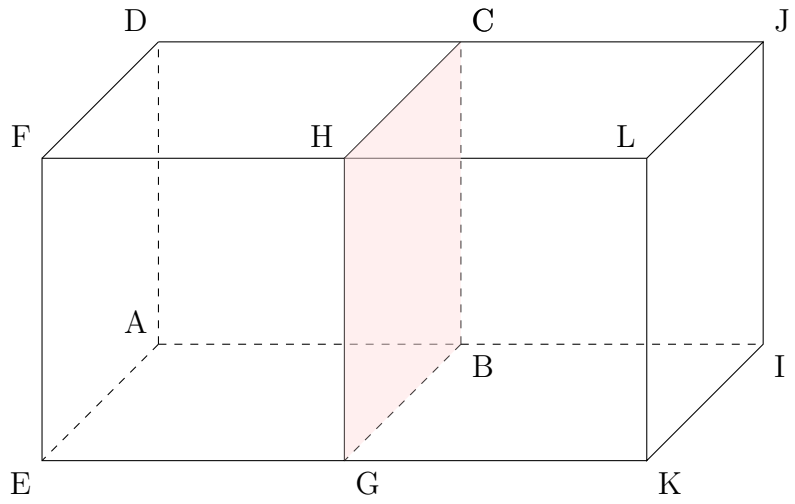


FIGURE 3.4 Exemple de deux cubes adjacents partageant une face.

Definition 3.5 (Face extérieure)**.** *Une face F est dite extérieure s'il existe un unique cube C tel que F est un face de C. Ce cube unique est noté C_F.*

Autrement dit, pour une phase enroulée Φ de taille $N \times L \times M$, une face F de $G_{grid}(\Phi)$ est extérieure si tous ses sommets sont sur l'un des plans limites du volume $x = 0$, $x = N - 1$, $y = 0$, $y = L - 1$, $z = 0$ ou $z = M - 1$.

Definition 3.6 (Résidu d'une face par rapport à un cube)**.** *Soit F une face du graphe $G_{grid}(\Phi)$, appartenant à un cube C. La valeur du résidu $Res_C(F)$ par rapport au cube C est définie comme suit :*

$$Res_C(F) = \frac{1}{2\pi} \sum_{(v_i, v_{i+1}) \in E_F} \text{wrap}\left(\Phi(v_{i+1}) - \Phi(v_i)\right),$$

où :

- *C est un cube auquel appartient la face F,*
- *les sommets (v_1, v_2, v_3, v_4) de F sont ordonnés selon la **convention canonique** induite par le cube C, cette convention suit la **règle de la main droite**. (cf. Définition 3.4),*
- *$\Phi(v_i)$ est la phase enroulée au sommet v_i,*
- *$\text{wrap}(\theta)$ ramène l'angle θ dans l'intervalle $(-\pi, \pi]$.*

Remarque : Pour une face qui appartient simultanément à deux cubes, le signe de son résidu par rapport au premier cube est opposé à celui par rapport au second cube. Dans

l'exemple illustré à la Figure 3.4, le cube C_1, défini par les sommets A, B, C, D, E, F, G, H, et le cube C_2, défini par les sommets B, C, I, J, G, H, K, L, partagent la face F, formée par les sommets B, C, G, H. Le résidu de cette face F par rapport au cube C_1 est opposé à son résidu par rapport au cube C_2, ce qui s'exprime par la relation $Res_{C_1}(F) = -Res_{C_2}(F)$. Les signes opposés des deux résidus, vient du fait que le sens canonique de F par rapport à C_1 est opposé au sens canonique de F par rapport à C_2.

Proposition 3.1. *Par construction, Pour tout cube C dans le graphe $G_{grid}(\Phi)$, et pour toute face F de C,*

- *$Res_C(F)$ est un nombre entier.*
- *$Res_C(F) \in \{-1, 0, 1\}$.*

Proposition 3.2. *Soit C un cube dans un graphe de grille $G_{grid}(\Phi)$, et $F_1, F_2, F_3, F_4, F_5, F_6$ les faces de C. Alors,*

$$Res(C) := \sum_{i=1}^{6} Res_C(F_i) = 0$$

En effet, le résidu est la somme de différences de phase enveloppées autour d'une face, cette différence est donc toujours un multiple de 2π [4,5]. La contrainte physique du résidu implique que ce dernier en valeur absolue est borné par 1 [7]. De plus, une autre propriété importante est que la somme des valeurs résiduelles des faces du même cube est égale à 0. Cette propriété découle de la conservation des résidus dans un volume fermé, comme démontré dans [8]. Il est donc possible d'étendre le résultat de la Proposition 3.2 à l'ensemble des faces extérieures du graphe $G_{grid}(\Phi)$. En effet, la somme des résidus des faces extérieures par rapport aux cubes auxquelles elles appartiennent est nulle.

Theorem 3.1. *Pour le graphe de grille $G_{grid}(\Phi)$, la somme des résidus des faces extérieures de $G_{grid}(\Phi)$ par rapport à leur cube est nulle.*

$$\sum_{F \text{ face extérieure}} Res_{C_F}(F) = 0$$

Ainsi le Théorème **(3.1.)** affirme que le $G_{grid}(\Phi)$ a autant de faces extérieures avec des résidus positifs que de faces extérieures avec des résidus négatifs.

3.1.2 Graphe de faces :

Definition 3.7. *Une boucle de singularité est une suite de faces* $(F_1, F_2, \ldots, F_m)$ *dans le graphe* $G_{grid}(\Phi)$ *tel que :*

1. $\forall i < m$ F_i *et* F_{i+1} *sont dans le même cube que l'on notera* C_i.
2. $\forall i < m$ $Res_{C_i}(F_i) = -Res_{C_i}(F_{i+1})$

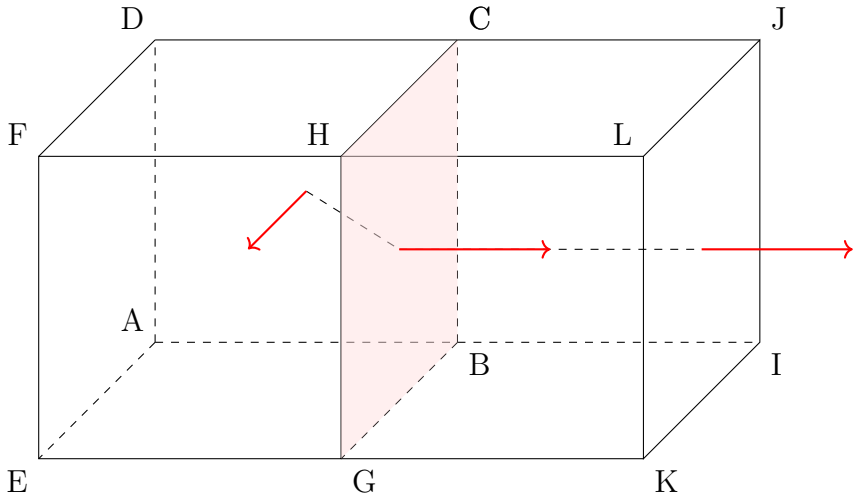


FIGURE 3.5 Exemple d'un tronc d'une boucle de singularité.

Dans la Figure 3.5, on considère trois faces distinctes que l'on notera F_{ABCD}, F_{BCHG} et F_{IJLK}, formées respectivement par les sommets $\{A, B, C, D\}$, $\{B, C, G, H\}$ et $\{I, J, K, L\}$. Pour chacune de ces faces, on utilise l'orientation du vecteur normal (indiqué en rouge)pour visualiser son résidu. Lorsque le vecteur pointe vers l'extérieur du cube, le résidu de la face par rapport à ce cube est positif $+1$, et À l'inverse, lorsqu'il pointe vers l'intérieur du cube, le résidu est négatif -1. Par exemple, pour le cube C_1 défini par les sommets A, B, C, D, E, F, G, H, $Res_{C_1}(F_{ABCD}) = -1$, car son vecteur normal rentre dans le cube C_1, tandis que $Res_{C_1}(F_{BCHG}) = 1$, car le vecteur en rouge sort du cube. De même, pour le cube C_2, $\text{Res}_{C_2}(F_{HCBG}) = -1$ et $\text{Res}_{C_2}(F_{IJLK}) = 1$. Ces trois faces sont liées, car elles satisfont les conditions énoncées dans la Définition 3.7. Les faces F_{ABCD} et F_{BCHG} sont dans le même cube C_1, et $Res_{C_1}(F_{ABCD}) = -Res_{C_1}(F_{BCHG}) = -1$. De même, les faces F_{BCHG} et F_{IJLK} sont dans le même cube C_2, et $Res_{C_2}(F_{BCHG}) = -Res_{C_2}(F_{IJLK}) = -1$.

Definition 3.8 (graphe de faces)**.** *Le graphe de faces* $G_{faces}(\Phi) = (V_{faces}(\Phi), A_{faces}(\Phi))$ *est un graphe* ***orienté*** *tel que :*

1. $V_{faces}(\Phi) = \{F \mid F$ *est une face de* $G_{grid}(\Phi)\}$.

2. $\forall F_1, F_2 \in V_{faces}(\Phi),\ ((F_1, F_2) \in A_{faces}(\Phi)) \Leftrightarrow \left\{ \begin{array}{l} F_1 \text{ et } F_2 \text{ sont dans le même cube } C, \\ Res_C(F_1) = -1 \\ Res_C(F_2) = 1 \end{array} \right\}$

Definition 3.9. *On définit deux ensembles de faces, qui correspondent également aux nœuds du graphe* $G_{faces}(\Phi)$ *:*

$$N_{no\ parent} = \{F \in V_{faces}(\Phi) \mid deg^-(F) = 0\},$$

représentant les faces n'ayant aucun prédécesseur dans le graphe $G_{faces}(\Phi)$*, et*

$$N_{no\ child} = \{F \in V_{faces}(\Phi) \mid deg^+(F) = 0\},$$

représentant les faces n'ayant aucun successeur $G_{faces}(\Phi)$*.*

Proposition 3.3. *Les ensembles* $N_{no\ parent}$ *et* $N_{no\ child}$ *regroupent les faces extérieures de* $G_{grid}(\Phi)$*.*

Démonstration. Les ensembles $N_{\text{no parent}}$ et $N_{\text{no child}}$ regroupent les faces qui appartiennent à un seul cube, c'est-à-dire les faces extérieures de $G_{grid}(\Phi)$ (cf. Définition 3.5). Supposons par l'absurde qu'une face $F \in N_{\text{no child}}$ appartienne à deux cubes C_1 et C_2, et que son prédécesseur F' dans $G_{faces}(\Phi)$ soit une face de C_1.

D'après la Définition 3.8, on a

$$\text{Res}_{C_1}(F') = -1 \quad \text{et} \quad \text{Res}_{C_1}(F) = 1.$$

Cependant, comme F est une face commune entre C_1 et C_2, on a également

$$\text{Res}_{C_2}(F) = -1.$$

Selon le Théorème 3.1, il existe alors une face F'' dans C_2 telle que

$$\text{Res}_{C_2}(F'') = 1,$$

ce qui implique que F a un prédécesseur dans $G_{faces}(\Phi)$, contredisant ainsi le fait que $F \in N_{\text{no parent}}$. Par un raisonnement analogue, on peut montrer que $N_{\text{no child}}$ regroupe uniquement les faces extérieures de $G_{grid}(\Phi)$.

□

Dans l'exemple de l'image 3.5 illustrant un tronçon d'une boucle de singularité, les faces F_{ABCD}, F_{BCHG} et F_{IJLK} sont des sommets connectés dans graphe $G_{faces}(\Phi)$. On a, par exemple, l'arc (F_{ABDC}, F_{BCHG}) dans le graphe $G_{faces}(\Phi)$, car F_{ABCD} et F_{BCHG} sont dans le même cube C_1, et $Res_{C_1}(F_{ABCD}) = -1$ et $Res_{C_1}(F_{BCHG}) = 1$. Ce graphe est construit pour faciliter l'identification des boucles de singularités : chaque cycle dans $G_{faces}(\Phi)$ correspond à une boucle de singularité fermée. En revanche, les boucles de singularité ouvertes commencent par un face dans $N_{\text{no parent}}$ représentant les nœuds de $G_{faces}(\Phi)$ sans prédécesseur et finissent par un face dans $N_{\text{no child}}$ représentant les nœuds de $G_{faces}(\Phi)$ sans successeur.

Proposition 3.4.

$$\text{Card}(N_{no\ child}) = \text{Card}(N_{no\ parent})$$

Cette proposition est un résultat direct du Théorème 3.1, et de la Proposition 3.3.

Proposition 3.5. *Pour tout $F \in V_{faces}(\Phi)$,*

$$deg^-(F) \leq 3$$

$$deg^+(F) \leq 3$$

Démonstration. La Proposition 3.5 découle du fait que chaque cube a au plus 6 faces, et selon la proposition 3.2, la somme des résidus de ses faces n'est nulle. Par conséquent, pour chaque résidu négatif, il existe au plus trois résidus positifs. De plus, d'après la définition d'un arc, celui-ci relie deux faces d'un même cube, partant d'une face avec un résidu négatif par rapport au cube et arrivant à une face avec un résidu positif par rapport à ce même cube. Ainsi, on peut déduire que :

$$\deg^+(F) \leq 3$$

Et de manière analogue, on a :

$$\deg^+(F) \leq 3$$

. □

3.2 Construire les boucles de singularité :

Pour construire **les boucles de singularité fermées**, on identifie les cycles dans les graphes de faces $G_{faces}(\Phi)$. Comme mentionné précédemment dans la section 3.1.2, chaque cycle dans

$G_{faces}(\Phi)$ correspond à une boucle de singularité fermée. Or, une fois qu'un cycle de faces a été identifié, il est marqué comme traité, et ne rentre plus dans la construction d'une autre boucle de singularité. En effet, **une face avec un résidu ne peut appartenir qu'à une seule boucle de singularité.** [8–10].

Pour ce qui concerne les **boucles de singularité ouvertes**, nous cherchons des chemins reliant les faces sans parents ($N_{noparent}$, cf. Définition 3.9) aux faces sans successeurs ($N_{nochild}$, cf. Définition 3.9). Ces chemins représentent un couplage entre les éléments de $N_{noparent}$ et $N_{nochild}$. L'existence de ce couplage est garantie par la Proposition 3.4 qui affirme que le nombre de faces dans $N_{noparent}$ est égal au nombre de faces dans $N_{nochild}$.

Le graphe $G_{faces}(\Phi)$ peut être décomposé en une union disjointe de cycles fermés et de chemins ouverts qui relient les nœuds face sans parents aux nœuds faces sans successeurs. Cette décomposition est garantie par le fait chaque face ne peut appartenir qu'à une seule boucle de singularité [8–10].

3.2.1 Démêlage du graphe de faces :

Un élément important qui complexifie la construction des boucles de singularité, en les liant les faces une par une, est le nombre de choix possibles à chaque nœud du graphe $G_{faces}(\Phi)$. En effet, dans un graphe de faces $G_{faces}(\Phi)$, une face peut avoir jusqu'à 3 prédécesseurs et 3 successeurs (cf. Proposition 3.5). Ainsi, il pourrait exister plusieurs configurations possibles pour créer la boucle de singularité qui passe par une face donnée. Or, nous savons que chaque face ne peut appartenir qu'à une seule boucle de singularité, qu'elle soit ouverte ou fermée.

Par conséquent, si on décide préalablement d'associer à chaque successeur un seul prédécesseur, et réciproquement, à chaque prédécesseur un seul successeur, alors les boucles de singularité seraient directement construites. Les méthodes décrites dans la littérature des algorithmes de déroulement de phase basés des boucles de singularité [8–10] utilisent des associations directes entre les résidus et des structures de données comme les *look-up tables*. Dans notre cas, et grâce à l'utilisation du graphe $G_{faces}(\Phi)$, nous avons la possibilité de résoudre ces choix de manière différente : nous détectons au préalable les faces où plusieurs successeurs ou prédécesseurs sont possibles, et nous choisissons explicitement un unique prédécesseur et un unique successeur pour chaque nœud.

Cette opération est ce que l'on appellera **le démêlage du graphe de faces** $G_{faces}(\Phi)$. Et nous allons appeler les nœuds du graphe $G_{faces}(\Phi)$ qui ont plusieurs successeurs des **faces noués**, qu'on définira plus formellement dans la définition suivante.

Definition 3.10. *Une face nouée est une face (nœud) dans le graphe* $G_{faces}(\Phi)$ *qui a plus*

d'un seul successeur, c.-à-d. Une face nouée est un face v tel que $deg^+(v) > 1$. On appelle une face nouée ***simple*** *si $deg^+(v) = 2$. Et* ***complexe*** *si $deg^+(v) = 3$. (cf. Proposition 3.5)*

Theorem 3.2. *Soit F_1 une face nouée simple dans le graphe $G_{faces}(\Phi)$. et F_1' et F_2' ses successeurs, il existe une unique face F_2 telle que : F_2 est une* ***face nouée simple*** *et F_2 est* ***le prédécesseur commun*** *entre F_1' et F_2' Cette face unique est ce que l'on appellera la* ***face nouée associée*** *à F_1.*

Démonstration. Soit F_1 un face nouée simple dans le graphe $G_{faces}(\Phi)$. Alors, il existe deux faces F_1' et F_2' tel que (F_1, F_1') et (F_1, F_2') sont des arcs dans $G_{faces}(\Phi)$. Selon la Définition du graphe de faces 3.8, F_1, F_1' et F_2' sont dans le même cube, que l'on notera C, et $\text{Res}_C(F_1) = -1$ et $\text{Res}_C(F_1') = 1$ et $\text{Res}_C(F_2') = 1$. Or selon la Proposition 3.2, la somme des résidus des faces d'un cube est nulle, donc il existe une face F_2 dans le cube C tel que $\text{Res}_C(F_2) = -1$. Ainsi, F_2 est le successeur commun entre F_1' et F_2'. □

Theorem 3.3. *Soit F_1 une face nouée complexe dans le graphe $G_{faces}(\Phi)$, avec trois successeurs F_1', F_2', et F_3'. Il existe exactement deux faces F_2 et F_3 telles que F_2 et F_3 sont des faces nouées complexes et F_2 et F_3 sont les prédécesseurs communs des trois faces F_1' , F_2', et F_3'. Ces deux faces sont appelées les* ***faces nouées associées*** *à F_1.*

La démonstration de ce théorème est similaire à celle du Théorème 3.2.

Pour les faces nouées simples :

Le Théorème 3.2 affirme que, pour toute face nouée simple F_1, il existe une autre face nouée simple F_2 partageant les mêmes deux successeurs F_1' et F_2'. Ainsi, lors de la construction d'une boucle de singularité passant par le nœud F_1, il existe deux possibilités : la boucle peut être prolongée via le successeur F_1' ou via le successeur F_2'.

Supposons par exemple, que la boucle passe par le nœud F_1 et continue via le successeur F_1'. Dans ce cas, les nœuds F_1 et F_1' seront marqués comme traités, et ne seront plus considérés dans la construction des boucles de singularité. Et donc une boucle de singularité passant par F_1 n'aura qu'une seule possibilité de continuer via le successeur F_2'. Ainsi, on remarque que les boucles de singularité ont créé un couplage entre les faces nouées simples et leurs successeurs.

Plus précisément, pour chaque face nouée simple et sa face associée, nous établissons un couplage arbitraire entre ces deux faces nouées simples et leurs successeurs respectifs. Dans ce couplage, chaque face nouée simple est associée à un unique successeur 3.6. Ainsi dans le

cube C contenant les deux faces nouées simples et leurs successeurs respectifs, chaque face aura exactement un prédécesseur et chaque prédécesseur sera associé à un unique successeur.

La Figure 3.6 que le sous-graphe formé par les faces nouées simples et leurs successeurs respectifs est un graphe biparti complet. Dans ce graphe, on choisit le couplage qui associe chaque face nouée simple à un unique successeur. Il faut noter ici que pour ce graphe biparti complet, il existe 2! couplages possibles [47]. On en choisit arbitrairement un, et on supprime les autres arrêtes du graphe biparti complet.

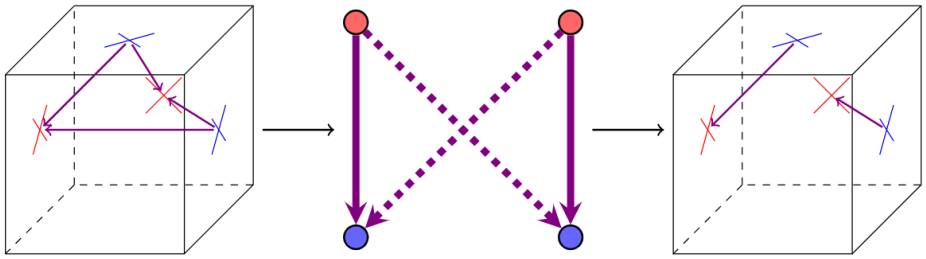


FIGURE 3.6 Illustration du couplage de deux faces nouées simples et leurs successeurs respectifs.

Pour les faces nouées complexes :

Pareillement que pour les faces nouées simples, la construction des boucles de singularité créent un coulage entre les faces nouées complexes associées 3.3 et leurs successeurs. La différence principale réside dans le fait que, pour les faces nouées complexes, il existe deux faces associées partageant les mêmes trois successeurs. Ainsi, le sous-graphe biparti complet formé par les faces nouées complexes et leurs successeurs est de degré 3, il existe donc $3! = 6$ couplages possibles [47]. Par conséquent, et de manière analogue aux faces nouées simples, dans un cube C contenant trois faces nouées complexes et leurs successeurs respectifs, ce processus de démêlage permet de s'assurer chaque face a exactement un prédécesseur et chaque prédécesseur est associé à un unique successeur.

L'exemple de la Figure 3.7 illustre un couplage où les trois faces nouées complexes et leurs trois successeurs respectifs sont connectés par un couplage parfait sélectionné parmi les six possibles.

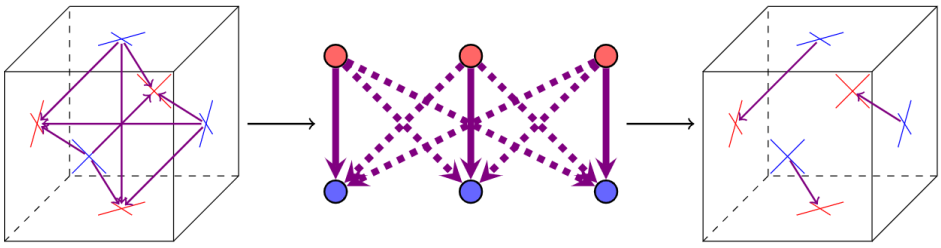


FIGURE 3.7 Le graphe au milieu (biparti complet) est le graphe formé par une paire de deux faces simples nouées dans le même cube, et qui partagent les mêmes successeurs. Un des deux couplages possible est gardé et les autres arrêtes en pointillés sont supprimées.

Proposition 3.6. *Le nouveau graphe démêlé* $\tilde{G}_{faces}(\Phi)$ *est un graphe orienté, tel que :*

$$\forall F \in deg_{+}(F) \leq 1$$

$$\forall F \in deg_{-}(F) \leq 1$$

À partir des résultats obtenus pour les faces nouées simples et complexes, le **démêlage** du graphe de faces $G_{\text{faces}}(\Phi)$ permet de garantir que chaque face possède au plus **un seul prédécesseur** et au plus **un seul successeur**.

Dans cette configuration, les boucles de singularité ouvertes reliant les faces sans parents (celles sans prédécesseurs) aux faces sans successeurs (celles sans successeurs) correspondent aux **composantes connexes sous forme de chemins dirigés** dans $G_{\text{faces}}(\Phi)$. De manière similaire, les **cycles dirigés** dans $G_{\text{faces}}(\Phi)$ représentent les **boucles de singularité fermées**, où chaque face est reliée circulairement à ses successeurs et prédécesseurs.

3.3 Algorithmes de construction des boucles de singularité :

Maintenant que le graphe de faces a été démêlé, nous allons détecter ses composantes connexes, qui correspondent aux boucles de singularité ouvertes et fermées. Pour ce faire, nous utiliserons les propriétés structurelles du graphe des faces afin de concevoir les algorithmes les plus efficaces possibles.

3.3.1 Les boucles ouvertes :

Algorithm 2 Identifier les boucles de singularité ouvertes

```
1: procedure FillOpenPaths
2:     FillStartingOpenPaths
3:     layer ← N_no parent                          ▷ Ensemble des nœuds sans prédécesseur
4:     origins ← dictionnaire vide                  ▷ Pour suivre les origines en temps O(1)
5:     paths ← dictionnaire vide                    ▷ Pour stocker les chemins en cours
6:     for all point ∈ layer do
7:         paths[point] ← [point]                   ▷ Initialise le chemin avec la face de départ
8:         origins[point] ← point
9:     end for
10:    while layer ≠ ∅ do                           ▷ Tant qu'il y a des nœuds à traiter
11:        new_layer ← ∅                            ▷ Pour stocker les successeurs à traiter ensuite
12:        for all point ∈ layer do
13:            if Successeurs(point) ≠ ∅ then
14:                for all successeur ∈ Successeurs(point) do
15:                    origins[successeur] ← origins[point]
16:                    paths[origins[successeur]].append(successeur)
17:                    new_layer ← new_layer ∪ {successeur}
18:                end for
19:            end if
20:        end for
21:        layer ← new_layer                        ▷ Passe à la couche suivante
22:    end while
23:    Retourner paths   ▷ Chaque entrée de paths est une boucle de singularité ouverte
24: end procedure
```

La correction de l'Algorithme 2 repose sur le fait que le graphe est équilibré, c'est-à-dire qu'il existe une bijection entre les faces sans prédécesseur ($N_{\text{no parent}}$) et les faces sans successeur ($N_{\text{no child}}$), conformément à la Proposition 3.4. De plus, dans le graphe de faces démêlé $\widetilde{G}_{\text{faces}}(\Phi)$, chaque face possède au plus un prédécesseur et un successeur. Les seules faces ne possédant pas exactement un prédécesseur et un successeur sont les faces externes, qui appartiennent soit à $N_{\text{no parent}}$, soit à $N_{\text{no child}}$. Par conséquent, en partant d'une face sans prédécesseur, on peut avancer dans le graphe en suivant les successeurs successifs jusqu'à atteindre une face sans successeur. Ce processus définit un chemin ouvert reliant une face de $N_{\text{no parent}}$ à une face de $N_{\text{no child}}$.

L'Algorithme 2 fait ceci en parallèle pour toutes les faces sans prédécesseur. Il commence par remplir les chemins ouverts à partir des faces sans prédécesseur, et au fur et à mesure, il avance dans le graphe en suivant les successeurs successifs jusqu'à atteindre les faces sans successeur. Chaque chemin ainsi construit correspond à une **boucle de singularité ouverte**.

Complexité temporelle : La complexité temporelle de cet algorithme est de l'ordre de

$O(L)$ où L est le plus long chemin ouvert dans le graphe. Donc dans le pire des cas, la complexité temporelle et de l'ordre de $O(|V_{faces}(\Phi)|)$.

3.3.2 Les boucles fermées :

Algorithm 3 Détection des boucles fermées dans le graphe démêlé

Require: $\widetilde{G}_{\text{faces}}(\Phi)$: le graphe démêlé des faces
Require: *open_loops* : ensemble des nœuds appartenant aux boucles ouvertes déjà identifiées
Ensure: *cycles* : liste des boucles fermées détectées

1: *nodes* ← ensemble des nœuds de $\widetilde{G}_{\text{faces}}(\Phi)$
2: *cycles* ← liste vide
▷ Marquer les nœuds des boucles ouvertes comme visités
3: **for all** *node* ∈ *open_loops* **do**
4: *nodes*.remove(*node*)
5: **end for**
▷ Détection des boucles fermées
6: **while** *nodes* ≠ ∅ **do**
7: *init* ← extraire un nœud de *nodes*
8: *cycle* ← [ ]
9: *current* ← *init*
10: **repeat**
11: *cycle*.append(*current*)
12: *nodes*.remove(*current*)
13: *current* ← successeur de *current*
14: **until** *current* = *init*
15: *cycles*.append(*cycle*)
16: **end while**
17: **Retourner** *cycles*

Pareillement dans cet algorithme (cf. Algorithme 3), on se base sur le fait que chaque face qui n'est ni dans $N_{noparent}$ ni dans $N_{nochild}$ est forcément dans une boucle fermée, et que le graphe démêlé est un graphe où chaque face a au plus un prédécesseur et un successeur. Par conséquent, si on marque tous les nœuds de boucles ouvertes déjà identifiées, alors chaque nœud restant a exactement **un unique successeur et un unique prédécesseur**.

On est sur une complexité qui est de l'ordre de la taille de la plus grande boucle fermée, et donc dans le pire des cas, la complexité est de l'ordre de $O(|V_{face}(\Phi)|)$.

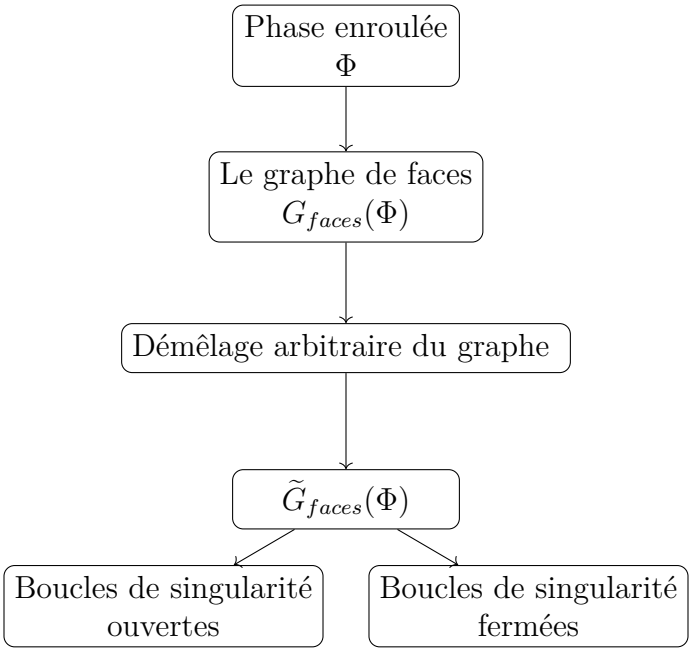


FIGURE 3.8 Étapes de construction des boucles de singularité ouvertes et fermées à partir de la phase enroulée Φ.

3.4 Amélioration des boucles de singularité :

Actuellement, un couplage arbitraire a été choisi pour chaque face nouée, qu'elle soit simple ou complexe, et les boucles de singularité ouvertes et fermées ont été construites en suivant ces couplages. Cependant, il est possible que certains de ces couplages produisent des boucles de singularité qui pourraient être **décomposées en sous-boucles distinctes**.

Considérons, par exemple, une boucle de singularité initiale, comme illustrée dans la Figure 3.9a. Cette boucle peut être représentée comme un cycle C dans le graphe de faces $G_{\text{faces}}(\Phi)$, défini par une séquence de faces $F_1, F_2, \ldots, F_k$ connectées par des successeurs. Si le couplage initial est remplacé par un nouveau couplage (Figure 3.9b). Avec ce nouveau couplage, la boucle de singularité initiale est décomposée en deux boucles, on arrive à extraire une boucle de singularité **fermée** plus petite.

En ajustant les couplages de manière appropriée, nous obtenons une structure optimisée des boucles de singularité, réduisant la complexité de leur représentation et assurant une meilleure partition des singularités.

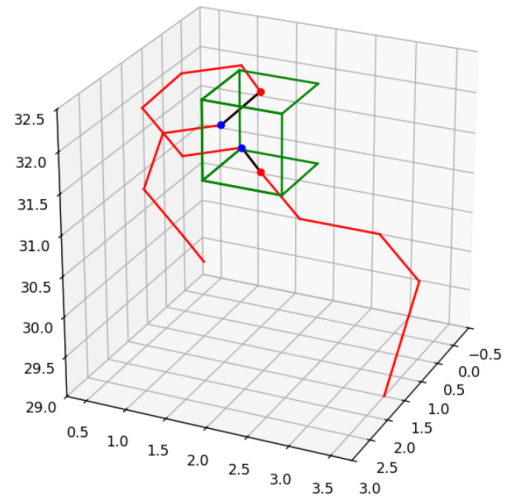
Pour une face nouée simple : Si une boucle de singularité traverse un cube en passant par une face nouée simple et revient dans le même cube, cela signifie que deux boucles différentes sont connectées à travers ce cube partagé. Autrement dit, nous détectons toutes les faces

nouées simples dont la face nouée associée appartient également à la même boucle.

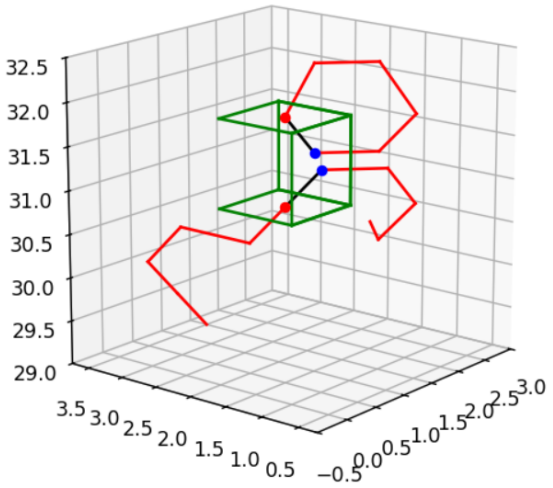
Dans ce cas, nous remplaçons le premier couplage choisi pour cette face nouée simple par le deuxième couplage qui va séparer la première boucle en une boucle plus petite. Ainsi, la boucle initiale est coupée en deux boucles distinctes et de plus petite taille (cf. Figure 3.9).

Pour une face nouée complexe : Pour les faces nouées complexes, nous suivons une approche similaire. Il existe plusieurs façons d'apparier ces nœuds avec leurs successeurs, mais notre objectif est de choisir le couplage qui minimise la taille des boucles résultantes.

En examinant les couplages possibles, nous choisissons celui qui divise les grandes boucles en boucles plus petites en évitant que les faces nouées complexes ne relient plusieurs boucles entre elles. Cela peut impliquer de modifier les arêtes sortantes des nœuds concernés pour sélectionner un couplage alternatif.



(a) Avec le couplage choisi aléatoirement.



(b) Avec le couplage adéquat.

FIGURE 3.9 Comparaison des deux couplages.

Conclusion :

Dans un premier temps, nous commençons par le démêlage initial du graphe des faces $G_{\text{faces}}(\Phi)$, ce qui produit un graphe démêlé $\widetilde{G}_{\text{faces}}(\Phi)$. Ce graphe est utilisé pour détecter les boucles de singularité initiales, en utilisant les algorithmes annoncés dans la partie 3.3. Ces boucles de singularité nous permettent d'identifier les couplages à améliorer dans le graphe. Une fois ces couplages identifiés, nous procédons à un démêlage optimisé de $G_{\text{faces}}(\Phi)$, en ajustant le couplage pour séparer les grandes boucles en boucles plus petites. Cela génère un nouveau graphe démêlé $\widetilde{G}'_{\text{faces}}(\Phi)$, qui reflète les améliorations apportées. Enfin, ce nouveau graphe démêlé est utilisé pour détecter les nouvelles boucles de singularité. Ces boucles sont plus petites et plus nombreuses, ce qui permet une analyse plus fine de la structure de la phase Φ. Ce processus est illustré dans la Figure 3.10, où chaque étape est représentée par un nœud et les transitions entre les étapes sont indiquées par des flèches. La flèche rouge courbée souligne l'étape cruciale où l'identification des couplages à améliorer conduit au démêlage optimisé du graphe.

Dans l'annexe C, nous analyserons sur les données présentées dans la section des résultats 5 comment l'amélioration proposée réduit la taille des boucles de singularité sur des données de test.

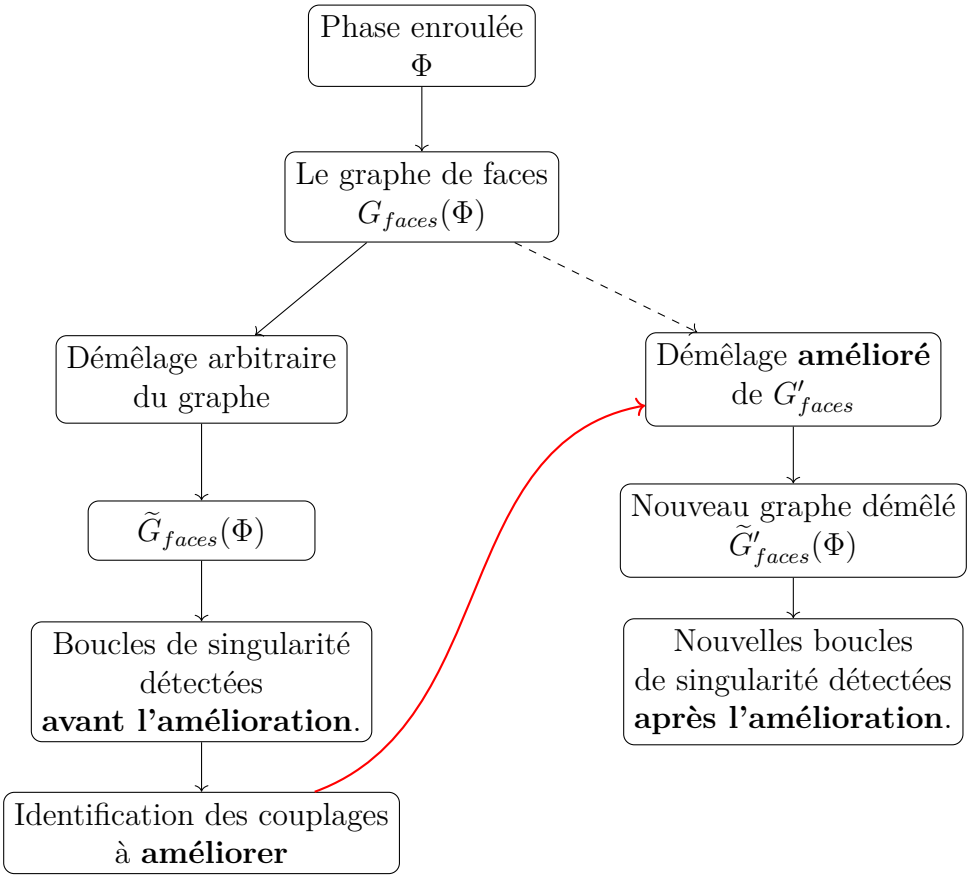


FIGURE 3.10 Construction des boucles de singularité avant et après l'amélioration.

3.5 Représentation des boucles de singularité dans l'espace Euclidien :

Cette section prépare l'analyse des boucles de singularité dans l'espace Euclidien, en vue de leur représentation dans la section suivante. L'objectif est de transformer les boucles, initialement définies comme des successions de faces dans le graphe $G_{\text{faces}}(\Phi)$, en des courbes composées de points dans l'espace Euclidien.

Pour cela, chaque face F de la boucle est représentée par son barycentre c_F, défini selon 3.2. Ainsi, une boucle de singularité, qu'elle soit ouverte ou fermée, composée de faces $F_1, F_2, F_3, \ldots$ dans $G_{\text{faces}}(\Phi)$, sera transformée en une suite de points $c_{F_1}, c_{F_2}, c_{F_3}, \ldots$ dans l'espace Euclidien. Cette transformation permet de représenter chaque boucle comme une courbe euclidienne, facilitant son étude géométrique.

Un second point important concerne les boucles ouvertes, qui doivent être **complétées**. Cela implique de créer des chemins supplémentaires pour relier les extrémités de ces boucles ouvertes dans l'espace Euclidien, garantissant ainsi une continuité et une fermeture dans leur

représentation.

Fermeture des boucles ouvertes :

Les boucles ouvertes sont transformées en boucles fermées grâce à l'ajout de chemins de fermeture sur leurs extrémités. Ce processus garantit que toutes les boucles sont désormais fermées, confinant les erreurs de phase autour des singularités. Les détails complets de cette procédure sont présentés dans l'Annexe (A).

Conclusion :

Ainsi, une fois que les boucles de singularité ouvertes ont été fermées, donc complétées, nous obtenons une représentation complète des boucles de singularité Et nous allons pouvoir les représenter dans l'espace Euclidien, comme décrit plus haut. Chaque succession de faces que ça soit dans les boucles ouvertes ou fermées sera remplacé par une succession de leurs barycentres respectifs, qui seront reliés par des chemins. Ainsi, l'algorithme final permettant de construire les boucles de singularité dans l'espace Euclidien est complet.

CHAPITRE 4 REMPLISSAGE DES SURFACES D'ERREUR

Après la détection des résidus et leur regroupement en formant des boucles de singularité, il est nécessaire de fermer ces boucles afin de garantir des chemins d'intégration cohérents et indépendants dans le processus de reconstruction de phase. Pour cela, nous avons fermé les boucles ouvertes, et nous avons opté à ce que toutes les boucles de singularité soient représentée comme des objets dans l'espace Euclidien (cf. la Section 3.5). Le travail de remplir ces boucles est essentiellement un problème géométrique. L'objectif est de trouver, dans l'espace Euclidien en $3D$, des surfaces qui remplissent ces boucles tout en empêchant les chemins d'intégration de les traverser. Ces surfaces de coupure *branch-cuts* servent à bloquer le passage à travers les boucles de singularité, garantissant ainsi une reconstruction de phase sans discontinuités [8–10].

Pour une boucle donnée Γ, plusieurs surfaces fermantes peuvent être identifiées. Chacune de ces surfaces fermantes est traversée par un certain ensemble d'arêtes du graphe $G_{grid}(\Phi)$ associé à la phase déroulée Φ. La Figure 4.1 illustre une simple boucle fermée dans l'espace euclidien tridimensionnel. Cette boucle est fermée par deux surfaces d'erreur, l'une en rose et l'autre en bleu. Chacune des deux surfaces est traversée par un ensemble d'arêtes du graphe $G_{grid}(\Phi)$.

Bien que les deux surfaces d'erreur remplissent leur fonction en empêchant les chemins d'intégration de traverser la boucle de singularité, l'une d'elles bloque un plus grand nombre d'arêtes que l'autre. L'objectif de cette partie est donc de déterminer la surface d'erreur optimale qui minimise le nombre d'arêtes bloquées, tout en garantissant que les chemins d'intégration ne traversent pas les boucles de singularité.

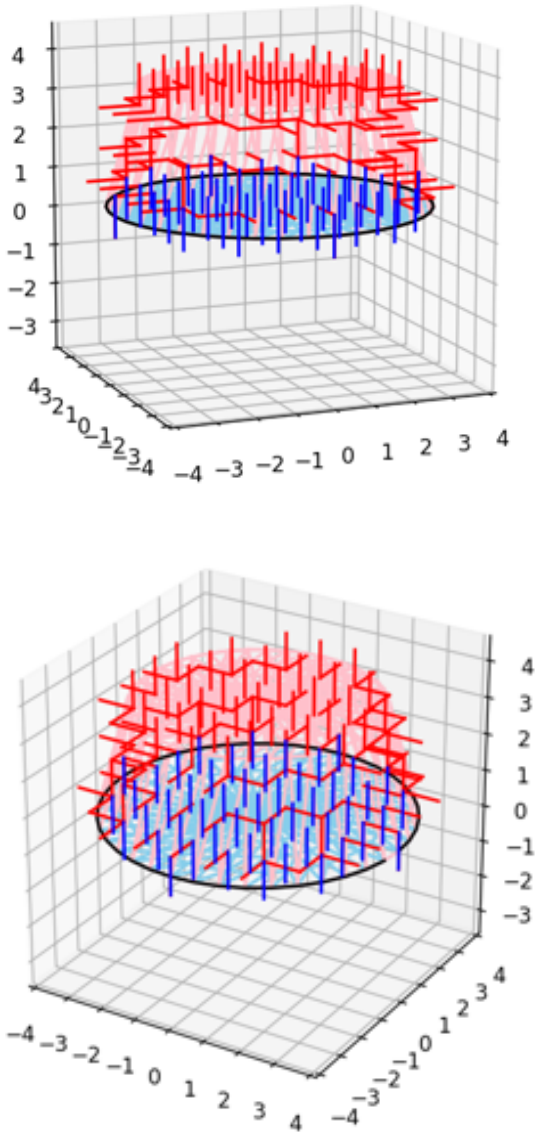


FIGURE 4.1 Les deux surfaces fermantes de la boucle fermée Γ.

4.1 Surface fermante initiale :

Dans un premier temps, la première tentative de remplissage de surface se fera grâce à des surfaces fermantes sous forme de maillage triangulaire. Ces surfaces seraient des surfaces continues qui vont fermer dans un premier temps les boucles détectées. (cf. Figure 4.1).

4.1.1 Surface de maillage :

Definition 4.1. ***Surface de maillage :*** *Un maillage M dans un espace Euclidien 3D, est formé par*

$$M = (I, T)$$

où $I = \{P_1, \ldots, P_n\}$ est un ensemble de points dans l'espace Euclidien 3D, et T est un ensemble de triangles.

La surface d'une surface de maillage M est la somme des surfaces de ses triangles $S(M) := \sum_{t \in T} S(t)$

On va utiliser l'algorithme développé par [48] dans la construction de cette surface de maillage, dont va découler la première surface d'erreur à remplir.

En effet, une fois on a obtenu la surface de maillage continue, on va voir pour chacun de ses triangles $t \in T$, les arêtes qui le traversent. Ces arêtes-ci formeront une surface d'erreur valide, mais pas forcément optimale.

4.1.2 Construction de la surface de maillage :

Pour un ensemble de points $\Gamma = \{Q_1, \ldots, Q_n\}$, construisons un maillage $M = (I, T)$ qui formera une surface fermante de Γ.

En effet, la première étape dans la construction de la surface de maillage est de construire les points I qui vont former ce maillage avant de les lier d'une manière à ce que l'on obtienne des triangles.

Le papier de [48] propose une méthode pour la construction de ces points-ci, en commencant par le barycentre de Γ, qui sera ensuite relié à chaque sommet de Γ par des segments. Ces segments-ci seront par la suite subdivisés pour obtenir des points intérieurs. Les points intérieurs seront liés entre eux de manière à former des polygones, qui seront par la suite subdivisés en triangles. (cf. Algorithme 4).

Cette construction est en effet comparable à la formation d'une toile d'araignée (cf. Figure 4.2). Pour le reste de l'algorithme, nous fixons que, pour chaque boucle de singularité de

Algorithm 4 Création d'un maillage triangulaire

1. Initialisation :
Soit $Q_1, \ldots, Q_n$ les sommets de la frontière.
Définir le nombre de triangles souhaité : m.
Calculer le point central du polygone : $P_c = \frac{1}{n} \sum_{i=1}^{n} Q_i$.

2. Création des sommets intérieurs :
Déterminer le nombre d'anneaux à ajouter :

$$s = \left\lfloor \frac{m-n}{2n} \right\rfloor$$

for chaque sommet Q_i **do**
  for $j = 1, 2, \ldots, s$ **do**
    Ajouter des sommets intérieurs P_i^j le long du segment $Q_i P_c$:

$$P_i^j = Q_i + \frac{j}{s+1}(P_c - Q_i)$$

  end for
end for

3. Formation des polygones :
for chaque j **do**
  Relier les sommets $P_1^j, P_2^j, \ldots, P_n^j$ dans l'ordre pour former un polygone.
end for

4. Construction de maillage :
for chaque facette quadrilatérale $P_i^j, P_{i+1}^j, P_{i+1}^{j+1}, P_i^{j+1}$ **do**
  Diviser en deux triangles en ajoutant une diagonale $P_i^j P_{i+1}^{j+1}$.
end for

5. Résultat :
Obtenir un maillage triangulaire contenant $n(2s+1)$ triangles ou ajuster à m triangles en subdivisant des triangles existants.

taille n, nous allons construire $m = 3n + 1$ arêtes pour former le maillage. Ainsi, pour chaque boucle Γ de taille n, le maillage contiendra $O(n)$ triangles.

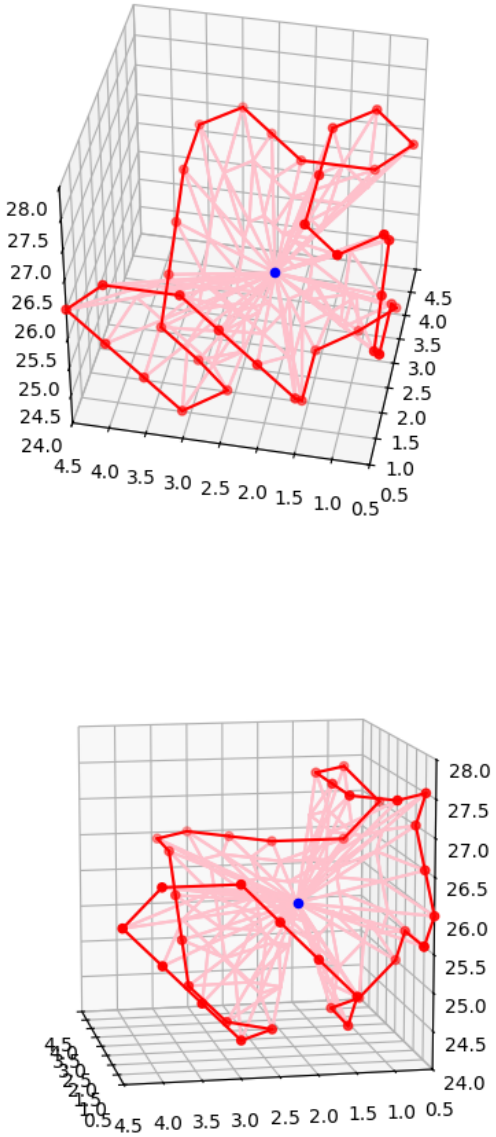


FIGURE 4.2 Une boucle Γ en rouge et la surface de maillage initiale qui la ferme en rose.

4.1.3 Minimisation de l'aire de la surface de maillage :

Il est important de noter que minimiser l'aire de la surface de maillage n'implique la minimisation des arêtes qui la traversent. Cependant, on peut juste la rapprocher à une surface qui va permettre de minimiser **le gap d'optimalité** entre la surface d'erreur minimale et les arrêtes qui traversent la nouvelle surface de maillage. Nous allons donc utiliser deux méthodes pour minimiser l'aire de la surface de maillage [48].

1. **Méthode de lissage de Laplace :** (*Laplacian Fairing*)
2. **Méthode d'échange d'arêtes :** (*Edge Swapping*)

Méthode de lissage de Laplace :

La méthode de lissage de Laplace ici consiste à déplacer chaque sommet P_i vers un nouveau barycentre pondéré de ses voisins. Ici nous allons garder les mêmes poids que ceux utilisés par les auteurs dans l'article [48].

Après chaque itération, nous allons calculer et stocker les poids

$$w_{ij} = \frac{S_{ij}}{\sum_{k \in N(i)} S_{ik}}$$

où

— S_{ij} est la surface des deux triangles qui partagent l'arête reliant P_i et P_j.
— $N(i)$ est l'ensemble des voisins de P_i.

Le nouveau sommet P_i' est alors donné par

$$P_i' = \sum_{j \in N(i)} w_{ij} P_j$$

Cette méthode rapproche chaque sommet de son voisin le plus éloigné, en donnant la priorité à ceux qui partagent la plus grande surface commune. Cela permet d'ajuster les positions des sommets et de réduire la surface des triangles formés. En ce qui concerne la complexité, chaque sommet a au plus 6 voisins dans le maillage [48], ce qui rend la complexité de l'algorithme linéaire par rapport au nombre de sommets. De plus, comme le nombre de sommets est linéairement proportionnel au nombre de triangles [48], la complexité reste également linéaire par rapport au nombre de triangles. Étant donné que le nombre de triangles est lui-même proportionnel à la taille de la boucle, la complexité finale de l'algorithme est linéaire par rapport à la taille de la boucle de singularité. Donc pour une boucle de singularité Γ de taille n, la complexité de l'algorithme est en $O(n)$.

Méthode d'échange d'arêtes :

La méthode d'échange d'arêtes consiste à échanger les arêtes de deux triangles adjacents si cela permet de réduire la surface totale de maillage. En effet, si on a deux triangles t_1 et t_2 qui partagent une arête e, on peut les échanger si cela permet de réduire la surface totale de maillage. (cf. Figure 4.3). Cette figure illustre l'échange d'arêtes entre les triangles ABC et ADC, on a $S_{ABC} + S_{ADC} \geq S_{ADB} + S_{CDB}$. Donc l'arête AC est donc échangée avec l'arête BD. Cet échange d'arêtes permet de remplacer les deux triangles ABC et ADC par les triangles ADB et CDB qui ont une surface totale plus petite.

Cette méthode est appliquée de manière itérative à chaque triangle du maillage. Étant donné que la comparaison de surface s'effectue uniquement entre le triangle considéré et ses voisins immédiats (au maximum trois voisins), la complexité de l'algorithme reste linéaire par rapport au nombre total de triangles dans le maillage, qui est par construction linéaire par rapport à la taille de la boucle de singularité. Ainsi, la complexité des échanges d'arêtes est également en $O(n)$, pour une boucle de singularité Γ de taille n.

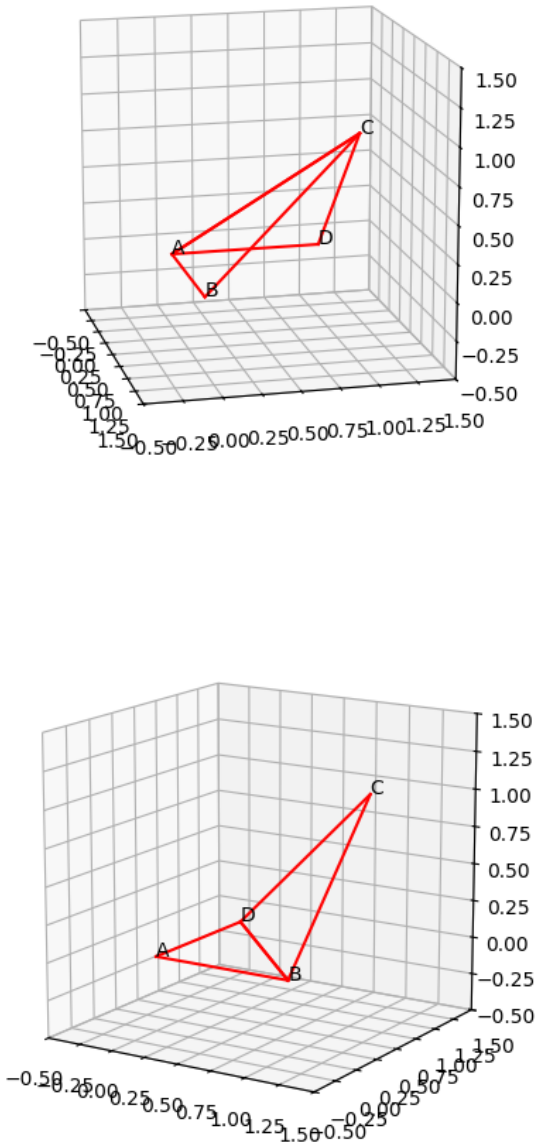


FIGURE 4.3 Échange d'arêtes entre les deux triangles ABC et ADC.

Conclusion :

On introduit un hyperparamètre K, appelé le **facteur de convergence**, qui détermine le nombre d'itérations où les deux méthodes — la méthode de lissage de Laplace et celle d'échange d'arêtes — sont appliquées de manière alternée. Ce paramètre contrôle directement le nombre total d'itérations de l'algorithme, permettant de trouver un équilibre entre l'amélioration de la qualité du maillage et le coût en temps de calcul.

Le maillage initial est soumis à K itérations alternées de lissage de Laplace et d'échanges d'arêtes. Pour une boucle Γ de taille n, la complexité par itération est $O(n)$, puisque le nombre de sommets et d'arêtes est linéaire par rapport à n. Ainsi, la complexité totale pour K itérations est $O(nK)$. Afin d'optimiser le lissage pour chaque boucle de singularité Γ de taille n, nous définissons un facteur K qui dépend de la taille de la boucle. Les expériences menées dans la section 5 montrent qu'un choix de $K(n) = 10 \cdot \log(n) + 10$ offre de bons résultats pour minimiser l'aire de la surface de maillage. En substituant $K(n)$ dans l'expression de la complexité, on obtient une complexité totale de :

$$O(n \cdot K(n)) = O(n \cdot (10 \cdot \log(n) + 10)) = O(n \log(n)).$$

Ainsi, la complexité finale de l'algorithme pour une boucle Γ de taille n est en $O(n \log(n))$.

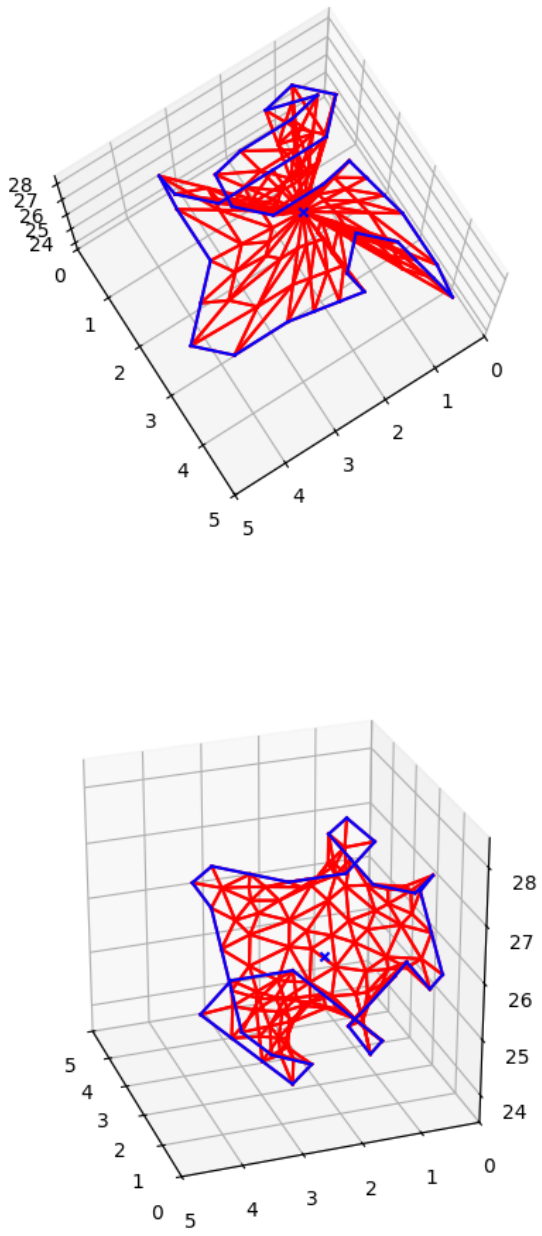


FIGURE 4.4 Après $K = 20$ itérations, l'aire de la surface est passée de 23.545 à 16.327.

4.2 Surface d'erreur optimale :

Après avoir obtenu une première surface initiale à base de maillage, on va détecter toutes les arêtes de G_{grid} qui traversent les triangle de cette surface-ci.

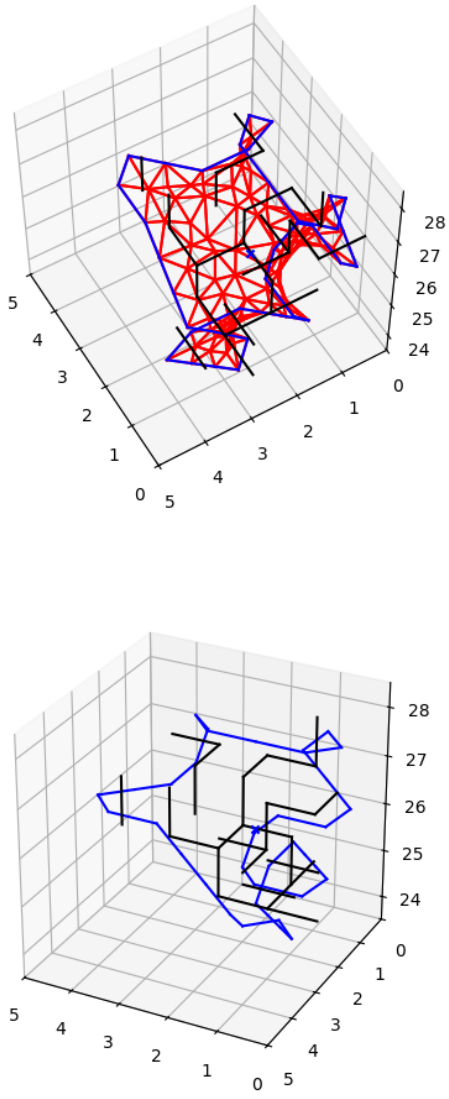


FIGURE 4.5 La surface d'erreur étant les arêtes en noir qui traversent les triangles de la surface fermante en rouge.

Ces arêtes forment une surface d'erreur valide, mais pas nécessairement optimale. La première surface est minimisée selon une métrique euclidienne, par exemple, l'aire totale des triangles qui composent le maillage. En revanche, la deuxième surface est optimisée pour minimiser le nombre d'arêtes traversées, ce qui est un objectif différent.

Jusqu'à présent, nous savons qu'il est possible de construire une surface d'erreur valide pour une boucle fermée Γ en se basant un maillage triangulaire initiale. L'étape suivante consiste à identifier une propriété fondamentale qui relie toutes les surfaces d'erreur possibles associées à Γ. Cette propriété permettra de poser des contraintes dans un problème d'optimisation visant à déterminer une surface d'erreur optimale à partir d'une surface d'erreur initiale.

Theorem 4.1. *Soit Γ une boucle fermée, et M_1 et M_2 deux surfaces fermantes de Γ. Ainsi, si un cycle fermée C traverse M_1* ***un nombre impair de fois****, alors C traverse M_2* ***un nombre impair de fois également****.*

Démonstration. Puisque M_1 et M_2 ont le même bord Γ, nous avons :

$$\partial M = \partial(M_1 - M_2) = \partial M_1 - \partial M_2 = \Gamma - \Gamma = 0.$$

Donc, M est une chaîne fermée sans bord, c'est-à-dire un 2-cycle dans $\mathbb{R}^3$.

Dans l'espace $\mathbb{R}^3$, le second groupe d'homologie $H_2(\mathbb{R}^3)$ est trivial, ce qui signifie que tout 2-cycle est le bord d'une 3-chaîne [49]. Autrement dit, il existe une 3-chaîne W telle que :

$$\partial W = M.$$

Étant donné que C est un cycle fermé de dimension 1 (une courbe fermée) dans $\mathbb{R}^3$, nous pouvons considérer le nombre d'intersections entre C et M.

Le principe fondamental utilisé est que le nombre d'intersections modulo 2 entre un cycle fermé C et le bord d'une chaîne de dimension supérieure est nul [49] :

$$I(C, \partial W) \equiv 0 \mod 2.$$

Comme $\partial W = M$, cela donne :

$$I(C, M) \equiv 0 \mod 2.$$

De plus le nombre d'intersections entre C et M est la somme des intersections entre C et M_1 et entre C et M_2 :

$$I(C, M) = I(C, M_1) + I(C, M_2).$$

$$I(C, M) \equiv 0 \mod 2.$$

Ainsi,

$$I(C, M_1) - I(C, M_2) \equiv 0 \mod 2.$$

Ce qui implique que :

$$I(C, M_1) \equiv I(C, M_2) \mod 2.$$

Par conséquent, si $I(C, M_1)$ est impair, alors $I(C, M_2)$ est également impair.

Cela conclut la preuve du théorème.

□

On admet que ce théorème reste valable pour les surfaces d'erreur, qui sont en réalité représentées comme des objets discrets dans l'espace euclidien, formés par des ensembles d'arêtes du graphe $G_{grid}(\Phi)$. Ces arêtes constituent une discrétisation des surfaces fermantes continues, qui préserveraient leurs propriétés topologiques essentielles.

Theorem 4.2. ***Pour les surfaces d'erreur :*** *Soit* Γ *une boucle fermée, et* S_1 *et* S_2 *deux surfaces d'erreur de* Γ. *Ainsi, si un cycle fermée* C *de* G_{grid} *traverse* S_1 ***un nombre impair de fois****, alors* C *traverse* S_2 ***un nombre impair de fois également****.*

$$|C \cap S_1| \equiv |C \cap S_1| \mod 2.$$

Ainsi, en disposant d'une surface d'erreur S_1 pour la boucle fermée Γ, nous pouvons caractériser l'ensemble des surfaces d'erreur possibles dans $G_{grid}(\Phi)$ associées à Γ comme étant celles qui partagent cette propriété modulo 2 pour tous les cycles C du graphe $G_{grid}(\Phi)$. Cette caractérisation fondamentale nous permet de formuler les contraintes du problème de minimisation. En exploitant le Théorème 4.2, nous visons à déterminer une surface d'erreur optimale, minimisant le nombre d'arêtes traversées, à partir de la surface d'erreur valide initiale S_1.

4.3 Problème linéaire de minimisation de la surface d'erreur :

Soit Γ une boucle fermée, et S_1 une surface d'erreur valide associée à Γ.

On pose

$$C_\Gamma := \{C \text{ cycles de } G_{grid}(\Phi) \mid |C \cap S_1| \equiv 1 \mod 2\}$$

On pose la variable de décision X_e qui vaut 1 si l'arête e est dans la surface d'erreur optimale, et 0 sinon.

On pose le problème de minimisation suivant :

$$\begin{aligned} \min_{X} \quad & \sum_{e \in E_{grid}} X_e \\ \text{s.t.} \quad & \sum_{e \in C} X_e \equiv 1 \mod 2, \quad C \in C_\Gamma, \\ & X_e \in \{0,1\}, \qquad\qquad e \in E_{grid}. \end{aligned}$$

Afin de linéariser le problème, on va ajouter les variables b_C pour tout $C \in C_\Gamma$ qui vont permettre de linéariser la modularité dans les contraintes.

On pose donc le nouveau problème linéaire à valeurs entières suivant :

$$\begin{aligned} \min_{X,b} \quad & \sum_{e \in E_{grid}} X_e \\ \text{s.t.} \quad & \sum_{e \in C} X_e - 2b_C = 1, \quad C \in C_\Gamma, \\ & X_e \in \{0,1\}, \qquad\qquad e \in E_{grid}, \\ & b_C \in \mathbb{N}, \qquad\qquad C \in C_\Gamma. \end{aligned} \tag{PLNE}$$

Cependant, pour le graphe $G_{grid}(\Phi)$, le nombre de cycles possibles est exponentiel en n. Le nombre de cycles possibles est en fait $O(2^{|V_{grid}(\Phi)|})$ [50]. Ainsi, il faut trouver une méthode qui peut résoudre ce problème en respectant les contraintes computationnelles et temporelles. On va donc faire appel à la méthode de **séparation des contraintes**.

4.3.1 Méthode de séparation des contraintes :

La méthode de séparation des contraintes consiste à rajouter les contraintes C_Γ, une par une, et s'arrêter dès qu'une solution, qui respecte toutes les contraintes, est trouvée [51].

En effet, cette méthode intitulée **la séparation des contraintes** reste très efficace en optimisation combinatoire quand on ne peut pas énumérer toutes les contraintes du modèle [52]. Cependant, deux conditions doivent être respectées pour que cette méthode soit efficace :

1. Il faut qu'il soit possible de trouver une condition d'arrêt équivalente pour vérifier la faisabilité de la solution intermédiaire, et ce, sans énumérer toutes les contraintes. Cette condition d'arrêt est donc souvent dynamique.
2. Les sous problèmes résolus doivent être moins complexes en termes de taille des contraintes que le problème initial.

Posons donc le sous problème $PLNE_i$ qui ne conserve que i contraintes de C_Γ, donc un

ensemble qu'on fixera $C_{\Gamma,i}$. On va donc appeler la solution de ce sous problème X^i.

$$
\begin{aligned}
\min_{X,b} \quad & \sum_{e \in E_{grid}} X_e & & \\
\text{s.t.} \quad & \sum_{e \in C} X_e = 1 + 2b_C & \forall C \in C_{\Gamma,i} & \\
& X_e \in \{0,1\} & e \in E_{grid} & \\
& b_C \in \mathbb{N} & C \in C_{\Gamma,i} &
\end{aligned}
\tag{$PLNE_i$}
$$

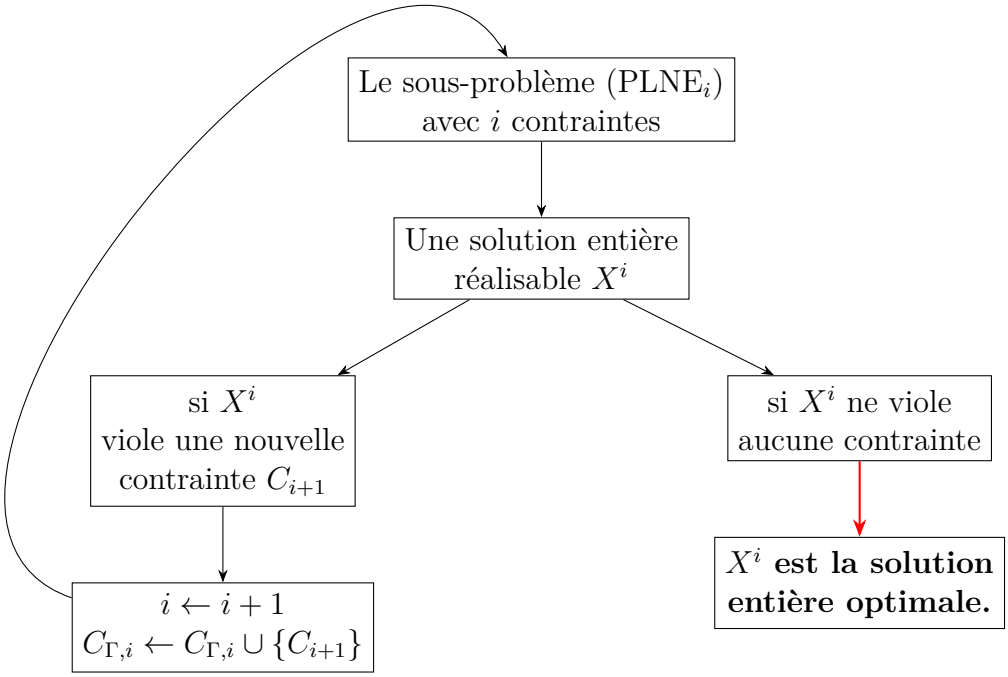


FIGURE 4.6 On continue à résoudre et mettre à jour $PLNE_i$ jusqu'à ce que la solution X^i ne viole aucune des contraintes de C_Γ.

4.3.2 La condition d'arrêt :

Supposons qu'on a une boucle fermée Γ avec la surface d'erreur S_1 et que X^i est la solution entière réalisable du sous problème $PLNE_i$ à l'étape i. On va donc regarder dans le graphe intermédiaire :

$$G_i = G_{grid}(\Phi) \setminus \{e \in E_{grid} \mid X^i_e = 1\}$$

soit le graphe $G_{\text{grid}}(\Phi)$ privé des arêtes appartenant à la solution actuelle X^i. Notre objectif est de vérifier s'il existe un cycle C dans G_i qui contient un nombre impair d'arêtes de S_1.

1. **Si un tel cycle C est trouvé**, cela signifie que la solution réalisable X^i viole une contrainte, car ce cycle traverse un nombre impair d'arêtes de S_1, mais n'est pas bloqué par X^i. Nous ajoutons alors une nouvelle contrainte correspondant à ce cycle au problème $PLNE_i$ et résolvons à nouveau.
2. **Si aucun tel cycle C n'est trouvé**, cela signifie que X^i satisfait toutes les contraintes imposées par S_1, et la solution X^i est alors optimale.

En pratique, cette méthode repose sur une détection efficace de cycles spécifiques dans le graphe intermédiaire G_i, ce qui permet d'assurer que toutes les contraintes pertinentes sont prises en compte sans reconstruire ou analyser l'ensemble des cycles possibles.

Les bases de cycles :

Definition 4.2. *Soit $G = (V, E)$ un graphe. Une **base de cycles** [50] de G est un ensemble minimal de cycles $\{C_1, C_2, \ldots, C_k\}$ tels que tout cycle C de G peut être exprimé comme une combinaison linéaire des C_i :*

$$C = \bigoplus_{i \in I} C_i$$

où $I \subseteq \{1, 2, \ldots, k\}$ et $\bigoplus$ représente la différence symétrique des ensembles d'arêtes.

*La taille d'une base de cycles, aussi appelée **rang cyclomatique** [53], est donnée par la formule suivante :*

$$\beta(G) = m - n + c$$

où m est le nombre d'arêtes, n est le nombre de sommets, et c est le nombre de composantes connexes du graphe.

Posons la fonction f_{S_1} qui associe à chaque cycle C de G_i le nombre d'arêtes de S_1 qu'il traverse (modulo 2).

$$\begin{aligned} f_{S_1} : P(E) &\to \mathbb{Z}_2 \\ b &\mapsto \sum_{c \in b} e \in S_1 \end{aligned}$$

Proposition 4.1. *Soit C_1, C_2 deux cycles de G_i, alors*

$$f_{S_1}(C_1 \bigoplus C_2) = f_{S_1}(C_1) + f_{S_1}(C_2)$$

Démonstration. Soient $B(C_1), B(C_2), B(C_1 \bigoplus C_2)$ les ensembles des arêtes de S_1 qui se retrouvent respectivement dans C_1, C_2 et $C_1 \bigoplus C_2$.

Il est facile de démontrer que :

$$B(C_1 \bigoplus C_2) = B(C_1) \bigoplus B(C_2)$$

Donc dans $\mathbb{R}$,

$$\begin{aligned}\text{Card}\left(B(C_1 \bigoplus C_2)\right) &= \text{Card}\left(B(C_1) \bigoplus B(C_2)\right) \\ &= \text{Card}\left(B(C_1)\right) + \text{Card}\left(B(C_2)\right) \\ &\quad - 2\,\text{Card}\left(B(C_1) \cup B(C_2)\right)\end{aligned}$$

Par conséquent, dans $\mathbb{Z}_2$,

$$\text{Card}\left(B(C_1 \bigoplus C_2)\right) = \text{Card}\left(B(C_1)\right) + \text{Card}\left(B(C_2)\right)$$

Puisque dans $\mathbb{Z}_2$, $f_{S_1}(C) = \text{Card}(B(C))$, cela conclut la preuve.

□

Chaque cycle de G_i peut être exprimé comme une combinaison linéaire des cycles de la base de cycles de G_i [50].

Theorem 4.3. *Soit B une base de cycles de G_i, et définissons $B_1 := \{b \in B \mid f_{S_1}(b) = 1\}$. Alors, pour tout cycle C de G_i, il existe un ensemble B_C de cycles dans B tel que $C = \bigoplus_{b \in B_C} b$. On a l'égalité suivante :*

$$f_{S_1}(C) = \sum_{b \in B_C} f_{S_1}(b) = \sum_{b \in B_C \cap B_1} 1 \mod 2$$

Ainsi, la condition d'arrêt est atteinte lorsqu'il n'est plus possible de trouver des cycles C tels que $f_{S_1}(C) = 1$ dans le graphe G_i. Et selon le théorème 4.3, on atteint la condition d'arrêt, quand on obtient un graphe G_i dont $B_1 = \emptyset$.

Proposition 4.2. *Soit B une base de cycles de G_i, et $B_1 := \{b \in B \mid f_{S_1}(b) = 1\}$.*

Si $B_1 = \emptyset$, alors X^i ne viole aucune contrainte dans $C_{\Gamma,i}$ et X^i est une solution entière optimale du problème $PLNE_i$.

4.3.3 La condition violée :

Pour ce qui concerne la condition violée, n'importe quel élément de B_1 est un cycle qui représente une contrainte violée.

Ainsi, l'algorithme de résolution du problème $PLNE$ est le suivant : (cf. Figure 4.7)

Pour illustrer comment le problème linéaire en nombres entiers contribue à la résolution, nous considérons l'exemple d'une boucle extraite des données synthétiques présentées à la section 5. La Figure 4.8 montre les différentes étapes de la construction de la surface d'erreur optimale pour cette boucle.

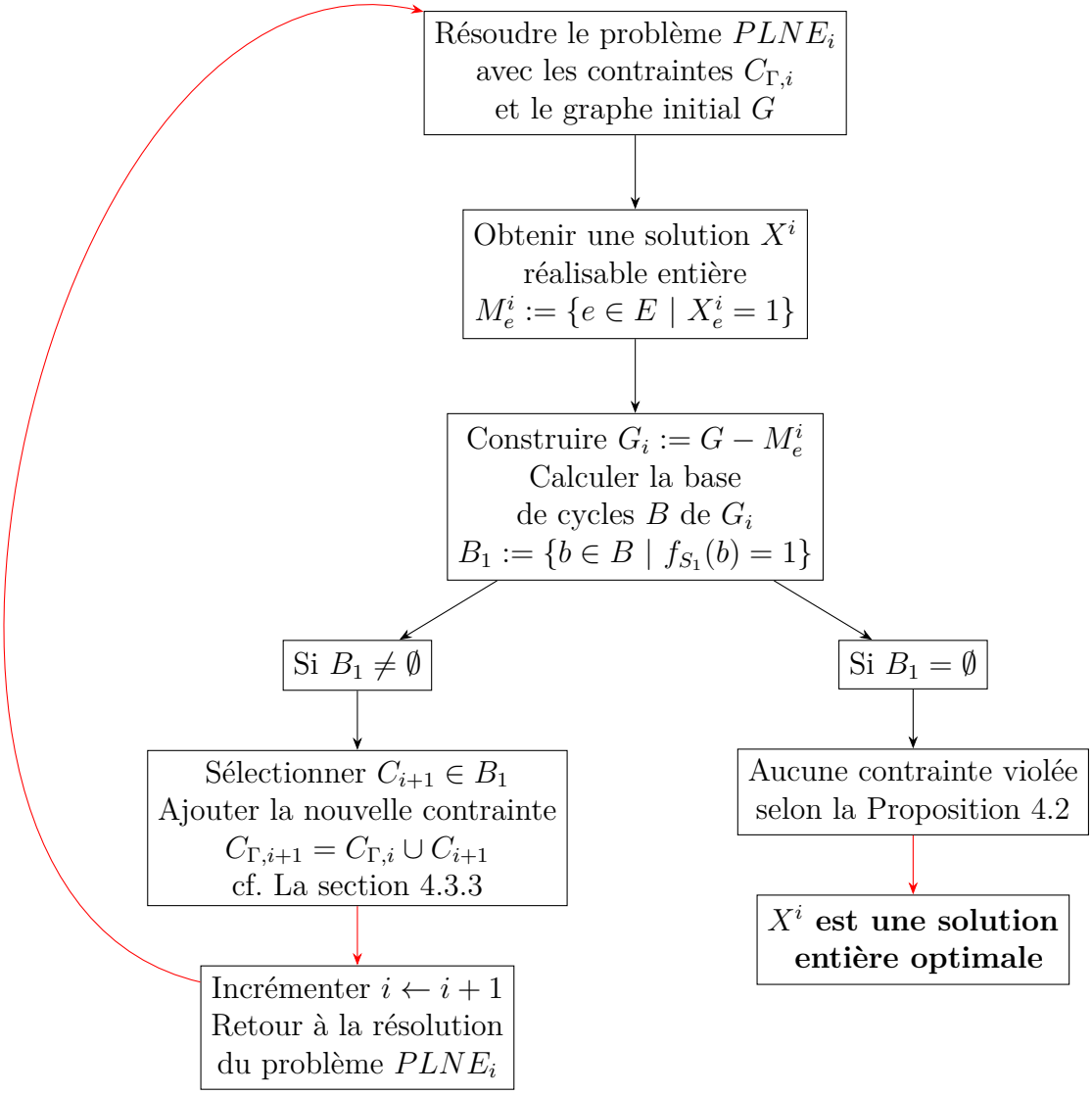
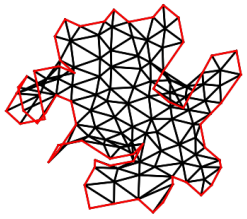
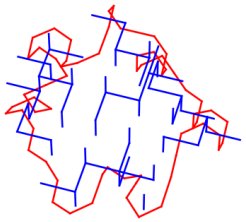


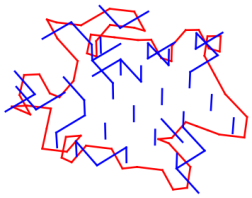
FIGURE 4.7 Algorithme de séparation des contraintes pour trouver une solution optimale entière.



(a) Construction du maillage.



(b) Surface d'erreur obtenue à partir du maillage initial (67 arêtes).



(c) Surface d'erreur obtenue à partir de la résolution du problème entier associée à la boucle (46 arêtes).

Paramètre	**Valeur**
`GUROBI version`	11.0.1
Threads	64
Lazy Constraints [54]	1
OS	Rocky Linux 8.10
CPU	AMD EPYC 7532
Cœurs physiques	64
Variables	1474 binaires
Temps total	468.75 s
Nœuds	12,242

(d) Cadre expérimental de la résolution du problème en nombres entiers.

FIGURE 4.8 Illustration du remplissage d'une boucle obtenue à partir des données synthétique.

4.4 Implémentation de l'algorithme de détection de la base des cycles :

4.5 Le chemin d'intégration :

Après avoir rempli de manière optimale les boucles de singularité avec des surfaces d'erreur, nous procédons à l'intégration du signal en **minimisant le nombre de passages à travers les arêtes** de ces surfaces d'erreur. Dans le cadre de la méthode choisie, nous parcourons le graphe G_{grid} à partir d'un point source, en assignant des poids élevés aux arêtes correspondant aux surfaces d'erreur optimisées et des poids faibles aux autres arêtes. Notre objectif est de construire un **arbre couvrant minimal** du graphe G_{grid}, en utilisant un algorithme tel que celui de Prim [46], afin de minimiser le nombre de passages à travers les arêtes indésirables.

L'intégration du chemin dans ce contexte se base sur l'accumulation des différences de phase de 2π. En effet, si l'on définit deux nœuds u et v dans le graphe G_{grid}, et si l'on suppose que la phase à u est déjà déroulée, nous utilisons la formule suivante [7, 21, 30] pour dérouler :

$$\Psi_{\text{unwrap}}(v) = \Psi_{\text{unwrap}}(u) + W(\Delta\Phi) \tag{4.1}$$

où $\Delta\Phi = \Phi(v) - \Phi(u)$ est la différence de phase enroulée, et $W(\Delta\Phi)$ est l'opérateur d'enroulement qui ramène la différence dans l'intervalle $(-\pi, \pi]$, défini par :

$$W(\Delta\Phi) = \Delta\Phi - 2\pi \cdot \text{round}\left(\frac{\Delta\Phi}{2\pi}\right)$$

Ainsi, la construction d'un arbre couvrant minimal du graphe G_{grid}, minimisant les passages à travers les arêtes des surfaces d'erreur, établit la base sur laquelle l'intégration du signal sera effectuée. Cette approche, en assignant des poids appropriés aux arêtes, permet de réduire l'impact des surfaces d'erreur sur le déroulement de phase, tout en maintenant une complexité algorithmique de l'ordre de $O(|E_{\text{grid}}| \log |V_{\text{grid}}|)$, compte tenu de l'utilisation d'algorithmes d'arbres couvrants minimaux.

4.6 Conclusion :

Le tableau 4.1 récapitule les différentes étapes de l'algorithme ainsi que leur complexité.

TABLEAU 4.1 Tableau des procédés et de leurs complexités avec définition des paramètres

Procédés	**Complexité**
Étape 1 : Construction de maillage et détection des arêtes initiales	
Construction de maillage initial pour une boucle Γ de taille n	$O(3n+1) = O(n)$
Minimisation de la surface de maillage pour une boucle Γ de taille n	
- Méthode de lissage de Laplace.	$O(K(n) \times n) = O(nlog(n))$
- Méthode d'échange d'arêtes.	$O(K(n) \times n) = O(nlog(n))$
Détection des arêtes traversant le maillage (pour obtenir la surface d'erreur initiale)	$O(\|E_{grid}(\Phi)\|)$
Étape 2 : Minimisation de la surface d'erreur	
1. Méthode de séparation des contraintes	
- Résolution du problème $PLNE_i$ à chaque itération	NP-difficile
- Sous-parties de la détection des cycles violant les contraintes :	
– a. Construction de l'arbre couvrant T_i du sous-graphe G_Γ	$O(\|V_{grid}(\Phi)\| + \|E_{grid}(\Phi)\|)$
– b. Prétraitement pour trouver les ancêtres communs avec l'Euler Tour (LCA)	$O(\|V_{grid}(\Phi)\| \log \|V_{grid}(\Phi)\|)$
– c. Construction de la base des cycles B_1	$O(\|E_{grid}(\Phi)\|)$
- Complexité totale de la détection des cycles pour chaque itération de séparation des contraintes	$O(\|V_{grid}(\Phi)\| \log \|V_{grid}(\Phi)\| + \|E_{grid}(\Phi)\|)$ par itération.
Étape 3 : Intégration du chemin	
Intégration du signal	$O(\|V_{\text{grid}}(\Phi)\| + \|E_{\text{grid}}(\Phi)\|)$

CHAPITRE 5 RÉSULTATS

5.1 Objectifs des analyses expérimentales :

Les analyses expérimentales réalisées dans cette étude ont pour objectif principal d'évaluer en profondeur l'efficacité et la robustesse de la méthode proposée dans ce mémoire pour le déroulement de la phase en trois dimensions, et qui est basée sur le remplissage des boucles de singularités. Pour valider la performance de notre méthode, nous allons compter sur l'analyse de l'aspect visuel des images déroulées. Le but étant de détecter toutes les possibles distorsions visuelles qui vont venir perturber la continuité spatiale de l'image déroulée. Parallèlement, nous allons aussi également se baser sur une métrique de qualité de déroulement qui a été utilisée dans la littérature à savoir l'erreur L_0. Cette métrique correspond à la norme L_0 de différence des gradients entre les phases roulée et déroulée, offrant un indicateur objectif de la précision de la reconstruction.

En outre, nous allons également analyser en détail la performance de la méthode proposée lorsque cette dernière est appliquée à des données empiriques. En effet, grâce à l'amélioration portée sur la détection des boucles de singularités, nous arrivons à détecter des boucles bien plus petits et bien moins complexes ce qui facilitera donc le remplissage des surface d'erreurs. L'analyse expérimentale va également permettre de mettre en avant les performances de la minimisation de la surface d'erreur proposée, qui en effet consiste à formuler le problème comme un problème linéaire en nombres entiers.

Finalement, un point clé qui apparaîtra principalement dans notre analyse expérimentale, et de manière moins marquée dans l'analyse théorique, concerne la robustesse de la méthode proposée, en particulier face à des données empiriques bien bruitées. Bien que notre objectif soit de minimiser l'erreur de manière optimale, il est évident que pour des données de grande taille avec beaucoup de bruit, il est nécessaire de faire des compromis afin de permettre à ce que l'exécution de la méthode reste faisable dans des conditions computationnelles réelles. La minimisation doit intégrer les contraintes physiques, computationnelles, et temporelles, qui influencent directement l'applicabilité de la méthode dans des contextes réels. Tout au long de notre analyse théorique et de la description de la méthode, nous avons veillé à aborder régulièrement la question de la complexité algorithmique, afin de nous assurer que chaque choix reste justifié et faisable.

Pour cette étude, les expériences de référence seront effectuées avec l'approche *ROMEO* [45]. Nous avons ensuite élaboré une solution hybride combinant *ROMEO* et notre méthode.

Cette combinaison s'est avérée stratégique, car *ROMEO* est simple à implémenter et assez performant surtout vu que ca prend en considération les contraintes physiques liées aux données de l'imagerie par résonance magnétique. Quant à notre méthode, bien que plus rigoureuse, est parfois trop exigeante en termes de ressources, et se base uniquement sur la théorie des résidus et des boucles de singularités.

5.2 Génération et Collections des données :

5.2.1 Génération des données :

Les données synthétiques seront définies par un fonction $F(x, y, z)$, avec x, y, z des entiers dans l'ensemble $[0, 63]$ pour x et y, et $[0, 30]$ pour z.

La fonction $F(x, y, z)$ est définie comme suit :

$$F(x, y, z) = 10 \times \left[\sigma_1(z) \cdot \frac{\sin(u(x))}{u(x)} + \sigma_2(z) \cdot \frac{\sin(v(y))}{v(y)}\right] + N(x, y, z)$$

où $u(x)$ et $v(y)$ sont des fonctions définies comme suit :

$$u(x) = x - 31.5$$
$$v(y) = y - 31.5$$

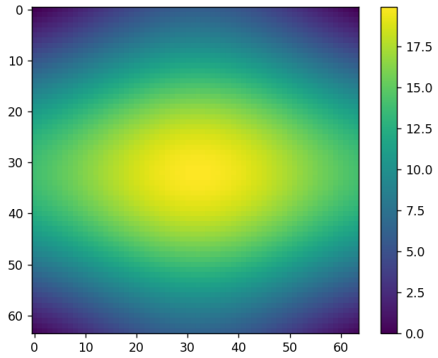
Le but des fonction u et v est de centrer les coordonnées x et y autour du centre de l'image $(31.5, 31, 5)$. Les fonctions σ_1 et σ_2 sont définies comme suit :

$$\sigma_1(z) = 1.50 - (0.01 \times (z + 1))$$
$$\sigma_2(z) = 0.49 + (0.01 \times (z + 1))$$

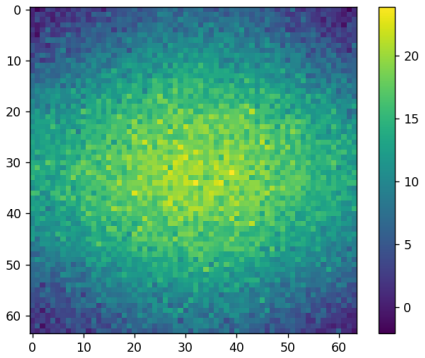
Les fonctions σ_1 et σ_2 permettent de modifier la forme de la surface en fonction de l'ordre de l'image. Afin de mieux perturber les données, un bruit $N(x, y, z)$ est ajouté à la phase, ce bruit est défini par :

$$N(x, y, z) = \begin{cases} 0, & \text{si } z < 15 \\ \mathcal{N}(0, 1), & \text{si } z \geq 15 \end{cases}$$

où $\mathcal{N}(0, 1)$ représente un bruit gaussien de moyenne nulle et d'écart-type 1.

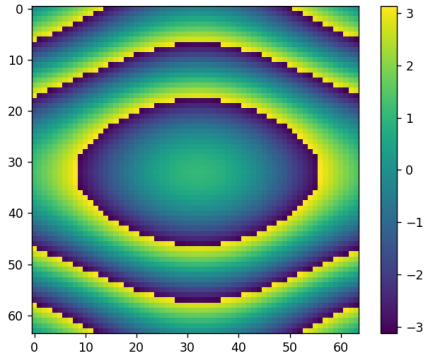


(a) Illustration pour $z = 10$.

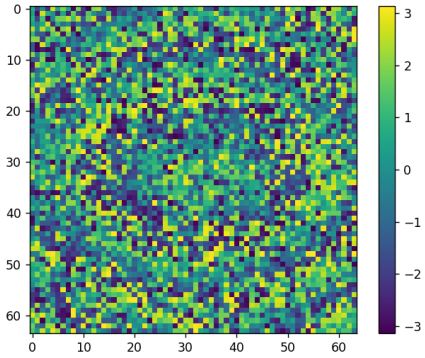


(b) Illustration pour $z = 20$.

FIGURE 5.1 Illustration des données synthétiques avant leur enroulement.



(a) Illustration pour $z = 10$.



(b) Illustration pour $z = 20$.

FIGURE 5.2 Illustration des données synthétiques après leur enroulement.

Après avoir généré les données synthétiques, nous avons appliqué l'enroulement de phase pour les restreindre à l'intervalle $[-\pi, \pi]$. Cette opération est réalisée en appliquant la fonction d'enroulement définie par :

$$F_{\text{wrapped}}(x, y, z) = (F(x, y, z) + \pi) \bmod 2\pi - \pi$$

Les figures 5.1 et 5.2 illustrent cet effet en montrant les données synthétiques avant et après l'enroulement, respectivement. Les valeurs de la phase dans la figure initiale varient entre `-5.05` et `23.99`. Or, après l'enroulement,

Après avoir généré les données synthétiques, nous avons appliqué l'enroulement de phase pour les restreindre à l'intervalle $[-\pi, \pi]$. Cette opération est réalisée en appliquant la fonction d'enroulement définie par :

$$F_{\text{wrapped}}(x, y, z) = (F(x, y, z) + \pi) \bmod 2\pi - \pi$$

Les figures 5.1 et 5.2 illustrent cet effet en présentant les données synthétiques avant et après l'enroulement, respectivement. Les valeurs de la phase dans la figure 5.1 varient entre `-5.05` et `23.99`, tandis qu'après l'enroulement, les valeurs de la phase F_{wrapped} sont restreintes entre `-3.14` et `3.14`. De plus, les images 5.2a et 5.1a montrent les données synthétiques pour $z = 10$, dans la zone de la phase où le bruit n'est pas encore ajouté. Quant aux images 5.2b et 5.1b, montrent les données synthétiques pour $z = 20$, dans la zone de la phase où le bruit est ajouté.

5.2.2 Collection des données :

En plus des données synthétiques, nous avons également utilisé des données expérimentales provenant de sources ouvertes. Le but étant de valider la robustesse et la scalabilité de la méthode proposée dans des conditions réelles, et bien plus complexes qu'un simple jeu de données synthétiques. Les données choisies proviennent de la plateforme *Harvard Dataverse*.

Plus précisément , nous avons utilisé la carte de phase 3D *GRE_7T*, qui est en effet également utilisée par les chercheurs qui ont conçu l'algorithme *ROMEO* [45] pour valider leur méthode de déroulement de phase. Les données empiriques choisies sont assez pertinentes pour tester l'efficacité de notre méthode, car elles se caractérisent par des variations topographiques complexes et des niveaux de bruit significatifs. En outre, le fait que ces données aient été utilisées pour la validation de l'algorithme *ROMEO* dans l'étude de [45] souligne leur pertinence pour notre travail. En effet, ces auteurs ont démontré que *ROMEO* est une

méthode rapide et efficace pour le déroulement de phase sur des données bruitées. Ainsi, en utilisant ces mêmes données, nous visons à tester notre méthode non seulement sur des données synthétiques, mais également sur des données empiriques, comme ils l'ont fait. Leur méthode présente l'avantage de détecter et de gérer le bruit (cf. 2) ; il est donc naturel qu'ils aient choisi des données fortement bruitées pour démontrer l'efficacité de leur approche. Nous avons donc adopté une démarche similaire.

Les principales caractéristiques de ce jeu de données sont résumées dans le tableau 5.1.

TABLEAU 5.1 Caractéristiques du jeu de données *7T_GRE* utilisé dans cette étude.

Titre	7T_GRE
Auteur	Dymerska, Barbara (University College London)
Identifiant	https ://doi.org/10.7910/DVN/6SWKI0
Date de publication	28 février 2020
Description	Jeu de données 7 T 3D GRE
Sujet	Médecine, Informatique, Physique
Déposant	Dymerska, Barbara
Date de dépôt	3 février 2020

La figure 5.3 illustre une tranche 2D des données empiriques enroulées, correspondant au plan $z = 50$.

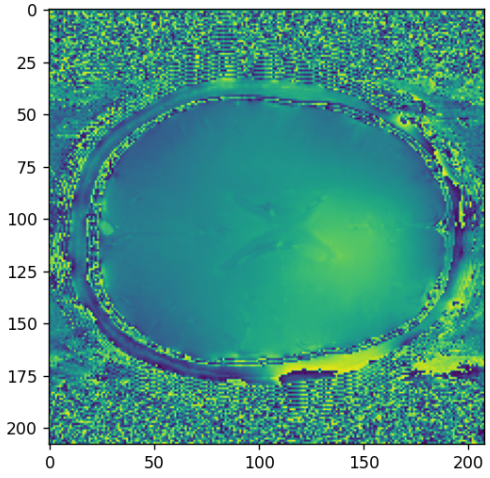


FIGURE 5.3 Tranche 2D des données empiriques enroulées, pour $z = 50$.

5.3 Le cadre expérimental :

Le cadre expérimental de cette thèse a été réalisé sur un système Linux, hébergé sur le nœud `narval3.narval.calcul.quebec`. La machine est équipée d'un processeur **AMD EPYC 7532** doté de **64 cœurs** et fonctionnant à une fréquence de **2423,56 MHz**. Selon les besoins des différentes expérimentations, l'utilisation des cœurs CPU variait entre **1 et 64 cœurs**, assurant une flexibilité optimale dans les calculs. La mémoire vive totale est de **251,35 GB**. Le stockage est géré par le système de fichiers **ZFS**, offrant une capacité totale de **1665,04 GB** ainsi qu'une partition supplémentaire de **80,00 GB**.

Sur le plan logiciel, le système d'exploitation utilisé est une distribution **Linux** avec le noyau `4.18.0-553.16.1.el8_10.x86_64`. L'environnement Python installé est la version `3.11.5`, garantissant la compatibilité et la performance des scripts et applications nécessaires aux expérimentations.

5.4 Résultats :

Avant de commencer la présentation des différentes méthodes testées, il est important de rappeler brièvement les étapes essentielles de l'algorithme proposé dans ce mémoire :

1. **Détection des résidus**.
2. **Détection des boucles de singularités**.
3. **Minimisation des surfaces d'erreur** : Pour chacune des boucles détectées, les étapes sont :
   (a) Construire la surfaces de maillage, et détecter toutes les arêtes qui la traversent afin de construire une première surface d'erreur.
   (b) Minimiser le nombre d'arêtes constituant la surface d'erreur en résolvant le problème de minimisation linéaire en nombres entiers qui est associée à la boucle.
4. **Reconstruction du signal avec l'intégration du chemin** : On intégre le signal sur tout le volume en minimisant le passage par les arêtes des surfaces d'erreur.

Dans cette section, nous comparons les performances de cinq méthodes de déroulement de phase 3D sur les données synthétiques et empiriques présentées précédemment. Les méthodes considérées sont :

1. **Méthode ROMEO (*Rapid, Optimal Multi-Echo* phase unwrapping)** [45] : En effet, la méthode ROMEO est une technique de déroulement de phase rapide qui utilise une approche locale en se basant sur les poids des arêtes du graphe. Les poids

sont calculés en fonction de la qualité du signal et des variations locales de phase. *ROMEO* ne traite pas explicitement les résidus de phase mais minimise les erreurs globales en attribuant des poids appropriés aux arêtes. Les poids vont de 1 à 255, et lors de l'intégration du chemin, l'algorithme va prioriser les arêtes avec des poids plus faibles. Le calcul de ces poids est présenté en détail dans la Section 2.

2. **Méthode de minimisation des surfaces d'erreur** : Notre méthode est basée sur la minimisation globale des surfaces d'erreur en traitant explicitement les résidus de phase.

   Notre méthode propose une approche globale basée sur la minimisation des surfaces d'erreur (ou *branch-cuts*), en traitant explicitement les résidus de phase. Dans cette méthode, nous procédons comme suit :

   (a) **Détection des arêtes problématiques :** Nous identifions les boucles de singularité, que nous fermons avec des surfaces d'erreur minimales. Les arêtes qui composent ces surfaces seront les arêtes critiques à bloquer lors de l'intégration du chemin.

   (b) **Intégration du chemin :** Nous procédons ensuite à l'intégration du chemin pour dérouler la phase, tout en attribuant un poids de 1 aux arrêtes qui font pas partie des arêtes problématiques, et un poids de 2 aux arêtes qui les composent. Le chemin d'intégration va donc prioriser le passages par les arêtes qui ne font pas partie des surfaces d'erreur.

3. **Méthode de surfaces de maillage :** Cette méthode reprend le principe de la méthode précédente, mais diffère par la détection des arêtes problématiques. Ici, seules les arêtes traversant les surfaces de maillage associées à chaque boucle sont considérées, sans résoudre le problème de minimisation. Ces arêtes sont alors identifiées comme problématiques et évitées autant que possible lors de l'intégration du chemin.

4. **Méthode combinée (*ROMEO* + Mininimisation)** : Une approche hybride qui combine les deux méthodes précédentes. On garde les poids des arêtes de *ROMEO*, et on ajoute aux arêtes minimales qui traversent les surfaces d'erreur des poids plus élevés 256. Ceci permet à ce que le chemin d'intégration minimise le plus possible le passage par les arêtes des surfaces d'erreur.

5. **Méthode combinée (*ROMEO* + Maillage)** : Dans cette méthode nous allons pareillement garder les poids attribuées aux arêtes par *ROMEO* [45], et nous attriburons aux arêtes qui traversent les surfaces de maillage des poids plus élevés 256.

Nous évaluons ces méthodes en termes de qualité de reconstruction, mesurée par l'erreur L_0, et le temps de calcul. Figure 5.4 résume les quatre méthodes de déroulement de phase

3D présentées dans ce mémoire, toutes basées sur le traitement explicite des boucles de singularité. La **méthode *ROMEO***, qui ne traite pas ces boucles, n'est pas incluse dans ce graphe.

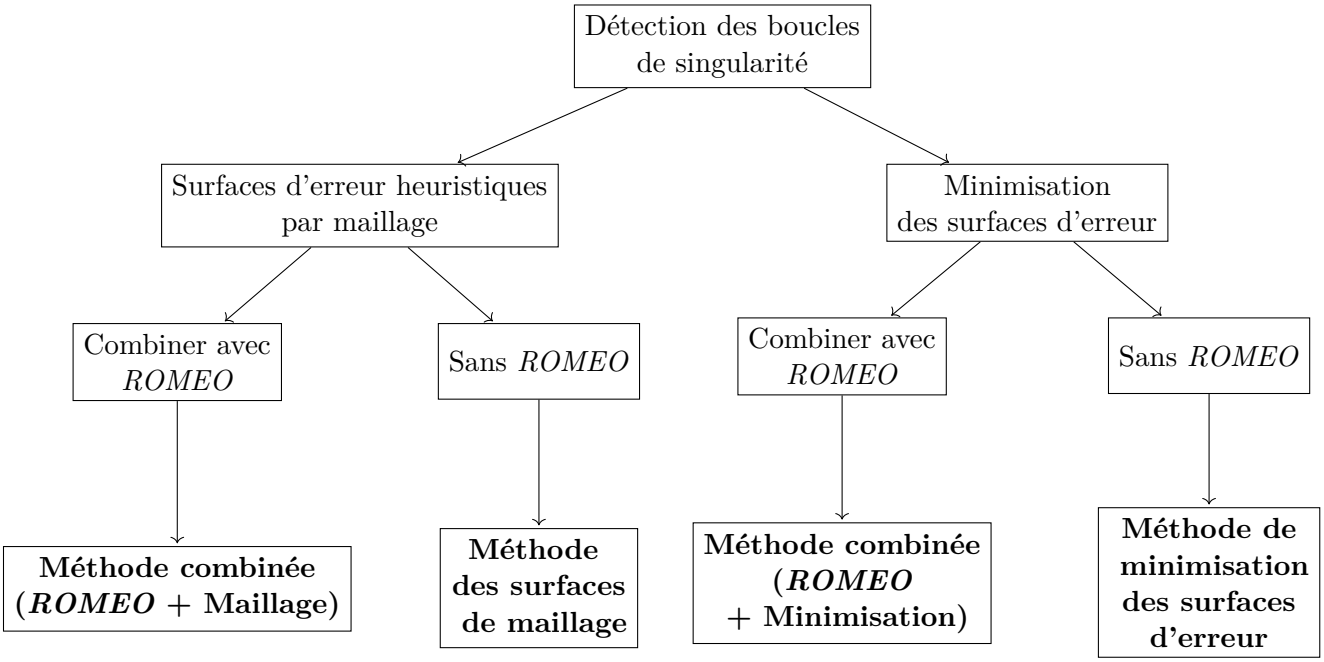


FIGURE 5.4 Graphe résumant les méthodes de déroulement de phase $3D$ utilisant les boucles de singularité et les surfaces d'erreurs.

5.4.1 Les résultats sur les données synthétiques :

Détection des résidus et construction des boucles de singularité : La détection des boucles de singularité a été fait en suivant les constructions et les algorithmes décrits en détail dans la Section 3.

Remplissage des boucles : Après l'amélioration des boucles, la plus grande boucle (ouvertes et fermées confondues) est de taille 114. Il est donc possible de résoudre le problème de minimisation qui lui est associée en quelques minutes, et ce en utilisant une station de travail puissante avec 64 coeurs CPU, et avec le threading activé lors de la résolution du problème.

La figure 4.8 montre le remplissage d'un boucle singularité obtenue à partir des données synthétiques, qui ne prends que 7.8 minutes avec 64 coeurs CPU. On peut voir que le nombre d'arêtes dans la surface obtenue après la résolution du problème en nombre entiers passe de 67 arêtes à 46 arêtes.

Le reste des boucles ouvertes et fermées ont été remplies en suivant le même procédé :

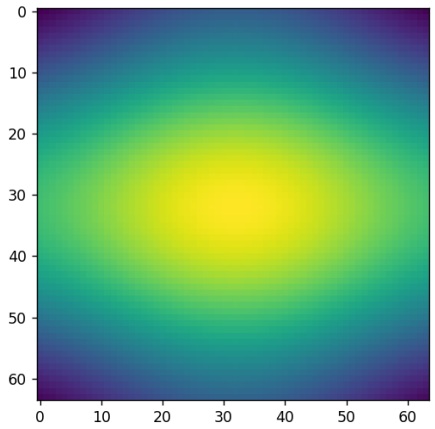
— Création du maillage avec un nombre de triangles de $K(n) = 10 \times log(n) + 10$ pour une boucle de taille n.
— Détection des arêtes qui traversent chacun des triangles, en complexité linéaire également en le nombre des triangles.
— Création d'un chemin d'intégration qui va venir dérouler la phase en évitant le passage par les arêtes qui font partie de la surface d'erreur.

Application et comparaison des méthodes :

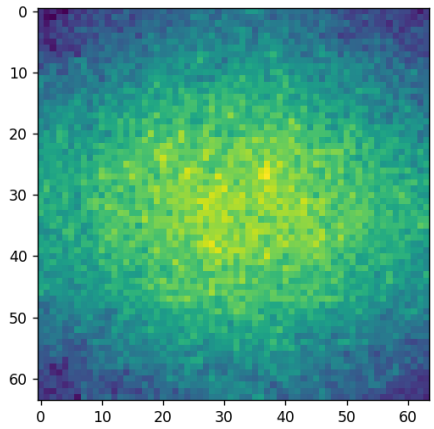
Le tableau 5.2 présente les temps de calcul détaillés pour chaque étape des différentes méthodes.

TABLEAU 5.2 Temps de calcul détaillés pour chaque étape.

Étape	**Temps de calcul**
Détection des résidus	$0,5$
Construction des boucles	$1,2$
Construction des surfaces d'erreurs initiales (maillage)	210
Minimisation des surfaces	7007
Calcul des poids	$0,2$
Intégration du chemin	$2,6$
Temps total	7304

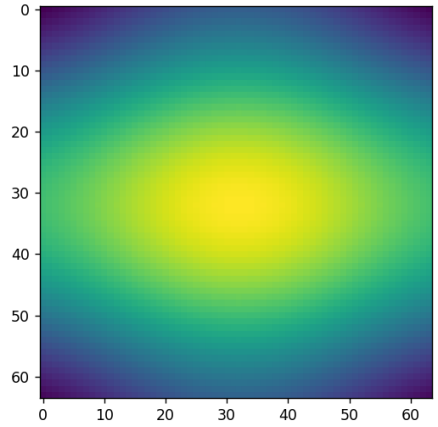


(a) La phase déroulée avec *ROMEO*, pour $z = 10$.

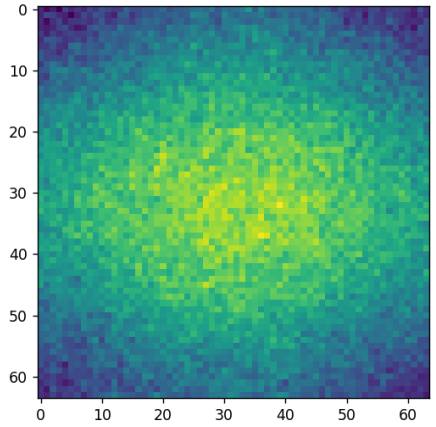


(b) La phase déroulée avec *ROMEO*, pour $z = 20$.

FIGURE 5.5 Illustration la phase déroulée par la méthode *ROMEO*.

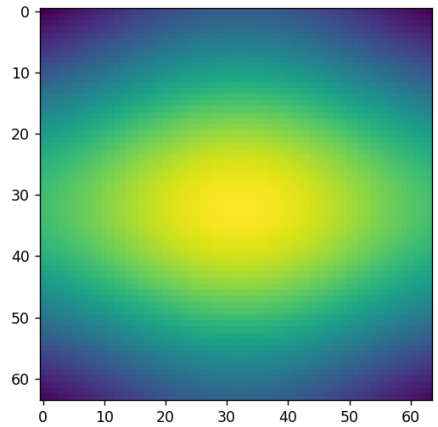


(a) La phase déroulée avec notre méthode, pour $z = 10$.

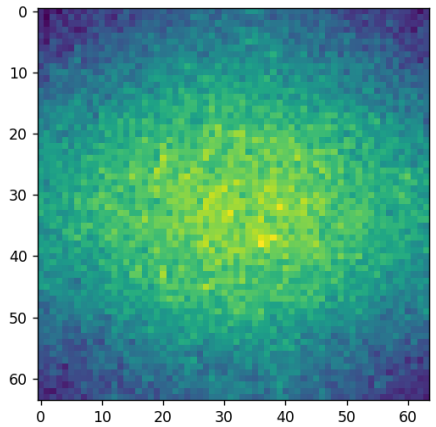


(b) La phase déroulée en utilisant notre méthode, pour $z = 20$.

FIGURE 5.6 Illustration la phase déroulée par la méthode de minimisation des surfaces d'erreur.

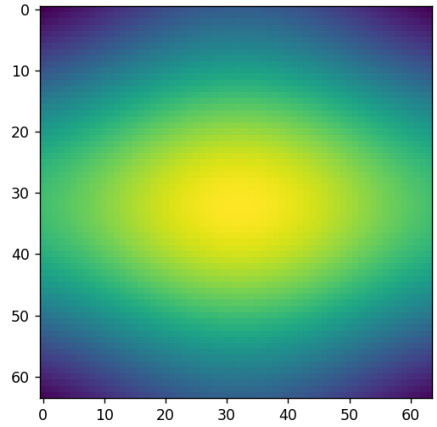


(a) La phase déroulée pour $z = 10$.

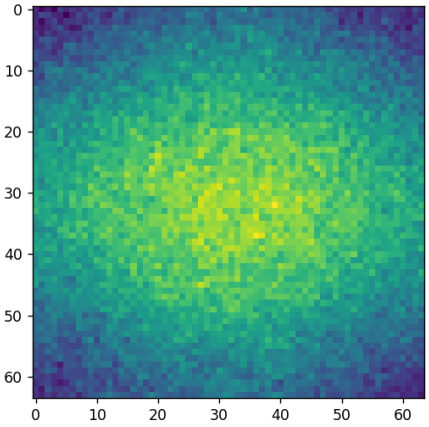


(b) La phase déroulée pour $z = 20$.

FIGURE 5.7 Illustration la phase déroulée par la méthode des surfaces de maillage.

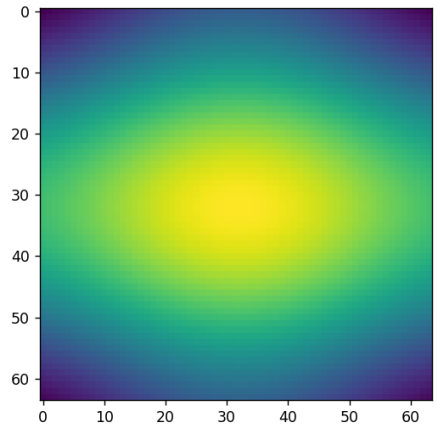


(a) La phase déroulée pour $z = 10$.

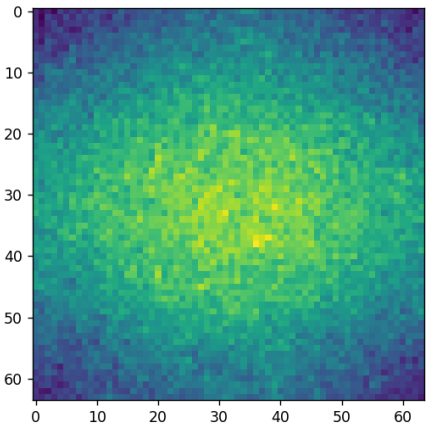


(b) La phase déroulée pour $z = 20$.

FIGURE 5.8 Illustration la phase déroulée par la méthode combinée (*ROMEO* + Minimisation).



(a) La phase déroulée pour $z = 10$.



(b) La phase déroulée pour $z = 20$.

FIGURE 5.9 Illustration la phase déroulée par la méthode combinée (*ROMEO* + Maillage).

Comparaison des performances :

Nous évaluons les performances des différentes méthodes de déroulement de phase 3D en termes de temps de calcul et de précision, mesurée par l'erreur L_0. Les résultats sont synthétisés dans le tableau 5.3.

TABLEAU 5.3 Comparaison des performances des différentes méthodes sur les données synthétiques.

Méthode	**Erreur L_0**	**Temps de calcul (s)**
ROMEO	37 174	**0,8**
Minimisation des surfaces d'erreur	36 943	7 304
Surfaces de maillage	37 166	211
Méthode combinée (*ROMEO* + Minimisation)	**36 356**	7 304
Méthode combinée (*ROMEO* + Maillage)	37 158	211

L'analyse des résultats met en évidence un compromis entre la précision du déroulement de phase et le temps de calcul requis par chaque méthode.

Méthode ROMEO : Cette méthode présente le temps de calcul le plus faible, avec seulement 0,8 seconde, grâce à sa complexité algorithmique linéaire en $O(N)$, où N est le nombre total de voxels. Cependant, elle affiche l'erreur L_0 la plus élevée (37 174), indiquant une précision moindre dans le déroulement de phase.

Minimisation des surfaces d'erreur : Cette méthode améliore la précision en réduisant l'erreur L_0 à 36 943. Néanmoins, cette amélioration s'accompagne d'un temps de calcul considérable de 7 304 secondes (environ 2,03 heures). Cette augmentation du temps de calcul est due à la résolution de problèmes d'optimisation en nombres entiers, dont la complexité est significative.

Surfaces de maillage : Offrant un compromis, cette méthode réduit le temps de calcul à 211 secondes tout en maintenant une erreur L_0 de 37 166, légèrement inférieure à celle de ROMEO. Elle évite la résolution de problèmes d'optimisation complexes en se basant sur des surfaces heuristiques.

Méthodes combinées :

— **Méthode combinée (*ROMEO* + Minimisation) :** Cette approche atteint l'erreur L_0 la plus faible (36 356), améliorant significativement la précision. Cependant, le

temps de calcul reste élevé (7 304 secondes), similaire à celui de la minimisation des surfaces d'erreur seule.

— **Méthode combinée (*ROMEO* + Maillage) :** Elle réduit l'erreur L_0 à 37 158 avec un temps de calcul de 211 secondes, combinant l'efficacité temporelle des surfaces de maillage et les avantages de ROMEO.

Les résultats montrent que les méthodes basées sur la minimisation des surfaces d'erreur offrent une meilleure précision au détriment du temps de calcul. À l'inverse, ROMEO est extrêmement rapide mais moins précis. La méthode combinée (*ROMEO* + Maillage) parvient à réduire l'erreur L_0 par rapport à ROMEO seul, sans augmenter significativement le temps de calcul.

Le choix de la méthode dépend des contraintes spécifiques de l'application. Si le temps de calcul est critique, ROMEO ou la méthode des surfaces de maillage sont préférables. Si la précision prime, les méthodes de minimisation des surfaces d'erreur, malgré leur coût computationnel élevé, sont recommandées. Les méthodes heuristiques utilisant les surfaces de maillage offrent des solutions intermédiaires, améliorant la précision sans nécessairement augmenter proportionnellement le temps de calcul.

5.4.2 Les résultats sur les données empiriques :

Détection des résidus et construction des boucles de singularité : La détection des boucles de singularité, ainsi que les effets de l'amélioration détaillée dans la Section 3.4, sont détaillées dans l'annexe C.

Remplissage des boucles : Il faut aussi noter que malgré le fait que les boucles ouvertes et fermées ont été améliorées, leurs tailles restent quand même assez grandes. En effet, la plus grande boucle ouverte est de taille 3198, et la plus grande boucle fermée est de taille 2501. Le remplissage optimal de ces boucles va donc être très coûteux en terme de temps de calcul. Si on utilise des solution optimale pour toutes les boucles, le temps de calcul sera alors très élevé et difficilement gérable.

Pour surmonter le coût élevé du remplissage des boucles de grande taille, nous avons adopté une stratégie hiérarchisée basée sur le classement des boucles par taille. Cette méthode permet d'adapter les ressources de calcul et le temps de traitement en ajustant les méthodes de remplissage selon la taille des boucles.

Le cadre expérimental reste le même que dans la section précédente. La résolution du problème en nombres entiers devient le facteur limitant pour les boucles de grande taille. Malgré les améliorations apportées et détaillées dans l'Annexe (C), les tailles des boucles ouvertes et

fermées demeurent importantes : la plus grande boucle ouverte compte 3 198 éléments et la plus grande boucle fermée en compte 2 501. Ces dimensions rendent le remplissage optimal de ces boucle particulièrement coûteux en temps de calcul, rendant impraticable une approche uniforme pour toutes les boucles.

En adoptant cette stratégie hiérarchisée, nous pouvons traiter efficacement les boucles volumineuses en adaptant les méthodes de remplissage selon leur taille. Cela réduit significativement le temps de calcul tout en maintenant la qualité du remplissage.

Pour les **boucles ouvertes**, nous avons classé les boucles par ordre décroissant de taille. La boucle numéro 3 100 (sur un total de 15 684 boucles ouvertes) a une taille de 173. Ainsi, les boucles de taille inférieure ou égale à 173 représentent environ les **80% des plus petites boucles** (puisque $3\,100/15\,684 \approx 19{,}8\%$, donc $100\% - 19{,}8\% \approx 80\%$). Pour ces boucles, nous avons appliqué une résolution exacte du problème de minimisation des surfaces d'erreur via le solveur PLNE (GUROBI). Pour les boucles de taille supérieure à 173 (les 20% plus grandes), nous avons utilisé uniquement les arêtes qui traversent les triangles du maillage.

De même, pour les **boucles fermées**, la boucle numéro 4 100 (sur un total de 140 043 boucles fermées) a une taille de 142. Les boucles de taille inférieure ou égale à 142 représentent environ les **97% des plus petites boucles** (puisque $4\,100/140\,043 \approx 2{,}9\%$, donc $100\% - 2{,}9\% \approx 97\%$). Pour ces boucles, nous avons appliqué la résolution exacte. Les boucles de taille supérieure à 142 (les 3% plus grandes) ont été traitées uniquement avec la méthode du maillage, en conservant la surface initiale.

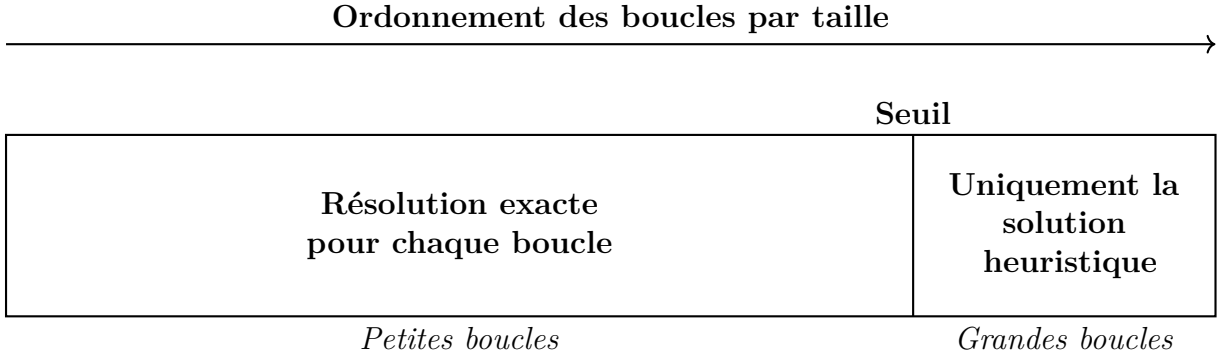


FIGURE 5.10 Schéma illustrant de la stratégie hiérarchisée.

Afin de démontrer l'efficacité de la résolution des problèmes de minimisation des surfaces d'erreur, nous comparons le nombre d'arêtes des boucles obtenu à partir des surfaces de maillage initiales avec celles optimisées par des problèmes de minimisation des surfaces d'erreur. Pour mieux évaluer et visualiser cette réduction dans la taille des boucles, nous avons

calculé le pourcentage de réduction :

$$\text{Pourcentage de réduction} = \frac{\text{Nombre d'arêtes initiales} - \text{Nombre d'arêtes optimisées}}{\text{Nombre d'arêtes initiales}} \times 100$$

Les résultats sont présentés à l'aide d'un diagramme en violon, qui illustre assez clairement la distribution des réductions.

Pour les **boucles ouvertes**, la réduction du nombre d'arêtes est significative, variant entre 30 % et 80 % dans la plupart des cas. En revanche, pour les **boucles fermées**, la réduction est généralement inférieure à 10 %.

Cette différence s'explique par la forme des boucles : les boucles ouvertes sont souvent plus complexes et les arrêtes de leur maillage initial s'éloignent davantage de la solution optimale.Cependant, les boucles fermées sont généralement plus de formes géométriques plus simples, et donc les arêtes de leur maillage initial sont souvent plus proches de la solution optimale. Autrement dit, le **gap d'optimalité** est plus i mportant pour les boucles ouvertes que pour les boucles fermées.

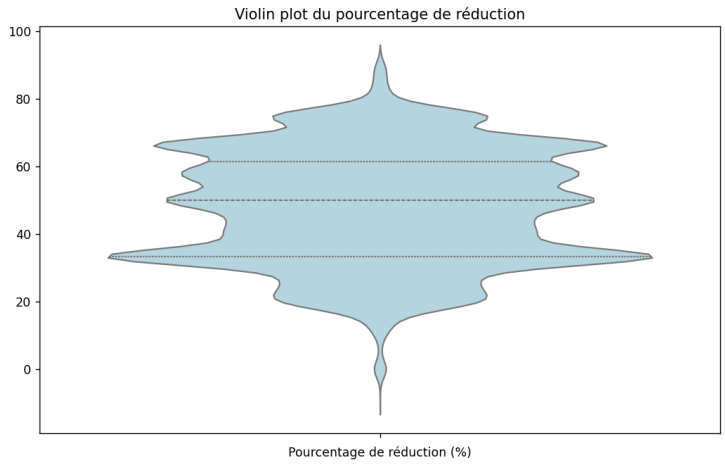


FIGURE 5.11 Distribution du pourcentage de réduction de nombre d'arêtes par la résolution du problème de minimisation des surfaces d'erreur, pour les boucles ouvertes.

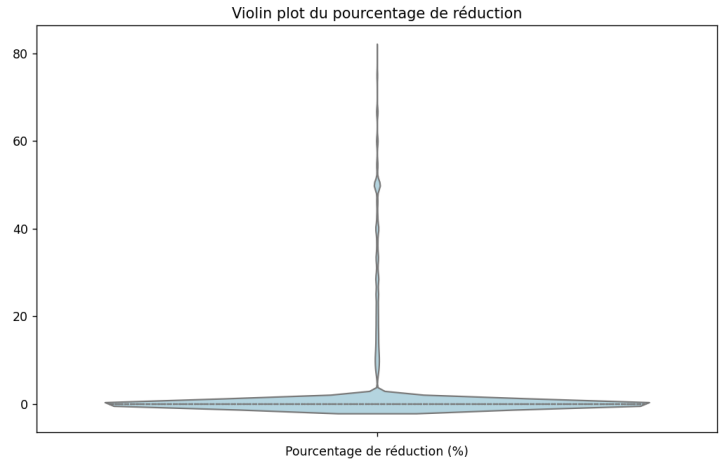


FIGURE 5.12 Distribution du pourcentage de réduction de nombre d'arêtes par la résolution du problème de minimisation des surfaces d'erreur, pour les boucles fermées.

Cadre expérimental : Les expérimentations ont été réalisées sur le même système que celui décrit précédemment, en exploitant pleinement les 64 cœurs CPU disponibles. Le temps de calcul pour chaque étape est résumé dans le tableau 5.4.

TABLEAU 5.4 Temps de calcul détaillés pour les données empiriques.

Étape	**Temps (s)**
Détection des résidus	$10,5$
Construction des boucles	$192,5$
Amélioration et reconstruction des boucles	$235,2$
Remplissage initial des boucles	$2\,800$
Remplissage des boucles (avec PLNE, 97% fermées et 80% ouvertes)	$35\,280 \equiv 9\,8h$
Intégration du chemin	$150,0$
Temps total	$38,794.2$

Comparaison des méthodes et analyse des performances :

Les résultats du tableau 5.5 montrent que la méthode ROMEO est à la fois rapide et performante sur les données empiriques, avec une erreur L_0 relativement faible et un temps de calcul de seulement 150 secondes. Cela s'explique par le fait que *ROMEO* est spécialement conçue pour traiter des données fortement bruitées, comme celles utilisées ici.

En revanche, la méthode de minimisation des surfaces d'erreur présente une erreur L_0 plus élevée et un temps de calcul nettement supérieur, atteignant près de 10,7 heures. Cette performance inférieure est due à la forte quantité de bruit dans les données empiriques,

Tableau 5.5 Comparaison des performances des différentes méthodes sur les données empiriques.

Méthode	**Erreur L_0**	**Temps de calcul (s)**
ROMEO	2 367 204	**150**
Minimisation des surfaces d'erreur	2 553 358	38 664
Surfaces de maillage	2 540 445	3 002
Méthode combinée (*ROMEO* + Minimisation)	**2 354 934**	38 664
Méthode combinée (*ROMEO* + Maillage)	2 503 119	3 002

qui contiennent environ 4 millions de résidus. Le grand nombre de boucles de singularité augmente la complexité computationnelle, ce qui rend cette méthode moins adaptée dans ce contexte.

La méthode combinée (*ROMEO* + Minimisation) offre la meilleure précision avec une erreur L_0 de 2 354 934, inférieure à celle de ROMEO seul. Cependant, le temps de calcul reste élevé. Cette approche tire parti des forces des deux méthodes, en combinant les contraintes géométriques et mathématiques de la minimisation avec les contraintes physiques de ROMEO.

Par rapport aux données synthétiques précédentes, les performances diffèrent en raison de la taille plus importante des données empiriques et du niveau de bruit plus élevé. Les données synthétiques étaient de taille plus réduite et moins bruitées, ce qui rendait la méthode de minimisation des surfaces d'erreur plus efficace sur ces dernières.

Pour la visualisation, les méthodes n'utilisant que des heuristiques (cf. Les Figures 5.15 et 5.17) présentent des discontinuités dans les zones fortement bruitées et à haute densité de résidus (cf. La Figure 5.3 permet de voir les zones fortement bruitées avant le déroulement.). En revanche, les méthodes basées sur la minimisation des surfaces d'erreur (cf. Les Figures 5.14 et 5.16) offrent un rendu visuel aussi satisfaisant que celui de *ROMEO* (cf. La Figure 5.13), avec une erreur plus faible, bien que nécessitant bien plus de temps de calcul. Ainsi, pour les données empiriques, l'approche heuristique par maillage s'avère insuffisante ; il est essentiel de procéder à une minimisation pour obtenir une phase déroulée de qualité, dépourvue de discontinuités significatives.

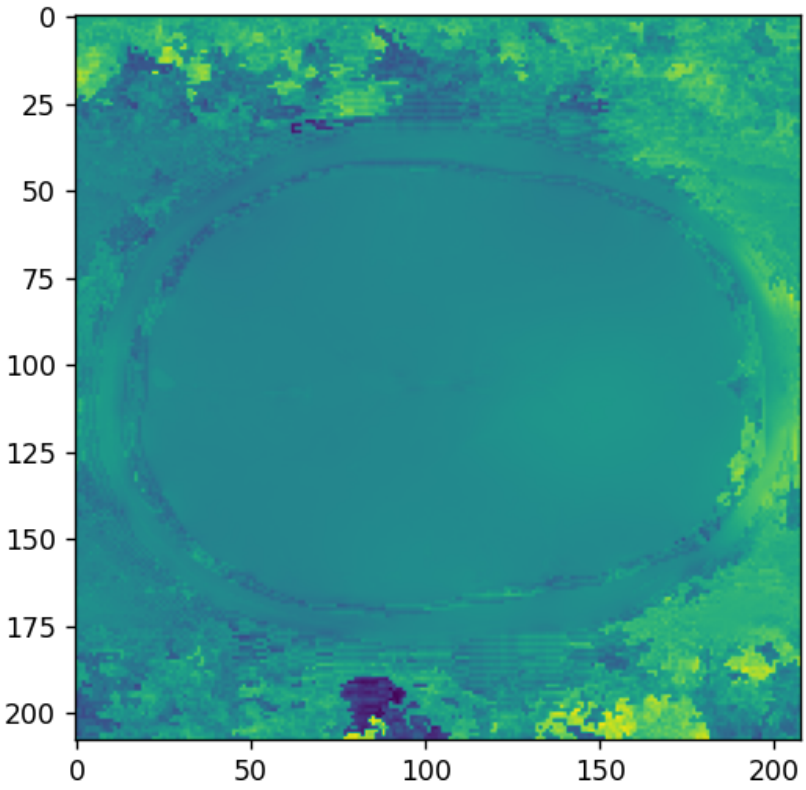


FIGURE 5.13 Visualisation de la phase déroulée par la méthode *ROMEO* sur les données empiriques.

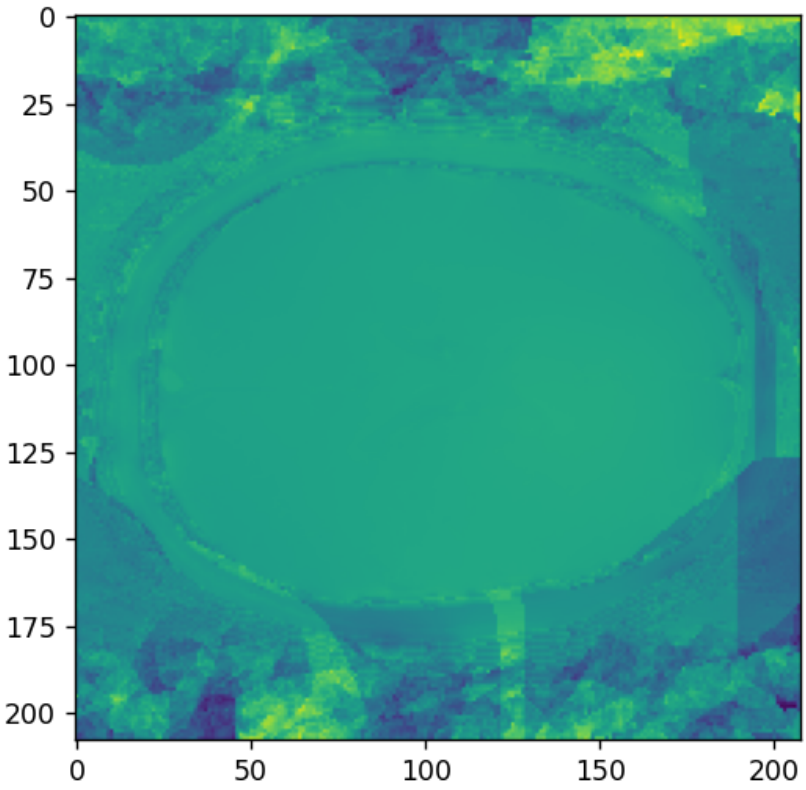


FIGURE 5.14 Visualisation de la phase déroulée par la méthode des minimisation des surfaces d'erreur sur les données empiriques.

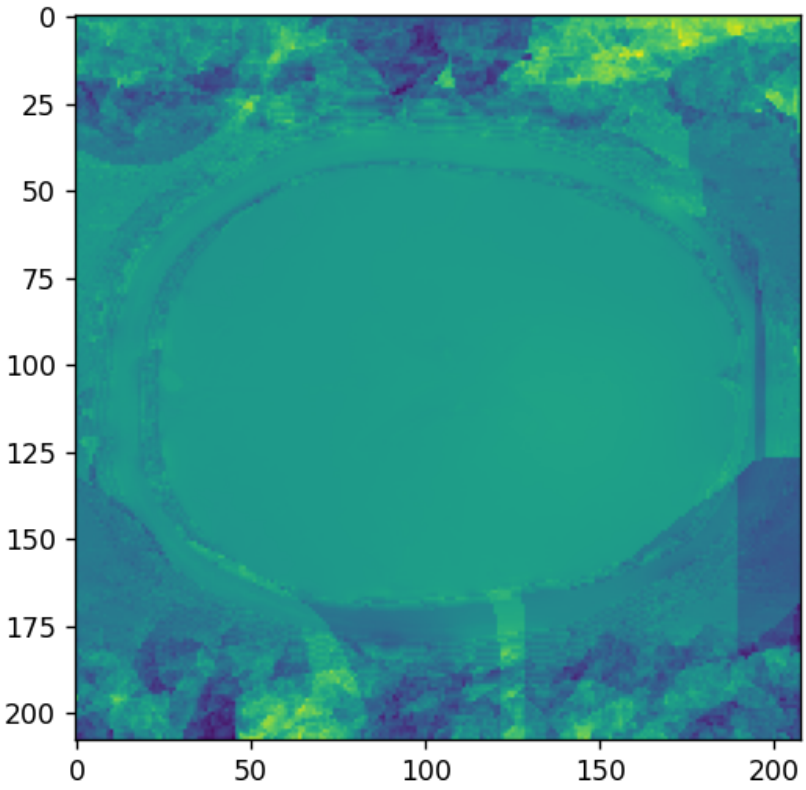


FIGURE 5.15 Visualisation de la phase déroulée par la méthode des surfaces de maillage sur les données empiriques.

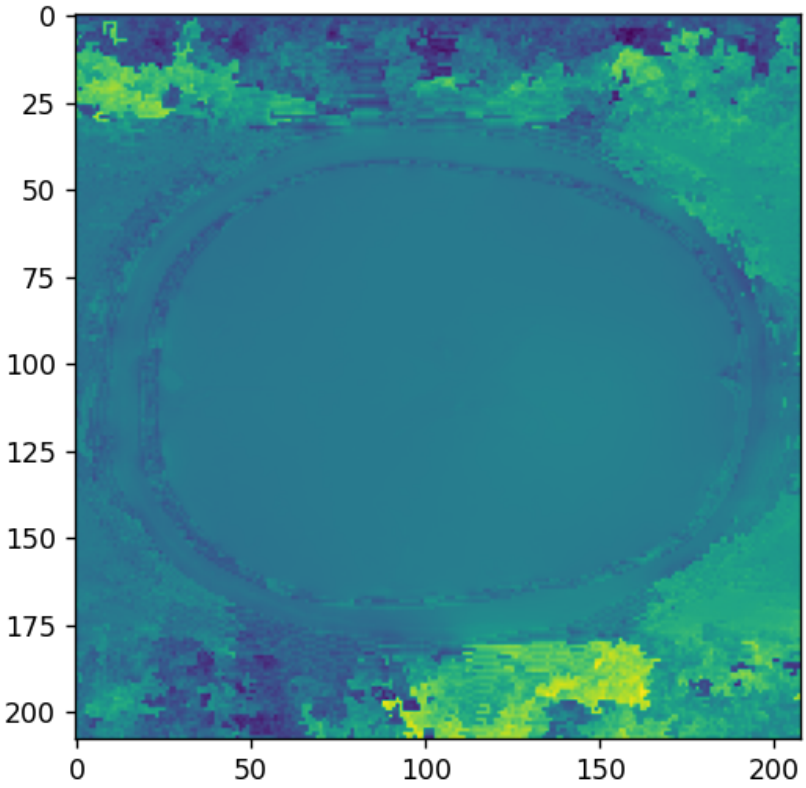


FIGURE 5.16 Visualisation de la phase déroulée par la méthode combinée (*ROMEO* + Minimisation) sur les données empiriques.

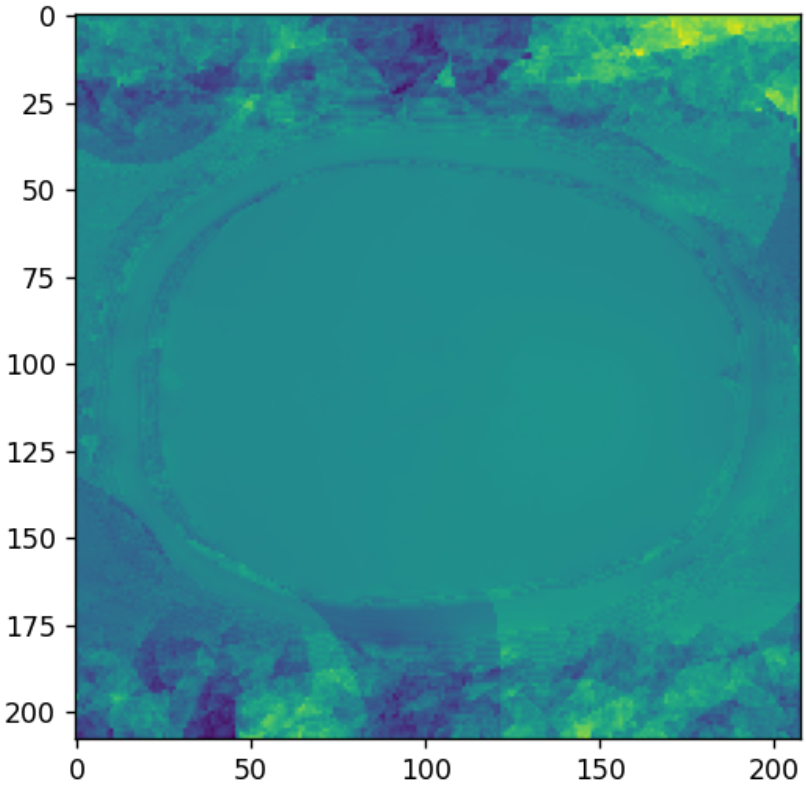


FIGURE 5.17 Visualisation de la phase déroulée par la méthode combinée (*ROMEO* + Maillage) sur les données empiriques.

CHAPITRE 6 CONCLUSION

6.1 Synthèse des travaux

Cette étude a porté sur le déroulement de phase en trois dimensions, en se focalisant sur la gestion des boucles de singularité et la minimisation des surfaces d'erreur dans des environnements bruités. Une méthodologie structurée a été développée, basée sur des contraintes géométriques pour détecter et optimiser les boucles de singularité. Contrairement aux approches existantes, la méthode proposée vise à atteindre une solution optimale pour identifier les surfaces d'erreur à bloquer, afin d'améliorer la précision du déroulement de phase.

L'objectif principal était de fournir une solution mathématiquement rigoureuse au problème du déroulement de phase en 3D. Sur les données synthétiques, la méthode a démontré une grande précision et une robustesse satisfaisante, attestant de son efficacité dans des contextes contrôlés. Cependant, lors des tests sur des données empiriques, souvent très bruitées, certaines limites ont émergé. La forte densité de résidus dans ces données a entraîné un nombre important de boucles de singularité, augmentant considérablement la complexité des calculs et réduisant l'efficacité de la méthode initiale.

Pour surmonter ces limitations, une approche combinée a été développée. Cette méthode intègre les contraintes géométriques de la technique proposée avec les propriétés physiques exploitées par des méthodes existantes, telles que la cohérence spatiale et temporelle. Ce compromis a permis de gérer plus efficacement les données bruitées, tout en maintenant un haut niveau de précision.

6.2 Limitations de la méthode proposée

Malgré les avancées réalisées, la méthode présente certaines limitations, particulièrement lors du traitement de données réelles fortement bruitées. La présence d'un grand nombre de résidus complexifie la construction des boucles de singularité, augmentant significativement la taille et la complexité du graphe associé. Cette surcharge computationnelle rend l'application de la méthode difficile dans des contextes où les données sont volumineuses et bruitées.

De plus, le coût computationnel élevé constitue un grand obstacle. Bien que la méthode vise une optimisation mathématiquement rigoureuse, elle repose sur des solveurs de programmation linéaire en nombres entiers, dont la complexité croît rapidement avec la taille du problème. Lorsque le nombre de résidus et de boucles est important, le temps de calcul de-

vient conséquent, limitant l'applicabilité de la méthode pour des jeux de données de grande taille ou des applications nécessitant des délais rapides.

Enfin, l'absence de prétraitement pour réduire le bruit dans les données constitue une faiblesse majeure. Sans cette étape, les données bruitées introduisent des erreurs supplémentaires dès la phase de construction des boucles, augmentant la complexité des calculs et réduisant l'efficacité globale de la méthode. Ce manque de filtration initiale peut également affecter la qualité des résultats finaux, en particulier lorsque la densité des résidus est élevée.

6.3 Améliorations futures

Pour remédier à ces limitations, plusieurs axes d'amélioration peuvent être proposés afin d'accroître l'efficacité et la robustesse de la méthode proposée. Une première piste essentielle concerne la réduction du bruit dans les données. Un prétraitement des données, utilisant des techniques de filtrage adaptées, permettrait de diminuer significativement la densité des résidus inutiles avant la construction des boucles de singularité. Cette étape simplifierait les structures des graphes utilisés et limiterait l'augmentation de la complexité computationnelle.

Une autre voie d'amélioration réside dans l'intégration des propriétés physiques des données dans les étapes d'optimisation. Bien que la méthode privilégie une approche purement mathématique, exploiter les propriétés intrinsèques des données, telles que la cohérence spatiale et temporelle, pourrait offrir des avantages significatifs. Cette intégration permettrait de réduire le temps de calcul tout en conservant une précision acceptable, notamment pour des jeux de données fortement bruités ou de grande taille.

Enfin, l'introduction de techniques de parallélisation plus avancées pourrait considérablement réduire le temps de calcul des étapes les plus coûteuses. La parallélisation est particulièrement utile pour résoudre plusieurs problèmes de programmation linéaire en nombres entiers simultanément, accélérant ainsi le processus d'optimisation tout en maintenant des résultats optimaux.

Ces perspectives visent à rendre la méthode proposée plus robuste, plus rapide et mieux adaptée aux données réelles, en particulier dans des contextes où le bruit et la complexité des données posent des défis majeurs. Elles permettent également de mieux équilibrer l'objectif d'optimalité mathématique avec des contraintes pratiques, notamment en termes de ressources et de temps de calcul, ouvrant ainsi la voie à des applications plus larges et plus efficaces du déroulement de phase en trois dimensions.

RÉFÉRENCES

[1] K. Itoh, "Analysis of the phase unwrapping algorithm," *Appl. Opt.*, vol. 21, n°. 14, p. 2470–2470, Jul 1982.

[2] D. C. Ghiglia et L. A. Romero, "Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods," *J. Opt. Soc. Am. A*, vol. 11, n°. 1, p. 107–117, Jan 1994.

[3] R. M. Goldstein, H. A. Zebker et C. L. Werner, "Satellite radar interferometry : Two-dimensional phase unwrapping," *Radio Science*, vol. 23, n°. 4, p. 713–720, 1988.

[4] Z. Deprem et E. Onat, "Phase unwrapping via hierarchical and balanced residue partitioning," *Signal, Image and Video Processing*, vol. 18, n°. 3, p. 2895–2902, 2024.

[5] R. Marhamati et M. A. Masnadi-Shirazi, "The principles of proper placement of branch cut in phase unwrapping using combined and extended methods based on residue searching," *Signal, Image and Video Processing*, vol. 14, p. 593–600, 2020.

[6] D. Ghiglia et M. Pritt, *Two-Dimensional Phase Unwrapping : Theory, Algorithms, and Software*. Wiley, 1998.

[7] I. Herszterg, M. Poggi et T. Vidal, "Two-dimensional phase unwrapping via balanced spanning forests," *INFORMS Journal on Computing*, vol. 31, n°. 3, p. 527–543, juill. 2019.

[8] J. Huntley, "Three-dimensional noise-immune phase unwrapping algorithm," *Applied optics*, vol. 40, p. 3901–8, 09 2001.

[9] O. Marklund, J. M. Huntley et R. Cusack, "Robust unwrapping algorithm for three-dimensional phase volumes of arbitrary shape containing knotted phase singularity loops," *Optical Engineering*, vol. 46, n°. 8, p. 085 601–085 601, 2007.

[10] H. Abdul-Rahman *et al.*, "Robust three-dimensional best path avoiding singularity loops phase unwrapping algorithm," *Applied optics*, vol. 48, p. 4582–96, 09 2009.

[11] M. Arevalillo-Herráez, M. A. Gdeisat et D. R. Burton, "Hybrid robust and fast algorithm for three-dimensional phase unwrapping," *Appl. Opt.*, vol. 48, n°. 32, p. 6313–6323, Nov 2009.

[12] H. Yu *et al.*, "Phase unwrapping in insar : A review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 7, n°. 1, p. 40–58, 2019.

[13] L. Pu *et al.*, "A robust insar phase unwrapping method via phase gradient estimation network," *Remote Sensing*, vol. 13, n°. 22, 2021.

[14] J. J. He *et al.*, “Deep spatiotemporal phase unwrapping of phase-contrast mri data,” dans *Proceedings of the 27th ISMRM Annual Meeting & Exhibition*, 2019.

[15] M. Löcke *et al.*, “A comparison of phase unwrapping methods in velocity-encoded mri for aortic flows,” *Magnetic resonance in medicine*, vol. 90, n°. 5, p. 2102–2115, 2023.

[16] S. Hubmer *et al.*, “On phase unwrapping via digital wavefront sensors,” *arXiv preprint arXiv :2405.15419*, 2024.

[17] V. Akondi *et al.*, “Phase unwrapping with a virtual hartmann-shack wavefront sensor,” *Optics Express*, vol. 23, p. 25 425–25 439, 09 2015.

[18] C. Wu *et al.*, “Phase unwrapping based on a residual en-decoder network for phase images in fourier domain doppler optical coherence tomography,” *Biomed. Opt. Express*, vol. 11, n°. 4, p. 1760–1771, Apr 2020.

[19] K. Wang *et al.*, “One-step robust deep learning phase unwrapping,” *Opt. Express*, vol. 27, n°. 10, p. 15 100–15 115, May 2019.

[20] G. Sun *et al.*, “Phase unwrapping based on channel transformer u-net for single-shot fringe projection profilometry,” *Journal of Optics*, p. 1–11, 2023.

[21] G. Valadão et J. M. Bioucas-Dias, “Edge preserving phase unwrapping using graph cuts,” dans *Proceedings of the EARSeL Symposium*, 2005.

[22] M. Jenkinson, “Fast, automated, n-dimensional phase-unwrapping algorithm,” *Magnetic Resonance in Medicine : An Official Journal of the International Society for Magnetic Resonance in Medicine*, vol. 49, n°. 1, p. 193–197, 2003.

[23] J. Cheng *et al.*, “A new 3d phase unwrapping method by region partitioning and local polynomial modeling in abdominal quantitative susceptibility mapping,” *Frontiers in Neuroscience*, vol. 17, 2023.

[24] L. L. García, A. G. Arellano et W. Cruz-Santos, “A parallel path-following phase unwrapping algorithm based on a top-down breadth-first search approach,” *Optics and Lasers in Engineering*, vol. 124, p. 105827, 2020.

[25] W. Xu et I. Cumming, “A region-growing algorithm for insar phase unwrapping,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, n°. 1, p. 124–134, Jan 1999.

[26] G. E. Spoorthi, S. Gorthi et R. K. S. S. Gorthi, “Phasenet : A deep convolutional neural network for two-dimensional phase unwrapping,” *IEEE Signal Processing Letters*, vol. 26, n°. 1, p. 54–58, 2019.

[27] F. Yang *et al.*, “Robust phase unwrapping via deep image prior for quantitative phase imaging,” *IEEE Transactions on Image Processing*, vol. 30, p. 7025–7037, 2021.

[28] C. Chen et H. Zebker, "Phase unwrapping for large sar interferograms : statistical segmentation and generalized network models," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, n°. 8, p. 1709–1719, 2002.

[29] H. A. Zebker et Y. Lu, "Phase unwrapping algorithms for radar interferometry : residue-cut, least-squares, and synthesis algorithms," *J. Opt. Soc. Am. A*, vol. 15, n°. 3, p. 586–598, Mar 1998.

[30] H. Yu *et al.*, "Large-scale L^0 -norm and L^1 -norm 2-d phase unwrapping," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, n°. 8, p. 4712–4728, 2017.

[31] R. Marhamati et M. Masnadi-Shirazi, "The principles of proper placement of branch cut in phase unwrapping using combined and extended methods based on residue searching," *Signal, Image and Video Processing*, vol. 14, p. 1–8, 04 2020.

[32] J. C. de Souza, M. E. Oliveira et P. A. M. dos Santos, "Branch-cut algorithm for optical phase unwrapping," *Opt. Lett.*, vol. 40, n°. 15, p. 3456–3459, Aug 2015.

[33] F. Sawaf et R. Tatam, "Finding minimum spanning trees more efficiently for tile-based phase unwrapping," *Measurement Science and Technology*, vol. 17, p. 1428, 05 2006.

[34] I. Ljubić, "Solving steiner trees : Recent advances, challenges, and perspectives," *Networks*, vol. 77, n°. 2, p. 177–204, 2021.

[35] D. C. Ghiglia et L. A. Romero, "Minimum lp-norm two-dimensional phase unwrapping," *J. Opt. Soc. Am. A*, vol. 13, n°. 10, p. 1999–2013, Oct 1996.

[36] C. W. Chen et H. A. Zebker, "Network approaches to two-dimensional phase unwrapping : intractability and two new algorithms," *J. Opt. Soc. Am. A*, vol. 17, n°. 3, p. 401–414, Mar 2000.

[37] Y. Gao *et al.*, "A phase slicing 2-d phase unwrapping method using the l_1-norm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, n°. 6, p. 4321–4332, 2020.

[38] A. Krizhevsky, I. Sutskever et G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[39] H. Abdul-Rahman, "Three-dimensional fourier fringe analysis and phase unwrapping," Thèse de doctorat, Liverpool John Moores University, January 2007.

[40] X. Su *et al.*, "Dynamic 3-d shape measurement method based on ftp," *Optics and Lasers in Engineering*, vol. 36, n°. 1, p. 49–64, 2001.

[41] C. Zuo *et al.*, "High-speed three-dimensional shape measurement for dynamic scenes using bi-frequency tripolar pulse-width-modulation fringe projection," *Optics and Lasers in Engineering*, vol. 51, n°. 8, p. 953–960, 2013.

[42] R. Cusack et N. Papadakis, “New robust 3-d phase unwrapping algorithms : Application to magnetic field mapping and undistorting echoplanar images,” *NeuroImage*, vol. 16, n°. 3, Part A, p. 754–764, 2002.

[43] A. Karsa et K. Shmueli, “Segue : A speedy region-growing algorithm for unwrapping estimated phase,” *IEEE transactions on medical imaging*, vol. 38, n°. 6, p. 1347–1357, 2018.

[44] M. Arevalillo-Herráez, D. R. Burton et M. J. Lalor, “Clustering-based robust three-dimensional phase unwrapping algorithm,” *Applied optics*, vol. 49, n°. 10, p. 1780–1788, 2010.

[45] B. Dymerska *et al.*, “Phase unwrapping with a rapid opensource minimum spanning tree algorithm (romeo),” *bioRxiv*, 2020.

[46] R. C. Prim, “Shortest connection networks and some generalizations,” *The Bell System Technical Journal*, vol. 36, n°. 6, p. 1389–1401, 1957.

[47] M. Katzman, “Bipartite graphs whose edge algebras are complete intersections,” *Journal of Algebra*, vol. 220, n°. 2, p. 519–530, 1999.

[48] W. Chen, Y. Cai et J. Zheng, “Constructing triangular meshes of minimal area,” *Computer-Aided Design and Applications*, vol. 5, p. 508–518, 06 2008.

[49] A. Hatcher, *Algebraic Topology*, ser. Algebraic Topology. Cambridge University Press, 2002.

[50] T. Kavitha *et al.*, “Cycle bases in graphs characterization, algorithms, complexity, and applications,” *Computer Science Review*, vol. 3, n°. 4, p. 199–243, 2009.

[51] J. Hooker, “Logic-based methods for optimization,” 01 1994.

[52] T. Ralphs *et al.*, “On the capacitated vehicle routing problem,” *Mathematical Programming*, vol. 94, p. 343–359, 01 2003.

[53] L. Marchenko et V. Podgornaya, “Some ideas about connected graphs isomorphism,” *Advances in Computer Science Research*, n°. 14, p. 105–123, 2018.

[54] I. P. Gent *et al.*, “Learning when to use lazy learning in constraint solving,” dans *ECAI 2010*. IOS Press, 2010, p. 873–878.

[55] A. K. Datta *et al.*, “Analysis of a memory-efficient self-stabilizing BFS spanning tree,” *CoRR*, vol. abs/1907.07944, 2019.

[56] S. Beamer, K. Asanovic et D. Patterson, “Direction-optimizing breadth-first search,” dans *SC ’12 : Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012, p. 1–10.

[57] D. Harel, “A linear time algorithm for the lowest common ancestors problem,” dans *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, 1980, p. 308–319.

[58] M. Kowaluk et A. Lingas, "Lca queries in directed acyclic graphs," dans *Automata, Languages and Programming*, L. Caires *et al.*, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, p. 241–248.

[59] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, n°. 2, p. 146–160, 1972.

[60] J. Fischer, "Optimal succinctness for range minimum queries," *CoRR*, vol. abs/0812.2775, 2008.

ANNEXE A FERMETURE DES BOUCLES OUVERTES

Une fois les boucles ouvertes créées, il faut les fermer afin de confiner les erreurs de phase autour de ces singularités [8,9,11]. La fermeture de ces boucles doit garantir que les transitions entre les résidus qui composent la boucle sont complètement bloquées, empêchant toute propagation d'erreur hors de la singularité [8,10] Pour cela, on crée un chemin supplémentaire entre les faces sans prédécesseur $N_{noparent}$ et les faces sans successeur $N_{nochild}$. Ce chemin est construit de manière à ne pas permettre l'accès à l'intérieur de la boucle de singularité ouverte, créant ainsi une barrière infranchissable.

1. **Association de chaque face à un nœud du graphe $G_{grid}(\Phi)$:** Nous commençons par lier la représentation géométrique de chaque face extérieure (c'est-à-dire les faces dans l'ensemble $N_{noparent}$ ou l'ensemble $N_{nochild}$) à un nœud spécifique du graphe de grille $G_{grid}(\Phi)$. Pour ce faire, nous utilisons le barycentre de la face, qui est calculé comme expliqué précédemment, et nous le projetons sur l'un des nœuds qui forment la face. Pour uniformiser la projection, nous allons choisir le nœud le plus proche de l'origine O avec les coordonnées (0, 0, 0) du graphe $G_{grid}(\Phi)$.
2. **Recherche du chemin de fermeture le plus court sur le contour du graphe :** Une fois que les faces terminales sont associées à des nœuds du graphe $G_{grid}(\Phi)$, nous cherchons à connecter ces nœuds en trouvant le chemin le plus court sur le contour extérieur du graphe. Afin de s'assurer qu'aucun passage entre les résidus du graphe n'est possible, nous limitons notre recherche aux arêtes du contour, c'est-à-dire aux arêtes situées sur les bords extérieurs du graphe.

Pour cela, nous construisons un sous-graphe, noté G_{contour}, qui représente le contour du graphe $G_{grid}(\Phi)$. Ce sous-graphe est composé des nœuds situés sur les faces extérieures du graphe, en particulier celles aux limites des axes x, y et z.

Definition A.1 (Graphe de Contour G_{contour})**.** *Le graphe de contour $G_{contour} = (V_{contour}, E_{contour})$ est un sous-graphe de $G_{grid}(\Phi)$ qui regroupe les sommets situés sur la paroi extérieure du domaine tridimensionnel. Il est défini comme suit :*

- $V_{contour} = \{(x, y, z) \in V_{grid}(\Phi) \mid x = 0$ *ou* $x = X_{\max}$ *ou* $y = 0$ *ou* $y = Y_{\max}$ *ou* $z = 0$ *ou* $z = Z_{\max}\}$*, où* $X_{\max}, Y_{\max}, Z_{\max}$ *représentent les dimensions maximales du domaine en* x, y, z*.*
- $E_{contour} \subseteq E_{grid}(\Phi)$*, avec chaque arête* $(u, v) \in E_{contour}$ *connectant des sommets adjacents de* $V_{contour}$ *sur la paroi extérieure.*

Le graphe $G_{contour}$ sert de cadre limitant le domaine, regroupant les sommets et les arêtes qui se situent exclusivement sur les frontières de la grille, c'est-à-dire les surfaces délimitées par $x = 0$, $x = X_{\max}$, $y = 0$, $y = Y_{\max}$, $z = 0$, et $z = Z_{\max}$.

Ensuite, pour chaque paire de nœuds associés aux extrémités d'une boucle ouverte, nous utilisons un algorithme de plus court chemin (tel que l'algorithme de Dijkstra ou un parcours en largeur) sur le graphe G_{contour} pour trouver le chemin optimal les reliant. Ce chemin de fermeture est alors ajouté à la boucle ouverte pour former une boucle fermée complète.

La démarche détaillée est la suivante :

— Soit $N_{\text{no parent}}$ le nœud d'entrée (sans parent) et $N_{\text{no child}}$ le nœud de sortie (sans successeur) d'une boucle ouverte dans le graphe de faces dénoué $G_{grid}(\Phi)^{\text{faces}}$.
— Nous calculons les barycentres de leurs faces correspondantes et les projetons sur les nœuds les plus proches du graphe $G_{grid}(\Phi)$. Ces nœuds projetés sont notés P_{start} et P_{end}.
— Nous construisons le sous-graphe G_{contour} en incluant les nœuds du contour de $G_{grid}(\Phi)$.
— Nous utilisons un algorithme de plus court chemin sur G_{contour} pour trouver le chemin C_{closing} reliant P_{start} et P_{end}.
— Le chemin de fermeture C_{closing} est ensuite concaténé à la boucle ouverte initiale pour former une boucle fermée :

Complexité de l'algorithme : La complexité de la fermeture des boucles

Cependant, le nombre de nœuds d'entrée $|N_{\text{no parent}}|$ est généralement très faible comparé au nombre total de nœuds du graphe $G_{grid}(\Phi)$. De plus, le graphe de contour G_{contour} est de taille réduite par rapport au graphe complet $G_{grid}(\Phi)$. Par conséquent, la complexité reste acceptable pour une application pratique.

Illustration du processus :

Considérons une boucle ouverte dont les extrémités sont associées aux faces F_{start} et F_{end}. Après avoir calculé les barycentres de ces faces et les avoir projetés sur les nœuds P_{start} et P_{end} du graphe $G_{grid}(\Phi)$, nous cherchons le chemin le plus court sur le contour G_{contour} reliant ces deux nœuds.

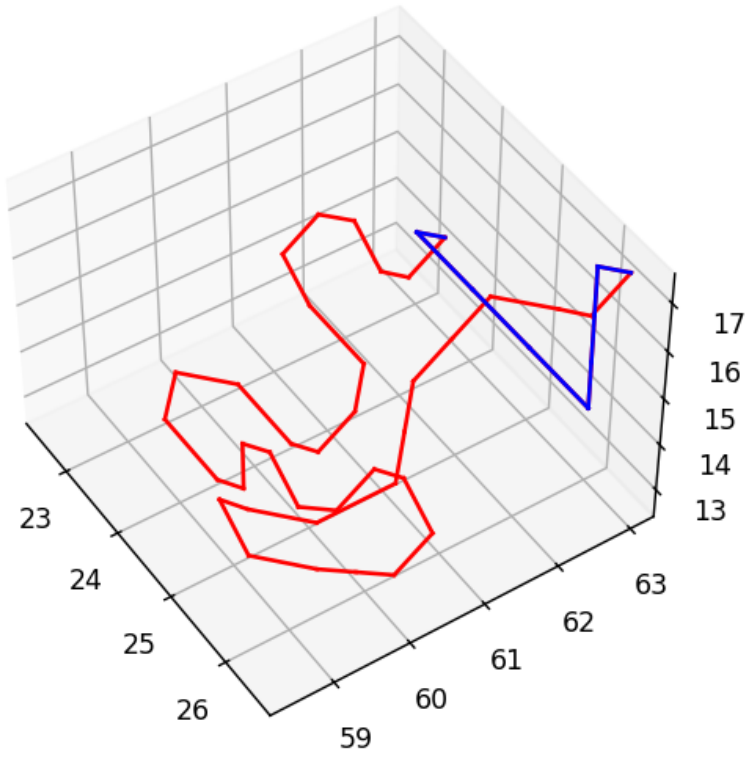


FIGURE A.1 Illustration de la fermeture d'une boucle ouverte en utilisant le contour du graphe $G_{grid}(\Phi)$. Les nœuds P_{start} et P_{end} sont reliés par le chemin de fermeture C_{closing} (en bleu) sur le contour.

En utilisant le chemin de fermeture trouvé, nous obtenons une boucle fermée complète.

ANNEXE B ALGORITHME DE DÉTECTION DE BASES DE CYCLES :

La détection de l'ensemble des bases de cycles B se fait en passant par un arbre couvrant de G_i, que l'on appellera T_i [50].

Definition B.1. *Un **arbre couvrant** [55] d'un graphe G est un sous-graphe de G qui est un arbre et qui contient tous les sommets de G.*

Détection de l'arbre couvrant :

Cet arbre existe toujours pour un graphe non orienté et connexe, on admet que le graphe G_i est toujours connexe.

On commence par construire l'arbre T_i en utilisant un parcours en largeur *(Breadth-First Search)* [56] à partir d'un sommet quelconque de G_i, noté u. Lors de ce parcours, on note le nombre d'arêtes de S_1 traversées lors de la construction de cet arbre, et ceci, pour chaque sommet depuis u.

Lors du parcours en largeur du graphe G_i, on garde de côté le nombre d'arêtes de S_1 traversées par l'arbre T_i pour chaque sommet, et ce, à partir de la racine u. Pour un sommet v de V_i, $dict_i[v]$ contient le nombre d'arêtes de S_1 qui sont sur le chemin $ch(u, v)$.

La **complexité** de cet algorithme est en $O(|V_i| + |E_i|)$ vu qu'on visite chaque sommet et chaque arête une seule fois lors du parcours en largeurs (BFS) [56].

Algorithm 5 Construction de l'arbre T_i avec un parcours en largeur depuis un sommet u

Require: Graphe $G_i = (V, E)$, ensemble d'arêtes $S_1 \subseteq E$, sommet de départ u
Ensure: Arbre T_i et le nombre d'arêtes de S_1 traversées pour chaque sommet

Marquer tous les sommets de G_i comme non visités
$T_i = \emptyset$
$queue = \{u\}$
$dict_i[u] = \{\}$
for chaque élément u dans $V_i = V(G_i)$ **do**
    $dict_i[u] = 0$
end for
while *queue* n'est pas vide **do**
    Retirer un sommet v de *queue*
    Marquer v comme visité
    for chaque voisin w de v **do**
        if w n'est pas visité **then**
            Ajouter l'arête (v, w) à l'arbre T_i
            Ajouter w à *queue*
            if $(v, w) \in S_1$ **then**
                $dict_i[w] = dict_i[v] + 1$
            else
                $dict_i[w] = dict_i[v]$
            end if
        end if
    end for
end while
return T_i, $dict_i$

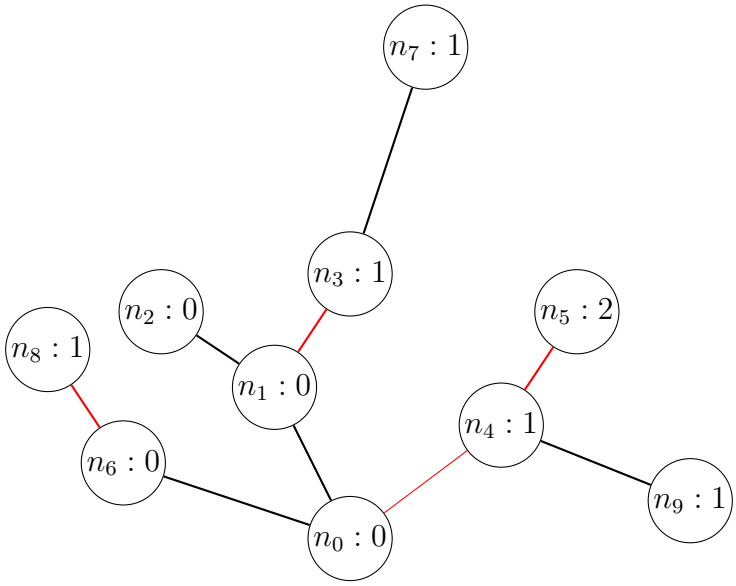


FIGURE B.1 L'arbre couvrant T_i de G_i, les arêtes de S_1 sont marquées en rouge. À côté de chaque nœud, on a le nombre d'arêtes de S_1 traversées pour atteindre ce nœud depuis la racine n_0 (c'est-à-dire $dict_i$).

Construction de la base de cycles

Une fois l'arbre T_i construit, nous procédons à la construction de la base de cycles B du graphe G_i. Le dictionnaire dict_i est utilisé pour ne conserver que les éléments de B qui sont des cycles traversant un nombre impair d'arêtes de S_1, constituant ainsi la base B_1. L'algorithme consiste à ajouter chaque arête de G_i qui n'apparaît pas dans l'arbre couvrant T_i et à identifier le cycle résultant dans T_i (cf. Algorithme 6). Pour mettre en œuvre cet algorithme de manière efficace, il est crucial de déterminer rapidement le chemin entre deux sommets de T_i.

Pour trouver le plus bas ancêtre commun de deux sommets u et v dans T_i, une approche simple serait d'effectuer une recherche en profondeur dans l'arbre. Cependant, cette méthode a une complexité linéaire en la taille de T_i, soit $O(|V_i|)$. Combinée à la recherche de tous les cycles, cela rend l'algorithme de construction de la base de cycles inefficace, avec une complexité totale de $O(|V_i| \times |E_i|)$. Pour améliorer l'efficacité, nous procédons donc à un prétraitement de l'arbre T_i. Cette étape permet de trouver le plus bas ancêtre commun de manière plus rapide, optimisant ainsi l'algorithme de construction de la base de cycles.

Utilisation de l'*Euler Tour* Dans la recherche du plus bas commun ancêtre *LCA* :

L'**Euler Tour** [57, 58] d'un arbre est un parcours en profondeur de l'arbre qui visite chaque arête deux fois, une fois en descendant dans l'arbre et une fois en remontant.

Algorithm 6 Génération des cycles B_1

```
Initialiser une liste vide B_1
Calculer la profondeur depth[x] et le parent parent[x] pour chaque nœud x dans T_i
for chaque arête (u, v) qui n'est pas dans T_i do
    Calculer N_uv = dict_i[u] + dict_i[v]
    if (u, v) ∈ S_1 then
        N_uv = N_uv + 1
    end if
    if N_uv est impair then
        LCA = LCA(u, v)
        Construire le chemin de u à v en passant par le LCA
        Ajouter l'arête (u, v) pour former un cycle
        Ajouter le cycle à B_1
    end if
end for
```

En construisant ce tour, on garde en mémoire **la profondeur** de chaque nœud visité, ainsi que **l'ordre** de visite de chaque nœud durant le parcours en profondeur.

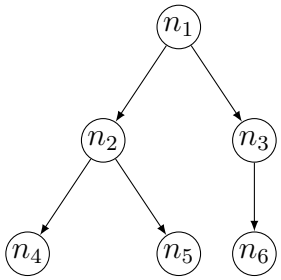


FIGURE B.2 Exemple de parcours d'Euler dans un arbre

Après le parcours en profondeur du graphe, on obtient deux listes :

— Une liste de profondeur qui contient la profondeur de chaque nœud visité.

— Une liste d'ordre qui contient l'ordre de visite de chaque nœud.

Pour l'exemple de la Figure B.2, le parcours d'Euler dans l'arbre donne les listes de profondeur et d'ordre suivantes : $[0, 1, 2, 1, 2, 1, 0, 1, 2, 1, 0]$ et $[n_1, n_2, n_4, n_2, n_5, n_2, n_1, n_3, n_6, n_3, n_1]$, et si on cherche le plus bas ancêtre commun de n_4 et n_5, il suffit de :

1. Trouver les indices de n_4 et n_5 dans la liste d'ordre, qui dans ce cas sont respectivement 2 et 4.
2. Trouver l'indice du minimum entre ces deux indices dans la liste de profondeur, qui est dans ce cas l'indice 3 avec une profondeur de 1.
3. Trouver le nœud correspondant à cet indice dans la liste d'ordre, qui est n_2.

On peut remarquer que les nœuds qui ne sont pas feuilles de l'arbre peuvent posséder plusieurs indices dans la liste d'ordre, ce qui est dû au fait que ces nœuds sont visités plusieurs fois lors du parcours en profondeur. Cependant, le choix de l'indice ne change en aucun cas le résultat final du calcul du plus bas ancêtre commun (LCA) [57].

Ceci est dû au fait que le choix des indices des nœuds ne change pas la valeur obtenue dans la liste des profondeurs. En effet, on choisit l'indice avec la plus petite profondeur, et qui est dans le passage du premier nœud au deuxième. Cette valeur reste la même vu que le parcours profondeur privilégie les nœuds les plus profonds. Ainsi, peu importe le passage entre les deux nœuds, le plus bas ancêtre commun aura toujours la plus petit profondeur.

En effet, pour deux nœuds u et v dans l'arbre T_i, (leurs indices respectifs étant a et b), ce qui nous intéresse ici, c'est de trouver **l'indice** k dans la liste de profondeur tel que : $k = \text{argmin}_{a \leq i \leq b}\text{profondeur}[i]$. Le *Range minimum query* est une structure de données qui permet de répondre à ce type de requête en temps constant. Cette opération permet de trouver en temps constant le minimum d'un intervalle donné d'un tableau.

Dernier point à noter est que vu l'invariance du choix de l'indice dans la représentation inverse des nœuds, on peut simplement choisir de garder le premier indice de chaque nœud.

Dans l'algorithme (cf. Algorithme 7) de recherche du plus bas ancêtre commun consiste à parcourir le graphe en profondeur, et à garder en mémoire :

1. L'ordre de visite de chaque nœud, ainsi que le premier indice de chaque nœud.
2. La profondeur associée à chaque nœud dans une structure de données *Range Minimum Query*, qui permet de trouver en temps constant le plus petit élément d'un intervalle donné.

La complexité du parcours en profondeur [59] dans l'arbre T_i est en $O(|V_i|)$, et la construction de la structure de données *Range Minimum Query* [60] via **table sparse** est en $O(|V_i|log(|V_i|))$.

Ainsi, le prétraitement pour la recherche du plus bas ancêtre commun est en $O(|V_i|log(|V_i|))$.

Une fois le prétraitement effectué, on peut trouver le plus bas ancêtre commun de deux sommets u et v avec une **complexité constante** grâce à la structure de données *Range Minimum Query*.

Algorithm 7 Prétraitement pour le LCA efficace avec Euler Tour

Require: Un arbre enraciné T avec racine r
Ensure: Structures pour calculer le LCA en temps constant

```
function PRÉTRAITEMENT-LCA
    Initialiser des listes vides Euler, profondeur
    Initialiser un dictionnaire PremierIndice
    DFS(r, 0)
    Construire la structure RMQ sur profondeur
end function

procedure DFS(u, d)
    Ajouter u à Euler
    Ajouter d à profondeur
    if u n'est pas dans PremierIndice then
        PremierIndice[u] = len(Euler) −1
    end if
    for chaque enfant v de u do
        DFS(v, d + 1)
        Ajouter u à Euler
        Ajouter d à profondeur
    end for
end procedure
```

Algorithm 8 Calcul du LCA avec Euler Tour

Require: Nœuds u et v, PremierIndice, profondeur, RMQ sur profondeur
Ensure: LCA de u et v

```
function LCA(u, v)
    i = PremierIndice[u]
    j = PremierIndice[v]
    if i > j then
        Échanger i et j
    end if
    k = RMQ(i, j)
    return Euler[k]
end function
```

ANNEXE C EFFETS DE L'AMÉLIORATION SUR LA TAILLE DES BOUCLES DE SINGULARITÉ

Application de l'amélioration sur les données synthétiques :

Dans cette partie, nous mettons en évidence l'effet de l'amélioration du démêlage du graphe de faces $G_{faces}(\Phi)$ sur le nombre de boucles de singularité détectées. La Figure 3.10 dans la Section 3.4 illustre le processus de création des boucles de singularité. Ce processus commence par un premier couplage pour créer un premier ensemble de boucles de singularité. À partir de ce premier lot, nous appliquons un couplage amélioré qui génère un second lot de boucles de singularité optimisées. Les Figures C.1 et C.5 illustrent cette transition sur des données synthétiques, où nous effectuons des statistiques sur les boucles du premier lot avant l'amélioration (cf. la Figure C.1), puis sur celles du second lot après l'amélioration (cf. la Figure C.2).

Statistiques	**Valeurs**
Nombre de boucles ouvertes	298
Taille moyenne	20.463
Taille médiane	10.0
Taille quantile 0.90%	38.0
Taille quantile 0.10%	6.0
Taille maximale	530
Taille minimale	5

(a) Statistiques des boucles ouvertes avant l'amélioration.

Statistiques	**Valeurs**
Nombre de boucles fermées	3075
Taille moyenne	8.15
Taille médiane	6.0
Taille quantile 0.90%	14.0
Taille quantile 0.10%	4.0
Taille maximale	114
Taille minimale	4

(b) Statistiques des boucles fermées avant l'amélioration.

FIGURE C.1 Statistiques des boucles ouvertes et fermés avant l'amélioration.

Statistiques	Valeurs
Nombre de boucles ouvertes	298
Taille moyenne	11.46
Taille médiane	8.0
Taille quantile 0.90%	22.60
Taille quantile 0.10%	6.0
Taille maximale	50
Taille minimale	5

(a) Statistiques des boucles ouvertes après l'amélioration.

Statistiques	Valeurs
Nombre de boucles fermées	3370
Taille moyenne	8.32
Taille médiane	6.0
Taille quantile 0.90%	14.0
Taille quantile 0.10%	4.0
Taille maximale	114
Taille minimale	4

(b) Statistiques des boucles fermées après l'amélioration.

FIGURE C.2 Statistiques et distributions des boucles ouvertes et fermés après l'amélioration.

Les figures C.1 montrent les statistiques des boucles de singularité détectées dans les données synthétiques avant que ces dernières ne soient améliorées par la méthode de détection et de traitement des **faces nouées** qui ont été présentés dans la section 4.

La figure C.2, quant à elle, montre les statistiques des boucles de singularité des données synthétiques après qu'elles ont été améliorées par la méthode de détection et de traitement des **faces nouées**. En effet, on remarque que le nombre et la taille des boucles ouvertes ont été réduits, alors que le nombre de boucles fermées a augmenté. La méthode des **faces nouées** vient séparer les grandes boucles ouvertes et fermées en sous boucles plus petites, ce qui permet de mieux gérer les surfaces d'erreurs (cf. Figure C.3). Et vu que les boucles ouvertes sont un alliage entre les nœuds $N_{noparent}$ et les points $N_{nochild}$, leur nombres restera constant.

Application sur les données empiriques :

La détection des boucles de singularité a pris environ `192.5 secondes` pour les données empiriques.

Les effets de l'amélioration, présentée dans la Section 3.4 sur les données empiriques sont similaires à ceux observés sur les données synthétiques. Le nombre de boucles ouvertes demeure constant, puisqu'il correspond au nombre de nœuds $N_{no\ parent}$. Cependant, en divisant les grandes boucles en plusieurs boucles de plus petite taille, la taille moyenne des boucles ouvertes diminue.

Avant l'amélioration, 1% des boucles ouvertes avaient une taille supérieure à 1748,51. Bien que ce pourcentage puisse sembler faible, il représente en réalité $0{,}01 \times 15\,684 = 156$ boucles.

(a) Boucle ouverte avant l'amélioration donc avec un premier couplage arbitraire.

(b) Boucle ouverte après l'amélioration donc avec un meilleure couplage.

FIGURE C.3 Exemple de boucle ouverte (a) qui se découpe en une boucle ouverte et une boucle fermée les deux de taille plus petite.

Ainsi, 156 boucles ouvertes étaient de très grande taille (plus de 1748,51). Trouver des surfaces d'erreur optimales pour ces boucles serait extrêmement coûteuse en termes de temps de calcul, d'autant plus que les plus grandes boucles présentent souvent des formes particulièrement complexes.

En ce qui concerne les boucles fermées, l'amélioration a permis de créer de nombreuses nouvelles boucles, leur nombre passant de 81 331 à 140 043. Ce phénomène est expliqué plus haut dans la partie des données synthétiques. Il est important de noter que la taille maximale des boucles fermées a également augmenté, passant de 468 à 2501. Cela s'explique par le fait que certaines boucles fermées extraites des boucles ouvertes sont assez grandes. Toutefois, seulement 0,048% des boucles fermées ont une taille supérieure à 463, ce qui indique que cette augmentation concerne un nombre très limité de cas.

Statistiques	Valeurs
Nombre de boucles ouvertes	15684
Taille moyenne	91.24
Taille médiane	12.0
Taille quantile 0.99%	1748.51
Taille quantile 0.90%	100.0
Taille quantile 0.10%	6.0
Taille maximale	17820
Taille minimale	4

(a) Statistiques des boucles ouvertes avant l'amélioration.

Statistiques	Valeurs
Nombre de boucles fermées	81331
Taille moyenne	8.28
Taille médiane	6.0
Taille quantile 0.90%	13.0
Taille quantile 0.10%	4.0
Taille maximale	468
Taille minimale	4

(b) Statistiques des boucles fermées avant l'amélioration.

FIGURE C.4 Statistiques et distributions des boucles ouvertes et fermés avant l'amélioration.

Statistiques	Valeurs
Nombre de boucles ouvertes	15684
Taille moyenne	42.11
Taille médiane	10.0
Taille quantile 0.90%	66.0
Taille quantile 0.10%	6.0
Taille maximale	3198
Taille minimale	4

(a) Statistiques des boucles ouvertes après l'amélioration.

Statistiques	Valeurs
Nombre de boucles fermées	140043
Taille moyenne	10.73
Taille médiane	6.0
Taille quantile 99.952 %	463.0
Taille quantile 0.90%	17.0
Taille quantile 0.10%	5.0
Taille maximale	2501
Taille minimale	4

(b) Statistiques des boucles fermées après l'amélioration.

FIGURE C.5 Statistiques et distributions des boucles ouvertes et fermés avant l'amélioration.