Fast multipole boundary element method approach for multicracked structures : application to road pavement reinforcement

Anicet Dansou

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE MSII (ED N°269)

laboratoire des sciences de l'ingénieur de l'informatique

Et de l'imagerie (ICUBE)-UMR 7357

THÈSE présentée par :

Anicet DANSOU

soutenue le : **21 novembre 2019**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : Mécanique/ Génie Civil

Méthode des éléments de frontière accélérée pour les structures multi-fissurées : Application au renforcement des chaussées

THÈSE dirigée par :

M. CHAZALLON Cyrille	Professeur, INSA de Strasbourg
M. BONNET Marc	Directeur de recherche, ENSTA Paris-Tech

RAPPORTEURS :

M. MILLARD Alain	Directeur de recherche, CEA Saclay
M. POUYA Amade	Professeur, Ecole des Ponts et Chaussées Paris-Tech

---

AUTRES MEMBRES DU JURY :

Mme MOUHOUBI Saïda	Co-encadreur, Maître de Conférences, INSA de Strasbourg
M. ZAHROUNI Hamid	Professeur, Université de Lorraine
M. CHUPIN Olivier	Chargé de Recherche, IFSTTAR Centre de Nantes
M. POUTEAU Bertrand	Directeur de recherche, Eurovia

Université de Strasbourg

Anicet DANSOU

Méthode des éléments de frontière accélérée pour les structures multi-fissurées : Application au renforcement des chaussées

Résumé

La fissuration est l'une des causes majeures de la dégradation des structures en génie civil. La modélisation numérique des fissures et de leur propagation nécessite le développement d'outils numériques performants. Cette thèse présente l'optimisation et l'extension d'un outil numérique existant, pour la simulation efficiente des problèmes de propagation de fissures dans les structures de génie civil. Le code de calcul présenté est basé sur les équations intégrales de Galerkin 3D, accélérées par la méthode multipôle rapide. Les méthodes intégrales sont performantes en mécanique de la rupture pour la détermination des champs singuliers au voisinage des fissures et présentent l'avantage de la réduction d'une dimension de maillage. La Méthode Multipôle Rapide, quant à elle, permet *via* une reformulation des fonctions fondamentales propres aux formulations intégrales, de réduire le coût des calculs. Les performances du code résultant sont améliorées dans ce travail, à travers la mise en place d'une technique de réutilisation de données, la parallélisation des parties chronophages et la proposition d'une nouvelle méthode de stockage de données. Des travaux d'extension sont également menés, pour la prise en compte des problèmes multizones complexes, le traitement des fissures débouchant en surface et l'étude de structures minces par couplage avec la méthode des éléments finis. Le code obtenu a permis de mener à bien des simulations en propagation de fissures dans des structures de chaussées. Nos travaux ont mis en évidence le rôle des grilles en fibre de verre dans le renforcement des chaussées, par limitation de la fissuration.

Mots clés : *Méthode des éléments de frontière ; Méthode multipôle rapide ; Propagation de fissures ; Renforcement des chaussées.*

Abstract

Cracking is one of the major causes of structural degradation in civil engineering. Numerical modeling of cracks and their propagation, requires the development of efficient algorithms. This thesis presents the optimization and extension of an existing numerical tool, for the efficient simulation of crack propagation problems in civil engineering structures. The presented code is based on Galerkin integral equations accelerated by the fast multipole method. Integral methods are accurate in fracture mechanics problems, for the computation of stress and displacement fields near cracks and have the advantage of reducing the discretization dimension. The calculation cost of integral methods can be reduced with the fast multipole method, which is based on a reformulation of the fundamental solutions into series of product of functions. The performance of the resulting code is improved in this work through the implementation of a data reusing technique, the parallelization of time-consuming parts and the proposal of a new method of data storage. Extension work is also carried out to consider complex multi-domain problems, the treatment of surface breaking cracks and the study of complex problems by coupling with the finite element method. The obtained code has made it possible to simulate crack propagation in road pavement structures. Our work has permitted to study the effect of fiberglass grid reinforcements on pavement cracking.

Keywords: *Symmetric Galerkin boundary element method; Fast multipole method; Crack propagation; Road pavement reinforcement.*

UNIVERSITY OF STRASBOURG

Doctoral School 269
MATHEMATIQUES, SCIENCES DE L'INFORMATION ET DE L'INGÉNIEUR
Laboratory ICUBE - INSA of Strasbourg

PHD THESIS

to obtain the title of

PhD of Science

of the University of Strasbourg
Speciality : Civil Engineering

Thesis Title

Fast Multipole Boundary Element Method approach for Multicracked Structures: Application to Road Pavement Reinforcement

Defended by
Anicet DANSOU

Jury:

Reviewers:	Mr A. Millard	-	Senior scientist, CEA Saclay
	Mr A. Pouya	-	Professor, ENPC
Thesis Directors:	Mr C. Chazallon	-	Professor, INSA of Strasbourg
	Mr M. Bonnet	-	Senior scientist, ENSTA ParisTech
Examinators:	Mrs S. Mouhoubi	-	Supervisor, Lecturer, INSA of Strasbourg
	Mr H. Zahrouni	-	Professor, University of Lorraine
	Mr O. Chupin	-	Scientist, IFSTTAR
	Mr B. Pouteau	-	Research director, Eurovia

November 21, 2019

A ma mère, mon père et ma soeur.
A tous les membres de ma famille.

A l'association A.C.E.S., en particulier à Kery James,
Omar SY, Florent MALOUDA et Claudio BEAUVUE.
A Stanislas MIGAN, Juan GOMEZ et Marcellin BOCOVE.

A toutes les personnes qui m'ont soutenu dans l'accomplissement de ce travail. Je ne saurais toutes vous citer.

Acknowledgments

This dissertation presents the research I have carried out during my PhD in the Civil Engineering team of the Laboratory ICube, National Institute of Applied Sciences (INSA) of Strasbourg, France. This research has been supported financially in part by the French National Research Agency (SolDuGri project ANR-14-CE22-0019) and in part by the Grand-Est region.

First, I would like to express my gratitude towards my thesis director, professor Cyrille Chazallon, and co-director, senior research scientist Marc Bonnet, who provided helpful advice, continuous encouragement and support. I am very grateful to my thesis supervisor, assistant professor Saïda Mouhoubi for all the precious suggestions to improve my thesis.

I would like to thank both reviewers, professor Amade Pouya and senior research scientist Alain Millard, for their careful reading of the manuscript and for thier insightful comments. I am grateful to Hamid Zahrouni, Olivier Chupin and Bertrand Pouteau for having accepted to be members of the jury. I am also grateful to Isabelle Charpentier (Scientist, ICube-CNRS) and Laurence Meylheuc (Lecturer, ICube-CNRS) for their suggestions during the sessions of the thesis supervision committee.

I would like to thank my colleagues of the civil engineering team Georg Koval, Juan-Carlos Quezada Guajardo, Hossein Nowamooz, and Safiullah Omary, for their hospitality and friendliness during my PhD. Sincere thanks to Sara Arami, Joseph Falana and Quoc Tuan Trinh for their invaluable assistance.

I would like to thank especially my fellow PhD students Peng Jing, Xiao Feng Gao, Mathias Cuny, Loba Sagnol, Leon Chiriatti, Gui Xian Liu, Hossein Assadollahi, Calypso Chadfeau, Laura Gaillard, Fu Jiao Tang, Paola Paul, Hashim Mohseni, Lei Ma, Léo Coulon, Chong Wang, Hai Tao Ge, Xiang Zhang for all the great moments we have spent together.

My gratitude also goes to the staff members, especially, Stéphanie Mathé, Vanessa Manglon, Vincent Leridez, Asvin Khisto, Sylvère Munier, Mathieu Grosse, Tristan Fauvin-Baumgartner, André Marc, Olivier Remy, for their assistance.

Abstract

In civil engineering, cracks are one of the major causes of structural degradation. Combined with water infiltration, crack propagation accelerates the destruction of structures. For this reason, the study of cracks and crack propagation is a major concern in civil engineering design, construction and maintenance. Experimental methods have been widely applied since a long time ago, but they can involve significant costs and long delays because of the equipment and samples. Numerical approaches are therefore an interesting alternative. They can provide very accurate and rapid solutions for many simple problems, however, it is still difficult to simulate accurately complex realistic engineering problems because they involve heterogeneous geometries, complicated loading and material behaviors.

The most widely used numerical method for the solution of problems in structural mechanics is the Finite Element Method (FEM). The Boundary Element Method (BEM) has emerged over the past decades as a very interesting alternative method. Although its applicability is not as wide ranging as that of FEM, there are important situations in which it can be as effective as the FEM and special contexts (unbounded domains, fracture mechanics, etc.) where it presents clear advantages over other numerical techniques. When coupled with an advanced technique namely Fast Multipole Method (FMM) for faster integral evaluations, the performance of a boundary analysis is greatly enhanced.

In our laboratory, a code (*MBEMv2.0*) has been developed based on the boundary element method and the fast multipole method for the simulation of fracture problems with an iterative solver. *MBEMv2.0* can simulate simple stationary crack problems but it encounters many issues for complex structures like cracked pavements. The computing time is also too long for crack propagation problems even for very simple geometries. That limits the use of the numerical tool to less realistic crack propagation problems. Development works are therefore initiated with the aim of making the necessary optimizations to obtain a powerful numerical code. This work has been supported financially, in part by the French National Research Agency (SolDuGri project ANR-14-CE22-0019) and in part by the Grand-Est region. The objective of *SolDuGri* project is to develop more rational and more mechanical approaches for the evaluation of fiberglass grids, and for the calculation of reinforced pavements. In this thesis, which represents part of the modeling component of the project, we study crack propagation in pavement structures as well as the influence of fiberglass grids on the cracked pavement. A new version of the code is then developed and named *MBEMv3.0*.

The fast multipole method has effectively permitted to overcome the usual bottlenecks of the boundary element method and has made the coupled fast method an excellent option. However, it is still not simple to simulate large-scale problems on moderate computational resources efficiently. Many developments have therefore been devoted to further efficiency improvements. We present a data reusing

technique which leads to the reduction of the matrix computation phase. Since multi-core computers are now very popular, we adapt the code to take advantage of multi-core environment. We use OpenMP directives to perform shared memory parallelization on time-consuming parts after reorganization work. On a parallel portion, we obtain a speedup of 13.3 while using 20 threads. We notice peaks of memory usage during the matrix computation phase, especially with the parallel code. To reduce the memory, we design a new sparse matrix method based on coordinate format and compressed sparse row format. This new method erases memory peaks and the duration of the matrix construction phase. These optimizations radically change code performance, especially for crack propagation problems for which the speed up can exceeded 50 compared to the previous version.

MBEMv2.0 has many limitations and can encounter many issues in some fracture problems. We thus perform many extension work in order to correct issues and to expand the scope of the code. We extend the existing multizone algorithm to consider complex multizone problems namely problems in which the zones can be in any configuration and the interfaces can have any orientation. The proposed algorithm allows the computation of problems with complex geometry such as composite structures. We implement a propagation law to direct the re-meshing algorithm during crack propagation. Since suitable criteria for crack propagation are still being debated, the proposed algorithm is flexible such that other laws can be added. We use the multiple node technique for the simulation of surface breaking cracks. Then an automatic multiple node algorithm is proposed to simulate the propagation of these types of cracks. One of the important extension work is the FEM-BEM coupling. A direct strategy is presented to couple the FM-SGBEM to the FEM in order to simulate the fiberglass grids. Membrane finite elements are implemented and validated, and an algorithm is proposed to take into account the finite element matrix when solving the FM-SGBEM equations. The technique can be extended in future work for other finite elements.

Finally, we apply the new code for the simulation of crack and crack propagation in multi-layered road pavements. Then we study the effects of fiberglass grid reinforcements by using FEM-BEM coupling. This study shows that the fiberglass grid can delay reflective cracking.

Résumé étendu

Contexte

En Génie Civil, la fissuration est l'une des causes majeures de la dégradation des structures. Combinée à l'infiltration de l'eau, la propagation des fissures accélère la ruine des structures. L'étude des fissures et de leur propagation est donc une préoccupation importante dans la conception, la construction et la maintenance en génie civil. Les méthodes expérimentales qui sont largement appliquées depuis longtemps, s'avèrent parfois limitées par le coût élevé des équipements auxquels elles font appel et les retards induits dans la phase de préparation des échantillons. Les approches numériques constituent souvent une alternative intéressante.

La méthode numérique la plus utilisée en mécanique des structures est la méthode des éléments finis (FEM, pour *Finite Element Method*). La méthode des éléments de frontière (BEM, pour *Boundary Element Method*) est apparue au cours des dernières décennies comme un outil numérique riche, varié et complémentaire à la méthode des éléments finis. Bien que son applicabilité ne soit pas aussi étendue que celle des éléments finis, son efficacité trouve son importance dans le traitement des domaines non bornés, des problèmes présentant de fortes concentrations de contraintes tels que ceux décrits par la mécanique de la rupture, etc. Les matrices pleines, issues de la phase de discrétisation par la BEM, limitent l'utilisation de cette dernière au traitement des domaines de petites tailles. Cet inconvénient majeur est pallié en faisant appel à un judicieux couplage avec la méthode multipôle rapide (FMM, pour *Fast Multipole Method*). Cette dernière réorganise complètement la phase de résolution en faisant appel à un processus itératif.

Dans notre laboratoire, un code (*MBEMv2.0*) a été développé avec les fondements de la méthode des éléments de frontière de Galerkin couplée à la méthode multipôle rapide pour l'étude des problèmes de la mécanique de la rupture. Un solveur itératif basé sur la généralisation de la méthode de minimisation du résidu (GMRES, pour *Generalized Minimal RESidual*) est utilisé pour la résolution des systèmes matriciels. *MBEMv2.0* peut simuler des problèmes, certes de très grandes tailles (plusieurs millions de degrés de libertés) dans lesquels les fissures restent stationnaires, mais rencontre de nombreux problèmes pour des structures complexes telles que des chaussées fissurées. Pour des problèmes de propagation de fissures, les durées de calculs sont trop longues même pour des géométries très simples, ce qui limite l'utilisation de l'outil numérique développé, à des configurations en propagation de fissures peu réalistes.

C'est dans ce contexte qu'une extension aux travaux existants, a été formulée avec l'objectif de mener à bien, les optimisations nécessaires et de disposer d'un outil numérique performant. Nos travaux de thèse viennent renforcer ces objectifs assignés au développement du code existant et s'intitulent : "Méthode des éléments de frontière accélérée pour les structures multi-fissurées : Application au renforce-

ment des chaussées".
Ce travail est soutenu financièrement par la région Grand-Est et le projet national *SolDuGri* (ANR-14-CE22-0019).

Ce dernier a vu le jour dans un contexte où les réseaux routiers vieillissent, et où les moyens consacrés à l'entretien de ces réseaux sont en diminution. Il est donc important de rechercher des solutions efficaces et durables. Dans ce domaine, les renforcements par des matériaux bitumineux intégrant des grilles en fibre de verre, s'imposent comme une solution pertinente, permettant un renforcement durable avec des économies de matériaux en comparaison avec les solutions traditionnelles. L'objectif du projet *SolDuGri*, qui associe à la fois des laboratoires de recherche et des partenaires industriels, est de développer des approches plus rationnelles et plus mécaniques pour l'évaluation des performances de ces grilles en fibre de verre, et la quantification de leurs effets sur le renforcement des chaussées.
Trois principaux verrous, constituant un frein au développement de la technique sont ciblés : (i) la compréhension des sollicitations auxquelles les grilles sont soumises lors de la mise en œuvre, afin d'optimiser la résistance de celles-ci à ces sollicitations, (ii) l'étude du comportement mécanique des interfaces entre les couches renforcées et leur support afin d'optimiser les caractéristiques des interfaces, (iii) l'amélioration de la prévision des durées de vie des chaussées renforcées, par l'étude et la modélisation du comportement en fatigue des grilles, des enrobés renforcés et des interfaces, et par la validation par un essai de fatigue en vraie grandeur sur le manège de fatigue. Dans cette thèse, qui représente une partie du volet modélisation du projet, on souhaite étudier la propagation des fissures dans les structures de chaussées ainsi que l'influence de la présence des grilles sur l'état de fissuration de ces dernières. Une nouvelle version du code (*MBEMv3.0*) est donc développée pour une simulation efficace des problèmes de propagation de fissures dans des structures complexes.

Plan de Mémoire

Ce mémoire est composé de sept chapitres dont le premier se résume à l'introduction générale. Ce premier chapitre introductif exhibe les méthodes classiques de modélisation de la propagation de fissures, une description du code numérique existant et initie le lecteur aux objectifs principaux assignés à nos travaux.

Dans le deuxième chapitre, les bases théoriques de la méthode des éléments de frontière et de la méthode multipôle rapide sont exposées dans le cadre de l'élasticité et de la mécanique linéaire de la rupture en 3D.

Le contenu du *chapitre 3* est dédié à la description des travaux d'optimisation numériques. Une technique de réutilisation des données y est présentée. Celle-ci permet d'éviter la reconstruction complète des matrices lors des simulations de propagations de fissures. Une phase de parallélisation en mémoire partagée avec OpenMP est menée sur des parties chronophages du code. Aussi, une nouvelle méthode de stockage de données est proposée pour éviter les pics d'utilisation de la mémoire, observés lors de la construction matricielle. Ces travaux d'optimisation

menés avec succès ont permis la réalisation de simulations de problèmes de grandes tailles.

Les différentes extensions au code existant sont présentées dans le *chapitre 4*. Une extension pour tenir compte des domaines multizones dans lesquels les interfaces séparant les zones peuvent être orientées de manière quelconque, est proposée et mise en œuvre numériquement. L'étude de la propagation de fissures par fatigue selon la *loi de Paris* y est décrite également. Les travaux d'extension du code ont concerné également l'étude de fissures débouchant sur une surface ou sur une interface. Divers tests de validation y sont aussi détaillés et présentés.

L'introduction de la grille en fibre de verre pour renforcer les structures a nécessité la construction d'une procédure de couplage judicieux avec la méthode des éléments finis. Les étapes menant à la construction de cette procédure sont présentées dans le *chapitre 5*. La stratégie de couplage direct est implémentée avec des éléments finis de type *membrane* et les différents exemples de validation attestent de la qualité de la mise en œuvre numérique.

Les applications pour l'étude de chaussées réelles multi-fissurées sont présentées avec détails dans le *chapitre 6*.

Le *chapitre 7* comporte les conclusions générales et discute des perspectives d'améliorations et de développements.

Trois annexes viennent renforcer le contenu de notre mémoire et donnent les détails sur les formulations des méthodes intégrales de Galerkin (Annexe A), sur celles de la méthode multipôle rapide (Annexe B) et dressent également un bref guide pour l'utilisation du code (Annexe C).

Eléments de frontière et méthode multipôle radide

Formulations intégrales de Galerkin

Les équations aux dérivées partielles régissant les problèmes de la mécanique des solides sont bien connues et peuvent être écrites facilement. Pour un solide élastique linéaire isotrope, ce sont les équations de Navier. En se basant sur une identité de réciprocité et sur une solution fondamentale (solution de Kelvin pour l'espace infini élastique), ces équations peuvent s'écrire sous une forme intégrale. Cette dernière permet de calculer les champs inconnus en tout point du domaine considéré lorsque ces champs sont connus à la frontière du domaine. Un processus de passage à la limite (*régularisation*) permet ensuite d'obtenir les équations à résoudre pour l'obtention des champs sur la frontière. La méthode des éléments de frontière se base sur la discrétisation des équations intégrales dont le support des inconnues est réduit à la frontière du domaine.

Notre travail fait appel essentiellement aux fondements théoriques des méthodes intégrales de Galerkin. Celles-ci sont basées sur une formulation variationnelle. Un principe variationnel fait intervenir une quantité scalaire, appelée fonctionnelle s'écrivant sous forme d'une intégrale où toutes les relations définissant le problème considéré sont présentes. La solution du problème rend stationnaire

la fonctionnelle ainsi définie. L'avantage du principe variationnel est qu'il permet d'obtenir des matrices symétriques. Cette propriété est très intéressante car elle facilite la résolution du système obtenu. Les équations dérivant du principe intégral variationnel de Galerkin se présentent sous des formes bilinéaires de type $\mathcal{I}(E_1, E_2) = \int_{E_1}\int_{E_2} K(\mathbf{x}, \mathbf{y})$ avec $K(\mathbf{x}, \mathbf{y}) \in O(r^{-1})$ et la détermination de chacune de ces dernières consiste à évaluer des doubles intégrales de surface portant sur deux supports géométriques de type surfacique E1 et E2 parcourus respectivement par les deux points d'intégration $\mathbf{x}$ et $\mathbf{y}$.

La phase de discrétisation des formulations théoriques conduit à la construction de systèmes matriciels symétriques, de tailles réduites. Néanmoins, ces derniers présentent l'inconvénient majeur d'être pleins, ce qui pénalise considérablement la phase de résolution lorsqu'on traite de structures de grandes tailles. La mise en place d'une procédure de couplage de ces formulations avec la méthode multipôle rapide (FMM) permet de s'affranchir de cette difficulté majeure. Lors de la phase d'intégration numérique, nous distinguerons donc le traitement des intégrales portant sur deux éléments éloignés, de celui des intégrales portant sur deux éléments proches. Dans le premier cas, le nombre d'intégrales à évaluer reste important et cette particularité s'accentue avec la taille du problème traité. Le recours à la méthode multipôle rapide pour les éléments éloignés permet donc de s'affranchir du stockage des matrices issues de la phase de discrétisation de ces intégrales. Dans le second cas, les matrices sont explicitement définies et stockées dans une matrice nommée $[Knear]$. Cette dernière est utilisée lors de la phase de pré-conditionnement du système matriciel.

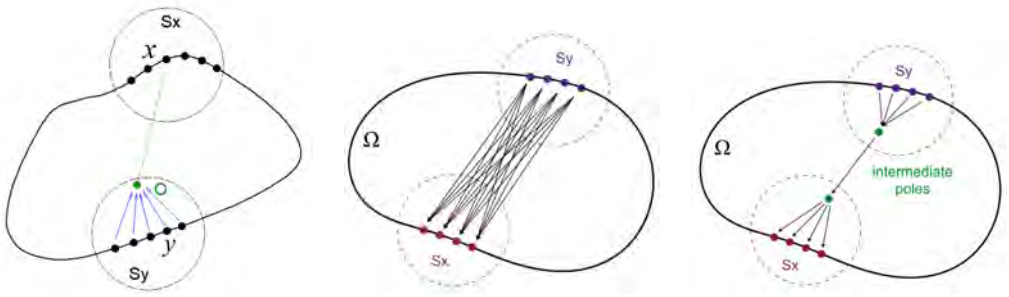
Méthode multipôle rapide

La méthode multipôle rapide (FMM) est basée sur la reformulation des noyaux constituant les fonctions fondamentales en séries multipôles ($K(\mathbf{x}, \mathbf{y}) \simeq \sum_i \phi(\overrightarrow{Ox})\psi(\overrightarrow{Oy})$) de manière à ce que les variables $\mathbf{x}$ et $\mathbf{y}$ de l'intégrale soient séparées. Le vecteur $\mathbf{r} = \mathbf{x} - \mathbf{y}$ est décomposé en $\mathbf{r} = (\mathbf{x} - \mathbf{O}) - (\mathbf{y} - \mathbf{O})$. $\mathbf{O}$ est un pôle choisi de manière à ce que $\overrightarrow{Oy} < \overrightarrow{Ox}$.

Une intégrale générique $\mathcal{I} = \int_{S_x}\int_{S_y} f(\mathbf{x})K(\mathbf{x}, \mathbf{y})g(\mathbf{y})dSydSx$ peut être évaluée par

$$\mathcal{I} \simeq \sum_i \int_{S_x} f(x)\phi(\overrightarrow{Ox})M_i(O)dSx \tag{1}$$

avec le multipôle moment $M(O) = \int_{S_y} \psi(\overrightarrow{Oy})g(\mathbf{y})dSy$. Dans cette expression de $\mathcal{I}$, les variables $\mathbf{x}$ et $\mathbf{y}$ étant séparées, il n'est plus nécessaire de recalculer les solutions fondamentales pour chaque couple de points. Il est donc possible de réutiliser les intégrations précédentes selon $\mathbf{x}$. Les contributions mutuelles entre tous les points $\mathbf{x}$ et $\mathbf{y}$ sont ainsi réduites à quelques contributions entre paquets de points $\mathbf{x}$ et paquets lointains de points $\mathbf{y}$. Ce principe permet une accélération considérable de la phase d'évaluation des intégrales doubles lors de chaque itération propre au calcul par



(a) (b) (c)

Figure 1 – FMM : (a) Illustration (b) Algorithme standard (c) Algorithme rapide

la méthode des éléments de frontière. La méthode multipôle rapide étendue aux concepts des méthodes intégrales permet d'effectuer les produits matrice-vecteur en un temps proportionnel au nombre d'inconnues nodales N alors que l'approche classique demande des temps de calculs assez prohibitifs (proportionnel à N^2). De plus, le coût d'utilisation de la mémoire centrale est considérablement réduit car la matrice du système n'est jamais explicitement assemblée contrairement à une analyse de frontière classique.

Code initial: *MBEMv2.0*

Dans notre laboratoire, un code (*MBEMv2.0*) a été développé sur la base de la méthode des éléments de frontière de Galerkin couplée à la méthode multipôle rapide pour la simulation des problèmes de mécanique de la rupture. Ce code permet la simulation des problèmes dans lesquels les fissures sont stationnaires. Lorsque les problèmes deviennent complexes (simulation de chaussées fissurées) ou lorsqu'il s'agit de traiter de la propagation de fissures, les durées de calculs deviennent trop longues. Considérons la propagation de 64 fissures circulaires dans un domaine homogène cubique en traction. Le nombre de degrés de liberté varie lors de la propagation de 40 974 à 105 486. Le nombre d'itérations et les durées de calculs sont présentés au tableau 1 pour chaque cycle de propagation. Dans ce tableau, N_{ddls} représente le nombre de degrés de liberté, T_{pre} est la durée de la phase de préparation du système matriciel, T_{sol} est la durée de la phase de résolution itérative et T_{tot} est la durée totale du cycle considéré. On peut remarquer que la durée d'un cycle de calcul passe d'une heure et 45 minutes à l'état initial à 40 heures au huitième incrément. Le nombre d'itérations aussi est croissant et le solveur n'a pas convergé au neuvième incrément après 250 itérations. *MBEMv2.0* n'est donc pas adapté pour une résolution efficace des problèmes de propagation de fissures et l'optimisation du code s'avère indispensable. La durée totale de calcul est principalement composée de la durée T_{pre} de préparation du système matriciel et de la durée T_{sol} de résolution itérative. Cette dernière dépend de la durée T_{iter}

d'une itération et du nombre d'itérations N_{iter}. Il faudrait ainsi envisager toute solution visant à réduire T_{pre}, T_{iter} ou N_{iter}.

Table 1 – Propagation de 64 fissures avec *MBEMv2.0*

#	N_{ddls}	N_{iter}	T_{pre} (s)	T_{sol} (s)	T_{tot} (s)
1	40 974	41	972	5293	6268
2	50 190	59	1818	9992	11 814
3	59 406	89	2904	18 624	21 533
4	68 622	100	8165	25 068	33 238
5	77 838	75	11 499	21 634	33 141
6	87 054	84	17 347	29 179	45 535
7	96 270	96	24 577	36 963	61 549
8	105 486	248	32 670	111 315	143 997

Travaux d'optimisation

Accélération du calcul des matrices

Lors des simulations de propagation de fissures, on observe une croissance de la durée de calculs des matrices comportant les doubles intégrales qui interviennent dans la SGBEM. Dans l'algorithme initial, à chaque cycle, de nouveaux éléments sont ajoutés à la géométrie et le système (en particulier la matrice K_{near}) doit être recalculé. Si l'on reconstruit totalement la matrice, les interactions entre des paires d'anciens éléments seront répétées et nécessiteront des opérations inutiles, car rien *a priori* ne change dans le calcul de ces interactions. Par conséquent, à partir du deuxième cycle, les interactions entre des paires d'anciens éléments sont réutilisées. Seules les parties de la matrice liées aux éléments nouvellement ajoutés sont calculées. Réutiliser les interactions entre des paires d'anciens éléments pour calculer K_{near} est une idée simple, mais il faut s'assurer de conserver les mêmes configurations que le calcul initial. Par exemple, la structure *octree* permettant le déploiement de la méthode multipôle rapide, doit être fixée d'un cycle à l'autre pour que les éléments proches restent proches et pareillement pour les éléments éloignés. Une attention particulière doit être aussi accordée aux éléments en front de fissures qui subissent des modifications d'un cycle à l'autre. Une fois les configurations conservées, cette procédure de mise à jour a permis de réduire la durée de la phase de construction des matrices d'un facteur variant de 2.2 à 6.6 sur les modèles de cubes et de chaussées simulés. Dans le même sens, une partie de la solution obtenue à un cycle donné est utilisée comme solution initiale du solveur itératif au cycle suivant. Cela a permis de réduire le nombre d'itérations nécessaire au solveur itératif pour atteindre la précision souhaitée.

Parallélisation

La parallélisation est une technique permettant de diviser une tâche principale en sous-tâches pouvant être exécutées simultanément. On réduit ainsi la durée d'exécution de la tâche principale. Le domaine de la parallélisation est très vaste, nécessite des connaissances spécifiques et les techniques utilisées dépendent des environnements numériques ciblés. Dans notre cas, nous disposons d'un ordinateur comportant vingt cœurs de calcul et qui fonctionne en mémoire partagée. Un très bon moyen de paralléliser un code existant sur un tel environnement numérique sans engendrer des modifications majeures est l'utilisation de l'interface OpenMP (Open Multi-Processing). Cette interface de programmation pour le calcul parallèle se présente sous la forme d'un ensemble de directives, d'une bibliothèque logicielle et de variables d'environnement. Elle permet de développer rapidement des applications parallèles en restant proche du code séquentiel. Les parties chronophages du code sont d'abord identifiées. Ensuite, elles sont réorganisées pour être optimisées d'une part et pour être parallélisables d'autre part. Enfin, les directives OpenMP sont utilisées pour le partage des tâches et la synchronisation des résultats. Une accélération significative du code est obtenue. Sur une portion parallélisée par exemple, on obtient une accélération de 13,3 en utilisant 20 cœurs.

Stockage de matrices

Lors de la simulation des problèmes à grande échelle, l'utilisation de la mémoire nécessite une attention particulière, en particulier dans un contexte de calcul parallèle. Lors de la phase de construction des matrices, l'espace des matrices est d'abord réservé en mémoire, puis après calcul, les matrices sont compressées en format CSR (Compressed Sparse Row). Durant les calculs, il y a donc une importante partie de la mémoire qui est réservée inutilement. La parallélisation de la phase de construction des matrices accentue ce problème d'utilisation de la mémoire si bien que certains problèmes peuvent nécessiter un espace mémoire valant plus de cinq fois la mémoire réellement utilisée. Une nouvelle méthode de stockage est alors proposée. L'espace mémoire est réservé de façon incrémentielle, les matrices sont enregistrées en format coordonnées lors du calcul et elles sont comprimées en format CSR au fur et à mesure. Cette méthode permet de réduire de manière très significative l'espace mémoire nécessaire aux simulations.

Performances

Le concours de l'ensemble des optimisations mises en place dans le cadre de notre travail confère au code de réelles performances, plus particulièrement lorsqu'on traite de la propagation des fissures. Une accélération de 52.6 est par exemple obtenue pour le problème de propagation de 64 fissures circulaires dans un domaine homogène. Il a fallu 5 jours au code initial pour la simulation de 10 cycles de propagation de ce problème mais seulement 2 heures et 17 minutes après optimisations. La Figure 2 ci-dessous, montre l'effet des optimisations sur les performances

du code. Il s'agit de la propagation de 8 fissures circulaires dans un domaine homogène et l'accélération totale est évaluée à 35.3.

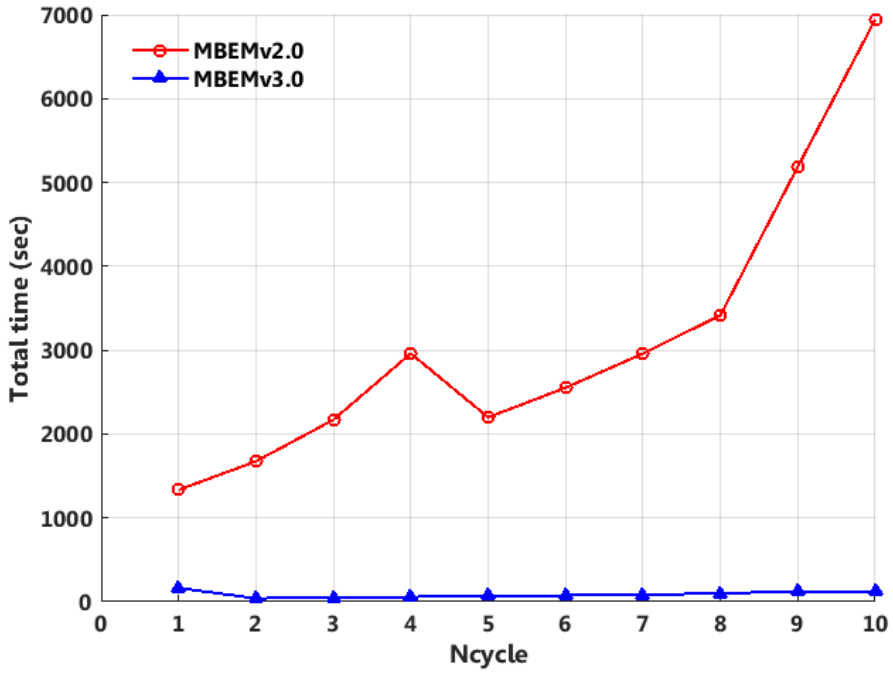


Figure 2 – Comparaison de performances : évolution de la durée totale

Travaux d'extension

Problèmes multizone complexes

L'algorithme multizone implémenté dans le code initial est limité à la simulation de domaines multizones dans lesquels les zones sont verticalement superposées. Plusieurs limitations sont également identifiées dans la définition de la géométrie : les orientations des surfaces sont par exemple fixées. Afin de simuler des problèmes multizones complexes, un nouvel algorithme multizone est proposé en se basant sur l'écriture des conditions de continuité et d'équilibre en termes de déplacements et de tensions aux interfaces. Grâce à cet algorithme, la simulation de domaines multizones avec des interfaces à orientation quelconque est possible. L'utilisateur a plus de liberté dans la définition des zones, des interfaces et de leur orientation. Les problèmes complexes où plus de trois zones intersectent peuvent poser des difficultés. En effet, il y a plusieurs inconnues de tensions aux nœuds d'intersection des zones. Ces nœuds spéciaux sont traités en utilisant la technique de nœuds multiples. Cela permet de calculer les différentes tensions sur chaque interface. Le nouvel algorithme proposé permet le calcul des domaines hétérogènes.

Fissures de fatigue et fissures débouchantes

La propagation des fissures ne suivait aucune loi dans le code précédent. Dans ce travail, une loi de propagation simple (Loi de *Paris*) est implémentée pour diriger l'algorithme de remaillage. Étant donné que des critères appropriés pour la propagation des fissures font encore l'objet de beaucoup de recherches, l'implémentation effectuée ici est flexible et donne la possibilité d'ajouter sans difficultés de nouvelles lois de propagation. La technique de nœuds multiples est aussi utilisée pour traiter les fissures débouchant sur une surface frontière. Les multiples inconnues (déplacements inférieur et supérieur, saut de déplacement) sont ainsi bien prises en compte. La technique est aussi étendue à l'étude des fissures débouchant sur une interface. Dans ces cas, en plus des multiples inconnues précédemment citées, des inconnues en tensions doivent être aussi prises en compte. Une fois ces fissures traitées, l'étude de leur propagation est explorée à travers la mise en place d'un algorithme détectant et multipliant automatiquement les nœuds spéciaux. Cette contribution permet d'envisager le calcul de structures réelles fissurées, car les fissures qu'elles comportent sont souvent des fissures débouchant en surface ou démarrant d'une interface (un joint par exemple).

Couplage avec la méthode des éléments finis

Afin de pourvoir modéliser les grilles en fibre de verre qui sont des éléments très minces, situation pour lesquelles la méthode des éléments de frontière trouve de grandes limitations, une procédure de couplage direct avec les éléments finis de type *membrane* a été développée dans le cadre de nos travaux. Des tests de validation et des simulations numériques ont permis d'attester de la qualité des résultats obtenus.

Application à des structures de chaussées

La dernière étape de nos travaux a consisté en l'étude de structures de chaussées réelles. Dans un premier temps, la déflexion sous l'effet du chargement de véhicules est étudiée pour des chaussées non-fissurées et les résultats ont été confrontés à ceux issus de la méthode des éléments finis. Dans un second temps, l'effet de la présence de plusieurs fissures transversales sur la déflexion de la chaussée est étudié, avant d'étendre nos investigations à la prise en compte de la propagation de fissures dans la première couche de la chaussée. Une attention particulière est portée au cas de la remontée des fissures. Enfin, la procédure de couplage a permis de mettre en évidence le rôle que joue la grille en fibre de verre dans la limitation de la fissuration.

Conclusions et perspectives

Dans cette thèse, les problèmes de propagation de fissures sont modélisés avec succès avec un code de calcul numérique basé sur la méthode des éléments de frontière couplée à la méthode multipôle rapide. Plusieurs travaux d'optimisation sont menés

pour réduire les durées de calcul. Les environnements développés permettent de mener des études de propagation de fissures sur des structures à plusieurs millions de degrés de liberté, dans un temps optimisé. Un couplage avec les éléments finis de type membrane est également proposé pour la modélisation de structures très minces telles que les grilles de renforcement. Ce couplage permet d'étudier l'influence de la présence des grilles en fibre de verre sur la remontée de fissures dans des chaussées renforcées. Les résultats encourageants obtenus laissent entrevoir des perspectives intéressantes.

Perspectives à court terme

Un traitement particulier sera réservé à l'étude de la propagation de fissures démarrant aux interfaces et en couche de roulement (Fig. 3). Aussi, nous nous proposons d'adapter les environnements de l'outil numérique *MBEMv3.0* pour assurer un pilotage en déplacement, et ce conformément aux conditions réelles des essais menés *in-situ* sur les chaussées. La gestion de la propagation de fissures nécessite également des améliorations. Cette phase sera menée en intégrant des éléments triangulaires.

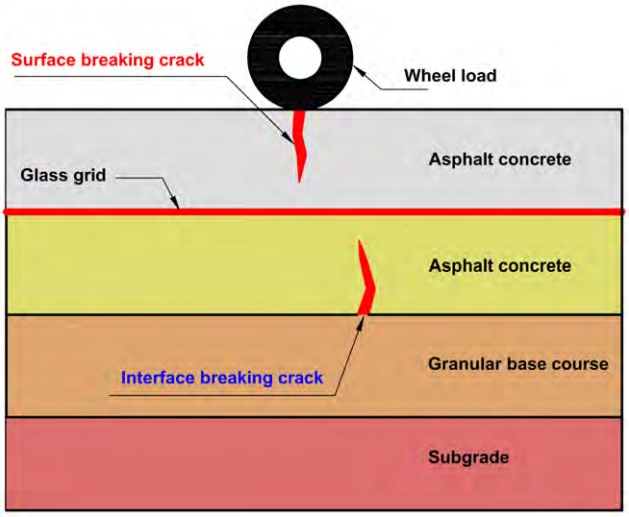


Figure 3 – Perspectives : propagation de fissures dans une chaussée

Perspectives à moyen terme

Une orientation majeure au développement du code reposera sur une meilleure prise en compte de la micro-structure du béton bitumineux en mécanique linéaire de la rupture, en intégrant dans les domaines fissurés, des inclusions polygonales rigides pour la simulation des essais de fatigue : flexion alternée et traction/compression.

Dans le cadre de son rayonnement régional et national, notre équipe a adhéré à l'Institut Thématique Inter-disciplinaire recherche et formation [*ITI: G-EAU-TE*

(Labex)]. Dans le cadre de ce projet, nos travaux doivent s'orienter et s'adapter au développement d'approches de mise à l'échelle pour évaluer des propriétés des réservoirs (perméabilité, conductivité thermique, modules élastiques) à une échelle de 100 m. Parallèlement, les effets des cycles de température et de pression hydrique, considérés comme des charges de fatigue ou des chargements rapides pouvant entrainer des dommages importants, doivent être pris en compte dans nos considérations de développements futurs. L'outil numérique ainsi étendu devra permettre à son utilisateur de mener des analyses portant sur le comportement et l'intégrité dans le temps des réservoirs et des infrastructures soumis aux cycles de chargements ainsi décrits.

Principales publications associées aux travaux

Articles dans des revues internationales à comité de relecture [1, 2]

- A. Dansou, S. Mouhoubi, C. Chazallon and M. Bonnet. Modeling multi-crack propagation by the Fast Multipole Symmetric Galerkin BEM. *Engineering Analysis with Boundary Elements*, 106:309-319, 2019.

- A. Dansou, S. Mouhoubi and C. Chazallon. Optimizations of a fast multipole symmetric Galerkin boundary element method code. *Numerical Algorithms*, 2019.

Articles dans des revues nationales [3]

- C. Chazallon, S. Mouhoubi and A. Dansou. Modèles de dégradation des structures. *Revue générale des routes et de l'aménagement*, 963, 2019.

Conférences internationales [4, 5]

- A. Dansou, S. Mouhoubi, C. Chazallon and M. Bonnet. Modeling crack propagation in 3D heterogeneous multi-cracked roads by Fast Multipole Symmetric Galerkin Boundary Element Method. *Symposium of the International Association for Boundary Element Methods (IABEM)*, Paris, France, 2018.

- A. Dansou, S. Mouhoubi and C. Chazallon. Data reusing techniques to accelerate a crack propagation boundary element method code. *VI International Conference on Computational Modeling of Fracture and Failure of Materials and Structures (CFRAC)*, Braunschweig, Germany, 2019.

Conférences nationales avec article [6]

- A. Dansou, S. Mouhoubi, C. Chazallon and M. Bonnet. A parallel boundary element method code to simulate multi-cracked structures. *14ème Colloque National en Calcul des Structures (CSMA)*, Presqu'île de Giens, France, 2019.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

Contents

1.1 General overview

This work done at the laboratory of engineering sciences, computer science and imaging (ICube, UMR 7357, CNRS-INSA Strasbourg) and in close collaboration with the laboratory of mathematical study and simulation of wave propagation (POEMS, UMR 7231, CNRS-INRIA-ENSTA) has been supported financially in part by the French National Research Agency (SolDuGri project ANR-14-CE22-0019) and in part by the Grand-Est region. The objective of *SolDuGri* project is to develop more rational and more mechanical approaches for the evaluation of fiberglass grids, and for the calculation of reinforced pavements. Three important problems, which delay the development of fiberglass grid reinforcement are targeted: (i) a better understanding of the loads to which the grids are subjected during the installation to optimize their resistance to these solicitations, (ii) the study of the mechanical behavior of the interfaces between the reinforced layers and the structure in order to optimize the characteristics of the interfaces, (iii) the improvement in the prediction of the duration of reinforced pavements by studying and modeling the fatigue behavior of grids, reinforced asphalt and interfaces, and validating the approach with a full-scale fatigue test.

This thesis is a part of the third targeted task. The main objective is the simulation of 3D crack propagation in civil engineering structures especially roads reinforced by fiberglass grids. Previous works provided a Fortran code based on boundary element method and fast multipole method for fracture problems, but they lead to high CPU costs and memory requirements. The main goal of this thesis is to improve and extend this code to crack propagation simulation and study the effect of the fiberglass grids on crack propagation. To this end, many acceleration

techniques are developed: data reusing techniques, parallel implementation, sparse matrix construction method. The code is then extended to take into account among others: complex multizone configurations and surface breaking cracks propagation. A coupling with finite element method is finally achieved to model the effect of fiberglass grids on reinforced pavements.

1.2 Crack propagation in roads

Cracks are one of the major causes of pavement degradation. Several types of cracks exist with different levels of severity and various causes. Fig. 1.1 presents some examples. The reflection of an existing crack or the thermal shrinkage of asphalt layer can lead to transverse cracking. Fatigue or poor joint construction or location can lead to longitudinal cracks. Soil movements (cycle of freezing and thawing, settlement, water withdrawal, etc.) can also lead to cracking. A load-related deterioration resulting from a weakened base course or sub-grade, too little pavement thickness, overloading, or a combination of these factors lead to interconnected cracks which resemble an alligator skin called fatigue cracks or alligator cracks. Excessive weight (heavier vehicles) or prolonged periods of stationary weight especially during the heat of summer can also make cracks occur. If they are not repaired, small cracks will propagate into large cracks. Combined with water infiltration, the crack propagation accelerates the degradation of pavements. For this reason, various complementing aspects of crack and crack propagation are studied: crack detection, in-situ measurements, laboratory tests, theory development, numerical modeling, etc. In this study, we are interested in the numerical modeling of crack propagation.

1.3 Modeling crack propagation

Experimental methods have been widely applied since a long time ago to predict crack propagation in structures. For example, Heyder and Kuhn [7] have studied 3D fatigue crack propagation on PMMA (poly methyl methacrylate) specimens. More recently, Dong et al. [8] have studied unstable crack propagation velocity on pipeline steel. However numerical methods are very interesting because several destructive experimental tests which can entail important costs and long delays can be avoided. Cracking is a phenomenon that occurs at all scales of the material: from the atom to the structure. Small-scale cracking on the scale of an infinitesimal volume, for example, can be studied by damage models. They consist in modeling the effect of all the cracks in this volume by decreasing their stiffness, that is to say by including this effect in the model of the behavior of the material. On a large scale, cracks in the structure are modeled explicitly and their growth is simulated using propagation criteria. The theory allowing this modeling is fracture mechanics.

Generally, in fracture mechanics the prediction of the crack propagation is based on parameters (energy-release rate, stress intensity factor, etc.) whose values de-

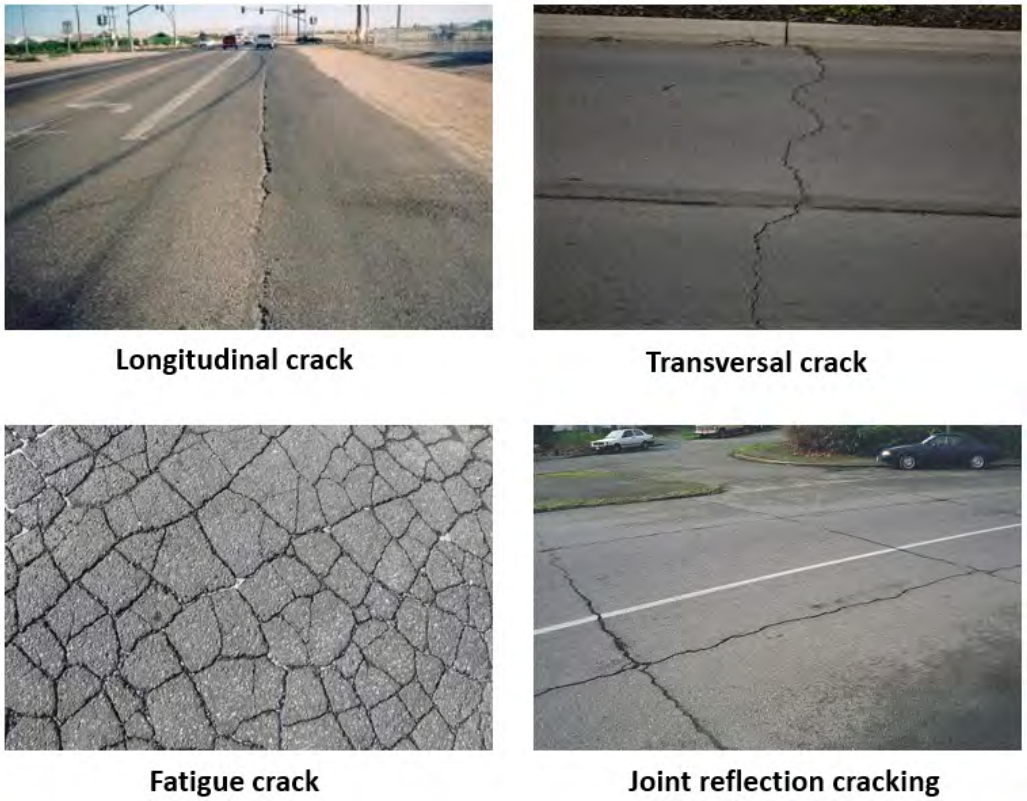


Figure 1.1 – Crack types

pend on the problem considered. For very simple fracture problems, analytic solutions exist, but very few practical problems can be solved analytically. Thus, numerical modeling has become an indispensable tool in fracture analysis. Numerical modeling permits the publication of the solutions of several fracture mechanics problems. With the continual increase of computational resources, and the development of more efficient numerical methods, realistic simulations of fracture problems in structures have become possible. The most applied classical numerical methods in fracture mechanics are the finite element method, the extended finite element method and the boundary element method. We will now very briefly review the main characteristics of these methods in the framework of crack propagation.

Finite element method (FEM)

Among the numerical methods, the finite element method (FEM) is the most widely used in fracture mechanics. The computation of accurate values of stress and displacement fields and of the stress intensity factor for the crack, is the first issue in FEM. For that purpose, the simple methods used are the stress method, the displacement method and the line integral method. These methods require a high degree of refinement at the crack tip. To avoid this refinement, special crack tip

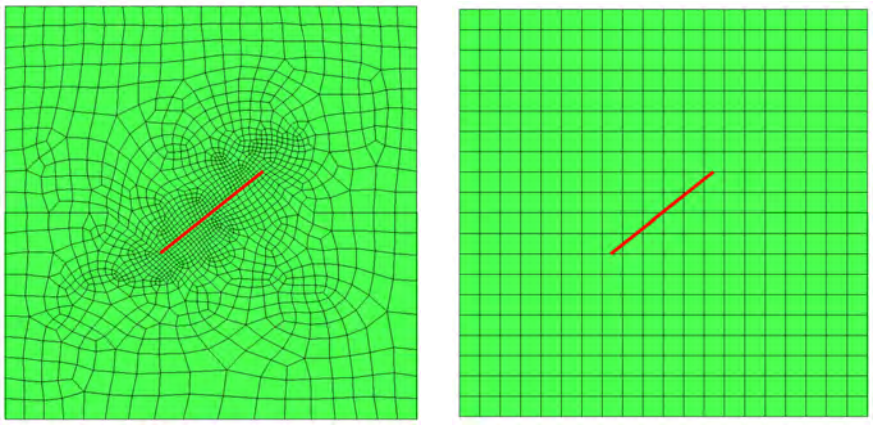
elements were developed to describe the singularity in the near-crack stress field. Tracey in [9] introduced a new type of finite element which embodies the inverse square root singularity present near a crack in an elastic medium. Barsoum in [10] used quadratic isoparametric elements which embody the inverse square root singularity in the calculation of stress intensity factors of elastic fracture mechanics. Bittencourt et al. in [11] presented a strategy for quasi-automatic simulation of crack propagation in two-dimensional, linear elastic finite element models. Based on the stress intensity factors (SIFs), the direction of the crack growth is evaluated and then the crack propagates. In order to re-evaluate the SIFs, a local re-meshing is performed at the new crack tip. This re-meshing is highly cumbersome specially in 3D problems and when the crack closure induced by plasticity is taken to account. Although several crack advancement techniques are developed (for example in Maligno et al. [12]), an accurate crack propagation simulation with finite element method using acceptable computational resources is still a difficult task. That is why crack propagation studies by pure FEM only concern one crack, otherwise very few, see for example crack closure analysis presented by Lei [13].

Extended finite element method (XFEM)

In order to avoid mesh dependency and the re-meshing of the FEM, a mesh-independent method with minimal re-meshing is published in 1999 by Belytschko and Black [14]. This method is then developed to a mesh-independent method without any re-meshing by Moës, Dolbow and Belytschko [15]. The method has later become known as the eXtended Finite Element Method (XFEM) [16, 17], and has become widely popular [18] for solving continuum mechanics problems containing discontinuities like cracks. A non-conforming mesh is used to model the crack. Special shape functions are added to the elements cut by the crack to take care of the local discontinuities and singularities around the crack [16]. So the cracks are modeled independently of the mesh [17] and the same mesh can be used for all stages of a growing fatigue crack. Fig. 1.2 shows a conceptual comparison between FEM and XFEM for a 2D cracked domain. Despite this independence, Ren and Guan [19] illustrate that the level of mesh refinement of the crack influences the accuracy of the representation of a three-dimensional crack.

Although XFEM can provide good predictions for the fatigue crack propagation in simple geometries (Sukumar, et al. [20]), the accuracy of the method decreases for complex problems (Bergara, et al. [21]). On the other hand, the method requires a large number of nodes and long computation time. Sukumar et al. in [22], studied the propagation of an inclined penny crack, embedded in a cube with a mesh consisting of 17 000 nodes (51 000 dofs). It is interesting to note that the authors did not report the exact computation time for this calculation, but stated: *On average, the time taken to compute a single crack growth step is between 30 to 45 minutes.* Therefore, a total of 20 propagation steps need between 10 to 15 hours for only one crack. Furthermore, after a comparative study on finite

element enrichments, Oliver et al. concluded in [23] that the computational cost of X-FEM increases linearly with the number of involved cracks. For the 3D case they considered this increase was up to around 20% of the total cost per every additional crack. The multi-cracked structures that interest us here will therefore be very difficult to study since they can involve hundreds or thousands of cracks. Researchers are still working to improve the technique for the simulation of complex problems like three-dimensional multiple crack propagation.



(a) (b)

Figure 1.2 – 2D cracked domain: (a) FEM mesh (b) XFEM mesh

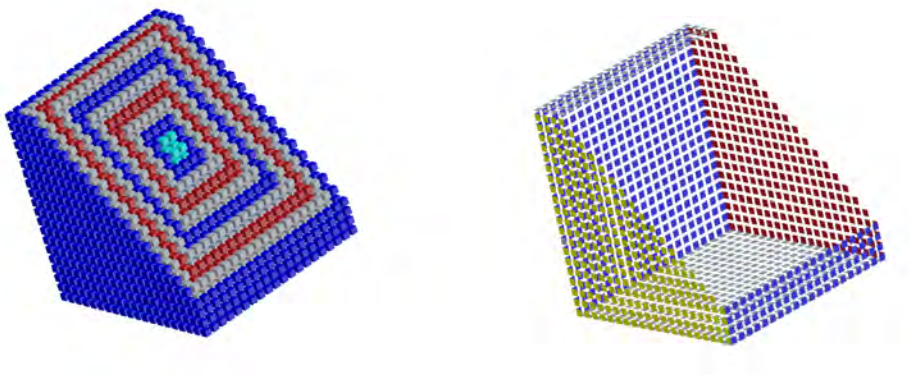
Boundary element method (BEM)

The boundary element method has emerged as a powerful alternative to FEM particularly in cases where better accuracy is required due to problems such as stress concentration. The most important feature of BEM is that the solution is approximated at the boundaries, while equilibrium and compatibility are exactly satisfied in the interior of the domain. The advantage of limiting the discretization to the boundaries is that the problem is reduced by one order. So, for a 3D problem, only the boundary surface is meshed rather than the volume, see Fig. 1.3. The technique in BEM consists of transforming the differential equations governing the entire domain to integral equations that only consider boundary values. Then, a numerical scheme is set up to solve the boundary integral equations. To do so, the boundary of the problem is discretized into elements, where the unknown source functions (generalized displacements) are assumed to vary using polynomials (constant, linear, quadratic, etc). The source functions are approximated via nodal values. To solve the integral equations, the error can be set to be zero at each node. This is called the point *collocation* technique. The element domain can also be chosen for the collocation, this is called the *Galerkin* technique which gives much less error than the collocation. The BEM is thus well suited for crack propagation problems since only the crack needs to be re-meshed during the propagation. In addition,

stress and displacement fields of cracks and other singularities can be computed accurately.

However, the method has disadvantages that should be specified. First, the method requires knowledge in mathematics that may seem unfamiliar to engineers. The mathematics used in BEM (integral equations and their numerical solution, regularization techniques, etc.) represent a classical part of mathematical analysis but they are still not very popular within applied mathematics and engineering. Also, BEM requires the explicit knowledge of a fundamental solution of the partial differential equation (PDE). This is available only for linear PDEs with constant or some specifically variable coefficients. Nonlinear problems or problems with inhomogeneities are difficult by pure BEM (In nonlinear problems, the interior must be modeled). For these problems, coupling of FEM and BEM is often applied. For thin structures, inaccuracies occur in the numerical integrations. This will be discussed in section 6.1.3.

For all its advantages and despite the limitations, the BEM is used in this work. The method will be further detailed in following chapters.



(a) (b)

Figure 1.3 – 3D problem: (a) FEM mesh (b) BEM mesh

In this work the symmetric Galerkin boundary element method (SGBEM) is used. Unlike the traditional collocation BEM, SGBEM yields symmetric coefficient matrix and presents an important gain from a numerical point of view as it reduces the computational cost and also permits BEM/FEM coupling (see references [24–26]). Another advantage of the SGBEM is the ability to treat hypersingular and singular integrals solely by means of standard continuous elements. For this reason, the SGBEM can easily capture the crack tip behavior and provide a smoother solution in the neighborhood of geometric discontinuities.

Over recent decades, the performance of boundary analysis is further improved with the advent of the Fast Multipole Method (FMM) [27] and other acceleration methodologies. The classical bottlenecks of the BEMs caused by the fully populated matrix are alleviated as the FMM splits all element integrals into *near-field* and *far-*

field interactions, the latter being clustered in a recursive, multilevel fashion. This process results in (i) a lessened storage complexity, typically defined by the sparse *near-field* matrix, and (ii) faster solution based on iterative solvers (complexity of order $O(N)$ instead of $O(N^2)$ per iteration, N being the number of BEM unknowns). This makes boundary element analysis applicable to large BEM models with very good performance. Takahashi et al. [28] applied the FMM for three-dimensional elastodynamics in time domain, Chaillat et al. [29] applied it in the frequency domain. Bapat et al. [30] solved 3-D half-space acoustic wave problems with FMM. An analysis of 3D Stokes flow by the multipole BEM is presented by Frangi and Gioia [31]. Nishimura and Liu [32] used FMM for thermal analysis of carbon-nanotube composites and Liu et al. [33] employed FMM to accelerate the BEM solution of the BIE for the Analysis of fiber-reinforced composites based on a rigid-inclusion model. The outcomes of these studies show great promises in dealing with large-scale engineering problems by boundary approach.

Modeling crack propagation with BEM

The advantages of the BEM are exploited in many works to simulate crack propagation. Ooi and Yang [34] used a scaled boundary finite element [35] coupled with finite element to model multiple cohesive crack propagation. Romlay et al., 2010 [36] study the modeling of fatigue crack propagation on a multiple crack site of a finite plate using BEM and the probabilistic approach of the Gaussian Monte Carlo method. Cordeiro and Leonel, 2016 [37] present the Tangent Operator Technique coupled to algebraic BEM equations to model the crack propagation process in anisotropic quasi-brittle bodies, using wood as a particular case. Nguyen et al., 2016 [38] and Nguyen et al., 2017 [39] used SGBEM by exploiting the advantages of isogeometric analysis. Recently, Sun et al. [40] use isogeometric analysis with non-uniform rational B-splines(NURBS) to analyze the crack propagation and the interaction between inclusions and cracks in 2D infinite isotropic medium.

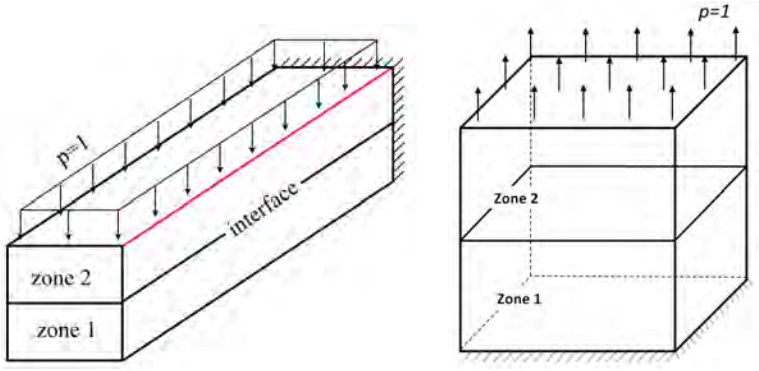
The Galerkin approach (SGBEM) is also used in many works for crack problems, see for example Frangi, 2002 [41] for a simple fatigue crack growth simulation, Roberts et al. [42] and Kitey et al. [43] for crack growth in particulate composites, Xu et al., 2004 [44] for 2D crack propagation, Tavara, 2011 [45] for cohesive crack growth in homogeneous media. In many works, the fast multipole method is used to accelerate the BEM. For crack propagation problems, Wang [46] applied it to the BEM for large-scale crack analysis in linear elastic fracture mechanics in 2D solids. Liu [47] also model multiple crack propagation in 2-D elastic solids by coupling BEM with the fast multipole method.

1.4 Initial code: MBEMv2.0

In our laboratory, a numerical code (*MBEMv2.0*) is developed based on symmetric Galerkin boundary element method coupled with the fast multipole method for crack problems simulation, see Trinh et al. [48,49]. The previous version of this code

(*MBEMv1.0*) was developed by Pham et al. [50, 51]. This Fortran code inherits a number of innovative algorithms from the BE community such as (i) the singular integration schemes by Andrä and Schnack [52, 53], (ii) the index of severity [54], (iii) the nested Flexible GMRES which makes use of the near interaction matrix [55]; and (iv) the extension of the BIEs to multizone configurations [56]. Subroutines of matrix-vector operations are taken from BLAS library. Flexible GMRES and GMRES scripts are downloaded from www.cerfacs.fr.

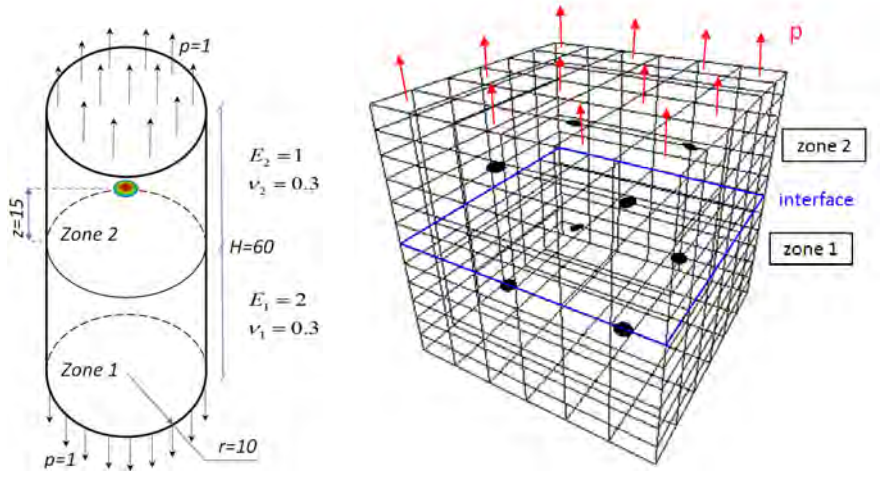
In *MBEMv2.0*, Trinh [48] implemented the SGBEM formulation for multizone problems presented by Gray and Paulino in [56]. The code can then simulate piece-wise homogeneous domains (Fig. 1.4) which can contain cracks (Fig. 1.5). For these examples, the preconditioned flexible GMRES solver used in *MBEMv2.0* gives accurate results largely disccused in [48].



(a) (b)

Figure 1.4 – Problems solved with *MBEMv2.0*: (a) Bi-material cantilever beam (b) Bi-material clamped cube

However, the code encounters several issues for complex structures like cracked pavements simulation. The calculations could not converge even when the pavement contains only one stationary crack. The crack propagation in which we are very interested was also introduced in *MBEMv2.0* but the treatment is very poor. It simply consists of resuming the elastostatic calculations. As a result, the computing cost increases at each propagation step and the total computing time is too long even for very simple problems. For example, let us consider the propagation of eight penny-shaped cracks in a clamped cube, see Fig. 1.5.b. The mesh density varies from 4 902 unknowns (initial state) to 16 038 unknowns (last increment). The computation time for each cycle changes from 4 minutes (initial state) to 22 minutes (last increment). The GMRES solver converges in 11 iterations for the initial state and in 17 iterations for the last increment. The 10 propagation increments take 2 hours. The problem is accentuated when the number of cracks increases. There is thus an obvious problem during crack propagation. Furthermore, *MBEMv2.0* has several limitations. Important engineering problems like surface breaking cracks can not be simulated. The re-meshing algorithm for crack propagation does not



(a) (b)

Figure 1.5 – Problems solved with *MBEMv2.0*: (a) Cracked bi-material cylinder (b) Multicracked bi-material clamped cube

follow any law. Problems involving non-linearities can not be simulated as they constitute one of the drawbacks of the BEM.

We can conclude that the results of the existing code are encouraging and *MBEMv2.0* deserves to be improved especially for crack propagation simulation.

1.5 Aims and outline of the thesis

The main goals of our work concerns, the development of a new version of the code noted *MBEMv3.0*. *MBEMv3.0* must be capable of stimulating three-dimensional crack propagation under mixed-mode loading in acceptable time. Complex crack configurations such as the propagation of surface breaking cracks must be taken into account. *MBEMv3.0* must be capable of stimulating large scale problems and to simulate practical civil engineering structures like cracked pavement reinforced by fiberglass grids. For this purpose, the scientific objectives to be achieved can be divided in two types:

- Reduction of computation time: In order to solve 3D problems of engineering interest, the first unavoidable challenge is the computational efficiency. The initial code thus needs to be accelerated. This challenge is considerable because several fast techniques are already used in the initial FM-SGBEM code. In this work, important efforts have been made to find additional techniques to accelerate the existing code.

- Extension works: The first challenge here is taking control of an already complex code written in Fortran and using optimized routines such as Flexible

GMRES. The second is to extend this code to take into account additional complex features such as surface breaking cracks and FEM-BEM coupling.

This thesis is divided into seven chapters whose first presents the general introduction. In *chapter 2*, the basic mathematical framework of the elasticity and fracture mechanics are briefly summarized. The symmetric Galerkin formulations for each problem are given. Then some aspects of the numerical solution by the boundary element method (discretization, evaluation of integrals, use of an iterative solver, etc.) are presented. We discuss afterward the need for a fast algorithm for the SGBEM. The principle of the fast multipole method is described and the formulation of the FMM is introduced. The computational scheme of the multi-level FMM is also illustrated. The numerical aspects of the program as well as a generic fast multipole SGBEM (FM-SGBEM) algorithm are finally described.

In *chapter 3*, some techniques to enhance the performance of the FM-SGBEM algorithm have been reported. The first issue when dealing with crack propagation problems resides in the constant growth of the storage of the near-interaction matrix. A data re-using technique has been proposed to solve this issue, the cost of re-constructing the coefficients matrix is greatly reduced especially while dealing with a few number of cracks. Secondly, a parallel implementation is achieved for identified time-consuming parts. Although the code is not entirely parallelized, the results of the parallelization are very good. Then a new sparse matrix method is designed based on coordinate format and compressed sparse row format to limit the memory required during the matrix construction phase. The remarkable performance of *MBEMv3.0* is shown through many simulations including large-scale problems. Furthermore, the existing multizone algorithm is improved to take into account general multizone problems.

Chapter 4 presents the implementation of a crack propagation criterion and the simulation of surface breaking cracks. The algorithm proposed for the propagation criterion is flexible such that the criterion can be changed easily. Multiple node treatment is then presented for the simulation of surface breaking cracks. Then an automatic multiple node algorithm is proposed to simulate the propagation of surface breaking cracks. The method is also extended to simulate the propagation of interface breaking cracks.

In *chapter 5*, a direct strategy is presented to couple the FM-SGBEM to the FEM in order to simulate the fiberglass grids. Firstly, membrane finite elements are implemented in plane stress and validated by comparison to CAST3M results for simple problems. An algorithm is then proposed to take into account the finite element matrix when solving the FM-SGBEM equations.

Chapter 6 presents the use of *MBEMv3.0* in practical civil engineering problems: multi-layer road structures. Real loading and material properties have been assigned for the models. The various problems encountered by *MBEMv2.0* for these problems are solved and multiple crack propagation is studied. Then the effects of fiberglass grid reinforcements are studied by using the coupled code proposed.

Lastly, some concluding remarks and discussions have been presented in *Chapter*

7. The perspectives as well as the directions for future research have also been introduced.

The thesis also contains three *Annexes* which present the details, descriptions, formulations and complementary techniques related to the method FM-SGBEM as well as the numerical program. Annex A presents the integral equations and regularization for fracture mechanics. Annex B shows the details of the fast multipole formulation when applied to the SGBEM. Annex C presents a brief user guide for the developed code.

Chapter 2

Fast Multipole Symmetric Galerkin BEM

Contents

In this chapter, the boundary integral equations and the symmetric Galerkin formulation of elasticity and fracture mechanics problems are presented. Some important aspects of the numerical solution including the use of an iterative solver are discussed. These aspects lead to the limitations of the BEM which motivate the use of the fast multipole method. A generic computational scheme of the FM-SGBEM is then described.

2.1 Galerkin formulation for elastostatic problems

2.1.1 Differential formulation

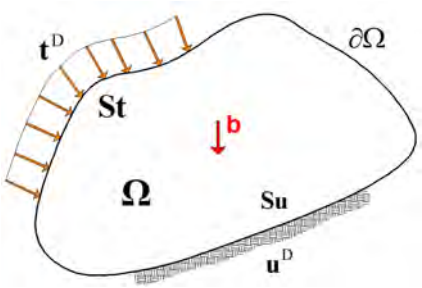


Figure 2.1 – Elastic solid

Let us consider a 3D elastic deformable body Ω (Fig.2.1), either bounded or unbounded, subjected to body force b_i, imposed boundary conditions of traction $t_i = t_i^d$ on surface S_t and prescribed displacement $u_i = u_i^d$ on surface S_u. The stress state at a point y inside Ω is described by the stress tensor σ_{ij}, while the tractions relevant to a direction n (being the outward unit normal vector to $\partial\Omega$) are given by:

$$t_i(\mathbf{y}) = \sigma_{ij}(\mathbf{y})n_j(\mathbf{y}) \qquad (i,j = 1,2,3) \tag{2.1}$$

If one considers an infinitesimal rectangular parallelepiped surrounding y, it readily follows that static equilibrium of forces and moments requires satisfaction of equation 2.2 called the Navier's equation in terms of stresses:

$$\sigma_{ij,j}(\mathbf{y}) + b_i(\mathbf{y}) = 0 \qquad (\mathbf{y} \in \Omega) \tag{2.2}$$

where $(.)_{,j}$ stands for the derivative of $(.)$ along the j^{th} direction. In the absence of body forces, the equilibrium equation can be written as:

$$\sigma_{ij,j}(\mathbf{y}) = 0 \qquad (\mathbf{y} \in \Omega) \tag{2.3}$$

If the displacements are such that their first derivatives are so small that the square and product of the partial derivatives of u_i are negligible, then the strains described by tensor ε_{ij}, are related to displacement by the Cauchy infinitesimal strain tensor:

$$\varepsilon_{ij}(\mathbf{y}) = \frac{1}{2}[u_{i,j}(\mathbf{y}) + u_{j,i}(\mathbf{y})] \qquad (\mathbf{y} \in \Omega) \tag{2.4}$$

For an elastic and isotropic material in which there is no change in temperature, Hooke's law relating stresses and strains can be written as:

$$\sigma_{ij}(\mathbf{y}) = \lambda\delta_{ij}\varepsilon_{kk}(\mathbf{y}) + 2\mu\varepsilon_{ij}(\mathbf{y}) \tag{2.5}$$

where λ and μ are the Lamé constant and δ_{ij} is the Kronecker symbol. Equation above can also be expressed as:

$$\sigma_{ij}(\mathbf{y}) = C_{ijhk}\varepsilon_{hk}(\mathbf{y}) \tag{2.6}$$

where the elastic coefficients of the fourth-order tensor C_{ijhk} are given by:

$$C_{ijhk} = \lambda\delta_{ij}\delta hk + \mu(\delta_{ik}\delta jh + \delta_{ih}\delta jk) \tag{2.7}$$

This tensor can also be represented in terms of the Poisson ratio ν and shear modulus μ:

$$C_{ijhk} = 2\mu[\frac{\nu}{1-2\nu}\delta_{ij}\delta hk + \delta_{ik}\delta jh + \delta_{ih}\delta jk] \tag{2.8}$$

Therefore, the equilibrium equation can be written as:

$$\mu u_{i,jj}(\mathbf{y}) + (\lambda+\mu)u_{j,ji}(\mathbf{y}) = 0 \qquad (\mathbf{y} \in \Omega) \tag{2.9}$$

Supplementing the above equation with the imposed conditions on the boundary, these Navier's equations represent the differential formulation for the elastostatic problem.

2.1.2 Maxwell-Betti reciprocal theorem

In order to write the integral formulation, let us introduce Maxwell-Betti reciprocal theorem. Two sets of stresses, body forces and boundary tractions σ^1_{ij}, b^1_i, t^1_i and σ^2_{ij}, b^2_i, t^2_i are said to be *statically admissible* if equations (2.1) and (2.3) are satisfied. Two sets of displacements and strains u^1_i, ε^1_{ij} and u^2_i, ε^2_{ij} are said to be *kinematically admissible* if equations (2.4) hold. According to the *principle of virtual work*, for any *statically admissible* and any *kinematically admissible* set of quantities, the following integral statement can be written for two different states:

$$\begin{aligned}\int_\Omega \sigma^1_{ij}\varepsilon^2_{ij}dV &= \int_{\partial\Omega} t^1_i u^2_i dS + \int_\Omega b^1_i u^2_i dV \\ \int_\Omega \sigma^2_{ij}\varepsilon^1_{ij}dV &= \int_{\partial\Omega} t^2_i u^1_i dS + \int_\Omega b^2_i u^1_i dV\end{aligned} \tag{2.10}$$

Betti's reciprocal theorem can then be obtained:

$$\int_\Omega (b^2_i u^1_i - b^1_i u^2_i)dV = \int_{\partial\Omega}(t^1_i u^2_i - t^2_i u^1_i)dS \tag{2.11}$$

2.1.3 Fundamental solution

In order to use the reciprocal theorem for a real elastic state (u^1, t^1, b^1), a known state (u^2, t^2, b^2) is required. (u^2, t^2, b^2) can be chosen to represent the response of the infinite domain Ω_∞ to a concentrated force acting at point $\mathbf{x}$:

$$\begin{aligned} b_i^2 &= \delta(\mathbf{x}, \mathbf{y}) e_i^l \\ u_i^2 &= U_i^k(\mathbf{x}, \mathbf{y}) e_k^l \\ t_i^2 &= \Sigma_{ij}^k(\mathbf{x}, \mathbf{y}) n_j(\mathbf{y}) e_k^l \end{aligned}$$

where $\delta(\mathbf{x}, \mathbf{y})$ is the Dirac delta function and e_i^l represents the unit vector in the l^{th} direction of the force.

The solution of Navier's equation for this unbounded elastic space is the so-called Kelvin fundamental solution which is at the basis of the integral formulations of the elastostatic problem. Expression of the fundamental solutions U_i^k and Σ_{ij}^k are given:

$$U_i^k(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{1}{8\pi\mu r}\Big[\frac{3-4\nu}{2(1-\nu)} - \frac{1}{2(1-\nu)} r_{,i} r_{,k}\Big] \tag{2.12}$$

$$\Sigma_{ij}^k(\mathbf{x}, \tilde{\mathbf{x}}) = -\frac{1}{8\pi(1-\nu)r^2}\Big[3r_{,i}r_{,k}r_{,j} + (1-2\nu)(\delta_{ik}r_{,j} + \delta_{jk}r_{,i} - \delta_{ij}r_{,k})\Big] \tag{2.13}$$

2.1.4 Integral representation

The integral equation for the elastostatic problem can then be derived from Betti's reciprocal theorem: letting (u^1, t^1, b^1) denote the real elastic state of the body Ω, while (u^2, t^2, b^2) represents the fundamental solution. The Somigliana integral equation for displacements can be obtained by introducing equation (2.12) into equation (2.11):

$$\begin{aligned} u_k(\mathbf{x}) = &-\int_{\partial\Omega} u_i(\mathbf{y}) n_j(\mathbf{y}) \Sigma_{ij}^k(\mathbf{x}, \mathbf{y}) dS_y \\ &+ \int_{\partial\Omega} t_i(\mathbf{y}) U_i^k(\mathbf{x}, \mathbf{y}) dS_y + \int_\Omega b_i(\mathbf{y}) U_i^k(\mathbf{x}, \mathbf{y}) dV \end{aligned} \tag{2.14}$$

apply Hooke's law for this equation, one gets:

$$\begin{aligned} \sigma_{ij}(\mathbf{x}) = &C_{klab} \int_{\partial\Omega} u_k(\mathbf{y}) n_j(\mathbf{y}) \frac{\partial}{\partial y_b} \Sigma_{ij}^a(\mathbf{y}, \mathbf{x}) dS_y \\ &- \int_{\partial\Omega} t_k(\mathbf{y}) \Sigma_{ij}^k(\mathbf{y}, \mathbf{x}) dS_y + \int_\Omega b_i(\mathbf{y}) \Sigma_{ij}^k(\mathbf{y}, \mathbf{x}) dV \end{aligned} \tag{2.15}$$

The above equations are called the integral representations which permit the calculation of the displacement and stresses at any point $\mathbf{x}$ interior to domain Ω

when the displacement and traction fields are known over the whole boundary $\partial\Omega$. These equations become invalid when the point $\mathbf{x} \in \partial\Omega$.

In order to obtain an equation which involves only the boundary quantities, the source point $\mathbf{x}$ has to be moved to the boundary $\partial\Omega$ and a limit process is performed (details of the procedure can be found in [57]). This procedure results in a boundary integral equation for displacements:

$$\int_{\partial\Omega} \{[u_i(\mathbf{y}) - u_i(\mathbf{x})]T_i^k(\mathbf{x},\mathbf{y}) - t_i(\mathbf{y})U_i^k(\mathbf{x},\mathbf{y})\}dS_y = \int_{\Omega} \rho b_i(\mathbf{y})U_i^k(\mathbf{x},\mathbf{y})dV \quad (2.16)$$

In an analogous manner, we get the boundary integral equation for tensions. These equations form the basis of the subsequent discretization progress which gives rise to the *collocations* approach.

2.1.5 Symmetric Galerkin formulation

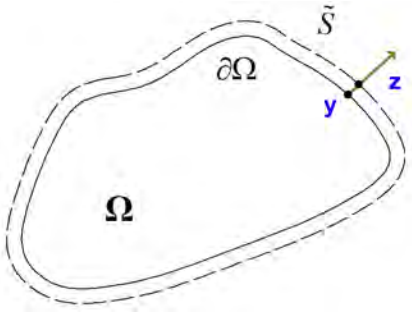


Figure 2.2 – Boundary $\partial\Omega$ and auxiliary surface $\tilde{S}$

Unlike the *collocations* approach, the Symmetric Galerkin BEM is based on a variational (weak) version of the integral equations. It provides a symmetric and sign-definite coefficient matrix through the evaluation of double integrations. Over many decades, the SGBEM has been the subject of many extensive researches. The interested readers are referred to [57] for more details about the method. Here, only a simple description of SGBEM for elastostatic problem is introduced:

Let $\tilde{S}$ be a closed, regular surface near the boundary $\partial\Omega$ and defined by means of a one-to-one mapping F onto $\partial\Omega$ (Fig.2.2):

$$\mathbf{y} \in \Omega \rightarrow \mathbf{z} = \mathbf{F}(\mathbf{y}) \in \tilde{S} \quad (2.17)$$

As the image of the boundary $\partial\Omega$, the surface $\tilde{S}$ also consists of two portions $\tilde{S}_u$ and $\tilde{S}_t$. The idea is to perform some analytic manipulation, for regularization purposes, on the double surface integrals over $\partial\Omega x\tilde{S}$ and then consider the limit process $\tilde{S} \rightarrow \partial\Omega$. Following the approach introduced by Sitori et al. [58], the SGBEM procedure consists basically of two distinct steps:

1. At first, the classical displacement and traction boundary integral equations are enforced in a weak sense on the auxiliary contours $\tilde{S}$ distinct from $\partial\Omega$. The displacement equation can be enforced on the surface $\tilde{S}_u$ in a weighted sense using a test function $\tilde{\mathbf{t}}(\tilde{\mathbf{x}})$. In the same manner, the traction equation can also be written on $\tilde{S}_t$ with the test function $\tilde{\mathbf{u}}(\tilde{\mathbf{x}})$. An analytic regularization procedure is carried out via integration by parts and Stokes theorem.

2. Secondly, the limit $\tilde{S} \to \partial\Omega$ is taken and the discretization phase is performed. The detailed symmetric Galerkin equations governing the unknown boundary traces $\mathbf{u}$ on S_t and $\mathbf{t}$ on S_u for a mixed boundary value problem in elastostatic (see [58–60]) is shown below:

$$\text{Find } (\mathbf{u},\mathbf{t}) \in \mathcal{V}_u \times \mathcal{V}_t, \begin{cases} \mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}}) + \mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}}) = \mathbf{F}_u(\tilde{\mathbf{u}}) \\ \mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}}) + \mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \mathbf{F}_t(\tilde{\mathbf{t}}) \end{cases} \quad \forall (\tilde{\mathbf{u}},\tilde{\mathbf{t}}) \in \mathcal{V}_u \times \mathcal{V}_t \tag{2.18}$$

using the bi-linear forms:

$$\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}}) = \int_{S_t}\int_{S_t} (Ru)_{iq}(\mathbf{x}) B_{ikqs}(\mathbf{r}) (R\tilde{u})_{ks}(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.19}$$

$$\mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}}) = -\int_{S_u}\int_{S_t} \mathbf{t}_k(\mathbf{x}) T_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{\mathbf{u}}_i(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.20}$$

$$\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}}) = -\int_{S_u}\int_{S_t} \tilde{\mathbf{t}}_k(\tilde{\mathbf{x}}) T_i^k(\tilde{\mathbf{x}},\mathbf{x}) \mathbf{u}_i(\mathbf{x}) dS_x dS_{\tilde{x}} \tag{2.21}$$

$$\mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \int_{S_u}\int_{S_u} \mathbf{t}_k(\mathbf{x}) U_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{\mathbf{t}}_i(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.22}$$

and the linear forms:

$$\begin{aligned} \mathbf{F}_u(\tilde{\mathbf{u}}) &= (\kappa-1)\int_{S_t} t_k^D(\mathbf{x}) \tilde{u}_k(\mathbf{x}) dS_x + \int_{S_t}\int_{S_t} t_k^D(\mathbf{x}) T_i^k(\mathbf{x},\tilde{\mathbf{x}}) [\tilde{u}_i(\tilde{\mathbf{x}}) - \tilde{u}_i(\mathbf{x})] dS_{\tilde{\mathbf{x}}} dS_x \\ &\quad - \int_{S_t}\int_{S_u} t_k^D(\mathbf{x}) T_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{u}_i(\mathbf{x}) dS_{\tilde{\mathbf{x}}} dS_x - \int_{S_u}\int_{S_t} (Ru)_{iq}^D(\mathbf{x}) B_{ikqs}(\mathbf{r}) (R\tilde{u})_{ks}(\tilde{\mathbf{x}}) dS_{\tilde{\mathbf{x}}} dS_x \\ \mathbf{F}_t(\tilde{\mathbf{t}}) &= \kappa \int_{S_u} u_k^D(\mathbf{x}) \tilde{t}_k(\mathbf{x}) dS_x + \int_{S_u}\int_{S_u} [\tilde{u}_i^D(\mathbf{x}) - \tilde{u}_i^D(\tilde{\mathbf{x}})] T_i^k(\tilde{\mathbf{x}},\mathbf{x}) \tilde{t}_k(\tilde{\mathbf{x}}) dS_x dS_{\tilde{\mathbf{x}}} \\ &\quad - \int_{S_t}\int_{S_u} t_k^D(\mathbf{x}) U_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{t}_i(\tilde{\mathbf{x}}) dS_{\tilde{\mathbf{x}}} dS_x - \int_{S_u}\int_{S_t} u_i^D(\tilde{\mathbf{x}}) T_i^k(\tilde{\mathbf{x}},\mathbf{x}) \tilde{t}_k(\tilde{\mathbf{x}}) dS_x dS_{\tilde{\mathbf{x}}} \end{aligned} \tag{2.23}$$

The coefficient $\kappa = 0$ or 1 depends on whether the unit normal to S_u or S_t is directed toward the exterior or interior of that surface. U_i^k and T_i^k denote respectively the components in the direction i of the Kelvin fundamental displacement and traction at $\mathbf{x} \in \mathbf{R}^3$ created in an elastic full-space by a point force applied at $\tilde{\mathbf{x}} \in \mathbf{R}^3$ and are written as:

$$U_i^k(\mathbf{x},\tilde{\mathbf{x}}) = \frac{1}{8\pi\mu r}\Big[\frac{3-4\nu}{2(1-\nu)} - \frac{1}{2(1-\nu)}r_{,i}r_{,k}\Big] \tag{2.24}$$

$$T_i^k(\mathbf{x},\tilde{\mathbf{x}}) = -\frac{1}{8\pi(1-\nu)r^2}\Big[3r_{,i}r_{,k}r_{,j} + (1-2\nu)(\delta_{ik}r_{,j} + \delta_{jk}r_{,i} - \delta_{ij}r_{,k})\Big]n_j(\mathbf{y}) \tag{2.25}$$

having set:

$$\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}} \qquad r = \| \mathbf{r} \| \qquad \widehat{\mathbf{r}} = \mathbf{r}/r \tag{2.26}$$

The space $\mathcal{V}_u$ and $\mathcal{V}_t$ of admissible boundary traces of displacements and tractions are definded as:

$$\begin{aligned}\mathcal{V}_u &= \{\mathbf{u} \in H^{1/2}(S), supp(\mathbf{u}) \subset S_t\}\\ \mathcal{V}_t &= \{\mathbf{u} \in H^{-1/2}(S), supp(\mathbf{t}) \subset \overline{S}_u\}\end{aligned} \tag{2.27}$$

and $\tilde{\mathbf{u}}$,$\tilde{\mathbf{t}}$ are test displacements and tensions. Natural finite-dimensional sub-spaces of $\mathcal{V}_u$ and $\mathcal{V}_t$ for Galerkin discretization consist of continuous interpolations of $\mathbf{u}$ over S_t with a zero trace on the edge ∂S_t and piece-wise continuous interpolation of $\mathbf{t}$ over S_u. In particular, in contrast to the case of the traction collocations BIE, the interpolation method puts no requirement on the derivatives of $\mathbf{u}$. Note that the data $\mathbf{u}^D$ appearing in (2.23) is actually an arbitrary extension to $\delta\Omega$ of the displacement value prescribed on S_u, having $\mathbf{u}^D \in H^{1/2}(\partial\Omega)$ regularity, so that the actual displacement on S_t is $\mathbf{u} + \mathbf{u}^D$. This allows $\mathbf{u}$ and $\tilde{\mathbf{u}}$ to belong to the same space $\mathcal{V}_u$.

Formulations (2.19) and (2.23) are written in regularized form [59], which involve only weakly singular double surface integrals with $O(r^{-1})$ integrands. The regularization inveles the Stokes theorem together with indirect regularization. The surface curl operator R arising as a result of this manipulation is defined [61] by $[Ru]_{ks}(\tilde{\mathbf{x}}) = e_{jrs}n_j u_{k,r}(\mathbf{y})$ (where e_{jrs} denotes the permutation symbol). The weakly singular fourth-rank tensor $B_{ikqs}(\mathbf{r})$ can be expressed as following:

$$B_{ikqs}(\mathbf{r}) = \frac{\mu}{4\pi(1-\nu)}\Big[\Big(\delta_{qs}\delta_{ik} - 2\delta_{is}\delta_{kq}\nu - (1-\nu)\delta_{iq}\delta_{ks}\Big)r^{-1} + \delta_{qs}r_{,i}r_{,k}\Big] \tag{2.28}$$

2.2 Galerkin formulation for fracture mechanics

In this work, the fracture equations are obtained by assuming the validity of elastic linear fracture mechanics. Quasi-brittle materials are considered. Plasticity is not taken into account. Considering a fractured solid Ω subjected to prescribed tractions t^D on the boundary S_t and displacement constraints u^D on Su. The boundary of Ω (including the crack S_c) is thus defined as $S = S_t \bigcup S_u \bigcup S_c$. S_c is conceived as a locus of displacement discontinuity, the jump of the displacements

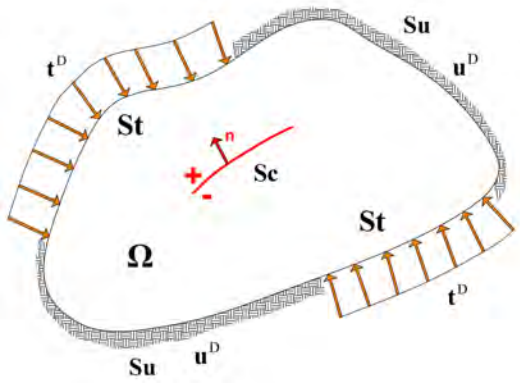


Figure 2.3 – Solid containing a crack

can be computed as $\boldsymbol{\Delta u}(x) = \boldsymbol{u}(x^+) - \boldsymbol{u}(x^-)$ where $\boldsymbol{u}(x^+)$ and $\boldsymbol{u}(x^-)$ are respectively the displacement of the upper and lower faces of the crack ($S_c = S_c^- \bigcup S_c^+$). The direction of the normal of the crack is by convention, pointing from S^- to S^+. Introducing now a fictitious surface $\tilde{S}$ interior to $\partial\Omega$. Assuming the existence of a one-on-one correspondence between points $\mathbf{x} \in S$ and $\tilde{\mathbf{x}} \in \tilde{S} : \tilde{\mathbf{x}} = \mathcal{X}(\mathbf{x}, h)$. The two surfaces coincide as the parameter $h = 0$. We also take into account the crack surfaces $\tilde{S_c^+}, \tilde{S_c^-}$ and their correspondences. The SGBEM procedure consists of two steps: at first, the classical displacement and traction boundary integral equations are written in a weak form on the auxiliary contours $\tilde{S}$ and $\tilde{S}_c$ distinct from S and S_c (i.e. with $h \neq 0$) and an analytical regularization procedure is carried out by integration by parts and Stoke theorem. Secondly, the limits $\tilde{S} \to S$, $\tilde{S}_c \to S_c (h \to 0)$ are taken and the discretization procedure is performed. The definition of an auxiliary surface $\tilde{S} \cup \tilde{S}_c$ is hence only an artifice which proves useful to guarantee a firm mathematical and computational basis in dealing with the double surface integrals involved in the formulation; however, $\tilde{S}$ and $\tilde{S}_c$ do not play any role in the final implementation of the method. The boundary integral formulation for this problem is written as follows (see details in [57]):

$$
\begin{aligned}
&\text{Find } (\mathbf{u}, \mathbf{t}, \Delta\mathbf{u}) \in \mathcal{V}_u \times \mathcal{V}_t \times \mathcal{V}_c, \\
&\begin{cases}
\mathbf{B}_{uu}(\mathbf{u}, \tilde{\mathbf{u}}) + \mathbf{B}_{tu}(\mathbf{t}, \tilde{\mathbf{u}}) + \mathbf{B}_{\Delta uu}(\boldsymbol{\Delta}\mathbf{u}, \tilde{\mathbf{u}}) = \mathbf{F}_u(\tilde{\mathbf{u}}) \\
\mathbf{B}_{ut}(\mathbf{u}, \tilde{\mathbf{t}}) + \mathbf{B}_{tt}(\mathbf{t}, \tilde{\mathbf{t}}) + \mathbf{B}_{\Delta ut}(\boldsymbol{\Delta}\mathbf{u}, \tilde{\mathbf{t}}) = \mathbf{F}_t(\tilde{\mathbf{t}}) \\
\mathbf{B}_{u\Delta u}(\mathbf{u}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) + \mathbf{B}_{t\Delta u}(\mathbf{t}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) + \mathbf{B}_{\Delta u\Delta u}(\boldsymbol{\Delta}\mathbf{u}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) = \mathbf{F}_{\Delta u}(\boldsymbol{\Delta}\tilde{\mathbf{u}}) \\
\qquad \forall (\tilde{\mathbf{u}}, \tilde{\mathbf{t}}, \Delta\tilde{\mathbf{u}}) \in \mathcal{V}_u \times \mathcal{V}_t \times \mathcal{V}_c
\end{cases}
\end{aligned}
\tag{2.29}
$$

using the bilnear forms introduced in 2.19 and the following:

$$\mathbf{B}_{\Delta uu}(\boldsymbol{\Delta}\mathbf{u},\tilde{\mathbf{u}}) = -\int_{S_c}\int_{S_t}(R\Delta u)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{2.30}$$

$$\mathbf{B}_{u\Delta u}(\mathbf{u},\boldsymbol{\Delta}\tilde{\mathbf{u}}) = -\int_{S_t}\int_{S_c}(Ru)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\Delta\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{2.31}$$

$$\mathbf{B}_{t\Delta u}(\mathbf{t},\boldsymbol{\Delta}\tilde{\mathbf{u}}) = \int_{S_u}\int_{S_c}\mathbf{t}_k(\mathbf{x})T_i^k(\mathbf{x},\tilde{\mathbf{x}})\boldsymbol{\Delta}\tilde{\mathbf{u}}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{2.32}$$

$$\mathbf{B}_{\Delta ut}(\boldsymbol{\Delta}\mathbf{u},\tilde{\mathbf{t}}) = \int_{S_u}\int_{S_c}\tilde{\mathbf{t}}_k(\tilde{\mathbf{x}})T_i^k(\tilde{\mathbf{x}},\mathbf{x})\boldsymbol{\Delta}\mathbf{u}_i(\mathbf{x})dS_xdS_{\tilde{x}} \tag{2.33}$$

$$\mathbf{B}_{\Delta u\Delta u}(\boldsymbol{\Delta}\mathbf{u},\boldsymbol{\Delta}\tilde{\mathbf{u}}) = \int_{S_c}\int_{S_c}(R\Delta u)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\Delta\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{2.34}$$

the linear form:

$$\begin{aligned}\mathbf{F}_{\Delta u}(\tilde{\mathbf{u}}) = &\int_{S_c}p_k(\mathbf{x})\Delta\tilde{u}_k(\mathbf{x})dS_x + \int_{S_u}\int_{S_c}(Ru^D)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\Delta\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x\\ &-\int_{S_t}\int_{S_c}t_k^D(\mathbf{x})T_i^k(\mathbf{x},\tilde{\mathbf{x}})\Delta\tilde{u}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x\end{aligned} \tag{2.35}$$

In (2.29), the spaces of admissible boundary traces are $\mathcal{V}_u$ and $\mathcal{V}_t$ (defined by (2.27)), and $\mathcal{V}_c = H_0^{1/2}(S_c)$. Natural finite-dimensional sub-spaces of $\mathcal{V}_c$ for Galerkin discretization then consist of continuous interpolations of $\boldsymbol{\Delta}\mathbf{u}$ over S_c with a zero trace on the crack front ∂S_c, with again no requirement on the derivatives of $\boldsymbol{\Delta}\mathbf{u}$.

2.3 Boundary element analysis

After defining the boundary integral equations, the numerical solution of the system is now considered. Analytic solutions of the integral equations are no easier to obtain than for the original differential equation, and thus it is necessary to reduce the continuous equations to a discrete system of linear equations that can be solved numerically. In this section, some basic steps in a boundary analysis such as geometries discretization, integral evaluation and system solution are briefly summarized.

2.3.1 Discretization

The geometry discretization in the BEM is based on a partitioning of the boundary surface $\partial\Omega$ in to N_e non-intersecting boundary elements $E_1, E_2, .., E_N$

$$\partial\Omega \simeq \bigcup_{e=1}^{N_e} S_e \tag{2.36}$$

One of the most convenient ways of having the necessary approximations is using the *isoparametric* method, in which the boundary and boundary functions

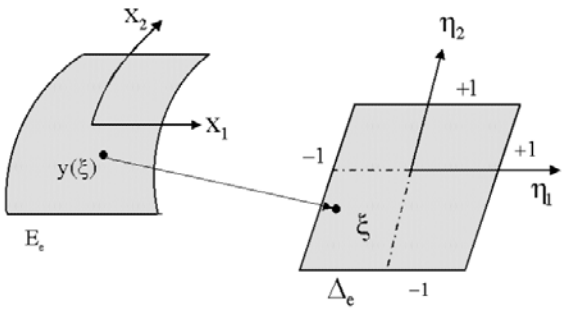


Figure 2.4 – Boundary element E_e and referent element Δ_e

are represented through the same set of shape functions defined on a parameters space. The discretization procedure can be briefly summarized as follows:

A generic field $\mathbf{f}$ (i.e. geometry, displacements or tensions) can be approximated in the point $\mathbf{x}$ over the elements E_e by:

$$f_j(\mathbf{y}) = \sum_{\alpha=1}^{N_n} \Phi_\alpha^e(\xi) f_j^{e\alpha} \tag{2.37}$$

where N_n is the number of nodes of the element, $\Phi_\alpha^e(\xi)$ is the shape function and $f_j^{e\alpha}$ are the nodal values belonging to element E_e. The vector $\mathbf{y}$ gathers the spatial coordinates of a point inside the element in the global reference system, while in ξ the local coordinates (with respect to the master element) are collected.

The use of linear elements is recommended even in large scale problems or in curve boundaries because the numerical performance can be greatly sped up. In case a very high accuracy is required (eg. behavior near the crack tip), quadratic elements are employed. Therefore, to optimize our code's functioning, in a generic fracture problem, the outer geometries (if they exist) are usually meshed with Q4-elements while the cracks are all modeled by Q8-elements.

2.3.2 Numerical evaluation of stress intensity factors

Before applying the discretization to the boundary equations and solving the obtained system, let us first see how to evaluate the stress intensity factors on the discrete geometry. For crack problems, the surface S and S_c are divided respectively into N_e and N_c boundary elements. For each element, the displacement $\boldsymbol{u}$ and the traction $\boldsymbol{t}$ on the regular surface are interpolated via (2.37). The displacement discontinuities $\Delta\boldsymbol{u}$ on the surface of the cracks will be interpolated with the usual shaped functions:

$$\Delta\boldsymbol{u}(\boldsymbol{y}) = \sum_{m=1}^{N_I} N_m(\boldsymbol{y}) \Delta\boldsymbol{u}^m \tag{2.38}$$

where $\Delta\boldsymbol{u}^m$ are the nodal values of the approximations of $\Delta\boldsymbol{u}$ on the element S_e and $N_m(\boldsymbol{\xi})$ are the interpolation functions.

First of all, it is evident that the fields of elastic deformation and stress are singular in proximity of the crack surface. The planes perpendicular to the crack front intersect the crack surface along lines, which will be called $\boldsymbol{s}$ -lines henceforth. Lines on S_c perpendicular to $\boldsymbol{s}$-lines are called $\boldsymbol{t}$-lines. Obviously the crack edge

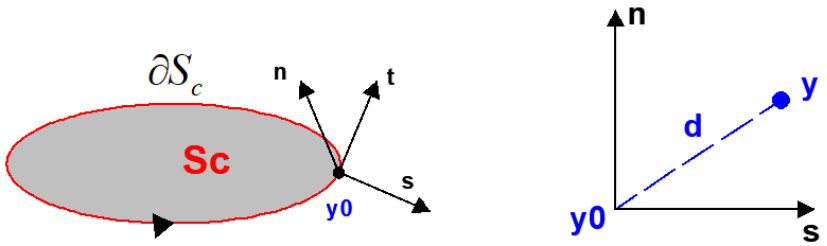


Figure 2.5 – Location of a point $\boldsymbol{y}$ in proximity of the crack surface

is a $\boldsymbol{t}$-line. The SIFs are evaluated through extrapolation from the displacement discontinuity field expressed in a local coordinate system as:

$$\Delta \boldsymbol{u} = \Delta u_n \boldsymbol{n} + \Delta u_s t_{\boldsymbol{s}} + \Delta u_t t_{\boldsymbol{t}} \tag{2.39}$$

where $t_{\boldsymbol{s}}$ and $t_{\boldsymbol{t}}$ are the local surface base unit vectors. The asymptotic expression for $\Delta \boldsymbol{u}$ components writes, denoted by $\boldsymbol{d}$ the arc-length distance from the crack front along $\boldsymbol{s}$-lines:

$$\begin{aligned}
\Delta u_n &= K_I \frac{4(1-\nu)}{\mu} [\frac{\boldsymbol{d}}{2\pi}]^{\frac{1}{2}} + O(\boldsymbol{d}) \\
\Delta u_s &= K_{II} \frac{4(1-\nu)}{\mu} [\frac{\boldsymbol{d}}{2\pi}]^{\frac{1}{2}} + O(\boldsymbol{d}) \\
\Delta u_t &= K_{III} \frac{4}{\mu} [\frac{\boldsymbol{d}}{2\pi}]^{\frac{1}{2}} + O(\boldsymbol{d})
\end{aligned}$$

These stress intensity factors can be evaluated from the opening displacement by:

$$K_I = \lim_{\boldsymbol{d}\to 0} \frac{\mu}{4(1-\nu)} [\frac{2\pi}{\boldsymbol{d}}]^{\frac{1}{2}} \Delta u_n \tag{2.40}$$

$$K_{II} = \lim_{\boldsymbol{d}\to 0} \frac{\mu}{4(1-\nu)} [\frac{2\pi}{\boldsymbol{d}}]^{\frac{1}{2}} \Delta u_s \tag{2.41}$$

$$K_{III} = \lim_{\boldsymbol{d}\to 0} \frac{\mu}{4} [\frac{2\pi}{\boldsymbol{d}}]^{\frac{1}{2}} \Delta u_t \tag{2.42}$$

Precisely, to compute the SIFs at a point on the crack front, the asymptotic expansions of the displacement discontinuities (which strictly hold only in the vicinity of the front) are used at the two nodes nearest the crack front (along the $\boldsymbol{s}$-line through the point for which the SIFs are evaluated) in order to obtain two estimates for each SIF. For determining these factors, we can apply the method based on utilization of a special element called quarter-point element.

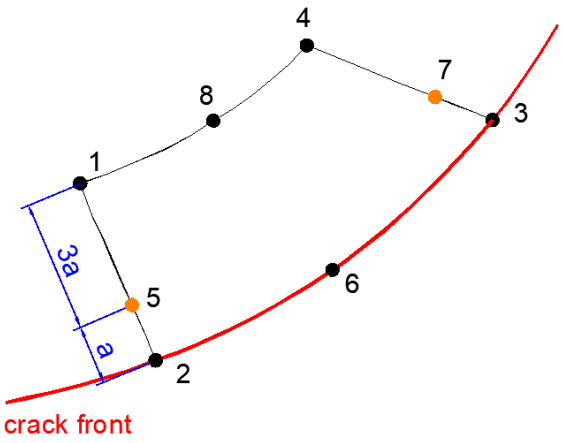


Figure 2.6 – Quarter point element

Quarter-point elements

In order to capture the behavior in the crack-front better (which presents the singularity), the elements adjacent to the front of the crack are modified. Let us consider a quadrilateral 8-node isoparametric element adjacent to the crack's front (see Fig.2.6). 2 middle nodes 5 and 7 are pushed closer to the crack front by a quarter of the element edge's length:

$$\mathbf{y}^5 - \mathbf{y}^2 = -as$$
$$\mathbf{y}^1 - \mathbf{y}^2 = -4as$$

The node $\mathbf{y}^5 \in [\mathbf{y}^2, \mathbf{y}^1]$ is at quarter of the length $\| \mathbf{y}^2 - \mathbf{y}^1 \| = 4a$. On the segment $[\mathbf{y}^2, \mathbf{y}^1]$, the interpolation is quadratic and the point $\mathbf{y}$ is interpolated by:

$$\mathbf{y} = N_2(\varepsilon)\mathbf{y}^2 + N_5(\varepsilon)\mathbf{y}^5 + N_1(\varepsilon)\mathbf{y}^1 \tag{2.43}$$

The interpolation functions are the ones of a 3-node quadratic element and are given by:

$$N_2(\varepsilon) = \frac{1}{2}\varepsilon(1-\varepsilon)$$
$$N_5(\varepsilon) = 1 - \varepsilon^2$$
$$N_1(\varepsilon) = \frac{1}{2}\varepsilon(1+\varepsilon)$$

We can write that:

$$\mathbf{y} - \mathbf{y}^2 = -(1+\varepsilon)^2 as$$

and the distance becomes:

$$\mathbf{d} = \| \mathbf{y} - \mathbf{y}^2 \| = a(1+\varepsilon)^2$$

The expression of the approximation of the displacement discontinuities $\Delta\mathbf{u}$:

$$\begin{aligned}\Delta\mathbf{u}(\mathbf{y}) &= N_5(\varepsilon)\Delta\mathbf{u}^5 + N_1(\varepsilon\Delta\mathbf{u}^1) \\ &= (1+\varepsilon)\left[\Delta\mathbf{u}^5 + (\frac{1}{2}\Delta\mathbf{u}^1 - \Delta\mathbf{u}^5)\varepsilon\right]\end{aligned}$$

In paying attention to $\Delta\mathbf{u}^2 = 0$ because the opening displacements are nulls on the crack front. And in function of $\mathbf{d}$, we can rewrite the above expression as:

$$\Delta\mathbf{u}(\mathbf{y}) = \left(\frac{\mathbf{d}}{2}\right)^{1/2}\left[2\Delta\mathbf{u}^5 - \frac{1}{2}\Delta\mathbf{u}^1 + \left(\frac{\mathbf{d}}{2}\right)^{1/2}\left(\frac{1}{2}\Delta\mathbf{u}^1 - \Delta\mathbf{u}^5\right)\right]$$

With the help of the usual interpolation functions, it is possible to represent the variation in $\frac{\mathbf{d}}{2}$ of $\Delta\mathbf{u}$ in proximity of the crack surface. These stress intensity factors K_I for example, can be evaluated from the nodal values of $\Delta\mathbf{u}^5$ and $\Delta\mathbf{u}^1$:

$$\begin{aligned}K_I^2 &= \lim_{\mathbf{d}\longrightarrow 0}\frac{\mu}{4(1-\nu)}\left(\frac{2\pi}{\mathbf{d}}\right)^{1/2}\Delta u_n \\ &= \lim_{\mathbf{d}\longrightarrow 0}\frac{\mu}{4(1-\nu)}\left(\frac{2\pi}{\mathbf{d}}\right)^{1/2}\left[2\Delta\mathbf{u}_n^5 - \frac{1}{2}\Delta\mathbf{u}_n^1 + \left(\frac{\mathbf{d}}{2}\right)^{1/2}\left(\frac{1}{2}\Delta\mathbf{u}_n^1 - \Delta\mathbf{u}_n^5\right)\right] \\ &= \frac{\mu}{4(1-\mu)}\left(\frac{2\pi}{a}\right)^{1/2}\left[2\Delta\mathbf{u}_n^5 - \frac{1}{2}\Delta\mathbf{u}_n^1\right] \qquad (2.44)\end{aligned}$$

The same procedure can be used to compute the other factors K_{II}^2 or K_{III}^2.

2.3.3 Galerkin approximation

Let us come back to the resolution of the boundary integral equations. In contrast to *collocation*, the Galerkin approach does not require that the boundary integral equations be satisfied at any point. Instead, the equations are enforced in a weighted average where the weight functions are composed of all shape functions that are non-zero on the studied node. The symmetric Galerkin formulations are written under double surface integral forms (2.30). The evaluation of the double boundary integrals represents a crucial aspect in SGBEM. The generic double surface integral equation takes the following form:

$$I(S_e, S_f) = \int_{S_e}\int_{S_f} f(\mathbf{x})K(\mathbf{x},\mathbf{y})g(\mathbf{y})dS_{\mathbf{y}}dS_{\mathbf{x}} \qquad (2.45)$$

where S_e and S_f are the surfaces of source and field elements ($\mathbf{x} \in S_e$,$\mathbf{y} \in S_f$), $f(\mathbf{x})$ and $g(\mathbf{y})$ are respectively known and test function. $K(\mathbf{x},\mathbf{y})$ is the Kernel

which contains the singularity $O(r^{-1})$ or $O(r^{-2})$. As an integration requires a pair of source and field elements, singularity will occur when these two elements are coincident, adjacent by edge or adjacent by vertex. For 3D problems, there are thus four possible configurations (Fig.2.7):

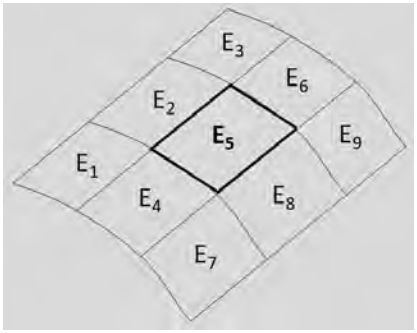


Figure 2.7 – Elements' interactions

- Regular case: source and field points belong to two distinct elements
- Coincident: two elements overlap (E_5 and itself)
- Adjacent by edge: two elements share one common edge (E_5 with E_2,E_4,E_6 and E_8)
- Adjacent by vertex: two elements share one common vertex (E_5 with E_1,E_3,E_7 and E_9)

The singular integrals are evaluated using special schemes described in [57]. The regular integrals can be evaluated with normal quadrature rule. The integral of a pair of elements can be written as:

$$I(S_e, S_f) = \int_{\Delta_e}\int_{\Delta_f} f(\mathbf{x}(\boldsymbol{\eta}))K(\mathbf{x}(\boldsymbol{\eta}),\mathbf{y}(\boldsymbol{\xi}))g(\mathbf{y}(\boldsymbol{\xi}))J_y(\boldsymbol{\xi})J_x(\boldsymbol{\eta})d\boldsymbol{\xi}d\boldsymbol{\eta} \tag{2.46}$$

where $\Delta_e \in [-1,1]\times[-1,1]$ and $\Delta_f \in [-1,1]\times[-1,1]$. This integral can be approximated by:

$$I(S_e, S_f) \simeq \sum_{i=1}^{N_{pge}}\sum_{j=1}^{N_{pgf}} f(\boldsymbol{\eta}_i)K(\boldsymbol{\eta}_i,\boldsymbol{\xi}_j)g(\boldsymbol{\xi}_j)J_y(\boldsymbol{\xi}_j)J_x(\boldsymbol{\eta}_i)A^j_{\boldsymbol{\xi}_j}A^i_{\boldsymbol{\eta}_i} \tag{2.47}$$

where $\boldsymbol{\eta}_i$ and $A^i_{\boldsymbol{\eta}_i}$ denote the abscissas and weights of the gaussian points for exterior elements; $\boldsymbol{\xi}_j$ and $A^j_{\boldsymbol{\xi}_j}$ denote the corresponding parameters for the interior elements. N_{pge} and N_{pgf} are the number of gaussian points for exterior and interior elements respectively.

2.3.4 System solution and GMRES iterative solver

The discretized equations system of the SGBEM can be written under matrix form: $[K]\{x\} = \{b\}$. $[K]$ is the coefficient matrix, the terms in $[K]$ are derived from (2.19) or (2.30). Vector $\{x\}$ regroups all the unknowns on the boundaries of the problem: $\mathbf{u}$ on S_t, $\mathbf{t}$ on S_u and $\mathbf{\Delta u}$ on S_c. The right-hand side vector $\{b\}$ contains the known values (2.23) or (2.35) of the system.

The BEM/SGBEM method usually leads to a smaller algebraic system of equations with respect to the finite element method (FEM) since only the boundary values are involved as unknowns. However, the fact that the resulting coefficients matrix $[K]$ is fully populated represents a very difficult computational task to deal with. Letting N denote the number of BEM unknowns, conventional solution methods for the SGBEM require $O(N^2/2)$ memory, $O(N^2)$ setting up computing time and $O(N^3/6)$ solution time using a direct solver. These complexities restrain the BEM/SGBEM in the treating of medium-size problems. On the contrary, the global stiffness matrix in FEM is symmetric, sparse, banded and positive definite. The FEM requires only $O(N_{FEM})$ set-up computing time and $O(N_{FEM})$ for memory, making domain methods very efficient in many scales.

In order to solve the linear equation system obtained in the BEM, iterative solvers are recognized as the primary alternative since direct methods are all computationally very expensive. Generalized Minimal RESidual (GMRES) [62] has been the most used iterative solver for boundary element calculations. A detailed description of GMRES in BEM is shown in [63]. GMRES approximates the solution by a candidate vector in a Krylov subspace with minimal residual. The Arnoldi iteration is used to find this vector. The convergence is achieved when the backward error is smaller than a predefined tolerance. Each GMRES iteration requires the product of the coefficient matrix and a candidate vector. The complexity of this task is of $O(N^2/2)$ either if $[K]$ is stored or if $[K]\{v\}$ is evaluated by means of conventional SGBEM. This is already a major improvement in comparison to direct solvers.

Nevertheless, the convergence rate of an iterative solver depends strongly on spectral properties of the matrix which eventually lead to the use of a preconditioner. In numerical analysis, a preconditioner is a matrix such that when applied to the original system, it helps decrease the *condition number* of the coefficient matrix. This technique is called preconditioning. Let us consider a simple linear system:

$$[K]\{x\} = \{b\} \tag{2.48}$$

in which $[K]$ is a generic square coefficient matrix. This system can be left-preconditioned by matrix $[P]$:

$$[P]^{-1}[K]\{x\} = [P]^{-1}\{b\} \tag{2.49}$$

Preconditioned iterative solvers generally outperform all direct solvers for large

matrices $O(N \geq 10^4)$ and have been the only option if the coefficient matrix $[K]$ is not stored explicitly but is only accessed by evaluating matrix-vector products.

Typically, there is a trade-off in the choice of $[P]$. Since the operator $[P]^{-1}$ must be executed at each step of the iterative solver, it should have a small cost (computing time) of applying the $[P]^{-1}$ operation. The cheapest preconditioner would therefore be $[P] = [I]$ since then $[P]^{-1} = [I]$ but this results back in the original equation and no improvement has been made. At the other extreme, the choice $[P] = [K]$ gives $[P]^{-1}.[K] = [I]$ which has optimal condition number of 1 and requires only one iteration for convergence. However, applying the preconditioning $[P]^{-1}$ is as difficult as solving the original system thus gaining nothing in terms of operation. Therefore, depending on different situations, one must choose $[P]$ somewhere between these two extremes in order to balance the cost of constructing and inverting matrix $[P]$ with the overall solution.

In summary, the bottlenecks of the method reside in the memory constraint and in the evaluation of matrix-vector product which is required at each step of the iterative solver. Because the coefficient matrix is full, the cost of applying these operations becomes excessive even when the problem size is relatively small ($\sim O(N^4)$). Therefore, the application of SGBEM into large-scale problems requires that the evaluation of the matrix-vector multiplication be fast and that the explicit storage of the matrix $[K]$ be avoided.

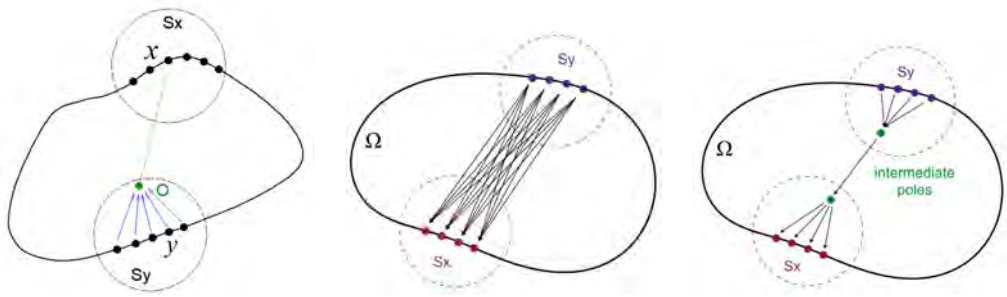
2.4 Fast multipole method

As mentioned in the previous section, the coefficient matrix of SGBEM is fully populated, so unfortunately, the build-up phase and operation count lead to a rapid exhaustion for a standard computer. This obstacle makes it impossible to apply the method to treating realistic problems which normally contain a considerable number of unknowns. The Fast multipole method (FMM) can, change this circumstance completely. First introduced by Rokhlin and Greengard [27, 64], the FMM is an alternative technique to enhance the performance of a boundary integral analysis. In a traditional boundary elements analysis, due to the presence of Kernel functions, the same calculation is repeated from one observation point to another, thus entailing a high amount of operations. In FMM, Rokhlin uses intermediate points (called poles) to represent distant particle groups and then introduces a local expansion to evaluate the distant contribution in the form of a series. The multipole moment associated with a faraway group can be translated into the coefficient of the local expansion associated with a local group. It has been proven that the FMM when combined with an iterative solver, can reduce the computational complexity of a BEM problem from $O(N^2)$ to $O(Nlog^{\alpha}N)$ (with α being a small non-negative number). This improvement has opened up a wide range of applications for the boundary analysis that have been restrained for many years due to the lack of efficiency during the solution stage. Various research fields have therefore been applied with the fast algorithm: Stokes flow [31,65], acoustics [30,66], electromagnetism [67],

elastodynamics [28, 29, 68] (read [65, 69–72] for more references). The outcomes of these studies show great promises in dealing with large-scale engineering problems by boundary approach. Some successful works have also been reported in composite materials [32, 33] and in electromagnetic wave scattering [67]. Our work inherits the developments of the FMM-SGBEM in elastostatics and fracture mechanics that are introduced in [73], [74], [51] and [49].

2.4.1 Principle of the fast multipole method

The fast multipole method is based on a reformulation of the fundamental solutions into a series of product of functions of $\mathbf{x}$ and $\mathbf{y}$. This technique allows one to re-use the integrals with respect to $\mathbf{y}$ when the observation point $\mathbf{x}$ is changed (Fig.2.8a).



(a) (b) (c)

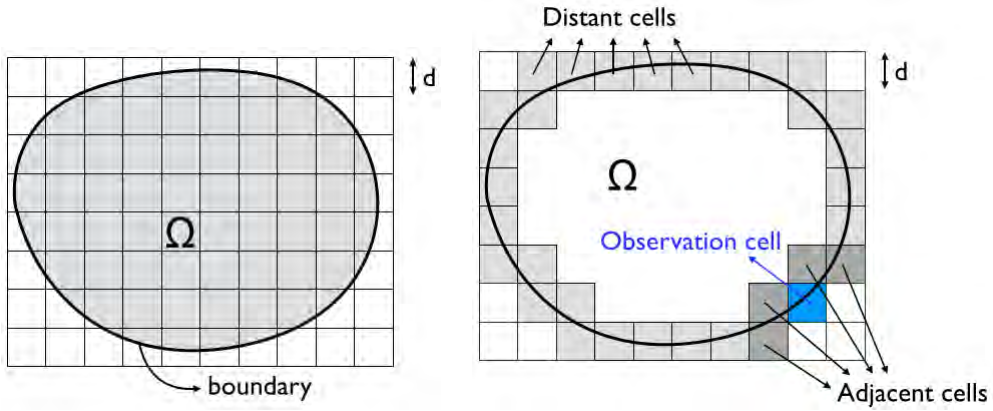
Figure 2.8 – (a) FMM simple illustration (b) Standard algorithm (c) Fast algorithm

The principle of the FMM can be illustrated as such: we need to compute the interaction between two groups of points $\mathbf{x}$ and $\mathbf{y}$ (respectively on $\boldsymbol{S_x}$ and $\boldsymbol{S_y}$). Supposing that we have n points on $\boldsymbol{S_x}$ and m points on $\boldsymbol{S_y}$, we should therefore need $m.n$ operations using the conventional approach. The FMM, on the other hand, uses the point $\mathbf{O}$ to represent $\boldsymbol{S_y}$, the contributions from $\boldsymbol{S_y}$ are thus carried out and transferred to every point $\mathbf{x}$ via $\mathbf{O}$, the total number of operations is now reduced to only $m + n$ which is much smaller than $m.n$. Therefore, the number of operations is reduced significantly in the evaluation of double integrals of SGBEM (which is also equal to the matrix-vector multiplication). This greatly improves the performance of the overall system solution. The figure (2.8b) shows the $O(N.M)$ complexity if standard evaluations of double integration are called, while with FMM, the operation count is reduced to $O(N + M)$.

2.4.2 Single-level fast multipole method

The first and simple variant of the FMM which is derived directly from the basic concept is called the Single-level FMM. In this approach, the domain Ω is contained and divided by a cubic grid of step d (Fig.2.9a). Only cells containing boundary elements are taken into consideration. The center of a cell plays the role of an

intermediate pole from which both the transfer of the contribution of its elements and the expansion of the faraway influences takes place. The conventional SGBEM is, on the other hand, considered when two cells are adjacent. The evaluation of the boundary integral is then composed of the traditional SGBEM and the quick computation of Fast algorithm (Fig.2.9b). Compared with the classical SGBEM, the single-level FMM is more efficient with a complexity of $O(N^{3/2}/2)$. However, a more efficient scheme can be achieved by adopting the multi-level FMM which is described in more detail in the next section.



(a) (b)

Figure 2.9 – (a) 2D grid (spacing d) occupying the domain Ω (b) Cells interaction in the single-level FMM configuration

2.4.3 Multi-level fast multipole method

In order to obtain maximal efficiency, the amount of traditional SGBEM calculation should be minimal while clustering the distant groups as much as possible. The fast multipole algorithm must therefore be applied in a hierarchical manner (in a multi-level approach). This is done with the help of an *oct-tree* structure (See Fig.2.10): at the first step ($level = 0$ or *roof*), a cube which contains the whole studied solid Ω is generated, then it is divided into 8 equal and smaller cubes ($level = 1$) (whose edge length is half of the parent cube's). The cell subdivision is continued until the number of elements in a cell is smaller than a given value (which is called max_elem). Any given boundary element is deemed to belong to one cell of a given level only, even if is geometrically shared by two or more same-level cells.

We will now give a number of definitions of usual terms of the fast multipole algorithm:

- ***cell*** - being a unit of octree structure, *Cell* takes the form of a square in 2D and a cube in 3D. They are divided in a hierarchical manner and all contain

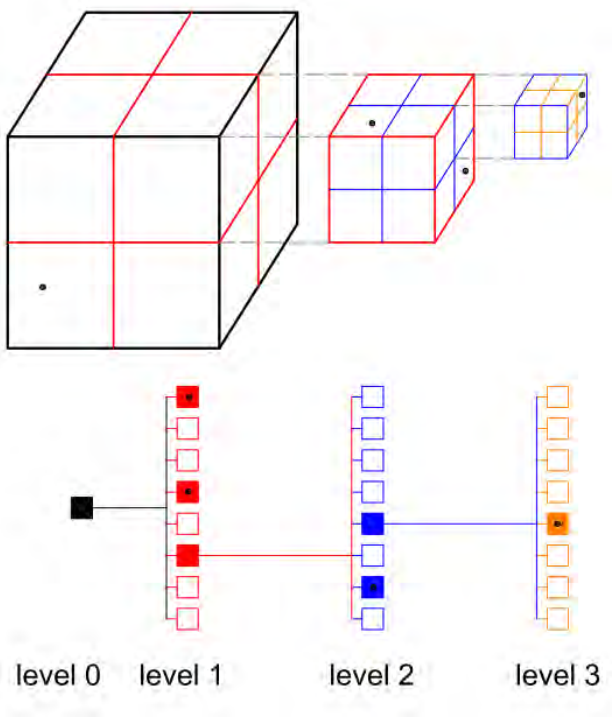


Figure 2.10 – Octree structure

boundary elements. The relative positions between cells are used to determine which operation or which calculation scheme should be used.

- ***parent, children*** - a generic cell C, at level l can be a child to a cell in level $l-1$ but it can also be the parent of cells in level $l+1$. A cell can have a maximum of four children in 2D and eight children in 3D.

- ***leaf*** - a cell is called a *leaf* either if it has no child or the number of elements in it does not exceed the predefined parameter *max_elem*. The fast multipole algorithm implies that the computation is valid when the *octree* structure has at least two levels (such that *far-away interactions* exist).

- ***adjacent*** - two cells of a same level l are called *adjacent* if they share at least one vertex, or edge, or surface. In 2D, a generic cell can have 8 adjacent cells. In 3D, a cell can have at most 26 adjacent cells.

- ***interaction list*** - two cells are said to be *well-separated* at level l if they are not adjacent at level l but their parent cells are adjacent at level $l-1$. A list of all cells that are *well-separated* with cell C, at level l is called *interaction list* of cell C. The maximum number of *well-separated* cells in 2D is $6^2 - 3^2 = 27$ and in 3D is $6^3 - 3^3 = 189$.

- ***near interaction*** - For a cell C, the *near interaction* between C and its adjacent cells are computed either if cell C is a *leaf* or C is not but an adjacent cell to C is a *leaf*. These interactions are computed with the conventional SGBEM formulations.
- ***far-away interaction*** - All the cells that are not adjacent to cell C, at level l are called *far-away* or *distant* to cell C and the interaction between them is computed with the FMM operations.

For simplicity purposes, the introduction of the FMM algorithm is coped with the SGBEM formulations. We choose to employ only the FMM operations for the term $\mathbf{B}_{tt}$ (the other bilinear terms are treated analogously and can be found in [51]). Considering the symmetry of the Kernel function U_i^k, $\mathbf{B}_{tt}$ can be written as:

$$\begin{aligned}\mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}}) &= \int_{S_u}\int_{S_u} t_k(\mathbf{x})U_i^k(\mathbf{x},\tilde{\mathbf{x}})\tilde{t}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \\ &= \int_{S_u}\int_{S_u} \tilde{t}_i(\mathbf{x})U_k^i(\mathbf{x},\tilde{\mathbf{x}})t_k(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x\end{aligned} \tag{2.50}$$

$$\begin{aligned}\mathbf{B}_{tt}(\mathbf{t}^D,\tilde{\mathbf{t}}) &= \int_{S_u}\int_{S_u} t_k^D(\mathbf{x})U_i^k(\mathbf{x},\tilde{\mathbf{x}})\tilde{t}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \\ &= \int_{S_u}\int_{S_u} \tilde{t}_i(\mathbf{x})U_k^i(\mathbf{x},\tilde{\mathbf{x}})t_k^D(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x\end{aligned} \tag{2.51}$$

The multipole expansion of r^{-1} in the Kelvin solution U_i^k is given in [75] by:

$$\frac{1}{r} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^n R_{nm}(\tilde{x}')\sum_{n'=0}^{n}\sum_{m'=-n'}^{n'}\overline{S_{n+n',m+m'}(\mathbf{r}_0)}R_{n'm'}(\mathbf{x}') \tag{2.52}$$

where

$$\begin{aligned}R_{nm}(\mathbf{y}) &= \frac{1}{(n+m)!}P_n^m(cos\alpha)e^{im\beta}\rho^n \\ S_{nm}(\mathbf{y}) &= (n-m)!P_n^m(cos\alpha)e^{im\beta}\rho^{-(n+1)}\end{aligned} \tag{2.53}$$

(ρ,α,β) are the spherical coordinates of the argument $\mathbf{y}$. P_n^m denotes the Legendre polynominals and the overbar denotes the complex conjugation. R_{nm} and S_{nm} can be effectively evaluated without actual recourse to spherical coordinates by means of the recursive formulae proposed in [75] (brief description in Annex B). Figure 2.11 demonstrates the principle of the FMM as one computes the interaction of two surfaces S_x and $S_{\tilde{x}}$:

In order to apply the FMM algorithm, the Kernels U_i^k, T_i^k, B_{ikqs} are decomposed into multipole series. For this purpose, the relative position vector $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$ is decomposed into: $\overrightarrow{\mathbf{xO}}+\overrightarrow{\mathbf{OO'}}+\overrightarrow{\mathbf{O'x_0}}+\overrightarrow{\mathbf{x_0x_1}}+\overrightarrow{\mathbf{x_1\tilde{x}}}$ (Fig.2.11). Hence, the interaction between 2 boundary portions S_x and $S_{\tilde{x}}$ is truncated into a number of steps:

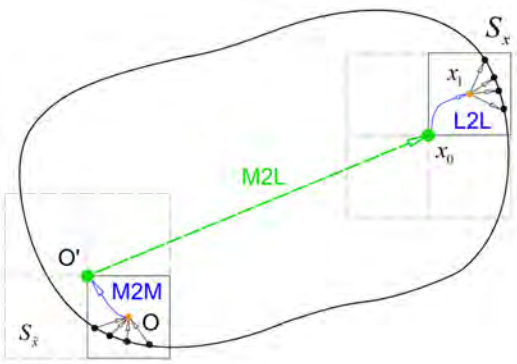


Figure 2.11 – Decomposition of the position vector

Multipole Moments

In this step, we start with introducing the pole **O** as a representative for the group of points in $S_{\tilde{x}}$. The multipole moments associating with the pole **O** are the first FMM operation being computed here.

The Kelvin fundamental solution U_i^k can be rewritten as:

$$U_k^i(\mathbf{x},\tilde{\mathbf{x}}) = \frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\overline{F_{ik,n,m}^{Stt}(\overrightarrow{Ox})} + \overline{G_{i,n,m}^{Stt}(\overrightarrow{Ox})}(\overrightarrow{Ox}_k)\right)R_{n,m}(\overrightarrow{O\tilde{x}}) \tag{2.54}$$

where

$$\begin{aligned} F_{ik,n,m}^{Stt}(\overrightarrow{Ox}) &= \left(\frac{3-4\nu}{2(1-\nu)}\delta_{ik} - \frac{1}{21-\nu)}(\overrightarrow{Ox_k})\frac{\partial}{\partial x_i}\right)S_{n,m}(\overrightarrow{Ox}) \\ G_{i,n,m}^{Stt}(\overrightarrow{Ox}) &= \frac{1}{2(1-\nu)}\frac{\partial}{\partial x_i}S_{n,m}(\overrightarrow{Ox}) \end{aligned} \tag{2.55}$$

The formula of $B_{tt}(\mathbf{t},\tilde{\mathbf{t}})$ can be written as:

$$B_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_i(\tilde{\mathbf{x}})\left[\overline{F_{ik,n,m}^{S}(\overrightarrow{Ox})}M_{k,n,m}^{1tt}(O) + \overline{G_{i,n,m}^{S}(\overrightarrow{Ox})}M_{n,m}^{2tt}(O)\right]dS_x \tag{2.56}$$

in which the multipole moments centered at **O** are:

$$\begin{aligned} M_{knm}^{1}(O) &= \int_{S_u} R_{nm}(\overrightarrow{O\tilde{x}})t_k(\tilde{\mathbf{x}})dS_{\tilde{x}} \\ M_{nm}^{2}(O) &= \int_{S_u} R_{nm}(\overrightarrow{O\tilde{x}})(\overrightarrow{O\tilde{x}})_k t_k(\tilde{\mathbf{x}})dS_{\tilde{x}} \end{aligned} \tag{2.57}$$

M2M translation

Now, the influence of $S_{\tilde{x}}$ is transferred from pole $\mathbf{O}$ to pole $\mathbf{O'}$. The multipole moment centered at $\mathbf{O'}$ is given by:

$$M^1_{knm}(O') = \int_{S_u} R_{nm}(\overrightarrow{O'\tilde{x}})t_k(\tilde{\mathbf{x}})dS_{\tilde{x}}$$
$$M^2_{nm}(O') = \int_{S_u} R_{nm}(\overrightarrow{O'\tilde{x}})(\overrightarrow{O'\tilde{x}})_k t_k(\tilde{\mathbf{x}})dS_{\tilde{x}} \tag{2.58}$$

taking into account the relation between solid harmonic $R_{n,m}$ and $S_{n,m}$:

$$\overline{S_{n,m}}(\overrightarrow{Ox}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n} R_{n',m'}(\overrightarrow{O'O})\overline{S_{n+n',m+m'}}(\overrightarrow{O'x}) \tag{2.59}$$

$$R_{n,m}(\overrightarrow{O'\tilde{x}}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n} R_{n-n',m-m'}(\overrightarrow{O\tilde{x}})\overline{R_{n',m'}}(\overrightarrow{O'O}) \tag{2.60}$$

we can have:

$$R_{n,m}(\overrightarrow{O'\tilde{x}}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n} R_{n',m'}(\overrightarrow{O'O})R_{n-n',m-m'}(\overrightarrow{O\tilde{x}}) \tag{2.61}$$

Substituting (2.61) into (2.58) we obtain:

$$M^1_{knm}(O') = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{O'O})M^{1tt}_{k,n-n',m-m'}(O)$$
$$M^2_{nm}(O') = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{O'O})\left[M^{2tt}_{n-n',m-m'}(O) - (\overrightarrow{OO'})_k M^{1tt}_{k,n-n',m-m'}(O)\right] \tag{2.62}$$

M2L translation

In this step, the M2L operation translates the coefficients from pole $\mathbf{O'}$ to pole $\mathbf{x_0}$. From (2.59) we have:

$$\begin{aligned}\overline{S_{n,m}}(\overrightarrow{Ox}) &= \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{x_0}\mathbf{x}})\overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{x_0}O}) \\ &= (-1)^{n'}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{x_0}\mathbf{x}})\overline{S_{n+n',m+m'}}(\overrightarrow{O\mathbf{x_0}})\end{aligned} \tag{2.63}$$

Replacing (2.63) into (2.56) we get:

$$B_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \frac{1}{8\pi\mu}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}\int_{S_u}\tilde{t}_i(\mathbf{x})\left(F^{Rtt}_{ik,n',m'}(\overrightarrow{\mathbf{x_0x}})L^{1tt}_{k,n',m'}(\mathbf{x_0}) + G^{Rtt}_{i,n',m'}(\overrightarrow{\mathbf{x_0x}})L^{2tt}_{n',m'}(\mathbf{x_0})\right)dS_x \tag{2.64}$$

where $L^{1tt}_{k,n,m}(\mathbf{x_0})$ and $L^{2tt}_{n,m}(\mathbf{x_0})$ are the coefficients of the local expansion centered at $\mathbf{x_0}$, given by:

$$L^{1tt}_{k,n,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}(-1)^n\overline{S_{n+n',m+m'}}(\overrightarrow{O\mathbf{x_0}})M^{1tt}_{k,n',m'}(O) \tag{2.65}$$

$$L^{2tt}_{n,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}(-1)^n\overline{S_{n+n',m+m'}}(\overrightarrow{O\mathbf{x_0}})\left(M^{2tt}_{n',m'}(O) - (\overrightarrow{O\mathbf{x_0}})_k M^{1tt}_{k,n',m'}(O)\right) \tag{2.66}$$

and

$$F^{Rtt}_{ik,n,m}(\overrightarrow{\mathbf{x_0x}}) = \left(\frac{3-4\nu}{2(1-\nu)}\delta_{ik} - \frac{1}{2(1-\nu)}(\overrightarrow{\mathbf{x_0x}_k})\frac{\partial}{\partial\mathbf{x}_i}\right)R_{n,m}(\overrightarrow{\mathbf{x_0x}}) \tag{2.67}$$

$$G^{Rtt}_{i,n,m}(\overrightarrow{\mathbf{x_0x}}) = \frac{1}{2(1-\nu)}\frac{\partial}{\partial\mathbf{x}_i}R_{n,m}(\overrightarrow{\mathbf{x_0x}}) \tag{2.68}$$

L2L translation

The last step consists of shifting from pole $\mathbf{x_0}$ to pole $\mathbf{x_1}$ which represents the group of source points S_x, then expanding the coefficients to each source points $\mathbf{x}$. By doing so, the boundary integral equation of B_{tt} is evaluated.

From (2.60) we have:

$$R_{n,m}(\overrightarrow{\mathbf{x_0x}}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}R_{n',m'}(\overrightarrow{\mathbf{x_1x}})R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0x_1}}) \tag{2.69}$$

Substituting (2.69) into (2.64) we obtain:

$$B_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \frac{1}{8\pi\mu}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}\int_{S_u}\tilde{t}(\mathbf{x})\left(F^{Rtt}_{ik,n',m'}(\overrightarrow{\mathbf{x_1x}})L^{1tt}_{k,n',m'}(\mathbf{x_1}) + G^{Rtt}_{i,n',m'}(\overrightarrow{\mathbf{x_1x}})L^{2tt}_{n',m'}(\mathbf{x_1})\right)dS_x \tag{2.70}$$

where $L^{1tt}_{k,n,m}(\mathbf{x_1})$ and $L^{2tt}_{n,m}(\mathbf{x_1})$ are the coefficients of the local expansion centered at pole $\mathbf{x_1}$, given by:

$$L^{1tt}_{k,n,m}(\mathbf{x_1}) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} (-1)^n R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}}) L^{1tt}_{k,n',m'}(\mathbf{x_0}) \tag{2.71}$$

$$L^{2tt}_{n,m}(\mathbf{x_1}) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} (-1)^n R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}}) \left(L^{2tt}_{n',m'}(\mathbf{x_0}) - (\overrightarrow{\mathbf{x_0}\mathbf{x_1}})_k L^{1tt}_{k,n',m'}(\mathbf{x_0}) \right) \tag{2.72}$$

and

$$F^{Rtt}_{ik,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) = \left(\frac{3-4\nu}{2(1-\nu)} \delta_{ik} - \frac{1}{2(1-\nu)} (\overrightarrow{\mathbf{x_1}\mathbf{x}_k}) \frac{\partial}{\partial \mathbf{x}_i} \right) R_{n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) \tag{2.73}$$

$$G^{Rtt}_{i,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) = \frac{1}{2(1-\nu)} \frac{\partial}{\partial \mathbf{x}_i} R_{n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) \tag{2.74}$$

Replacing all the coefficients of local expansion in the formula of B_{tt}, we finally obtain the integral equation evaluated.

To conclude, the FMM implicitly splits the SGBEM matrix $\mathbf{K}$ into $\mathbf{K} = \mathbf{K}_{\text{near}} + \mathbf{K}_{\text{FMM}}$, where $\mathbf{K}_{\text{FMM}}$ gathers the contributions arising from multipole expansions and $\mathbf{K}_{\text{near}}$ the close-range influence coefficients that have to be computed by traditional BEM quadrature (see Fig. 2.12 for a schematic description). The matrix $\mathbf{K}_{\text{FMM}}$ is of course not actually set up; rather, the FMM evaluates products $\mathbf{K}_{\text{FMM}}.\mathbf{X}$ that are used by an iterative solver.

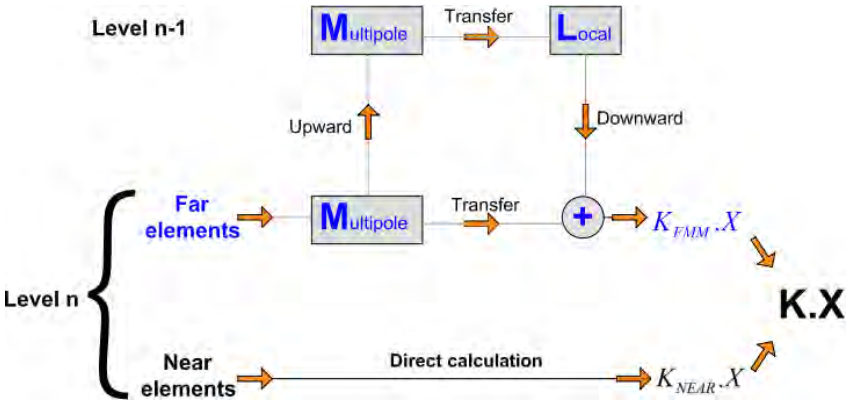


Figure 2.12 – Two-level fast multipole operations

2.5 Conclusions

In this chapter, the basic foundations of *MBEMv2.0* are introduced. The Galerkin formulation for elasticity problems and fracture mechanics problems have been introduced. The boundary element analysis of the Galerkin equations are then presented. The fast multipole method is then introduced to accelerate the conventional

Galerkin boundary element method. The extension of the Galerkin formulations to multizone problems is finally introduced. In the next chapter, we will discuss further optimizations of the algorithm.

Chapter 3

Optimizations of *MBEMv2.0*

Contents

Although *MBEMv2.0* inherits a number of innovative algorithms from the BE community, it is still not simple to simulate large-scale realistic engineering problems on moderate computational resources efficiently. In this chapter, many developments have been devoted to further efficiency improvements of the existing code. First, existing useful computational results are saved and reused during the propagation. A non-zero initial guess is used to reduce the iterative solver duration. Some time-consuming phases of the code are accelerated by a shared memory parallelism. A new sparse matrix method is designed based on coordinate format and compressed sparse row format to limit the memory required during the matrix construction phase. The remarkable performance of the new code is shown through many simulations including large-scale problems ($N \geq 3.10^6$).

3.1 Presentation of *MBEMv2.0*

3.1.1 *MBEMv2.0* algorithm

MBEMv2.0 is based on SGBEM coupled with FMM for the simulation of fracture problems. The code can then simulate piece-wise homogeneous domains (Fig. 1.4) which can contain cracks (Fig. 1.5). Let us summarize some important numerical aspects of the code in this section. The main phases of the initial crack propagation algorithm are summarized in Fig. 3.1. In a nutshell, the aim is to solve a linear system ($K.X = b$) with an iterative solver: the Flexible GMRES. The system matrix K is composed of double surface integrals $\mathcal{B}_{tt}$, $\mathcal{B}_{tu}$ and $\mathcal{B}_{\Delta u \Delta u}$. $\mathcal{B}_{tt}$ for example, is double integral over two surfaces Su. The generic double surface integral takes the following aspect:

$$I(S_e, S_f) = \int_{S_e} \int_{S_f} f(\mathbf{x}) G(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) dS_{\mathbf{y}} dS_{\mathbf{x}} \tag{3.1}$$

where S_e and S_f are the surfaces of source and field elements ($\mathbf{x} \in S_e$,$\mathbf{y} \in S_f$), $f(\mathbf{x})$ and $g(\mathbf{y})$ are respectively known and test function. $G(\mathbf{x}, \mathbf{y})$ is the kernel which contains the singularity $O(r^{-1})$ or $O(r^{-2})$.

The matrix K can be separated in two parts: K_{near} and K_{FMM}. K_{FMM} consists of the integrals over far enough surfaces, calculated with the FMM. K_{near} consists of the integrals over near surfaces. Singularities will occur when these two surfaces S_e and S_f are coincident, adjacent by edge or adjacent by vertex. The singular integrals are evaluated using special schemes presented by Andrä and Schnack [52]. The regular integrals are evaluated with normal quadrature rule:

$$I(S_e, S_f) = \int_{\Delta_e} \int_{\Delta_f} f(\mathbf{x}(\boldsymbol{\eta})) G(\mathbf{x}(\boldsymbol{\eta}), \mathbf{y}(\boldsymbol{\xi})) g(\mathbf{y}(\boldsymbol{\xi})) J_y(\boldsymbol{\xi}) J_x(\boldsymbol{\eta}) d\boldsymbol{\xi} d\boldsymbol{\eta} \tag{3.2}$$

where $\Delta_e \in [-1,1] \times [-1,1]$ and $\Delta_f \in [-1,1] \times [-1,1]$. This integral can be approximated by:

$$I(S_e, S_f) \simeq \sum_{i=1}^{N_{pge}} \sum_{j=1}^{N_{pgf}} f(\boldsymbol{\eta}_i) G(\boldsymbol{\eta}_i, \boldsymbol{\xi}_j) g(\boldsymbol{\xi}_j) J_y(\boldsymbol{\xi}_j) J_x(\boldsymbol{\eta}_i) A^j_{\boldsymbol{\xi}_j} A^i_{\boldsymbol{\eta}_i} \tag{3.3}$$

where $\boldsymbol{\eta}_i$ and $A^i_{\boldsymbol{\eta}_i}$ denote the abscissas and weights of the gaussian points for exterior elements; $\boldsymbol{\xi}_j$ and $A^j_{\boldsymbol{\xi}_j}$ denote the corresponding parameters for the interior elements. N_{pge} and N_{pgf} are the number of gaussian points for exterior and interior elements respectively.

Number of Gaussian points

The number of gaussian points is crucial to determine the trade-off between the computational speed and precision. As the surface integral equations deal with singularity $O(r^{-1})$, higher degree of precision should be applied for close interactions

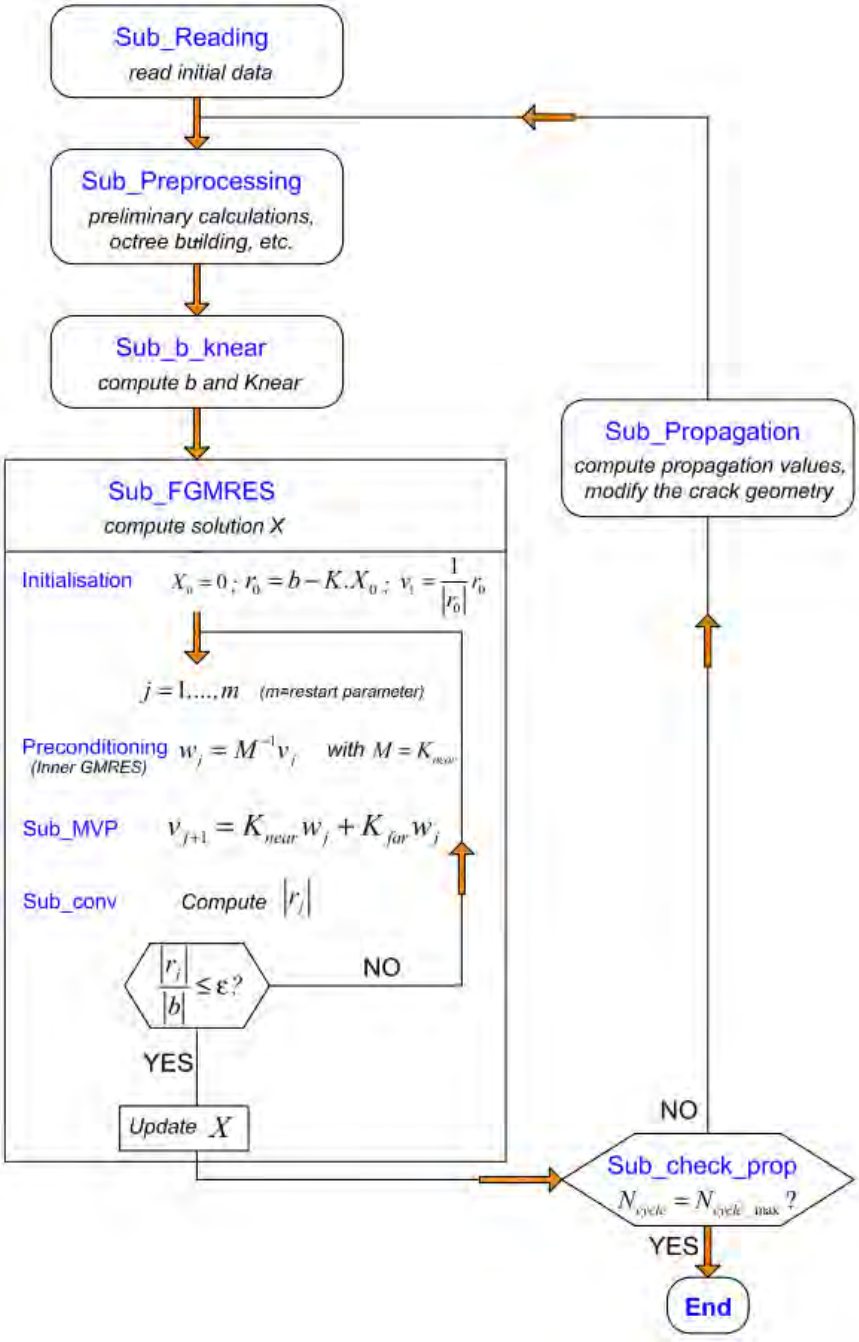


Figure 3.1 – Initial FM-SGBEM code for crack propagation

and less points for far interactions. An empirical scheme introduced by Rezayal et al. [54] is used for selecting the appropriate number of gaussian points. The choice depends on the ratio of element size to the distance between two elements. For quadrilateral elements, the element size, H is taken here as the length of the

longest diagonal. The distance between the centers of two elements is denoted by d. The line between these two centers form an angle θ with the normal of the exterior element, see Fig. 3.1. An index of severity IS has been introduced to quantify the variation in the required degree of the integration formula, see Eq 3.4. IS is rounded off to the nearest integer. The correlation between IS and the number of Gaussian points is given in Table 3.4. This technique implemented in large-scale tests by Trinh [48] obtained a boost in terms of near computational time.

$$IS = (2.37 + 0.424 \cos\theta)\, H/d \tag{3.4}$$

Table 3.1 – Table of severity index

IS	Number of Gauss points
1	2x2=4
2	3x3=9
3	4x4=16
4	5x5=25
5	6x6=36
6	4x(4x3)=64
7	4x(5x5)=100
8	4x(6x6)=144

With the matrix separated, the linear system to solve can be written as in equation 3.5. It's important to note that due to the FMM, K_{FMM} is not stored, the matrix-vector product $K_{FMM} * X$ is computed directly. When the convergence is achieved in the iterative solver, the crack propagates and the elastostatic code is repeated. The main phases of *MBEMv2.0* algorithm (Fig. 3.1) can be described as follows (a detailed user guide is presented in Annex C) :

$$(K_{near} + K_{FMM})\, .X = b \tag{3.5}$$

Sub_Reading At the first step, initial data are read. They are geometry files containing the node coordinates, the connectivity table, etc. The geometry files are Generated automatically with *GiD* software. There are also parameter files which contain various parameters for example about the iterative solver.

Sub_Preprocessing Then, some preliminary operations are realized. They include unknown generation, octree structure generation for fast multipole method operations, etc.

Sub_b_knear The right hand side vector b is computed. This can be divided in b_{near} and b_{FMM}. The matrix K_{near} of the integrals over near surfaces is also computed and stored. We call this step the *preparation phase* or the *matrix computation phase*. At the end of this step, the system is ready for resolution.

Sub_FGMRES This step called *the solving phase* consists of using Flexible GMRES to obtain an approximate solution. Basically, Flexible GMRES is a scheme of two iterative solvers: GMRES plays the role of the outer solver (main problem) and for the inner solver (preconditioning task), any iterative method can be used. An inner GMRES is used here for the preconditioning task and K_{near} is employed as a preconditioner in the algorithm of the Flexible GMRES. The evaluation of matrix-vector product which is required at each step of the iterative solver is divided in two parts $K_{global}w = K_{near}w + K_{FMM}w$. $K_{near}w$ is evaluated in compressed form and $K_{FMM}w$ is evaluated with multipole operations. When the backward error reaches the given tolerance, the last solution is computed, and the solving phase stopped.

Sub_check_prop This routine manages the propagation. In a nutshell, it permits to know if the calculations should continue or not.

Sub_Propagation Here is an important part of the propagation which consists of the extension of the crack geometry. This will be discussed in *chapter 4*.

3.1.2 Computation time and directions for optimization work

Computation time

Computation time is largely presented in [48] for several simulations. As presented in Fig. 3.1, the crack propagation consists of resuming the elastostatic calculations. We notice that the computing cost increases at each propagation step and the total computing time is too long even for very simple problems. Let us consider the propagation of 64 penny-shaped cracks in a clamped cube, (similar to Fig. 1.5.b.). Table 3.2 presents the computation time at each propagation step. In this table, N_{dofs} is the number of degrees of freedom, N_{iter} is the number of iterations of the solver, T_{pre} is the duration of the preparation phase, T_{sol} is the duration of the solving phase and T_{tot} is the total duration of the simulation. The mesh density varies from 32 718 unknowns (initial state) to 106 446 unknowns (last increment). The computation time for each cycle changes from 35 minutes (initial state) to 10 hours (last increment). The GMRES solver converges in 15 iterations for the initial state and in 103 iterations for the last increment. The ten propagation increments take 24 hours. After 9 increments the solver could not converge after 300 iterations. It can be clearly seen that all durations increase from one step to another.

The total computation time T_{tot} is mainly composed of two parts: the duration of the preparation phase T_{pre} and the duration of the solving phase T_{sol}. It can also be written in the form of:

$$T_{tot} = T_{pre} + N_{iter}T_{iter} \tag{3.6}$$

where N_{iter} is the number of iterations and T_{iter} is the duration of one iteration.

Table 3.2 – Crack propagation time with *MBEMv2.0*

#	N_{dofs}	N_{iter}	T_{pre} (s)	T_{sol} (s)	T_{tot} (s)
1	32 718	15	624	1509	2135
2	41 934	15	1039	1883	2925
3	51 150	16	1568	2394	3966
4	60 366	18	2219	3118	5342
5	69 582	19	3020	3753	6778
6	78 798	25	3717	5520	9244
7	88 014	20	4002	5007	9018
8	97 230	35	4219	9543	13 774
9	106 446	103	4421	30 676	35 111

For the simulation of a three-layered pavement (that will be presented in chapter 6, the calculations could not converge even when the pavement contains only one stationary crack. To be able to simulate crack propagation in pavements, it is thus essential to find other techniques to speed up the code.

An important parameter: max _*elem*

max _*elem* is the maximal number of elements in a octree leaf. This input parameter permits to construct the octree structure, see subsection 2.4.3. Table 3.3 presents the influence of max _*elem* value on the computing cost. In this table, T_{octree} is the duration of the octree construction. If this parameter is small, the octree's depth increases, the number of cells too. In large scale, the duration of octree construction becomes too long and the number of near interactions decreases. K_{near} matrix is thus very sparse, can be computed in a short time and the storage does not require large memory space. The low quality of K_{near} leads to a low convergence rate of the Flexible GMRES.

Table 3.3 – Influence of max _*elem*

max _*elem*	Small (10, 15, etc.)	Large (500, 1000, etc.)
Memory space	Low	Large
T_{octree}	Long	Short
T_{pre}	Short	Long
N_{iter}	High	Small
T_{iter}	Long	Short

Directions for optimization work

- Reduction of memory space: In this work the computed matrices are stored in symmetric compressed sparse row format (CSRSYM), see Rose and Willoughby [76] or Brameller et al. [77]. This algorithm takes only three

vectors to store all the necessary information of the matrix. It allows matrix-vector product in compressed form. Due to the great performance of the format (see [48]), other storage reduction methods are not studied in this work. However, a threshold can be set to avoid storing values that are too small. Indeed, only the exact zeros are not stored with the current format, all non-zero values, even very small ones, are stored.

- Reduction of T_{pre} : To accelerate the computation of K_{near} matrix, one possibility is parallel computing. This will be discussed in section 3.3. Another possibility in crack propagation problems, is the possible reusing of some parts of the K_{near} matrix from one propagation step to another. Only the parts relating to the new elements would be computed. We should get a significant reduction because the new elements are not too much.

- Reduction of N_{iter} : Flexible GMRES is used in this work for the iterative solver. To reduce its iteration number, the preconditioning task can be improved. An efficient algebraic preconditioner can be constructed by hierarchical matrix representations (see [78]) for example to speed up the convergence of the solver. The construction of a preconditioner is a complex task and is not considered in this work. Another possibility is the use of a non-zero initial guess. This can be achieved for crack propagation simulation by using part of the solution of a propagation step as an initial guess for the following step.

- Reduction of T_{iter} : In this work we do not consider an optimization of the Flexible GMRES solver. The other possibilities are therefore the external tasks it requires. The main external tasks are the preconditioning and the matrix-vector product. For the preconditioning task, a fast-direct solver can be investigated. The matrix-vector product is already optimized as it is achieved in part in compressed format and in part with FMM formulation. One of the remaining possibilities is again parallel computing which will be discussed in section 3.3.

3.2 Data reusing and fast matrix update

3.2.1 Data reusing technique

In the initial algorithm, during each cycle, a layer of new elements is added to the crack geometry during the re-meshing routine (Sub_Propagation in Fig. 3.1). After the re-meshing the system needs to be recomputed, especially the matrix K_{near} of near interactions. If one rebuilds the coefficient matrix, the interactions between pairs of old elements will be repeated and will require wasteful operations because nothing (*a priori*) changes in the calculation of those interactions. Therefore, starting from the second cycle, the interactions between pairs of old elements are re-used. Only the parts of the matrix that are related to the newly added elements are computed, see Fig. 3.2. Algorithm 1 presents the initial matrix computation while

Algorithm 2 presents the data reusing strategy.

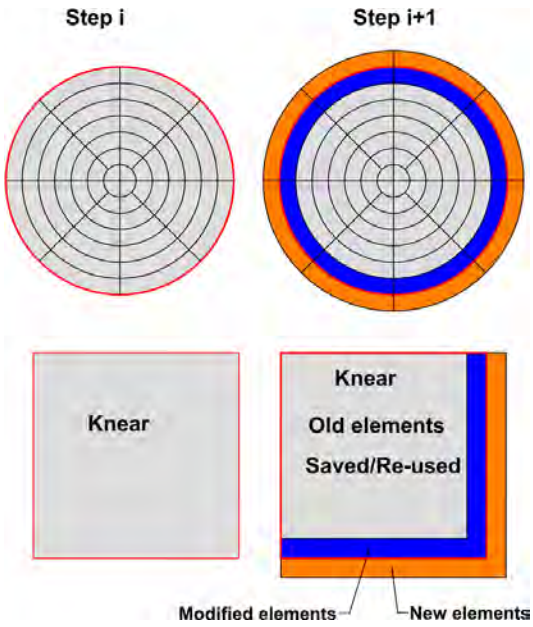


Figure 3.2 – Efficient K_{near} updating.

Algorithm 1 Initial computation

```
for i = 1, Ncyclemax do
    Call Build_Octree
    K_near = 0
    for ee = 1, Nelem do
        Compute contributions of ee
    end for
    Call Sub_FGMRES
    Call Sub_Propagation
end for
```

Algorithm 2 Fast computation

```
K_near = 0 ; K_near^front = 0
for i = 1, Ncyclemax do
    if i = 1 then
        Call Build_Octree
        Set all elements ee to new
    else
        Call Update_Octree
        Set newly added elements to new
        Set old crack front elements to new
    end if
    K_near = K_near − K_near^front
    K_near^front = 0
    for ee = 1, Nelem do
        if ee is new then
            Compute contributions of ee
            Add contributions to K_near
            if ee is in front then
                Save contributions in K_near^front
            end if
        end if
    end for
    Call Sub_FGMRES
    Call Sub_Propagation
end for
```

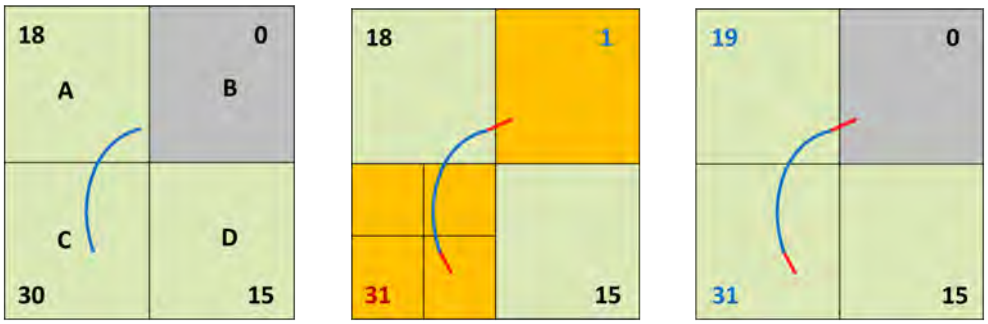
Fixed octree structure

Re-using the interactions between pairs of old elements for the computation of K_{near} is a simple idea, but one must be sure to keep the same conditions as the initial configuration. For example, the octree structure must be fixed from one cycle to another so that near elements stay near elements and the same thing goes for far elements. So, as shown in algorithm 2, the octree is built only during the first cycle by taking as main input the maximum number of elements in a leaf *max_elem* and by using the position of the elements, see subsection 2.4.3. If the same algorithm is reused in the following propagation cycles, complications can occur. Let us consider for illustration purposes a 2D crack with a quad-tree structure presented in Fig. 3.3.a in which only some cells around the crack are shown. Let us assume that *max_elem* is 30 and the value at each corner is the number of elements in the cell. When the crack propagates, if the same octree construction algorithm is used, the configuration presented in Fig. 3.3.b will be obtained. Cell B must be created, and cell C is no longer a leaf. The near interactions change and can not therefore be reused.

To solve this problem, we propose a fixed octree structure starting from the second propagation cycle. Newly added elements are affected to existing octree cells by the *Update_Octree* routine. At a given cycle i, for each existing octree cell c, this routine consists of:

1. Identifying old crack front elements $elem_{c,i-1}^{front}$ belonging to cell c.
2. Identifying for each $elem_{c,i-1}^{front}$, the corresponding new crack front element $elem_{c,i}^{front}$.
3. Adding the new element $elem_{c,i}^{front}$ to the element list of cell c.

The configuration presented in Fig. 3.3.c is then obtained for the 2D illustration. max_elem can therefore be exceeded as it can be noticed in cell C.



(a) (b) (c)

Figure 3.3 – (a) Initial quad-tree (b) Rebuilded (c) Fixed

Quarter-point elements

In the initial code quarter-point elements are used at the crack front, and they are returned to normal Q8 elements before adding new quarter-point elements at the new crack-front. So, those elements are modified and their contributions in the matrix are not the same after the propagation. That means their contributions can not be simply re-used. To solve this problem, the contributions of modified elements are saved format in another matrix. After each increment, these contributions are removed from the existing matrix (algorithm 2, line 11). Then the modified elements are set as new elements (algorithm 2, line 9) so that their new contributions are computed (algorithm 2, line 15). The treatment of crack front elements is therefore based on the equation 3.7. It is important to note that the position of the unknowns can change from one cycle to another. Permutation operations must be performed in these cases on the matrices of the previous cycle. All these matrix manipulations are performed while maintaining a compressed matrix format: the Compressed Sparse Row (CSR).

$$K_i = \left(K_{i-1} - K_{i-1}^{front}\right) + \left(K_i^{oldfront} + K_i^{front}\right) \tag{3.7}$$

Models description

Let us present the models used in this work for performance comparison. The models are about a crack array embedded in a clamped cube of edge 3 000 mm, subjected to uniform tensile load $p = 1MPa$ at the top face. The crack array contains n_c^3 randomly-oriented penny-shaped cracks ($r_c = 25mm$) on a cubic grid of step d_c. The center of the crack array is located at the center of the cube. The distance d_c is sufficiently large to avoid influences between cracks. Each crack of the crack array (see Fig. 3.6, the distance d_c is reduced for this figure) is meshed with 48 Q8 elements with 161 nodes, see Fig. 3.4. For some simulations, the cracks are meshed with 768 Q8 elements with 2 369 nodes, see Fig. 3.5. The cube and the position of the cracks are presented in Fig.3.7 and 3.8. The models are named Cn, n being the crack number. For example, a model of cube containing 8 cracks is noted $C8$. Under the load, the cracks propagate horizontally. Complex configurations like road structures will be discussed in following chapters. Here we choose simple models to be able to compare with the initial version *MBEMv2.0*.

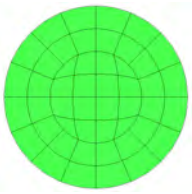


Figure 3.4 – 48 elements crack mesh

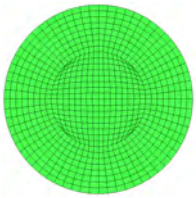


Figure 3.5 – 768 elements crack mesh

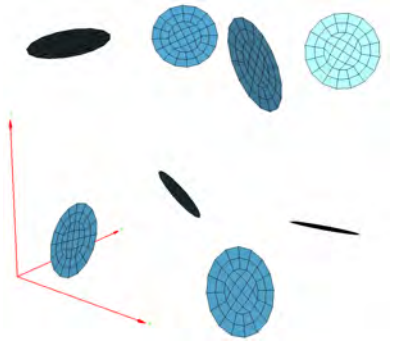


Figure 3.6 – Crack array 2x2x2

3.2.2 Data reusing results

With fast computation, the cost of re-constructing the coefficient matrix K_{near} can be greatly reduced especially while dealing with a few number of cracks. The effect of fast computation is shown in Table 3.4 for different models. In Table 3.4, T_{pre} is the accumulated preparation time from cycle 2 to cycle 10. Fig. 3.9 presents the duration of the preparation phase for each cycle, for model C8. It is important to notice that this fast matrix update based on equation 3.7 is not theoretically subjected to additional errors. But in practice, minimal differences (less than 1% can be observed, due to numerical errors: lack of precision, accumulation of round off error, etc. The speedup due to this fast matrix update obviously depends on the ratio of the number of new elements added (to propagate the cracks) to the number

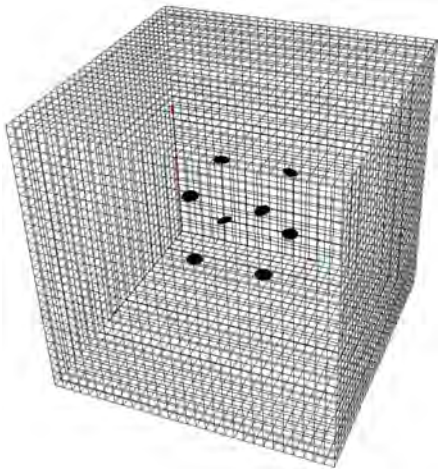


Figure 3.7 – Model C8

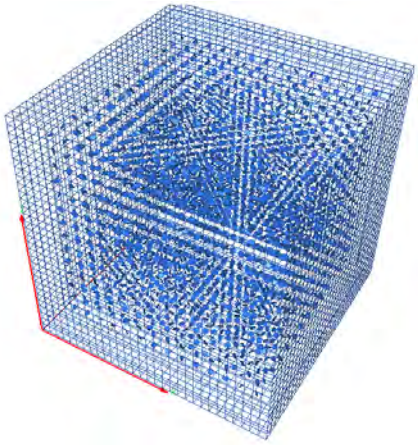


Figure 3.8 – Model C2744

of existing element (including existing crack elements).

$$ratio = \frac{N_{new}}{N_{existing}}$$

The smaller this ratio is, the greater the speedup will be. During the propagation, the number of existing elements increases from one cycle to another. As a result, the speedup increases during the propagation. This is shown on Fig. 3.9. This data reusing technique is thus very interesting for large scale problems.

Table 3.4 – Fast K_{near} computation: Results

#	Model	N_{dofs}^{init}	N_{cycles}	N_{dofs}^{end}	T_{pre} (s)	T_{pre}^{old} (s)	*Speedup*
1	C1	16 593	10	17 889	1 051	478	**2.2**
2	C4	17 754	10	22 938	2 892	440	**6.6**
3	C8	19 302	10	29 670	4 925	857	**5.7**
4	C16	22 398	10	40 830	14 387	3 059	**4.7**

3.2.3 Non-zero initial guess

The data reusing results presented in the previous section only concern the matrix construction phase (preparation phase). After this phase, the system is solved using an iterative solver (Flexible GMRES). In *MBEMv2.0* the initial guess is set to zero each time the solver is called. From one cycle to another, the system solution does not vary a lot. Although the number of unknowns increases, the new results (displacement and traction) of existing nodes are not too different from their previous results. So, starting from the second cycle, we develop an algorithm to use as initial guess for the iterative solver, the previous system solution. This leads to an important reduction of the number of iterations of the flexible GMRES. Table

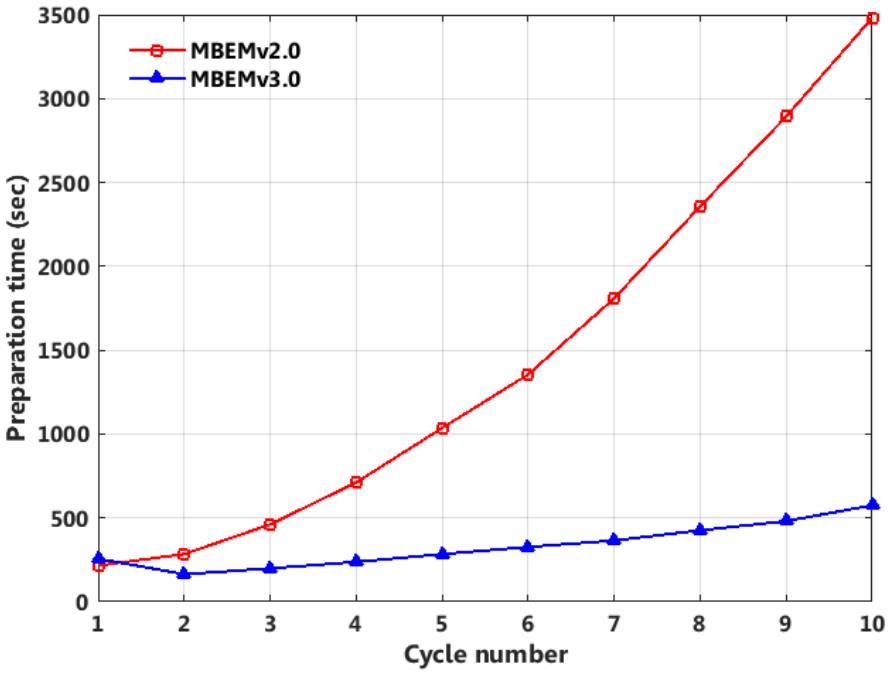


Figure 3.9 – Fast K_{near} computation: C16 result

3.5 presents for model C8, the reduction of the number of iterations at each cycle. For this example, the combined solution time for cycles 2 to 10 is reduced from 12 415 s to 3697 s. That represents a speedup of 3.4 of the solution time. It is clear that this solution reusing technique becomes very impressive when the number of degrees of freedoms is larger and even more while dealing with a few number of cracks.

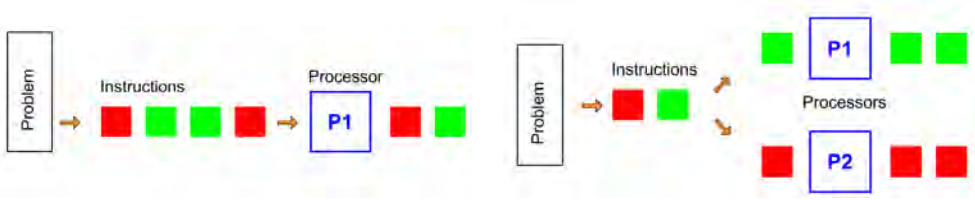
Table 3.5 – Non-zero initial guess: Results

# cycle	N_{dofs}	N_{iter}^{old}	N_{iter}^{new}	T_{sol}^{old}	T_{sol}^{new}
1	19 302	24	24	833	862
2	20 454	24	3	921	171
3	21 606	24	3	1037	205
4	22 758	24	4	1157	276
5	23 910	24	5	1180	379
6	25 062	24	5	1289	410
7	26 214	24	5	1384	449
8	27 366	24	6	1502	569
9	28 518	30	5	1963	542
10	29 670	28	6	1984	695

3.3 Parallel implementation

3.3.1 Parallel computing

Over recent decades, computers have evolved a lot. Parallel architecture machines have become standard. Parallel computing techniques can significantly increase the performance of existing serial codes. In a serial program, A problem is divided into a discrete series of instructions which are executed sequentially one after another on a single processor. Only one instruction may execute at any moment in time, see Fig. 3.10.a. Parallel computing is the simultaneous use of multiple computing resources to solve a large problem, , see Fig. 3.10.b. The aim is to solve the initial problem in the smallest possible time. The problem is divided into small problems that can be solved simultaneously. Each part is then solved using different computing resources. The technique used to achieve parallelization depends on the hardware. Several parallel computers exist: from a simple multi-core, to cluster computers using distribute computing, to for example the Worldwide LHC Computing Grid (WLCG), consisting of a grid-based computer network infrastructure incorporating over 170 computing centers in 42 countries in 2017. In this work, the hardware is a simple 20-core Intel Xeon computer with the possibility of hyper-threading and with 128 Go of memory (RAM) using non-uniform memory access (NUMA). A shared memory parallelization is achieved on this computer by using OpenMP, see Chandra et al. [79]. OpenMP is a widely used application program interface (API) for parallel computing on shared memory architecture. It simplifies writing multi-threaded applications by using compiler directives and library routines. With these directives, the programmer can focus mostly on the program and algorithms and less on the details of the computer system. Readers who are interested in parallel computing are refered to the book of Scott et al. [80] or that of Magoules et al. [81] or to Trobec et al. [82] for a review on shared memory parallelism using OpenMP.



(a) (b)

Figure 3.10 – (a) Serial program (b) Parallel program

3.3.2 Profiling and parallelization

Many researchers have used parallelization to accelerate the fast multipole boundary element method. In 1990, Greengard and Gropp [83] presented a parallel version

of the fast multipole method. Recently, Adelman et al. [83] in 2017, presented a fast multipole method/graphics processing unit-accelerated BEM for computational magnetics and electrostatics via the Laplace equation. Gu and Zsaki [84] developed a parallel computation of field quantities for BEM applied to stress analysis using multi-core central processing units (CPUs). Ptaszny [85] presented a parallel fast multipole BEM applied to computational homogenization. The goal here is to speed up the existing code (*MBEMv2.0*) by avoiding big changes. A simple observation of Trinh's [48] results (see table 3.6) shows that the solving phase is time-consuming. To reduce this duration, it is necessary to reduce the number of iterations or the duration of one iteration. With the non-zero initial guess presented in subsection 3.2.3, the number of iterations is already reduced in crack propagation. So here we focus on the reduction of the duration of one iteration. Code profiling shows that almost all the time of one iteration (more than 90%) is spent in the routine *Sub_MVP*. So the time-consuming parts of the routine *Sub_MVP* are parallelized by using OpenMP directives. As illustrated in Fig. 3.1, *Sub_MVP* consists of the computation of matrix vector product. Code profiling shows that $K_{far}.w_j$ is the most time-consuming part of *Sub_MVP* due to multiple imperfectly nested loops. In multizone configurations, the number of these deeply nested loops can reach 12:

$$\sum_{zone}^{nzones}\sum_{cell}^{ncells}\sum_{el}^{nel}\sum_{gauss}^{4,9,16}\sum_{node}^{4,8}\sum_{dir}^{3}\sum_{order}^{7}\sum_{M=-N}^{N}\sum_{A}^{3}\sum_{B}^{3}\sum_{J}^{3}\sum_{I}^{3} \tag{3.8}$$

This part of the code is completely reorganized in high performance computing (HPC) point of view and then parallelized. Loop invariants are identified and moved out of loops. Vectorization work is performed for the innermost loops. For this part, the parallelization of the first loop is not interesting due to the large amount of data that would be in private. Moreover, the problems considered here have a few number of zones. Thus, the second and the third outer loops are parallelized. In the second loop, the number of iterations is the number of octree cells *ncells*, while in the third it is the number of elements in a cell *nel*. These numbers vary from one problem to another and depend on the octree construction, especially on the parameter *max_elem*. A parametric nested parallelism is performed with OpenMP, see Algorithm 3 and Algorithm 4 for B_{tt} computation. The subroutines used in these Algorithm 3 can be described as follows:

- *CLEAN_MULTIPOLE()*: Reset FMM variables used in the matrix-vector computation.
- *UPWARD_1()*: Execute the upward pass of the FMM operation (input by the candidate vector).
- *DOWNWARD_1()*: Execute the downward pass of the FMM operation.
- *Sub_MVP_CELL(cell)*: Compute matrix-vector product for elements in the cell considered.

In Algorithm 4 *Preliminary_calculations_i()* routines permit to compute variables that are constants in interior loops.

The user can enable or disable one or both of the parallel loops and if enabled, the number of threads ($nth1$ and $nth2$) can be given. For small problems, the parameter *max_elem* can have a high value (100, 200, 1000, ...). *nel* is thus high and *ncells* is small. For these cases, the third parallel loop should be activated. For large scale problems, *max_elem* is generally limited (50, 30, 15,...) by the computer RAM memory. *nel* is thus small, *ncells* is high and the second parallel loop should be activated.

Algorithm 3 Nested parallelism: Sub_MVP()

```
1: for NAME_BODY = 1, NBODY do
       Prodcut matrix_vector of each zone
2:     VECT_XN=0.D0
3:     Call CLEAN_MULTIPOLE()
4:     Call UPWARD_1()
5:     Call DOWNWARD_1()

       Loop over cells computed in parallel
6:     !$OMP PARALLEL DO IF(par1.eq.1) SCHEDULE(dynamic)
7:     !$NUM_THREADS(nth1) PRIVATE(C)
8:     for cell = 2, Ncells do
9:         Call Sub_MVP_CELL(cell)
10:    end for
11:    !$OMP END PARALLEL DO

       Accumulate the products in VECT_X
12:    VECT_X(:)=VECT_X(:) + VECT_XN(:)
13: end for
```

3.3.3 Parallel efficiency

The speedup and the efficiency of the parallelization are shown in table 3.7 for the model C216 and for one iteration. Simulations are done on a 20-core Intel Xeon E5-2630v4 processor running at 2.2 GHz. After the acceleration of the solving phase, the preparation phase is also accelerated by the parallelization of the time-consuming part of subroutine *Sub_b_knear*. The speedup and the efficiency of the parallelization are similar to those of the solving phase (Table 3.7). Table 3.8 shows the global speedup and efficiency due to the parallel implementation for the model C216. The code can be parallelized more, but it will become more complex and extension work will be difficult. Although the code is not entirely parallelized, the parallelism results are very good.

Algorithm 4 Nested parallelism: Sub_MVP_CELL(CELL_NAME)

```
1: Call Preliminary_calculations_1()
2: !$OMP PARALLEL DO IF(par2.eq.1) SCHEDULE(dynamic)
3: !$NUM_THREADS(nth2) PRIVATE(− − −)
4: for e1 = 1, Nel do
5:     Call Preliminary_calculations_2()
6:     for gauss = 1, Ngauss do
7:         Call Preliminary_calculations_3()
8:         Call EVAL_RNM()
9:         for node = 1, Nnode do
10:            for dir = 1, 3 do
11:                Call Preliminary_calculations_4()
12:                Call Compute_Btt()          ▷ computed with vectorization
13:                !$OMP CRITICAL
14:                Call Save_Btt()             ▷ save results in VECT_XN
15:                !$OMP END CRITICAL
16:            end for
17:        end for
18:    end for
19: end for
20: !$OMP END PARALLEL DO
```

Table 3.6 – Bi-material cube with crack array by *MBEMv2.0* [48]

#	N_{dofs}	T_{pre} (s)	N_{iter}	T_{sol} (s)	T_{tot} (s)	T_{sol}/T_{tot} (%)
1	401 412	5 457	79	44 319	50 986	**87**
2	683 148	12 197	66	79 464	95 584	**83**
3	1 061 928	11 903	102	190 944	206 114	**93**

Table 3.7 – Parallelization: Efficiency (Sub_MVP)

N_{th}	T_{Sub_MVP} (s)	Speedup	Eff.(%)
1	221	–	–
2	113	2.0	**98**
4	64	3.5	**86**
8	34	6.6	**82**
12	24	9.4	**78**
16	18	12.0	**75**
20	17	13.3	**67**

Table 3.8 – Parallelization: Global Efficiency

N_{th}	T_{tot} (s)	Speedup	Eff.(%)
1	3 771	–	–
2	2 217	1.7	**85**
4	1 241	3.0	**76**
8	760	5.0	**62**
12	603	6.3	**52**
16	534	7.0	**44**
20	528	7.1	**36**

Maximum Speedup

To predict the theoretical speedup when using multiple processors in parallel computing, Amdahl's law [86] is often used:

$$Sp^{\max} = \frac{1}{1 - p + \frac{p}{N}} \tag{3.9}$$

where p is the portion of the code run in parallel, and N the number of processors. For example, if a program needs 20 hours using a single processor core, and a particular part of the program which takes one hour to execute cannot be parallelized, while the remaining 19 hours (p = 0.95) of execution time can be parallelized, then regardless of how many processors are devoted to a parallelized execution of this program, the minimum execution time cannot be less than that critical one hour. Hence, the theoretical speedup is limited to at most 20 times. Fig.3.11 presents a comparison between the obtained speedup of Table 3.8 with p=0.95 and the speedup limit given by Amdahl's law. The speedup limit for p=0.90 is also plotted on this figure. We can conclude that our results are acceptable, since Amdahl's law does not consider the additional costs of parallelization, typically communications and synchronizations. In parallel computing, the amount of data processed can increase. This is taken into account in the law proposed by Gustafson et al. [87].

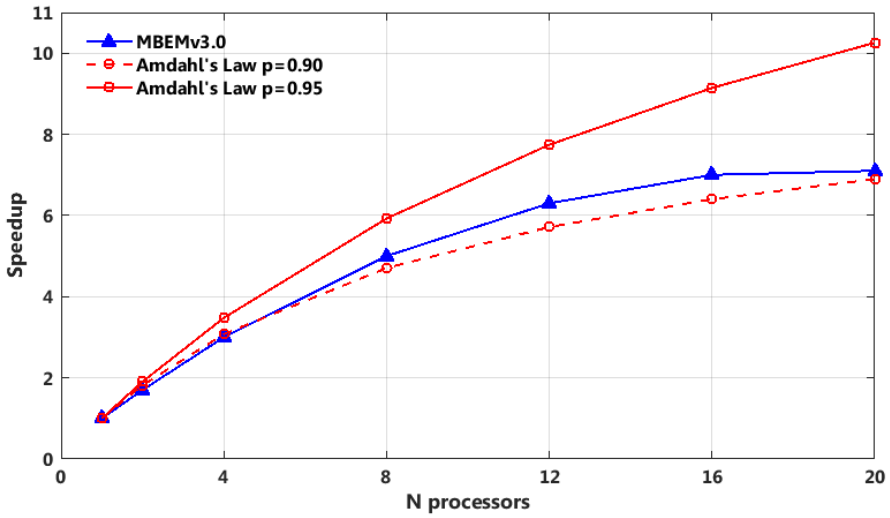


Figure 3.11 – Maximum Speedup

3.4 Upper bounded incremental coordinate method

3.4.1 Matrix storage

In FM-SGBEM algorithm, the matrix of near-interactions should be computed and stored in RAM memory before the resolution. Several storage technique exist for that purpose. The storing choice depends on how one uses and access to the matrix.

Dense format(DNS)

The simplest format is the dense format which is composed of n rows and n columns. This storage, can take advantage of the optimized BLAS routines to compute in-

stantaneously the matrix-matrix or matrix-vector operations, but it requires huge amount of memory and thus limit the size of the problems which can be treated.

Coordinate format (COO)

This algorithm requires three vectors to store the necessary information of the matrix. For each entry, the first vector saves the row index, the second the column index and the third the corresponding value.

Compressed Sparse Row (CSR)

Compressed Sparse Row (CSR) [76, 77] is a simple algorithm to store sparse matrices. This algorithm requires three vectors to store the necessary information of the matrix: The first vector (***AA***) collects the value of the non-zeros on the upper part of the matrix (including the diagonal). The second vector (***JA***) saves the column index of the corresponding term while the third one (***IA***) contains the information on the total number of non-zeros on each row of the matrix.

A simple example of the symmetric CSR is shown below. Matrix $[K]$ is symmetric and contains many zero terms. The symmetric CSR will scan the upper part of this matrix and extract all the non-zero coefficients for the storage. So the matrix can be converted to the vectors ***AA***, ***JA*** and ***IA*** below. The CSR is used in *MBEMv2.0* to store the matrix of near-interactions.

$$[\mathrm{K}]= \begin{pmatrix} 1 & 2 & 0 & 0 & -3 \\ 2 & -4 & 0 & 5 & 0 \\ 0 & 0 & 6 & -7 & 0 \\ 0 & 5 & -7 & 8 & 9 \\ -3 & 0 & 0 & 9 & 10 \end{pmatrix}$$

AA	1	2	-3	-4	5	6	-7	8	9	10
JA	1	2	5	2	4	3	4	4	5	5
IA	1	4	6	8	10	11				

3.4.2 UBI-COO method

In large scale simulations, memory usage requires special attention, especially in a context of parallel computing. In the initial code, the Compressed Sparse Row (CSR) format is used to store the matrices after computation, but dense format (DNS) is used before and during the computation (see subroutine *Sub_b_knear*). Using DNS for construction causes large allocated but not used memory. Since the construction is in parallel, dense format causes a peak in allocated memory. To avoid this, an upper bounded incremental coordinate method (UBI-COO) is designed for the construction of the matrices. Based on Sparsekit subroutines written by Saad [88], necessary subroutines for the manipulation of the matrices in COO or CSR format are written. The coordinate format (COO) is well-known for constructing sparse matrices.

A comparison between DNS and UBI-COO is presented in Fig.3.12. Using DNS is simple: a dense matrix is allocated and is converted to CSR format at the end of the construction. With the UBI-COO, the coordinate format is used in an incremental way. A parameter fixes the maximal number of data in the COO. When the limit is reached (Fig.3.12: step 1.6 and 2.6), the COO matrix is converted to CSR (step 1.7 and 2.7) and the CSR is accumulated with an existing CSR (step 1.8). At this step, multiple entries are also accumulated. The COO is then reinitialized for the rest of the construction. In fact, the number of non-zero is not known before the computation, so the dimension of the arrays which contains the coordinates and the non-zero values is not known. So, two parameters (*p1* and *p2*) are used to achieve incremental COO. The first is the initial dimension of the arrays and the second is the increment to re-size the arrays. Optimal value must be found for each parameter. While constructing a matrix, multiple entries often happen. Multiple entries can greatly increase the size of the COO arrays because each entry is saved independently. It is difficult to deal with multiple entries in the COO format while the matrix is in construction. So, a parameter (*p3*) is used. *p3* represents the maximum size of the COO arrays. When the COO size reaches *p3*, the COO matrix is converted to CSR and accumulated with a temporal CSR matrix and the COO arrays are reinitialized and then the multiple entries are accumulated in the temporal CSR matrix.

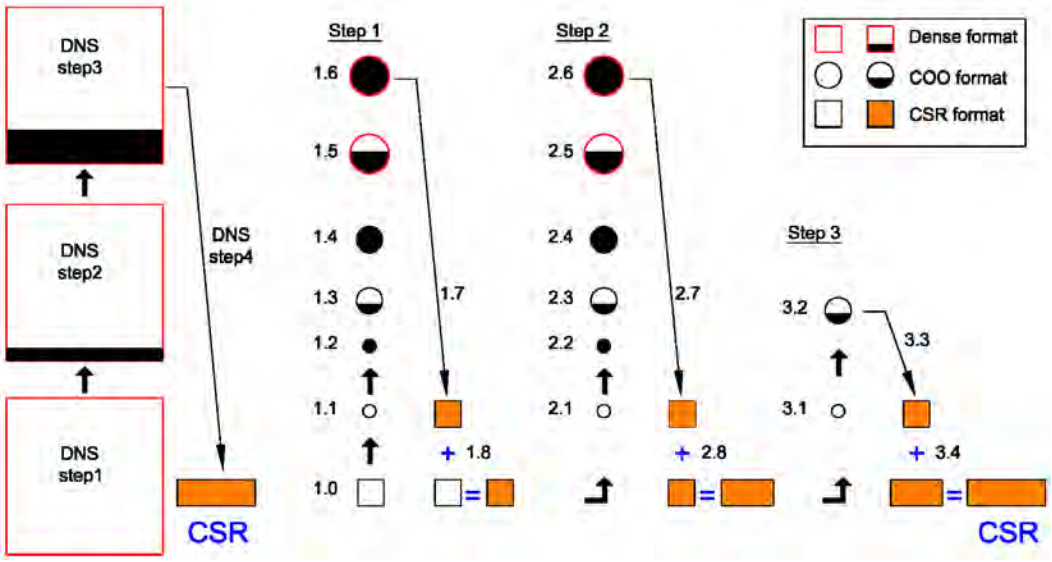


Figure 3.12 – DNS and UBI-COO principles

3.4.3 UBI-COO results

This upper bounded incremental coordinate method erases memory peaks. Figure 3.13 presents the virtual memory needed using DNS and UBI-COO during the

matrix computation for the model C8. There is no memory variation during the solution phase, so only one iteration is performed in order to focus on the preparation phase (matrix computation). It can be noticed that the maximum memory needed is greatly reduced. The small peaks observed can also be reduced by changing the parameters previously presented. It can also be noticed that the duration of the construction phase is reduced (for cycle 2 and 3) because less data is manipulated. The new method proposed can thus permit to simulate problems with more degrees of freedom.

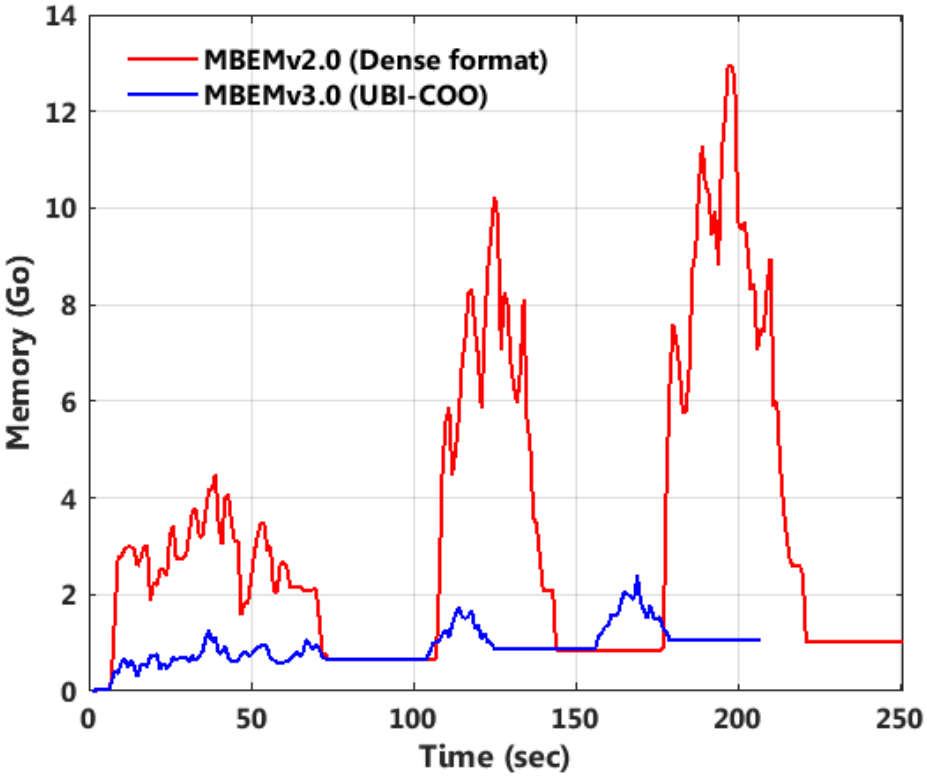


Figure 3.13 – UBI-COO results: 3 cycles with model C8

3.5 Performance tests

This section presents the results of all the optimizations presented in this chapter. The calculation times are measured on an Intel Xeon (20 cores, 2.2 GHz) computer with 128 Go of RAM. T_{tot} is the total time including pre-processing (input reading, octree construction, etc.) and post-processing (results writing in files). This duration is compared to *MBEMv2.0* [48] duration noted T_{tot}^{old}. Table 3.9 shows computational data for static analyses. Table 3.10 shows computational data for propagation analyses. For the cube with 8 cracks (model C8), the evolution of

the total time according to the cycle number is presented in Fig.3.14. In this last picture, the great performance of *MBEMv3.0* can be seen.

Table 3.9 – Static tests

#	Model	N_{dofs} (s)	T_{tot} (s)	T_{tot}^{old} (s)	*Speedup*
1	C8	19 302	169	1 469	**8.7**
2	C64	40 974	611	6 417	**10.5**
3	C1000	403 206	4 478	72 107	**16.1**
4	C1728	684 942	8 721	119 249	**13.7**
5	C1000	1 075 206	15 446	166 808	**10.8**
6	C8000	3 112 206	53 288	848 162	**15.9**

Table 3.10 – Propagation tests: Cube with crack array

#	Model	N_{dofs}^{init}	N_{cycles}	N_{dofs}^{end}	T_{tot} (s)	T_{tot}^{old} (s)	*Speedup*
1	C8	19 302	10	29 670	889	31 396	**35.3**
2	C64	40 974	10	123 918	8 217	432 214	**52.6**
3	C512	199 950	12	1 010 958	71 905	–	–
4	C1000	388 806	6	1 108 806	61 500	–	–

3.6 Conclusions

In this chapter *MBEMv2.0* performance is improved. First, a data reusing technique permitted to save useful matrix values and reuse them in the next cycle. That led to the reduction of the matrix computation phase. For the solution phase, non-zero initial guess is used for the iterative solver. The number of iterations is thus reduced and therefore the solution phase.

The second optimization is the parallel implementation. Time-consuming parts of the code are identified, then reorganization work is performed and finally a shared memory parallelism is achieved using OpenMP directives. The parallel results on a 20-core computer are very good. Despite all, the code is not parallelized. On a parallel part, a speedup of 13.3 is obtained while using 20 threads.

The parallel implementation of the matrix computation phase caused peaks of memory use. To solve this problem, a new sparse matrix method is designed based on coordinate format and compressed sparse row format. This new method erases memory peaks during the matrix construction phase. The duration of this phase is also reduced because less data is manipulated.

Together, these optimizations radically change code performance, specially for crack propagation problems for which a speedup of 52.6 is obtained for the simulation of 64 cracks in a homogeneous domain. *MBEMv2.0* required 5 days to simulate this problem while *MBEMv3.0* only needs 2 hours and 17 minutes. Crack propagation until one million degrees of freedom is also achieved in 17 hours while *MBEMv2.0* would require more than a month.

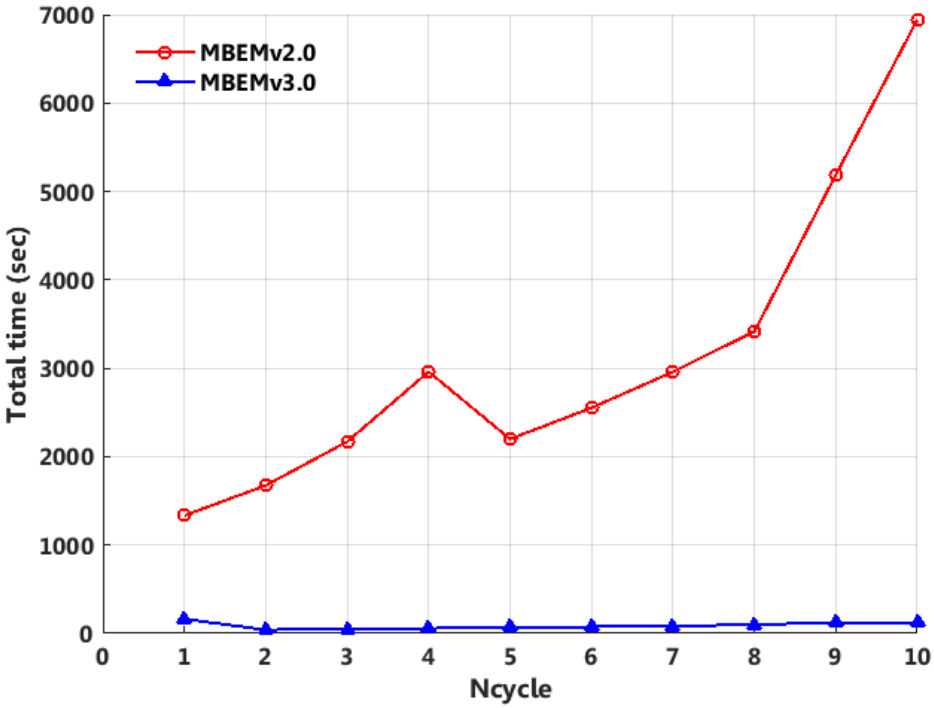


Figure 3.14 – Evolutions of total time

Chapter 4

Refinements and investigations

Contents

MBEMv2.0 has several limitations: complex multizone problems can not be simulated, the crack re-meshing algorithm does not follow any law, the cracks treated can not intersect a surface or an interface, etc. In this chapter, various extension works are introduced. An extension to complex multizone simulation will be proposed. Then we will discuss propagation laws and re-meshing algorithm. In the rest of the chapter we will discuss surface breaking crack problems, their simulation and their propagation. The proposed method will also be extended to cracks which start at an interface. In the last section, some investigations on contact problems will be considered.

4.1 Complex multizone problems

4.1.1 Multizone domains

Multizone problems refer to the treatment of a domain containing different materials separated by internal interfaces. These problems can be seen in many practical applications: composite materials, geomechanical systems, study of fractures, etc. At the common boundary between two sub-domains (interface), the corresponding full matching behaviors must be enforced. Sliding contact is not treated. Interested readers may refer to the work of Phan et al. [89] or Kuo [90]. One important aspect in a multi-domain algorithm is to enforce the continuity and equilibrium conditions at interfaces. The numerical implementation of multizone problems in *MBEMv2.0* is limited to domains whose sub-domains are superimposed vertically, see Fig. 4.1. This limitation is due, among others, to the difficulty to identify for each zone, the elements on the boundary surfaces or interfaces and their orientations. To circumvent this difficulty, several rules are imposed in *MBEMv2.0*, even for very simple n zones geometries:

- the normal of the outer boundary of the domain must be oriented outward.
- the normal of the interfaces must be oriented from bottom to top.
- zones are systematically numbered from 1 to n, from bottom to top.
- interfaces are systematically numbered from 1 to n-1 from bottom to top.

It is clear that this implementation is very restrictive but especially becomes impossible for the simulation of complex multizone problems. As a result, several real problems like composite slab shown in Fig. 4.2, can not be simulated. In this section, the implementation is extended to a generic multizone problem as presented in Fig. 4.3 for three zones.

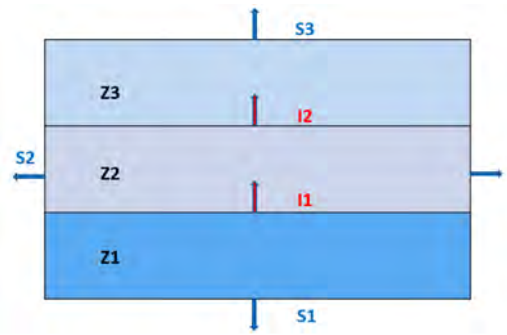


Figure 4.1 – A 3-zone domain

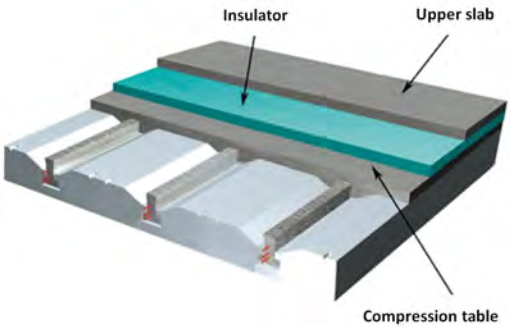


Figure 4.2 – A composite slab

4.1.2 Symmetric Galerkin formulation

SGBEM provides a symmetric system matrix but when applied to the sub-domains, this property cannot be completely achieved. In order to conserve the global sym-

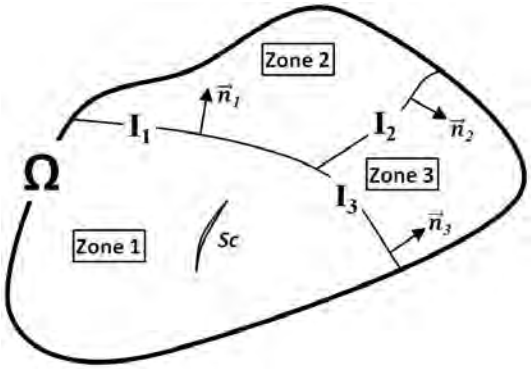


Figure 4.3 – A multizone fractured domain

metry of the method, an appropriate technique must be adopted during the construction of the matrices. Ganguly et al. [25] introduced an algorithm that can lead to a partly symmetric matrix by putting the unknowns on the interface ahead. The block matrices corresponding to interfaces are non-symmetric, while the rest are symmetric. In [56], Gray and Paulino have studied a fully symmetric Galerkin BEM in heat transferring. This method is based on an appropriate combination of usual SGBEM equations on interfacial and non-interfacial boundaries. This technique is later adopted in elastostatics [74] and fracture mechanics [91]. In this work, the approach described in [56] by Gray and Paulino has been exploited. Perfect bonding between sub-domains is assumed first, imposing the continuity of displacement and the equilibrium of tension across the interface. Via some appropriate term rearrangement and sign adoptions, the symmetry of the global matrix can be achieved.

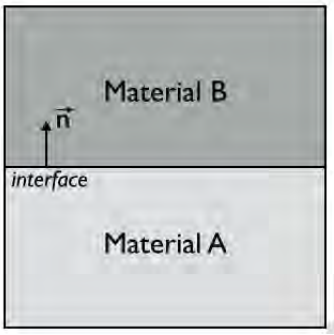


Figure 4.4 – A bi-material interface problem

A simpler geometry is used here to describe the multizone formulation. The extension to more complicated interface problems would follow analogous principles. For instance, a geometry having a single common boundary is employed (Fig. 4.4). The solid contains two materials A and B located at the bottom and top regions

respectively. The interface $S_i = I_A = I_B$ has the normal vector oriented from solid A toward B. There are two sets of unknowns related to the two sub-domains:

$$u^A, t^A, u^{I_A}, t^{I_A} \text{ and } u^B, t^B, u^{I_B}, t^{I_B} \tag{4.1}$$

where u^A, t^A, u^B, t^B and $u^{I_A}, t^{I_A}, u^{I_B}, t^{I_B}$ are non-interface and interface unknowns of A and B respectively.

The basic idea is to write the usual Galerkin equations on all boundaries of each sub-domain. It is then convenient to write the double integral terms of these equations in the block-matrix format:

For zone A:

$$\left[\begin{array}{cc|cc} B_{uu}^{AA} & B_{tu}^{AA} & B_{uu}^{I_AA} & B_{tu}^{I_AA} \\ B_{ut}^{AA} & B_{tt}^{AA} & B_{ut}^{I_AA} & B_{tt}^{I_AA} \\ \hline B_{uu}^{AI_A} & B_{tu}^{AI_A} & B_{uu}^{I_AI_A} & B_{uu}^{I_AI_A} \\ B_{ut}^{AI_A} & B_{tt}^{AI_A} & B_{ut}^{I_AI_A} & B_{uu}^{I_AI_A} \end{array}\right] \left(\begin{array}{c} u^A \\ t^A \\ u^{I_A} \\ t^{I_A} \end{array}\right) = \left(\begin{array}{c} \mathcal{F}_u(\tilde{u}^A) \\ \mathcal{F}_t(\tilde{t}^A) \\ \mathcal{F}_u(\tilde{u}^{I_A}) \\ \mathcal{F}_t(\tilde{t}^{I_A}) \end{array}\right) \tag{4.2}$$

and zone B:

$$\left[\begin{array}{cc|cc} B_{uu}^{BB} & B_{tu}^{BB} & B_{uu}^{I_BB} & B_{tu}^{I_BB} \\ B_{ut}^{BB} & B_{tt}^{BB} & B_{ut}^{I_BB} & B_{tt}^{I_BB} \\ \hline B_{uu}^{BI_B} & B_{tu}^{BI_B} & B_{uu}^{I_BI_B} & B_{tu}^{I_BI_B} \\ B_{ut}^{BI_B} & B_{tt}^{BI_B} & B_{ut}^{I_BI_B} & B_{tt}^{I_BI_B} \end{array}\right] \left(\begin{array}{c} u^B \\ t^B \\ u^{I_B} \\ t^{I_B} \end{array}\right) = \left(\begin{array}{c} \mathcal{F}_u(\tilde{u}^B) \\ \mathcal{F}_t(\tilde{t}^B) \\ \mathcal{F}_u(\tilde{u}^{I_B}) \\ \mathcal{F}_t(\tilde{t}^{I_B}) \end{array}\right) \tag{4.3}$$

the upper scripts indicate the surfaces on which the integrals are written, for instance:

$$B_{tu}^{I_AA} = \int_{S_{tA}} \int_{I_A} t_i^{I_A}(\mathbf{x}) T_i^k(\mathbf{x}, \tilde{\mathbf{x}}) \tilde{u}_k(\tilde{\mathbf{x}}) dS_x dS_{\tilde{x}}$$

The continuity conditions $u^{I_A} = u^{I_B} = u^{S_i}$ and $t^{I_A} = -t^{I_B} = t^{S_i}$ are then embedded in the above systems in an appropriate way. The interface tension of the top region (B) is replaced by the negative of the bottom interface tension (A). The equation (4.3) is thus transformed to:

$$\left[\begin{array}{cc|cc} B_{uu}^{BB} & B_{tu}^{BB} & B_{uu}^{S_iB} & -B_{tu}^{S_iB} \\ B_{ut}^{BB} & B_{tt}^{BB} & B_{ut}^{S_iB} & -B_{tt}^{S_iB} \\ \hline B_{uu}^{BS_i} & B_{tu}^{BS_i} & B_{uu}^{S_iS_i} & -B_{tu}^{S_iS_i} \\ -B_{ut}^{BS_i} & -B_{tt}^{BS_i} & -B_{ut}^{S_iS_i} & B_{tt}^{S_iS_i} \end{array}\right] \left(\begin{array}{c} u^B \\ t^B \\ u^{S_i} \\ t^{S_i} \end{array}\right) = \left(\begin{array}{c} \mathcal{F}_u(\tilde{u}^B) \\ \mathcal{F}_t(\tilde{t}^B) \\ \mathcal{F}_u(\tilde{u}^{S_i}) \\ -\mathcal{F}_t(\tilde{t}^{S_i}) \end{array}\right) \tag{4.4}$$

From (4.2) and (4.4), the global matrix can be easily constructed by linear combination as:

$$\left[\begin{array}{c|c|c} [SG]_{AA} & [SG]_{S_iA} & 0 \\ \hline [SG]_{AS_i} & [SG]_{S_iS_i} & [SG]_{BS_i} \\ \hline 0 & [SG]_{S_iB} & [SG]_{BB} \end{array}\right] \left(\begin{array}{c} u^A \\ t^A \\ u^{S_i} \\ t^{S_i} \\ u^B \\ t^B \end{array}\right) \tag{4.5}$$

Block $[SG]_{\mathcal{X},\mathcal{Y}}$ corresponds to the Symmetric Galerkin equations written for the surfaces $\mathcal{X}$ and $\mathcal{Y}$ respectively. The diagonal blocks (1,1) and (3,3) are symmetric as a consequence of the SG procedure. The blocks (1,3) and (3,1) are zero since the top and bottom equations are not related. The pairs of off-diagonal blocks (1,2) = $(2,1)^T$, (2,3) = $(3,2)^T$ (T indicating the transpose) also result from the SGBEM procedure. The block (2,2) is a linear combination of the SG equations for interface of top and bottom materials. There are single integral terms embedded in this block that are locally non-symmetric. Due to the change of sign across the interface and the material-independence property, these integrals drop out and leave only the double integral terms that are all symmetric in the global system. Eventually, the global matrix is symmetric and is also of reduced size since only one set of unknowns is invoked from the interface.

4.1.3 Multizone algorithm

The multizone algorithm can be summarized in a few steps, see Algorithm 5: (a) an *octree* structure is constructed, covering the whole solid. (b) a loop is set on all bodies. Near interactions are computed then stored locally in $[K_{near}]$. (c) an iterative solver (GMRES) is used to approximate the solution. As GMRES requires a global matrix-vector multiplication, a second loop is called and takes care of the product in a block-matrix manner. An example is shown in Fig. 4.5: Zone-i shares two interfaces with zone i-1 and zone i+1. (1) We take out the part of the global candidate vector which corresponds to the unknowns of this zone. (2) This local vector is used at first in fast multipole evaluations, then it is multiplied with the near coefficients which are already stored in step (a). The sum of these two operations forms the product of zone-i with the candidate vector. (3) This product is then returned to the global coordination and the next zone is studied. (4) By accumulating all these local products, we eventually obtain the global matrix-vector product for the iterative solver. After the convergence is achieved, the post-processing does not differ from the case of single domain.

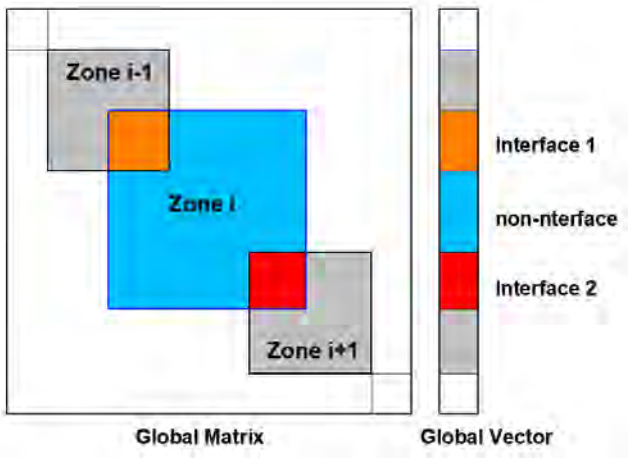


Figure 4.5 – Block matrices in Multizone problem

Algorithm 5 Multizone FM-SGBEM

1. Import Geometries and Parameters
2. Build octree
3. Compute known terms
- **Do** i = 1,nbody *(Loop on all bodies)*
  Call <upward>$_i$ → <downward>$_i$ → <expansion>$_i$
  → compute $\{b\}_i$ → $\{b\}_{global} := \{b\}_{global} + \{b\}_i$
  → compute and store $[Knear]_{zone\ i}$
- **EndDo**

4. Iterative Solution
Outer GMRES
- Global Matrix-Vector multiplication
  - **Do** i = 1,nbody *(Loop on all bodies)*
    Call <upward>$_i$ → <downward>$_i$ → <expansion>$_i$
    → $[K]_{zone\ i}\{x\}_i = [Knear]_{zone\ i}\{x\}_i + [K^{FMM}]_{zone\ i}\{x\}_i$
    → store $[K]\{x\} := [K]\{x\} + [K]_{zone\ i}\{x\}_i$
  - **EndDo**
- Preconditioning Task Inner GMRES
  - **Do** i = 1,nbody *(Loop on all bodies)*
    → $[Knear]\{w\} := [Knear]\{w\} + [Knear]_{zone\ i}\{w\}_i$
    → inner preconditioning
  - **EndDo**

5. Post-Processing

Treatment of complex configurations

Let us consider a 4-zone domain shown in figure 4.6. In *MBEMv3.0*, the user can give the list of elements on each interface or simply give the geometric characteristics of the interfaces (for example: $z = 0$ and $x < 0$). The orientation of the boundary surfaces and interfaces is free. Boundary surfaces and the numbering of the interfaces are free. To identify the boundary surfaces and interfaces of each zone, a two-dimensional ($N_{zones} \times N_{faces}$) array noted *BodyOut* is used. For each zone i, and each surface j (boundary surface or interface), the value of $BodyOut(i, j)$ is set to 1 if j belongs to zone i and is oriented outward, -1 if j belongs to zone i and is oriented inward, 0 if j do not belong to zone i. An example of numbering is shown in figure 4.6 and the corresponding *BodyOut* is presented in table 4.1. With this free numbering algorithm, later multizone treatments (interface inverting, determination of the sign of the Galerkin formulation terms, etc.) are simplified.

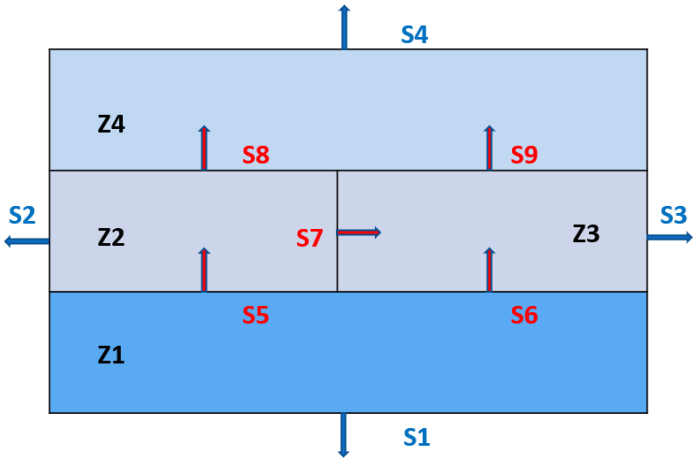


Figure 4.6 – A 4-zone domain

Table 4.1 – *BodyOut* array for a 4-zone domain

#	S1	S2	S3	S4	S5	S6	S7	S8	S9
Z1	1				1	1			
Z2		1			-1		1	1	
Z3			1			-1	-1		1
Z4				1				-1	-1

But everything is not simplified in complex configurations. In the multizone formulation presented for two sub-domains, continuity conditions are prescribed on the interface of the bi-material considered : $u^{I_A} = u^{I_B} = u^{S_i}$ and $t^{I_A} = -t^{I_B} = t^{S_i}$. These conditions hold for the three-zone domains shown in Fig. 4.1. But when intersections of interfaces appear as in Fig. 4.3 (intersection of I_1, I_2 and I_3) or

Fig. 4.6 (intersection of S5, S6 and S7 and intersection of S7, S8 and S9), the condition on tensions is no longer valid at those interface intersections. Let us consider the intersection between S5, S6 and S7. In fact, there are two tension unknowns to be determined at the corner point where the three interfaces meet. Indeed, the orientation of the normal vector of interface S7 is different from the orientation of the normal of S5 (or S6). So the two tension unknowns are: t^{S5} and t^{S7}. For these problems, after writing the Galerkin formulation as described in the previous chapter, a double node technique is used at interface intersections.

Double node technique

This technique is detailed in Sutradhar et al. [92] to treat corner and edge problems. We briefly present two dimensional corner treatment by double node technique. The corner is represented by a double node pair, one node for each side, the coordinates of the two nodes being the same. The flexibility provided by the choice of weight function in the outer integration of Galerkin formulation is then exploited, see [93]. Each node in the pair has its own weight function, which is non-zero only on its side of the corner. With the weight functions defined, the Galerkin equations are first assembled as usual, and then adjusted appropriately depending upon the boundary conditions. There are three possibilities illustrated in Fig. 4.7. First (Dirichlet

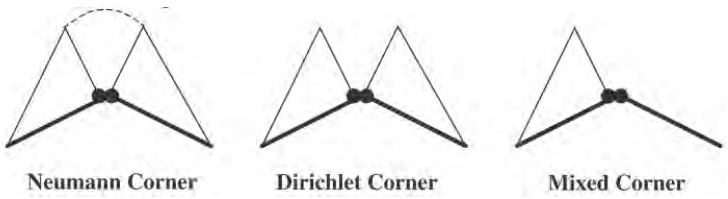


Figure 4.7 – Corner treatment with double node technique

corner), if the displacement is specified, then there are two tension values to be determined, and no further manipulation is required. Galerkin provides two independent equations for solving for the two (different) unknown normal derivatives. If tension is specified on one side and displacement on the other (Mixed corner), then the only unknown is the tension at the Dirichlet node, as the displacement must be continuous at the corner. In this case the equation at the Neumann node can be ignored. Finally (Neumann corner), if tension is specified on both sides, then the single unknown is the displacement, which is the same on both sides. Hence, the two equations should be added, and now the weight function spans both sides of the corner.

So, this technique permits to obtain two independent equations for solving the two different tension unknowns t^{S5} and t^{S7}. The techniques can also be extended to a multiple node technique for more complex configurations.

4.1.4 Validation

For validation purposes, a simple example of a 7-zone problem shown in figure 4.8 is studied. This example involves a domain with a fixed base and subjected to a vertical tension p on the opposite side. The material properties are the same for all the sub-domains and are taken so that the theoretical vertical displacement on the top is 1. The compute vertical displacement is shown in figures 4.9 and 4.10. Although the validation example shown here is simple, it is important to note that the extension work achieved here allows to simulate any complex multizone configuration with various material properties.

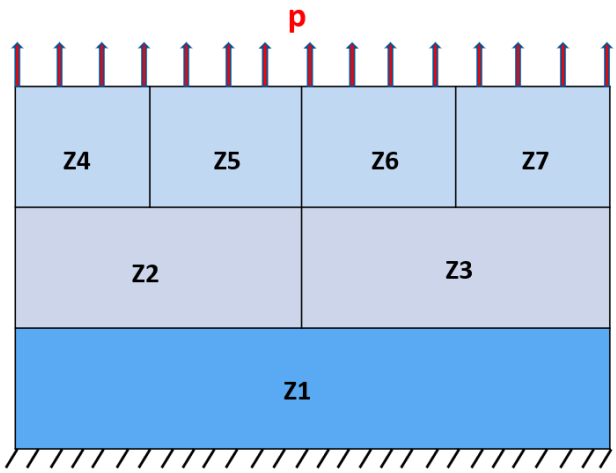


Figure 4.8 – A 7-zone problem

4.2 Crack propagation laws

4.2.1 Variation of crack-growth rate

Suitable criteria for crack propagation are still being debated, especially for 3D configurations. The criterion for fracture propagation is usually given either by conventional energy approach which states that a fracture propagates when the energy release rate reaches a critical value related to material fracture toughness or by the stress intensity approach which states that a fracture propagates when the stress intensity factor at the tip exceeds the material toughness. The SIF approach is used here. The local configuration of the crack front is described as in Fig. 4.11. The geometrical advance of the crack is described by moving points of the crack front in the local $(\boldsymbol{\nu}(s), \boldsymbol{n}(s))$ plane orthogonal to the front. The direction and length of the local crack advancement are represented respectively by the angle $\theta_0(s)$ and the step $\Delta a(s)$. The determination of these values is the main objective of a propagation law.

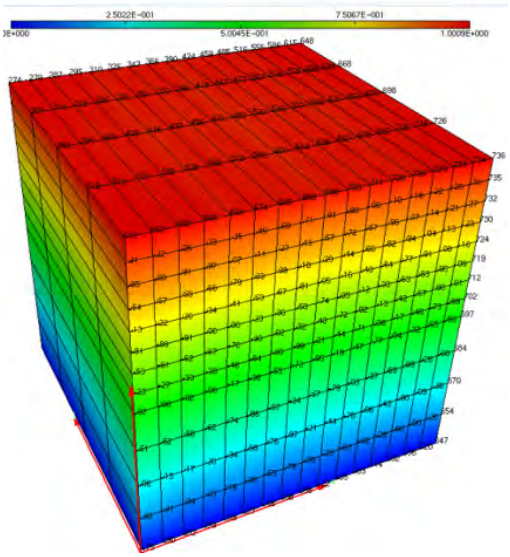


Figure 4.9 – 7-zone problem: vertical displacement

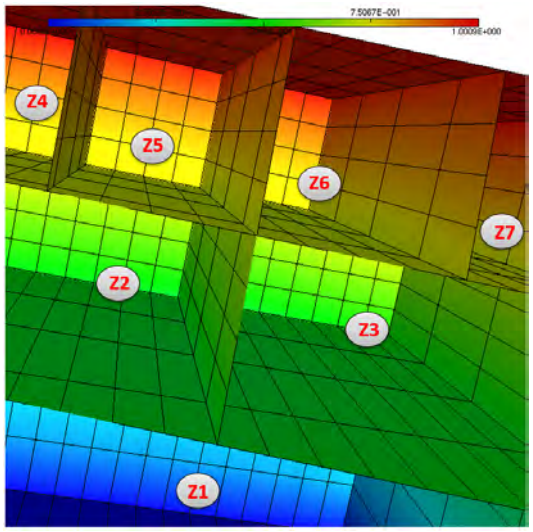


Figure 4.10 – 7-zone problem: vertical displacement (Zoom in)

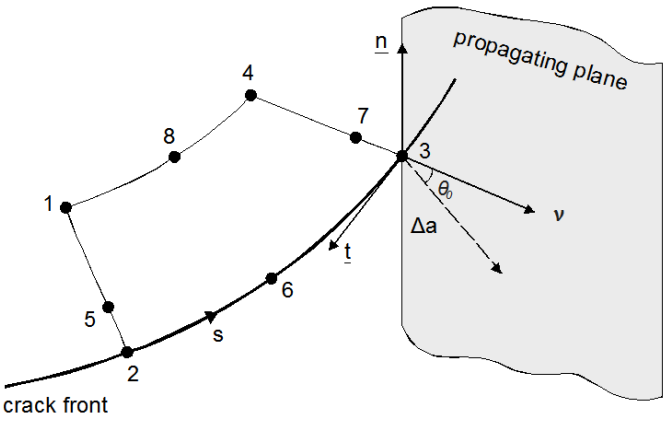


Figure 4.11 – Quarter-point element at the crack front. Vectors $\boldsymbol{\nu}, \boldsymbol{n}, \boldsymbol{t}$ constitute the local coordinate frame at node **3**. Their directions respectively correspond to the opening, sliding and tearing local modes of fracture propagation.

In experiments, crack propagation has been measured as a function of the stress intensity factor, a schematic representation is shown in Fig. 4.12. In this figure, the growth rate and the stress intensity range are plotted on a log-log scale. There exists a threshold value of ΔK below which fatigue cracks will not propagate. At the other extreme, $K_{\max}$ will approach the fracture toughness K_C, and the material will fail. Note that ΔK depends on several factors like the crack size. This is not shown in this figure. For small values of ΔK the propagation is difficult to predict. It depends on micro-structure and flow properties of the material and the growth can come to a stop. The growth rate is also dependent on to the size of the grains.

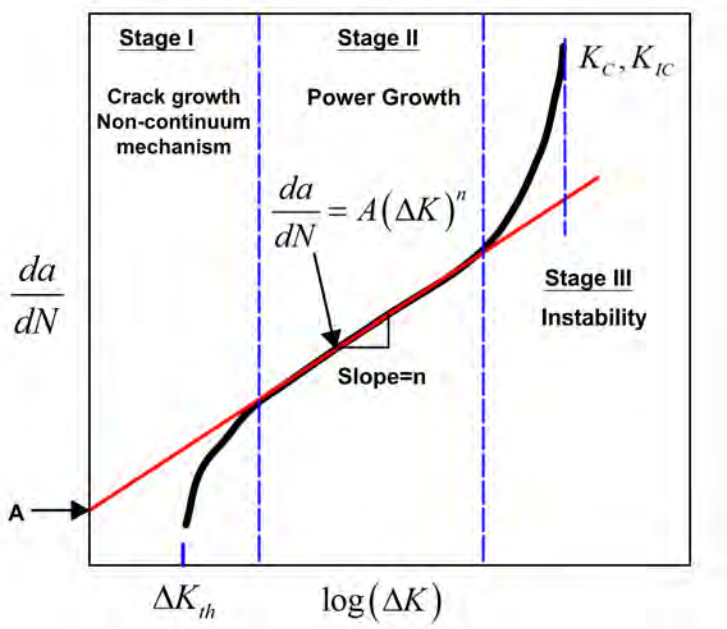


Figure 4.12 – Variation of crack-growth rate

In region III, the crack growth rate accelerates and finally a fracture will occur. The behavior of this fracture is also dependent on the micro-structure and flow properties of the material. In region II, a linear relationship can be written between $\log\left(\frac{da}{dN}\right)$ and ΔK. The crack growth rate is then governed by a power law. The crack growth rate is fairly insensitive to the micro-structure, however, the constants m and A are, of course, different for different materials. The fatigue life can be directly estimated by the power law if region II includes its dominating part.

Crack propagation equations

Many propagation laws are proposed in the literature, see Beden et al. [94] for a review of fatigue crack propagation models. They are based on the relationship between the stress intensity factor and the crack advance. In following, $\sigma_{\max}$ is the maximum value of applied stress and $\sigma_{\min}$ is the minimum value of applied stress for the cyclic loading considered. The stress ratio R, commonly called the R-ratio is defined as:

$$R = \frac{\sigma_{\min}}{\sigma_{\max}} = \frac{K_{\min}}{K_{\max}}. \tag{4.6}$$

where $K_{\min}$ and $K_{\max}$ correspond to $\sigma_{\min}$ and $\sigma_{\max}$ respectively.

The simplest propagation law is the Paris law [95] which represents the fatigue crack growth in the linear region, see equation 4.7 where C and m are material constants. C (A on Fig. 4.12) is an intercept constant and m is the slope on a

log-log scale.

$$\frac{da}{dN} = C \cdot \Delta K^m \tag{4.7}$$

The main limitation of the Paris law is its inability to consider the stress ratio R. As the stress ratio R increases, the crack growth rate in a material also increases, and *vice versa*. Increasing R has the effect of shifting the crack growth rate up, but it does not affect the slope of the growth rate curve. So, R has no effect on m but affect the intercept constant C. The effect of stress ratio R is taken into account by Walker [96], see equation 4.8 where C, m and γ are constants. C_0 is the intercept constant C for the case where stress ratio $R = 0$. The γ term indicates how strongly the stress ratio R affects crack growth rate in the material.

$$\frac{da}{dN} = C_0 \left[\frac{\Delta K}{(1-R)^{1-\gamma}} \right]^m \tag{4.8}$$

Forman [97] improved the Walker model by suggesting a new model, which is capable of describing the instability of the crack growth when the SIF approaches its critical value K_C (Stage III) and includes the stress ratio effect, see equation 4.9.

$$\frac{da}{dN} = \frac{C \cdot \Delta K^m}{(1-R)\, K_C - \Delta K} \tag{4.9}$$

Further modifications of the Forman equation to represent all the stages, have been accomplished by including the threshold stress intensity parameter ΔK_{th}, see equation 4.10 proposed by Hartman and Schijve [98].

$$\frac{da}{dN} = \frac{C \cdot (\Delta K - \Delta K_{th})^m}{(1-R)\, K_C - \Delta K} \tag{4.10}$$

Another related development has led to NASGRO equation 4.11, see [99]. It accounts for stress ratio, crack closure, and the tails at the upper and lower ends of growth rate curve.

$$\frac{da}{dN} = C_0 \left[\left(\frac{1-f}{1-R} \Delta K \right)^m \right] \frac{\left(1 - \frac{\Delta K_{th}}{\Delta K}\right)^p}{\left(1 - \frac{K_{\max}}{K_C}\right)^q} \tag{4.11}$$

The crack opening function, f , for plasticity-induced crack closure has been defined by Newman [100]. The values p and q are empirical coefficients that determine the curvature of the growth rate curve in the tail regions. Their values are selected to fit the growth rate curve to experimental data. The coefficient p controls the curve in the low growth rate (threshold) region, and q controls the curve in the high growth rate region.

A recent model for fatigue crack propagation is proposed by Castillo et al. [101] and modified by Blasón et al. [102] to take into account crack closure effects. The model uses normalized variables and parameters are computed by the least-squares technique.

As models become more sophisticated, the number of parameters increases, their identification becomes difficult and the implementation in a crack propagation code can become difficult too.

Paris law

Due to the simplicity of its implementation, Paris law is still very popular in crack propagation simulation. Mi and Aliabadi [103] used Paris law to simulate crack propagation with the collocation BEM. Bouchard et al. [104] used it in finite element method. The law is also used by Ma [105] to analyse mixed mode crack in a welded specimen. Paris postulated that sub-critical crack growth under fatigue loading can be predicted in terms of the ranges of (SIFs). Abundant experimental evidence supports the view that the crack growth rate can be correlated with the cyclic variation in the SIFs, e.g. through

$$\frac{\mathrm{d}a}{\mathrm{d}N}(s) = A \cdot \Delta K^m(s) \tag{4.12}$$

where s is the arc length coordinate along the crack front ∂S_c, N is the current number of loading cycles, $\mathrm{d}a/\mathrm{d}N$ is the fatigue crack advancement rate per cycle, $\Delta K(s) = K^{\max}(s) - K^{\min}(s)$ is the SIF range for the current cycle, while A and m are parameters that depend on the material, environment, frequency, temperature and stress ratio. In this work, Paris law is then used.

Evaluation of $\theta_0(s)$ and $\Delta a(s)$

In this work, the angle $\theta_0(s)$ is assumed to be given by the maximum circumferential stress criterion, see Erdogan and Sih [106]. This criterion is also used by Frangi in [41].

$$\tan\frac{\theta_0}{2} = \frac{1}{4}\left(\frac{K_{\mathrm{Ieff}}}{K_{\mathrm{II}}} - sign(K_{\mathrm{II}})\sqrt{\left(\frac{K_{\mathrm{Ieff}}}{K_{\mathrm{II}}}\right)^2 + 8}\right), \tag{4.13}$$

where $K_{\mathrm{Ieff}} = K_{\mathrm{I}} + B|K_{\mathrm{III}}|$ is an "effective" or "equivalent" local mode I stress intensity factor which accounts for the tearing mode being active ($K_{\mathrm{III}} \neq 0$), B being a material parameter. The local geometrical advancement Δa along $\cos\theta_0(s)\boldsymbol{\nu} + \sin\theta_0\boldsymbol{n}$ is determined from the Paris law (4.12) and by taking into account mixed mode:

$$\Delta a(s) = A\big(\Delta K_{\mathrm{I}}^2(s) + \Delta K_{\mathrm{II}}^2(s)\big)^{m/2}\Delta N. \tag{4.14}$$

SIF evaluation

In section 2.3.2 quarter-point elements are used to evaluate the stress intensity factors. The SIFs are hence evaluated through extrapolation from the displacement discontinuity field expressed in a local coordinate system. K_I for example, is evaluated from the nodal values of $\Delta\mathbf{u}^5$ and $\Delta\mathbf{u}^1$ (see Fig. 2.6).

$$\begin{aligned} K_I^2 \quad &= \lim_{\mathbf{d}\longrightarrow 0} \tfrac{\mu}{4(1-\nu)}\left(\tfrac{2\pi}{\mathbf{d}}\right)^{1/2}\Delta u_n \\ &= \tfrac{\mu}{4(1-\mu)}\left(\tfrac{2\pi}{a}\right)^{1/2}\left[2\Delta\mathbf{u}_n^5 - \tfrac{1}{2}\Delta\mathbf{u}_n^1\right] \end{aligned} \tag{4.15}$$

One can alternatively use only the COD value at the quarter-point node, whereby K_I at the node 2 is evaluated as:

$$\begin{aligned} K_I^2 &= \lim_{\boldsymbol{d}\to 0} \frac{\mu}{4(1-\nu)}[\frac{2\pi}{\boldsymbol{d}}]^{\frac{1}{2}}\Delta u_n \\ &= \frac{\mu}{4(1-\nu)}[\frac{2\pi}{\boldsymbol{d}}]^{\frac{1}{2}}\Delta \boldsymbol{u}_n^2 \end{aligned} \tag{4.16}$$

4.2.2 Re-meshing algorithm

At this stage based on crack opening displacements, the SIFs are evaluated with one of the two proposed extrapolation formulas. The SIFs permit to evaluate the propagation angle $\theta_0(s)$ and the crack advancement $\Delta a(s)$. Thus, everything is ready to numerically propagate the crack (re-meshing). We discuss how the re-meshing is done in this subsection.

In fact, when treating the Paris law in explicit fashion, two possibilities arise: either (i) set ΔN and deduce Δa or (ii) set Δa and deduce ΔN. In *MBEMv2.0* the propagation length was simply fixed for all the crack nodes, no laws were followed. Choice (i) may produce a crack increment Δa that is too-large if ΔN is inappropriately set, leading to numerical inaccuracies and significant re-meshing work. We thus prefer to follow the approach (ii), by fixing *a priori* the maximum propagation length $\Delta a^{\max}$, finding the node(s) where $\Delta K(s)$ is largest, evaluating the corresponding ΔN from (4.12) at that node, and then computing the extensions $\Delta a(s^i)$ by applying (4.12) at all crack front nodal positions s^i, see Algorithm 6. In some rare complex cases, some values of $\Delta a(s^i)$ can be too small. This situation also leads to numerical complications. Indeed, the elements may have a too long side, leading to significant errors when calculating integrals. This can also lead to convergence problems during the resolution phase. To avoid this, a minimum propagation length $\Delta a^{\min}$ is also defined. The minimum value is then used for these special nodes and the corresponding Δn^i is evaluated. When these cases occur, the local number of cycles Δn^i is computed for all the nodes and can be plotted as a graph.

The obtained values of $\theta_0(s^i)$ and $\Delta a(s^i)$ are then used to define new quarter-point elements extending the current crack. At crack front nodes s^i shared by two elements, quantities such as $\theta_0(s^i)$ relative to each element may not coincide exactly due to the discontinuity of $\boldsymbol{\nu}$ at $s = s^i$. In such cases, the adjacent values are averaged. The quarter-points of formerly frontal elements are moved to the middle of the element side while the crack mesh is updated by adding a row of quarter-point elements, see Fig. 4.13. The incremental process is repeated until the final number of loading cycles is reached.

4.2.3 Fatigue life of cracked cylinder

The lifetime of a cracked domain is typically expressed as the number of cycles that it takes to grow the crack from some initial condition to a critical condition. In

Algorithm 6 Computation of crack advancement Δa

```
da_max=edge_length                          ▷ input: edge length for example
dK_max=MAXVAL(dK)
dN=da_max/(A*dK_max^m)

Compute da for all nodes
for i = 1, NNOD do
    da(i)=A*(dK(i)**m)*dN
    K_near = 0
end for

Correct too small value of da
da_min=0.02*da_max                          ▷ input: 2% of da_max for example
dNloc=dN                                    ▷ initialize local dN
for i = 1, NNOD do
    if (da(i).lt.da_min) then
        dNloc(i)=da_min*dN/da(i)            ▷ compute local value of dN
        da(i)=da_min                        ▷ correct too small da
    end if
end for
```

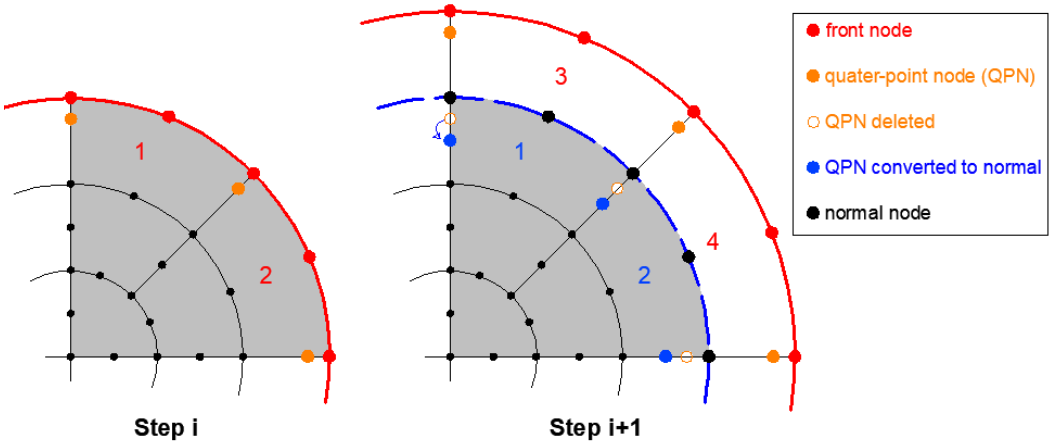


Figure 4.13 – Crack re-meshing

this section we will validate the global evolution of the crack length. We consider a circular crack of radius $1mm$ in a homogeneous cylinder ($E = 2MPa$, $\nu = 0.3$) of dimensions $R = 60mm$, $H = 120mm$ subjected to tension $\sigma = 2MPa$. The cylinder is meshed with 192 four-node elements and 194 nodes, while the crack is meshed with 128 eight-node elements and 417 nodes (see Fig. 4.14). The material properties for the Paris law are $A = 10^{-8}$ mm/cycle and $m = 4.5$. For this simulation, $\Delta a^{\max}$ is taken equal to $r/10$ and the SIF for a penny-shaped crack in an infinite medium

can be used, see Williams [107]:

$$\Delta K_{\mathrm{I}} = \frac{2}{\pi}\sigma\sqrt{\pi r}, \qquad \Delta K_{\mathrm{II}} = 0. \tag{4.17}$$

The computed fatigue growth (crack radius against number of loading cycles) is plotted in Fig. 4.15 and compared to the analytic solution derived from the above SIF formulas.

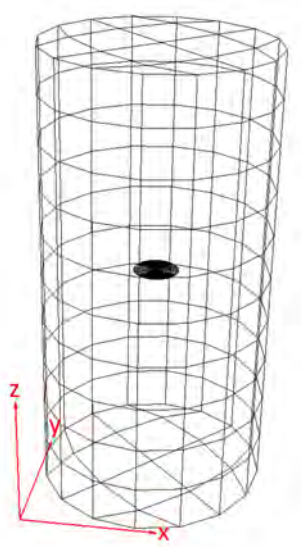


Figure 4.14 – Cracked cylinder model

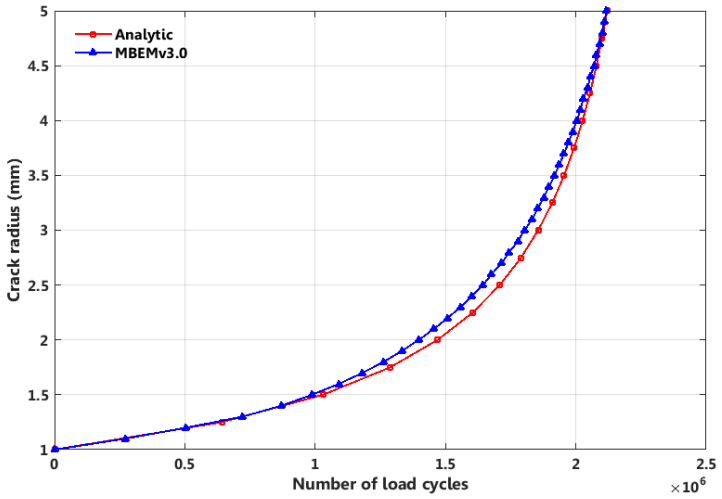


Figure 4.15 – Fatigue life validation

4.2.4 Rigidity loss of a multi-cracked sample

In this application, we consider the simulation of a tensile/compression fatigue test on cylindrical sample of a semi-coarse asphalt concrete considered as homogeneous ($E = 6000MPa$, $\nu = 0.3$). The sample of dimensions $R = 60mm$, $H = 120mm$ contains 120 small cracks with initial radius of 5 mm, see Fig. 4.16. The material properties for the Paris law are $A = 10^{-8}$ mm/cycle and $m = 4.5$. When the number of cycles increases, the cracks propagate and we can observe a loss of rigidity of the sample, see Fig. 4.17. At the end of the propagation, the final radius of the cracks is 115 mm. In this Figure we can notice the acceleration of the rigidity loss. We can also determine the fatigue life of the sample which is the number of cycles required to lose half of the rigidity. In this simulation, the fatigue life is 4.5 10^6 cycles.

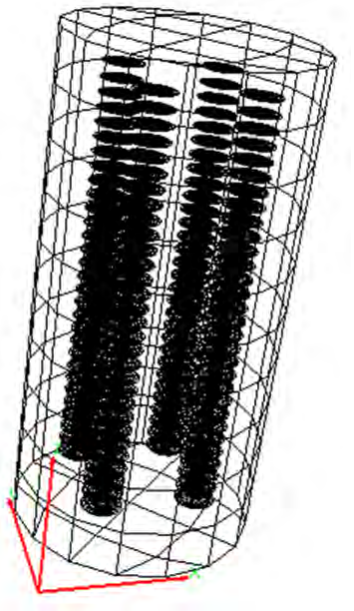


Figure 4.16 – Model of a multi-cracked sample

4.3 Surface breaking cracks

4.3.1 Stationary SBC treatment

Surface-breaking cracks (SBCs) are among the critical sources of structural degradation. Their detection and the prediction of their evolution are of great interest in civil engineering. Many investigations have been devoted to SBCs. For example, Raju and Newman [108] provided stress intensity factors by using the FEM, Feng and Hong [109] presented an expression of the surface crack opening displacement in a plate under tension and bending, Frangi [41] used the SGBEM to simulate a surface breaking crack, Ramezani e al. [110] used the dual BEM to evaluate stress intensity factors of surface cracks in round bars. In *MBEMv2.0*, all the crack prob-

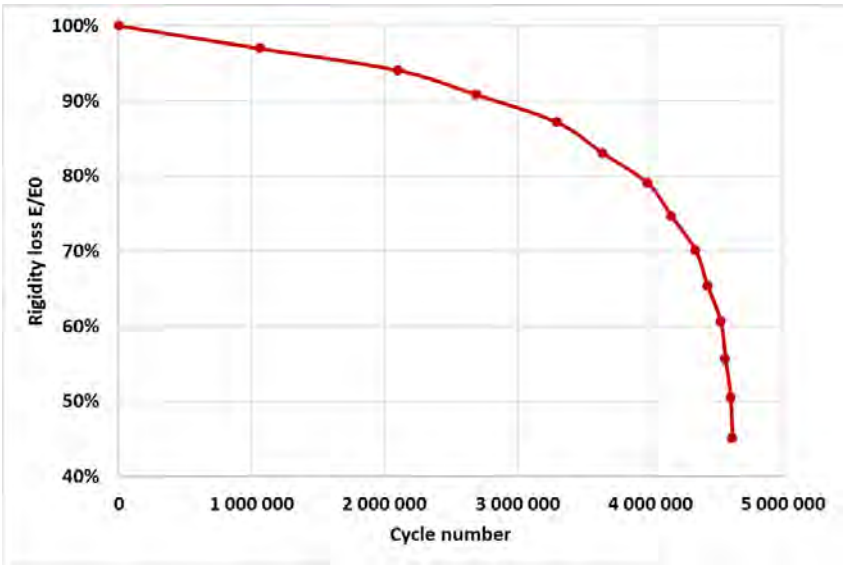


Figure 4.17 – Rigidity loss of a multi-cracked sample

lems treated are internal cracks which can not intersect a surface or an interface. We therefore have included SBC analysis in our code.

Let us consider a semi-circular horizontal edge crack (Fig. 4.18) of radius r breaking at the center of a plate (whose dimensions are $H \times H \times b$). The difference according to an internal crack is that there are three unknowns to be determined at the intersection segment where the plate and the crack meet. So, the simulation of a surface breaking crack, requires a proper treatment of the DOFs at nodes on the intersection $S_t \cap S_c$ (see Fig. 4.19). Each node of this intersection carries three unknowns, namely the displacements $\boldsymbol{u}^{\text{upper}}$ and $\boldsymbol{u}^{\text{lower}}$ of the upper and lower faces of the breaking crack (the node belonging to S_t) and the COD $\boldsymbol{\phi}$ (the node also belonging to S_c), linked by

$$\boldsymbol{\phi} = \boldsymbol{u}^{\text{upper}} - \boldsymbol{u}^{\text{lower}} \tag{4.18}$$

Such nodes are treated as multiple nodes, see details in [92]. The technique is already used in section 4.1 to treat interface intersections as double nodes. The principle is the same but here, the special nodes are treated as triple nodes (see Fig. 4.19). The technique permits to obtain three independent equations and equation (4.18) is then used to eliminate one of the unknowns in the SGBEM system. The resulted system is solved as usual and the displacements and crack opening displacement are obtained.

4.3.2 Stationary SBC validation

To validate this feature, we recall the semi-circular horizontal edge crack (Fig. 4.18) of radius $r = 10mm$ breaking at the center of a plate (whose dimensions are $H \times H \times b$, with $H = 10r, b = 2,5r$) under tension $\sigma = 0.1MPa$. The adopted plate and crack dimensions are such that the model reasonably represents an edge crack in an infinite

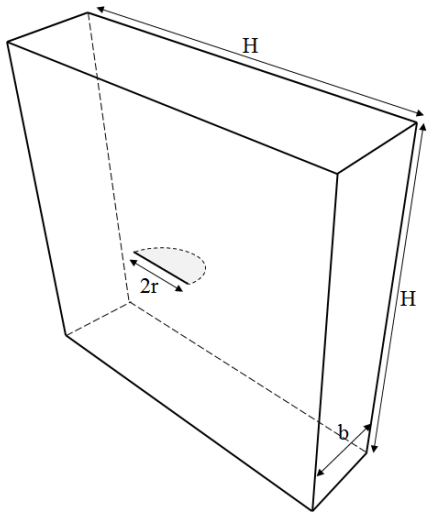


Figure 4.18 – Surface-breaking crack example: geometry

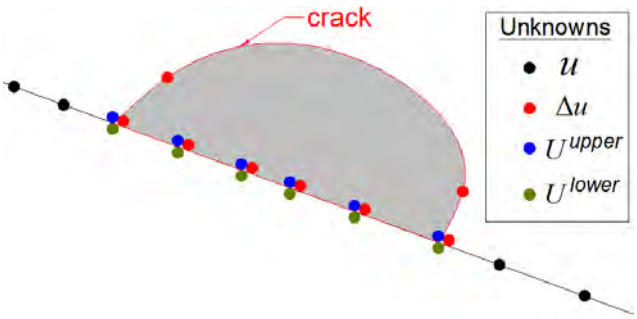


Figure 4.19 – Surface-breaking crack: multiple nodal unknowns

plate. The mesh features 1 395 eight-noded elements (1 200 elements for the plate surface and 195 for the crack, see Fig. 4.20) and 12 651 DOFs. The tolerance for the iterative solver is set to $\varepsilon = 10^{-3}$.

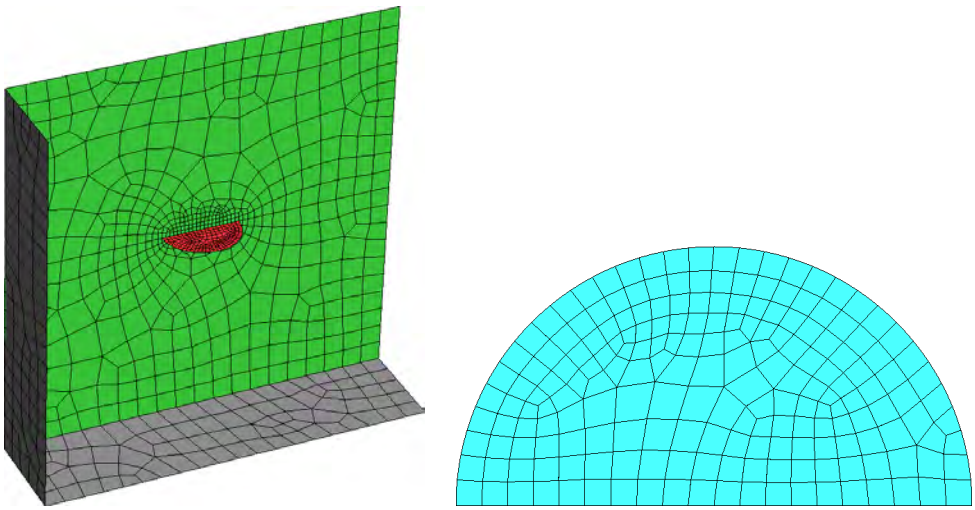
The obtained surface crack opening displacement is compared with that obtained by Feng and Hong [109] equation 4.19 and Frangi's free surface correction [41].

$$SCOD = S * COD \tag{4.19}$$

where S is a free surface correction, and COD is the COD for an internal crack. For this example, the difference with respect to an internal crack is that the SIF varies along the crack front, with the maximum value at the surface-breaking point. This variation can be represented in terms of the normalized stress intensity factor (NSIF) given by:

$$K_{\mathrm{I}}^{\star}(s) = \frac{K_{\mathrm{I}}}{2\sigma}\sqrt{\frac{\pi}{r}}. \tag{4.20}$$

In Fig. 4.22, values of $K_{\mathrm{I}}^{\star}$ computed using the present FM-SGBEM code (*MBEMv3.0*) are compared with numerical results by Frangi [41] and with ap-



(a) (b)

Figure 4.20 – Semi-circular edge crack: (a) mesh (b) crack mesh

proximate values by Sun and Jin [111] and Anderson [112]. The obtained NSIF based on extrapolation equation (4.15) noted *MBEMv3.0 2pts*, agrees within 1% (except at the surface-breaking node) with the computed values of [53] while the NSIF based on equation (4.16) noted *MBEMv3.0 1pt* , follows the trend of the curve to the surface-breaking node. The vertical crack opening displacement on the deformed mesh and a zoom on the crack-outer surface intersection are presented on Fig. 4.23.

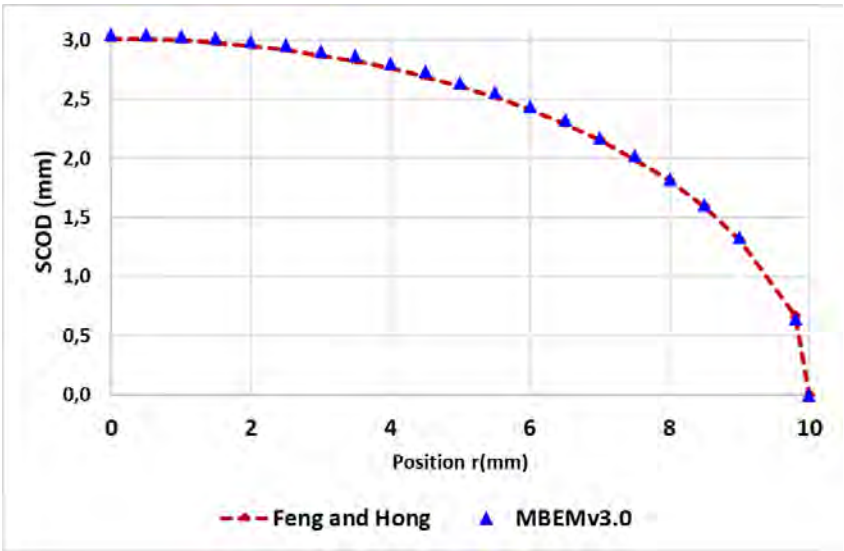


Figure 4.21 – Surface-breaking crack: crack opening displacement along the crack front

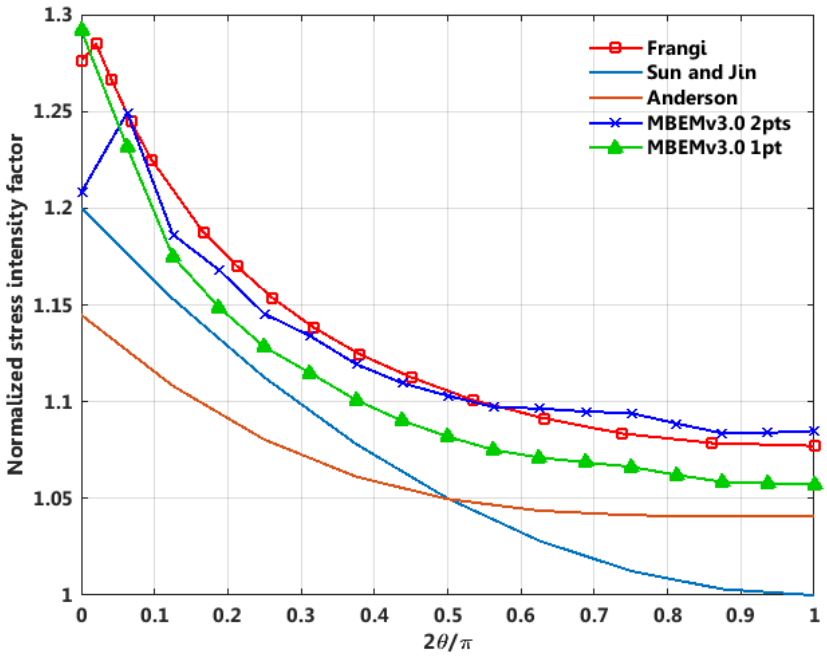


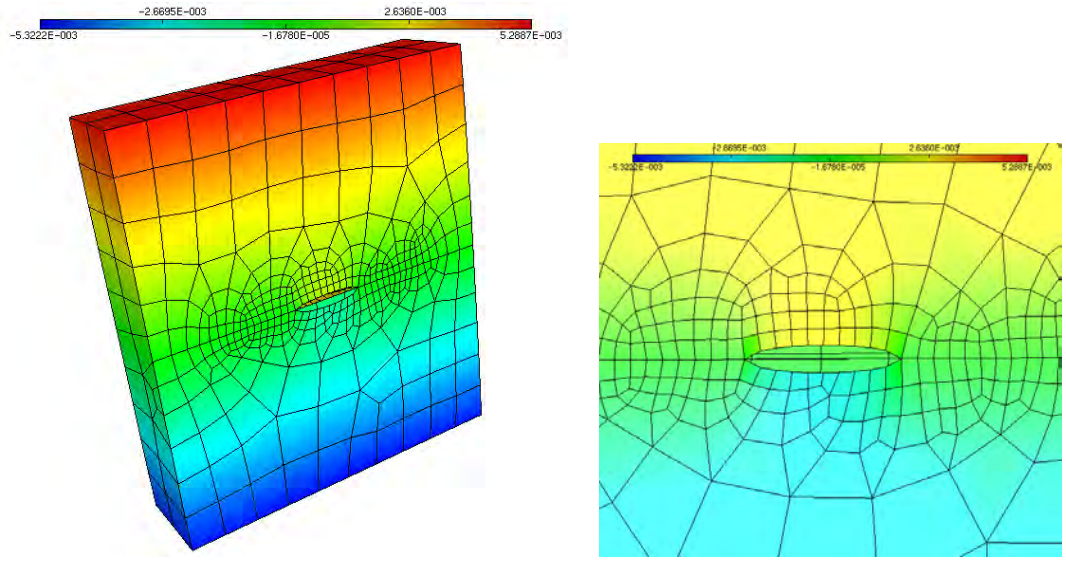
Figure 4.22 – Surface-breaking crack: normalised SIF $K_{\mathrm{I}}^{\star}$ along the crack front

MBEMv3.0 also accommodates multiple surface-breaking cracks, an important feature for civil engineering applications. To illustrate this, consider two surface-breaking cracks in a plate (see Fig. 4.24a, where a part of the surface is removed so that the interior can be seen). Figure 4.24b presents the COD on the cracks and the vertical displacement on the plate. The case of interface-crossing cracks, also very useful, is left for future work.

4.3.3 Propagation of horizontal SBC

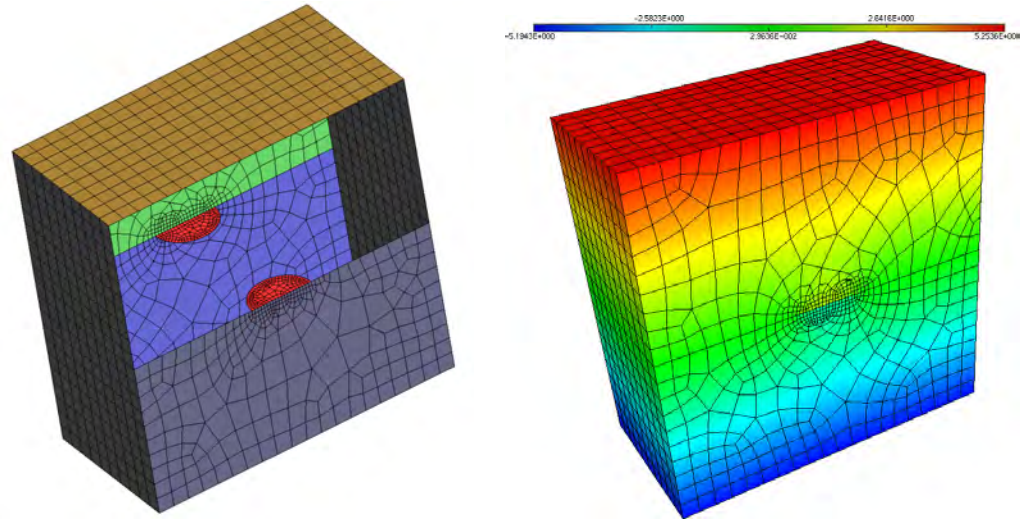
In the previous subsection, multiple node technique is used to simulate surface breaking cracks. Let us see now how to propagate a surface breaking crack. Let us recall the surface breaking crack presented in Fig. 4.18. The elastostatic simulation is already achieved, so, the crack opening displacements are known at the crack front. The SIFs can then be evaluated based on equation (4.15) or (4.16). When the SIFs are known, the propagation angle $\theta_0(s)$ and the crack advancement $\Delta a(s)$ can be computed with the Algorithm 6 presented in subsection 4.2.2, that involves $\Delta a^{\max}$ and $\Delta a^{\min}$. For internal cracks, it is straightforward to generate new frontal elements. But for surface breaking cracks, one must pay attention to the propagation of the crack front elements which intersect the outer surface, see Fig. 4.25.

As shown in Fig. 4.25, two crack front elements intersect the outer surface: E_{left} and E_{right}. The problem here comes from the extension of the node P. According to the outer surface mesh, the near existing nodes on the intersection are noted $A1$,



(a) (b)

Figure 4.23 – Surface breaking cracks: (a) vertical COD on deformed mesh (b) Zoom on intersection



(a) (b)

Figure 4.24 – Double surface breaking cracks: (a) model (b) vertical COD

$A2$, etc. After extension, the new node which must be created is noted M. If the advancement Δa of the node P is not exactly the same as the distance $d(P, A1)$, re-meshing task must be performed. Generally the outer surface mesh is coarser than the crack mesh. So, $d(P, A1)$ is greater than Δa and the outer surface needs

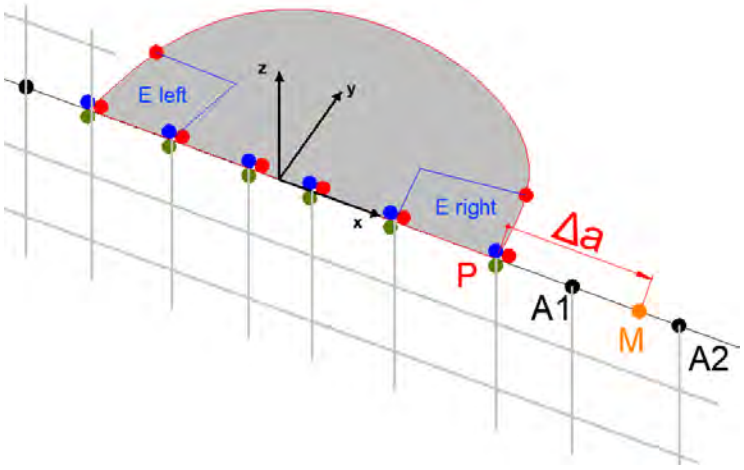


Figure 4.25 – Horizontal SBC propagation: re-meshing problem

to be re-meshed. This re-meshing is counterproductive because one of the main advantages of the BEM will be lost. The flexibility offered by Algorithm 6 permits to avoid the re-meshing task. The advancement of the node P is taken equal to $d(P, A1)$ so that M is at the same position as $A1$ and the corresponding local cycle number is determined, see Algorithm 7. The crack propagation of this example can then be achieved by re-using multiple node technique. Fig. 4.26 shows the vertical COD after seven increments of crack propagation and a zoom on the deformed mesh. Fig. 4.27 presents the local cycle number graph. The cycle number of the initial nodes of the semi-circular crack is zero. A non-zero value represents the real number of cycles required to reach the position of the concerned node. So, the real shape of the crack is given by the iso-values of this graph. We can notice that the crack opens more on the sides than in depth of the plate.

4.3.4 Propagation of inclined SBC

Let us consider now an inclined surface breaking crack. As shown in subsection 4.3.3, the problem of surface breaking crack propagation is the crack re-meshing. We used the flexibility to set Δa to achieve the propagation in the last subsection. But here the propagation angle is also an issue. In fact, for the horizontal surface breaking crack, the load was simple, and the crack propagated in the horizontal direction, otherwise the same problem can occur, see Fig. 4.28. In this example, the outer surface mesh is generated in the propagation direction (see Fig. 4.29) to avoid re-meshing task. The aim here is just to show that it is possible to simulate surface breaking crack propagation with the MBEMv3.0. Multiple node technique is then used, and the crack propagation is achieved. Fig. 4.30 shows the vertical COD and a zoom on the deformed mesh. Fig. 4.30 presents two different 3D views of the inclined crack.

The propagation of inclined surface breaking cracks is one of the limits of

Algorithm 7 Computation of crack advancement Δa for SBC

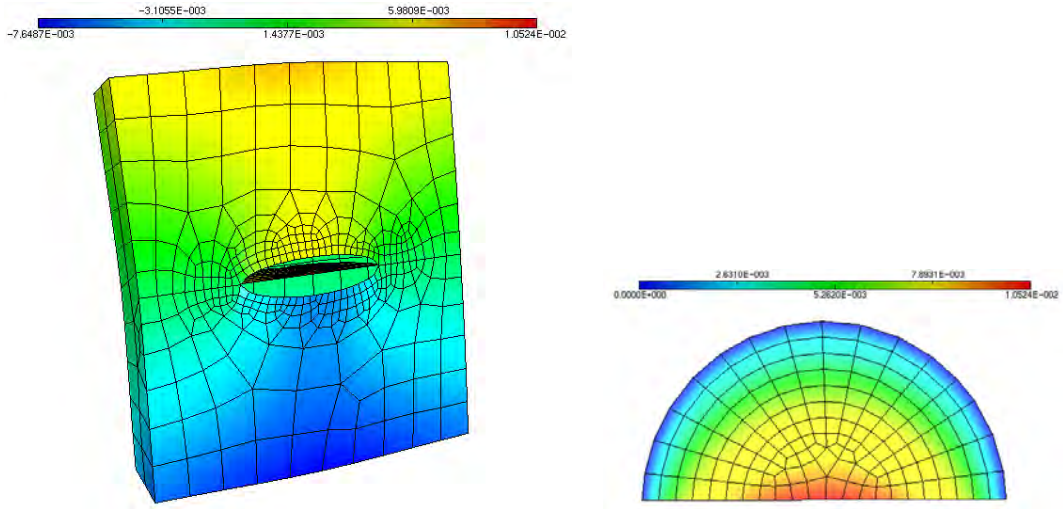
```
da_max=edge_length                          ▷ input: edge length for example
dK_max=MAXVAL(dK)
dN=da_max/(A*dK_max^m)
Compute da for all nodes
for i = 1, NNOD do
    da(i)=A*(dK(i)**m)*dN
    K_near = 0
end for
Correct too small value of da
da_min=0.02*da_max                          ▷ input: 2% of da_max for example
dNloc=dN                                    ▷ initialize local dN
for i = 1, NNOD do
    if (da(i).lt.da_min) then
        dNloc(i)=da_min*dN/da(i)            ▷ compute local value of dN
        da(i)=da_min                        ▷ correct too small da
    end if
end for

Crack front nodes at the outer surface
for i = 1, NNOD do
    if snode(i) then
        Call determine_distance_d (i,d)
        dNloc(i)=d*dN/da(i)                 ▷ compute local value of dN
        da(i)=d                             ▷ use d for advancement
        Call multiple_nodes(i)              ▷ Triple the node
    end if
end for
```

MBEMv3.0. In the example presented here, the propagation direction is known before the simulation and is taken into account to generate the mesh. The code should be improved to properly treat these types of crack.

4.3.5 Interface breaking cracks

We consider now interface breaking cracks. The example here concerned a cracked two-layered cube. The radius of the crack is taken $r = 1cm$. The cube, dimension $10 \times 10 \times 10$, is composed of two bodies which have identical material properties (for validation purposes): $E_1 = E_2 = 1MPa$ and $\nu_1 = \nu_2 = 0,3$. The cube is fixed at the bottom and subjected to uniform compression $p = 1$ on the top face. This feature will be useful for the simulation of cracks which start at a joint. This situation is very common in civil engineering especially in roads or in composite structures. The problem with an interface according to an outer surface is that



(a) (b)

Figure 4.26 – Propagation of horizontal surface breaking cracks: (a) vertical COD on deformed mesh (b) Zoom on intersection

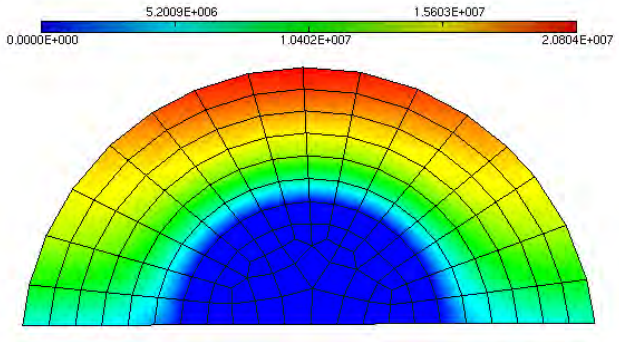


Figure 4.27 – SBC propagation: local cycle number graph

two unknowns (displacement and tension) should be determined. So we have four unknowns at the special nodes involved in interface breaking crack problems namely the displacements $\boldsymbol{u}^{\text{upper}}$ and $\boldsymbol{u}^{\text{lower}}$ of the upper and lower faces of the breaking crack, the crack opening displacement $\boldsymbol{\phi}$ and the tension T on the interface linked by:

$$\boldsymbol{\phi} = \boldsymbol{u}^{\text{upper}} - \boldsymbol{u}^{\text{lower}} \tag{4.21}$$

The treatment of an interface breaking cracks is similar to that of a surface breaking crack. Taking into account the tension unknown is the only difference. The triple node technique used in the previous section is thus extended in this section as quadruple node technique: the additional node being that of the tension. If, in particular the intersection is located on an edge, others nodes should be added to determine the different tensions involved.

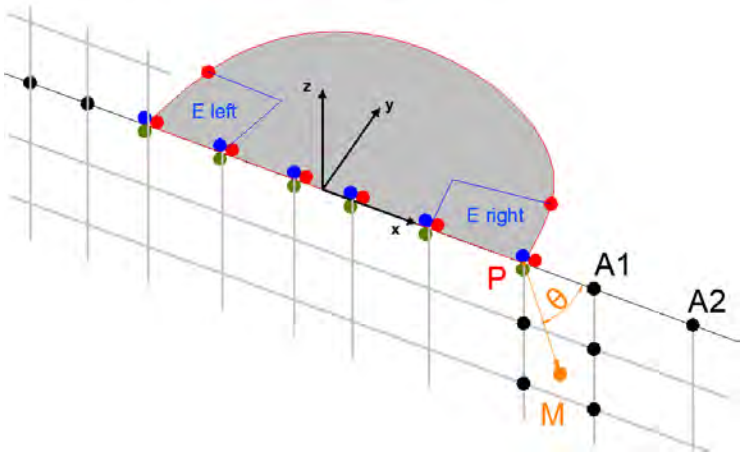


Figure 4.28 – Inclined SBC propagation: re-meshing problem

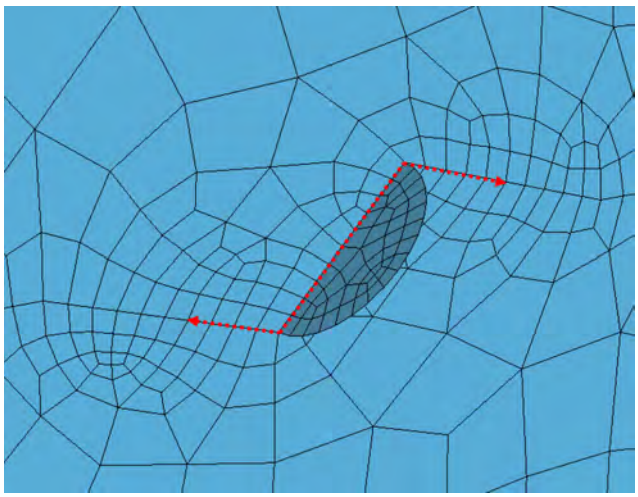


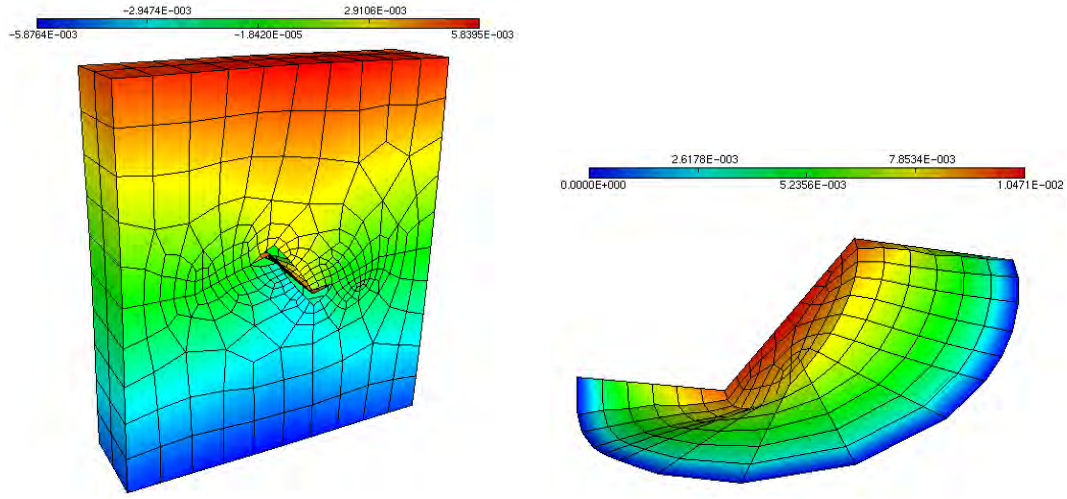
Figure 4.29 – Inclined SBC propagation: Mesh

The propagation of interface breaking cracks encounters the same issues as that of a general surface breaking crack. In the example presented below the propagation direction is simple and known before the simulation and is taken into account to generate the mesh. The issues being bypassed, MBEMv3.0 can solve this example. Fig. 4.32 shows the model and the vertical displacements on the cube and normal COD on the crack after 10 propagation cycles.

4.4 Investigations on contact problems

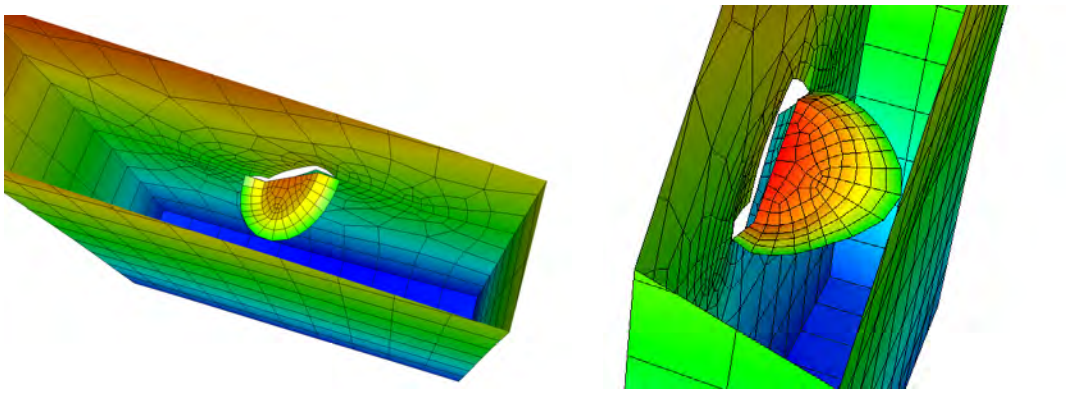
When a solid is subject to loads there is a jump of the displacement, the so-called crack opening displacement (COD):

$$\Delta \boldsymbol{u} = \boldsymbol{u}^{+} - \boldsymbol{u}^{-}$$



(a) (b)

Figure 4.30 – Propagation of inclined SBC: (a) vertical COD on deformed mesh (b) Zoom on intersection

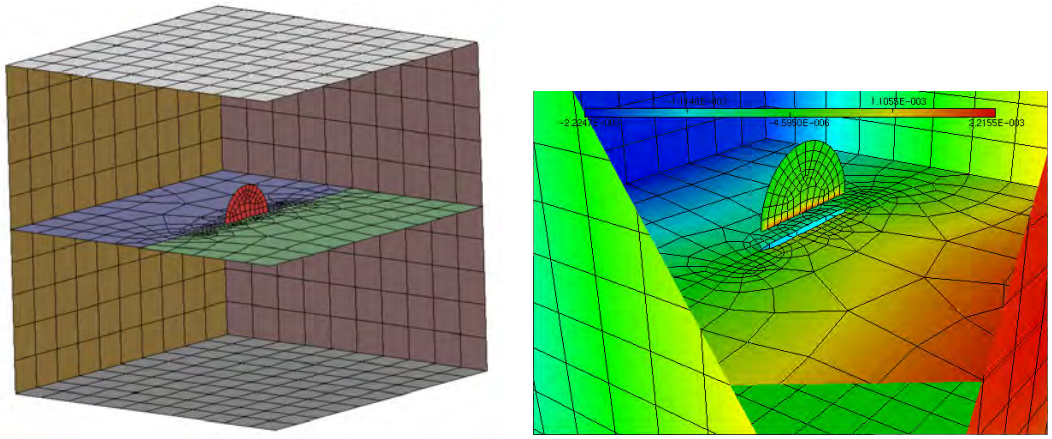


(a) (b)

Figure 4.31 – Propagation of inclined SBC: (a) Perspective 1 (b) Perspective 2

In chapter 2, we said that the normal jump is non negative, $\Delta \boldsymbol{u} \geq 0$. Generally, this condition is discarded in the literature because most problems in fracture mechanics involve tensile loads which open the crack. For complex civil engineering problems, condition $\Delta \boldsymbol{u} = 0$ must be considered in order to avoid overlapping phenomena. In fact, this problem can be highlighted easily. Let us consider a horizontal circular crack (radius $r_c = 10mm$) in a cube of dimensions $100 \times 100 \times 100mm^3$ (Fig. 4.33) subject to compression (see Fig. 4.34). The vertical crack opening displacement is shown in Fig. 4.35.a. The negative COD values represent an overlap.

To avoid this, condition $\Delta \boldsymbol{u} = 0$ must be enforced at the nodes where the COD



(a) (b)

Figure 4.32 – Interface breaking cracks: (a) model (b) Displacements and COD

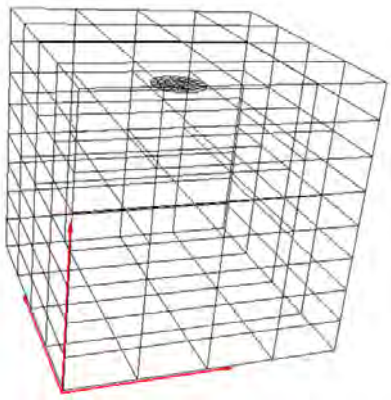


Figure 4.33 – Horizontal crack in a cube

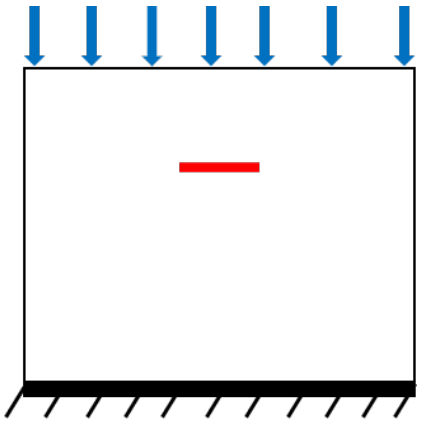
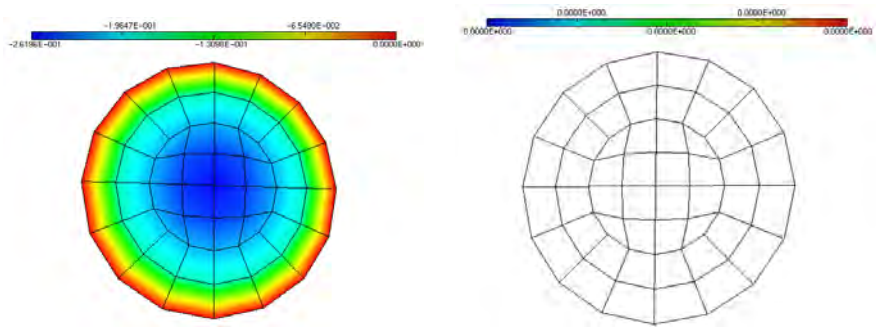


Figure 4.34 – Crack under compression

is negative ($\Delta \boldsymbol{u} < 0$). But it is not possible to predict these nodes for complex problems. So, an iterative algorithm is proposed. The fracture problem is first solved simply without any condition. Then the negative COD nodes are identified, the $\Delta \boldsymbol{u} = 0$ condition is enforced and the new system is solved. The correction algorithm requires a very few number of iterations. With only two iterations, the compression test is solved well see Fig. 4.35.b. No value is present in this figure because all COD values are null.

Let us consider now a circular crack (radius $r_c = 10mm$) in a cube (dimensions $100 \times 100 \times 100mm^3$) subject to bending, see Fig. 4.36. The position of the crack and the position of the load is so that a part of the crack should be opened, and the other part should be closed. Fig. 4.37.a shows the COD obtained without correction. We can see an overlap on the part which should be closed. The new results obtained by using the correction procedure are presented in Fig. 4.37.b.



(a) (b)

Figure 4.35 – Crack under compression: (a) initial results (b) modified results

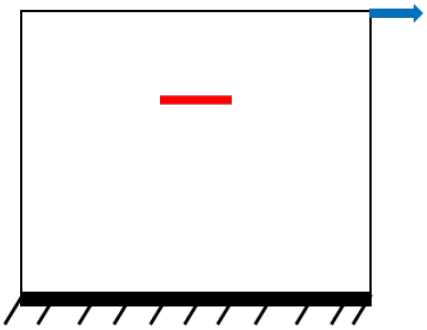
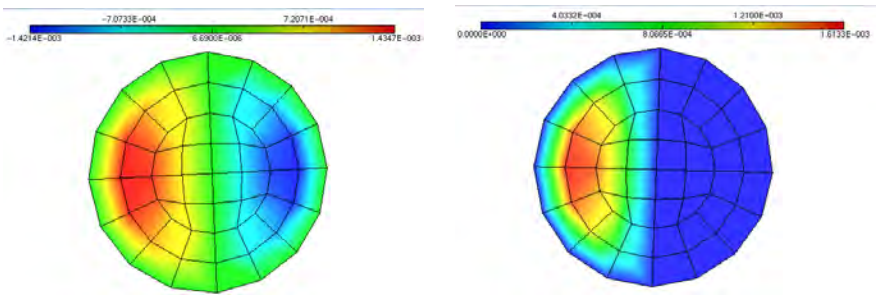


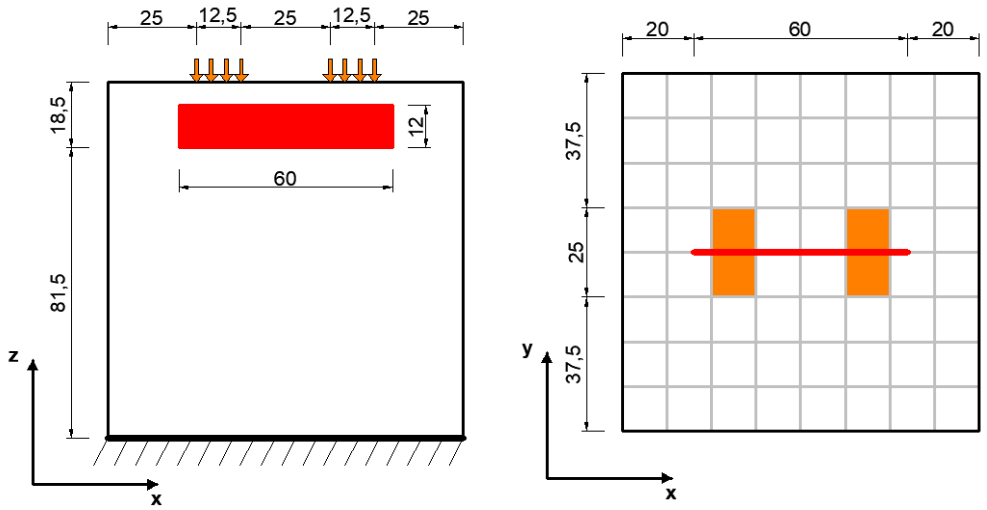
Figure 4.36 – Crack under flexion: model



(a) (b)

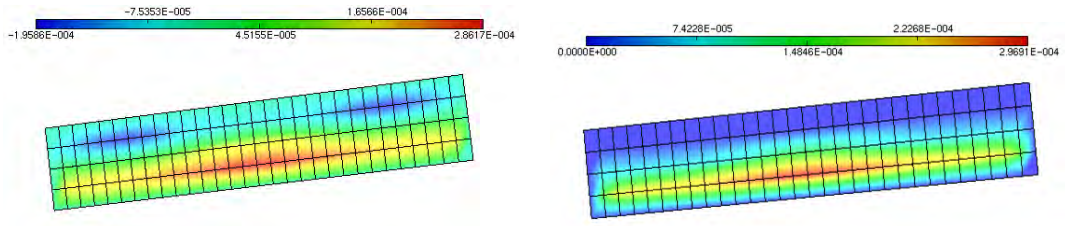
Figure 4.37 – Crack under flexion: (a) initial results (b) modified results

Let us consider another example: a rectangular crack in a cube subjected to special bending, see Fig. 4.38. This model is chosen so that the crack can present opened parts and closed parts. Fig. 4.39.a shows the COD obtained without correction. We can see an overlap phenomenon on the upper part of the crack (negative COD in blue). The new results obtained only after two iterations of the correction procedure are presented in Fig. 4.39.b.



(a) (b)

Figure 4.38 – Rectangular crack under flexion model: (a) xz plane (b) xy plane



(a) (b)

Figure 4.39 – Rectangular crack under flexion: (a) initial results (b) modified results

4.5 Conclusions

In this chapter, many studies are realized to extend *MBEMv2.0*. Complex multizone problems are treated. A propagation law is implemented to direct the re-meshing algorithm during crack propagation. This re-meshing algorithm itself is improved to be flexible. Then multiple node technique is used to solve surface breaking crack problems. An automatic multiple node algorithm combined to the flexibility of the re-meshing algorithm permits the simulation of the propagation of surface breaking cracks in particular cases. The method is then extended for the simulation of a particular interface breaking crack. However, the propagation of a general surface breaking crack or interface breaking crack is one of the limits of *MBEMv3.0*. In the example presented in this chapter, the propagation direction is known before the simulation and is taken into account to generate the mesh. The code should be improved to properly treat these types of crack.

Chapter 5

FEM-BEM coupling

Contents

Despite all the possibilities that the boundary element method offers, the finite element method retains enormous advantages such as the ability to perform nonlinear calculations or the simplicity in treating thin structures. It is therefore interesting to couple the two methods. In this chapter FEM-BEM coupling is investigated to model fiberglass grids. A direct coupling to membrane finite elements is implemented and validated.

5.1 Finite element formulation

5.1.1 Overview

The essence of finite element method is to divide a complex problem (in terms of geometry, loading, boundary conditions, etc.) into a finite number of sub-divisions (elements) inter-connected at nodes. This process is called discretization. From a mathematical point of view, the finite element method permits to calculate an approximation of a function that verifies a system of partial differential equations inside a domain and a system of boundary conditions on the boundaries of that domain. Many reference books exist on this topic. In the field of structural calculation, Zienkiewicz and Taylor [113] or Batoz [114] can be cited for a general presentation of the method. In this work, the displacement approach is used. As the displacement field can not generally be calculated explicitly, we are looking for an approximation in the form of:

$$u = \sum N_i u_i$$

where the N_i are the shape functions defined on the domain and the parameters u_i are the unknowns of the problem. So, the unknown u, continuous function all over the domain is replaced by a discrete set of n unknowns u_i. The next step is to divide the domain into sub-domains or finite elements. The shape functions are then defined on each of the sub-domains and the approximation of the field u is computed element by element.

A key point of the finite element method is the transition from a local differential form to an integral form. The integral form on the domain can be approximated by the summation on the elements belonging to the domain. The transition from the local form to the integral form can be formulated for example by a weighted residual method or, in the field of structural calculation, by applying the virtual work principle which allows to obtain a weak form directly, that is to say, an equation showing only the first derivatives of the displacement field.

For the linear-elastic-static analysis of structures, the final form of equation will be made in the form of:

$$\{F\} = [K]\{u\} \tag{5.1}$$

where $\{F\}$, $[K]$ and $\{u\}$ are the nodal loads, global stiffness and nodal displacements respectively. Varieties of engineering problems like solid and fluid mechanics and heat transfer, can easily be solved by the concept of finite element technique.

5.1.2 Variational principle

Variational formulation is the generalized method of formulating the element stiffness matrix and load vector. Let us consider a 3D structural problem. The strain energy is given by the relation:

$$U = \frac{1}{2} \iiint_{\Omega} \{\varepsilon\}^T \{\sigma\} \, d\Omega$$

The strain-displacement relationship can be expressed as:

$$\{\varepsilon\} = [B]\{u\}$$

where $\{u\}$ is the displacement vector in x, y and z directions and $[B]$ is called the strain displacement relationship matrix. The stress can be represented in terms of its constitutive relationship matrix:

$$\{\sigma\} = [D]\{\varepsilon\}$$

where $[D]$ is the constitutive relationship matrix. Using the above relationship in the strain energy equation one can arrive at:

$$U = \frac{1}{2}\iiint_{\Omega} ([B]\{u\})^T [D][B]\{u\}\, d\Omega$$

Applying the variational principle one can express:

$$\{F\} = \frac{\partial U}{\partial \{u\}} = \iiint_{\Omega} [B]^T [D][B]\, d\Omega\, \{u\}$$

This relation can be rewritten $\{F\} = [K]\{u\}$, with the stiffness matrix $[K]$:

$$[K] = \iiint_{\Omega} [B]^T [D][B]\, d\Omega$$

5.1.3 Membrane finite element

In this study, a simple membrane finite element is implemented, but the procedure can easily be extended to other types of elements. Membrane is a two-dimensional element in which it is assumed that the stresses are uniform in the thickness, see Fig. 5.1. It is used to model thin structures working in membrane, that is to say without flexural rigidity or thick structures when one can consider that the components of the stress tensor do not vary in the thickness. For a membrane finite element, there are two unknowns per node: the two components of the displacement vector:

$$\{u\} = \begin{Bmatrix} u_x \\ u_y \end{Bmatrix}$$

The strain–displacement relation is given by:

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{Bmatrix} u_x \\ u_y \end{Bmatrix}$$

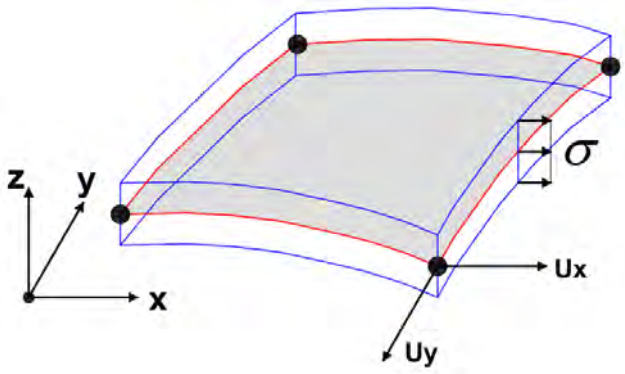


Figure 5.1 – Membrane element

The strain–displacement operator is thus:

$$[A] = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}$$

For plane stress, the stresses in the z direction are considered to be negligible $\sigma_{zz} = \sigma_{yz} = \sigma_{xz} = 0$ and the stress-strain relation is given by:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [D] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_x \\ \gamma_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_x \\ \gamma_{xy} \end{Bmatrix}$$

For the discretization, let us consider quadrilateral isoparametric elements, see Fig. 5.2. The elementary stiffness matrix can be written below, where t is the thickness of the membrane.

$$[K_E] = \iint t[B]^T [D] [B] \, dxdy \tag{5.2}$$

The interpolation functions are:

$$N_1 = \tfrac{1}{4}(1-\xi)(1-\eta) \quad ; \quad N_2 = \tfrac{1}{4}(1-\xi)(1+\eta)$$

$$N_3 = \tfrac{1}{4}(1+\xi)(1+\eta) \quad ; \quad N_4 = \tfrac{1}{4}(1+\xi)(1-\eta)$$

The derivatives of the shape functions with respect to the isoparametric coordinates, can be written in a matrix noted $[D_N]$:

$$[D_N] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \tag{5.3}$$

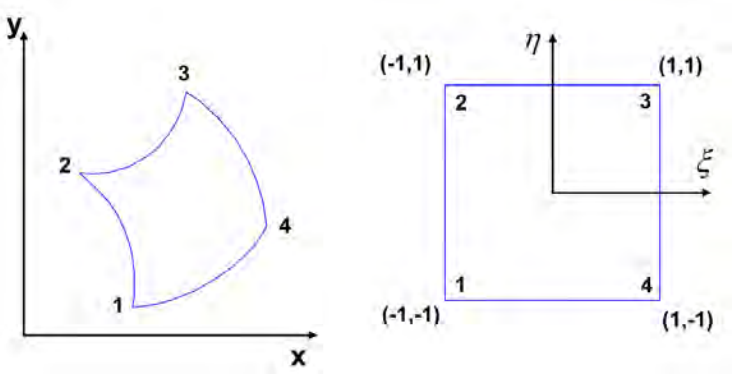


Figure 5.2 – Global and local elements

The coordinate transformation between global and local coordinate systems is:

$$x = \sum_{i=1}^{4} N_i(\xi,\eta)x_i \quad ; \quad y = \sum_{i=1}^{4} N_i(\xi,\eta)y_i$$

The displacements within the element are also interpolated from the nodal values:

$$u_x = \sum_{i=1}^{4} N_i(\xi,\eta)u_x^i \quad ; \quad u_y = \sum_{i=1}^{4} N_i(\xi,\eta)u_y^i$$

The displacement derivatives in the two coordinate systems can be related by the jacobian matrix:

$$\left\{ \begin{array}{c} \frac{\partial u_x}{\partial \xi} \\ \frac{\partial u_x}{\partial \eta} \end{array} \right\} = [J] \left\{ \begin{array}{c} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_x}{\partial y} \end{array} \right\} \quad ; \quad \left\{ \begin{array}{c} \frac{\partial u_y}{\partial \xi} \\ \frac{\partial u_y}{\partial \eta} \end{array} \right\} = [J] \left\{ \begin{array}{c} \frac{\partial u_y}{\partial x} \\ \frac{\partial u_y}{\partial y} \end{array} \right\}$$

By differentiating the displacement with respect to (ξ,η) by invoking the chain rule:

$$\frac{\partial u_x}{\partial \xi} = \frac{\partial u_x}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial u_x}{\partial y}\frac{\partial y}{\partial \xi} \quad ; \quad \frac{\partial u_x}{\partial \eta} = \frac{\partial u_x}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial u_x}{\partial y}\frac{\partial y}{\partial \eta}$$

the elements of the matrix $[J]$ are obtained:

$$[J] = \left[\begin{array}{cc} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{array} \right] = \left[\begin{array}{cc} \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} x_i & \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} y_i \\ \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} x_i & \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} y_i \end{array} \right] = [D_N] \left[\begin{array}{cc} x & y \end{array} \right]$$

The nodal displacements of the element can be listed as a vector:

$$\left\{u^T\right\} = \left\{u_x^1 \quad u_y^1 \quad u_x^2 \quad u_y^2 \quad u_x^3 \quad u_y^3 \quad u_x^4 \quad u_y^4\right\}$$

The strain–displacement matrix $[B]$ needed to compute the elementary stiffness matrix can be written:

$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix}$$

We have to find the derivatives of the shape functions with respect to the global (x,y) coordinates, but the shape functions are expressed in the isoparametric (ξ, η) coordinates. To overcome this problem we will define $[B]$ as a product of three matrices:

$$[B] = [H] \left[J_{\text{exp}}^{-1}\right] [D_{N,\text{exp}}] \tag{5.4}$$

The first matrix, $[H]$, relates strains and displacement derivatives:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_x \\ \gamma_{xy} \end{Bmatrix} = [H] \begin{Bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial x} \\ \frac{\partial u_y}{\partial y} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{Bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial x} \\ \frac{\partial u_y}{\partial y} \end{Bmatrix}$$

$$\boxed{[H] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} \tag{5.5}$$

The second matrix, $\left[J_{\text{exp}}^{-1}\right]$, is an expanded form of the inverse jacobian matrix $[J]^{-1}$:

$$\begin{Bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial x} \\ \frac{\partial u_y}{\partial y} \end{Bmatrix} = \begin{bmatrix} [J]^{-1}_{2x2} & 0_{2x2} \\ 0_{2x2} & [J]^{-1}_{2x2} \end{bmatrix} \begin{Bmatrix} \frac{\partial u_x}{\partial \xi} \\ \frac{\partial u_x}{\partial \eta} \\ \frac{\partial u_y}{\partial \xi} \\ \frac{\partial u_y}{\partial \eta} \end{Bmatrix}$$

$$\boxed{\left[J_{\text{exp}}^{-1}\right] = \begin{bmatrix} [J]^{-1}_{2x2} & 0_{2x2} \\ 0_{2x2} & [J]^{-1}_{2x2} \end{bmatrix}} \tag{5.6}$$

The third matrix, $[D_{N,\text{exp}}]$, is an expanded form of the matrix, $[D_N]$:

$$\begin{Bmatrix} \frac{\partial u_x}{\partial \xi} \\ \frac{\partial u_x}{\partial \eta} \\ \frac{\partial u_y}{\partial \xi} \\ \frac{\partial u_y}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} & 0 \\ \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} & 0 \\ 0 & \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} \\ 0 & \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \begin{Bmatrix} u_x^1 \\ u_y^1 \\ u_x^2 \\ u_y^2 \\ u_x^3 \\ u_y^3 \\ u_x^4 \\ u_y^4 \end{Bmatrix}$$

$$[D_{N,\text{exp}}] = \begin{bmatrix} D_{N,11} & 0 & D_{N,12} & 0 & D_{N,13} & 0 & D_{N,14} & 0 \\ D_{N,21} & 0 & D_{N,22} & 0 & D_{N,23} & 0 & D_{N,24} & 0 \\ 0 & D_{N,11} & 0 & D_{N,12} & 0 & D_{N,13} & 0 & D_{N,14} \\ 0 & D_{N,21} & 0 & D_{N,22} & 0 & D_{N,23} & 0 & D_{N,24} \end{bmatrix} \tag{5.7}$$

We now have the tools to evaluate the stiffness matrix of an isoparametric quadrilateral element:

$$[K_E] = \int_{-1}^{1}\int_{-1}^{1} \left(HJ_{\text{exp}}^{-1} D_{N,\text{exp}}\right)^T .t.D. \left(HJ_{\text{exp}}^{-1} D_{N,\text{exp}}\right) . \det(J)\, d\xi d\eta \tag{5.8}$$

The integral is evaluated with Gauss quadrature and then the elementary stiffness matrices are assembled to form the global stiffness matrices of the domain.

5.2 FEM-BEM coupling

5.2.1 Introduction to FEM-BEM coupling

The boundary element method is presented in this work. Despite all works done to improve the capabilities of the boundary element, the finite element method remains the most efficient for a very large class of situations, including for example those with heterogeneous or non-linear constitutive properties, or finite deformations.

The first formulations allowing FEM-BEM coupling are presented in Zienkiewicz et al. [115] and later in Brebbia and Georgiou [116] or Kelly et al. [117]. The studied domain is subdivided into two sub-domains: a finite element domain and a boundary element domain. The main idea was to construct the stiffness matrix of each such domain and then couple them. Due to the unsymmetrical matrix obtained by using the traditional collocation BEM, numerical difficulties were encountered. Techniques are then proposed to force the symmetry of the matrices, see [115, 118–120]. However, the results obtained by using these techniques are criticized in many works [121–124]. The non-symmetric FEM-BEM coupling is improved in recent works for Laplace equation by Salim et al. in [125], linear elasticity by

Steinbach in [126] and non-linear elasticity by Feischl et al. in [127]. The symmetric Galerkin approach used in this work was proposed for FEM-BEM coupling to avoid the difficulties due to unsymmetrical matrix, see Costabel [24] and Costabel and Stephan [128]. In recent works Rungamornrat and Mear [129] used it for crack analysis in anisotropic media. Nguyen et al. [130] applied it for mode-I planar cracks in elastic media.

In fact two classical coupling approaches are used [115, 116, 131, 132]. In the first (FEM hosted), the BEM sub-domain can be interpreted as a macro finite element that can be assembled easily to the FEM stiffness formulation [57, 133, 134]. Mouhoubi [73], applied this coupling strategy to model fracture problems. Haas et al. [135] applied this strategy for coupling finite shell elements with 3D-SGBEM domains. In the second (BEM hosted), the FEM sub-domain is interpreted as boundary element and the global obtained system is a boundary element system, see [115, 116]. Another interesting possibility is iterative coupling, see for example [136] for 3D transient elastodynamics, [137] for dynamic soil–structure interaction, [138] for wave propagation in 3D multidomains. Recent developments in FEM-BEM coupling can be found in Elleithy and Leong [139] or Gwinner and Stephan [140]. In this work a direct coupling strategy will be used.

5.2.2 Coupling procedure

In this work the BEM hosted approach is used. The stiffness equations of the FEM are converted to BEM-like equations and coupled with those of the BEM while satisfying continuity and equilibrium along the interface. The adopted procedure is described in Margonari [74]. Let us consider the general coupling problem shown in Fig. 5.3. The Galerkin equations of the boundary element zone can be written following the multizone procedure described in subsection 4.1.2, zone A being the boundary element zone:

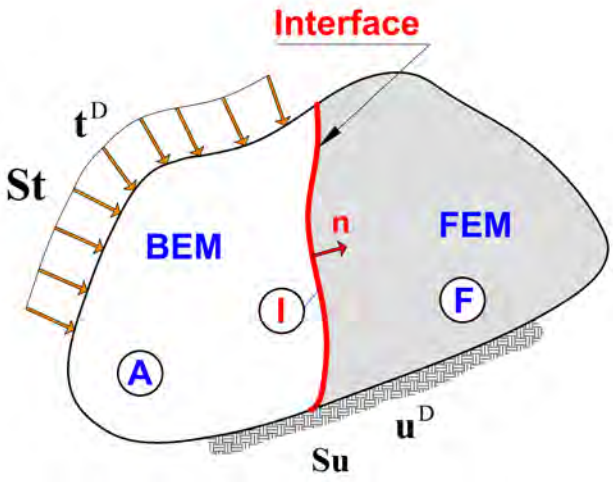


Figure 5.3 – General coupling problem

$$\begin{bmatrix} B_{tt}^{AA} & B_{ut}^{AA} & B_{tt}^{IA} & B_{ut}^{IA} \\ B_{tu}^{AA} & B_{uu}^{AA} & B_{tu}^{IA} & B_{uu}^{IA} \\ B_{tt}^{AI} & B_{ut}^{AI} & B_{tt}^{II} & B_{ut}^{II} \\ B_{tu}^{AI} & B_{uu}^{AI} & B_{tu}^{II} & B_{uu}^{II} \end{bmatrix} \begin{Bmatrix} t^A \\ u^A \\ t^I \\ u^I \end{Bmatrix} = \begin{Bmatrix} \mathcal{F}_u(\tilde{t}^A) \\ \mathcal{F}_t(\tilde{u}^A) \\ \mathcal{F}_u(\tilde{t}^I) \\ \mathcal{F}_t(\tilde{u}^I) \end{Bmatrix} \tag{5.9}$$

The system of equations provided by the finite element method (equation 5.1) can be organized as follows:

$$\begin{bmatrix} K^{II} & K^{IF} & M^I \\ K^{FI} & K^{FF} & 0 \end{bmatrix} \begin{Bmatrix} u^I \\ u^F \\ t^I \end{Bmatrix} = \begin{Bmatrix} F^I \\ F^F \end{Bmatrix} \tag{5.10}$$

where u^F is the unknown displacements of the finite element zone and matrix M^I represents the relationship between tractions and nodal forces on the interface I. Each term m that contributes to the construction of the matrix M^I is given by:

$$m = \int_I N^T N dI \tag{5.11}$$

where N is a matrix of the traditional shape functions. Equations 5.9 and 5.10 can be reorganized in the following system:

$$\begin{bmatrix} B_{tt}^{AA} & B_{ut}^{AA} & B_{tt}^{IA} & B_{ut}^{IA} & 0 \\ B_{tu}^{AA} & B_{uu}^{AA} & B_{tu}^{IA} & B_{uu}^{IA} & 0 \\ B_{tt}^{AI} & B_{ut}^{AI} & B_{tt}^{II} & B_{ut}^{II} & 0 \\ B_{tu}^{AI} & B_{uu}^{AI} & B_{tu}^{II} + M^I & B_{uu}^{II} + K^{II} & K^{IF} \\ 0 & 0 & 0 & K^{FI} & K^{FF} \end{bmatrix} \begin{Bmatrix} t^A \\ u^A \\ t^I \\ u^I \\ u^F \end{Bmatrix} = \begin{Bmatrix} \mathcal{F}_u(\tilde{t}^A) \\ \mathcal{F}_t(\tilde{u}^A) \\ \mathcal{F}_u(\tilde{t}^I) \\ \mathcal{F}_t(\tilde{u}^I) + F_I \\ F^F \end{Bmatrix} \tag{5.12}$$

Let us consider now a particular case when the finite element zone is reduced to the interface. The two sub domains thus form a BEM multizone problem. The multizone equations are then coupled with the interface equations described with FEM. This causes the nullity of the coefficients K^{FF}, K^{FI} and K^{IF}. The zone A matrix in equation 5.12 becomes:

$$\begin{bmatrix} B_{tt}^{AA} & B_{ut}^{AA} & B_{tt}^{IA} & B_{ut}^{IA} \\ B_{tu}^{AA} & B_{uu}^{AA} & B_{tu}^{IA} & B_{uu}^{IA} \\ B_{tt}^{AI} & B_{ut}^{AI} & B_{tt}^{II} & B_{ut}^{II} \\ B_{tu}^{AI} & B_{uu}^{AI} & B_{tu}^{II} + M^I & B_{uu}^{II} + K^{II} \end{bmatrix} \tag{5.13}$$

Due to the equilibrium of the interface described in subsection 4.1.2, the matrix M^I vanished in the global matrix.

5.2.3 Coupling algorithm

For coupling purposes, the finite element stiffness matrix is coupled with the BEM matrix as discussed in subsection 5.2.2. During the iterative resolution by flexible GMRES, algorithm 5 presented in subsection 4.1.3 is modified to take into account the finite element stiffness matrix; a flowchart is presented in Fig. 5.4. The influence of K_{FEM} is added when computing the global Matrix-Vector multiplication as shown in equation 5.14 below:

$$[K]\{x\} = [K_{NEAR}]\{x\} + [K_{FMM}]\{x\} + [K_{FEM}]\{x\}. \tag{5.14}$$

5.3 Validation

Elementary stiffness matrix validation

In this section the finite element implementation will be validated step by step. To validate the finite element implementation, we consider here a simple plate under tension, see Fig. 5.5. The geometrical dimensions are $Lx = 4$, $Ly = 1$, $Lz = 0.02$ and the tension is $\sigma = 1$. The material is homogeneous with $E = 1$ and $\nu = 0.3$.

Only one finite element is considered to simulate the problem. The stiffness matrix is then computed and compared to the one computed with Cast3m. The computed stiffness matrix is presented below as well as the relative difference compared to the stiffness matrix given by Cast3m. The maximum relative difference is $3,3.10^{-5}$. After resolution of the system, the displacements U_x and U_y are also compared to those given by Cast3m, see Table 5.1. In fact, it is difficult to print the Cast3m stiffness matrix with high precision. The observed differences are thus due to truncation as shown in Table 5.1.

$$\begin{bmatrix}
1,21.10^{7} & . & . & . & . & . & . & . \\
3,57.10^{6} & 2,99.10^{7} & . & . & . & . & . & . \\
3,30.10^{6} & 2,75.10^{5} & 1,21.10^{7} & . & . & . & . & . \\
-2,75.10^{5} & 1,40.10^{7} & -3,57.10^{6} & 2,99.10^{7} & . & . & . & . \\
-6,04.10^{6} & -3,57.10^{6} & -9,34.10^{6} & 2,75.10^{5} & 1,21.10^{7} & . & . & . \\
-3,57.10^{6} & -1,50.10^{7} & -2,75.10^{5} & -2,90.10^{7} & 3,57.10^{6} & 2,99.10^{7} & . & . \\
-9,34.10^{6} & -2,75.10^{5} & -6,04.10^{6} & 3,57.10^{6} & 3,30.10^{6} & 2,75.10^{5} & 1,21.10^{7} & . \\
2,75.10^{5} & -2,90.10^{7} & 3,57.10^{6} & -1,50.10^{7} & -2,75.10^{5} & 1,40.10^{7} & -3,57.10^{6} & 2,99.10^{7}
\end{bmatrix}$$

Computed stiffness matrix

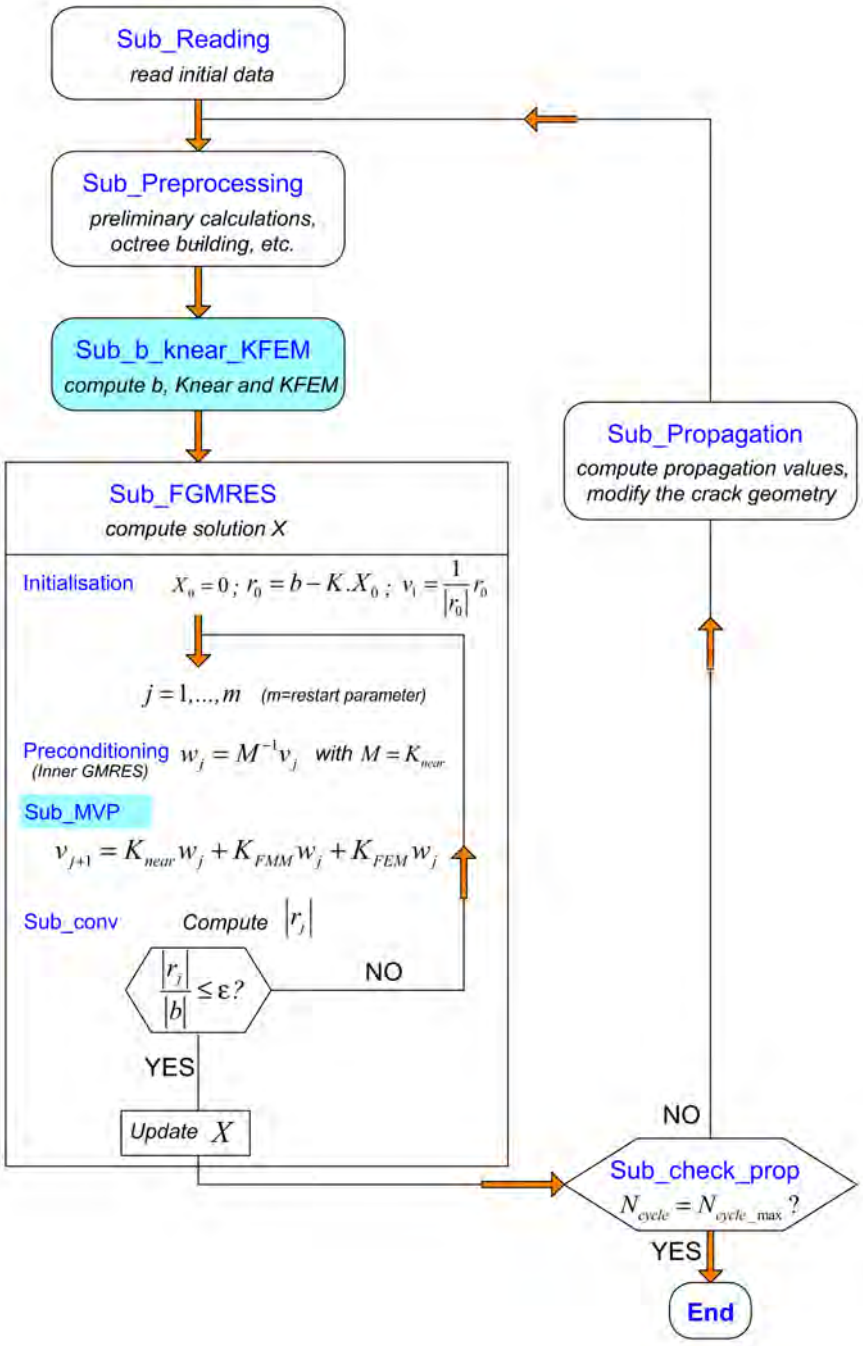


Figure 5.4 – Coupling algorithm

$$\begin{bmatrix} 9,1 & . & . & . & . & . & . & . \\ 6,2 & 0,0 & . & . & . & . & . & . \\ 0,9 & 19,4 & 9,1 & . & . & . & . & . \\ 19,4 & 2,6 & 6,2 & 0,0 & . & . & . & . \\ 9,1 & 6,2 & 6,2 & 19,4 & 9,1 & . & . & . \\ 6,2 & 33,4 & 19,4 & 18,5 & 6,2 & 0,0 & . & . \\ 6,2 & 19,4 & 9,1 & 6,2 & 0,9 & 19,4 & 9,1 & . \\ 19,4 & 18,5 & 6,2 & 33,4 & 19,4 & 2,6 & 6,2 & 0,0 \end{bmatrix} \times 10^{-6}$$

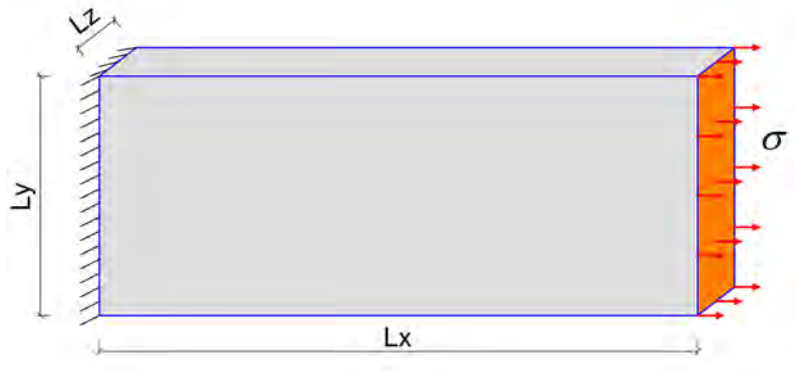


Figure 5.5 – Plate geometry

Relative difference compared to Cast3m

Table 5.1 – FEM validation: plate with one finite element

	Cast3m	*MBEMv3.0*	Relative difference
U_x	$0,00390195$	$0,0039019490171601$	$2,52.10^{-7}$
U_y	$0,00021829$	$0,0002182908628012$	$6,29.10^{-7}$

Global matrix validation

To simulate a real problem, several finite elements are generally used, and the elementary matrices are assembled together. To validate the global matrix assembly, we reconsider the plate under tension of Fig. 5.5. 64 finite elements are used for the simulation, see Fig. 5.6. The global matrix is also in very good agreement with the one given by Cast3m. After resolution of the system, the displacement U_x is compared the one given by Cast3m, see Table 5.2. The relative difference is less than 1% except at position $x = 0,50$.

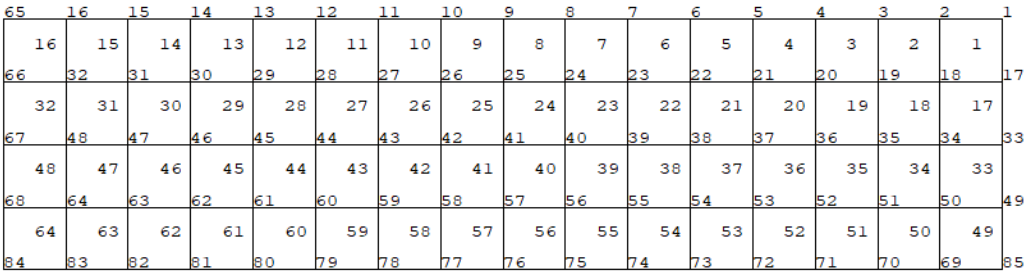


Figure 5.6 – Plate meshed with 64 finite elements

Table 5.2 – FEM validation: plate with 64 finite elements

x	Node	Cast3m	*MBEMv3.0*	Difference (%)
0,25	16	$2,63.10^{-4}$	$2,61.10^{-4}$	0,60
0,50	15	$4,99.10^{-4}$	$4,82.10^{-4}$	3,31
0,75	14	$7,39.10^{-4}$	$7,32.10^{-4}$	0,98
1,00	13	$9,85.10^{-4}$	$9,83.10^{-4}$	0,27
1,25	12	$1,23.10^{-3}$	$1,23.10^{-3}$	0,16
1,50	11	$1,48.10^{-3}$	$1,48.10^{-3}$	0,15
1,75	10	$1,73.10^{-3}$	$1,73.10^{-3}$	0,14
2,00	9	$1,98.10^{-3}$	$1,98.10^{-3}$	0,13
2,25	8	$2,23.10^{-3}$	$2,23.10^{-3}$	0,12
2,50	7	$2,48.10^{-3}$	$2,48.10^{-3}$	0,10
2,75	6	$2,73.10^{-3}$	$2,73.10^{-3}$	0,09
3,00	5	$2,98.10^{-3}$	$2,98.10^{-3}$	0,09
3,25	4	$3,23.10^{-3}$	$3,23.10^{-3}$	0,08
3,50	3	$3,48.10^{-3}$	$3,48.10^{-3}$	0,07
3,75	2	$3,73.10^{-3}$	$3,73.10^{-3}$	0,07
4,00	1	$3,98.10^{-3}$	$3,98.10^{-3}$	0,06

Elliptical membrane simulation

Let us consider now an elliptical membrane of dimensions $a1 = 2.0$, $b1 = 1.0$, $rx = 1.25$, $ry = 1.75$ and thickness $Lz = 0.02$ submitted to a linear force on one side, see Fig. 5.7. The membrane is meshed with 24 finite elements, see Fig. 5.8. The computed displacements U_x and U_y are compared to those given by Cast3m, see Table 5.3 and 5.4. The obtained results on the elliptical membrane are validated since the maximum relative difference is $6,7.10^{-5}$.

Table 5.3 – Elliptical membrane: U_x validation

Node	Cast3m	*MBEMv3.0*	Relative difference
21	0,00	0,00	$--$
10	$1,04.10^{-4}$	$1,04.10^{-4}$	$2,0.10^{-7}$
9	$1,65.10^{-4}$	$1,65.10^{-4}$	$6,1.10^{-6}$
8	$1,55.10^{-4}$	$1,55.10^{-4}$	$1,8.10^{-6}$
7	$8,24.10^{-5}$	$8,24.10^{-5}$	$4,1.10^{-8}$
6	$-6,92.10^{-6}$	$-6,92.10^{-6}$	$6,7.10^{-5}$
5	$-4,68.10^{-5}$	$-4,68.10^{-5}$	$1,4.10^{-5}$

Reinforced beam

In this application, we consider a homogeneous beam ($E = 22GPa$) of dimensions $L = 6000mm$, $b = 200mm$ and $h = 500mm$ submitted to a linear force $p =$

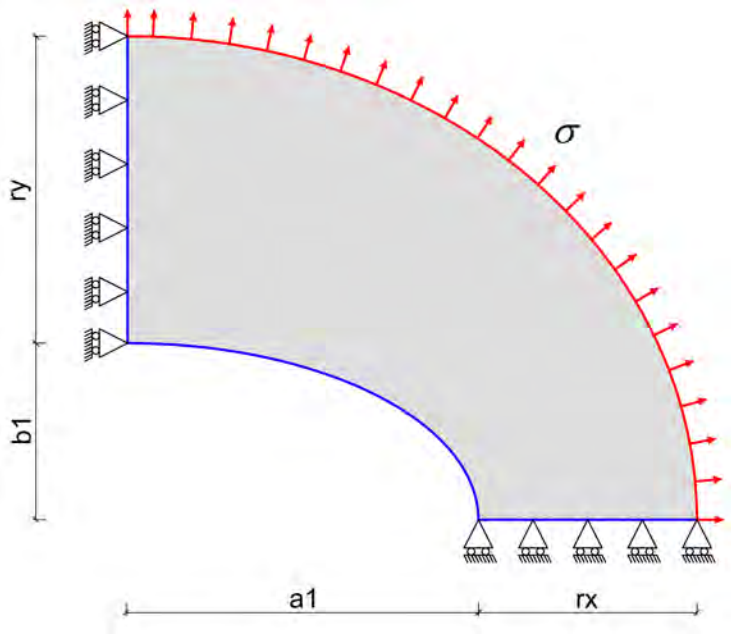


Figure 5.7 – Elliptical membrane geometry

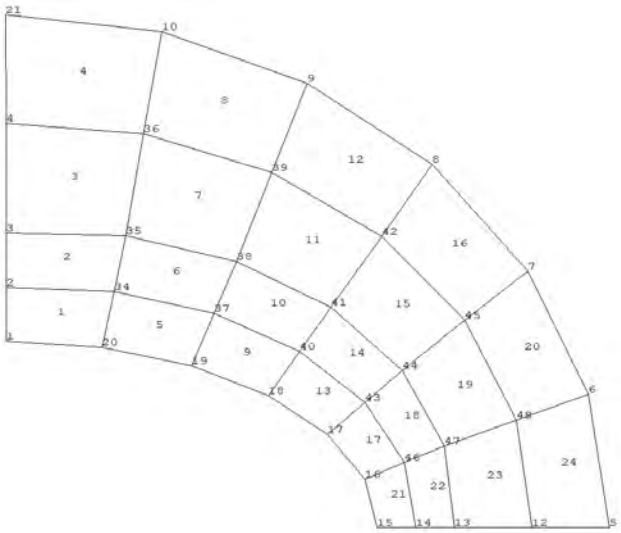


Figure 5.8 – Elliptical membrane mesh

$200kN/m$, see Fig. 5.9. The beam is simulated with *MBEMv3.0* by taking a mesh of 1 200 boundary elements, see Fig. 5.10. We then consider a composite reinforcement ($E_f = 200GPa$) on the underside of the beam. This reinforcement is modeled as a membrane element with finite element method. Fig. 5.11 and 5.12 show the effect of the reinforcement on the displacements for two values of the reinforcement thickness $Ep = 1.2mm$ and $Ep = 2.5mm$. Although the membrane has no flexural rigidity, the displacement reduction that it leads to is noticeable.

Table 5.4 – Elliptical membrane: U_y validation

Node	Cast3m	*MBEMv3.0*	Relative difference
21	$5,06.10^{-4}$	$5,06.10^{-4}$	$3,0.10^{-6}$
10	$4,61.10^{-4}$	$4,61.10^{-4}$	$1,9.10^{-6}$
9	$3,41.10^{-4}$	$3,41.10^{-4}$	$3,3.10^{-6}$
8	$1,88.10^{-4}$	$1,88.10^{-4}$	$6,2.10^{-6}$
7	$6,13.10^{-5}$	$6,13.10^{-5}$	$4,9.10^{-6}$
6	$1,16.10^{-6}$	$1,16.10^{-6}$	$2,4.10^{-5}$
5	$0,00$	$0,00$	$--$

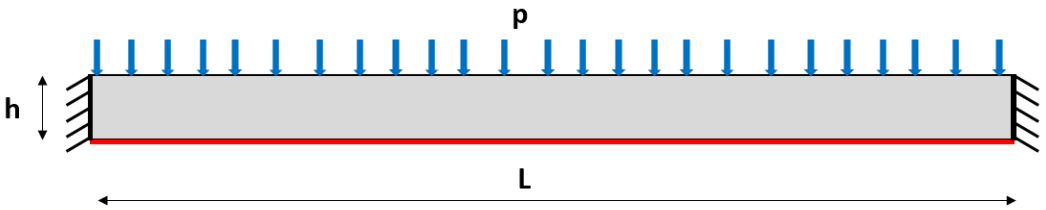


Figure 5.9 – Reinforced beam: geometry

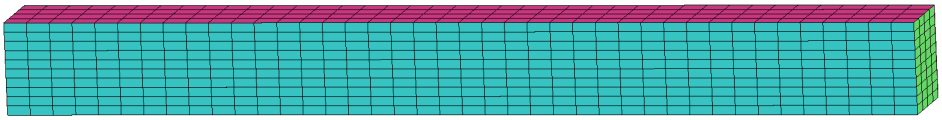


Figure 5.10 – Reinforced beam: mesh

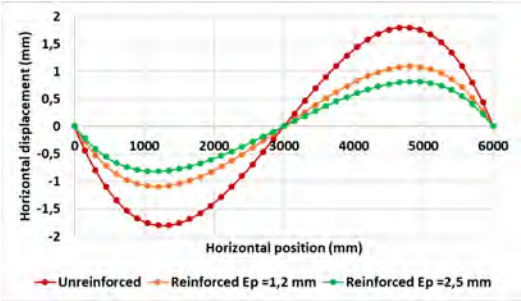


Figure 5.11 – Reinforced beam: Ux

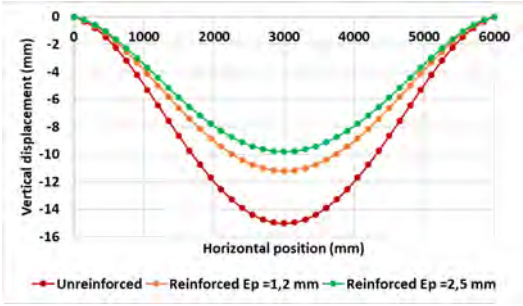


Figure 5.12 – Reinforced beam: Uz

5.4 Conclusions

In this chapter FEM-BEM coupling is investigated. The finite element formulation is presented. The membrane finite element is implemented and validated. Classical coupling strategies are briefly reviewed, and a direct coupling is introduced. This study mainly permits to show that our BEM code can be extended to take advantage of finite element possibilities. Other applications using the coupled code will be discussed in next chapter.

Chapter 6

Application to Road Structures

Contents

Pavement systems are vital elements in the infrastructure network of all societies. The BEM is well-suited to model the semi-infinite boundaries associated with layered pavement systems and has the benefit of dimension reduction. Besides, the method is also capable of accounting for the presence of cracks and crack propagation with less computational effort. In this chapter, cracked pavement structures will be simulated including the effect of fiberglass grid reinforcement.

6.1 Reinforced pavement modeling

6.1.1 Fiberglass grid reinforcement

To delay the development of reflective cracking, water infiltration and development of fatigue cracking, various stress-absorbing interlayers have been used such as: styrene-butadiene-styrene (SBS)-modified asphalt sand concrete interlayer, asphalt-rubber sand concrete interlayer and stress absorbing membrane interlayer (SAMI). Since a few decades ago, fiber composite materials have been used to reinforce road pavements. Among these materials, glass fiber grids are the most used, and currently one of the most efficient maintenance systems.

The glass fiber grid is a material composed of glass fiber filaments with a diameter of 10 to 30 microns, assembled using a synthetic resin. Glass fiber presents interesting properties as a reinforcing material: it is strong and flexible, the Young modulus is approximately 70 GPa, see Darling and Woolstencroft [141]. Many studies have been conducted to evaluate cracking resistance of glass fiber grid based reinforcement systems, see for example [142–147]. A review of glass fiber grid use for pavement reinforcement is presented by Nguyen et al. [148]. Recent works can be found in [149–153].

In practice, the glass fiber grid is placed in the lower part of the bituminous layers. Considering its functions, it is often provided at the interface between the base layer and the bituminous asphalt surface course. Glass fiber grids can be used for localized pavement repairs (reinforcement of concrete joints, crack repair) or for full width coverage of the entire pavement. They can be used in all climates (cold, temperate and hot climate zones) and geographic areas, performing equally well in desert conditions and in near arctic regions that are subject to intense cold and seasonal temperature fluctuations [141].

The SolDuGri project (see Godart et al. [154] or Chazallon et al. [155]) which supports this study permits to industrialists and public researchers to work together for a better understanding of fiberglass grid effects. A glass grid associated with a light non-woven polyester is presented in Fig. 6.1. An example of fiberglass grid installation is shown in Fig. 6.2.

6.1.2 Simulation of reinforced pavement

Three main types of numerical tools allow the study of the influence of grids in reinforced pavement. First, there are several numerical studies to replicate laboratory experiments. These laboratory studies consist of studying the effect of grids by modeling the behavior of reinforced layer on specimens, see Bacchi [146] or Arsenie [147] in which Bodin [156] and Castro-Sanchez [157] models are used to simulate two-dimensional beams with Cast3m.

Secondly, there are simplified models developed by companies and universities. These models are based on empirical data and on numerical simulation results. Vanelstraete et al. [158] developed BITUFOR, based on a three-dimensional finite element analysis of various structures with steel reinforced grids. The Notting-

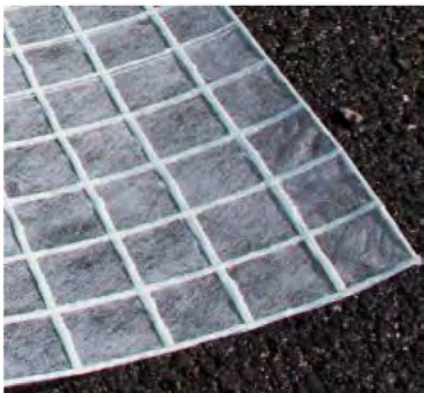


Figure 6.1 – Glass grid associated with a light non-woven polyester

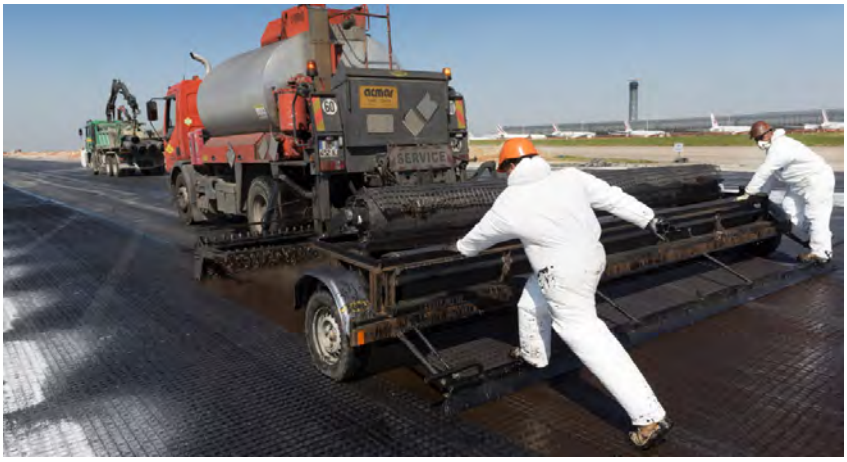


Figure 6.2 – Glass fiber grid installation at Roissy CDG (COLAS)

ham university developed OLCRACK/THERMCR [159], predictive programs for overlay design. They are capable of replicating test results from beams on semi-continuous support and from the pilot scale pavement at Nottingham University. Ooms Nederland Holding developed ARCDESO [160] (Anti-Reflective Cracking Design Software) a mechanistic empirical program that deals with reflective cracking in asphalt layers.

Thirdly, there is the simulation of the whole reinforced cracked structure. This aspect, which we are interested in here, is not much concerned in the literature. Most of the work concerning the simulation of the whole reinforced cracked structure use two-dimensional finite element analysis or axial symmetric analysis, see [151, 161–165]. They are, therefore similar to the simplified models. In 3D work, the cracks are almost always modeled using a non-local approach, [166–168]. For example, Coni and Bianco [169] investigated the crack propagation process in the presence of steel reinforcement based on a finite element using ANSYS. Baek and Al-Qadi [170] used three-dimensional finite element modeling to investigate the fracture behavior of hot-mix asphalt (HMA) overlays by using a bi-linear cohesive

zone model.

In following, 3D crack propagation will be studied in pavement structure by using a local approach. The case of reflective cracking and the influence of fiberglass grids will be also included.

6.1.3 Simulation of thin structures by BEM

There is a major difficulty when the commonly used boundary integral equations (BIEs) are applied directly to thin-body problems (including thin voids or open cracks, thin shell-like structures and thin layered structures), where two parts of the boundary become close to each other, see Krishnasamy et al. [171] and Martinez [172]. This difficulty is the nearly singular integrals which arise in such problems. Nearly singular integrals are not singular in a mathematical sense. However, from the point of view of numerical integration, these integrals can not be calculated accurately by using the conventional numerical quadrature since the integrand oscillates very fiercely within the integration interval. Although that difficulty can be overcome by using very fine meshes, the process requires too much CPU time.

Many works are devoted to deriving convenient integral forms or sophisticated computational techniques for calculating the nearly singular integrals. One approach is to avoid calculating the nearly singular integrals by establishing new regularized BIE [173–176]. Another approach is the direct calculation of the nearly singular integrals by various methods such as interval subdivision method (Jun et al. [177]), special Gaussian quadrature method (Lutz [178]), exact integration method (Niu et al. [179], Zhang and Sun [180]), and various nonlinear transformation methods (Huang and Cruse [181], Ma and Kamiya [182], Zhang and Gu [183], Zhang and Sun [184]). Recent works on nearly singular integrals can be found in Lenoir and Salles [185] for an explicit method based on a recursive dimension reduction, Li and Su [186] for *sinh* transformation and Han and al. [187] for a semi-analytical evaluation in isogeometric BEM.

In this work, even though the treatment of nearly singular integrals is very interesting, we choose to apply finite element method to model the grids. The behavior of grids is very similar to that of a membrane. So, the FEM-BEM coupling presented in the last chapter can then be applied. The grid is modeled as a homogeneous membrane finite element and the rest of the structure is modeled with the BEM.

6.1.4 Simulation of thin structures by FEM-BEM coupling

The coupling procedure described in subsection 5.2.2 whose algorithm is presented in 5.2.3 is used here to simulate the reinforced pavement. The model is presented in Fig. 6.3. The thickness of the grid is taken equal to $t = 1mm$ and the stiffness equal to 70 GPa.

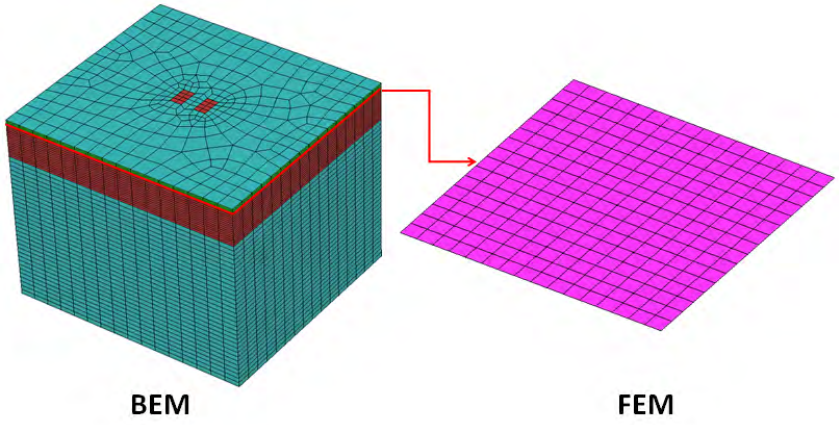


Figure 6.3 – Reinforced road model

6.2 A reference pavement

6.2.1 Presentation

We recall here a three-layered pavement structure that Trinh [48] attempted to simulate with *MBEMv2.0*. The characteristics of the layers are shown in table 6.1 and can be visualized in Fig.6.4. For simplicity purposes, the contact area of the wheel on the road surface is supposed to be a rectangle of dimensions 180 × 300 mm, as depicted in the Fig.6.6. The vertical load of these wheels is 65 kN which is equal to the distributed load of $p = 0,6\ MPa$. The dimensions of the boundary element mesh are chosen as $3555 \times 3300 \times 2786\ mm$ (following respectively three directions x,y and z) which represent the entire model (loaded by one half-axle), see Fig.6.5. This boundary mesh is composed of 4 276 four-node quadrilateral elements which generate overall about 11 000 unknowns in displacement and in traction. The pavement deflection subjected to half-axle load is validated in [48] by comparison with the numerical solution produced by finite element method modeled with CAST3M by Chazallon [188].

Table 6.1 – Pavement characteristics

Layer	Layer Constitution	Thickness (mm)	E (MPa)	μ
1	Asphalt concrete	66	6610	0,35
2	Unbound granular base course	500	180	0,3
3	Subgrade	2220	80	0,25

6.2.2 Deflection of fractured pavement

We consider N_c rectangular cracks (3 000×60 mm^2) embedded in the first layer of the road. Each crack is meshed with 120 eight-noded elements and 429 nodes (Fig. 6.7a).

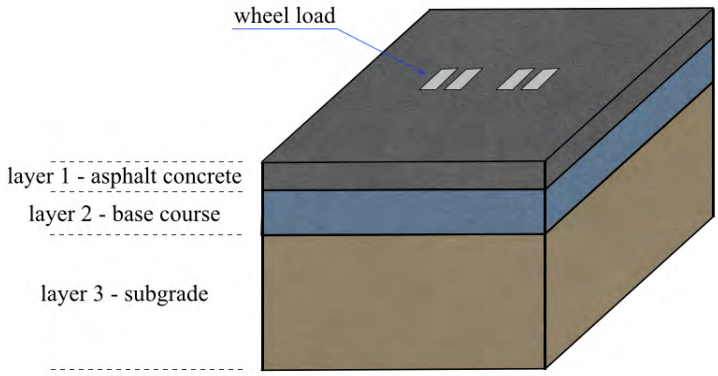


Figure 6.4 – Modeling of a 3-layer pavement

The cracks are embedded in the first thinnest layer; their spatial arrangement is shown in Figs 6.7b,c.

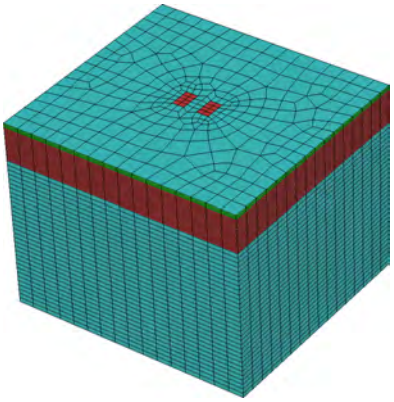


Figure 6.5 – Road pavement model

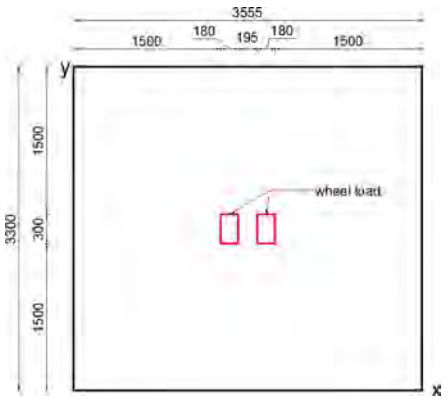
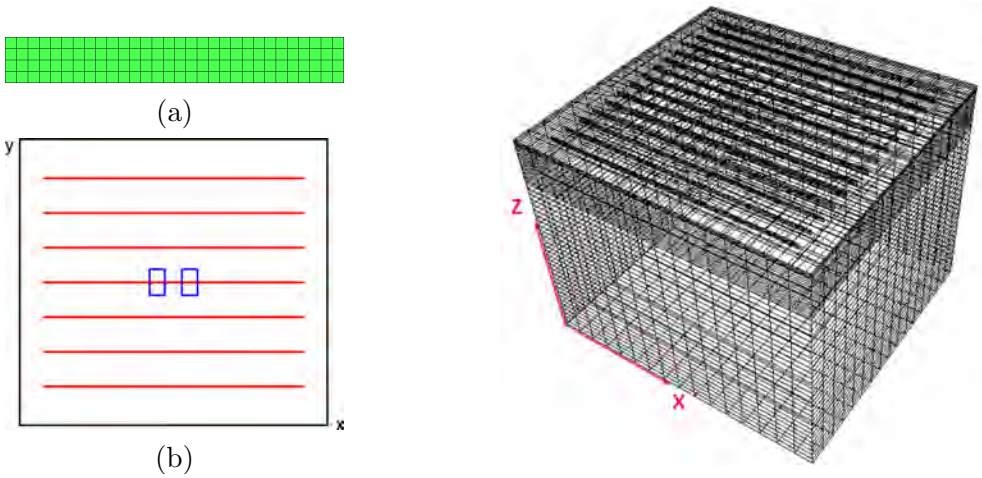


Figure 6.6 – Road pavement geometry

Trinh [48] did not succeed to simulate this problem with *MBEMv2.0*. Even with only one crack, *MBEMv2.0* still could not converge after 1 000 iterations with a stopping criterion of 10^{-3}. In fact, several unfavorable factors lead to an ill-conditioned matrix and subsequently limit the convergence rate of the code. Some investigations were performed on the simple pavements (without crack). The identified unfavorable factors were:

- the small thickness of the surface course,
- the large material stiffness ratio,
- the high concentration of elements in the first thin layer.

Even though these factors are unfavorable, one must deal with them since they represent the real pavement structure. With *MBEMv3.0*, this real problem is simulated with all the unfavorable factors without issues. Fig. 6.8 shows the deflection of the cracked pavement with N_c from 1 to 30. Table 6.2 shows computational data



(a) (b) (c)

Figure 6.7 – Fractured pavement: (a) Rectangular crack mesh (b) Positions of rectangular cracks (c) Pavement with 15 rectangular cracks

for FM-SGBEM analyses, with N_c ranging from 1 to 30. While the simulation of one crack is impossible for *MBEMv2.0*, 30 cracks are simulated with *MBEMv3.0* in only 24 minutes. This demonstrates once again the great superiority of *MBEMv3.0* over *MBEMv2.0*.

Table 6.2 – Three-layered road with stationary cracks: computational data

#	N_c	N_{dofs}	T_{pre} (s)	N_{iter}	T_{sol} (s)	T_{tot} (s)
1	1	15 669	121	69	283	404
2	3	18 243	151	70	355	507
3	7	23 391	148	65	366	515
4	15	33 687	210	67	635	846
5	30	52 992	396	79	1 416	1 813

6.2.3 Crack propagation in pavement

In this application, we consider initially penny-shaped cracks (with radius $r = 10$ mm and the distance $d_c = 250$ mm between them) embedded in the heterogeneous road (Fig. 6.5 and 6.6), whose characteristics are again given in Table 6.1 . The crack centers are located on the mid-plane of the first layer (Fig. 6.9). Fatigue propagation is computed with $N_{\text{cycles}} = 10$ and $\Delta a^{max} = r/4$. The material properties for the Paris law are $A = 10^{-8} mm/cycle$ and $m = 4.5$. Table 6.3 presents computational data for the propagation analyses. The final shape of the propagated cracks and the crack opening displacement in z direction are shown in Fig. 6.10 for simulation 1 (one crack) and in Fig. 6.11 for simulation 2 (three cracks).

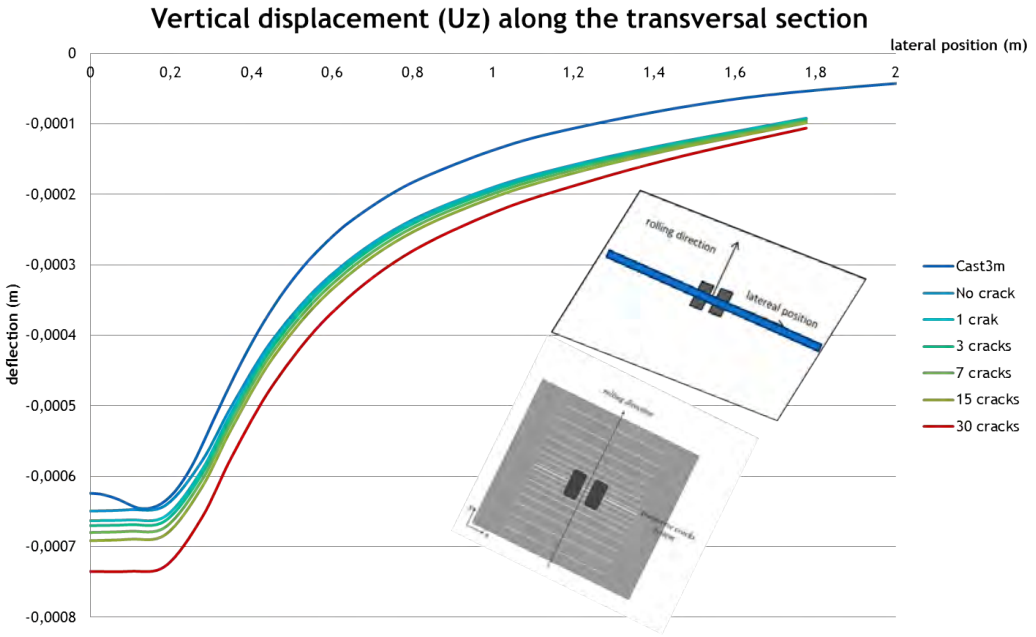


Figure 6.8 – Deflection of cracked pavement: transverse section

Table 6.3 – Three-layered road with multiple propagating cracks: computational data

#	N_c	N_{init}	N_{end}	T_{pre}(s)	T_{sol}(s)	T_{tot}(s)
1	1	15 441	18 033	201	667	883
2	3	17 559	25 335	345	1 382	1 749
3	5	19 677	32 637	1 241	3 166	4 434

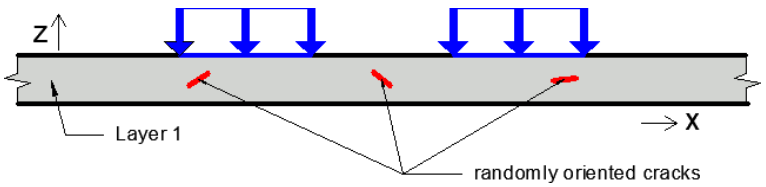


Figure 6.9 – Initially penny-shaped cracks in top layer of three-layered road

Reflective cracking

A very interesting case is the reflective crack. Due to the repeated stress concentration, a crack starts in the overlay of an asphalt pavement and then propagates to the outer surface. This can also happen in overlays placed on joints or cracks in concrete pavements. It can affect the general performance and durability of the pavement. Let us consider a vertical penny shaped crack in longitudinal direction, located in the first layer of the studied pavement (Fig. 6.12). Under the load, the

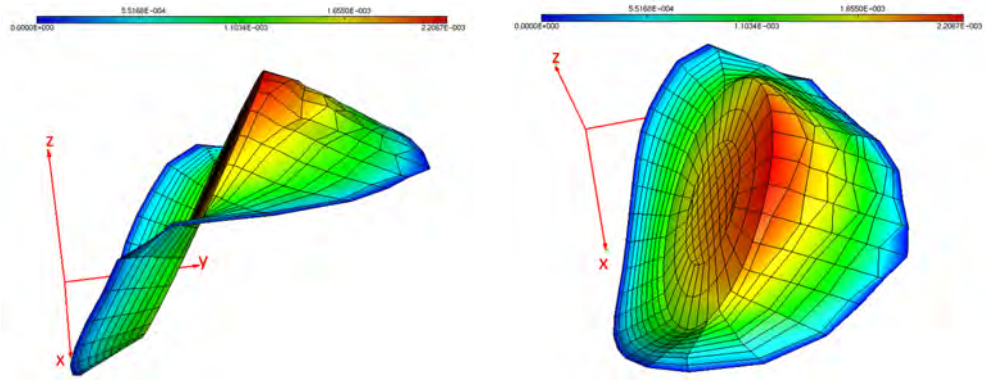


Figure 6.10 – Propagation of single crack in three-layered road: COD in z direction (views from two directions)

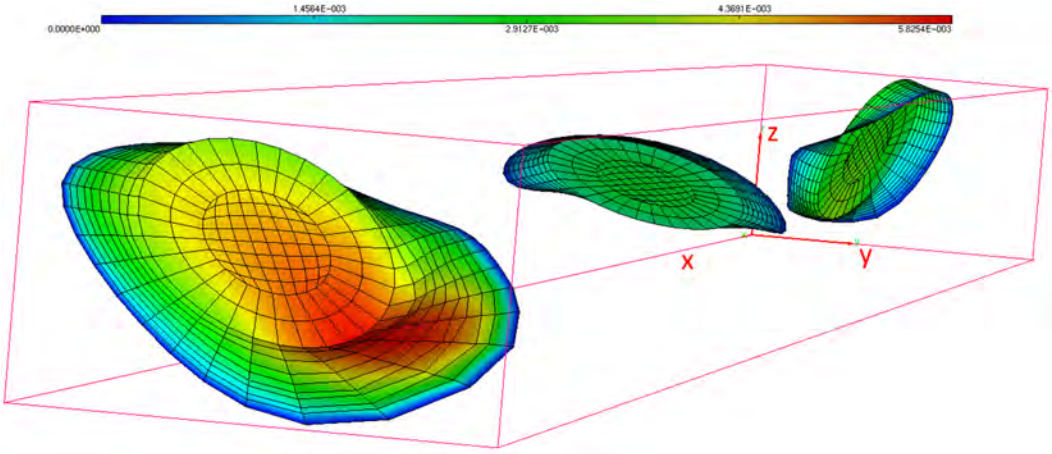


Figure 6.11 – Propagation of multiple cracks in three-layered road: COD in z direction

crack propagates to the outer surface as shown in Fig. 6.13.

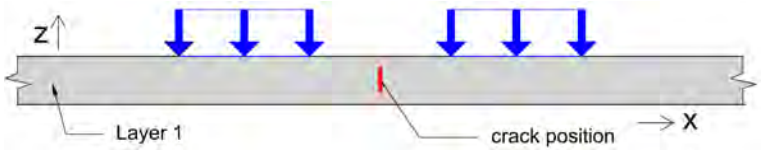


Figure 6.12 – Reflective cracking: Initially penny-shaped crack

6.2.4 Reinforced pavement

We now consider that the pavement is reinforced by a composite grid. The reinforced layer can be taken homogeneous ($E_f = 40GPa$ and $h = 1mm$). The reinforced layer is located at the interface between the base course (layer 2) and the

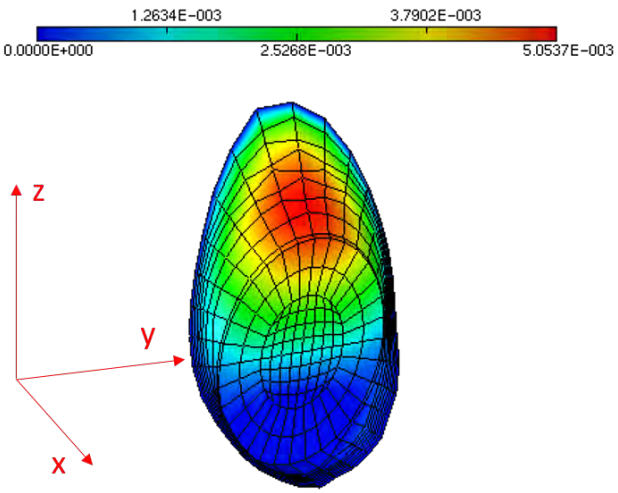


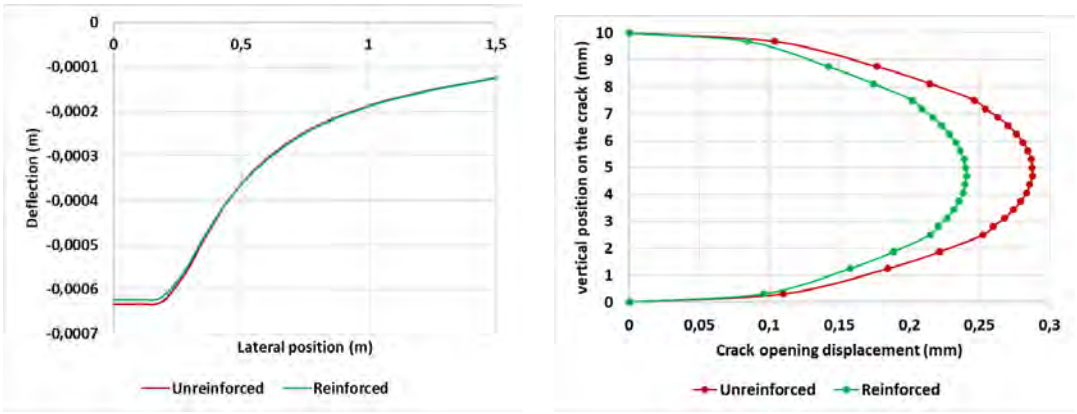
Figure 6.13 – Reflective cracking propagation: COD

asphalt concrete layer (layer 1). The stiffness of the composite grid is calculated with finite element method and coupled with the BEM system. Fig. 6.14a presents a comparison between the deflection of the unreinforced pavement and that of the reinforced pavements. This figure permits us to conclude that the fiberglass grid does not have a significant effect on pavement deflection. Experimental studies also come to the same conclusion, see Nguyen et al. [148]. Now, let us study the fiberglass grid effect on reflective cracking. For this purpose, a vertical crack (see Fig. 6.12) of radius $r = 5mm$, in longitudinal direction, is positioned $dz = 5mm$ above the grid. An elastostatic simulation is performed and the crack opening displacement is compared between the unreinforced pavement and the reinforced one. This comparison is presented in Fig. 6.14b. We can notice that the fiberglass grid reduces the COD values. By doing so, it delays reflective cracking. It can then be concluded that fiberglass grids can extend the fatigue life of pavements.

6.3 SolDuGri pavement

6.3.1 Presentation

This pavement is the typical pavement structure tested for SolDuGri project with the IFSTTAR accelerated pavement testing facility in Nantes which is an outdoor installation dedicated to full-scale pavement experiments. The pavement has four layers, is divided in six study sections and is subjected to thickness variation. To simplify the model, the characteristics and the thickness of the layers are kept constant and shown in Table 6.4. The fiberglass grid is inserted between the two first asphalt layers. The boundary mesh is similar to that of the reference pavement structure.



(a) (b)

Figure 6.14 – Composite reinforced layer $E_f = 40GPa$ at $z = 66mm$ in reference road (a) Effect on deflection (b) Effect on COD

Table 6.4 – Pavement characteristics

Layer	Layer Constitution	Thickness (mm)	E (MPa)	μ
1	Asphalt concrete	60	11 364	0,35
2	Asphalt concrete	50	11 364	0,35
3	Granular base course	300	400	0,35
4	Subgrade	2600	200	0,35

6.3.2 Unreinforced pavement

A simple elastostatic test is first performed on the pavement structure. The solution provided by *MBEMv3.0* is compared to the numerical solution produced by the finite element method modeled with Cast3m. The following diagrams (Fig.6.15 and 6.16) show the deflection of the models respectively across and along the rolling direction of the studied models under the effect of half-axle loading. The exhibited results from the boundary analysis correspond very well with the output from Cast3m (the region of the most significant deflection - left half of the diagrams). The difference between the BEM and FEM results in the right half of these diagrams is also observed by Trinh [48] with the reference pavement. The effect of multiple cracks on the deflection and the simulation of crack propagation led to results similar to those presented in the previous section for the reference pavement structure.

6.3.3 Reinforced pavement

We now consider that the pavement is reinforced by a fiberglass grid. The reinforced layer can be taken homogeneous ($E_f = 16.8GPa$ and $h = 1mm$). The grid is located $z = 60mm$ at the interface between the asphalt concrete layers (layer 1

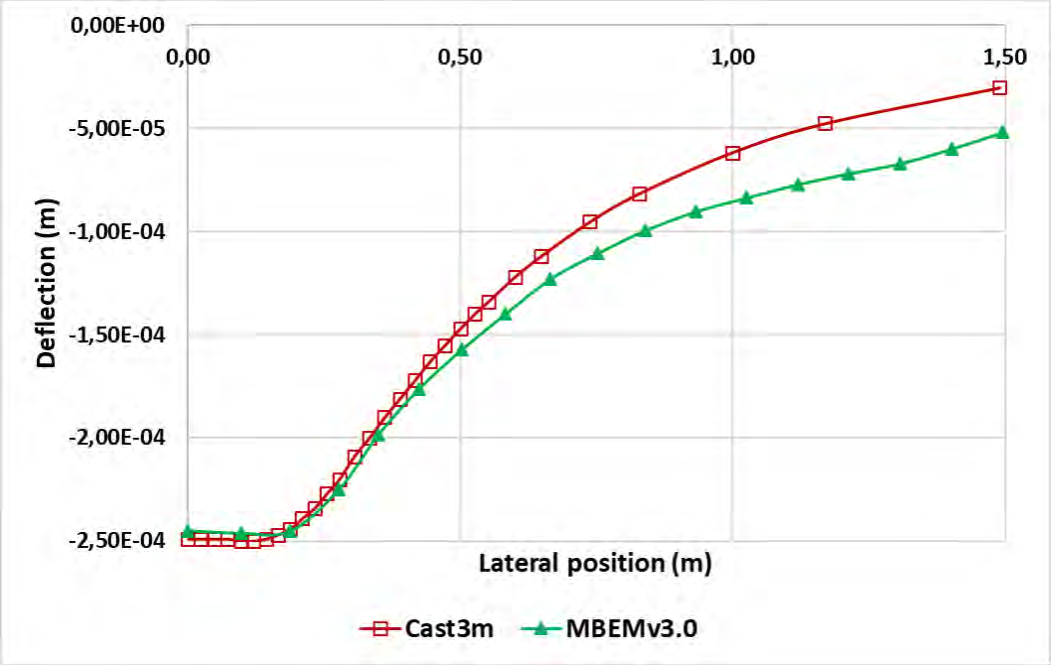


Figure 6.15 – Deflection: transverse section

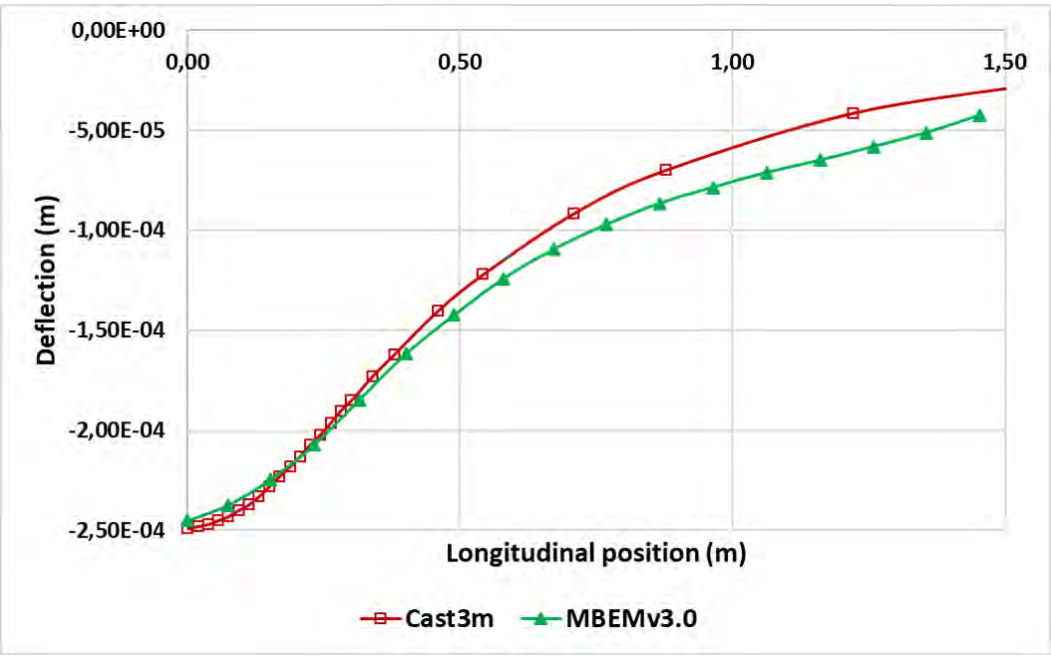
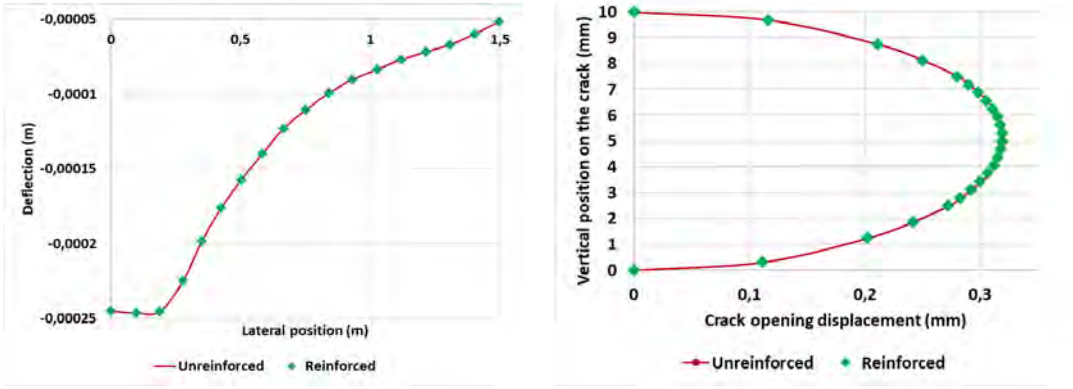


Figure 6.16 – Deflection: longitudinal section

and 2 of Table 6.4). The stiffness of the reinforced layer is calculated with finite element method and coupled with the BEM system. Fig. 6.17a presents a comparison between the deflection of the unreinforced pavement and that of the reinforced pavements. As for the reference pavement, the fiberglass grid does not have a significant effect on the deflection. In this simulation there is almost no influence because the stiffness of the fiberglass grid reinforced layer $E_f = 16.8GPa$ is close to that of the asphalt concrete layer $E = 11.4GPa$. The observation is the same for the influence on cracking: almost no influence is noticed. The obtained results for reflective cracking presented in Fig. 6.17b concern a vertical penny-shaped crack of radius $r_c = 5mm$ in longitudinal direction, located $dz = 5mm$ above the fiberglass grid. An influence can be observed by increasing the stiffness of the fiberglass grid reinforced layer or by decreasing the asphalt layer stiffness. We present in Table 6.5 the reduction of COD observed by varying the stiffness of the fiberglass grid reinforced layer. The COD reduction presented is that observed at the crack center. This table also presents the influence of the crack size. It is noticed that when the fiberglass grid is less effective when the crack size increase. It is therefore necessary to monitor road pavements in order to consider fiberglass grid reinforcement at the earliest. In all these cases the COD reduction is not significant. This is because the grid is not at its optimum position.

Table 6.5 – COD reduction: influence of the crack size and of the stiffness of the reinforced layer

	$E_f = 16.8GPa$	$E_f = 40GPa$	$E_f = 70GPa$
$r_c = 5mm$	0.50 %	2.01 %	2.71 %
$r_c = 15mm$	0.63 %	1.13 %	1.65 %
$r_c = 25mm$	0.27 %	0.66 %	1.10 %



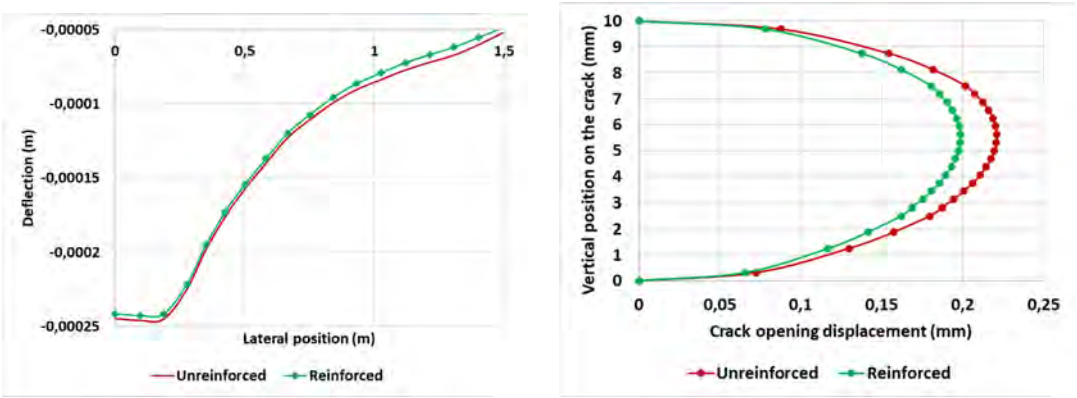
(a) (b)

Figure 6.17 – SolDuGri, $z = 60mm$, $E_f = 16.8GPa$ (a) Effect on deflection (b) Effect on COD

To be effective, the grid should be placed where high tensile stresses are acting. Virgili et al. [189] also come to this conclusion and indicate that fiberglass grids begin to work only when high tensile stress is reached at the interface. However, Kerzrého et al. [190] demonstrated that, in case of a cracked bituminous pavement which still shows a structural capacity, the section reinforced with an open glass grid placed under a thin asphalt overlay (25 mm) performs better than two unreinforced sections (one with the same overlay thickness and one with a thicker overlay). For validation purposes, let us place the fiberglass grid at $z = 110mm$ the interface between the second asphalt concrete layer and the granular base course (layer 2 and 3 of Table 6.4). Crack radius is taken $r_c = 5mm$ and the reductions observed are presented in Table 6.6. The contribution of the fiberglass grid reinforced layer $E_f = 40GPa$ is shown in Fig. 6.18a for the deflection and in Fig. 6.18b for the cracking. We can conclude that, to make full use of the benefits of fiberglass grids, a minimum covering layer is necessary. The thickness of the over-layer depends on the stiffness of the fiberglass grids and the structural characteristics (layers, stiffness, load, etc.) of the pavement.

Table 6.6 – Influence of fiberglass position on COD reduction

	$E_f = 16.8GPa$	$E_f = 40GPa$	$E_f = 70GPa$
$z = 60mm$	0.50 %	2.01 %	2.71 %
$z = 110mm$	4.87 %	10.53 %	17.22 %



(a) (b)

Figure 6.18 – SolDuGri, $z = 110mm$, $E_f = 40GPa$ (a) Effect on deflection (b) Effect on COD

6.4 Conclusions

In this chapter, the new version of our code (*MBEMv3.0*) based on symmetric Galerkin BEM coupled with the fast multipole method has been applied to pavement study. The issues encountered by *MBEMv2.0* are solved. Multiple cracked pavements are simulated in an acceptable duration. Crack propagation is also studied in heterogeneous pavement structure. The special case of reflective cracking is also simulated. This demonstrates the superiority of the present version of the code over the previous one. Finally, the effects of fiberglass grids on reflective cracking are studied by FEM-BEM coupling. We can conclude that to make full use of fiberglass grid benefits, a minimum covering layer is necessary. The optimum characteristics (thickness, stiffness) of the fiberglass grids and the overlayer to delay cracking can be found by simulation with *MBEMv3.0*.

Chapter 7

Conclusions and directions for future work

Contents

7.1 Conclusions

The main aim of this work was to develop a new version of an existing boundary element code to efficiently simulate crack propagation in engineering structures. The basic foundations of the code are introduced in *chapter 2* including some numerical aspects. The main contributions of this work can be divided into three parts.

The first part concerns performance enhancement. Multiple strategies have been proposed and implemented in order to improve the performance of the existing code. These strategies are discussed in *chapter 3*. A data reusing technique has permitted to reduce the duration of the matrix computation phase and non-zero initial guess is used to reduce the iterative solver phase. Then the code is adapted to take advantage of multi-core environment. After identification of time-consuming parts and reorganization, shared memory parallelism is achieved using OpenMP directives. We notice peaks of memory usage during the matrix computation phase, especially with the parallel code. To solve this problem, a new sparse matrix method is designed based on coordinate format and compressed sparse row format. This new method erases memory peaks during the matrix construction phase and also reduces the duration of the phase because less data is manipulated. The enhanced code (*MBEMv3.0*) has been run on various large-scale tests ($N = O(10^6)$) and has proved to be very robust and of excellent accuracy. The speed up has exceeded 50 in some crack propagation simulations. Based on the comparison of the results of the two versions, we can conclude that code performance is greatly improved.

The second part concerns extension work presented in *chapter 4* and *chapter 5*. First, the existing code is extended to consider complex multizone problems, namely problems in which the zones can be in any configuration and the interfaces can have any orientation. The second extension work concerns crack propagation.

Propagation laws are presented, and Paris law is implemented to direct the re-meshing algorithm. The third extension work also presented in *chapter 4* concerns the simulation of surface and interface breaking cracks. Multiple node technique is used for that purpose and an automatic multiple node algorithm is proposed for the simulation of the propagation of these types of cracks. The last extension work concerns FEM-BEM coupling and is presented in *chapter 5*. Membrane finite element is implemented and validated to model fiberglass grids. A direct coupling technique is then used to take into account the finite element part when solving the boundary integral equations. This study mainly permits to show that the code in development can be extended to take advantage of the possibilities of finite elements.

The third part (*Chapter 6*) presents the application of *MBEMv3.0* in simulating road structures (pavements). The structure of pavements is multi-layered and multi-fractured. The behaviors (deformations) of the asphalt surface under vehicle loads are computed. The issues previously encountered by *MBEMv2.0* are solved and multiple cracked pavements are simulated in acceptable duration. Furthermore, crack propagation is studied in the pavement structure including the interesting case of reflective cracking. The simulation of a pavement reinforced by fiberglass grid is also explored through FEM-BEM coupling. This shows that the fiberglass grids can delay reflective cracking.

7.2 Directions for future work

The numerical work developed in this thesis has provided a robust algorithm in treating various large-scale problems of multizone and multiple crack propagation in the context of linear fracture mechanics. This consists a step in the development of a fast solver *MBEM* in fracture mechanics. There are a great number of aspects and factors that must be accounted for if one wants the numerical approach to get closer to the real behavior of this phenomenon. Some possible directions for future work are now briefly discussed.

Crack propagation

In this work a simple power law (Paris law) is implemented to simulate the fatigue crack growth. Since, suitable criteria for crack propagation are still being debated, other propagation laws should be implemented in the code. It should be possible for the user to choose the propagation law to use and set thereafter the input parameters. Apart from the laws, the simulation of surface breaking cracks presented in this work is not general. An automatic re-meshing technique should be implemented for the simulation of the propagation of these types of cracks. It will then be possible to study a crack that propagates and crosses the interface delimiting a zone. The crack propagation code can also be improved by implementing triangular boundary elements.

Opitimizations

Despite the great performance of *MBEMv3.0*, there are still several possibilities for optimization. First, parallelization work can be extended to accelerate some additional parts of the code such as the pre-processing phase which can be long in some large-scale simulations. One of the limits of large-scale simulation is the memory requirements to store the matrix of near contributions. In order to reduce it, some special techniques such as hierarchical matrix representation *via* adaptive cross approximation can be explored. This can be combined to investigations on using a fast-direct solver for the flexible GMRES preconditioning task performed here by using an inner GMRES. Another possibility concerns the quality of the boundary meshes used. In this study, the boundary meshes are generated mainly based on the geometries. Undue discretization errors are not expected due to the relative simplicity of the geometrical configurations used. On the other hand, controlling discretization errors is important for applications on complex configurations. Adaptive mesh refinement methods such as the one developed by Chaillat et al. in [191] for the BEM can be implemented to control the accuracy of computed solutions.

FEM-BEM coupling

Another interesting alternative for the future work is the extension of the FEM-BEM coupling. Finite elements are very popular, and several codes already exist. implementing a home version of finite elements as we did here is not the best solution. Ideally the capabilities of existing FE software should be used to generate the FE stiffness matrix. Of course, to do so, one must understand all the details about the chosen software. Understanding is not sufficient, one must also have access to its subroutines, and it must be possible to modify them. Once these difficulties have been overcome, the obtained code will allow an efficient simulation of real civil engineering structures.

Simulation of engineering structures

We are interested in several practical cases of cracking in pavements. In following months, we would like to study the propagation of a crack which breaks the upper surface of the pavement, Fig. 7.1. For the simulation, the surface breaking crack will intersect the loading area. In experiments, these types of cracks propagate to the lower layers of the pavement. We would also like to study the propagation of an interface breaking crack in the first layer of the pavement as shown in Fig. 7.1. These types of cracks propagate to the upper surface as a reflective crack.

A major development of the code will be achieved by taking into consideration the micro-structure of asphalt concrete in linear fracture mechanics. This can be achieved by integrating rigid polygonal inclusions in cracked domains for the simulation of fatigue tests: alternate bending and traction/compression. For its regional and national influence, our team has joined the Thematic Inter-disciplinary Institute, research and training [*ITI: G-EAU-TE* (Labex)]. For this project, our work

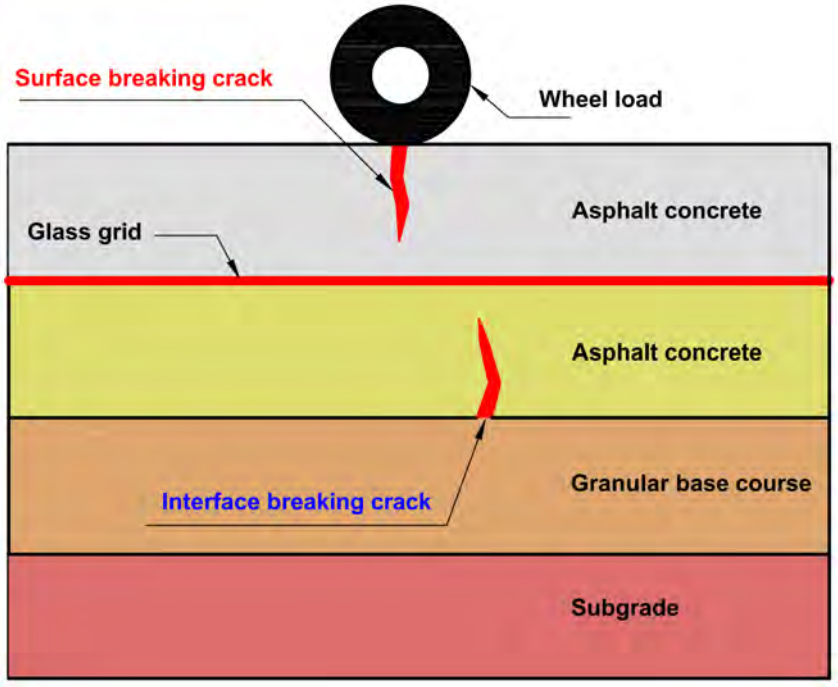


Figure 7.1 – Perspectives in cracked pavement simulation

must adapt to the development of scaling approaches to evaluate reservoir properties (permeability, thermal conductivity, elastic modulus) at a scale of 100 meters. At the same time, the effects of temperature cycles and water pressure, considered as fatigue loads or rapid loads that can cause significant damage, must be considered in future developments. The extended code thus will allow the user to carry out analyses on the behavior and integrity (over time) of tanks and infrastructures subjected to the described loads.

7.3 Main publications associated with this work

Articles in peer-reviewed international journals [1, 2]

- A. Dansou, S. Mouhoubi, C. Chazallon and M. Bonnet. Modeling multi-crack propagation by the Fast Multipole Symmetric Galerkin BEM. *Engineering Analysis with Boundary Elements*, 106:309-319, 2019.

- A. Dansou, S. Mouhoubi and C. Chazallon. Optimizations of a fast multipole symmetric Galerkin boundary element method code. *Numerical Algorithms*, 2019.

Articles in national journals [3]

- C. Chazallon, S. Mouhoubi and A. Dansou. Modèles de dégradation des structures. *Revue générale des routes et de l'aménagement*, 963, 2019.

International conferences [4, 5]

- A. Dansou, S. Mouhoubi, C. Chazallon and M. Bonnet. Modeling crack propagation in 3D heterogeneous multi-cracked roads by Fast Multipole Symmetric Galerkin Boundary Element Method. *Symposium of the International Association for Boundary Element Methods (IABEM)*, Paris, France, 2018.

- A. Dansou, S. Mouhoubi and C. Chazallon. Data reusing techniques to accelerate a crack propagation boundary element method code. *VI International Conference on Computational Modeling of Fracture and Failure of Materials and Structures (CFRAC)*, Braunschweig, Germany, 2019.

National conferences with article [6]

- A. Dansou, S. Mouhoubi, C. Chazallon and M. Bonnet. A parallel boundary element method code to simulate multi-cracked structures. *14ème Colloque National en Calcul des Structures (CSMA)*, Presqu'île de Giens, France, 2019.

Appendix A

SGBEM for fracture mechanics

A.1 Integral equations for linear elastic problems

A.1.1 Variational integral formulation

Stationarity condition

The solution $\boldsymbol{u}$ to the elastic problem (Navier's equations section 2.1.1) minimizes the following augmented potential energy functional E($\boldsymbol{v}$):

$$E(\boldsymbol{v}) = \int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{u}) : C : \boldsymbol{\varepsilon}(\boldsymbol{v}) dV - \int_{S_T} \boldsymbol{t}^D . \boldsymbol{v} dS - \int_{S_u} \boldsymbol{t} . (\boldsymbol{v} - \boldsymbol{u}^D) dS \tag{A.1}$$

The last integral term in (A.1) has the effect that $\boldsymbol{u}$ is an unconstrained minimizer of E($\boldsymbol{v}$): no kinematic constraints are imposed on $\boldsymbol{v}$. Letting $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{\delta u}$, we have:

$$\delta E(\boldsymbol{v}).\delta \boldsymbol{u} = \int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{u}) : C : \boldsymbol{\varepsilon}(\delta \boldsymbol{u}) dV - \int_{S_T} \boldsymbol{t}^D . \delta \boldsymbol{u} dS - \int_{S_u} \boldsymbol{t} . \delta \boldsymbol{u} dS = 0 \tag{A.2}$$

with no kinematic constraint on $\delta \boldsymbol{u}$. Indeed, from

$$\int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{u}) : C : \boldsymbol{\varepsilon}(\delta \boldsymbol{u}) dV = \int_{\partial \Omega} \boldsymbol{T}^n(\boldsymbol{u}) . \delta \boldsymbol{u} dS - \int_{\Omega} div[C : \boldsymbol{\varepsilon}(\delta \boldsymbol{u}) . \delta \boldsymbol{u}] dV,$$

it is readily shown that $\delta \boldsymbol{u}$ solves the Navier's equations in section 2.1.1 and that the Lagrange multiplier $\boldsymbol{t}$ equals the traction vector $\boldsymbol{T}^n(\boldsymbol{u})$ on S_u. Now let us suppose that the minimization of E($\boldsymbol{v}$) is attempted only for those $\boldsymbol{v}$ which satisfy the local equilibrium equation (2.2) in neglecting the effects of body forces ($\boldsymbol{b} = 0$). Since this is also satisfied by the solution $\boldsymbol{u}$, one is then led to restrict (A.2) to the trial functions $\boldsymbol{\delta u} = \boldsymbol{v} - \boldsymbol{u}$ which themselves satisfy (2.2). Upon integration by parts of the domain integral, the stationarity condition (A.2) thus takes the form

$$\delta E(\boldsymbol{v}).\delta \boldsymbol{u} = \int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{u}) : C : \boldsymbol{\varepsilon}(\delta \boldsymbol{u}) dV - \int_{S_T} \boldsymbol{t}^D . \delta \boldsymbol{u} dS - \int_{S_u} \boldsymbol{t} . \delta \boldsymbol{u} dS = 0 \tag{A.3}$$

which constitutes the starting point for the derivation of a direct variational BIE formulation.

Test functions

The next important step is the actual construction of test functions $\delta\boldsymbol{u}$ that satisfy (2.2). In order to do so, recall that any such $\delta\boldsymbol{u}$ admits the integral representation ($\boldsymbol{x}$ interior to):

$$\delta\boldsymbol{u}_k(\boldsymbol{x}) = -\int_{\partial\Omega} \delta\boldsymbol{u}_i(\boldsymbol{y}) n_j(\boldsymbol{y}) \Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{y}) dS_y + \int_{\partial\Omega} \delta\boldsymbol{t}_i(\boldsymbol{y}) U_i^k(\boldsymbol{x},\boldsymbol{y}) dS_y \tag{A.4}$$

where $\delta\boldsymbol{t}$ is the traction vector $\boldsymbol{T}^n(\boldsymbol{u})$. Similarly, any $\delta\boldsymbol{u}^+$ which solves (2.2) in the exterior domain $\Omega^+ = R^3 - \Omega$ satisfies at any $\boldsymbol{x}$ interior to the complementary (with reference to Ω^+) integral representation:

$$0 = -\int_{\partial\Omega} \delta\boldsymbol{u}_i^+(\boldsymbol{y}) n_j(\boldsymbol{y}) \Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{y}) dS_y + \int_{\partial\Omega} \delta\boldsymbol{t}_i^+(\boldsymbol{y}) U_i^k(\boldsymbol{x},\boldsymbol{y}) dS_y \tag{A.5}$$

where $\delta\boldsymbol{t}^+$ stands for $\boldsymbol{T}^n(\boldsymbol{u}^+)$. The normal $\boldsymbol{n}$, outwards form Ω is used in both (A.4) and (A.5). The symbols $U_i^k(x,y)$,$\Sigma_{ij}^k(x,y)$ denote the i- and ij-components of the elastic displacement and stress fields created at $\boldsymbol{y}$ by a unit point force applied at $\boldsymbol{x}$ along the k-direction (elastostatic fundamental solution). The full space (Kelvin) solution, a half space solution with a free-surface condition (Mindlin), or any other fundamental solution defined on a subset of R^3 that includes Ω may be used for this purpose. They all have the symmetry property:

$$U_k^a(\boldsymbol{x},\boldsymbol{y}) = U_a^k(\boldsymbol{x},\boldsymbol{y}) \tag{A.6}$$

which in turn implies:

$$C_{ijab}\frac{\partial}{\partial x_b} U_k^a(x,y) = \Sigma_{ij}^k(y,x) \qquad C_{ijab}\frac{\partial}{\partial x_b}\Sigma_{kl}^a(y,x) = C_{ijab}\frac{\partial}{\partial y_b}\Sigma_{ij}^k(y,x) \tag{A.7}$$

Next, (A.5) is subtracted from (A.4), giving:

$$\delta\boldsymbol{u}_k(\boldsymbol{x}) = \int_{\partial\Omega} \widetilde{\boldsymbol{u}}_i(\boldsymbol{y}) n_j(\boldsymbol{y}) \Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{y}) dS_y - \int_{\partial\Omega} \widetilde{\boldsymbol{t}}_k(\boldsymbol{y}) U_i^k(\boldsymbol{x},\boldsymbol{y}) dS_y \tag{A.8}$$

using the displacement and traction jumps across : $\widetilde{\boldsymbol{u}} = \delta\boldsymbol{u}^+ - \delta\boldsymbol{u}$ and $\widetilde{\boldsymbol{t}} = \delta\boldsymbol{t}^+ - \delta\boldsymbol{t}$. It is indeed legitimate to restrict the above formula by imposing $\delta\boldsymbol{u}^+ = \delta\boldsymbol{u}$, i.e. $\widetilde{\boldsymbol{u}} = 0$ on S_u and $\delta\boldsymbol{t}^+ = -\delta\boldsymbol{t}$, i.e. $\widetilde{\boldsymbol{t}} = 0$ on S_T, which gives:

$$\begin{aligned} \delta\boldsymbol{u}_k(\boldsymbol{x}) &= \int_{S_T} \widetilde{\boldsymbol{u}}_i(\boldsymbol{y}) n_j(\boldsymbol{y}) \Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{y}) dS_y - \int_{S_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{y}) U_i^k(\boldsymbol{x},\boldsymbol{y}) dS_y & \text{(A.9)} \\ &= \delta\boldsymbol{u}_k^T(\boldsymbol{x}) + \delta\boldsymbol{u}_k^U(\boldsymbol{x}) & \text{(A.10)} \end{aligned}$$

Such test functions $\delta\boldsymbol{u}$ satisfy (2.2) and reflect the boundary condition structure of the initial mixed elastostatic problem under study. Note that, since $\widetilde{\boldsymbol{u}}$ must be continuous all over $\partial\Omega$, one has:

$$\widetilde{\boldsymbol{u}}|_{\partial\widetilde{S}_T} = 0 \tag{A.11}$$

The stress tensor $\delta\boldsymbol{\sigma} = \boldsymbol{\sigma}(\partial\boldsymbol{u})$ is then given from (A.8) by the representation:

$$\delta\boldsymbol{\sigma}_{ij}(\boldsymbol{x}) = C_{klab}\int_{S_T}\widetilde{\boldsymbol{u}}_k(\boldsymbol{y})n_j(\boldsymbol{y})\frac{\partial}{\partial y_b}\Sigma^a_{ij}(\boldsymbol{y},\boldsymbol{x})dS_y - \int_{S_u}\widetilde{\boldsymbol{t}}_k(\boldsymbol{y})\Sigma^k_{ij}(\boldsymbol{y},\boldsymbol{x})dS_y \tag{A.12}$$

where use has been made of (A.6), (A.7).

Symmetric BIE formulation

The symmetric BIE formulation comes from the substitution of (A.8), (A.12) into the stationarity equation (A.3). This leads to a formulation in terms of double surface integrals. However, integrability problems, related to the presence of kernels behaving like r^{-1}, r^{-2}, r^{-3} where $r = |y - x|$ (especially the last two), require a limiting process of some kind.

A.1.2 Regularization

Auxiliary surface

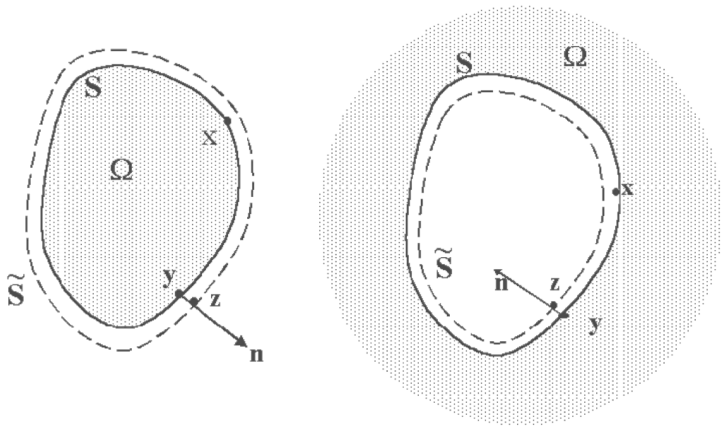


Figure A.1 – Boundary $\partial\Omega$ and auxiliary surface $\widetilde{S}$

Let $\widetilde{S}$ be a closed, regular surface exterior to Ω (hence exterior to $\partial\Omega$ for interior problems, but interior to $\partial\Omega$ for exterior problems), defined by means of a one-to-one mapping F onto $\partial\Omega$:

$$\boldsymbol{y} \in \Omega \longrightarrow \boldsymbol{z} = \boldsymbol{F}(\boldsymbol{y}) \in \widetilde{S}$$

which is left unspecified. This surface $\widetilde{S}$ is substituted to $\partial\Omega$ in (A.8), (A.12). The idea is to perform some analytic manipulations, for regularization purposes,

on the double surface integrals over $\partial\Omega x\widetilde{S}$ (in which case the elastic kernels are nonsingular) and then consider the limiting process $\widetilde{S} \to \partial\Omega$.
The test functions δu, $\delta\sigma$ are thus given by:

$$\delta \boldsymbol{u}_k(\boldsymbol{x}) = \int_{\widetilde{S}_T} \widetilde{\boldsymbol{u}}_i(\boldsymbol{y}) n_j(\boldsymbol{y}) \Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{y}) dS_y - \int_{\widetilde{S}_u} \widetilde{\boldsymbol{t}}_i(\boldsymbol{y}) U_i^k(\boldsymbol{x},\boldsymbol{y}) dS_y \tag{A.13}$$

$$\delta \boldsymbol{\sigma}_{ij}(\boldsymbol{x}) = C_{klab} \int_{\widetilde{S}_T} \widetilde{\boldsymbol{u}}_k(\boldsymbol{y}) n_l(\boldsymbol{y}) \frac{\partial}{\partial y_b} \Sigma_{ij}^a(\boldsymbol{y},\boldsymbol{x}) dS_y - \int_{\widetilde{S}_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{y}) \Sigma_{ij}^k(\boldsymbol{y},\boldsymbol{x}) dS_y \tag{A.14}$$

(with $\widetilde{S}_u = F(S_u)$, $\widetilde{S}_T = F(S_T)$). Recall that eqns. (A.13), (A.14) apply for either bounded or unbounded, n being the unit normal outwards from Ω.

Upon substitution of (A.13), (A.14), the stationarity condition (A.3) for E(u) can be split into two independent parts:

$$\begin{cases} \forall \widetilde{u} & \delta E^T = 0 \\ \forall \widetilde{t} & \delta E^U = 0 \end{cases} \tag{A.15}$$

where δE_T and δE_U collect the terms containing $\widetilde{u}$ and $\widetilde{t}$ respectively that arise from the substitution of (A.13), (A.14) into (A.3).

Evaluation of δE^u and δE^T

From eqns (A.13), (A.14),(A.3), one has:

$$\delta E^U = I_1 + I_2 \tag{A.16}$$

$$I_1 = \int_{S_u} \int_{\widetilde{S}_u} \boldsymbol{t}_k(\boldsymbol{x}) \widetilde{\boldsymbol{t}}_i(\boldsymbol{z}) U_i^k(\boldsymbol{x},\boldsymbol{z}) dS_z dS_x + \int_{S_T} \int_{\widetilde{S}_u} \boldsymbol{t}_k^D(\boldsymbol{x}) \widetilde{\boldsymbol{t}}_i(\boldsymbol{z}) U_i^k(\boldsymbol{x},\boldsymbol{z}) dS_z dS_x \tag{A.17}$$

$$I_2 = \int_{S_u} \int_{\widetilde{S}_u} \boldsymbol{u}_i^D(\boldsymbol{x}) \boldsymbol{n}_j(\boldsymbol{x}) \widetilde{\boldsymbol{t}}_k(\boldsymbol{z}) \Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x}) dS_z dS_x + \int_{S_T} \int_{\widetilde{S}_u} \boldsymbol{u}_i(\boldsymbol{x}) \boldsymbol{n}_j(\boldsymbol{x}) \widetilde{\boldsymbol{t}}_k(\boldsymbol{z}) \Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x}) dS_z dS_x \tag{A.18}$$

First, the term I_1 is made of a (potentially) weakly singular surface integral followed by a regular one and thus needs no regularization; it is left unchanged. Then, I_2, eqn. (A.18), is rewritten as:

$$\begin{aligned} I_2 = & - \int_{\widetilde{S}_u} \int_{S_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{z}) \left[\boldsymbol{u}_i^D(\boldsymbol{x}) - \boldsymbol{u}_i^D(\boldsymbol{y}) \right] \boldsymbol{n}_j(\boldsymbol{x}) \Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x}) dS_x dS_z \\ & - \int_{\widetilde{S}_u} \int_{S_T} \widetilde{\boldsymbol{t}}_k(\boldsymbol{z}) \left[\boldsymbol{u}_i(\boldsymbol{x}) - \boldsymbol{u}_i^D(\boldsymbol{y}) \right] \boldsymbol{n}_j(\boldsymbol{x}) \Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x}) dS_x dS_z \\ & - \int_{\widetilde{S}_u} \int_{\partial\Omega} \widetilde{\boldsymbol{t}}_k(\boldsymbol{z}) \boldsymbol{u}_i^D(\boldsymbol{y}) \boldsymbol{n}_j(\boldsymbol{x}) \Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x}) dS_x dS_z \end{aligned} \tag{A.19}$$

where $\boldsymbol{y} \in S_u$ is the point such that $F(\boldsymbol{y}) = \boldsymbol{z} \in \widetilde{S}_u$ (hence $\boldsymbol{u}(\boldsymbol{y}) = \boldsymbol{u}^D(\boldsymbol{y})$). Integrals over $\partial\Omega$ and $\widetilde{S}$ have been interchanged, this operation being valid because of the current non-singular character of the integrals.

Besides, one notes that the fundamental stress tensor satisfies, by virtue of equilibrium, the following identity (the source point $\boldsymbol{z}$ being exterior to Ω):

$$\int_{\partial\Omega} \boldsymbol{n}_j(\boldsymbol{x})\Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x})dS_x = \kappa\delta_{ik} \tag{A.20}$$

where $\kappa = 0$ (Ω bounded, $\boldsymbol{n}$ exterior to $\partial\Omega$) or $\kappa = 1$ ($R^3 - \Omega$ bounded, $\boldsymbol{n}$ interior to $\partial\Omega$).
Thus, the last integral in (A.19) becomes:

$$\int_{\widetilde{S}_u}\int_{\partial\Omega} \widetilde{\boldsymbol{t}}_k(\boldsymbol{z})\boldsymbol{u}_i^D(\boldsymbol{y})\boldsymbol{n}_j(\boldsymbol{x})\Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x})dS_x dS_z = \kappa\int_{\widetilde{S}_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{z})\boldsymbol{u}_i^D(\boldsymbol{y})dS_z \tag{A.21}$$

These manipulations result in a regularization of the initial strongly singular integral w.r.t. $\boldsymbol{x}$ in (A.18). Indeed, one notes that, since

$$U(\boldsymbol{z},\boldsymbol{x}) \sim |\boldsymbol{z},\boldsymbol{x}|^{-1} \qquad \Sigma(\boldsymbol{z},\boldsymbol{x}) \sim |\boldsymbol{z},\boldsymbol{x}|^{-2} \tag{A.22}$$

the expressions (A.17), (A.19) involve weakly singular integrals w.r.t. $\boldsymbol{x}$ followed by nonsingular integrals w.r.t. $\boldsymbol{z}$, provided the regularity requirement $\boldsymbol{u} \in C^{0,\alpha}(\partial\Omega)$, i.e.:

$$\exists C > 0, \exists\alpha \in]0,1] \qquad |\boldsymbol{u}(\boldsymbol{y}) - \boldsymbol{u}(\boldsymbol{x})| \leq C|\boldsymbol{y} - \boldsymbol{x}|^{\alpha} \tag{A.23}$$

is met. Then, the limiting expression of I_2, I_1 for $\widetilde{S} \to \partial\Omega$ is obtained by a mere substitution of ($\widetilde{S}_u|\boldsymbol{z}$) by ($S_u|\boldsymbol{y}$) in eqns. (A.17), (A.19). The result is thus:

$$\begin{aligned} I_1 \;=\; & \int_{S_u}\int_{S_u} \boldsymbol{t}_k(\boldsymbol{x})\widetilde{\boldsymbol{t}}_i(\boldsymbol{y})U_i^k(\boldsymbol{x},\boldsymbol{y})dS_y dS_x \\ & + \int_{S_T}\int_{S_u} \boldsymbol{t}_k^D(\boldsymbol{x})\widetilde{\boldsymbol{t}}_i(\boldsymbol{y})U_i^k(\boldsymbol{x},\boldsymbol{y})dS_y dS_x \end{aligned} \tag{A.24}$$

$$\begin{aligned} I_2 \;=\; & -\int_{S_u}\int_{S_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{y})\left[\boldsymbol{u}_i^D(\boldsymbol{x}) - \boldsymbol{u}_i^D(\boldsymbol{y})\right]\boldsymbol{n}_j(\boldsymbol{x})\Sigma_{ij}^k(\boldsymbol{y},\boldsymbol{x})dS_x dS_z \\ & -\int_{S_u}\int_{S_T} \widetilde{\boldsymbol{t}}_k(\boldsymbol{y})\left[\boldsymbol{u}_i(\boldsymbol{x}) - \boldsymbol{u}_i^D(\boldsymbol{y})\right]\boldsymbol{n}_j(\boldsymbol{x})\Sigma_{ij}^k(\boldsymbol{y},\boldsymbol{x})dS_x dS_y \\ & -\kappa\int_{S_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{y})\boldsymbol{u}_i^D(\boldsymbol{y})dS_y \end{aligned} \tag{A.25}$$

For eqn. (A.25) above, use has been made of the substitution of (A.20) into (A.18).
From eqns (A.3), (A.13), (A.14), one has:

$$\delta E^T = J_1 + J_2 \tag{A.26}$$

$$J_1 = \int_{S_u}\int_{\widetilde{S}_T} \boldsymbol{t}_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{z})\boldsymbol{n}_j(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x + \int_{S_T}\int_{\widetilde{S}_T} \boldsymbol{t}^D_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{z})\boldsymbol{n}_i(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \tag{A.27}$$

$$J_2 = \int_{S_u}\int_{\widetilde{S}_T} \boldsymbol{u}^D_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{x})C_{klab}\widetilde{\boldsymbol{u}}_k(\boldsymbol{z})\boldsymbol{n}_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma^a_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \int_{S_T}\int_{\widetilde{S}_T} \boldsymbol{u}_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{x})C_{klab}\widetilde{\boldsymbol{u}}_k(\boldsymbol{z})\boldsymbol{n}_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma^a_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \tag{A.28}$$

First, J_1 is subjected to a treatment similar to that of I_2. For this purpose, eqn. (A.27) is rewritten, upon addition and subtraction of $\widetilde{\boldsymbol{u}}(\boldsymbol{x})$ to $\widetilde{\boldsymbol{u}}(\boldsymbol{z})$, as:

$$\begin{aligned} J_2 = &- \int_{S_u}\int_{\widetilde{S}_T} \boldsymbol{t}_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{z})\boldsymbol{n}_j(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \\ &- \int_{S_T}\int_{\widetilde{S}_T} \boldsymbol{t}^D_k(\boldsymbol{x})\left[\widetilde{\boldsymbol{u}}_i(\boldsymbol{z}) - \widetilde{\boldsymbol{u}}_i(\boldsymbol{x})\right]\boldsymbol{n}_j(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \\ &- \int_{S_T}\int_{\widetilde{S}_T} \boldsymbol{t}^D_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \end{aligned} \tag{A.29}$$

where the fact that $\widetilde{\boldsymbol{u}} = 0$ for $\boldsymbol{x} \in S_u$ has been used. Besides, the fundamental stress tensor satisfies the following identity, similar to (A.20) but with the source point $\boldsymbol{x}$ now being interior to Ω:

$$\int_{\widetilde{S}} \boldsymbol{n}_j(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z = (\kappa - 1)\delta_{ik} \tag{A.30}$$

where again $\kappa = 0$ (Ω bounded, $\boldsymbol{n}$ exterior to $\partial\Omega$) or $\kappa = 1$ ($R^3 - \Omega$ bounded, $\boldsymbol{n}$ interior to $\partial\Omega$). The last integral in (A.29) then becomes:

$$\begin{aligned} &\int_{S_T}\int_{\widetilde{S}_T} \boldsymbol{t}^D_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \\ = &\ (\kappa - 1)\int_{S_T} \boldsymbol{t}^D_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_k(\boldsymbol{x})dS_x - \int_{S_T}\int_{\widetilde{S}_u} \boldsymbol{t}^D_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{z})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \end{aligned} \tag{A.31}$$

Substitution of (A.31) into (A.29) allows for an expression of J_1 which is made, in the limiting case $\widetilde{S} \to \partial\Omega$, of weakly singular integrals over S_T followed by non-singular integrals over S_u, S_T , as follows:

$$
\begin{aligned}
J_1 = \; & - \int_{S_u}\int_{S_T} \boldsymbol{t}_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{y})\boldsymbol{n}_j(\boldsymbol{y})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{y})dS_y dS_x \\
& - \int_{S_T}\int_{S_T} \boldsymbol{t}^D_k(\boldsymbol{x})\left[\widetilde{\boldsymbol{u}}_i(\boldsymbol{y}) - \widetilde{\boldsymbol{u}}_i(\boldsymbol{x})\right]\boldsymbol{n}_j(\boldsymbol{y})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{y})dS_y dS_x \\
& + (1-\kappa)\int_{S_T} \boldsymbol{t}^D_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_k(\boldsymbol{x})dS_x \\
& + \int_{S_T}\int_{S_u} \boldsymbol{t}^D_k(\boldsymbol{x})\widetilde{\boldsymbol{u}}_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{y})\Sigma^k_{ij}(\boldsymbol{x},\boldsymbol{y})dS_y dS_x
\end{aligned}
\tag{A.32}
$$

The weakly singular character of the inner integral in the above equation relies upon $\widetilde{\boldsymbol{u}}$ being $C^{0;\alpha}$continuous over $\partial\Omega$, eqn. (A.23). The property (A.11) of $\widetilde{\boldsymbol{u}}$, together with (A.23), implies:

$$
\exists C>0, \exists \alpha \in]0,1] \qquad |\boldsymbol{u}(\boldsymbol{y})| \leq C|\boldsymbol{y}-\boldsymbol{x}| \qquad \boldsymbol{x}\in S_u, \boldsymbol{y}\in S_T \tag{A.33}
$$

and thus, also contributes to the weakly singular character of (A.32).

Next, let us consider the second integral J_2 in (A.26), which involves the hyper-singular kernel $C_{klab}\frac{\partial}{\partial z_b}\Sigma^a_{ij}(z,x)$. Using the symmetry property (A.7), one notes that, for $\boldsymbol{x}$ and $\boldsymbol{y}$ distinguished:

$$
\begin{aligned}
\frac{\partial}{\partial x_j}\left\{C_{klag}\frac{\partial}{\partial x_b}\Sigma^a_{ij}(\boldsymbol{z},\boldsymbol{x})\right\} &= \frac{\partial}{\partial x_b}\left\{C_{klag}\frac{\partial}{\partial x_j}\Sigma^a_{ij}(\boldsymbol{z},\boldsymbol{x})\right\} = 0 \\
\frac{\partial}{\partial z_l}\left\{C_{klag}\frac{\partial}{\partial z_b}\Sigma^a_{ij}(\boldsymbol{z},\boldsymbol{x})\right\} &= \frac{\partial}{\partial z_b}\left\{C_{klag}\frac{\partial}{\partial z_l}\Sigma^a_{kl}(\boldsymbol{z},\boldsymbol{x})\right\} = 0
\end{aligned}
\tag{A.34}
$$

This implies the existence of a fourth-order tensor A such that:

$$
C_{klag}\frac{\partial}{\partial z_l}\Sigma^a_{kl}(\boldsymbol{z},\boldsymbol{x}) = \boldsymbol{e}_{iep}\boldsymbol{e}_{jfq}\boldsymbol{e}_{kgr}\boldsymbol{e}_{lhs}\frac{\partial}{\partial x_e}\frac{\partial}{\partial x_f}\frac{\partial}{\partial z_g}\frac{\partial}{\partial z_h}A_{pqrs}(\boldsymbol{z},\boldsymbol{x}) \tag{A.35}
$$

where e_{ijk} are the components of the permutation tensor. For instance, the tensor $\mathrm{A}(\boldsymbol{z},\boldsymbol{x})$ associated with the three-dimensional isotropic Kelvin fundamental solution is given by:

$$
A_{pqrs}(\boldsymbol{z},\boldsymbol{x}) = \frac{\mu}{8\pi}\left[\delta_{pr}\delta_{qs} + \delta_{ps}\delta_{qr} + \frac{2\nu}{1-\nu}\delta_{pq}\delta_{rs}\right]|\boldsymbol{z}-\boldsymbol{x}| \tag{A.36}
$$

The decomposition (A.35) allows for the use of the following variant of Stokes formula:

$$
\int_{S_T} g\boldsymbol{e}_{abc}\boldsymbol{n}_a f_{,b}dS = -\int_{S_T} f\boldsymbol{e}_{abc}\boldsymbol{n}_a g_{,b}dS \tag{A.37}
$$

where S is any regular surface; moreover, either S is closed or $f = 0$ on ∂S. Here it is applied twice (once w.r.t. $\boldsymbol{z}$ and once w.r.t. $\boldsymbol{x}$) to J_2, eqn. (A.28); this gives:

$$\begin{aligned} J_2 &= \int_{S_u}\int_{\widetilde{S}_T} (Ru^D)_{iq}(\boldsymbol{x}) B_{ikqs}(\boldsymbol{r}) (R\widetilde{u})_{ks}(\boldsymbol{z}) dS_z dS_x \\ &\quad + \int_{S_T}\int_{\widetilde{S}_T} (Ru)_{iq}(\boldsymbol{x}) B_{ikqs}(\boldsymbol{r}) (R\widetilde{u})_{ks}(\boldsymbol{z}) dS_z dS_x \end{aligned} \tag{A.38}$$

using the notations:

$$(Ru)_{iq} = \boldsymbol{e}_{ifq}\boldsymbol{n}_j \boldsymbol{u}_{i,f} \tag{A.39}$$

$$B_{ikqs}(\boldsymbol{z},\boldsymbol{x}) = \boldsymbol{e}_{iep}\boldsymbol{e}_{kgr}\frac{\partial}{\partial x_e}\frac{\partial}{\partial z_g} A_{pqrs}(\boldsymbol{z},\boldsymbol{x}) \tag{A.40}$$

The result (A.38) makes use of the property (A.11) of $\widetilde{\boldsymbol{u}}$. One notes that:

$$B(\boldsymbol{z},\boldsymbol{x}) \sim |\boldsymbol{z}-\boldsymbol{x}|^{-1} \tag{A.41}$$

which stems from (A.35), (A.40). This implies that eqn. (A.38) involves a weakly singular inner (w.r.t. $\boldsymbol{x}$) integral followed by a non-singular outer (w.r.t. $\boldsymbol{z}$) integral. The limiting expression of J_2 for $\widetilde{S} \to \partial\Omega$ is thus obtained by a simple replacement of $\widetilde{S}_T|\boldsymbol{z}$ with $S_T|\boldsymbol{y}$ in (A.38):

$$\begin{aligned} J_2 &= \int_{S_u}\int_{S_T} (Ru^D)_{iq}(\boldsymbol{x}) B_{ikqs}(\boldsymbol{r}) (R\widetilde{u})_{ks}(\boldsymbol{y}) dS_y dS_x \\ &\quad + \int_{S_T}\int_{S_T} (Ru)_{iq}(\boldsymbol{x}) B_{ikqs}(\boldsymbol{r}) (R\widetilde{u})_{ks}(\boldsymbol{y}) dS_y dS_x \end{aligned} \tag{A.42}$$

A.2 Fracture mechanics problems

Let Ω denote a generic body, whose boundary portions S_T and S_u are subject to prescribed tractions and displacements, respectively:

$$(S \equiv S_u \cup S_T)$$

Let surface S_c denote a crack inside Ω. Two faces S_c^+ and S_c^- of the crack are associated with the normal unit vectors $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$ opposite and choosen by a unit vector $\boldsymbol{n}$ pointing from S_c^- towards S_c^+.

$$\boldsymbol{n} = \boldsymbol{n}^- = -\boldsymbol{n}^+$$

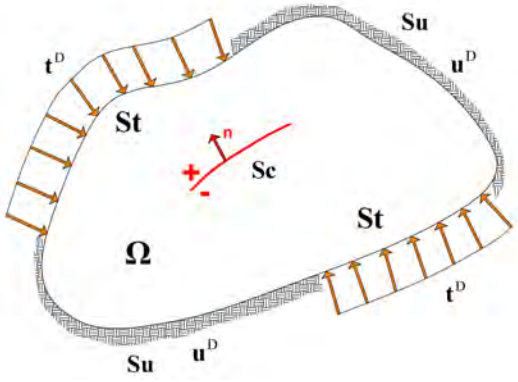


Figure A.2 – 3D linear elastic fracture mechanics

The surface $S_c = S_c^+ = S_c^-$ is considered as an extension of the regular exterior surface $\partial\Omega$ of domain: $\partial\Omega = S \cup S_c^+ \cup S_c^-$. The crack S_c is conceived as a prescribed traction surface; equal and opposite tensions are applied on S_c:

$$\boldsymbol{p}^+ = -\boldsymbol{p}^- = -\boldsymbol{p}.$$

We define the displacement discontinuity as:

$$\Delta\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x}^+) - \boldsymbol{u}(\boldsymbol{x}^-)$$

The surface S_c is an ideal representation of the crack, we have considered that the boundary $\bar{S}_c = S_c^+ \cup S_c^-$ does not differ from S_c in the last definition. The surface $\bar{S}_c$ is the function of a parameter I, for $I \to 0$ we have $\bar{S}_c \to S_c$. It becomes clear that we can write that:

$$\begin{aligned} &\boldsymbol{x} \leftrightarrow \boldsymbol{x}^+ = \Lambda^+(\boldsymbol{x}, I) && : \boldsymbol{x} \in S_c, \boldsymbol{x}^+ \in S_c^+ \\ &\boldsymbol{x} \leftrightarrow \boldsymbol{x}^- = \Lambda^-(\boldsymbol{x}, I) && : \boldsymbol{x} \in S_c, \boldsymbol{x}^- \in S_c^- \\ &0 \leq I \ll 1 \end{aligned}$$

This means that:

$$\Lambda^+(\boldsymbol{x}, 0) = \Lambda^-(\boldsymbol{x}, 0) = \boldsymbol{x}$$

Hence the two surfaces S_c^+ and S_c^- coincide as $I = 0$. For the initial configuration where the crack is represented by the open surface S_c, we have to rewrite the integral equations after taking the limits $I \to 0$ and $\bar{S}_c \to S_c$. We introduce an auxiliary surface $\widetilde{S}_{total} = \widetilde{S} \cup \widetilde{S}_c$ on which a point $\boldsymbol{z}$ crosses and which will have for limit, the surface $S \cup S_c$. Neglecting the effects of body forces, the stationary condition can be written as follows:

$$\begin{aligned}\delta E(\boldsymbol{u}).\delta\boldsymbol{u} &= \int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{u}) : C : \boldsymbol{\varepsilon}(\delta\boldsymbol{u})dV - \int_{S_T} \boldsymbol{t}^D.\delta\boldsymbol{u}dS - \int_{S_u} \boldsymbol{t}.\delta\boldsymbol{u}dS = 0\\ &= \int_{S_u} \boldsymbol{u}^D.T^n(\delta\boldsymbol{u})dS + \int_{S_T} \boldsymbol{u}.T^n(\delta\boldsymbol{u})dS - \int_{S_T} \boldsymbol{t}^D.\delta\boldsymbol{u}dS\\ &- \int_{S_u} \boldsymbol{t}.\delta\boldsymbol{u}dS + \int_{S_c^+} \boldsymbol{u}^+.T^n(\delta\boldsymbol{u})dS + \int_{S_c^-} \boldsymbol{u}^-.T^n(\delta\boldsymbol{u})dS\\ &+ \int_{S_c^+} \boldsymbol{p}.\delta\boldsymbol{u}^+dS - \int_{S_c^-} \boldsymbol{p}.\delta\boldsymbol{u}^-dS = 0 \end{aligned} \tag{A.43}$$

With the same expression of test functions in displacement and in stress(A.13), (A.14), we then introduce the new forms of test functions as:

Test function in displacement:

$$\begin{aligned}\delta\boldsymbol{u}_k(\boldsymbol{x}) &= \int_{\widetilde{S}_T} \widetilde{\boldsymbol{u}}_i(\boldsymbol{z})n_j(\boldsymbol{z})\Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{z})dS_z + \int_{\widetilde{S}_c^+} \widetilde{\boldsymbol{u}}_i^+(\boldsymbol{z})n_j^+(\boldsymbol{z})\Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{z})dS_z\\ &+ \int_{\widetilde{S}_c^-} \widetilde{\boldsymbol{u}}_i^-(\boldsymbol{z})n_j^-(\boldsymbol{z})\Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{z})dS_z - \int_{\widetilde{S}_u} \widetilde{\boldsymbol{t}}_i(\boldsymbol{z})U_i^k(\boldsymbol{x},\boldsymbol{z})dS_z \end{aligned} \tag{A.44}$$

Test function in stress:

$$\begin{aligned}\delta\boldsymbol{\sigma}_{ij}(\boldsymbol{x}) &= C_{klab}\int_{\widetilde{S}_T} \widetilde{\boldsymbol{u}}_k(\boldsymbol{z})n_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x})dS_z + C_{klab}\int_{\widetilde{S}_c^+} \widetilde{\boldsymbol{u}}_k(\boldsymbol{z})n_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x})dS_z\\ &+ C_{klab}\int_{\widetilde{S}_c^-} \widetilde{\boldsymbol{u}}_k(\boldsymbol{z})n_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x})dS_z - \int_{\widetilde{S}_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{z})\Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x})dS_z \end{aligned} \tag{A.45}$$

We rewrite the equation (A.43) as follows:

$$\begin{aligned}\delta E(\boldsymbol{u}).\delta\boldsymbol{u} &= \int_{S_u} \boldsymbol{u}_i^D(\boldsymbol{x}).n_j(\boldsymbol{x})\delta\sigma_{ij}(\boldsymbol{x})dS + \int_{S_T} \boldsymbol{u}_i(\boldsymbol{x}).n_j(\boldsymbol{x})\delta\sigma_{ij}(\boldsymbol{x})dS - \int_{S_T} \boldsymbol{t}_k^D(\boldsymbol{x}).\delta\boldsymbol{u}_k(\boldsymbol{x})dS\\ &- \int_{S_u} \boldsymbol{t}_k(\boldsymbol{x}).\delta\boldsymbol{u}_k(\boldsymbol{x})dS + \int_{S_c^+} \boldsymbol{u}_i^+(\boldsymbol{x}).n_j^+(\boldsymbol{x})\delta\sigma_{ij}(\boldsymbol{x})dS + \int_{S_c^-} \boldsymbol{u}_i^-(\boldsymbol{x}).n_j^-(\boldsymbol{x})\delta\sigma_{ij}(\boldsymbol{x})dS\\ &+ \int_{S_c} \boldsymbol{p}_k(\boldsymbol{x}).\delta\Delta\boldsymbol{u}_k(\boldsymbol{x})dS = 0 \end{aligned} \tag{A.46}$$

We have:

$$\begin{aligned}\delta\boldsymbol{u}^+(\boldsymbol{x}) &= \delta\boldsymbol{u}(\boldsymbol{x}^+)\\ \delta\boldsymbol{u}^-(\boldsymbol{x}) &= \delta\boldsymbol{u}(\boldsymbol{x}^-)\\ \delta\boldsymbol{u}(\boldsymbol{x}^+) - \delta\boldsymbol{u}(\boldsymbol{x}^-) &= \delta\Delta\boldsymbol{u}(\boldsymbol{x})\end{aligned}$$

The equation (A.46) can be written as:

$$\begin{aligned}\delta E(\boldsymbol{u}).\delta\boldsymbol{u} &= \int_{S_u} \boldsymbol{u}_i^D(\boldsymbol{x}).n_j(\boldsymbol{x})\delta\sigma_{ij}(\boldsymbol{x})dS + \int_{S_T} \boldsymbol{u}_i(\boldsymbol{x}).n_j(\boldsymbol{x})\delta\sigma_{ij}(\boldsymbol{x})dS - \int_{S_T} \boldsymbol{t}_k^D(\boldsymbol{x}).\delta\boldsymbol{u}_k(\boldsymbol{x})dS\\ &- \int_{S_u} \boldsymbol{t}_k(\boldsymbol{x}).\delta\boldsymbol{u}_k(\boldsymbol{x})dS - \int_{S_c} \Delta\boldsymbol{u}_i(\boldsymbol{x}).n_j(\boldsymbol{x})\delta\sigma_{ij}(\boldsymbol{x})dS + \int_{S_c} \boldsymbol{p}_k(\boldsymbol{x}).\delta\Delta\boldsymbol{u}_k(\boldsymbol{x})dS = 0 \end{aligned} \tag{A.47}$$

We know this expression:

$$\Delta\widetilde{\boldsymbol{u}} = \widetilde{\boldsymbol{u}}^{+} - \widetilde{\boldsymbol{u}}^{-} = -\delta\Delta\widetilde{\boldsymbol{u}}$$

is the test function of the displacement discontinuity on S_c. The expressions A.44 and A.45 can be rewritten as:

$$\begin{aligned}\delta\boldsymbol{u}_k(\boldsymbol{x}) &= \int_{\widetilde{S}_T} \widetilde{\boldsymbol{u}}_i(\boldsymbol{z}) n_j(\boldsymbol{z}) \Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{z}) dS_z - \int_{\widetilde{S}_c} \Delta\widetilde{\boldsymbol{u}}_i(\boldsymbol{z}) n_j(\boldsymbol{z}) \Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{z}) dS_z \\ &\quad - \int_{\widetilde{S}_u} \widetilde{\boldsymbol{t}}_i(\boldsymbol{z}) U_i^k(\boldsymbol{x},\boldsymbol{z}) dS_z \end{aligned} \tag{A.48}$$

$$\begin{aligned}\delta\boldsymbol{\sigma}_{ij}(\boldsymbol{x}) &= C_{klab} \int_{\widetilde{S}_T} \widetilde{\boldsymbol{u}}_k(\boldsymbol{z}) n_l(\boldsymbol{z}) \frac{\partial}{\partial z_b} \Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x}) dS_z - \int_{\widetilde{S}_u} \widetilde{\boldsymbol{t}}_k(\boldsymbol{z}) \Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x}) dS_z \\ &\quad - C_{klab} \int_{\widetilde{S}_c} \Delta\widetilde{\boldsymbol{u}}_k(\boldsymbol{z}) n_l(\boldsymbol{z}) \frac{\partial}{\partial z_b} \Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x}) dS_z \end{aligned} \tag{A.49}$$

Upon substitution of (A.48), (A.49), the stationarity condition for E(u) can be split into three independent parts:

$$\begin{cases} \forall\widetilde{t} & \delta E^U = I_1 + I_2 + I_3 = 0 \\ \forall\widetilde{u} & \delta E^T = J_1 + J_2 + J_3 = 0 \\ \forall\Delta\widetilde{u} & \delta E_c^T = K_1 + K_2 + K_3 = 0 \end{cases} \tag{A.50}$$

The terms I_1, I_2, J_1 and J_2 have the same expression for an uncracked domain. Let us consider the term:

$$I_3 = \int_{S_c} \int_{\widetilde{S}_u} \Delta\boldsymbol{u}_i(\boldsymbol{x}) \boldsymbol{n}_j(\boldsymbol{x}) \widetilde{\boldsymbol{t}}_k(\boldsymbol{z}) \Sigma_{ij}^k(\boldsymbol{z},\boldsymbol{x}) dS_z dS_x$$

This integral is always regular. The expression of the limit of I_3 is obtained in replacing $(\widetilde{S}_u, \boldsymbol{z})$ by $(S_u, \boldsymbol{y})$:

$$I_3 = \int_{S_u} \int_{S_c} \widetilde{\boldsymbol{t}}_k(\boldsymbol{y}) \Delta\boldsymbol{u}_i(\boldsymbol{x}) \boldsymbol{n}_j(\boldsymbol{x}) \Sigma_{ij}^k(\boldsymbol{y},\boldsymbol{x}) dS_y dS_x \tag{A.51}$$

Analogously, we obtain:

$$\begin{aligned} J_3 &= -\int_{S_c} \int_{\widetilde{S}_T} \Delta\boldsymbol{u}_i(\boldsymbol{x}) \boldsymbol{n}_j(\boldsymbol{x}) C_{klab} \widetilde{\boldsymbol{u}}_k(\boldsymbol{z}) n_l(\boldsymbol{z}) \frac{\partial}{\partial z_b} \Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x}) dS_z dS_x \\ J_3 &= -\int_{S_c} \int_{S_T} (R\Delta u)_{iq}(\boldsymbol{x}) B_{ikqs}(\boldsymbol{r}) (R\widetilde{u})_{ks}(\boldsymbol{y}) dS_y dS_x \end{aligned} \tag{A.52}$$

Now, we have to take account of the terms in the third equation of the system

(A.50) that are as following :

$$\begin{aligned}
K_1 &= -\int_{S_u}\int_{\widetilde{S}_c} \boldsymbol{u}_i^D(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{x})C_{klab}\Delta\widetilde{\boldsymbol{u}}_k(\boldsymbol{z})n_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x})dS_z dS_x \\
&\quad -\int_{S_T}\int_{\widetilde{S}_c} \boldsymbol{u}_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{x})C_{klab}\Delta\widetilde{\boldsymbol{u}}_k(\boldsymbol{z})n_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x})dS_z dS_x \\
K_2 &= \int_{S_T}\int_{\widetilde{S}_c} \boldsymbol{t}_k^D(\boldsymbol{x})\Delta\widetilde{\boldsymbol{u}}_i(\boldsymbol{z})\boldsymbol{n}_j(\boldsymbol{z})\Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{z})dS_z dS_x \\
&\quad +\int_{S_u}\int_{\widetilde{S}_c} \boldsymbol{t}_k(\boldsymbol{x})\Delta\widetilde{\boldsymbol{u}}_i(\boldsymbol{z})\boldsymbol{n}_j(\boldsymbol{z})\Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{z})dS_z dS_x - \int_{S_c}\boldsymbol{p}_k(\boldsymbol{x})\Delta\widetilde{\boldsymbol{u}}_k(\boldsymbol{x})dS \\
K_3 &= \int_{S_c}\int_{\widetilde{S}_c} \Delta\boldsymbol{u}_i(\boldsymbol{x})\boldsymbol{n}_j(\boldsymbol{x})C_{klab}\Delta\widetilde{\boldsymbol{u}}_k(\boldsymbol{z})n_l(\boldsymbol{z})\frac{\partial}{\partial z_b}\Sigma_{ij}^a(\boldsymbol{z},\boldsymbol{x})dS_z dS_x
\end{aligned}$$

The limits ($\widetilde{S} \to S$) and ($\boldsymbol{z} \to \boldsymbol{y}$) are taken, these formulas become:

$$\begin{aligned}
K_1 &= -\int_{S_u}\int_{S_c} (Ru)_{iq}^D(\boldsymbol{x})B_{ikqs}(\boldsymbol{r})(R\Delta\widetilde{u})_{ks}(\boldsymbol{y})dS_y dS_x \\
&\quad -\int_{S_T}\int_{S_c} (Ru)_{iq}(\boldsymbol{x})B_{ikqs}(\boldsymbol{r})(R\Delta\widetilde{u})_{ks}(\boldsymbol{y})dS_y dS_x \\
K_2 &= \int_{S_T}\int_{S_c} \boldsymbol{t}_k^D(\boldsymbol{x})\Delta\widetilde{\boldsymbol{u}}_i(\boldsymbol{y})\boldsymbol{n}_j(\boldsymbol{y})\Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{y})dS_y dS_x \\
&\quad +\int_{S_u}\int_{S_c} \boldsymbol{t}_k(\boldsymbol{x})\Delta\widetilde{\boldsymbol{u}}_i(\boldsymbol{y})\boldsymbol{n}_j(\boldsymbol{y})\Sigma_{ij}^k(\boldsymbol{x},\boldsymbol{y})dS_y dS_x - \int_{S_c}\boldsymbol{p}_k(\boldsymbol{x})\Delta\widetilde{\boldsymbol{u}}_k(\boldsymbol{x})dS \\
K_3 &= \int_{S_c}\int_{S_c} (R\Delta u)_{iq}(\boldsymbol{x})B_{ikqs}(\boldsymbol{r})(R\Delta\widetilde{u})_{ks}(\boldsymbol{y})dS_y dS_x
\end{aligned} \tag{A.53}$$

A.3 Surface rotors

Surface rotors are defined as:

$$(R\Delta u)_{iq}(\boldsymbol{x}) = e_{bcq}\boldsymbol{n}_b(\boldsymbol{x})\frac{\partial\Delta u_i(\boldsymbol{x})}{\partial\boldsymbol{x}_c} \tag{A.54}$$

$$(R\Delta\widetilde{u})_{ks}(\boldsymbol{y}) = e_{bcs}\boldsymbol{n}_b(\boldsymbol{y})\frac{\partial\Delta\widetilde{u}_k(\boldsymbol{y})}{\partial\boldsymbol{y}_c} \tag{A.55}$$

They express the vector product between the gradient of the argument function and the unit normal vector to the surface:

$$(R\Delta u)_{iq}(\boldsymbol{x}) = (\boldsymbol{n}\wedge\bigtriangledown\Delta\boldsymbol{u}_i)\cdot\boldsymbol{e}_q \quad \text{at } \boldsymbol{x} \tag{A.56}$$

$$(R\Delta\widetilde{u})_{ks}(\boldsymbol{y}) = (\boldsymbol{n}\wedge\bigtriangledown\Delta\boldsymbol{u}_k)\cdot\boldsymbol{e}_s \quad \text{at } \boldsymbol{y} \tag{A.57}$$

where $\boldsymbol{e}_k$ denotes the unit vector along the k^{th} Cartesian axis. By expressing the surface rotor operator in terms of intrinsic coordinates g_1, g_2 by means of the

associated local co-variant base vectors g_1, g_2

$$\begin{aligned}(R\Delta\widetilde{u})_{ks}(\boldsymbol{y}) &= e_{bcs}\boldsymbol{n}_b\frac{\partial\Delta\widetilde{u}_k}{\partial\boldsymbol{y}_c} = e_{bcs}e_{bij}g_{1i}g_{2j}\frac{\partial\Delta\widetilde{u}_k}{\partial\boldsymbol{y}_c}J^{-1}\\ &= (\delta_{ic}\delta_{js}-\delta_{is}\delta_{jc})g_{1i}g_{2j}\frac{\partial\Delta\widetilde{u}_k}{\partial\boldsymbol{y}_c}J^{-1}\\ &= (\frac{\partial\Delta\widetilde{u}_k}{\partial\boldsymbol{\eta}_1}g_{2s}-\frac{\partial\Delta\widetilde{u}_k}{\partial\boldsymbol{\eta}_2}g_{1s})J^{-1}\end{aligned} \tag{A.58}$$

the dependence on the sole in-plane components of the argument function gradient is highlighted.

Appendix B

FMM applied to SGBEM terms

B.1 Terms $\mathbf{B}_{uu}(\mathbf{u}, \tilde{\mathbf{u}})$ and $\mathbf{B}_{uu}(\mathbf{u^D}, \tilde{\mathbf{u}})$

By using the symmetry property of the 4^{th} order tensor B_{ikqs} in the SGBEM formulations, the terms $\mathbf{B}_{uu}(\mathbf{u}, \tilde{\mathbf{u}})$ and $\mathbf{B}_{uu}(\mathbf{u^D}, \tilde{\mathbf{u}})$ can be written as:

$$\begin{aligned}\mathbf{B}_{uu}(\mathbf{u}, \tilde{\mathbf{u}}) &= \int_{S_t}\int_{S_t}(Ru)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \\ &= \int_{S_t}\int_{S_t}(R\tilde{u})_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(Ru)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \qquad \text{(B.1)} \\ \mathbf{B}_{uu}(\mathbf{u^D}, \tilde{\mathbf{u}}) &= \int_{S_t}\int_{S_t}(Ru^D)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \\ &= \int_{S_t}\int_{S_t}(R\tilde{u})_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(Ru^D)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \qquad \text{(B.2)}\end{aligned}$$

The term $\mathbf{B}_{uu}(\mathbf{u}, \tilde{\mathbf{u}})$ is chosen to be evaluated. The expression of B_{ikqs} is given in the simplified form as:

$$B_{ikqs}(\mathbf{r}) = \mu^2[-4\delta_{qs}F_{ik} + (4\delta_{is}\delta_{qs} - 4\nu\delta_{is}\delta_{kq} - 2(1-\nu)\delta_{iq}\delta_{ks})F_{,pp}] \qquad \text{(B.3)}$$

where:

$$F(\mathbf{x}-\tilde{\mathbf{x}}) = \frac{\mathbf{r}}{16\pi\mu(1-\nu)} \qquad \text{(B.4)}$$

$$F_{,pp}(\mathbf{x}-\tilde{\mathbf{x}}) = \frac{1}{8\pi\mu\mathbf{r}(1-\nu)} \qquad \text{(B.5)}$$

$$F_{,ik}(\mathbf{x}-\tilde{\mathbf{x}}) = \frac{1}{16\pi\mu\mathbf{r}(1-\nu)}(\delta_{ik} - \mathbf{r}_{,i}\mathbf{r}_{,k}) \qquad \text{(B.6)}$$

Substituting the equations (B.4) into (B.3) and taking the expansion of the function $1/r$ we obtain:

$$B_{ikqs}(\mathbf{r}) = \frac{\mu}{4\pi(1-\nu)}\left[(\delta_{ik}\delta_{qs} - 2\delta_{is}\delta_{kq}\nu - (1-\nu)\delta_{iq}\delta_{ks})\frac{1}{\mathbf{r}} + \delta_{qs}\frac{\mathbf{r}_{,i}\mathbf{r}_{,k}}{\mathbf{r}}\right] \qquad \text{(B.7)}$$

$$= \frac{\mu}{4\pi(1-\nu)}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\overline{F^{S_{uu}}_{ikqs,n,m}(\overrightarrow{Ox})} + \overline{G^{S_{uu}}_{iqs,n,m}(\overrightarrow{Ox})}(\overrightarrow{Oy_k})\right)R_{n,m}(\overrightarrow{Oy}) \qquad \text{(B.8)}$$

where the functions $F^{S_{uu}}_{ikqs,n,m}$ and $G^{S_{uu}}_{iqs,n,m}$ are defined by:

$$F^{S_{uu}}_{ikqs,n,m}(\overrightarrow{Ox}) = \left[(\delta_{ik}\delta_{qs} - 2\delta_{is}\delta_{kq}\nu - (1-\nu)\delta_{iq}\delta_{ks}) - \delta_{qs}\overrightarrow{Ox}_k\frac{\partial}{\partial \mathbf{x}_i}\right] S_{n,m}(\overrightarrow{Ox}) \tag{B.9}$$

$$G^{S_{uu}}_{iqs,n,m} = \delta_{qs}\frac{\partial}{\partial \mathbf{x}_i}S_{n,m}(\overrightarrow{Ox}) \tag{B.10}$$

And the expression of $\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}})$ can be rewritten as:

$$\begin{aligned}\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}}) =& \frac{\mu}{4\pi(1-\nu)}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_t}(R\tilde{u}_{iq}(\mathbf{x})\left(\overline{F^{S_{uu}}_{ikqs,n,m}(\overrightarrow{Ox})}M^{1uu}_{ks,n,m}(O)\right.\\ &\left.+\overline{G^{S_{uu}}_{iqs,n,m}(\overrightarrow{Ox})}M^{2uu}_{sn,m}(O)\right)dS_x\end{aligned} \tag{B.11}$$

where the multipole moments are:

$$M^{1uu}_{ks,n,m}(O) = \int_{S_t} R_{n,m}(\overrightarrow{O\tilde{x}})(Ru)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}} \tag{B.12}$$

$$M^{2uu}_{sn,m}(O) = \int_{S_t} R_{n,m}(\overrightarrow{O\tilde{x}})(\overrightarrow{O\tilde{x}})_k(Ru)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}} \tag{B.13}$$

Analogous to the treatment of the term $\mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}})$, we obtain the translations of the term $\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}})$:

The M2M translation:

$$M^{1uu}_{ks,n,m}(\mathbf{O}') = \sum_{n'=o}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{OO}'})M^{1uu}_{ks,n-n',m-m'}(\mathbf{O}) \tag{B.14}$$

$$\begin{aligned}M^{2uu}_{s,n,m}(\mathbf{O}') =& \sum_{n'=o}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{OO}'})\left(M^{2uu}_{s,n-n',m-m'}(\mathbf{O})\right.\\ &\left.-(\overrightarrow{\mathbf{OO}'})_k M^{1uu}_{ks,n-n'.m-m'}(\mathbf{O})\right)\end{aligned} \tag{B.15}$$

The M2L translation:

$$L^{1uu}_{ks,n,m}(\mathbf{x_0}) = \sum_{n'=o}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n\overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{Ox_0}})M^{1uu}_{ks,n',m'}(\mathbf{O}) \tag{B.16}$$

$$\begin{aligned}L^{2uu}_{s,n,m}(\mathbf{x_0}) =& \sum_{n'=o}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n\overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{Ox_0}})\left(M^{2uu}_{s,n',m'}(\mathbf{O})\right.\\ &\left.-(\overrightarrow{\mathbf{Ox_0}})_k M^{1uu}_{ks,n',m'}(\mathbf{O})\right)\end{aligned} \tag{B.17}$$

The L2L translation:

$$L^{1uu}_{ks,n,m}(\mathbf{x_1}) = \sum_{n'=o}^{\infty}\sum_{m'=-n'}^{n'}(-1)^n R_{n-n',m'-m}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}})L^{1uu}_{ks,n',m'}(\mathbf{x_0}) \tag{B.18}$$

$$\begin{aligned}L^{2uu}_{s,n,m}(\mathbf{x_1}) = &\sum_{n'=o}^{\infty}\sum_{m'=-n'}^{n'}(-1)^n R_{n-n',m'-m}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}})\left(M^{2uu}_{s,n',m'}(\mathbf{x_0})\right.\\ &\left.-(\overrightarrow{\mathbf{x_0}\mathbf{x_1}})_k M^{1uu}_{ks,n',m'}(\mathbf{x_0})\right)\end{aligned} \tag{B.19}$$

The integral (B.1) can be evaluated via $L^{1uu}_{ks,n,m}$ and $L^{2uu}_{s,n,m}$:

$$\begin{aligned}\mathbf{B}_{uu}(\mathbf{x},\tilde{\mathbf{u}}) = &\frac{\mu}{4\pi(1-\nu)}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_t}(R\tilde{u})_{iq}(\mathbf{x})\left(F^{Ruu}_{ikqs,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}})L^{1uu}_{ks,n,m}(\mathbf{x_1})\right.\\ &\left.+G^{Ruu}_{iqs,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}})L^{2uu}_{sn,m}(\mathbf{x_1})\right)dS_x\end{aligned} \tag{B.20}$$

where $F^{Ruu}_{ikqs,n,m}$ and $G^{Ruu}_{iqs,n,m}$ are defined as:

$$F^{Ruu}_{ikqs,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) = \left[(\delta_{ik}\delta_{qs}-2\delta_{is}\delta_{kq}\nu-(1-\nu)\delta_{iq}\delta_{ks})-\delta_{qs}\overrightarrow{\mathbf{x_1}\mathbf{x}}_k\frac{\partial}{\partial\mathbf{x_i}}\right]R_{n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) \tag{B.21}$$

$$G^{Ruu}_{iqs,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) = \delta_{qs}\frac{\partial}{\partial\mathbf{x}_i}R_{n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) \tag{B.22}$$

B.2 Terms $\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}})$ and $\mathbf{B}_{ut}(\mathbf{u^D},\tilde{\mathbf{t}})$

We now take into consideration the terms $\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}})$ and $\mathbf{B}_{ut}(\mathbf{u^D},\tilde{\mathbf{t}})$. The term $\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}})$ is chosen to be evaluated:

$$\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}}) = -\int_{S_u}\int_{S_t}\tilde{t}_k(\tilde{\mathbf{x}})T^k_i(\tilde{\mathbf{x}},\mathbf{x})u_i(\mathbf{x})dS_x dS_{\tilde{x}} \tag{B.23}$$

$$\mathbf{B}_{ut}(\mathbf{u^D},\tilde{\mathbf{t}}) = -\int_{S_u}\int_{S_u}\tilde{t}_k(\tilde{\mathbf{x}})T^k_i(\tilde{\mathbf{x}},\mathbf{x})u^D_i(\mathbf{x})dS_x dS_{\tilde{x}} \tag{B.24}$$

where the traction fundamental solution is written as:

$$\begin{aligned}T^k_i(\tilde{\mathbf{x}},\mathbf{x}) &= \Sigma^k_{ij}(\tilde{\mathbf{x}},\mathbf{x})n_j(\mathbf{x})\\ &= C_{ijab}\frac{\partial}{\partial\mathbf{x}_b}U^k_a(\tilde{\mathbf{x}},\mathbf{x})n_j(\mathbf{x})\\ &= \frac{1}{8\pi\mu}C_{ijab}\frac{\partial}{\partial\mathbf{x}_b}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\overrightarrow{F^{Stt}_{ka,n,m}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}})}\right.\\ &\left.+\overrightarrow{G^{Stt}_{k,n,m}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}})}(\overrightarrow{\mathbf{O}\mathbf{x}}_a)\right)R_{n,m}(\overrightarrow{\mathbf{O}\mathbf{x}})n_j(\mathbf{x})\end{aligned} \tag{B.25}$$

The functions $F^{Stt}_{ka,n,m}$ and $G^{Stt}_{k,n,m}$ are detailed in the section of term $\mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}})$. Substituting these functions in $\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}})$ we get:

$$\mathbf{B}_u t(\mathbf{u},\tilde{\mathbf{t}}) = -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_k(\tilde{\mathbf{x}})\left(\overline{F^S_{ka,n,m}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}})}M^{1ut}_{a,n,m}(\mathbf{O}) + \overline{G^S_{k,n,m}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}})}M^{2ut}_{n,m}(boldO)\right)dS_{\tilde{x}} \tag{B.26}$$

where the multipole moments are:

$$M^{1ut}_{a,n,m}(\mathbf{O}) = \int_{S_t} C_{ijab}\frac{\partial}{\partial \mathbf{x}_b}R_{n,m}(\overrightarrow{\mathbf{O}\mathbf{x}})n_j(\mathbf{x})u_i(\mathbf{x})dS_x \tag{B.27}$$

$$M^{2ut}_{n,m}(\mathbf{O}) = \int_{S_t} C_{ijab}\frac{\partial}{\partial \mathbf{x}_b}\left[(\overrightarrow{\mathbf{O}\mathbf{x}_a})R_{n,m}(\overrightarrow{\mathbf{O}\mathbf{x}})\right]n_j(\mathbf{x})u_i(\mathbf{x})dS_x \tag{B.28}$$

Again, by following the same principle, we obtain the FMM operations.

The M2M translation:

$$M^{1ut}_{a,n,m}(\mathbf{O}') = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'}R_{n',m'}(\overrightarrow{\mathbf{O}\mathbf{O}'})M^{1ut}_{a,n-n',m-m'}(\mathbf{O}) \tag{B.29}$$

$$M^{2ut}_{n,m}(\mathbf{O}') = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'}R_{n',m'}(\overrightarrow{\mathbf{O}\mathbf{O}'})\left(M^{2ut}_{n-n',m-m'}(\mathbf{O}) - (\overrightarrow{\mathbf{O}\mathbf{O}'})_a M^{1ut}_{a,n-n',m-m'}(\mathbf{O})\right) \tag{B.30}$$

The M2L translation:

$$L^{1ut}_{a,n,m}(\tilde{\mathbf{x}}_\mathbf{0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}(-1)^n\overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}_\mathbf{0}})M^{1ut}_{a,n',m'}(\mathbf{O}) \tag{B.31}$$

$$L^{2ut}_{n,m}(\tilde{\mathbf{x}}_\mathbf{0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}(-1)^n\overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}_\mathbf{0}})\left(M^{2ut}_{n',m'}(\mathbf{O}) - (\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}_\mathbf{0}})_a M^{1ut}_{a,n',m'}(\mathbf{O})\right) \tag{B.32}$$

The L2L translation:

$$L^{1ut}_{a,n,m}(\tilde{\mathbf{x}}_\mathbf{1}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}R_{n'-n,m'-m}(\overrightarrow{\tilde{\mathbf{x}}_\mathbf{0}\tilde{\mathbf{x}}_\mathbf{1}})L^{1ut}_{a,n',m'}(\tilde{\mathbf{x}}_\mathbf{0}) \tag{B.33}$$

$$L^{2ut}_{n,m}(\tilde{\mathbf{x}}_\mathbf{1}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}R_{n'-n,m'-m}(\overrightarrow{\tilde{\mathbf{x}}_\mathbf{0}\tilde{\mathbf{x}}_\mathbf{1}})\left(L^{2ut}_{n',m'}(\tilde{\mathbf{x}}_\mathbf{0}) - (\overrightarrow{\tilde{\mathbf{x}}_\mathbf{0}\tilde{\mathbf{x}}_\mathbf{1}})_a L^{1ut}_{a,n',m'}(\tilde{\mathbf{x}}_\mathbf{0})\right) \tag{B.34}$$

So the integral $\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}})$ is computed via $L^{1ut}_{a,n,m}$ and $L^{2ut}_{n,m}$ as:

$$\mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}}) = -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_k(\tilde{\mathbf{x}})\left[F^R_{ka,n,m}(\overrightarrow{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}})L^{1ut}_{a,n,m}(\tilde{\mathbf{x}}_1)\right.$$
$$\left.+G^R_{k,n,m}(\overrightarrow{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}})L^{2ut}_{n,m}(\tilde{\mathbf{x}}_1)\right]dS_{\tilde{x}} \tag{B.35}$$

B.3 Terms $\mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}})$ and $\mathbf{B}_{tu}(\mathbf{t^D},\tilde{\mathbf{u}})$

We choose the term $\mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}})$ to evaluate:

$$\mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}}) = -\int_{S_u}\int_{S_t}t_k(\mathbf{x})T^k_i(\mathbf{x},\tilde{\mathbf{x}})\tilde{u}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{B.36}$$

$$\mathbf{B}_{tu}(\mathbf{t^D},\tilde{\mathbf{u}}) = -\int_{S_t}\int_{S_t}t^D_k(\mathbf{x})T^k_i(\mathbf{x},\tilde{\mathbf{x}})\tilde{u}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{B.37}$$

(B.38)

This term can also be written as:

$$\mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}}) = -\int_{S_t}\int_{S_u}\tilde{u}_k(\mathbf{x})T^i_k(\tilde{\mathbf{x}},\mathbf{x})t_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{B.39}$$

The fundamental solution of traction can also be written as:

$$\begin{aligned}
T^i_k(\tilde{\mathbf{x}},\mathbf{x}) =& \Sigma^i_{kj}n_j(\mathbf{x})\\
=& C_{kjab}\frac{\partial}{\partial\mathbf{x}_b}U^i_a(\tilde{\mathbf{x}},\mathbf{x})n_j(\mathbf{x})\\
=& C_{kjab}\frac{\partial}{\partial\mathbf{x}_b}U^i_a(\mathbf{x},\tilde{\mathbf{x}})n_j(\mathbf{x})\\
=& C_{kjab}\frac{\partial}{\partial\mathbf{x}_b}U^a_i(\mathbf{x},\tilde{\mathbf{x}})n_j(\mathbf{x})\\
=& -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}C_{kjab}\frac{\partial}{\partial\mathbf{x}_b}\left(\overline{F^S_{ai,n,m}(\overrightarrow{\mathbf{Ox}})}\right.\\
&\left.+\overline{G^S_{a,n,m}(\overrightarrow{\mathbf{Ox}})}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}_i})\right)R_{n,m}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}})n_j(\mathbf{x})
\end{aligned} \tag{B.40}$$

The functions $F^{Stt}_{ka,n,m}$ and $G^{Stt}_{k,n,m}$ are defined in the section of term $\mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}})$. Substituting these elements in equation (B.39) we get:

$$\mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}}) = -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_t}\tilde{\mathbf{u}}_k(\mathbf{x})C_{kjab}n_j(\mathbf{x})\frac{\partial}{\partial\mathbf{x}_b}\left(\overline{F^{Stt}_{ai,n,m}(\overrightarrow{\mathbf{Ox}})}M^{1tu}_{i,n,m}(\mathbf{O})\right.$$
$$\left.+\overline{G^{Stt}_{a,n,m}\overrightarrow{\mathbf{Ox}}}M^{2tu}_{n,m}(\mathbf{O})\right)dS_x \tag{B.41}$$

The multipole moments are:

$$M^{1tu}_{i,n,m}(\mathbf{O}) = \int_{S_u} R_{n,m}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}}) t_i(\tilde{\mathbf{x}}) dS_{\tilde{x}} \tag{B.42}$$

$$M^{2tu}_{n,m}(\mathbf{O}) = \int_{S_u} R_{n,m}(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}})(\overrightarrow{\mathbf{O}\tilde{\mathbf{x}}_i}) t_i(\tilde{\mathbf{x}}) dS_{\tilde{x}} \tag{B.43}$$

The M2M translation:

$$M^{1tu}_{i,n,m}(\mathbf{O}') = \sum_{n'=0}^{n} \sum_{m'=n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{O}'\mathbf{O}}) M^{1tu}_{i,n-n',m-m'}(\mathbf{O}) \tag{B.44}$$

$$\begin{aligned} M^{2tu}_{n,m}(\mathbf{O}') = \sum_{n'=0}^{n} \sum_{m'=n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{O}'\mathbf{O}}) \big[& M^{2tu}_{n-n',m-m'}(\mathbf{O}) \\ & - (\overrightarrow{\mathbf{O}\mathbf{O}'})_i M^{1tu}_{i,n-n',m-m'}(\mathbf{O}) \big] \end{aligned} \tag{B.45}$$

The M2L translation:

$$L^{1tu}_{i,n,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} (-1)^n \overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{O}\mathbf{x_0}}) M^{1tu}_{i,n',m'}(\mathbf{O}) \tag{B.46}$$

$$\begin{aligned} L^{1tu}_{i,n,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} (-1)^n \overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{O}\mathbf{x_0}}) \big[& M^{2tu}_{n',m'}(\mathbf{O}) \\ & - (\overrightarrow{\mathbf{O}\mathbf{x_0}})_i M^{1tu}_{i,n',m'}(\mathbf{O}) \big] \end{aligned} \tag{B.47}$$

The L2L translation:

$$L^{1tu}_{i,n,m}(\mathbf{x_1}) = \sum_{n'=n}^{\infty} \sum_{m'=-n'}^{n'} R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}}) L^{1tu}_{i,n',m'}(\mathbf{x_0}) \tag{B.48}$$

$$\begin{aligned} L^{2tu}_{n,m}(\mathbf{x_1}) = \sum_{n'=n}^{\infty} \sum_{m'=-n'}^{n'} R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}}) \big[& L^{2tu}_{n',m'}(\mathbf{x_0}) \\ & - (\overrightarrow{\mathbf{x_0}\mathbf{x_1}})_i L^{1tu}_{i,n',m'}(\mathbf{x_0}) \big] \end{aligned} \tag{B.49}$$

The term $\mathbf{B}_{tu}(\mathbf{t}, \tilde{\mathbf{u}})$ is therefore computed as:

$$\begin{aligned} \mathbf{B}_{tu}(\mathbf{t}, \tilde{\mathbf{u}}) = - \frac{1}{8\pi\mu} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \int_{S_t} \tilde{u}_k(\mathbf{x}) C_{kjab} n_j(\mathbf{x}) \frac{\partial}{\partial x_b} \big(& F^{Rtt}_{ai,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) L^{1tu}_{i,n,m}(\mathbf{x_1}) \\ & + G^{Rtt}_{a,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) L^{2tu}_{n,m}(\mathbf{x_1}) \big) \, dS_x \end{aligned} \tag{B.50}$$

where the functions $F^{Rtt}_{ai,n,m}$ and $F^{Gtt}_{a,n,m}$ are defined in the section of the term

$\mathbf{B}_{tt}(\mathbf{t}, \tilde{\mathbf{t}})$. We can evaluate the 2^{nd} order derivative of these functions as:

$$\begin{aligned}\frac{\partial}{\partial x_b}F^{Rtt}_{ai,n,m}(\overrightarrow{\mathbf{x_1x}}) =& \frac{1}{2(1-\nu)}\left((3-4\nu)\delta_{ai}\frac{\partial}{\partial x_b}R_{n,m}(\overrightarrow{\mathbf{x_1x}})\right.\\ &\left.-\frac{\partial}{\partial x_b}\left[(\overrightarrow{\mathbf{x_1x}})_i\frac{\partial}{\partial x_a}R_{n,m}(\overrightarrow{\mathbf{x_1x}})\right]\right)\\ =& \frac{1}{2(1-\nu)}\left((3-4\nu)\delta_{ai}\frac{\partial}{\partial x_b}R_{n,m}(\overrightarrow{\mathbf{x_1x}})\right.\\ &\left.-\left[(\overrightarrow{\mathbf{x_1x}})_i\frac{\partial}{\partial x_b}\frac{\partial}{\partial x_a}R_{n,m}(\overrightarrow{\mathbf{x_1x}})+\frac{\partial}{\partial x_b}(\overrightarrow{\mathbf{x_1x}})_i+\frac{\partial}{\partial x_a}R_{n,m}(\overrightarrow{\mathbf{x_1x}})\right]\right)\end{aligned} \tag{B.51}$$

$$\frac{\partial}{\partial x_b}G^{Rtt}_{a,n,m}(\overrightarrow{\mathbf{x_1x}}) = \frac{1}{2(1-\nu)}\frac{\partial}{\partial x_b}\frac{\partial}{\partial x_a}R_{n,m}(\overrightarrow{\mathbf{x_1x}}) \tag{B.52}$$

B.4 Terms $\mathbf{B}_{tu2}(\mathbf{t^D}, \tilde{\mathbf{u}})$ and $\mathbf{B}_{ut2}(\mathbf{u^D}, \tilde{\mathbf{t}})$

We choose the term $\mathbf{B}_{ut2}(\mathbf{u^D}, \tilde{\mathbf{t}})$ to evaluate:

$$\mathbf{B}_{ut2}(\mathbf{u^D}, \tilde{\mathbf{t}}) = -\int_{S_u}\int_{\partial\Omega}\tilde{t}_k(\tilde{\mathbf{x}})u^D_i(\tilde{\mathbf{x}})T^k_i(\tilde{\mathbf{x}}, \mathbf{x})dS_x dS_{\tilde{x}} \tag{B.53}$$

$$\mathbf{B}_{tu2}(\mathbf{t^D}, \tilde{\mathbf{u}}) = -\int_{S_t}\int_{\partial\Omega}t^D_k(\mathbf{x})\tilde{u}_i(\mathbf{x})T^k_i(\mathbf{x}, \tilde{\mathbf{x}})dS_{\tilde{x}} dS_x \tag{B.54}$$

The formula of the term $\mathbf{B}_{ut2}(\mathbf{u^D}, \tilde{\mathbf{t}})$ is expressed under the expansion form as:

$$\begin{aligned}\mathbf{B}_{ut2}(\mathbf{u^D}, \tilde{\mathbf{t}}) =& -\int_{S_u}\int_{\partial\Omega}\tilde{t}_k(\tilde{\mathbf{x}})u^D_i(\tilde{\mathbf{x}})\frac{1}{8\pi\mu}C_{ijab}\frac{\partial}{\partial x_b}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\overline{F^{Stt}_{ka,n,m}(\overrightarrow{\mathbf{O\tilde{x}}})}\right.\\ &\left.+\overline{G^{Stt}_{k,n,m}(\overrightarrow{\mathbf{O\tilde{x}}})}(\overrightarrow{\mathbf{Ox}}_a)\right)R_{n,m}(\overrightarrow{\mathbf{Ox}})n_j(\mathbf{x})dS_x dS_{\tilde{x}}\\ =& \frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_k(\tilde{\mathbf{x}})u^D_i(\tilde{\mathbf{x}})\left(\overline{F^{Stt}_{ka,n,m}(\overrightarrow{\mathbf{O\tilde{x}}})}M^{1ut2}_{ia,n,m}(\mathbf{O})\right.\\ &\left.+\overline{G^{Stt}_{k,n,m}(\overrightarrow{\mathbf{O\tilde{x}}})}M^{2ut2}_{in,m}(\mathbf{O})\right)dS_{\tilde{x}}\end{aligned} \tag{B.55}$$

where the multipole moments are:

$$M^{1ut2}_{ia,n,m}(\mathbf{O}) = \int_{\partial\Omega}C_{ijab}\frac{\partial}{\partial x_b}R_{n,m}(\overrightarrow{\mathbf{Ox}})n_j(\mathbf{x})dS_x \tag{B.56}$$

$$M^{2ut2}_{in,m}(\mathbf{O}) = \int_{\partial\Omega}C_{ijab}\frac{\partial}{\partial x_b}R_{n,m}\left[(\overrightarrow{\mathbf{Ox}}_a)R_{n,m}(\overrightarrow{\mathbf{Ox}})\right]n_j(\mathbf{x})dS_x \tag{B.57}$$

The M2M translation:

$$M^{1ut2}_{ia,n,m}(\mathbf{O}') = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{O}'\mathbf{O}})M^{1ut2}_{ia,n-n',m-m'}(\mathbf{O}) \tag{B.58}$$

$$M^{2ut2}_{in,m}(\mathbf{O}') = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{O}'\mathbf{O}})\left(M^{2ut2}_{i,n-n',m-m'}(\mathbf{O}) - (\overrightarrow{\mathbf{O}'\mathbf{O}})_a M^{1ut2}_{ia,n-n',m-m'}(\mathbf{O})\right) \tag{B.59}$$

The M2L translation:

$$L^{1ut2}_{ia,n,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n \overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{O}\mathbf{x_0}})M^{1ut2}_{ia,n',m'}(\mathbf{O}) \tag{B.60}$$

$$L^{2ut2}_{in,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n \overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{O}\mathbf{x_0}})\left(M^{2ut2}_{i,n',m'}(\mathbf{O}) - (\overrightarrow{\mathbf{O}\mathbf{x_0}})_a M^{1ut}_{ia,n',m'}(\mathbf{O})\right) \tag{B.61}$$

The L2L translation:

$$L^{1ut2}_{ia,n,m}(\mathbf{x_1}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n R_{n'-n,m'-m}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}})L^{1ut2}_{ia,n',m'}(\mathbf{x_0}) \tag{B.62}$$

$$L^{2ut2}_{in,m}(\mathbf{x_1}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n R_{n'-n,m'-m}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}})\left(L^{2ut2}_{i,n',m'}(\mathbf{x_0}) - (\overrightarrow{\mathbf{x_0}\mathbf{x_1}})_a L^{1ut2}_{ia,n-n',m-m'}(\mathbf{x_0})\right) \tag{B.63}$$

The integral is then evaluated with the help of $L^{1ut2}_{ia,n,m}$ and $L^{2ut2}_{i,n,m}$:

$$\mathbf{B}_{ut2}(\mathbf{u^D},\tilde{\mathbf{t}}) = -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_k(\tilde{\mathbf{x}})u^D_i(\tilde{\mathbf{x}})\left[F^R_{ka,n,m}(\overrightarrow{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}})L^{1ut2}_{ia,n,m}(\tilde{\mathbf{x}_1}) + G^R_{k,n,m}(\overrightarrow{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}})L^{2ut2}_{in,m}(\tilde{\mathbf{x}_1})\right]dS_{\tilde{x}} \tag{B.64}$$

B.5 FMM - Useful formulas

This section provides the practical computation of the solid harmonics $R_{n,m}(\mathbf{y})$ and $S_{n,m}(\mathbf{x})$ and the derivatives of the $R_{n,m}$ and $S_{n,m}(\mathbf{x})$ using only the Cartesian coordinates of the generic arguments $\mathbf{x}$ and $\mathbf{y}$. The brief description of the recursive procedure can be summarized as follows:
(a) The $R_{n,m}(\mathbf{y})$ are computed recursively by setting $R_{0,0}(\mathbf{y}) = 1$ and using:

$$R_{n+1,n+1}(\mathbf{y}) = \frac{y_1 + iy_2}{2(n+1)}R_{n,n}(\mathbf{y}) \tag{B.65}$$

$$\left((n+1)^2 - m^2\right)R_{n+1,m}(\mathbf{y}) - (2n+1)y_3 R_{n,m}(\mathbf{y}) + \|\mathbf{y}\|^2 R_{n-1,m}(\mathbf{y}) = 0 \tag{B.66}$$

(b) The $S_{n,m}(\mathbf{x})$ are computed recursively by setting $S_{0,0}(\mathbf{x}) = \dfrac{1}{\|\mathbf{x}\|}$ and using:

$$S_{n+1,n+1}(\mathbf{x}) = \frac{(2n+1)(x_1+ix_2)}{\|\mathbf{x}\|^2} S_{n,n}(\mathbf{x}) \tag{B.67}$$

$$\|\mathbf{x}\|^2 S_{n+1,m}(\mathbf{x}) - (2n+1)x_3 S_{n,m}(\mathbf{x}) + (n^2-m^2)S_{n-1,m}(\mathbf{x}) = 0 \tag{B.68}$$

(c) Finally, the negative terms are computed ($n > m$) following the properties:

$$R_{n,-m}(\mathbf{y}) = (-1)^m \overline{R_{n,m}(\mathbf{y})} \tag{B.69}$$

$$S_{n,-m}(\mathbf{x}) = (-1)^m \overline{S_{n,m}(\mathbf{x})} \tag{B.70}$$

The first order derivatives of $R_{n,m}$ and $S_{n,m}$ are computed via:

$$\frac{\partial}{\partial y_1} R_{n,m} = \frac{1}{2}(R_{n-1,m-1} - R_{n-1,m+1}) \tag{B.71}$$

$$\frac{\partial}{\partial y_2} R_{n,m} = \frac{i}{2}(R_{n-1,m-1} - R_{n-1,m+1}) \tag{B.72}$$

$$\frac{\partial}{\partial y_3} R_{n,m} = R_{n-1,m} \tag{B.73}$$

$$\frac{\partial}{\partial x_1} S_{n,m} = \frac{1}{2}(S_{n+1,m-1} - R_{n+1,m+1}) \tag{B.74}$$

$$\frac{\partial}{\partial x_2} S_{n,m} = \frac{i}{2}(S_{n+1,m+1} - S_{n+1,m-1}) \tag{B.75}$$

$$\frac{\partial}{\partial x_3} S_{n,m} = -S_{n+1,m} \tag{B.76}$$

And the second order derivatives:

$$\frac{\partial}{\partial y_1}\frac{\partial}{\partial y_1} R_{n,m} = \frac{1}{4}(R_{n-2,m-2} - 2R_{n-2,m} + R_{n-2,m+2}) \tag{B.77}$$

$$\frac{\partial}{\partial y_1}\frac{\partial}{\partial y_2} R_{n,m} = \frac{\partial}{\partial y_2}\frac{\partial}{\partial y_1} R_{n,m} = \frac{i}{4}(R_{n-2,m-2} - R_{n-2,m+2}) \tag{B.78}$$

$$\frac{\partial}{\partial y_1}\frac{\partial}{\partial y_3} R_{n,m} = \frac{\partial}{\partial y_3}\frac{\partial}{\partial y_1} R_{n,m} = \frac{1}{2}(R_{n-2,m-1} - R_{n-2,m+1}) \tag{B.79}$$

$$\frac{\partial}{\partial y_2}\frac{\partial}{\partial y_2} R_{n,m} = -\frac{1}{4}(R_{n-2,m-2} + 2R_{n-2,m} + R_{n-2,m+2}) \tag{B.80}$$

$$\frac{\partial}{\partial y_2}\frac{\partial}{\partial y_3} R_{n,m} = \frac{\partial}{\partial y_3}\frac{\partial}{\partial y_2} R_{n,m} = \frac{i}{2}(R_{n-2,m-1} + R_{n-2,m+1}) \tag{B.81}$$

$$\frac{\partial}{\partial y_3}\frac{\partial}{\partial y_3} R_{n,m} = R_{n-2,m} \tag{B.82}$$

$$\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_1}S_{n,m} = \frac{1}{4}(S_{n+2,m-2} - 2S_{n+2,m} + S_{n+2,m+2}) \tag{B.83}$$

$$\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}S_{n,m} = \frac{\partial}{\partial x_2}\frac{\partial}{\partial x_1}S_{n,m} = \frac{i}{4}(S_{n+2,m-2} - S_{n+2,m+2}) \tag{B.84}$$

$$\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_3}S_{n,m} = \frac{\partial}{\partial x_3}\frac{\partial}{\partial x_1}S_{n,m} = -\frac{1}{2}(S_{n+2,m-1} - S_{n+2,m+1}) \tag{B.85}$$

$$\frac{\partial}{\partial x_2}\frac{\partial}{\partial x_2}S_{n,m} = -\frac{1}{4}(S_{n+2,m-2} + 2S_{n+2,m} + S_{n+2,m+2}) \tag{B.86}$$

$$\frac{\partial}{\partial x_2}\frac{\partial}{\partial x_3}S_{n,m} = \frac{\partial}{\partial x_3}\frac{\partial}{\partial x_2}S_{n,m} = -\frac{i}{2}(S_{n+2,m-1} + S_{n+2,m+1}) \tag{B.87}$$

$$\frac{\partial}{\partial x_3}\frac{\partial}{\partial x_3}S_{n,m} = S_{n+2,m} \tag{B.88}$$

Appendix C

Description of *MBEMv3.0*

C.1 General overview

MBEMv3.0 is a numerical calculation code that allows the simulation of three dimensional crack propagation problems in multi-cracked engineering structures. The code is based on the symmetric Galerkin boundary element method coupled with the fast multipole method. The multizone formulation of the method is implemented so that piece-wise homogeneous domains can be simulated. A nested Flexible GMRES is used for the iterative resolution of the obtained matrix system. The code can run in parallel and thus take advantage of the benefits of the multiprocessor numerical environment. *MBEMv3.0* is a Fortran code written with visual studio and compiled by Intel parallel studio. The complete resolution of a problem, using *MBEMv3.0*, is composed of three steps: pre-processing (domain geometry and surface mesh generation, definition of input files, etc.), Main processing (the formation of the system matrix and its resolution) and post-processing (creating output files, calculation time, visualization, etc.). In the following, a generic example of the method adopted in these three steps is given, and the software used for the mesh generation and for the visualization of the results are listed.

C.2 Input files and Pre-processing

Basic data file

This file contains all the geometrical data such as node coordinates, the connectivity table, the number of Gauss points for the integral evaluation, etc. The parts of a data file (*project.inp*) are shown in Listing C.1. For a cracked domain, the solid mesh file and the crack mesh file can be separated. The data file consists of six parts:

- PROBLEM. This section is devoted to defining the number of Gauss points used for various double integrals. The two numbers correspond respectively to the number of Gauss points for singular integrals (recommended value: 4) and the number of Gauss points for regular integrals (recommended value: 2).
- NODE. In this section, the coordinates of all nodes are given.
- ELEMENT. In this section, the connectivity table of all elements is given.

```
1  *PROBLEM
2     4    2
3  **
4  *NODE
5        1          3555          0          0
6        ...
7  **
8  *ELEMENT
9  TIPO
10 5
11           1       4       7      3       1
12           ...
13 **
14 *INTERFACE
15 *BODY
16 TIPO
17 1
18 GENERATE
19 2968       3223
20 **
21 *GENCONSTR
22 TIPO
23 1    1
24       1
25       256
26 **
27 *GENTRAC
28 TIPO
29 3       1.00000e+00
30       1416       1431
31 **
```

Listing C.1 – Content of a data file

- INTERFACE. Each interface is described with the list of its elements. In this example, elements from 2968 to 3223 are located on interface number 1.

- GENCONSTR. Displacement boundary conditions are given in this section. For this example, elements from 1 to 256 are blocked in x direction.

- GENTRAC. Tension boundary conditions are given in this section. For this example, a stress in z direction of value 1 MPa is prescribed on elements from 1416 to 1431.

Parameter file

It is a file in which all the necessary parameters concerning the iterative solver and concerning the octree structure are given. The content of a parameter file is shown in Listing C.2. This file consists of two parts:

```
*SOLVER
  SOLVER TYPE=FGMRES
  PRECISION=    1E-3
  PRECONDITION= LEFT
  ORTH= MODIF_G-S
  MAX ITERATIONS=   1000
  RESTART PARAMETER=   200
*END

*OCTREE
  MAX ELEMENT IN A LEAF=100
  ORDER=7
*END
```

Listing C.2 – Content of a parameter file

- SOLVER. This section defines all the parameters needed by the iterative solver. The first is the solver type, only GMRES is currently valid. The relative error used to stop GMRES is defined in the precision variable. In the precondition variable, four values are possible: NO, LEFT, RIGHT, DOUBLE (Left-right preconditioning). *ORTH* is the orthogonalization procedure used by the solver. The possible values are: MODIF_G-S (Modified Gram-Schmidt orthogonalization), ITER_MODIF_G-S (Iterative Modified Gram-Schmidt orthogonalization), CLASS_G-S (Classical Gram-Schmidt orthogonalization), ITER_CLASS_G-S (Iterative Classical Gram-Schmidt orthogonalization). The parameter *MAX ITERATIONS* sets the maximum number of iterations for the iterative solver (recommended value: 1000). Finally the restart parameter is given through the variable *RESTART PARAMETER* (recommended value: 100).

- OCTREE. In this section, the main variables related to the fast multipole method are given. The maximum number of elements in leaf is given in the variable *MAX ELEMENT IN A LEAF* (recommended value: 100) and the truncation parameter of the infinite series is fixed in the variable *ORDER* (recommended value: 7).

Additional parameter file

In order to avoid variable changes within the subroutines, an additional parameter file can be used. An example of this file is presented in Listing C.3. *FILESLAB* and *FILECRACK* are the names of data files that the code should read. *Ep_crack* is a factor by which the coordinates of the crack mesh will be multiplied. This allows to change the size of cracks without effort. *Ncrack* permits to construct a crack array. The three values are the number of cracks in x, y and z direction. Then *Dcrack* gives the distances between the cracks in the crack array. The position of the cracks or the crack array can be moved by applying a translation. The coordinates of the

translation vector are given in the variable *Movecrack*. *CYCLE_N_MAX* is the maximum number of cycles for a crack propagation simulation. The minimum and the maximum crack advancement are given in *da_min_max*. Finally, *n_threads_1*, *n_threads_2* and *n_threads_3* are the number of threads used for the nested parallelism.

```
1  FILESLAB      project.inp
2  FILECRACK     crack.inp
3  ----------------------------------------
4  Ep_crack      10
5  Ncrack        1    1    1
6  Dcrack        100  100  100
7  Movecrack     500  500  500
8  ----------------------------------------
9  CYCLE_N_MAX   10
10 da_min_max    0.25      2.0
11 ----------------------------------------
12 n_threads_1   20
13 n_threads_2   20
14 n_threads_3   20
15 ----------------------------------------
```

Listing C.3 – Content of an additionnal parameter file

Preprocessing

In our work, the geometric mesh described in the data file is automatically generated with the help of **GiD** software. Once the input files are ready, the calculation can start. First, parameters and mesh data are read, the cracks are modified following the quarter-point scheme and the geometries are concatenated to form the global arrays. Then we proceed to the management of unknowns, a node must be connected to a type of unknown either in traction or in displacement, except for nodes located on an interface. The unknowns are numbered from the node and direction numbering with the subroutine *unkn_manager*. The pre-processing ends with an important step for the variational integrals accelerated by the fast multipole method: the generation of the octree structure. This task is performed by a recursive subroutine named *build_octree*. At the first step ($level = 0$ or *roof*), a cube which contains the whole studied solid Ω is generated. It is the only cell in the octree structure that has no parents. This cube is then divided into eight equal and smaller cubes ($level = 1$) whose edge length is half of the parent cube's. All these cubes must be subdivided into eight small cubes in the same way. The cell subdivision is continued until the number of elements in a cell is smaller than the given parameter *max_elem*. Any given boundary element is deemed to belong to one cell of a given level only, even if it is geometrically shared by two or more same-level cells. If the octree structure has less than two levels, the fast multipole method is disabled, and the computing mode is set to SGBEM.

C.3 Main processing

After the pre-processing, Gauss points, shape functions and their derivatives are computed. The main processing can be subdivided into two: the preparation phase and the solution phase.

Preparation phase

During this step, the known vector and the matrix coefficients are computed. In SGBEM mode, the integrals are computed, and the known vector and the matrix can be obtained. In FM-SGBEM mode, the known vector for near interactions b_{NEAR} is first computed and then that for far enough interactions b_{FMM} is computed with multipole operations (*upward*, *downward*, etc). These two vectors are summed to form the global known vector. For the system matrix, only the matrix for near interactions K_{NEAR} is computed.

Solution phase

In SGBEM mode, the solution phase consists simply of the iterative resolution with the GMRES solver. In FM-SGBEM mode the matrix-vector product must be computed with multipole operations: *upward*, *downward* and integral evaluation.
Upward : A solution $\mathbf{X}$ of the system equation is chosen as a null vector. Then, the multipole moments to the center of the cell, are evaluated at the leaf cells. This process is started from the deepest level (the largest number of level numbering) in the octree and proceeds to the lower levels. For the other cells, multipole moments can be calculated by using the contribution of child cells to the higher level. M2M equations are then used.
Downward: In this step, local expression for all cells from second level is calculated. The local expression for a generic C (center x_1, level $\mathbf{l}$) cell consists of two parts. The first presents the contribution of the boundary elements which are in the cells of the interaction list of C. The M2L transformation is then used. The second part consists of the contribution of elements in the cells that are not adjacent to the parent cell of C, center x_0, level $\mathbf{l-1}$. We can thus evaluate a M2L transformation with the center x_0 and then a L2L transformation (poles x_0 and x_1).
Integral evaluation: The contribution of integrals must be transformed from the center of the cell to all the nodes of that cell.

By this procedure, the matrix-vector product can be computed, and the solution vector can be obtained when the relative error is less than the given tolerance.

C.4 Post-process and Output files

After execution, the program generates several files containing results and information about the calculation process. The simple one is the computed solution vector. But it is difficult to use this file directly since it contains all the results

(displacement and tension) in the unknowns's order. Files with results by nodes or elements are therefore more interesting.

Result per node

The nodal displacement (Listing C.4) in x,y,z directions as well as the nodal tensions (Listing C.5) are written.

```
1  NODE          UX                  UY                  UZ
2
3      1    0.00000000D+00      0.00000000D+00      0.00000000D+00
4  ...
```

Listing C.4 – DISP_NODES.txt: Nodal displacements

```
1  NODE          TX                  TY                  TZ
2
3      1    0.45855680D-04     -0.41087755D-04      0.34438641D-03
4  ...
```

Listing C.5 – TRAC_NODES.txt: Nodal tensions

Result per element

For each element the nodal displacement (Listing C.6) in x,y,z directions as well as the nodal tensions (Listing C.7) are written.

```
1                1
2    2505    0.96225909D-04      0.21316566D-03     -0.46321395D-03
3    2462    0.12050645D-03      0.12688042D-03     -0.80216415D-03
4    2412    0.21112168D-03      0.92256370D-04     -0.44277922D-03
5    2459    0.14611115D-03      0.14099940D-03     -0.26507434D-03
6     ...
```

Listing C.6 – DISP_ELEM.txt: displacement per element

```
1                1
2    2505    0.00000000D+00      0.00000000D+00      0.00000000D+00
3    2462    0.00000000D+00      0.00000000D+00      0.00000000D+00
4    2412    0.00000000D+00      0.00000000D+00      0.00000000D+00
5    2459    0.00000000D+00      0.00000000D+00      0.00000000D+00
6     ...
```

Listing C.7 – TRAC_ELEM.txt: tension per element

Report files

The report file (Report.txt) contains a general report of the simulation. Details about the number of elements, the number of nodes, the parameters used, the number of unknowns, the CPU time spent in each part of the program, and the number of iterations are presented. An example of this file is presented in Listing C.8.

```
1  NODES                          =          499
2  ELEMENTS                       =          288
3  DDLs                           =         1353
4  ------------------------------------------------
5  NODES IN SLAB                  =          194
6  ELEMENTS IN SLAB               =          192
7  DDLs of SLAB                   =          582
8  ------------------------------------------------
9  NUMBER OF CRACKS               =            1
10 NODES IN CRACKS                =          161
11 ELEMENTS IN CRACKS             =           96
12 DDLs of CRACKS                 =          483
13 ------------------------------------------------
14             PARAMETERS
15 SOLVERTYPE                     = Flexible GMRES
16 TOLERANCE                      =         1D-3
17 MAX_ELEM IN LEAF               =           30
18 RESTART_PARAMETER              =          100
19 ORDER                          =            7
20 GAUSS POINT SLAB               =            2
21 GAUSS POINT CRACK              =            4
22 ------------------------------------------------
23 TRACTIONS                      =            0
24 DISPLACEMENTS                  =         1353
25 TOTAL UNKNOWNs                 =         1353
26 ------------------------------------------------
27 NUMBER OF ITERATIONS           =            6
28 ------------------------------------------------
29                                  TIME (sec.)
30 PREPROCESSING PHASE            =          0.1
31 KNEAR and VECT_Y EVALUATION    =          4.9
32 SOLUTION PHASE                 =         15.7
33 POSTPROCESSING PHASE           =          0.4
34 TOTAL TIME                     =         61.1
```

Listing C.8 – REPORT.txt: Report file

To avoid repeating this same file in crack propagation simulation, a special file is generated. This file contains the number of unknowns and the calculation duration of each propagation cycle. An example is presented in Listing C.9.

```
1  ********************************
2  Studying cyle      1 / 15831 ddls
3  Time near start    11 : 31 : 40
4  Time near end      11 : 33 : 08
5  Number of iteration     80
6  Time GMRES start   11 : 33 : 08
7  Time GMRES end     11 : 37 : 40
8  This cycle start   11 : 31 : 40
9  This cycle end     11 : 37 : 42
10
11 ********************************
12 Studying cyle      2 / 16263 ddls
13 Time near start    11 : 37 : 42
14 Time near end      11 : 37 : 52
15 Number of iteration     7
16 Time GMRES start   11 : 37 : 52
17 Time GMRES end     11 : 38 : 22
18 This cycle start   11 : 37 : 42
19 This cycle end     11 : 38 : 23
```

Listing C.9 – prop_info.txt: Propagation file

Error file

This file contains a description and the location of errors encountered during the calculation that may cause the program to be terminated. For example, in Listing C.10, the impossibility to find the data file is notified.

```
1
2     Can not find: road_project_10.inp
3
4 ANALYSIS TERMINATED
```

Listing C.10 – ERROR.err: input file nor found

Visualization

For visualization purposes, routines are created to generate files in .mesh and .bb format, MEDIT [192] type files allowing 3-D visualization.

Bibliography

[1] A. Dansou, S. Mouhoubi, C. Chazallon, and M. Bonnet. Modeling multi-crack propagation by the fast multipole symmetric galerkin bem. *Engineering Analysis with Boundary Elements*, 106:309 – 319, 2019. (Cited on pages xv and 128.)

[2] A. Dansou, S. Mouhoubi, and C. Chazallon. Optimizations of a fast multipole symmetric galerkin boundary element method code. *Numerical Algorithms*, Aug 2019. (Cited on pages xv and 128.)

[3] C. Chazallon, S. Mouhoubi, and A. Dansou. Modèles de dégradation des structures. *Revue générale des routes et de l'aménagement*, 963, 2019. (Cited on pages xv and 129.)

[4] A. Dansou, S. Mouhoubi, C. Chazallon, and M. Bonnet. Modeling crack propagation in 3d heterogeneous multi-cracked roads by Fast Multipole Symmetric Galerkin Boundary Element Method. Paris, France, June 2018. (Cited on pages xv and 129.)

[5] A. Dansou, S. Mouhoubi, and C. Chazallon. Data reusing techniques to accelerate a crack propagation boundary element method code. Braunschweig, Germany, June 2019. (Cited on pages xv and 129.)

[6] A. Dansou, S. Mouhoubi, C. Chazallon, and M. Bonnet. A parallel boundary element method code to simulate multicracked structures. Giens, France, May 2019. (Cited on pages xv and 129.)

[7] M. Heyder and G. Kuhn. 3D fatigue crack propagation: Experimental studies. *International Journal of Fatigue*, 28(5):627 – 634, 2006. (Cited on page 2.)

[8] S. Dong, L. b. Zhang, H. Zhang, Y. Chen, and H. Zhang. Experimental study of unstable crack propagation velocity for x80 pipeline steel. *Fatigue & Fracture of Engineering Materials & Structures*, 42(4):805–817, 2019. (Cited on page 2.)

[9] D. M. Tracey. Finite elements for determination of crack tip elastic stress intensity factors. *Engineering Fracture Mechanics*, 3(3):255 – 265, 1971. (Cited on page 4.)

[10] R. S. Barsoum. On the use of isoparametric finite element in linear fracture mechanics. *Int. J. Num. Meth. Eng.*, 10:25–37, 1976. (Cited on page 4.)

[11] T. Bittencourt, P. Wawrzynek, A. Ingraffea, and J. Sousa. Quasi-automatic simulation of crack propagation for 2d lefm problems. *Engineering Fracture Mechanics*, 55(2):321 – 334, 1996. (Cited on page 4.)

[12] A. Maligno, S. Rajaratnam, S. Leen, and E. Williams. A three-dimensional (3d) numerical study of fatigue crack growth using remeshing techniques. *Engineering Fracture Mechanics*, 77(1):94 – 111, 2010. (Cited on page 4.)

[13] Y. Lei. Finite element crack closure analysis of a compact tension specimen. *International Journal of Fatigue*, 30(1):21 – 31, 2008. (Cited on page 4.)

[14] T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering*, 45(5):601–620, 1999. (Cited on page 4.)

[15] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999. (Cited on page 4.)

[16] T.-P. Fries and T. Belytschko. The extended/generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, 84(3):253–304, 2010. (Cited on page 4.)

[17] T. Belytschko, W. Liu, B. Moran, and K. Elkhodary. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, Ltd, 2014. (Cited on page 4.)

[18] L. Dong and S. N. Atluri. Fracture & Fatigue Analyses: SGBEM-FEM or XFEM? Part 1: 2D Structures. *Comput. Model. Eng. Sci.*, 90:91–146, 2013. (Cited on page 4.)

[19] X. Ren and X. Guan. Three dimensional crack propagation through mesh-based explicit representation for arbitrarily shaped cracks using the extended finite element method. *Engineering Fracture Mechanics*, 177:218 – 238, 2017. (Cited on page 4.)

[20] N. Sukumar, D. Chopp, and B. Moran. Extended finite element method and fast marching method for three-dimensional fatigue crack propagation. *Engineering Fracture Mechanics*, 70(1):29 – 48, 2003. (Cited on page 4.)

[21] A. Bergara, J. Dorado, A. Martin-Meizoso, and J. Martínez-Esnaola. Fatigue crack propagation in complex stress fields: Experiments and numerical simulations using the extended finite element method (xfem). *International Journal of Fatigue*, 103:112 – 121, 2017. (Cited on page 4.)

[22] N. Sukumar, D. L. Chopp, E. Béchet, and N. Moës. Three-dimensional non-planar crack growth by a coupled extended finite element and fast marching method. *International Journal for Numerical Methods in Engineering*, 76(5):727–748, 2008. (Cited on page 4.)

[23] J. Oliver, A. Huespe, and P. Sánchez. A comparative study on finite elements for capturing strong discontinuities: E-fem vs x-fem. *Computer Methods in Applied Mechanics and Engineering*, 195(37):4732 – 4752, 2006. (Cited on page 5.)

[24] M. Costabel. Symmetric Methods for the Coupling of Finite Elements and Boundary Elements (Invited contribution). In C. A. Brebbia, W. L. Wendland, and G. Kuhn, editors, *Mathematical and Computational Aspects*, pages 411–420, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg. (Cited on pages 6 and 100.)

[25] S. Ganguly, J. B. Layton, C. Balakrishna, and J. H. Kane. A fully symmetric multi-zone Galerkin boundary element method. *International Journal for Numerical Methods in Engineering*, 44(7):991–1009, 1999. (Cited on pages 6 and 65.)

[26] R. Springhetti, G. Novati, and M. Margonari. Weak coupling of the Symmetric Galerkin BEM with FEM for potential and elastostatic problems. *Computer Modeling in Engineering and Sciences*, 2006. (Cited on page 6.)

[27] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187 – 207, 1985. (Cited on pages 6 and 28.)

[28] T. Takahashi, N. Nishimura, and S. Kobayashi. A fast BIEM for three-dimensional elastodynamics in time domain. *Engineering Analysis with Boundary Elements*, 27(5):491 – 506, 2003. (Cited on pages 7 and 29.)

[29] S. Chaillat, M. Bonnet, and J.-F. Semblat. A multi-level fast multipole BEM for 3-D elastodynamics in the frequency domain. *Computer Methods in Applied Mechanics and Engineering*, 197(49):4233 – 4249, 2008. (Cited on pages 7 and 29.)

[30] M. Bapat, L. Shen, and Y. Liu. Adaptive fast multipole boundary element method for three-dimensional half-space acoustic wave problems. *Engineering Analysis with Boundary Elements*, 33(8):1113 – 1123, 2009. (Cited on pages 7 and 28.)

[31] A. Frangi and A. d. Gioia. Multipole BEM for the evaluation of damping forces on MEMS. *Computational Mechanics*, 37(1):24–31, Dec 2005. (Cited on pages 7 and 28.)

[32] N. Nishimura and Y. J. Liu. Thermal analysis of carbon-nanotube composites using a rigid-line inclusion model by the boundary integral equation method. *Computational Mechanics*, 35(1):1–10, Dec 2004. (Cited on pages 7 and 29.)

[33] Y. J. Liu, N. Nishimura, Y. Otani, T. Takahashi, X. L. Chen, and H. Munakata. A Fast Boundary Element Method for the Analysis of Fiber-Reinforced Composites Based on a Rigid-Inclusion Model. *Journal of Applied Mechanics*, 72(1):115–128, February 2005. (Cited on pages 7 and 29.)

[34] E. Ooi and Z. Yang. Modelling multiple cohesive crack propagation using a finite element–scaled boundary finite element coupled method. *Engineering Analysis with Boundary Elements*, 33(7):915 – 929, 2009. (Cited on page 7.)

[35] J. P. Wolf and C. Song. Finite-element modelling of unbounded media. 1996. (Cited on page 7.)

[36] F. Romlay, H. Ouyang, A. Ariffin, and N. Mohamed. Modeling of fatigue crack propagation using dual boundary element method and gaussian monte carlo method. *Engineering Analysis with Boundary Elements*, 34(3):297 – 305, 2010. (Cited on page 7.)

[37] S. G. F. Cordeiro and E. D. Leonel. Cohesive crack propagation modelling in wood structures using bem and the tangent operator technique. *Engineering Analysis with Boundary Elements*, 64:111 – 121, 2016. (Cited on page 7.)

[38] B. Nguyen, H. Tran, C. Anitescu, X. Zhuang, and T. Rabczuk. An isogeometric symmetric galerkin boundary element method for two-dimensional crack problems. *Computer Methods in Applied Mechanics and Engineering*, 306:252 – 275, 2016. (Cited on page 7.)

[39] B. Nguyen, X. Zhuang, P. Wriggers, T. Rabczuk, M. Mear, and H. Tran. Isogeometric symmetric galerkin boundary element method for three-dimensional elasticity problems. *Computer Methods in Applied Mechanics and Engineering*, 323:132 – 150, 2017. (Cited on page 7.)

[40] F. Sun, C. Dong, and H. Yang. Isogeometric boundary element method for crack propagation based on bézier extraction of nurbs. *Engineering Analysis with Boundary Elements*, 99:76 – 88, 2019. (Cited on page 7.)

[41] A. Frangi. Fracture propagation in 3D by the symmetric Galerkin boundary element method. *International Journal of Fracture*, 116(4):313–330, August 2002. (Cited on pages 7, 75, 79 and 81.)

[42] D. J. Roberts, A.-V. Phan, H. V. Tippur, L. J. Gray, and T. Kaplan. SGBEM modeling of fatigue crack growth in particulate composites. *Archive of Applied Mechanics*, 80(3):307–322, March 2010. (Cited on page 7.)

[43] R. Kitey, A.-V. Phan, H. V. Tippur, and T. Kaplan. Modeling of crack growth through particulate clusters in brittle matrix by symmetric-Galerkin boundary element method. *International Journal of Fracture*, 141(1):11–25, September 2006. (Cited on page 7.)

[44] K. Xu, S. T. Lie, and Z. Cen. Crack propagation analysis with Galerkin boundary element method. *International Journal for Numerical and Analytical Methods in Geomechanics*, 28(5):421–435, 2004. (Cited on page 7.)

[45] L. Távara, V. Mantič, A. Salvadori, L. J. Gray, and F. París. Sgbem for cohesive cracks in homogeneous media. *Key Engineering Materials*, 454:1–10, 1 2011. (Cited on page 7.)

[46] P. B. Wang and Z. H. Yao. Fast multipole dbem analysis of fatigue crack growth. *Computational Mechanics*, 38(3):223–233, Aug 2006. (Cited on page 7.)

[47] Y. Liu, Y. Li, and W. Xie. Modeling of multiple crack propagation in 2-d elastic solids by the fast multipole boundary element method. *Engineering Fracture Mechanics*, 172:1 – 16, 2017. (Cited on page 7.)

[48] Q. T. Trinh. *Modelling multizone and multicrack in three-dimensional elastostatic media: a Fast multipole Galerkin Boundary Element Method*. PhD Thesis, INSA de Strasbourg, 2014. (Cited on pages xxv, 7, 8, 42, 43, 45, 53, 55, 59, 113, 114 and 119.)

[49] Q. T. Trinh, S. Mouhoubi, C. Chazallon, and M. Bonnet. Solving multizone and multicrack elastostatic problems: A fast multipole symmetric Galerkin boundary element method approach. *Eng. Anal. Bound. Elem.*, 50:486 – 495, 2015. (Cited on pages 7 and 29.)

[50] A. D. Pham. *Méthode multipôle rapide pour les équations intégrales variationnelles en élasticité tridimensionnelle et en mécanique de la rupture*. PhD thesis, INSA de Strasbourg, 2010. (Cited on page 8.)

[51] A. D. Pham, S. Mouhoubi, C. Chazallon, and M. Bonnet. Fast multipole method applied to Symmetric Galerkin boundary element method for 3D elasticity and fracture problems. *Eng. Anal. Bound. Elem.*, 36:1838 – 1847, 2012. (Cited on pages 8, 29 and 32.)

[52] H. Andrä. Integration of singular integrals for the Galerkin-type boundary element method in 3D elasticity. *Computer methods in applied mechanics and engineering*, 157(3-4):239–249, 1998. (Cited on pages 8 and 40.)

[53] A. Frangi, G. Novati, R. Springhetti, and M. Rovizzi. 3D fracture analysis by the symmetric Galerkin BEM. *Computational Mechanics*, 28(3-4):220–232, April 2002. (Cited on pages 8 and 82.)

[54] M. Rezayat, D. J. Shippy, and F. J. Rizzo. On time-harmonic elastic-wave analysis by the boundary element method for moderate to high frequencies. *Computer Methods in Applied Mechanics and Engineering*, 55(3):349 – 367, 1986. (Cited on pages 8 and 41.)

[55] S. Chaillat. *Fast Multipole Method for 3-D elastodynamic boundary integral equations. Application to seismic wave propagation.* PhD Thesis, Ecole des Ponts ParisTech, December 2008. (Cited on page 8.)

[56] L. J. Gray and G. H. Paulino. Symmetric Galerkin boundary integral formulation for interface and multi-zone problems. *International Journal for Numerical Methods in Engineering*, 40(16):3085–3101, 1997. (Cited on pages 8 and 65.)

[57] M. Bonnet. *Boundary integral equation methods for solids and fluids.* John Wiley and sons, 1999. (Cited on pages 17, 20, 26 and 100.)

[58] S. Sirtori, G. Maier, G. Novati, and S. Miccoli. A galerkin symmetric boundary-element method in elasticity: Formulation and implementation. *International Journal for Numerical Methods in Engineering*, 35(2):255–282, 1992. (Cited on pages 17 and 18.)

[59] M. Bonnet. Regularized direct and indirect symmetric variational BIE formulations for three-dimensional elasticity. *Engineering Analysis with Boundary Elements*, 15(1):93 – 102, 1995. (Cited on pages 18 and 19.)

[60] M. Bonnet, G. Maier, and C. Polizzotto. Symmetric Galerkin Boundary Element Methods. *Applied Mechanics Reviews*, 51(11):669–704, November 1998. (Cited on page 18.)

[61] E. Becache, J.-C. Nedelec, and N. Nishimura. Regularization in 3D for anisotropic elastodynamic crack and obstacle problems. *Journal of Elasticity*, 31(1):25–46, Apr 1993. (Cited on page 19.)

[62] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Scientific Computing*, 14:461–469, 1993. (Cited on page 27.)

[63] C. Y. Leung and S. P. Walker. Iterative solution of large three-dimensional BEM elastostatic analyses using the GMRES technique. *International Journal for Numerical Methods in Engineering*, 40(12):2227–2236, 1997. (Cited on page 27.)

[64] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325 – 348, 1987. (Cited on page 28.)

[65] J. Gómez and H. Powert. A multipole direct and indirect BEM for 2D cavity flow at low Reynolds number. *Engineering Analysis with Boundary Elements*, 19(1):17 – 31, 1997. (Cited on pages 28 and 29.)

[66] Y. Yasuda and T. Sakuma. A technique for plane-symmetric sound field analysis in the fast multipole boundary element method. *Journal of Computational Acoustics*, 13(01):71–85, 2005. (Cited on page 28.)

[67] W. Chew, H. Chao, T. Cui, C. Lu, S. Ohnuki, Y. Pan, J. Song, S. Velamparambil, and J. Zhao. Fast integral equation solvers in computational electromagnetics of complex structures. *Engineering Analysis with Boundary Elements*, 27(8):803 – 823, 2003. (Cited on pages 28 and 29.)

[68] Y. H. Chen, W. C. Chew, and S. Zeroug. Fast multipole method as an efficient solver for 2D elastic wave surface integral equations. *Computational Mechanics*, 20(6):495–506, Nov 1997. (Cited on page 29.)

[69] A. P. Peirce and J. A. L. Napier. A spectral multipole method for efficient solution of large-scale boundary element models in elastostatics. *International Journal for Numerical Methods in Engineering*, 38(23):4009–4034, 1995. (Cited on page 29.)

[70] Y. Fu, K. J. Klimkowski, G. J. Rodin, E. Berger, J. C. Browne, J. K. Singer, R. A. Van De Geijn, and K. S. Vemaganti. A fast solution method for three-dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering*, 42(7):1215–1229, 1998. (Cited on page 29.)

[71] N. Nishimura, K. ichi Yoshida, and S. Kobayashi. A fast multipole boundary integral equation method for crack problems in 3D. *Engineering Analysis with Boundary Elements*, 23(1):97 – 105, 1999. (Cited on page 29.)

[72] A. A. Mammoli and M. S. Ingber. Stokes flow around cylinders in a bounded two-dimensional domain using multipole-accelerated boundary element methods. *International Journal for Numerical Methods in Engineering*, 44(7):897–917, 1999. (Cited on page 29.)

[73] S. Mouhoubi. *Symmetric FEM-BEM coupling in solid mechanics: formulation and implementation in a finite element code*. PhD Thesis, Université de Limoges, December 2000. (Cited on pages 29 and 100.)

[74] M. Margonari. *Boundary element techniques for three dimensional problems in Elastostatics*. PhD Thesis, University of Trento, 2004. (Cited on pages 29, 65 and 100.)

[75] K. Yoshida. *Applications of fast multipole method to boundary integral equation method*. PhD Thesis, Kyoto University, 2001. (Cited on page 32.)

[76] D. Rose and R. A. Willoughby, editors. *Sparse Matrices and Their Applications*. 1972. (Cited on pages 44 and 57.)

[77] A. Brameller, R. N. Allan, and Y. M. Hamam. *Sparsity : its practical application to systems analysis*. London ; New York : Pitman, 1976. (Cited on pages 44 and 57.)

[78] F. Amlani, S. Chaillat, and A. Loseille. An efficient preconditioner for adaptive fast multipole accelerated boundary element methods to model time-harmonic 3D wave propagation. *Computer Methods in Applied Mechanics and Engineering*, 352:189 – 210, 2019. (Cited on page 45.)

[79] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, and R. Menon. *Parallel Programming in OpenMP*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001. (Cited on page 52.)

[80] L. R. Scott, T. Clark, and B. Bagheri. *Scientific Parallel Computing*. Princeton University Press, Princeton, NJ, USA, 2005. (Cited on page 52.)

[81] F. Magoules, F.-X. Roux, and G. Houzeaux. *Parallel Scientific Computing*. Computer Engineering Series. Wiley-ISTE, December 2015. (Cited on page 52.)

[82] R. Trobec, B. Slivnik, P. Bulić, and B. Robič. *Programming Multi-core and Shared Memory Multiprocessors Using OpenMP*, pages 47–86. Springer International Publishing, Cham, 2018. (Cited on page 52.)

[83] L. Greengard and W. D. Gropp. A parallel version of the fast multipole method. *Computers & Mathematics with Applications*, 20(7):63 – 71, 1990. (Cited on pages 52 and 53.)

[84] J. Gu and A. M. Zsaki. Accelerated parallel computation of field quantities for the boundary element method applied to stress analysis using multi-core CPUs, GPUs and FPGAs. *Cogent Engineering*, 5(1):1–21, 2018. (Cited on page 53.)

[85] J. Ptaszny. Parallel fast multipole boundary element method applied to computational homogenization. *AIP Conference Proceedings*, 1922(1):140003, 2018. (Cited on page 53.)

[86] G. M. Amdahl. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *AFIPS Conference Proceedings*, pages 483–485, Atlantic City, New Jersey, 1967. AFIPS Press. (Cited on page 55.)

[87] J. L. Gustafson, G. R. Montry, and R. E. Benner. Development of Parallel Methods for a 1024-Processor Hypercube. *SIAM JOURNAL ON SCIENTIFIC AND STATISTICAL COMPUTING*, 9(4):609–638, 1988. (Cited on page 56.)

[88] Y. Saad. Sparskit, a basic tool kit for sparse matrix computations. Technical report, Center for Supercomputing Research and Development, 1990. (Cited on page 57.)

[89] A.-V. Phan, J. A. L. Napier, L. J. Gray, and T. Kaplan. Stress intensity factor analysis of friction sliding at discontinuity interfaces and junctions. *Computational Mechanics*, 32(4):392–400, Dec 2003. (Cited on page 64.)

[90] C.-H. Kuo. *Boundary Element Methods for Contact Analysis*, pages 255–258. Springer US, Boston, MA, 2013. (Cited on page 64.)

[91] T. Chen, B. Wang, Z. Cen, and Z. Wu. A symmetric galerkin multi-zone boundary element method for cohesive crack growth. *Engineering Fracture Mechanics*, 63(5):591 – 609, 1999. (Cited on page 65.)

[92] A. Sutradhar, G. H. Paulino, and L. J. Gray. *Symmetric Galerkin Boundary Element Method*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. (Cited on pages 70 and 80.)

[93] F. de Paula and J. Telles. A comparison between point collocation and galerkin for stiffness matrices obtained by boundary elements. *Engineering Analysis with Boundary Elements*, 6(3):123 – 128, 1989. (Cited on page 70.)

[94] S. Beden, S. Abdullah, and A. Mohd Ihsan. Review of fatigue crack propagation models for metallic components. *European Journal of Scientific Research*, 28(3):364–397, 2009. (Cited on page 73.)

[95] P. Paris and F. Erdogan. A critical analysis of crack propagation laws. *J. Basic Eng.*, 85:528–533, 1963. (Cited on page 73.)

[96] K. Walker. The effect of stress ratio during crack propagation and fatigue for 2024-T3 and 7075-T6 aluminum. In *Effects of environment and complex load history on fatigue life*. ASTM International, 1970. (Cited on page 74.)

[97] R. Forman. Study of fatigue crack initiation from flaws using fracture mechanics theory. *Engineering Fracture Mechanics*, 4(2):333 – 345, 1972. (Cited on page 74.)

[98] A. Hartman and J. Schijve. The effects of environment and load frequency on the crack propagation law for macro fatigue crack growth in aluminium alloys. *Engineering Fracture Mechanics*, 1(4):615 – 631, 1970. (Cited on page 74.)

[99] J. Newman. A crack-closure model for predicting fatigue crack growth under aircraft spectrum loading. In *Methods and models for predicting fatigue crack growth under random loading*. ASTM International, 1981. (Cited on page 74.)

[100] J. C. Newman. A crack opening stress equation for fatigue crack growth. *International Journal of Fracture*, 24(4):R131–R135, Apr 1984. (Cited on page 74.)

[101] E. Castillo, A. Fernández-Canteli, and D. Siegele. Obtaining s–n curves from crack growth curves: an alternative to self-similarity. *International Journal of Fracture*, 187(1):159–172, May 2014. (Cited on page 74.)

[102] S. Blasón, J. Correia, N. Apetre, A. Arcari, A. D. Jesus, P. Moreira, and A. Fernández-Canteli. Proposal of a fatigue crack propagation model taking

into account crack closure effects using a modified CCS crack growth model. *Procedia Structural Integrity*, 1:110 – 117, 2016. (Cited on page 74.)

[103] Y. Mi and M. Aliabadi. Three-dimensional crack growth simulation using bem. *Computers & Structures*, 52(5):871 – 878, 1994. (Cited on page 75.)

[104] P. Bouchard, F. Bay, and Y. Chastel. Numerical modelling of crack propagation: automatic remeshing and comparison of different criteria. *Computer Methods in Applied Mechanics and Engineering*, 192(35):3887 – 3908, 2003. (Cited on page 75.)

[105] S. Ma. *Propagation of the mixed mode crack in a welded specimen in elastic-plastic material*. Theses, Université Blaise Pascal - Clermont-Ferrand II, January 2005. (Cited on page 75.)

[106] F. Erdogan and G. C. Sih. On the crack extension in plates under plane loading and transverse shear. *J. Basic Eng.*, 85:519–525, 1963. (Cited on page 75.)

[107] M. Williams. On the stress distribution at the base of a stationary crack. *Journal of Applied Mechanics*, 24:109–114, 1957. (Cited on page 78.)

[108] I. Raju and J. Newman. Stress-intensity factors for a wide range of semi-elliptical surface cracks in finite-thickness plates. *Engineering Fracture Mechanics*, 11(4):817 – 829, 1979. (Cited on page 79.)

[109] D.-Z. Feng and Q.-C. Hong. Investigation of surface crack opening displacement and its application in pressure vessels and piping. *International Journal of Pressure Vessels and Piping*, 52(2):227 – 239, 1992. (Cited on pages 79 and 81.)

[110] M. Ramezani, J. Purbolaksono, A. Andriyana, S. Ramesh, and N. Mardi. Analysis of surface cracks in round bars using dual boundary element method. *Engineering Analysis with Boundary Elements*, 93:112 – 123, 2018. (Cited on page 79.)

[111] C. Sun and Z.-H. Jin. *Fracture Mechanics*. Academic Press, 2012. (Cited on page 82.)

[112] T. Anderson. *Fracture Mechanics: Fundamentals and Applications*. Taylor and Francis Group, 2005. (Cited on page 82.)

[113] O. C. Zienkiewicz and R. L. Taylor. *The finite element method, ; volume 1: basic formulation and linear problems*. 1994. (Cited on page 94.)

[114] J.-L. Batoz and G. Dhatt. *Modélisation des structures par éléments finis: Solides élastiques*. Presses Université Laval, 1990. (Cited on page 94.)

[115] O. C. Zienkiewicz, D. W. Kelly, and P. Bettess. The coupling of the finite element method and boundary solution procedures. *International Journal for Numerical Methods in Engineering*, 11(2):355–375, 1977. (Cited on pages 99 and 100.)

[116] C. Brebbia and P. Georgiou. Combination of boundary and finite elements in elastostatics. *Applied Mathematical Modelling*, 3(3):212 – 220, 1979. (Cited on pages 99 and 100.)

[117] D. Kelly, G. Mustoe, and O. Zienkiewicz. *Coupling Boundary Element Methods with other Numerical Techniques, P. K. Banerjee, R. Butterfield (eds.)*. Applied Science Publishers, 1979. (Cited on page 99.)

[118] K. Kohno, T. Tsunada, H. Seto, and M. Tanaka. Hybrid stress analysis of boundary and finite elements by a super-element method. *Finite Elements in Analysis and Design*, 7(4):279 – 290, 1991. (Cited on page 99.)

[119] G. Maier. On elastoplastic analysis by boundary elements. *Mechanics Research Communications*, 10(1):45 – 52, 1983. (Cited on page 99.)

[120] A. Kanarachos and C. Provatidis. On the symmetrization of the bem formulation. *Computer Methods in Applied Mechanics and Engineering*, 71(2):151 – 165, 1988. (Cited on page 99.)

[121] P. K. Banerjee and R. Butterfield. *Boundary element methods in engineering science*, volume 17. McGraw-Hill London, 1981. (Cited on page 99.)

[122] T. Belytschko, H. Chang, and Y. Lu. A variationally coupled finite element-boundary element method. *Computers & Structures*, 33(1):17–20, 1989. (Cited on page 99.)

[123] L. Hong-Bao, H. Guo-Ming, H. A. Mang, and P. Torzicky. A new method for the coupling of finite element and boundary element discretized subdomains of elastic bodies. *Computer Methods in Applied Mechanics and Engineering*, 54(2):161–185, 1986. (Cited on page 99.)

[124] O. Tullberg and L. Bolteus. A critical study of different boundary element stiffness matrices. *Boundary element methods in engineering*, pages 621–635, 1982. (Cited on page 99.)

[125] S. Meddahi, F.-J. Sayas, and V. Seglas. Nonsymmetric coupling of BEM and mixed FEM on polyhedral interfaces. *Mathematics of Computation*, 80(273):43–68, 2011. (Cited on page 99.)

[126] O. Steinbach. On the stability of the non-symmetric BEM/FEM coupling in linear elasticity. *Computational Mechanics*, 51(4):421–430, Apr 2013. (Cited on page 100.)

[127] M. Feischl, T. Führer, M. Karkulik, and D. Praetorius. Stability of symmetric and nonsymmetric FEM–BEM couplings for nonlinear elasticity problems. *Numerische Mathematik*, 130(2):199–223, Jun 2015. (Cited on page 100.)

[128] M. Costabel and E. P. Stephan. Coupling of finite and boundary element methods for an elastoplastic interface problem. *SIAM Journal on Numerical Analysis*, 27(5):1212–1226, 1990. (Cited on page 100.)

[129] J. Rungamornrat and M. E. Mear. SGBEM–FEM coupling for analysis of cracks in 3D anisotropic media. *International Journal for Numerical Methods in Engineering*, 86(2):224–248, 2011. (Cited on page 100.)

[130] T. Nguyen, J. Rungamornrat, T. Senjuntichai, and A. Wijeyewickrema. Fem-sgbem coupling for modeling of mode-i planar cracks in three-dimensional elastic media with residual surface tension effects. *Engineering Analysis with Boundary Elements*, 55:40 – 51, 2015. (Cited on page 100.)

[131] G. Beer and J. L. Meek. The coupling of boundary and finite element methods for infinite domain problems in elasto- plasticity. In C. A. Brebbia, editor, *Boundary Element Methods*, pages 575–591, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg. (Cited on page 100.)

[132] G. Beer. Finite element, boundary element and coupled analysis of unbounded problems in elastostatics. *International Journal for Numerical Methods in Engineering*, 19(4):567–580, 1983. (Cited on page 100.)

[133] C. Balakrishna, L. Gray, and J. Kane. Efficient analytical integration of symmetric Galerkin boundary integrals over curved elements: Thermal conduction formulation. *Computer Methods in Applied Mechanics and Engineering*, 111(3):335 – 355, 1994. (Cited on page 100.)

[134] T. Belytschko and Y. Y. Lu. A variationally coupled FE–BE method for transient problems. *International Journal for Numerical Methods in Engineering*, 37(1):91–105, 1994. (Cited on page 100.)

[135] M. Haas, B. Helldörfer, and G. Kuhn. Improved coupling of finite shell elements and 3D boundary elements. *Archive of Applied Mechanics*, 75(10):649, Jun 2006. (Cited on page 100.)

[136] O. von Estorff and C. Hagen. Iterative coupling of FEM and BEM in 3D transient elastodynamics. *Engineering Analysis with Boundary Elements*, 29(8):775 – 787, 2005. (Cited on page 100.)

[137] P. Coulier, S. François, G. Lombaert, and G. Degrande. Coupled finite element – hierarchical boundary element methods for dynamic soil–structure interaction in the frequency domain. *International Journal for Numerical Methods in Engineering*, 97(7):505–530, 2014. (Cited on page 100.)

[138] A. Aimi, M. Diligenti, A. Frangi, and C. Guardasoni. Energetic BEM–FEM coupling for wave propagation in 3D multidomains. *International Journal for Numerical Methods in Engineering*, 97(5):377–394, 2014. (Cited on page 100.)

[139] W. Elleithy and L. T. Leong. Some recent developments in coupling of finite element and boundary element methods - Part I: An overview. In *Advances in Civil and Industrial Engineering IV*, volume 580 of *Applied Mechanics and Materials*, pages 2936–2942. Trans Tech Publications Ltd, 8 2014. (Cited on page 100.)

[140] J. Gwinner and E. P. Stephan. *FEM-BEM Coupling*, pages 451–536. Springer International Publishing, Cham, 2018. (Cited on page 100.)

[141] J. Darling and J. Woolstencroft. A study of fiber glass pavement reinforcement used in different climatic zones and their effectiveness in retarding reflective cracking in asphalt overlays. In *Cracking in Pavements: Mitigation, Risk Assessment and Prevention, Proceedings the 5th International RILEM Conference, Limoges, France*, pages 5–8, 2004. (Cited on page 110.)

[142] V. J. Marks. Glasgrid fabric to control reflective cracking. Technical report, Iowa Department of Transportation, 1990. (Cited on page 110.)

[143] S. Brown, N. Thom, and P. Sanders. A study of grid reinforced asphalt to combat reflection cracking. *Journal of the Association of Asphalt Paving Technologists*, 70:543–571, 2001. (Cited on page 110.)

[144] R. Hufenus, R. Rüegger, D. Flum, and I. J. Sterba. Strength reduction factors due to installation damage of reinforcing geosynthetics. *Geotextiles and Geomembranes*, 23(5):401 – 424, 2005. (Cited on page 110.)

[145] A. J. Bush, E. W. Brooks, et al. Geosynthetic materials in reflective crack prevention. Technical report, Oregon. Dept. of Transportation. Research Unit, 2007. (Cited on page 110.)

[146] M. Bacchi. Evaluation of the effects of geo-composite reinforcement on fatigue life of asphalt pavements. In *Proceedings of the 2nd Workshop on four-point bending. Guimarães: University of Minho*, 2009. (Cited on page 110.)

[147] I. M. Arsenie. *Etude et modélisation des renforcements de chaussées à l'aide de grilles en fibre de verre sous sollicitations de fatigue*. PhD thesis, INSA de Strasbourg, 2013. (Cited on page 110.)

[148] M.-L. Nguyen, J. Blanc, J.-P. Kerzrého, and P. Hornych. Review of glass fibre grid use for pavement reinforcement and apt experiments at ifsttar. *Road Materials and Pavement Design*, 14(sup1):287–308, 2013. (Cited on pages 110 and 118.)

[149] L. Sagnol. *Experimental and analytical study of the reinforcement of pavements by glass fibre grids*. PhD thesis, INSA de Strasbourg, 2017. (Cited on page 110.)

[150] I. M. Arsenie, C. Chazallon, J.-L. Duchez, and P. Hornych. Laboratory characterisation of the fatigue behaviour of a glass fibre grid-reinforced asphalt concrete using 4pb tests. *Road Materials and Pavement Design*, 18(1):168–180, 2017. (Cited on page 110.)

[151] F. Gu, X. Luo, R. Luo, E. Y. Hajj, and R. L. Lytton. A mechanistic-empirical approach to quantify the influence of geogrid on the performance of flexible pavement structures. *Transportation Geotechnics*, 13:69 – 80, 2017. (Cited on pages 110 and 111.)

[152] L. Sagnol, J. C. Quezada, C. Chazallon, and M. Stockner. Effect of glass fibre grids on the bonding strength between two asphalt layers and its contact dynamics method modelling. *Road Materials and Pavement Design*, 20(5):1164–1181, 2019. (Cited on page 110.)

[153] J.-H. Lee, S.-B. Baek, K.-H. Lee, J.-S. Kim, and J.-H. Jeong. Long-term performance of fiber-grid-reinforced asphalt overlay pavements: A case study of korean national highways. *Journal of Traffic and Transportation Engineering*, 6(4):366 – 382, 2019. (Cited on page 110.)

[154] E. Godard, C. Chazallon, P. Hornych, A. Chabot, M. L. Nguyen, D. Doligez, and H. Pelletier. Projet ANR SolDuGri : Pour une solution durable du renforcement des infrastructures par grilles en fibre de verres. *Revue Générale des Routes et de l'Aménagement*, (949):pp.24–30, January 2017. (Cited on page 110.)

[155] C. Chazallon, E. Godard, D. Doligez, M. Gharbi, P. Hornych, A. Chabot, M. L. Nguyen, and H. Pelletier. Pour une solution durable du renforcement des infrastructures par grilles en fibre de verre. In *12ème Rencontres Géosynthétiques*, page 10p, NANCY, France, March 2019. (Cited on page 110.)

[156] D. Bodin, G. Pijaudier-Cabot, C. de La Roche, J.-M. Piau, and A. Chabot. Continuum damage approach to asphalt concrete fatigue modeling. *Journal of Engineering Mechanics*, 130(6):700–708, 2004. (Cited on page 110.)

[157] M. Castro and J. A. Sánchez. Estimation of asphalt concrete fatigue curves – a damage theory approach. *Construction and Building Materials*, 22(6):1232 – 1238, 2008. (Cited on page 110.)

[158] A. Vanelstraete, D. Leonard, and J. Veys. Structural design of roads with steel reinforcing nettings. In *Proceedings of the 4th Int. RILEM conf. Cracking in Pavements*, pages 57–67, 2000. (Cited on page 110.)

[159] N. Thom. A simplified computer model for grid reinforced asphalt overlays. In *Proceedings of the 4th International RILEM Conference on Reflective Cracking in Pavements*, pages 37–46, 2000. (Cited on page 111.)

[160] A. d. Bondt, J. Schrader, W. v. Bijsterveld, and D. Long. Scientific background of arcdeso-version april 2005. Technical report, Internal Report-Ooms Nederland Holding, 2005. (Cited on page 111.)

[161] J. Kwon. *Development of a mechanistic model for geogrid reinforced flexible pavements*. PhD thesis, University of Illinois at Urbana-Champaign, 2007. (Cited on page 111.)

[162] S. Perkins, B. Christopher, E. Cuelho, G. Eiksund, C. Schwartz, and G. Svanø. A mechanistic–empirical model for base-reinforced flexible pavements. *International Journal of Pavement Engineering*, 10(2):101–114, 2009. (Cited on page 111.)

[163] M. D. Nazzal, M. Y. Abu-Farsakh, and L. N. Mohammad. Implementation of a critical state two-surface model to evaluate the response of geosynthetic reinforced pavements. *International Journal of Geomechanics*, 10(5):202–212, 2010. (Cited on page 111.)

[164] M. Y. Abu-Farsakh, J. Gu, G. Z. Voyiadjis, and Q. Chen. Mechanistic–empirical analysis of the results of finite element analysis on flexible pavement with geogrid base reinforcement. *International Journal of Pavement Engineering*, 15(9):786–798, 2014. (Cited on page 111.)

[165] X. Tang, S. M. Stoffels, and A. M. Palomino. Mechanistic-empirical approach to characterizing permanent deformation of reinforced soft soil subgrade. *Geotextiles and Geomembranes*, 44(3):429 – 441, 2016. (Cited on page 111.)

[166] S. Cafiso and A. Di Graziano. Evaluation of flexible reinforced pavement performance by ndt. In *TRB, 82nd Annual Meeting, Washington*, 2003. (Cited on page 111.)

[167] G. Tesoriere and D. Ticali. Verifica sperimentale sul comportamento delle pavimentazioni stradali di tipo flessibile rinforzate con rete metallica all'azione tangenziale dei carichi. 2004. (Cited on page 111.)

[168] A. Montepara, G. Tebaldi, and A. Costa. Performance evaluation of a surface pavement steel reinforcement. In *Proceedings of the 5th International*, 2005. (Cited on page 111.)

[169] M. Coni and P. Bianco. Steel reinforcement influence on the dynamic behaviour of bituminous pavement. In *PRO 11: 4th International RILEM Conference on Reflective Cracking in Pavement Research in Practice*, volume 11, page 6. RILEM Publications, 2000. (Cited on page 111.)

[170] J. Baek and I. Al-Qadi. Reflective cracking: Modeling fracture behavior of hot-mix asphalt overlays with interlayer systems. *Asphalt Paving Technology: Association of Asphalt Paving Technologists-Proceedings of the Technical Sessions*, 78:789–827, 12 2009. (Cited on page 111.)

[171] G. Krishnasamy, F. J. Rizzo, and Y. Liu. Boundary integral equations for thin bodies. *International Journal for Numerical Methods in Engineering*, 37(1):107–121, 1994. (Cited on page 112.)

[172] R. Martinez. The thin-shape breakdown (TSB) of the Helmholtz integral equation. *Acoustical Society of America Journal*, 90:2728–2738, November 1991. (Cited on page 112.)

[173] J. Granados and R. Gallego. Regularization of nearly hypersingular integrals in the boundary element method. *Engineering Analysis with Boundary Elements*, 25(3):165 – 184, 2001. (Cited on page 112.)

[174] Y. Liu. On the simple-solution method and non-singular nature of the BIE/BEM — a review and some new results. *Engineering Analysis with Boundary Elements*, 24(10):789 – 795, 2000. (Cited on page 112.)

[175] S. Mukherjee, M. K. Chati, and X. Shi. Evaluation of nearly singular integrals in boundary element contour and node methods for three-dimensional linear elasticity. *International Journal of Solids and Structures*, 37(51):7633 – 7654, 2000. (Cited on page 112.)

[176] Y. Zhang and H. Sun. Theoretic analysis on virtual boundary element. *Chinese Journal of Computational Mechanics*, 17(1):56–62, 2000. (Cited on page 112.)

[177] L. Jun, G. Beer, and J. Meek. Efficient evaluation of integrals of order 1/r 1/r2, 1/r3 using Gauss quadrature. *Engineering Analysis*, 2(3):118 – 123, 1985. (Cited on page 112.)

[178] E. Lutz. Exact Gaussian quadrature methods for near-singular integrals in the boundary element method. *Engineering Analysis with Boundary Elements*, 9(3):233 – 245, 1992. (Cited on page 112.)

[179] Z. Niu, X. Wang, and H. Zhou. A general algorithm for calculating the quantities at interior points close to the boundary by the BEM. *Chinese Journal of Theoretical and Applied Mechanics*, 33(2):275–283, 2001. (Cited on page 112.)

[180] Y.-M. Zhang and H.-C. Sun. Analytical treatment of boundary integrals in direct boundary element analysis of plan potential and elasticity problems. *Applied Mathematics and Mechanics*, 22(6):664–673, Jun 2001. (Cited on page 112.)

[181] Q. Huang and T. A. Cruse. Some notes on singular integral techniques in boundary element analysis. *International Journal for Numerical Methods in Engineering*, 36(15):2643–2659, 1993. (Cited on page 112.)

[182] H. Ma and N. Kamiya. Distance transformation for the numerical evaluation of near singular boundary integrals with various kernels in boundary element method. *Engineering Analysis with Boundary Elements*, 26(4):329 – 339, 2002. (Cited on page 112.)

[183] Y.-M. Zhang and Y. Gu. An effective method in BEM for potential problems of thin bodies. *J Mar Sci Technol*, 18(1):137–144, 2010. (Cited on page 112.)

[184] Y. Zhang and C. Sun. A general algorithm for the numerical evaluation of nearly singular boundary integrals in the equivalent non-singular BIEs with indirect unknowns. *Journal of the Chinese Institute of Engineers*, 31(3):437–447, 2008. (Cited on page 112.)

[185] M. Lenoir and N. Salles. Evaluation of 3-d singular and nearly singular integrals in galerkin bem for thin layers. *SIAM Journal on Scientific Computing*, 34(6):A3057–A3078, 2012. (Cited on page 112.)

[186] X. Li and Y. Su. Three-dimensional stress analysis of thin structures using a boundary element method with sinh transformation for nearly singular integrals. *Computers & Mathematics with Applications*, 72(11):2773 – 2787, 2016. (Cited on page 112.)

[187] Z. Han, C. Cheng, Z. Hu, and Z. Niu. The semi-analytical evaluation for nearly singular integrals in isogeometric elasticity boundary element method. *Engineering Analysis with Boundary Elements*, 95:286 – 296, 2018. (Cited on page 112.)

[188] C. Chazallon, G. Koval, P. Hornych, F. Allou, and S. Mouhoubi. Modelling of rutting of two flexible pavements with the shakedown theory and the finite element method. *Computers and Geotechnics*, 36(5):798 – 809, 2009. (Cited on page 113.)

[189] A. Virgili, F. Canestrari, A. Grilli, and F. Santagata. Repeated load test on bituminous systems reinforced by geosynthetics. *Geotextiles and Geomembranes*, 27(3):187 – 195, 2009. (Cited on page 122.)

[190] J. P. Kerzreho, J. P. Michaut, and P. Hornych. Enrobé armé de grille en fibre de verre: Test sur le manège de fatigue de l'ifsttar. *Revue générale des routes et des aérodromes*, (890):48 – 51, 2010. (Cited on page 122.)

[191] S. Chaillat, S. P. Groth, and A. Loseille. Metric-based anisotropic mesh adaptation for 3D acoustic boundary element methods. *Journal of Computational Physics*, 372:473–499, November 2018. (Cited on page 127.)

[192] P. Frey. MEDIT : An interactive Mesh visualization Software. Technical Report RT-0253, INRIA, December 2001. (Cited on page 162.)