Modélisation tridimensionnelle en élastostatique des domaines multizones et multifissurés : une approche par la méthode multipôle rapide en éléments de frontière de Galerkin

Quoc-Tuan Trinh

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE 269

MATHEMATIQUES, SCIENCES de l'INFORMATION et de l'INGENIEUR

Laboratoire ICUBE – INSA de Strasbourg

THÈSE

présentée par :

Quoc-Tuan TRINH

Soutenue le : **18 septembre 2014**

Pour obtenir le grade de :
Docteur de l'université de Strasbourg

Discipline/ Spécialité : Génie Civil

Modélisation tridimensionnelle en élastostatique des domaines multizones et multifissurés : une approche par la méthode multipôle rapide en éléments de frontière de Galerkin

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, CEN Saclay
M. POUYA Ahmad	Professeur, Ecole de Ponts et Chaussées Paris-Tech

---

AUTRES MEMBRES DU JURY :

M. SHAROUR Isam	Professeur, Université de Lille 1
Mme. MOUHOUBI Saïda	Maître de Conférences, INSA de Strasbourg

UNIVERSITÉ DE STRASBOURG

Quoc-Tuan TRINH

Modélisation tridimensionnelle en élastostatique des domaines multizones et multifissurés : une approche par la méthode multipôle rapide en éléments de frontière de Galerkin

Résumé

La modélisation numérique de la multi-fissuration et son influence sur les ouvrages du Génie Civil reste un sujet ouvert et nécessite le développement de nouveaux outils numériques de plus en plus performants. L'approche retenue dans cette thèse est basée sur l'utilisation des concepts des équations intégrales de Galerkin accélérées par la méthode multipôle rapide. Les méthodes intégrales sont bien connues pour leur souplesse à définir des géométries complexes en 3D. Elles restent également très performantes en mécanique de la rupture, lors de la détermination des champs singuliers au voisinage des fissures. La Méthode Multipôle Rapide, quant à elle, permet via une judicieuse reformulation des fonctions fondamentales propres aux formulations intégrales, de réduire considérablement le coût des calculs. La mise en œuvre de la FM-SGBEM a permis de pallier les difficultés rencontrées lors de la phase de résolution et ce lorsqu'on traite de domaines de grandes tailles par équations intégrales de Galerkin pures. Les présents travaux, viennent en partie optimiser et renforcer cette phase dans les environnements numériques existants. D'autre part, des adaptations et des développements théoriques des formulations FM-SGBEM pour prendre en compte le caractère hétérogène des domaines en Génie Civil qui en découlent, ont fait l'objet d'une large partie des travaux développés dans cette thèse. La modélisation du phénomène de propagation de fissures par fatigue a également été étudiée avec succès. Enfin, une application sur une structure de chaussée souple a permis de valider les modèles ainsi développés en propagation de fissures par fatigue dans des structures hétérogènes. De réelles perspectives d'optimisations et de développements de cet outil numérique restent envisagées.

Mots-clés : SGBEM, FMM, GMRES, fissure, multizone, propagation de fissures

Abstract

The numerical modeling of cracks and its influence on the understanding of the behaviors of the civil engineering structures is an open topic since many decades. To take into consideration complex configurations, it is necessary to construct more robust and more efficient algorithms. In this work, the approach Galerkin of the boundary integral equations (**S**ymmetric **G**alerkin **B**oundary **E**lement **M**ethod) coupled with the **F**ast **M**ultipole **M**ethod (FMM) has been adopted. The boundary analyses are well-known for the flexibility to treat sophisticated geometries (unbounded/semi-unbounded) whilst reducing the problem dimension or for the good accuracy when dealing with the singularities. By coupling with the FMM, all the bottle-necks of the traditional BEM due to the fully-populated matrices or due to the slow evaluations of the integrals have been reduced, thus making the FM-SGBEM an attractive alternative for problems in fracture mechanics. In this work, the existing single-region formulations have been extended to multi-region configurations along with several types of solicitations. Many efforts have also been spent to improve the efficiency of the numerical algorithms. Fatigue crack propagations have been implemented and some practical simulations have been considered. The obtained results have validated the numerical program and have also opened many perspectives of further developments for the code.

Key-words: SGBEM, FMM, GMRES, cracks, crack-growth, multizone

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

Modeling multizone and multicrack in three-dimensional elastostatic media: a Fast Multipole Galerkin Boundary Elements Method

Defended by

Quoc-Tuan Trinh

Jury :

Reviewers :	Mr A. Millard	-	Research Director (CEA, CEN Saclay)
	Mr A. Pouya	-	Professor, ENPC
Thesis Directors :	Mr C. Chazallon	-	Professor, INSA of Strasbourg
	Mr M. Bonnet	-	Research Director, ENSTA ParisTech
Examinators :	Mr I. Sharour	-	Professor, University of Lille 1
	Mrs S. Mouhoubi	-	Lecturer, INSA of Strasbourg
Invited :	Mr E. Bourdarot	-	Expert of Dams, EDF-CIH

September 2014

List of Figures

List of Tables

Acknowledgments

This work collects the research i have carried out during my PhD at the Civil Engineering team of the Laboratory ICUBE, National Institute of Applied Sciences (INSA) of Strasbourg, France.

I would like to express my gratitude towards my advisors, assistant professor Saida Mouhoubi and thesis director Cyrille Chazallon, for all the helpful advice, continuous encouragement and support.

Sincere thanks to my thesis co-director, professor Marc Bonnet, senior research scientist at the POEMS team of the Superior National School of Advanced Techniques (ENSTA ParisTech) for his kindness and his precious suggestions.

I also want to thank all the colleagues i have met at the INSA of Strasbourg (Georg Koval, Hossein Nowamooz, Lucile Gondor) for their hospitality and friendliness during my PhD time. Special thank to my fellow PhD students (Danh, Nam, Kai, Peng, XiaoFeng, Andrea, Ioana, Lauba, Saeid) for all the great moments we have spent.

Finally i wish to thank my family and my wife for their endless unconditional encouragements and supports.

Abstract

Background

In Civil engineering, there is a great need for a better understanding of the fracture performance of the structures. Defined as the displacement discontinuities of the material, fractures (or cracks) occur due to many causes. One of which is when the applied solicitations surpass the material rigidity or when the micro defects during fabrication become visibly large under the repeated service loading... The development of cracks reduces notably the bearing capacity of the components and sometimes lead to complete failure of the entire structure. The study of crack(s) and crack propagation is therefore a major concern in civil engineering designs, constructions and maintenances. Since the experimental tests sometimes take longer times and are rather costly because of the equipments and samples, numerical approaches are an interesting alternative for calculations and predictions. Thanks to reliable developed models, the numerical methods can provide very accurate and rapid solution for many realistic engineering problems. The behavior of cracks and crack propagation, however, is still difficult to capture and to simulate as they concern heterogeneous geometries, complicated loadings as well as sophisticated material behaviors.

The best-known numerical approach in Civil Engineering is the Finite Element Method (FEM) for its advantages when dealing with complex geometries, material non-linearities etc. The produced coefficient matrix is banded, symmetric, sparse and diagonal dominant which lessens the computational work during the build-up phase and also dispose of a good convergence rate. The application of the FEM can be seen in almost every domain in civil engineering: elastostatics, elastodynamics, electromagnetics, acoustics, etc. Nevertheless, there exist numerous circustances where solely the knowledge of the boundary values is already sufficient without investigating further in the structure body. For these problems, the Boundary Integral Equations Method (BIEM) arises as an interesting alternative because of its distinct advantages. When coupled with an advanced technique namely Fast Multipole Method (FMM) for faster integral evaluations, the performance of a boundary analysis is greatly enhanced and become sometimes competitive with the finite elements calculations. With the possibility of dealing with unbounded/semi-unbounded media and with evolutive geometries such as the simulation of crack-growth, the Fast Multipole Boundary Element Method is a favorable option in fracture mechanics. In this work, the integral formulation of Galerkin coupled with the Fast multipole method has been used. The modeling of static cracks and crack propagation in single/multi-region domain has been considered.

Summary of Contributions of the Thesis

The aim of this thesis is to develop a numerical tool based on the Fast Multipole Symmetric Galerkin Boundary Element Method (FM-SGBEM) to effectively deal with practical problems. The study of the performance of fractured pavements and the simulation of crack propagation in such complex geometries are some interesting applications that the authors wish to achieve.

In order to reach these main objectives, it is necessary to follow a number of steps:

1. Optimization: The usefulness of a numerical approach can be described by many factors among which the efficiency is indispensible. This characteristic is also known as the compromise between the qualities of the results against the computational costs. For the BEM, this issue matters considerably since the coefficient matrix is fully-populated which penalizes not only the build-up phase but also the data storage and the solution. The performance of the method is on the other hand very difficult to grasp as it has to rely on the convergence rate of the iterative solver (if one is adopted). Even though the Fast multipole method nullifies most of the usual bottlenecks of a traditional boundary analysis, the treatment s of large-scales problems (whose number of unknowns are of order 10^6) still pose a lot of constraints on a modest computer in terms of necessary storage and computational speed. In this work, we have incorporated some interesting refinements concerning (1) the compressed storage and more efficient evaluation of the near-field interactins, (2) a more robust iterative solver based on a nested Generalized Minimal Residual (GMRES). By doing these, the simulations take minimal space for the coefficient storage and produce much quicker convergences thanks to the better preconditioning strategy. The example of multiple cracks in an unbounded domain features up to 3×10^6 unknowns and has been successfully computed on a single-processor PC (under only 20h of calculation).

2. Extension to Multizone Problems: Realistic structures are usually presented with heterogeneity, complex material behaviors and sophisticated loading. It is therefore necessary, at the first step, to adapt the single-region formulation of the SGBEM to multi-region but piecewise homogeneous problems of linear, isotropic materials. The adopted technique takes into consideration the continuity conditions across the interfaces to formulate the Galerkin integral equations for each sub-domain; then via a suitable arrangement of variables, the global system can be obtained by simple linear combination of the sub-matrices. With opposite signs associated with the traction at interface, all the terms sources of dissymmetry simply vanish and leaves the resulted global matrix symmetric. This property is very useful from a computational point of view and can be easily used to couple with the FEM. The algorithm of the multizone SGBEM is represented as a loop on all sub-domains. On each sub-domain, the corresponding local terms are computed, then these contributions are transferred to the global system to form the desired values (Eg. right-handed side vector, matrix-vector product). The second step is to apply the Fast Multipole algorithm to the multizone SGBEM context. Even though, some

cares must be taken in this implementation to ensure the efficiency of the multizone FM-SGBEM, the transition from the multizone SGBEM to the multizone FM-SGBEM in general can be done in a rather straightforward manner and are well-discussed in the thesis. Various tests featuring different types of geometries and different types of solicitations have been considered to verify the validity of the implemented multizone FM-SGBEM.

3. Fatigue Crack Propagation: This application takes advantages of the versatility of a boundary analysis during the re-meshing process. While the domain approaches have higher difficulty for regenerating and updating the mesh at the crack-tip, the BEM can perform this task rather smoothly. A simple example is when the cracks are completely embedded in the solid; the re-meshing is thus done simply by adding new elements to the crack-tip according to the pre-calculated angle and direction of the crack advancement. In this work, the 3D fatigue crack propagation governed by *Paris* law have been considered. Multiple simulations of crack(s) propagation in both single-region and multi-region configurations have been carried out and have produced satisfactory results. More complicated schemes of crack propagation (Eg. surface breaking cracks, interfacial cracks or cross-interfacial cracks) are expected in the future researches.

4. Simulation of Pavements: One possible application of the multizone FM-SGBEM is in the simulation of road structures (pavements). Since these structures are exposed to many unfavorable factors, complex crack distributions are usually present and need to be taken into consideration. Several numerical simulations have been carried-out. Due to the large contrast in geometrical and material characteristics between the sub-layers, the computations can hardly converge. Nevertheless, the successful calculations (on moderate-size pavements) still produce very satisfactory results and promise better performances with further investigations and refinements.

Résumé étendu

Contexte

La modélisation numérique de la multi-fissuration et son influence sur le comportement mécanique des ouvrages du Génie Civil reste un sujet ouvert qui nécessite le développement de nouveaux outils numériques de plus en plus performants. L'approche retenue dans le cadre de nos travaux de thèse est basée sur l'utilisation des concepts des équations intégrales de Galerkin (3D) accélérées par la méthode multipôle rapide (Fast Multipole Method). Les méthodes intégrales sont bien connues pour leur souplesse à définir des géométries complexes, spécialement celles des domaines tridimensionnels et pour la grande précision qui caractérise leurs résultats lors de la détermination des champs singuliers au voisinage de la fissure en mécanique de la rupture. La mise en œuvre d'une stratégie de couplage entre les deux approches (FM-SGBEM: équations intégrales et méthode multipôle rapide) vient renforcer la phase de résolution largement pénalisée lors du traitement des domaines de grandes tailles (nombre de degrés de liberté élevé) par équations intégrales de Galerkin 3D pures. D'autre part, l'étude du comportement des structures qui relèvent du domaine du Génie Civil nécessite la prise en compte dans les modèles, des hétérogénéités et de l'état de fissuration caractérisant leurs domaines tout en considérant la complexité des chargements les sollicitant.

C'est dans ce contexte que nous avons proposé des adaptations aux environnements numériques existants mais aussi des développement de nouvelles procédures afin d'appréhender aux mieux le comportement de ces structures.

Plan de Mémoire

Le contexte et les objectifs de la thèse étant fixés, ce mémoire est découpé en sept chapitres dont le premier se résume à l'introduction générale.

Dans le *chapitre 2*, de nature bibliographique, on y présente les différentes notations et définitions utilisées nécessaires à la compréhension des différents concepts mathématiques introduits dans ce mémoire. Les bases théoriques de la méthode des équations intégrales de Galerkin et de la méthode multipôle rapide sont ensuite exposées. Nous nous sommes efforcés de décrire le plus fidèlement possible et d'une manière exhaustive de l'environnement numérique basé sur la FM-SGBEM. Le *chapitre 3* présente les améliorations apportées dans un premier temps au code de calcul pour en augmenter l'efficacité. Outre un gain de place en mémoire moyennant l'utilisation d'une technique classique de stockage de matrice creuse, les deux principaux apports concernent, d'une part l'utilisation de la méthode 'GMRES flexible' au lieu de la méthode GMRES, dans la résolution itérative du système linéaire et d'autre part la mise en œuvre d'une intégration 'sélective', en utilisant un nombre de points de Gauss variable en fonction de la géométrie des éléments considérés.

Ces travaux d'optimisation ont permis de mener des calculs de grandes tailles avec une accélération significative. L'extension de la méthode au traitement des structures à zones multiples a fait l'objet du *chapitre 4*. Les interfaces entre zones sont supposées parfaites. La prise en compte de la multiplicité des zones a constitué un niveau de boucle supplémentaire et a nécessité des travaux d'adaptation de l'algorithme 'mono-zone'. Les résultats issus des divers tests de validations y sont détaillés et présentés. Le *chapitre 5* traite de l'étude de la propagation de fissures par fatigue selon la *loi de Paris*. Différents aspects concernant l'analyse de contraintes et la procédure du remaillage sont discutés. Dans le *chapitre 6*, la méthode est appliquée à l'étude d'une chaussée souple fissurée. Les fissures transversales sont présentées dans la couche supérieure. Le comportement du modèle sous l'effet de la charge d'un demi-essieu est étudié. Le *chapitre 7* donne, quant à lui, les conclusions générales et discute des perspectives pour les améliorations et les développements des environnements numériques mis en place.

Le mémoire est renforcé par l'insertion de quatre annexes qui donnent des détails relatifs aux formulations de la FM-SGBEM ainsi que les techniques et les schémas d'intégration numériques considérés dans le cadre de cette thèse. Une liste exhaustive des subroutines du code numérique a été également jointe.

Concepts théoriques: Formulations intégrales de Galerkin et Méthode Multipôle Radide

Les formulations intégrales symétriques de Galerkin: La méthode des éléments de frontière par son approche symétrique (Symmetric Galerkin Boundary Element Method), se base sur la discrétisation des équations intégrales de Galerkin [6] dont le support des inconnues est réduit à la frontière du domaine Ω (Fig.2.6) et la détermination des champs caractérisant le comportement du solide fissuré en termes de déplacements et de tensions à la frontière du domaine et de sauts de déplacements à travers les lèvres de fissures permet de réduire d'une dimension la taille des problèmes étudiés.

Les équations décrivant 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 $\mathbf{y}$.

Lors de la phase d'intégration numérique, nous distinguerons 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 permet donc de s'affranchir du stockage des matrices issues de la phase de discrétisation des 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 préconditionnement du système matriciel. Nous reviendrons sur ce dernier point, d'une manière plus détaillée dans les prochaines sections.

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.

Le concept de la méthode multipôle rapide: La FMM est basée sur la reformulation des noyaux constituant les fonctions fondamentales en termes de développements 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 variable $\mathbf{x}$ 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 point $\mathbf{y}$. Ce principe permet une accélération considérable de la phase d'évaluations des intégrales doubles lors de chaque itération propre au calcul par la méthode des éléments de frontière. La FMM 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 calcul assez prohibitifs (proportionnel au 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 un analyse de frontière classique).

Travaux d'optimisation

Le solveur FM-SGBEM pour les équations de l'élastostatique 3D présenté dans le *chapitre 2* a déjà permis d'améliorer les performances de la SGBEM standard. Toutefois, la méthode peut encore être améliorée et différents points qui peuvent augmenter les performances de la FM-SGBEM sont donc présentés dans le corps de ce document. Un important travail d'optimisation des outils numériques existants a été mené afin d'améliorer les temps CPU affichés. Deux stratégies d'optimisation basées, d'une part, sur l'évaluation et le stockage des matrices issues des intégrales

proches (interactions proches stockées matriciellement dans $[Knear]$), et d'autre part sur la procédure de *préconditionnement* du solveur Flexible GMRES, ont été proposées et implantées.

Les intégrations traditionnelles de la SGBEM sont efficacement calculées en adoptant le critère de sévérité (**IS**) [85] pour optimiser l'évaluation numérique des intégrales. La matrice $[Knear]$ issue de la phase de discrétisation des intégrations proches est symétrique et creuse; son stockage comprimé fait appel aux algorithmes de CSRSYM (Symmetric Compressed Sparse Row) [30] minimisant ainsi l'usage de la mémoire vive.

La définition d'un préconditionneur est cruciale mais délicate dans le cadre de la FMM car la matrice du système n'est jamais explicitement formée. On propose ici d'utiliser comme préconditionneur la seule matrice dont on dispose, à savoir $[Knear]$. La solution du système linéaire est réalisée par Flexible GMRES, un outil puissant basé sur l'utilisation de deux solveurs itératifs emboîtés. Le solveur extérieur est un GMRES flexible et le solveur intérieur est un GMRES classique permettant de calculer l'inverse du préconditionneur. La phase de résolution est menée donc avec un processus itératif impliquant le solveur Flexible GMRES à préconditionnement par $[Knear]$; matrice à propriétés spectrales qui permettent une accélération et une convergence remarquables de nos calculs. Les simulations numériques menées à l'aide d'un calculateur à simple processeur sur des structures à plusieurs millions degrés de liberté ont permis de bénéficier d'importantes réductions en temps de calculs pouvant atteindre 80% des temps initiaux. Par ailleurs, les performances de la méthode dépendent de manière très sensible d'un certain nombre de paramètres, qui ont été ajustés au mieux d'après les résultats réalisés, mais qui semblent difficiles à déterminer dans le cas général.

Multizone FM-SGBEM

La méthode présentée au *chapitre 2* est limitée aux milieux homogènes. Pour étudier des configurations réalistes, cette limitation est trop restrictive et c'est dans ce contexte et à travers le *chapitre 4* que nous nous somme proposés d'étendre la formulation de la FM-SGBEM à des configurations multi-domaines.

L'extension des formulations théoriques existantes et le développement des environnements numériques s'y rattachant pour appréhender le comportement des domaines hétérogènes, caractérisés par la variabilité de leurs caractéristiques mécaniques, ont fait l'objet de la seconde partie de nos travaux. Ces structures dites hétérogènes (Fig.1), sont amenées à être représentées par plusieurs sous-domaines séparés par des interfaces internes fictives et sur lesquelles, il conviendra de respecter l'écriture des conditions de continuité et d'équilibre en termes de déplacements et de tensions. La multizone FM-SGBEM est une formulation qui se base sur l'écriture des équations usuelles de la SGBEM pour chaque sous-domaine de la structure globale et qui fait appel par la suite à une judicieuse combinaison des sous-matrices dérivant de chaque sous-domaine pour construire un système global assemblé.

L'implantation numérique de la FMM pour le cas multizone fait appel à une

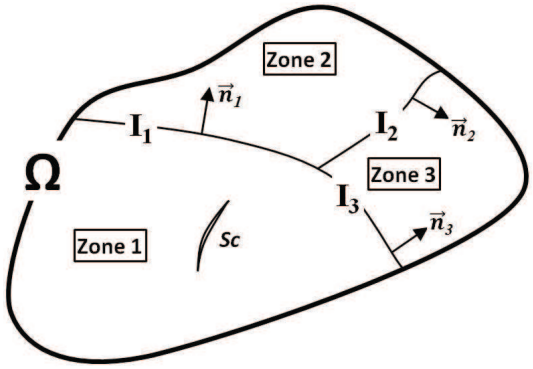
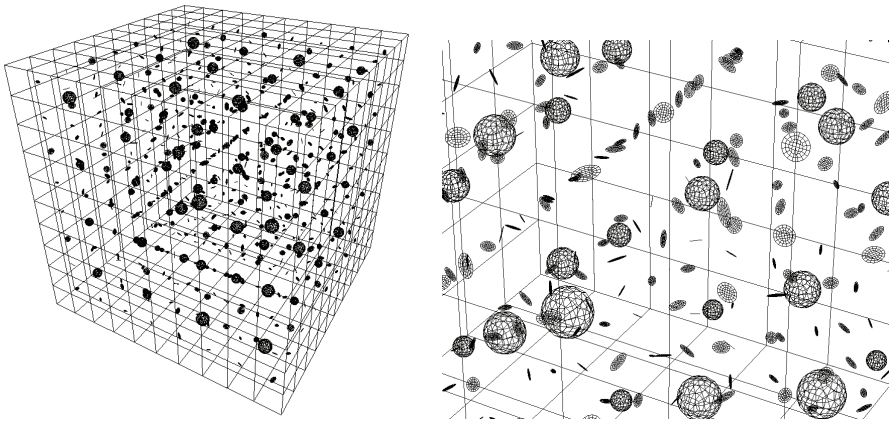


Figure 1: Domaine multizone Ω avec une fissure S_c

procédure de construction de l'arbre d'*octree* englobant toutes les zones du domaine. Nous y traitons chaque zone localement : le calcul des moments, le transfert des termes, les expansions locales se trouvent uniquement dans la zone étudiée. Le résultat des produits matrice-vecteur locaux est ensuite traité dans le repère global du le solveur itératif.

Plusieurs tests et applications numériques ont été étudiés pour différentes configurations de structures hétérogènes telles que l'enveloppe sphérique à trois sous-couches sous pression interne, le barreau sous l'effet de son poids propre, la fissuration d'un matériau du type matrice-inclusion (Fig.2) etc. Ces derniers sont présentés avec détails dans le corps du manuscrit de thèse. Les temps de calculs prouvent que la FM-SGBEM est un outil performant et compétitif par rapport aux autres approches.



(a) (b)

Figure 2: (a) Modèle d'un matériau composite fissuré (avec réseaux de 4x4x4 inclusions sphériques & 8x8x8 fissures) (b) Zoom-in du modèle

Étude de la propagation de fissures

Les aspects liés à la problématique de propagation de fissures 3D dans ces structures ont été intégrés à nos travaux. Les difficultés caractérisant la phase de remaillage 3D lors de la propagation de fissures, soulevées par les méthodes classiques telles que celles des éléments finis, ne sont pas rencontrées lors d'un traitement par la FM-SGBEM et le 3D-remaillage par éléments de frontière est une étape relativement simple. Nous avons fait appel au critère de propagation de fissures découlant de la *loi de Paris* en fatigue.

A partir des sauts de déplacements à travers les lèvres de fissures fournis par la méthode SGBEM étendue aux solides fissurés, il convient de calculer les facteurs d'intensité de contraintes pour déterminer ensuite l'angle et l'amplitude de propagation de la fissure. Un remaillage de la pointe de fissure est alors réalisé sans difficulté particulière. Une illustration de la technique est, dans un premier temps, proposée sur un cylindre comportant une seule fissure circulaire inclinée et dans un deuxième temps, sur des solides multifissurés. Les résultats obtenus sont cohérents avec les références et ont permis de valider les modèles et les environnements numériques développés. La procédure de couplage des méthodes intégrales avec la méthode multipôle rapide a trouvé toute son importance lors de calculs menés en propagation de fissures sur des domaines hétérogènes.

Application à une structure réelle de Génie Civil

La dernière étape de nos travaux a consisté en l'étude d'une structure de chaussée routière souple (Fig.3) présentant un état de fissuration élevé. Cette structure est également caractérisée par un état de contraintes hétérogènes.

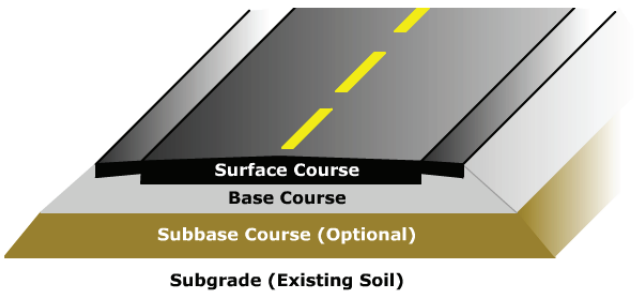


Figure 3: Une structure générale de chaussée

Le modèle de chaussée se compose de trois sous-couches d'épaisseurs respectives 2200 mm pour la plus profonde, 500 mm pour la médiane et 66 mm pour la superficielle. Les valeurs des modules d'Young correspondant sont égales à 80 MPa, 180 MPa et 6110 MPa. Nous nous intéressons à calculer la déflexion de chaussée sous l'effet de chargement de véhicules. Quelques limitations sont apparues lors de la phase de calcul et ce en raison d'un fort contraste entre les dimensions des couches

constitutives et de la grande variabilité de leurs rigidités. Des améliorations aux tests numériques doivent être apportées.

Conclusions et Perspectives

Les environnements numériques développés dans le cadre de cette thèse ont doté l'équipe d'un outil de modélisation original lui permettant de mener des calculs sur des structures à plusieurs millions de degrés de liberté, dans un temps optimisé et moyennant l'utilisation d'un calculateur modeste à simple processeur. Nous nous sommes également affranchis des difficultés liées à la modélisation des géométries et des configurations complexes reflétant au mieux le comportement des ouvrages du Génie Civil. Des modèles pour étudier des structures hétérogènes, pesantes, présentant un état de fissuration avancé en propagation ont été intégrés. Les résultats encourageants obtenus laissent entrevoir des perspectives intéressantes quant à l'adaptation de ces travaux pour inclure des aspects en élasto-statique avec déformations ou contraintes initiales, se présentant notamment en thermoélasticité ou en micromécanique lors de la présence d'inclusions, sièges de déformations anélastiques.

Principales publications associées à nos travaux

- Q.T. Trinh, S. Mouhoubi, C. Chazallon, M. Bonnet, *Les éléments de frontière accélérés par la Méthode Multipôle Rapide (FMM) pour modéliser des domaines 3D multizone fissurés*, 26-30 Août 2013, Bordeaux.

- Q.T. Trinh, S. Mouhoubi, C. Chazallon, M. Bonnet, *Les éléments de frontière accélérés par la Méthode Multipôle Rapide (FMM) pour modéliser des domaines 3D multizone fissurés*, Rencontres universitaires de Génie Civil, 4-6 juin 2014, Orléans.

- Q.T. Trinh, S. Mouhoubi, C. Chazallon, M. Bonnet, *Three-dimensional modeling of fracture by fast multipole symmetric Galerkin boundary elements method. Application to multi-fractured media*, European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS, 10-14 Septembre 2012, Vienne.

- Q.T. Trinh, S. Mouhoubi, C. Chazallon, M. Bonnet, *Fast multipole boundary elements method for multizone problems*, 11^{th} World Congress on Computational Mechanics, 20-25 juillet 2014, Barcelona.

- Q.T. Trinh, S. Mouhoubi, C. Chazallon, M. Bonnet, *Multizone and Multicrack media modelled with the Fast Multipole Method applied to Symmetric Galerkin Boundary Element Method*, (accepté) à Engineering Analysis with Boundary Elements.

Contents

Chapter 1

General Introduction

Contents

1.1 Context and Motivation

Even though the Finite Element Method (FEM) is arguably the best approach for all problems in engineering, its effectiveness and efficiency is not comparable to the Boundary Element Method (BEM) in certain circumstances such as unbounded media, crack propagation... The BEM is a numerical computational method of solving linear partial differential equations which have been reformulated as integral equations. The application of BEM can be found widely in many areas of engineering including elasticity [72], geomechanics [70, 71], structural mechanics [81], electromanetics [79], acoustics [76], fracture mechanics [11].... Conceptually, the BEM is based on the discretization of the boundary integral equations (BIEs), so only the boundary of the problem is needed to be discretized. This distinguishing feature (opposing to volume methods) typically reduces the geometrical dimension by one thus makes the data generation in a boundary analysis much more simple and faster. In correlation to this fact, if one considers a BE model and a FE model of the same physical problem which give two approximate solutions of comparable accuracy, it turns out that the number of unknowns in the BE model is much lower than the FE model where 3D elements are involved.

An approach based on a variational (weak) version of the integral equations (namely Symmetric Galerkin BEM) is a highly robust and efficient alternative boundary elements method. The SGBEM has been the subject of extensive investigations since it was first proposed in 1979 [13]. The key advantage of the Galerkin formulations is the ability to treat hypersingular (as well as standard singular) equations by means of standard continuous elements. The more commonly-used method called *collocations* requires a differentiable boundary interpolation which is inherently difficult and computationally expensive. Considering the essential role of hypersingular equations in the fracture mechanics [68–70], this advantage of SGBEM become more significant. Additionally, despite that SGBEM is more expensive than the *collocations* due to the evaluation of the double surface integrals,

the method still stays very attractive because of its symmetric matrix which can be exploited to lessen the numerical work or to couple with finite element [3, 5]. Application of the Galerkin method can be found in a broad range of engineering problems: 3D steady and incompressible flow by Capuana et al. [73], a Stokes problem with general boundary condition with slop condition was reported by Reidinger and Steinbach [80], a fully symmetric formulation for interface and multi-zone problems by Maier [78], by Gray and Paulino [74], dynamic soil-structure interaction by Lehman and Antes [77] ...

However, the boundary elements only show its efficiency than other methods in special contexts where there is a high ratio of surface or where the geometries evolve over time. Moreover, the number of unknowns which a traditional boundary analysis is capable of solving, is also limited: Because the matrix issued from the discretization phase is fully-populated, the storage requirements and computational time will tend to grow according to the square of the problem size. As a result, the computational resources exhaust rapidly on standard devices and boundary elements are restraint in treating only problems of moderate size. Many methods have been introduced to alleviate this $O(N^2)$ scaling. One of the most successful algorithm is the Fast Multipole Method (FMM) [33]. Based on a reformulation of the Green functions, the method is proved to reduce the complexity to nearly linear with the number of unknowns $O(N)$. With the apperance of the FMM in a boundary elements computation, the storage requirements and computational cost can be effectively reduced to $O(NlogN)$. These improvements make the boundary integral methods numerically very efficient and competitive with domain approaches. Over recent decades, the Fast algorithm has been applied in various fields: elastostatics [5,32], fracture problems [4,11], fluid dynamics [46], electromagnetic [47]... The fact that this method is nominated one of the top 10 algorithms of the century show how much influence it has on numerical analysis.

In order to solve 3D problems of engineering interest, the first unavoidable challenge is the computational efficiency. Considerable efforts have been spent in this thesis for this starting purpose. The main goals of our work concerns the extensions and applications of the FM-SGBEM in complex and large-scale issues such as fractured multizone, fractured composite and crack propagation.

All the numerical codes have been written in Fortran 90. There are a number of authors who have contributed for the early developments of this code: Mouhoubi [3] and Pham [1] in 2010 - FM-SGBEM ... The optimization and extensions of the FM-SGBEM code to multizone, composite and crack propagation have been developed in-house by the author. Some basic operations (matrix, vector, algebra), however, have been adopted from the optimized libraries (BLAS). To generate the meshes, a commercial pre- and post-processor software (Gid) has been used. To visualize and analyze the output results, Medit has been utilized. SAP 2000 has also been used to provides some simple finite elements references.

1.2 Outlines of the Thesis

In *chapter 2*, the basic mathematical framework of the Elasticity and Fracture Mechanics are briefly recalled. The Symmetric Galerkin formulations for each problem are given. Then some aspects of the numerical solution by the Boundary Elements Method (discretization, evaluation of integrals, use of an iterative solver ...) are mentioned. We discuss afterward the need of a Fast algorithm for the SGBEM. The principle of the Fast Multipole Method is described and the formulation of the FMM are introduced. The computational scheme of the multi level FMM is also illustrated. The numerical aspects of the program as well as a generic 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 issues when dealing with large-scale problems resides in the storage of the near-interaction matrix. A simple compress format has been proposed, the memory constrain has been easily tackled as the numerical tests have been able to deal with problems of 3×10^6 unknowns. Secondly, the long iterative solution times produced by GMRES has also been reduced with help of a more powerful preconditioning strategy by Flexible GMRES. Furthermore, analysis on different choices of input parameters have also been conducted. The validation tests have been run and the clear improvements of the numerical code have been reported.

Chapter 4 introduces the extension of the FM-SGBEM in the context of heterogeneity, or more precisely, the study of internal interfaces. An advanced and efficient technique which render symmetric the global system has been adopted. The multizone formulation as well as the numerical implementation of the method has been described. A number of validation tests in elasticity or fracture mechanics has shown a great accuracy of the algorithm. As the numerical code has become able to treat more complex configurations, many interesting applications have been opened: multiple-layers road-structure or composite materials... The last example in this chapter illustrates a model of solid with presence of different sizes and shapes of random spherical inclusions and cracks.

In *chapter 5*, a simple crack propagation based on the *Paris* fatigue law has been discussed. We introduced first the propagation criterion then the remeshing strategy. The numerical aspects of the FM-SGBEM to deal with the crack propagation have been presented. A couple of numerical experiments have been carried out and the obtained outputs are very encouraging.

Chapter 6 presents an application of the method FM-SGBEM in a practical civil engineering problem: A multi-layer road structure is considered. Real loading and material properties have been assigned for the models. For verification purposes, there are a number of references that have been attributed. One of which is extracted from a finite calculation of CAST3M and the others are provided from real measurements taken on site by the colleagues in COMPANY NAME. Ideally, these configurations can be solved easily with the implemented FM-SGBEM but in fact, they feature unfavorable conditions for a 3D boundary analysis such as the

very thin layer design or considerable contrasts between the constitutive materials' properties. These elements may have caused the coefficient matrix ill-conditioned and have therefore led to very slow or nearly unachievable convergences. Nevertheless, the converged calculations show a very good agreement with the provided references and promise to obtain better results with future refinements.

Lastly, some concluding remarks and discussions have been given in *Chapter 7*. The perspectives as well as the directions for future researches have also been introduced.

The thesis also contains 4 *Annexes* which present the details, descriptions, formulations and complementary techniques related to the method FM-SGBEM as well as the numerical program. Annex A introduces the integration techniques for regular and singular cases. Annex B and D show the detail of the Fast Multipole formulation when applied to the SGBEM. Annex C describes the structure of a generic FM-SGBEM code along with the name and utility of the most important subroutines.

Chapter 2

Basic Framework of the Fast Multipole SGBEM

Contents

Among many alternatives in linear elastic fracture mechanics, boundary element method (BEM) is a very attractive option. The advantages in the BEM arise from the reduction of problem dimension and from its superiority in treating specific domains of application. Unlike domain methods, the BEM can treat infinite/semi-infinite domains by discretizing only the finite boundary (Eg. Interaction soil/structure, exterior problems ...). Another key feature of the BEM in fracture mechanics is that the singular stress field at crack front is not approximated (it is shown in [19] that more accurate results for stress can be obtained). Also, with evolutive boundaries (Eg. crack propagation...), it is much easier to re-mesh the cracks by BEM than by finite methods and the outer boundary is not required to be re-meshed during the modeling of crack propagation. Moreover, the

study of cracks often leads to the modeling of realistic thus very complex configurations where many materials are considered. This class of problem is also known as multidomain or interface problems. The BEM approach is still very attractive as it provides a natural treatment of the interface's continuity. Taking an example in elasticity, for the displacement-based FEM, it is difficult to enforce the continuity of traction, however, for boundary integral equations, this condition appears directly in the formulations. In [10], a simple technique is introduced to incorporate all the interfacial conditions in the SGBEM formulations and to construct the global symmetric system in a very efficient manner.

During its development, with help of advanced techniques and optimizations, BEM has become an efficient tool and can be used in many other interesting application fields. An approach of the BEM, based on the Galerkin approximation, namely Symmetric Galerkin BEM (SGBEM) [6] is highly advantageous in fracture studies [25]. First, less restrictive requirement than the collocation approach is imposed on the displacements, thus standard continuous elements can be employed to evaluate the hypersingular integrals. Secondly, the discretized coefficient matrix is symmetric which lessens greatly the numerical costs (for both matrix construction and storage). The main drawback of the method is that the double surface integrals typically give rise to symmetric but fully-populated matrix. Consequently, the solution becomes very expensive as the number of unknowns grows: the buildup requirements are of order $O(N^2)$, N being the number of unknowns, and a direct solver may take $O(N^3)$ operations to achieve the results. Therefore, the boundary method is limited to relatively small problems. On the other hand, since the coefficient matrices in domains approaches are banded and the computational complexity is of $O(N)$, it is easier and more suited to treat large-scale problems. Fortunately, the usual slow evaluation of double integrals in the SGBEM can be sped up by the fast multipole method (FMM). Initially introduced by Rohklin [33], this algorithm considers one group of particles and represents it by an intermediate pole. As all the interactions with this group is transferred via this pole, the overall number of operations is greatly reduced. By coupling the SGBEM with the Fast Multipole Method (FMM) and an iterative solver, the complexity of the method is significantly reduced: $O(N)$ for the storage requirements and $O(NlogN)$ for the operation count [4]. Therefore, the range of boundary analysis can now be extended to large-scale practical issues with a very good performance (see, for example, application of FMM in elastodynamics [2]).

In this chapter, some basic concepts of the linear elasticity and fracture mechanics are briefly recalled. The principal of virtual work and Betti's reciprocal theorem are described in order to derive the boundary integral equations. The symmetric Galerkin approach (SGBEM) is also mentioned and the regularized version is presented. Some important aspects and limitations of the numerical solution including the use of a preconditioner and an iterative solver are discussed. The computational limitations of the BEMs arise in this phase which motivate the use of the Fast multipole method. Simple descriptions of the principle and the formulations of the Fast algorithm are given. Lastly, a generic computational scheme of a Fast

Multipole SGBEM is described and discussed.

2.1 Elastostatics

All Boundary Elements Analysis are rooted in the mathematical theory of linear elasticity which provided central concepts such as effect superposition, influence functions, reciprocity relationships and basic ingredients such as Kelvin's fundamental solution (1848), Somigliana's identity (1886)... Therefore, in a most natural possible approach, we would like to introduce briefly the basic aspects of elasticity then the formulation of the boundary integral equation and the Symmetric Galerkin BEM in linear elastostatics. Afterward, we will discuss about its developments and extensions in complex fractured configurations in the following sections.

2.1.1 Elastostatics problems

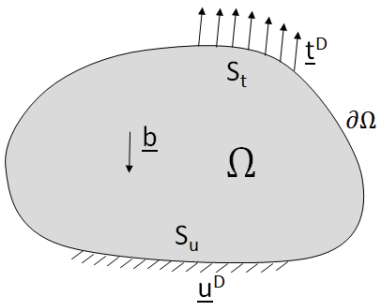


Figure 2.1: Elastic solid

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 (shown in figure etc). The stress state at the point y inside Omega 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:

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

In the absence of body forces, the equilibrium equation can be written as:

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

where $(.)_{,j}$ stands for the derivative of $(.)$ along the j^{th} direction.

Strains, described by tensor ε_{ij}, are related to displacement by:

$$\varepsilon_{ij}(\mathbf{y}) = \frac{1}{2}[u_{i,j}(\mathbf{y}) + u_{j,i}(\mathbf{y})] \qquad (\mathbf{y} \in \Omega) \tag{2.3}$$

The consecutive law for linear elastic and isotropic material (Hooke's law) can be written as:

$$\sigma_{ij}(\mathbf{y}) = \lambda\delta_{ij}\varepsilon_{kk}(\mathbf{y}) + 2\mu\varepsilon_{ij}(\mathbf{y}) \tag{2.4}$$

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

$$\sigma_{ij}(\mathbf{y}) = C_{ijhk}\varepsilon_{hk}(\mathbf{y}) \tag{2.5}$$

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

$$C_{ijhk} = \lambda\delta_{ij}\delta hk + \mu(\delta_{ik}\delta jh + \delta_{ih}\delta jk) \tag{2.6}$$

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

$$C_{ijhk} = 2\mu[\frac{\nu}{1-2\nu}\delta_{ij}\delta hk + \delta_{ik}\delta jh + \delta_{ih}\delta jk] \tag{2.7}$$

Therefore, the equilibrium equation can be written as:

$$\mu u_{i,jj}(\mathbf{y}) + (\lambda+\mu)u_{j,ji}(\mathbf{y}) = 0 \qquad (\mathbf{y} \in \Omega) \tag{2.8}$$

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 (2.2) 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 (2.3) 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 2 different states:

$$\begin{aligned}\int_\Omega \sigma^1_{ij}\varepsilon^2_{ij}dV &= \int_{\partial\Omega} t^1_i u^2_i dS + \int_\Omega b^1_i u^2_i dV \\ \int_\Omega \sigma^2_{ij}\varepsilon^1_{ij}dV &= \int_{\partial\Omega} t^2_i u^1_i dS + \int_\Omega b^2_i u^1_i dV\end{aligned} \tag{2.9}$$

Betti's reciprocal theorem can be obtained:

$$\int_\Omega (b_i^2 u_i^1 - b_i^1 u_i^2) dV = \int_{\partial\Omega} (t_i^1 u_i^2 - t_i^2 u_i^1) dS \tag{2.10}$$

Integral equations

The integral equation for the elastostatic problem can be derived from the 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) can be chosen to represent the response of the infinite domain Ω_∞ to a concentrated force acting at point $\mathbf{x}$:

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

where $\delta(\mathbf{x}, \mathbf{y})$ is the Dirac delta function.

Expression of the fundamental solutions U_i^k and Σ_{ij}^k are given:

$$U_i^k(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{1}{8\pi\mu r}\Big[\frac{3-4\nu}{2(1-\nu)} - \frac{1}{2(1-\nu)} r_{,i} r_{,k}\Big] \tag{2.11}$$

$$\Sigma_{ij}^k(\mathbf{x}, \tilde{\mathbf{x}}) = -\frac{1}{8\pi(1-\nu)r^2}\Big[3r_{,i}r_{,k}r_{,j} + (1-2\nu)(\delta_{ik}r_{,j} + \delta_{jk}r_{,i} - \delta_{ij}r_{,k})\Big] \tag{2.12}$$

The Somigliana integral equation for displacements can be obtained by introducing equation (2.11) into equation (2.10):

$$\begin{aligned} u_k(\mathbf{x}) = &- \int_{\partial\Omega} u_i(\mathbf{y}) n_j(\mathbf{y}) \Sigma_{ij}^k(\mathbf{x}, \mathbf{y}) dS_y \\ &+ \int_{\partial\Omega} t_i(\mathbf{y}) U_i^k(\mathbf{x}, \mathbf{y}) dS_y + \int_\Omega b_i(\mathbf{y}) U_i^k(\mathbf{x}, \mathbf{y}) dV \qquad (\mathbf{x} \in \Omega) \end{aligned} \tag{2.13}$$

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

$$\begin{aligned} \sigma_{ij}(\mathbf{x}) = &C_{klab} \int_{\partial\Omega} u_k(\mathbf{y}) n_j(\mathbf{y}) \frac{\partial}{\partial y_b} \Sigma_{ij}^a(\mathbf{y}, \mathbf{x}) dS_y \\ &- \int_{\partial\Omega} t_k(\mathbf{y}) \Sigma_{ij}^k(\mathbf{y}, \mathbf{x}) dS_y + \int_\Omega b_i(\mathbf{y}) \Sigma_{ij}^k(\mathbf{y}, \mathbf{x}) dV \qquad (\mathbf{x} \in \Omega) \end{aligned} \tag{2.14}$$

The above equations are called the integral representations which permit to calculate 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 [6]). This procedure results in a boundary integral equation for displacements:

$$\int_{\partial\Omega}\{[u_i(\mathbf{y})-u_i(\mathbf{x})]T_i^k(\mathbf{x},\mathbf{y})-t_i(\mathbf{y})U_i^k(\mathbf{x},\mathbf{y})\}dS_y=\int_{\Omega}\rho b_i(\mathbf{y})U_i^k(\mathbf{x},\mathbf{y})dV \quad (2.15)$$

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

2.1.2 Symmetric Galerkin formulations in Elastostatics

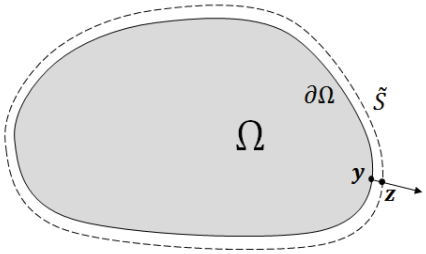


Figure 2.2: Boundary $\partial\Omega$ and auxiliary surface $\tilde{S}$

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 refered to [6] for more details of 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):

$$\mathbf{y}\in\Omega\rightarrow\mathbf{z}=\mathbf{F}(\mathbf{y})\in\tilde{S} \quad (2.16)$$

As the image of the boundary $\partial\Omega$, the surface $\tilde{S}$ also consists of 2 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 limiting process $\tilde{S}\rightarrow\partial\Omega$. Following the approach introduced by Sitori et al. [43], the SGBEM procedure consists basically of 2 distinct steps:

1. At first, the classical displacement and traction boundary integral equations are enforced in a weak sense on the auxilary 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 analytical 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 [36, 37, 43]) is shown below:

$$\text{Find } (\mathbf{u},\mathbf{t}) \in \mathcal{V}_u \times \mathcal{V}_t, \ \begin{cases} \mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}}) + \mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}}) = \mathbf{F}_u(\tilde{\mathbf{u}}) \\ \mathbf{B}_{ut}(\mathbf{u},\tilde{\mathbf{t}}) + \mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \mathbf{F}_t(\tilde{\mathbf{t}}) \end{cases} \quad \forall (\tilde{\mathbf{u}},\tilde{\mathbf{t}}) \in \mathcal{V}_u \times \mathcal{V}_t \tag{2.17}$$

using the the bilinear forms:

$$\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}}) = \int_{S_t}\int_{S_t} (Ru)_{iq}(\mathbf{x}) B_{ikqs}(\mathbf{r}) (R\tilde{u})_{ks}(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.18}$$

$$\mathbf{B}_{tu}(\mathbf{t},\tilde{\mathbf{u}}) = -\int_{S_u}\int_{S_t} \mathbf{t}_k(\mathbf{x}) T_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{\mathbf{u}}_i(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.19}$$

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

$$\mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \int_{S_u}\int_{S_u} \mathbf{t}_k(\mathbf{x}) U_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{\mathbf{t}}_i(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.21}$$

and the linear forms:

$$\begin{aligned} \mathbf{F}_u(\tilde{\mathbf{u}}) &= (\kappa-1)\int_{S_t} t_k^D(\mathbf{x})\tilde{u}_k(\mathbf{x}) dS_x + \int_{S_t}\int_{S_t} t_k^D(\mathbf{x}) T_i^k(\mathbf{x},\tilde{\mathbf{x}})[\tilde{u}_i(\tilde{\mathbf{x}}) - \tilde{u}_i(\mathbf{x})] dS_{\tilde{\mathbf{x}}} dS_x \\ &\quad - \int_{S_t}\int_{S_u} t_k^D(\mathbf{x}) T_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{u}_i(\mathbf{x}) dS_{\tilde{\mathbf{x}}} dS_x - \int_{S_u}\int_{S_t} (Ru)_{iq}^D(\mathbf{x}) B_{ikqs}(\mathbf{r}) (R\tilde{u})_{ks}(\tilde{\mathbf{x}}) dS_{\tilde{\mathbf{x}}} dS_x \\ \mathbf{F}_t(\tilde{\mathbf{t}}) &= \kappa \int_{S_u} u_k^D(\mathbf{x})\tilde{t}_k(\mathbf{x}) dS_x + \int_{S_u}\int_{S_u} [\tilde{u}_i^D(\mathbf{x}) - \tilde{u}_i^D(\tilde{\mathbf{x}})] T_i^k(\tilde{\mathbf{x}},\mathbf{x}) \tilde{t}_k(\tilde{\mathbf{x}}) dS_x dS_{\tilde{\mathbf{x}}} \\ &\quad - \int_{S_t}\int_{S_u} t_k^D(\mathbf{x}) U_i^k(\mathbf{x},\tilde{\mathbf{x}}) \tilde{t}_i(\tilde{\mathbf{x}}) dS_{\tilde{\mathbf{x}}} dS_x - \int_{S_u}\int_{S_t} u_i^D(\tilde{\mathbf{x}}) T_i^k(\tilde{\mathbf{x}},\mathbf{x}) \tilde{t}_k(\tilde{\mathbf{x}}) dS_x dS_{\tilde{\mathbf{x}}} \end{aligned} \tag{2.22}$$

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:

$$U_i^k(\mathbf{x},\tilde{\mathbf{x}}) = \frac{1}{8\pi\mu r}\Big[\frac{3-4\nu}{2(1-\nu)} - \frac{1}{2(1-\nu)} r_{,i} r_{,k}\Big] \tag{2.23}$$

$$T_i^k(\mathbf{x},\tilde{\mathbf{x}}) = -\frac{1}{8\pi(1-\nu)r^2}\Big[3r_{,i}r_{,k}r_{,j} + (1-2\nu)(\delta_{ik}r_{,j} + \delta_{jk}r_{,i} - \delta_{ij}r_{,k})\Big] n_j(\mathbf{y}) \tag{2.24}$$

having set:

$$\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}} \qquad r = \| \mathbf{r} \| \qquad \widehat{\mathbf{r}} = \mathbf{r}/r \tag{2.25}$$

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

$$\begin{aligned} \mathcal{V}_u &= \{\mathbf{u} \in H^{1/2}(S), supp(\mathbf{u}) \subset S_t\} \\ \mathcal{V}_t &= \{\mathbf{u} \in H^{-1/2}(S), supp(\mathbf{t}) \subset \overline{S}_u\} \end{aligned} \tag{2.26}$$

and $\tilde{\mathbf{u}}$,$\tilde{\mathbf{t}}$ are test displacements and tractions. Natural finite-dimensional subspaces of $\mathcal{V}_u$ and $\mathcal{V}_t$ for Galerkin discretization consist of continuous interpolatons of $\mathbf{u}$ over S_t with a zero trace on the edge ∂S_t and piecewise-continuous interpolation of $\mathbf{t}$ over S_u. In particular, in contrast to the case of the traction CBIE, the interpolation method puts no requirement on the derivatives of $\mathbf{u}$. Note that the data $\mathbf{u}^D$ appearing in (2.22) is actually an arbitrary extension to $\Delta\Omega$ of the dispalcement 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 (2.18) and (2.22) are written in regularized form [36], which involve only weakly singular double surface integrals with $O(r^{-1})$ integrands. The regularization involes the Stokes theorem together with indirect regularization. The surface curl operator R arising as a result of this manipulation is defined [35] 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:

$$B_{ikqs}(\mathbf{r}) = \frac{\mu}{4\pi(1-\nu)} \Big[\Big(\delta_{qs}\delta_{ik} - 2\delta_{is}\delta_{kq}\nu - (1-\nu)\delta_{iq}\delta_{ks} \Big) r^{-1} + \delta_{qs} r_{,i} r_{,k} \Big] \tag{2.27}$$

2.2 Fracture Mechanics

It is well-known that fractures are critical phenomena in civil engineering since they can lead to complete destruction of the structures. Because of its importance and complexity, fracture mechanics have become a field of research interest to mathematicians, scientists and engineers since many decades. Analysis of fracture are one of the most successful application areas for the boundary integral equations. The method possesses inherent advantages for these calculations over domain approaches especially when cracks are directly represented as displacement discontinuity loci and the traction integral equation is employed to enforce static conditions on the crack itself.

Fracture mechanics are analogous to the study of cracks. In order to investigate precisely the phenomenon related to fractures, three-dimensional analysis of cracks are indispensable. However, 3D calculations always tend to become large and sophisticated if we take into account the propagation of cracks. In these models, a sufficiently fine mesh is needed for the crack front's orientation and advancement.

Afterward, the remeshing is required and the computation must be repeated at each configuration. These tasks have proven to be very difficult with classic domain approaches since a large amount of volume-elements have to be considered.

The SGBEM, on the contrary, has several key advantages in fracture applications: (i) it considers only the boundary of the problem thus lessen the most the cost of build-up data and remeshing (ii) the matrix issued from the discretization is symmetric (iii) no smoothness is required on the displacement to evaluate the hypersingular integral thus making use of highly efficient quarter-point elements to capture the crack-tip behavior (iv) the weighted formulation of Galerkin provides a smoother solution in the neighborhood of geometric discontinuity. In the following subsections, the basic concepts of fracture mechanics are recalled and the formulation of the SGBEM in this matter is given.

2.2.1 Fracture Mechanics problems

Fracture mechanics is the field of mechanics concerned with the study of the crack propagation of cracks in materials. It uses methods of analytical solid mechanics to calculate the driving force on a crack and those experimental solid mechanics to characterize the material's resistance to fracture. There are three ways of applying a force to enable a crack to propagate (shown in Fig. 2.3):

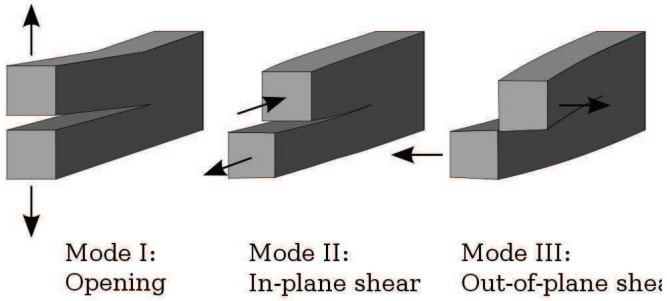


Figure 2.3: Cracks modes

- **Mode I** - Opening mode (a tensile stress normal to the plane of the crack)
- **Mode II** - Sliding mode (a shear stress acting parallel to the plane of the crack and perpendicular to the crack front)
- **Mode III** - Tearing mode (a shear stress acting parallel to the plane of the crack and parallel to the crack font)

Stress intensity Factors - K (SIFs) are used in fracture mechanics to predict the stress state near the tip of a crack caused by a remote load. It is a theoretically

construct usually applied to a homogeneous, linear elastic material and is useful for providing a failure criterion for brittle materials and is a critical technique in the discipline of damage tolerance. The magnitude of K depends on sample geometry, the size and location of the crack, and the magnitude and the modal distribution of loads on the material.

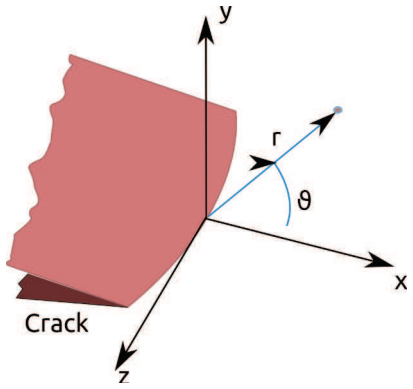


Figure 2.4: Coordinates at the crack tip

Linear elastic predicts that the stress distribution (σ_{ij}) near the crack tip, in polar coordinates (r, θ) (Fig.2.4) with origin at the crack tip, has the form:

$$\sigma_{ij}(r,\theta) = \frac{K}{\sqrt{2\pi r}} f_{ij}(\theta) + O(1) \tag{2.28}$$

where K is the SIF (with units of *stress* x $length^{0.5}$) and f_{ij} is a dimensionless quantity that depends on the load and geometry. Different subscripts are used to designate the SIF for three different modes:

$$K_I = \lim_{r\to 0} \sqrt{2\pi r}\sigma_{yy}(r,0) \tag{2.29}$$

$$K_{II} = \lim_{r\to 0} \sqrt{2\pi r}\sigma_{yx}(r,0) \tag{2.30}$$

$$K_{III} = \lim_{r\to 0} \sqrt{2\pi r}\sigma_{yz}(r,0) \tag{2.31}$$

In the numerical modeling of linear elastic fracture mechanics problems by the SGBEM, the cracks are represented as displacement discontinuity and the traction integral equation is employed to enforce the static condition on the crack itself. The crack S_c separates locally the solid into two parts, S_{c-} with the unit outward normal n, S_{c+} with opposite normal -n. Surface S_c is called a crack. When the solid is subjected to loads, there is a crack opening displacement:

$$\Delta \mathbf{u}_i = \mathbf{u}_i^+ - \mathbf{u}_i^- \tag{2.32}$$

The normal jump is non negative. The condition $\Delta u_3 \geq 0$ must be considered in some interface crack problems in order to avoid overlapping phenomena, also

guarantee the uniqueness of the solution. When the normal crack discontinuity is positive, the stress vector vanishes on the crack.

Quarter-point elements

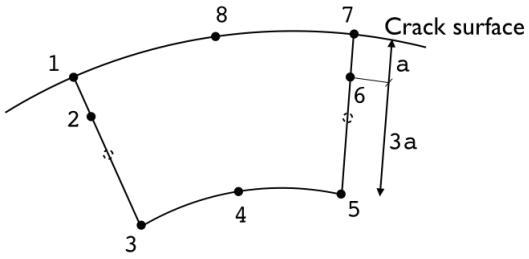


Figure 2.5: Quarter point element

In order to better capture the behavior of the fissure in the crack-front (which presents the singularity), the elements adjacent to the front of the crack are modified. Let us consider a quadrilateral 8-nodes isoparametric element adjacent to the crack's front (see Fig.2.5). 2 middle nodes 2 and 6 are pushed closer to the crack front by a quarter of the element edge's length:

$$\mathbf{y}^2 - \mathbf{y}^1 = -as$$
$$\mathbf{y}^3 - \mathbf{y}^1 = -4as$$

The node $\mathbf{y}^2 \in [\mathbf{y}^1, \mathbf{y}^3]$ is at quarter of the length $\| \mathbf{y}^1 - \mathbf{y}^3 \| = 4a$. On the segment $[\mathbf{y}^1, \mathbf{y}^3]$, the interpolation is quadratic and the point $\mathbf{y}$ is interpolated by:

$$\mathbf{y} = N_1(\varepsilon)\mathbf{y}^1 + N_2(\varepsilon)\mathbf{y}^2 + N_3(\varepsilon)\mathbf{y}^3 \tag{2.33}$$

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

$$N_1(\varepsilon) = \frac{1}{2}\varepsilon(1-\varepsilon)$$
$$N_2(\varepsilon) = 1 - \varepsilon^2$$
$$N_3(\varepsilon) = \frac{1}{2}\varepsilon(1+\varepsilon)$$

We can pose that:

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

and the distance becomes:

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

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

$$\begin{aligned}\Delta\mathbf{u}(\mathbf{y}) &= N_2(\varepsilon)\Delta\mathbf{u}^2 + N_3(\varepsilon\Delta\mathbf{u}^3) \\ &= (1+\varepsilon)\left[\Delta\mathbf{u}^2 + (\frac{1}{2}\Delta\mathbf{u}^3 - \Delta\mathbf{u}^2)\varepsilon\right]\end{aligned}$$

In paying attention to $\Delta\mathbf{u}^1 = 0$ because the opening displacements are nulls on the crack front. And in function of $\mathbf{d}$, we can rewrite the above expression as:

$$\Delta\mathbf{u}(\mathbf{y}) = \left(\frac{\mathbf{d}}{2}\right)^{1/2}\left[2\Delta\mathbf{u}^2 - \frac{1}{2}\Delta\mathbf{u}^3 + \left(\frac{\mathbf{d}}{2}\right)^{1/2}\left(\frac{1}{2}\Delta\mathbf{u}^3 - \Delta\mathbf{u}^2\right)\right]$$

With 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 $\Delta\mathbf{u}^2$ and $\Delta\mathbf{u}^3$:

$$\begin{aligned}K_I^1 &= \lim_{\mathbf{d}\longrightarrow 0}\frac{\mu}{4(1-\nu)}\left(\frac{2\pi}{\mathbf{d}}\right)^{1/2}\Delta u_n \\ &= \lim_{\mathbf{d}\longrightarrow 0}\frac{\mu}{4(1-\nu)}\left(\frac{2\pi}{\mathbf{d}}\right)^{1/2}\left[2\Delta\mathbf{u}_n^2 - \frac{1}{2}\Delta\mathbf{u}_n^3 + \left(\frac{\mathbf{d}}{2}\right)^{1/2}\left(\frac{1}{2}\Delta\mathbf{u}_n^3 - \Delta\mathbf{u}_n^2\right)\right] \\ &= \frac{\mu}{4(1-\mu)}\left(\frac{2\pi}{a}\right)^{1/2}\left[2\Delta\mathbf{u}_n^2 - \frac{1}{2}\Delta\mathbf{u}_n^3\right] \qquad (2.34)\end{aligned}$$

Besides, we can use the same procedure to compute the other factors K_{II}^1 or K_{III}^1.

2.2.2 Symmetric Galerkin formulations in Fracture Mechanics

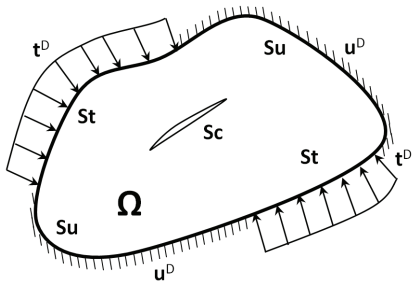


Figure 2.6: Solid containing a crack

Considering a fractured solid Ω subjected to prescribed tractions t^D on the boundary St 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 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. Details of the mathematical developments of the SGBEM can be found in [6]. The boundary integral formulation for this problem is written as follow:

$$\begin{aligned}
&\text{Find } (\mathbf{u}, \mathbf{t}, \boldsymbol{\Delta u}) \in \mathcal{V}_u \times \mathcal{V}_t \times \mathcal{V}_c, \\
&\begin{cases}
\mathbf{B}_{uu}(\mathbf{u}, \tilde{\mathbf{u}}) + \mathbf{B}_{tu}(\mathbf{t}, \tilde{\mathbf{u}}) + \mathbf{B}_{\Delta uu}(\boldsymbol{\Delta u}, \tilde{\mathbf{u}}) = \mathbf{F}_u(\tilde{\mathbf{u}}) \\
\mathbf{B}_{ut}(\mathbf{u}, \tilde{\mathbf{t}}) + \mathbf{B}_{tt}(\mathbf{t}, \tilde{\mathbf{t}}) + \mathbf{B}_{\Delta ut}(\boldsymbol{\Delta u}, \tilde{\mathbf{t}}) = \mathbf{F}_t(\tilde{\mathbf{t}}) \\
\mathbf{B}_{u\Delta u}(\mathbf{u}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) + \mathbf{B}_{t\Delta u}(\mathbf{t}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) + \mathbf{B}_{\Delta u\Delta u}(\boldsymbol{\Delta u}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) = \mathbf{F}_{\Delta u}(\boldsymbol{\Delta}\tilde{\mathbf{u}})
\end{cases} \\
&\forall (\tilde{\mathbf{u}}, \tilde{\mathbf{t}}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) \in \mathcal{V}_u \times \mathcal{V}_t \times \mathcal{V}_c
\end{aligned} \tag{2.35}$$

using the bilinear forms introduced in 2.18 and the following:

$$\mathbf{B}_{\Delta uu}(\boldsymbol{\Delta u}, \tilde{\mathbf{u}}) = -\int_{S_c}\int_{S_t} (R\Delta u)_{iq}(\mathbf{x}) B_{ikqs}(\mathbf{r}) (R\tilde{u})_{ks}(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.36}$$

$$\mathbf{B}_{u\Delta u}(\mathbf{u}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) = -\int_{S_t}\int_{S_c} (Ru)_{iq}(\mathbf{x}) B_{ikqs}(\mathbf{r}) (R\Delta\tilde{u})_{ks}(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.37}$$

$$\mathbf{B}_{t\Delta u}(\mathbf{t}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) = \int_{S_u}\int_{S_c} \mathbf{t}_k(\mathbf{x}) T_i^k(\mathbf{x}, \tilde{\mathbf{x}}) \boldsymbol{\Delta}\tilde{\mathbf{u}}_i(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.38}$$

$$\mathbf{B}_{\Delta ut}(\boldsymbol{\Delta u}, \tilde{\mathbf{t}}) = \int_{S_u}\int_{S_c} \tilde{\mathbf{t}}_k(\tilde{\mathbf{x}}) T_i^k(\tilde{\mathbf{x}}, \mathbf{x}) \boldsymbol{\Delta u}_i(\mathbf{x}) dS_x dS_{\tilde{x}} \tag{2.39}$$

$$\mathbf{B}_{\Delta u\Delta u}(\boldsymbol{\Delta u}, \boldsymbol{\Delta}\tilde{\mathbf{u}}) = \int_{S_c}\int_{S_c} (R\Delta u)_{iq}(\mathbf{x}) B_{ikqs}(\mathbf{r}) (R\Delta\tilde{u})_{ks}(\tilde{\mathbf{x}}) dS_{\tilde{x}} dS_x \tag{2.40}$$

the linear form is given by:

$$\mathbf{F}_{\Delta u}(\tilde{\mathbf{u}}) = \int_{S_c} p_k(\mathbf{x})\Delta\tilde{u}_k(\mathbf{x})dS_x + \int_{S_u}\int_{S_c} (Ru^D)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\Delta\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x - \int_{S_t}\int_{S_c} t_k^D(\mathbf{x})T_i^k(\mathbf{x},\tilde{\mathbf{x}})\Delta\tilde{u}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{2.41}$$

In (2.35), the spaces of admissible boundary traces are $\mathcal{V}_u$ and $\mathcal{V}_t$ (defined by (2.26)), and $\mathcal{V}_c = H_0^{1/2}(S_c)$. Natural finite-dimensional subspaces of $\mathcal{V}_c$ for Galerkin discretization then consist of continuous interpolations of $\Delta\mathbf{u}$ over S_c with a zero trace on the crack front ∂S_c, with again no requirement on the derivatives of $\Delta\mathbf{u}$.

2.3 Boundary Element Analysis

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

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 $E_1, E_2, .., E_N$

$$\partial\Omega \simeq \bigcup_{e=1}^{N_e} S_e \tag{2.42}$$

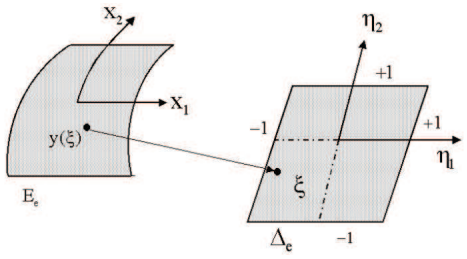


Figure 2.7: Boundary element E_e and referent element Δ_e

One of the most convenient ways of having the necessary approximations is using *isoparametric* method, in which the boundary and boundary functions are represented through the same set of shape functions defined on a parameters space. The discretization procedure can be briefly summarized as follow:

A generic field **f** (i.e. geometry, displacements or tractions) can be approximated in the point **x** over the elements E_e:

$$f_j(\mathbf{y}) = \sum_{\alpha=1}^{N_n} \Phi_\alpha^e(\xi) f_j^{e\alpha} \tag{2.43}$$

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 bilinear 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 present) are usually meshed with Q4-elements while the cracks are all modeled by modified Q8-elements.

2.3.2 Galerkin approximation

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 (2.36). The evaluation of the double boundary integrals represents a crucial aspect in SGBEM. The generic double surface integral equation takes the following form:

$$I(S_e, S_f) = \int_{S_e} \int_{S_f} f(\mathbf{x}) K(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) dS_\mathbf{y} dS_\mathbf{x} \tag{2.44}$$

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, there will occur the singularity when these two elements are coincident, adjacent by edge or adjacent by vertex. For a 3D problems, there are thus four possible configurations (Fig.2.8):

(i) Coincident: two identical elements

(ii) Edge-adjacent: two elements share one common edge (E_5 with E_2,E_4,E_6 and E_8)

(iii) Vertex-adjacent: two elements share one common vertex (E_5 with E_1,E_3,E_7 and E_9)

(iv) Disjoint: two well-separated elements (Eg. $E_1 - E_9$, $E_3 - E_7$ etc...)

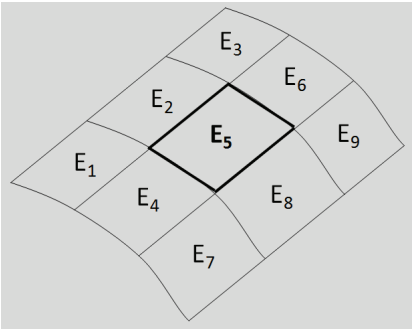


Figure 2.8: Different elements interactions

Case (i), (ii), (iii) lead to singular integrals, while case (iv) is non singular. The singular integrals are evaluated using special schemes described in [6]. The regular integrals can be evaluated with normal quadrature rule. The integral of a pair of elements can be written as:

$$I(S_e, S_f) = \int_{\Delta_e} \int_{\Delta_f} f(\mathbf{x}(\boldsymbol{\eta}))K(\mathbf{x}(\boldsymbol{\eta}), \mathbf{y}(\boldsymbol{\xi}))g(\mathbf{y}(\boldsymbol{\xi}))J_y(\boldsymbol{\xi})J_x(\boldsymbol{\eta})d\boldsymbol{\xi}d\boldsymbol{\eta} \tag{2.45}$$

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

$$I(S_e, S_f) \simeq \sum_{i=1}^{N_{pge}} \sum_{j=1}^{N_{pgf}} f(\boldsymbol{\eta}_i)K(\boldsymbol{\eta}_i, \boldsymbol{\xi}_j)g(\boldsymbol{\xi}_j)J_y(\boldsymbol{\xi}_j)J_x(\boldsymbol{\eta}_i)A^j_{\boldsymbol{\xi}_j}A^i_{\boldsymbol{\eta}_i} \tag{2.46}$$

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 element respectively.

2.3.3 System solution and limitations

The discretized equations system of the SGBEM can be written in matrix form: $[K]\{x\} = \{b\}$, $[K]$ is the influence matrix, the terms in $[K]$ are derived from (2.18) or (2.36). Vector $\{x\}$ regroups all the unknowns on the boundaries of the problem: $\mathbf{u}$ on S_t, $\mathbf{t}$ on S_u and $\boldsymbol{\Delta}\mathbf{u}$ on S_c. The right-hand side vector $\{b\}$ contains the known values (2.22) or (2.41) of the system.

The BEM/SGBEM method usually leads to a algebraic system of equations with less unknowns than that produced by 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 restrains 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.

Iterative Solvers and Preconditioned System

In order to solve the linear equations 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) [9] has been the most used iterative solver for BE calculation. A detail description of GMRES in BEM is shown in [67]. 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 tolerane. 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 $[K]\{v\}$ is evaluated by means of conventional SGBEM. This is already a major improvement in comparison with direct solvers.
Nevertheless, the convergence rate of an iterative solver depends strongly on spectral properties of the matrix which eventually leads to the use of a preconditioner. In numerical analysis, a preconditioner is a matrix such that when applied to the original system, it helps decreasing the *condition number* of the coefficient matrix. Such technique is called preconditioning. Let us consider a simple linear system:

$$[K]\{x\} = \{b\} \tag{2.47}$$

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

$$[P]^{-1}[K]\{x\} = [P]^{-1}\{b\} \tag{2.48}$$

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 it requires only one iteration for convergence. However, applying the preconditioning $[P]^{-1}$ is as difficult as solving the original system thus gaining none in terms of operation. Therefore, depending on different situations, one has to choose $[P]$

somewhere between these two extremes in order to balance the cost of constructing and inversing matrix $[P]$ with the overall solution.

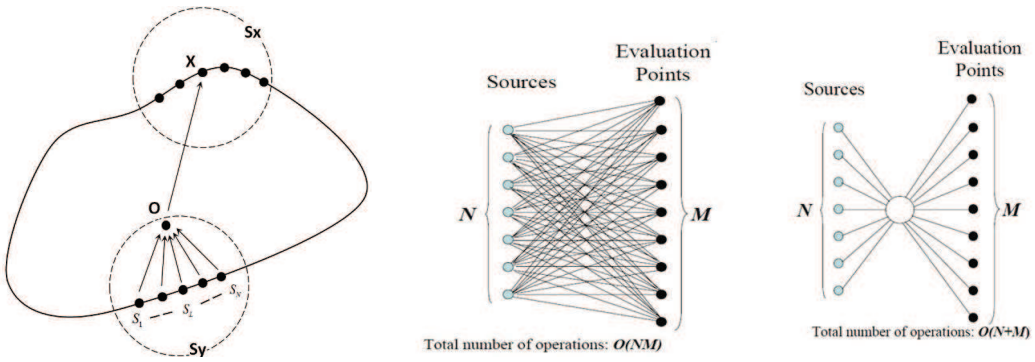
In summary, the bottlenecks of the method reside in the memory constraint and also in the evaluation of matrix-vector product which is required at each step of the iterative solver. Because the coefficient matrix is full, the cost of applying these operations becomes excessive even when the problem size 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 must be fast and that the explicite storage of the matrix $[K]$ should be avoided.

2.4 Fast Multipole Method

As mentioned in the previous sections, due to the fact that the coefficient matrix of SGBEM is fully-populated, the build-up phase and operation counts unfortunately lead to a rapid exhaustion for a standard computer. This obstacle makes it impossible to apply the method into treating realistic problems which normally contains a considerable amount of unknowns. The Fast Multipole Method (FMM) can, fortunately, change completely this circumstance. Introduced first by by Rohklin [33] and Greengard, 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, Rohklin 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 far-away groups can be translated into the coefficient of the local expansion associated with a local group. It has been proven that the FMM when combine 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 [41, 42], acoustics [34, 45], electromagnetics [40], elastodynamics [38, 39, 44] (read [59–63] for more references). The outcomes of these studies show great promises in dealing with large-scale engineering problems by boundary approach. Some recent successful works have also been reported in composite materials [64, 65] and in electromagnetic wave scattering [66]. Our work inherits the developments of the FMM-SGBEM in elastostatics and fracture mechanics that are introduced in [3], [5], [1].

2.4.1 Basic concept of the FMM

The Fast Multipole Method is based on a reformulation of the fundamental solutions into series of product of functions of $\mathbf{x}$ and $\mathbf{y}$. This technique allows one to re-use the integrations with respect to $\mathbf{y}$ when the observation point $\mathbf{x}$ is changed (Fig.2.9a).



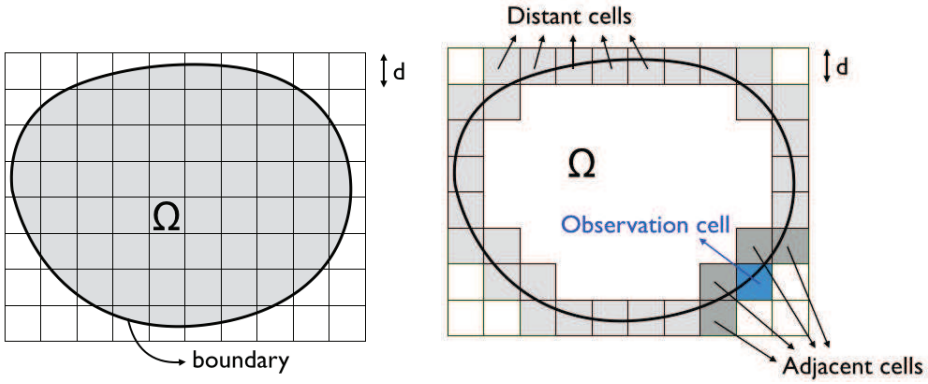
(a) (b)

Figure 2.9: (a) Simple illustration of the Fast Multipole Method on a generic boundary (b) Standard algorithm (left) and Fast algorithm (right)

The principle of the FMM can be illustrated as such: We need to compute the interaction between 2 groups of points $\mathbf{x}$ and $\mathbf{y}$ (respectively on S_x and S_y). Supposing that we have n points on S_x and m points on S_y, we should therefore need mn operations by conventional approach. The FMM, on the other hand, uses the point $\mathbf{O}$ to represent S_y, the contributions from 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 integrations of SGBEM (which is also equal to the matrix-vector multiplication). This improves greatly the performance of the overall system solution. The figure (2.9b) shows the $O(NM)$ complexity if standard evaluations of double integration are called, while with FMM, the operations count is reduced to $O(N + M)$.

Single-level Fast Multipole

The first and simplest variant of the FMM, 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.2.10a). Only cells containing boundary elements are taken into consideration. The center of a cell plays the role of 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, othe other hand, considered when 2 cells are adjacent. The evaluation of the boundary integral is then composed of the traditional SGBEM and the quick computation of Fast algorithm (Fig.2.10b). Compared with the classical SGBEM, the single-level FMM is more efficient with a complexity of $O(N^{4/3})$. However, more efficient scheme can be achieved by adopting the multi-level FMM which is described more in details in the next section.



(a) (b)

Figure 2.10: (a) 2D grid (spacing d) occupying the domain Ω (b) Cells interaction in the single-level FMM configuration

2.4.2 Multi-level Fast Multipole Formulation

In order to obtain maximal efficiency, the amount of traditional SGBEM calculation should be minimal while clustering the most possible the distant groups. The Fast Multipole algorithm must therefore be applied in an hierarchical manner (in a Multi-level approach). This is done with help of an *oct-tree* structure (See Fig.2.11): At the first step ($level = 0$ or *roof*), a cube which contains the whole studied boundary $\partial\Omega$ 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.

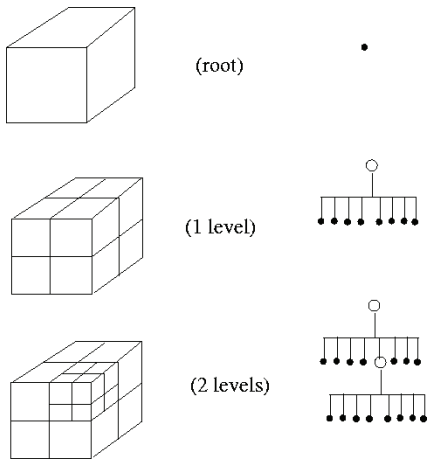


Figure 2.11: Oct-tree structure

We now give some notions that are used in the Fast Multipole algorithm:

- ***cell*** - Being an unit of octreee structure, *Cell* takes form of a square in 2D and a cube in 3D. They are divided in an hierarchical manner and all contain 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 maximal 4 children in 2D and 8 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 2 levels (such that *far-away interactions* exist).

- ***adjacent*** - 2 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*** - 2 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 $l-1$. 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 a *leaf* 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 purpose, the introduction of the FMM algorithm is done 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 [1]). Considering the symmetry of the Kernel function U_i^k, $\mathbf{B}_{tt}$ can be written as:

$$\mathbf{B}_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \int_{S_u}\int_{S_u} t_k(\mathbf{x})U_i^k(\mathbf{x},\tilde{\mathbf{x}})\tilde{t}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{2.49}$$
$$= \int_{S_u}\int_{S_u} \tilde{t}_i(\mathbf{x})U_k^i(\mathbf{x},\tilde{\mathbf{x}})t_k(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x$$
$$\mathbf{B}_{tt}(\mathbf{t}^D,\tilde{\mathbf{t}}) = \int_{S_u}\int_{S_u} t_k^D(\mathbf{x})U_i^k(\mathbf{x},\tilde{\mathbf{x}})\tilde{t}_i(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \tag{2.50}$$
$$= \int_{S_u}\int_{S_u} \tilde{t}_i(\mathbf{x})U_k^i(\mathbf{x},\tilde{\mathbf{x}})t_k^D(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x$$

The multipole expansion of r^{-1} in the Kelvin solution U_i^k is given in [4] by:

$$\frac{1}{r} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^n R_{nm}(\tilde{x}')\sum_{n'=0}^{n}\sum_{m'=-n'}^{n'}\overline{S_{n+n',m+m'}(\mathbf{r}_0)}R_{n'm'}(\mathbf{x}') \tag{2.51}$$

where

$$R_{nm}(\mathbf{y}) = \frac{1}{(n+m)!}P_n^m(cos\alpha)e^{im\beta}\rho^n$$
$$S_{nm}(\mathbf{y}) = (n-m)!P_n^m(cos\alpha)e^{im\beta}\rho^{-(n+1)} \tag{2.52}$$

(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 [4] (brief description in Annex). Figure 2.12 demonstrates the principle of the FMM as one computes the interaction of two surface S_x and $S_{\tilde{x}}$:

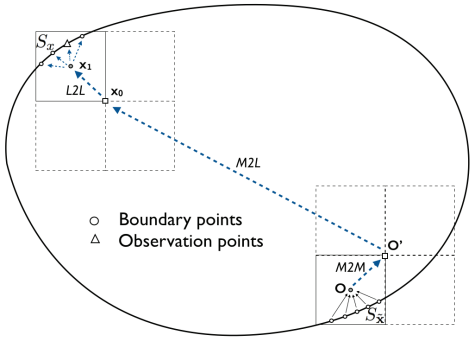


Figure 2.12: Decomposition of the position vector

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.2.12). Hence, the interaction between 2 boundary portions S_x and $S_{\tilde{x}}$ is truncated into a number of steps:

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** is the first FMM operation being computed here.

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

$$U_k^i(\mathbf{x},\tilde{\mathbf{x}}) = \frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\overline{F_{ik,n,m}^{Stt}(\overrightarrow{Ox})} + \overline{G_{i,n,m}^{Stt}(\overrightarrow{Ox})}(\overrightarrow{O\tilde{x}}_k)\right)R_{n,m}(\overrightarrow{O\tilde{x}}) \tag{2.53}$$

where

$$\begin{aligned} F_{ik,n,m}^{Stt}(\overrightarrow{Ox}) &= \left(\frac{3-4\nu}{2(1-\nu)}\delta_{ik} - \frac{1}{21-\nu)}(\overrightarrow{Ox_k})\frac{\partial}{\partial x_i}\right)S_{n,m}(\overrightarrow{Ox}) \\ G_{i,n,m}^{Stt}(\overrightarrow{Ox}) &= \frac{1}{2(1-\nu)}\frac{\partial}{\partial x_i}S_{n,m}(\overrightarrow{Ox}) \end{aligned} \tag{2.54}$$

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

$$B_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_i(\tilde{\mathbf{x}})\left[\overline{F_{ik,n,m}^{S}(\overrightarrow{Ox})}M_{k,n,m}^{1tt}(O) + \overline{G_{i,n,m}^{S}(\overrightarrow{Ox})}M_{n,m}^{2tt}(O)\right]dS_x \tag{2.55}$$

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

$$\begin{aligned} M_{knm}^{1}(O) &= \int_{S_u} R_{nm}(\overrightarrow{O\tilde{x}})t_k(\tilde{\mathbf{x}})dS_{\tilde{x}} \\ M_{nm}^{2}(O) &= \int_{S_u} R_{nm}(\overrightarrow{O\tilde{x}})(\overrightarrow{O\tilde{x}})_k t_k(\tilde{\mathbf{x}})dS_{\tilde{x}} \end{aligned} \tag{2.56}$$

M2M translation

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

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

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

$$\overline{S_{n,m}}(\overrightarrow{Ox}) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n} R_{n',m'}(\overrightarrow{O'O})\overline{S_{n+n',m+m'}}(\overrightarrow{O'x}) \tag{2.58}$$

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

we can have:

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

Substituting (2.60) into (2.57) we obtain:

$$\begin{aligned} M^1_{knm}(O') &= \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{O'O})M^{1tt}_{k,n-n',m-m'}(O) \\ M^2_{nm}(O') &= \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{O'O}) \left[M^{2tt}_{n-n',m-m'}(O) - (\overrightarrow{OO'})_k M^{1tt}_{k,n-n',m-m'}(O)\right] \end{aligned} \tag{2.61}$$

M2L translation

In this step, the M2L operation translates the coefficients from pole $\mathbf{O'}$ to pole $\mathbf{x_0}$. From (2.58) we have:

$$\begin{aligned} \overline{S_{n,m}}(\overrightarrow{Ox}) &= \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{x_0}\mathbf{x}})\overline{S_{n+n',m+m'}}(\overrightarrow{\mathbf{x_0}O}) \\ &= (-1)^{n'} \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{x_0}\mathbf{x}})\overline{S_{n+n',m+m'}}(\overrightarrow{O\mathbf{x_0}}) \end{aligned} \tag{2.62}$$

Replacing (2.62) into (2.55) we get:

$$B_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \frac{1}{8\pi\mu} \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} \int_{S_u} \tilde{t}_i(\mathbf{x}) \left(F^{Rtt}_{ik,n',m'}(\overrightarrow{\mathbf{x_0}\mathbf{x}})L^{1tt}_{k,n',m'}(\mathbf{x_0}) + G^{Rtt}_{i,n',m'}(\overrightarrow{\mathbf{x_0}\mathbf{x}})L^{2tt}_{n',m'}(\mathbf{x_0})\right) dS_x \tag{2.63}$$

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:

$$L^{1tt}_{k,n,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n \overline{S_{n+n',m+m'}}(\overrightarrow{O\mathbf{x_0}}) M^{1tt}_{k,n',m'}(O) \tag{2.64}$$

$$L^{2tt}_{n,m}(\mathbf{x_0}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n \overline{S_{n+n',m+m'}}(\overrightarrow{O\mathbf{x_0}}) \left(M^{2tt}_{n',m'}(O) - (\overrightarrow{O\mathbf{x_0}})_k M^{1tt}_{k,n',m'}(O) \right) \tag{2.65}$$

and

$$F^{Rtt}_{ik,n,m}(\overrightarrow{\mathbf{x_0}\mathbf{x}}) = \left(\frac{3-4\nu}{2(1-\nu)}\delta_{ik} - \frac{1}{2(1-\nu)}(\overrightarrow{\mathbf{x_0}\mathbf{x}_k})\frac{\partial}{\partial \mathbf{x}_i} \right) R_{n,m}(\overrightarrow{\mathbf{x_0}\mathbf{x}}) \tag{2.66}$$

$$G^{Rtt}_{i,n,m}(\overrightarrow{\mathbf{x_0}\mathbf{x}}) = \frac{1}{2(1-\nu)}\frac{\partial}{\partial \mathbf{x}_i} R_{n,m}(\overrightarrow{\mathbf{x_0}\mathbf{x}}) \tag{2.67}$$

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 (2.59) we have:

$$R_{n,m}(\overrightarrow{\mathbf{x_0}\mathbf{x}}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}}) \tag{2.68}$$

Substituting (2.68) into (2.63) we obtain:

$$B_{tt}(\mathbf{t},\tilde{\mathbf{t}}) = \frac{1}{8\pi\mu}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} \int_{S_u} \tilde{t}(\mathbf{x}) \left(F^{Rtt}_{ik,n',m'}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) L^{1tt}_{k,n',m'}(\mathbf{x_1}) + G^{Rtt}_{i,n',m'}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) L^{2tt}_{n',m'}(\mathbf{x_1}) \right) dS_x \tag{2.69}$$

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:

$$L^{1tt}_{k,n,m}(\mathbf{x_1}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}}) L^{1tt}_{k,n',m'}(\mathbf{x_0}) \tag{2.70}$$

$$L^{2tt}_{n,m}(\mathbf{x_1}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n R_{n-n',m-m'}(\overrightarrow{\mathbf{x_0}\mathbf{x_1}}) \left(L^{2tt}_{n',m'}(\mathbf{x_0}) - (\overrightarrow{\mathbf{x_0}\mathbf{x_1}})_k L^{1tt}_{k,n',m'}(\mathbf{x_0}) \right) \tag{2.71}$$

and

$$F^{Rtt}_{ik,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) = \left(\frac{3-4\nu}{2(1-\nu)}\delta_{ik} - \frac{1}{2(1-\nu)}(\overrightarrow{\mathbf{x_1}\mathbf{x_k}})\frac{\partial}{\partial \mathbf{x}_i} \right) R_{n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) \tag{2.72}$$

$$G^{Rtt}_{i,n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) = \frac{1}{2(1-\nu)}\frac{\partial}{\partial \mathbf{x}_i} R_{n,m}(\overrightarrow{\mathbf{x_1}\mathbf{x}}) \tag{2.73}$$

Replacing all the coefficients of local expansion in the formula of B_{tt}, we finally obtain the integral equation evaluated. Fig.2.13 summarizes and simplifies these Fast multipole operations in a tree-like representation:

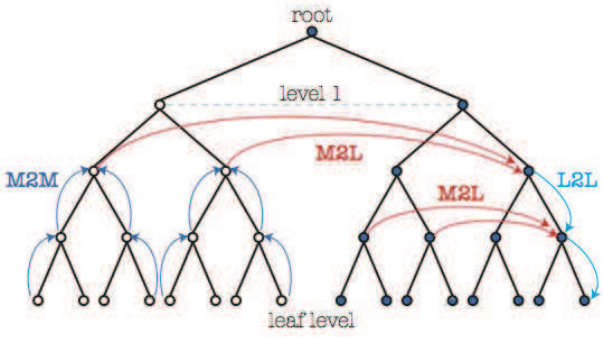


Figure 2.13: Multi-level Fast multipole operations

2.4.3 Multi-level Fast Multipole Algorithm

We introduce in this section the algorithm of the Fast Multipole method. For simplicity purpose, some figures and some explanations are in 2D. The extension to 3D follows similar principle.

Step 1 - Discretization

The boundary of the problem is discretized in the same approach as the conventional boundary elements analysis.

Step 2 - Construction of Octree structure

The dimensions of the $level-0$ cubic octree structure are taken from the minimal and maximal coordinates (in respectively 3 directions) of the nodes.

Step 3 - Upward pass

At the lowest level (*leaf*) and recursively aggregated by moving upward, the multipole moments are computed. Here, the contribution of the elements in a *leaf* is transferred to its center then it will be shifted to the center of the father cell as we move to the higher level (by M2M operation illustrated in Fig.2.14). The process stops when level 2 is reached (the highest level that features non-adjacent cells).

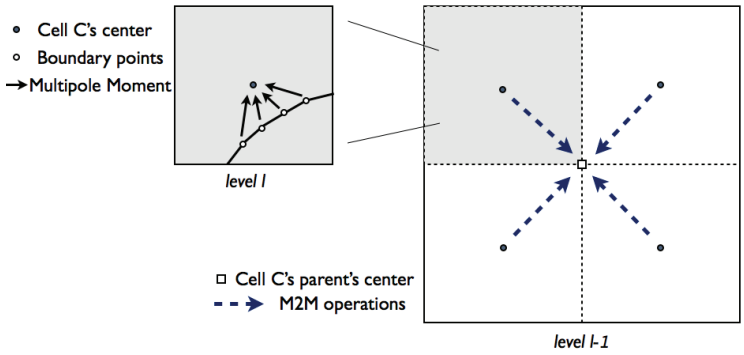


Figure 2.14: Computing Multipole moments at cell C (being a *leaf* at level l) and transferring to the center of Cell C's parent at level $l-1$

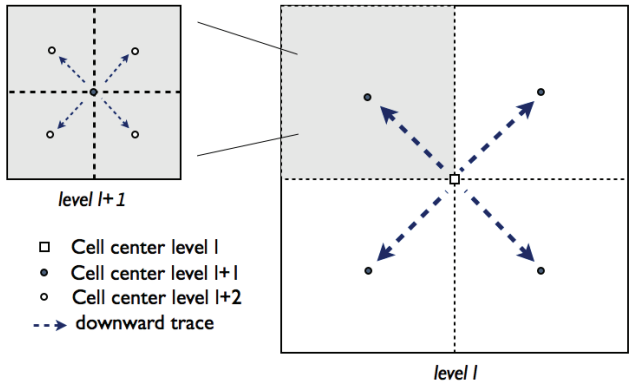


Figure 2.15: Local expansions from level l to level $l+1$

Step 4 - Downward pass

The procedure is now oriented from level 2 down to the *leaf*-level. The local expansions are computed at level 2 then evaluated at selected lower-level cells by tracing down the octree structure (see Fig.2.15).

This step translates the contribution of far-away cells toward the center of the considered *leaf*-cell C using M2L and L2L operations. Assuming the *leaf* cell C is at level l, the distant influence toward its center is composed of 2 parts: The contributions at level $l-1$ of distant cells toward cell C's parent, the contribution at level l of cells in the interaction list of C.

This can be illustrated as follow: (Fig.2.16-Left) - distant contribution toward cell C's father via M2L translation (level $l-1$) and L2L operation translates these influences from cell C's parent toward cell C's center. (Fig.2.16-Right) - distant contribution toward cell C from cells in the interaction list of C at level l is transferred to C's center via M2L operation.

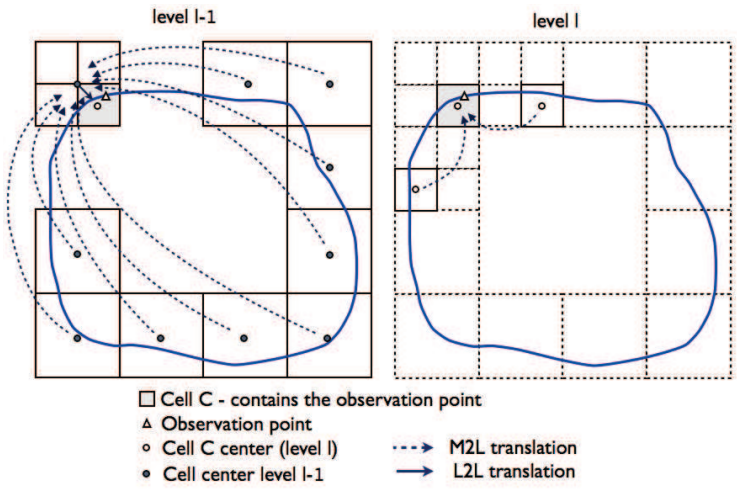


Figure 2.16: Distant influence toward cell C which contains the observation point: M2L and L2L translations at level $l-1$ (left) and M2L translation at level l (right)

Step 5 - Local expansion and Direct calculation

The **Local expansion** is evaluated when cell C is a *leaf*. The contribution of the far field is now located at the center of cell C, it is then shifted to the observation points via the coefficients of the local expansion. We also perform the **Direct calculation** either if cell C is a *leaf* or one adjacent cell of C is a *leaf*. For this calculation, all remaining double element integrals (corresponding to pairs of cells that are not well-separated at any level) are evaluated using conventional (singular or non-singular) integration methods. All singular integrals are, in particular, handled in this step.

By combining this near calculation with faraway contribution (Fig.2.17), we get the integral equations evaluated at all the elements of the boundary.

There are two alternatives to work with the near contributions: (a) assembling all near-field coefficients into a sparse matrix, namely $[Knear]$ then call it when needed or (b) Recomputing at each GMRES iteration the near-field contribution to the matrix-vector multiplication without assembling the matrix $[Knear]$. The option (a) has been chosen in this work.

2.4.4 Numerical aspects of a Fast Multipole Algorithm

2.4.4.1 Components of a Cell in the Octree structure

The octree structure consists practically of cells, each cell must therefore contain necessary information so that the appropriate routine can be called during the computation. It can be expressed as following:

$$Octree(Cell\_Name)\%Info\_Name$$

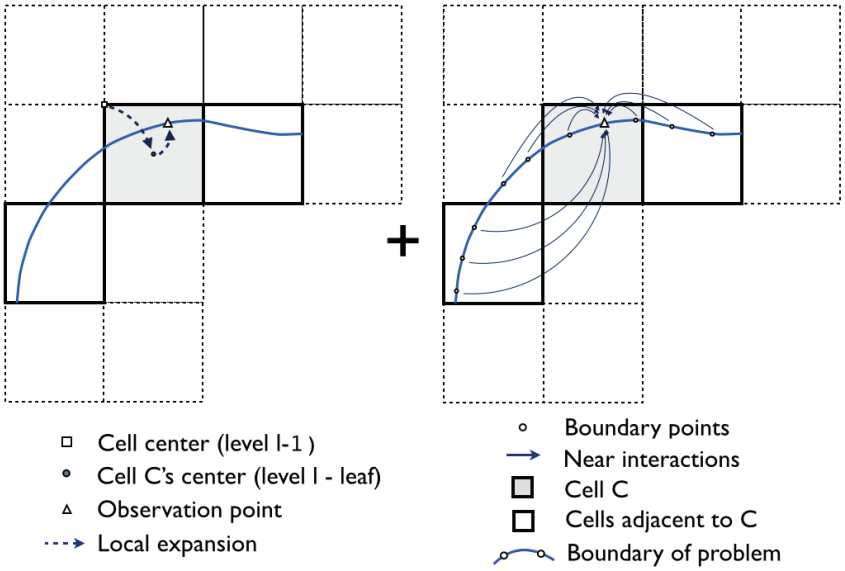


Figure 2.17: Total influence of the boundary toward the observation point = far away interaction (left) + near interactions (right)

where *Octree* is a pointer type that contains many arrays in Fortran code, *Cell_Name* is the name of the observed Cell (namely **cell C**), $Info\_Name$ is the information that is taken into account. These information can be divided in 2 categories. The first one carries geometrical properties of cell C and is described as follow:

- *Location(3)* - center of cell C
- *Dimension(3)* - dimension of cell C
- *Nchildr* - number of children of cell C
- *Adjlist(:)* - list of cells that are adjacent with cell C at the same level
- *Nelems* - number of elements in cell C
- *Elems(:)* - name of the elements in cell C

the second one is addressed with the storage of the *near interaction*:

- *Nnull* - number of non-null terms computed in the *near interactions*
- *Icol* - number of degrees of freedom contained in cell C
- *Coc(:) - list of degrees of freedom in cell C*
- *AAC(:), JAC(:), IAC(:)* - arrays contain the matrix $[Knear]$ of unknowns contained in cell C

2.4.4.2 Storage of matrix [Knear]

According to the algorithm of the FMM, the entire global matrix is not required to be explicitly constructed and stored. The only matrix that needs to be fully or partly built is the matrix of near-interaction $[Knear]_{global}$. Since the problem's geometry is divided and managed in many levels and cells by the *octree* structure, the construction and storage of the $[Knear]_{global}$ should follow the same manner. In this case, matrix $[Knear]_{global}$ is divided in many sub-matrices of the corresponding cells. As mentioned above, the near interaction is computed either if cell C is a *leaf* or cell C is not a *leaf* but an adjacent cell of C is a *leaf*. Therefore, if the studied cell C satisfies this condition, C will have one part of $[Knear]_{global}$, namely $[Knear]_c$, stored:

Loop over the octree structure

```
Do c=2,ncells
   If (C = leaf) Then
      Compute [Knear]_c
   ElseIf (C ≠ leaf but C's adjacent = leaf) Then
      Compute [Knear]_c
   Else
      [Knear]_c = ∅
EndDo
```

Fig.2.18 illustrates a small portion of the *octree* structure where it is easy to notice the storage of $[Knear]$: cells 4,5,6,7,11,12,13 are *leaf* thus they contain the coefficients of near interactions. Cell 10 is not a *leaf* but a neighboring cell 11 is a *leaf* so it still has matrix $[Knear]$ stored. Cells 1,2,3,9 do not match the condition so they are void.

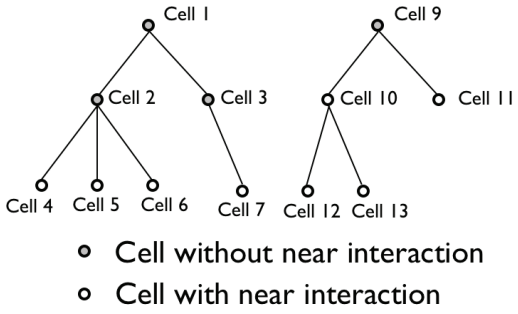


Figure 2.18: A portion of octree structure and its storage of $[Knear]$

Because there is no need to be explicitly assembled, the matrix $[Knear]_{global}$ has only a 'representative' appearance. The role of the global matrix of near interaction is replaced by the union of its sub-matrices:

$$[Knear] = \sum_{c=1}^{ncells} [Knear]_c \tag{2.74}$$

The partitioning of $[Knear]$ induces that all the latter matrix-vector or matrix-matrix operations involving $[Knear]$ be modified. This is a simple task based on the principle of superposition. The use and role of $[Knear]$ is, however, an important issue in a Fast multipole boundary analysis. We will discuss more about this aspect in the following sections of this thesis.

2.4.4.3 Memory complexity

The sub-matrices $[Knear]_c$ $(c = 1, ncells)$ are the only matrices that need to be explicitly stored. They possess similar spectral properties as the matrix $[Knear]$ and are very sparse (most of the terms are zero). The sparsity of these matrices is usually high (around 90% or even more) so that the memory requirement of the method is reduced to an order much lower than N^2. One can either store this matrix in a standard manner or as lists of indexes and non-zero values: if the storage is more important than access speed, it may be preferable to use the second option. On the contrary, the standard storage can take advantage of the optimized basic linear algebra subroutines (BLAS), thus renders the matrix operations instantaneous. However, for some particular sparsity structures, it may be possible to define their specific and efficient linear system algorithms (Eg. a diagonal matrix requires only N terms to store and N operation for a matrix-vector product).

2.4.4.4 Matrix-vector multiplication

The product between matrix $[K]$ and a vector $\{x\}$ is now composed of 2 components:

$$[K]\{x\} = [Knear]\{x\} + [K_{FMM}]\{x\}$$

Matrix $[Knear]$ (or to be more precise, the summation of $[Knear]_c$ with $c = 1, ncells$) has been computed and stored in RAM, it is now called onto this process to perform the matrix-vector product. The second part of the summation is carried out every time the iterative solver updates the candidate vector. This is where the computation is sped up: while the near part is of order $O(N^2)$ as the conventional approach is employed, the far-away interaction's complexity is of order N by re-using the groups of distant points. Because the proportion of far contribution in this summation is normally dominant in comparison with the near interactions, the complexity of the matrix-vector multiplication is effectively reduced to N (N being the problem dimension). This results in a faster solution of the BE analysis.

2.4.4.5 Overall computation complexity

The Fast Multipole algorithm integrates perfectly in the iterative solution of a boundary elements problem because the iterative solver requires specifically a

matrix-vector multiplication. For an elastostatics problem, the Fast multipole process exhibits linear dependence versus the problem size. Therefore the complexity of this process becomes $N_{iter} \times N$ (where N_{iter} denotes the number of iterations). Hence, the efficiency of the FMM becomes more important as the problem size N grows. Additionally, the memory requirements is also negated because the coefficient matrix is not obligated to be explicitly stored. Consequently, all the bottlenecks of the traditional BEM can be avoided and the combined method becomes computationally very efficient. Fast multipole boundary elements has therefore become a powerful numerical tool capable of treating a wide range of realistic problems with moderate computational resources.

2.5 Numerical implementation of the FM-SGBEM

The numerical code has been written and developed in Fortran and run on a single-processor computer (RAM: 48Gb, CPU: 3Ghz). The multi-level Fast Multipole scheme has been used in all the examples. The FM-SGBEM program consists of around 80 subroutines which feature about 15.000 code lines. The structure of the program as well as the utility of each subroutine is detailed in Annex C. We introduce here the most simple representation of the program in Fig.2.19.

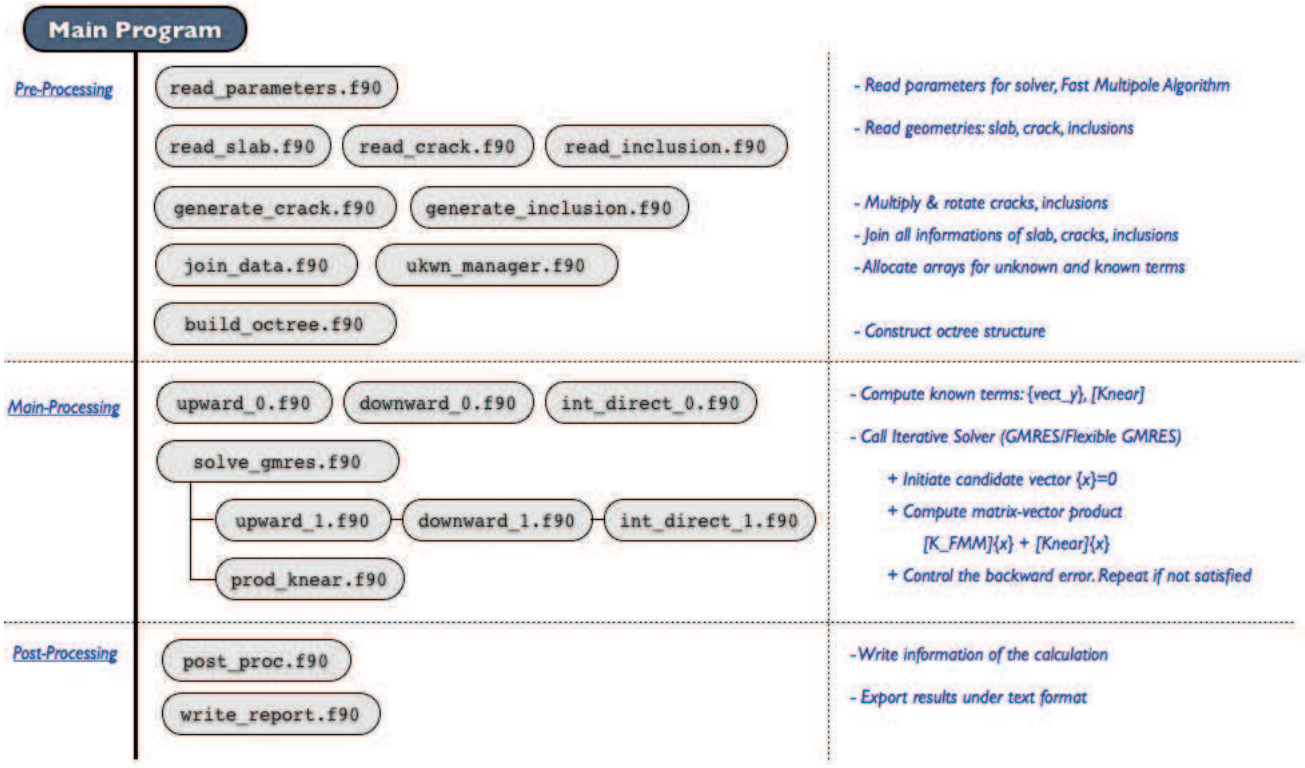


Figure 2.19: Generic FM-SGBEM program in Fortran

The appropriate element type, the number of Gaussian points and other various

input parameters have been chosen to optimize the code's performance. The meshes are generated by the commercial program **GiD**, the personal pre and post-processor. The results with less than 10^6 unknowns are visualized by **Medit**. Source codes of matrix-vector operations are taken from the BLAS libraries. Normally, the cost of the pre-processing and post-processing are insignificant in comparison with the main-processing. For this reason, later on in the thesis, many comparisons are related to only different parts of the main phase and the total cost of this phase can represent the whole . For instance: the main processing is composed of 2 phases: preparation phase (to compute $[Knear]$ and $\{vect\_y\}$) and iterative solution phase (GMRES or Flexible GMRES).

2.6 Conclusions

In this chapter, we have introduced the basic foundations of the integral equations method as well as the regularized version of the symmetric Galerkin approach in two contexts: elasticity and fracture mechanics. The limitations of the method in the numerical solution phase have been identified which lead to the application of the Fast Multipole Method. The fundamental principle and algorithm of the method are thus described and presented. In order to demonstrate the superiority of the FMM to the conventional SGBEM, some aspects related to the numerical solution and complexity of the Fast algorithm have been discussed. Finally, a generic program to solve the problems in elastostatics/fracture mechanics by the FM-SGBEM is given. In the next chapter, we will discuss about further refinements for the algorithm.

Chapter 3

Preconditioning and refinements of the FM-SGBEM

Contents

The aim of many studies over recent decades is to improve the performance of the boundary elements analysis and to make it an efficient and powerful numerical tool for solving realistic problems in engineering. The appearance of the Fast Multipole Method in the boundary algorithm has effectively overcome all the usual bottlenecks and has made the coupled fast method a formidable option along with the infamous finite elements method. However, it is still not simple to simulate efficiently large-scale models on moderates computational resources. Many developments have therefore been devoted for further efficiency improvements. In our work, two strategies of optimization have been adopted to reduce the storage constraint and to reduce the iterative resolution time. Appropriate choices on input parameters have also been analyzed. Lastly, some numerical validation tests in the context of elasticity and fracture mechanics have been carried out. The performance enhancements on large-scale problems ($N \geq 10^6$) are well captured and reported.

3.1 Storage of the Matrix of near interactions

.

In the FM-SGBEM algorithm, the global matrix does not need to be fully constructed. The evaluation of the double integrals (as known as the matrix-vector product) is divided in two parts: (1) Fast computation by FMM and (2) Standard

computation by traditional SGBEM. While the fast computation is repeated at each iteration, the (slow) near-interactions should be computed and stored before the resolution. As mentioned in the previous sections, the choice of storing the matrix $[Knears]_{global}$ depends on how one uses and access to it:

(i) *Standard storage*: the matrices $[Knear]_c$ $(c = 2, ncells)$ are stored in RAM using the standard matrix format which is composed of n rows and n columns. For this storage, we lose in term of memory but in return can take advantage of the optimized BLAS routines to compute instantaneously the matrix-matrix or matrix-vector operations. Providing that a standard matrix of 32.000^2 takes already 1Gb of RAM, the appropriate number of unknowns can be treated should not surpass 10^4.

(ii) *Out-of-core storage*: $[Knear]_c$ $(c = 2, ncells)$ can also be saved in the hard disk. There is thus no limitation on how big the matrix is but whenever one wants to access to the matrices, a routine should be called to read the matrix's coefficients entailing considerably long processes. This procedure is the main reason that slows down significantly the performance of this approach and renders it unusable.

(iii) *Compressed storage*: This strategy of storage takes advantage of the spectral properties of the matrix in question by converting it into arrays on non-zero terms. By doing so, the amount of terms transferred to RAM is reduced to minimal while keeping the quick access to the matrix. Hence, this approach is well suited for treating large-scale problems.

3.1.1 Symmetric Compressed Sparse Row

Symmetric Compressed Sparse Row (CSRSYM) [30] is a simple algorithm to store symmetric and sparse matrices. This algorithm takes only 3 vectors to store all the necessary information of the matrix in question: The first vector ($\boldsymbol{AA}$) collects the value of the non-zeros on the upper part of the matrix (including the diagonal). The second vector ($\boldsymbol{JA}$) saves the column index of the corresponding term while the third one ($\boldsymbol{IA}$) contains the information on the total number of non-zeros on each row of the matrix.

A simple example of the CSRSYM is shown below. Matrix $[K]$ is symmetric and contains many zero terms. The CSRSYM will scan the upper part of this matrix and extract all the non-zero coefficients for the storage.

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

$\downarrow$ can be converted to $\downarrow$

$\boldsymbol{AA}$	1	2	-3	-4	5	6	-7	8	9	10
$\boldsymbol{JA}$	1	2	5	2	4	3	4	4	5	5
$\boldsymbol{IA}$	1	4	6	8	10	11				

3.1.2 CSRSYM in FM-SGBEM

The usage of the CSRSYM in the context of FM-SGBEM comes naturally from the symmetry and sparsity of the sub-matrices $[Knear]_c$ ($c = 1, ncells$). These particular features can be explained from the nature of the FMM algorithm. At *leaf-level*, the standard double integration of the SGBEM is performed, producing symmetric and non-zero blocks of near interactions between cell C and its adjacent cells. On the other hand, no calculation between the cells in the adjacent list of C is conducted, leaving void blocks in the structure of $[Knear]_c$. In a 3D *octree*, due to the high number of adjacent cells (maximum 26), each sub-matrix is particularly very sparse (See Fig.3.1).

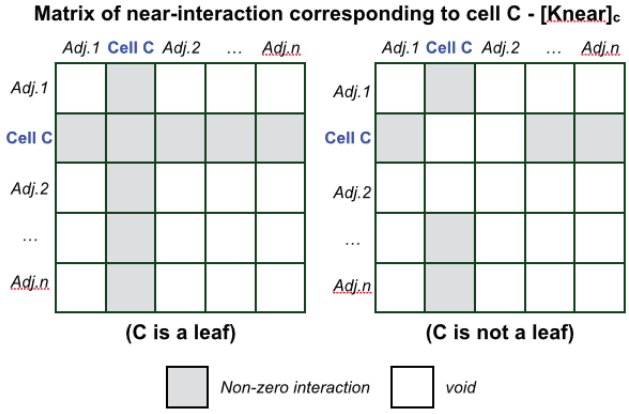


Figure 3.1: Block-wise constitution of $[Knear]_c$ for 2 cases: C is a *leaf* (left) and C is not a *leaf* but adjacent cell(s) of it is (right). There are n cells adjacent to cell C (in 3D, $n \leq 26$). When cell C is a *leaf* - interactions of C with itself and with all of it adjacent cells (being either *leaf* or not) are computed (left). When C is not a *leaf*, only interactions of *leaf*-adjacent cells with C are computed (Eg. cell *Adj.2* is not a *leaf* so no interaction between cell C and cell *Adj.2*).

Taking into account the clear advantages of the symmetry and sparsity, it is convenient to store all $[Knear]_c$ using the algorithm of CSRSYM. The following Algorithm 1 explains how to store a symmetric and sparse sub-matrix using CSRSYM.

Supplementary arrays should be needed if a sub-matrix is used in the global operations. $\boldsymbol{Coc}_c$ is one of these arrays. It relates the *local* positions in $[Knear]_c$ with the *global* positions:

$$\boldsymbol{Coc}_c \mid_{i=1,size[Knear]_c} = local\_index_i \leftrightarrow global\_index$$

Finally, the global positioning of a sub-matrix $[Knear]_c$ by CSRSYM can be illustrated in Fig.3.2 (Remark that the overlapping of the component matrices

Algorithm 1 Symmetric Compressed Sparse Row

```
for i = 1, n do
  for j = i, n do
    if [Knear]_ij ≠ 0 then
      - Store [Knear]_ij in vector AA
      - Store index j in vector JA
      - Count number of non-zero in row i and update in vector IA
    end if
  end for
end for
```

$[Knear]_c$ in the global scale is to illustrate the mutual interactions between pairs of elements that belong to different adjacent cells.):

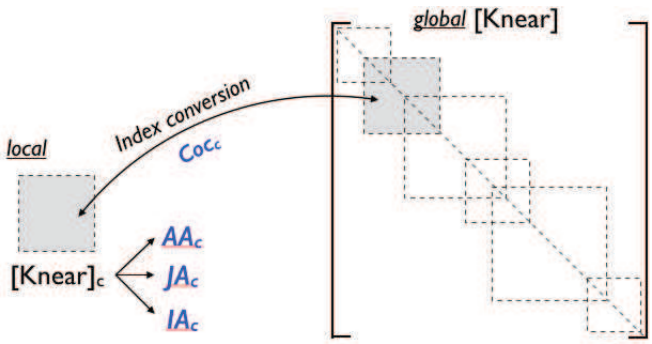


Figure 3.2: Representation of $[Knear]_{global}$ as a summation of all $[Knear]_c$

Memory Usage

Compared with the standard strategies, the CSRSYM is clearly superior since it uses the minimal space to store a symmetric sparse matrix. During numerical tests, the construction and storage of $[Knear]_{global}$ are observed. From problem size of 10^3 to 10^6, we notice that the proportion of *zeros* in the matrix can go up to 95%, hence using sparse compressed strategies is the best option. Besides, CSRSYM can take advantage of the symmetry feature of the method SGBEM so it takes roughly 50% less memory than the original CSR algorithm and also take less time in the matrix-vector multiplication.

Matrix-Vector multiplication

The matrix $[Knear]_{global}$ is never explicitly assembled in the algorithm of the FM-SGBEM. All the operations (Eg. matrix-vector product) inquiring $[Knear]_{global}$ are conducted via the union of the constitutive matrices $[Knear]_c$ $(c = 2, ncells)$. Since these sub-matrices are stored in an unpopular format which is the CSRSYM,

it is thus necessary to describe the specific algorithm of the matrix-vector product between $[Knear]_{global}$ and a given vector. Considering therefore a generic product of $[Knear]_{global}$ and $\{v\}$. We have:

$$[Knear]_{global}\{v\} = \sum_{c=2}^{ncells} [Knear]_c\{v_c\}$$

where $\{v_c\}$ is a part of vector $\{v\}$ which corresponds to the unknowns in $[Knear]_c$. The summation of all block-wise matrix-vector product $[Knear]_c\{v_c\}$ constitutes the required term. The algorithm of this operation can be expressed as in Algorithm 2.

Algorithm 2 Matrix-Vector Product of CSRSYM: $[Knear]_c\{v_c\}$

- A compressed sub-matrix $[Knear]_c$ will have 3 corresponding vectors: AA_c, JA_c and IA_c where the number of rows in $[Knear]_c$ is $nrow = IA_c - 1$
- The vector $\{v_c\}$ in the product should be of similar dimension ($nrow$)
- The matrix-vector multiplication is looped on every row:

for $row = 1, nrow$ **do**

- Compute the number of nonzero terms in row: $nnul_{row} = IA_{row+1} - IA_{row}$
- The value and column index of $nnul_{row}$ nonzero terms of this row are then taken from AA_c and JA_c
- Multiply this row with the corresponding part of $\{v_c\}_{row}$ to get the row value of the product
- Mirror the above operation to get the symmetric part
- Accumulate the result in the appropriate places of the product vector

end for

3.2 Preconditioning strategy

The second goal is to reduce the computational time of the method. In FM-SGBEM, there are two main procedures: (1) Build-up phase and (2) Iterative Solution phase. The first procedure consists on reading inputs, constructing *octree* structure and computing the conventional near interactions. This part is heavy but unavoidable. By only choosing an appropriate ratio of near and far contributions, we can rationally limit the computation times in this step (mentioned in [2]). In our work, we aim to accelerate the iterative solution phase by applying a more robust solver and a better preconditioner. In [1], GMRES left-preconditioned by block-diagonal of $[Knear]_{global}$ was used as a solver, it had a relatively good convergence rate, however it was still slower than our expectation and is only effective when the matrix has the diagonal dominance [20]. More complex preconditioners were also introduced such as Incomplete LU (ILU) factorization [21], sparse approximate inverse (SPAI) [5] [22]... Nevertheless, their construction cost is still high and can not be used in large-scale problems. In general, it is shown that a good precondi-

tioner is one that contains more information of the coefficient matrix. In [18], the symmetric successive over-relaxation (SSOR) preconditioner (which is based on the near-part of the matrix) is proposed. In [2], Chaillat mentioned that the use of a full $[Knear]_{global}$ as the preconditioner in Flexible GMRES could reduce greatly the number of iterations thus lower the computational time. Flexible GMRES is a variant of the GMRES algorithm which typically provides the use of a second solver during the preconditioning process. Basically, it 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. For example, the CGNR (or CGNE - conjugate gradient method applied to the normal equations) can be used as such. In [8], it is shown that the Flexible GMRES with a GMRES as an inner iterative solver can outperform the standard GMRES and is very easy to implement.

Inheriting from the previous studies, we aim to apply a strategy in which the matrix $[Knear]_{global}$ is employed as a preconditioner in the algorithm of the Flexible GMRES to enhance the solution phase of the FM-SGBEM. In the next sections, brief revisions of the GMRES and Flexible GMRES are presented, followed by the implementation of this strategy into our work and we show at last some numerical tests for validation purposes.

3.2.1 GMRES and Flexible GMRES

Before discussing about Flexible GMRES, it is important to recall the principle and algorithm of the original solver: the Generalized Minimal RESidual (GMRES). This is an iterative method developed by Y. Saad [9] and M. H. Schultz in 1986. GMRES is utilized for solving approximately a system of linear equations by a vector in a Krylov subspace with minimal residual.

Considering the linear equations system:

$$[A]\{x\} = \{b\} \tag{3.1}$$

and denoting the euclidean norm of a vector $\overrightarrow{v}$ by $\|v\|$. The matrix $[A]$ is assumed to be invertible. The n^{th} Krylov subspace for this problem is:

$$K_n = K_n(A, b) = span\{b, Ab, A^2b, ..., A^{n-1}b\}$$

GMRES approximates the exact solution of $[A]\{x\} = \{b\}$ by the vector $x_n \in K_n$ that minimizes the Euclidean norm of the residual $\| Ax_n - b \|$
The vectors $b, Ab, A^2b, ..., A^{n-1}b$ might be almost linearly independent, so instead of this basis, the Arnoldi iteration is used to find orthonormal vectors $q_1, q_2, ..., q_n$ which form a basis for K_n. Hence, the vector $x_n \in K_n$ can be written as $x_n = Q_n y_n$ with $y_n \in \mathbf{R}^3$, where Q_n is the $m - by - n$ matrix formed by $q_1, q_2, ..., q_n$.
The Arnoldi process also produces an $(n + 1) - by - n$ upper Hessenberg Matrix $\tilde{\mathbf{H}}_n$ with:

$$AQ_n = Q_{n+1}\tilde{H}_n$$

Because Q_n is orthogonal, we have:

$$\|Ax_n - b\| = \|\tilde{H}_n y_n - \beta e_1\|$$

where $e_1 = \{1, 0, 0, ..., 0\}$ is the first vector in the standard basis of $\mathbf{R}^{n+1}$ and $\beta = \|b - Ax_0\|$ - x_0 being the first trial vector (usually zero). Hence, x_n can be found by minimizing the Euclidean norm of the residual $r_n = \tilde{H}_n y_n - \beta e_1$ which is a linear least square problem of size n.
So, at every step of a GMRES iteration, these actions are performed:

1. Do one step of the Arnoldi method
2. Find the y_n which minimizes $\|r_n\|$
3. Compute $x_n = Q_n y_n$
4. Repeat if the residual is not yet small enough

The equation (3.1) is now right-preconditioned by matrix $[M]$ such that

$$[A][M]^{-1}([M]\{x\}) = \{b\} \tag{3.2}$$

The algorithm of GMRES for this modified system can be expressed like the following:

Algorithm 3 GMRES with right preconditioning

1. Start: Choose x_0 and a dimension m of the Krylov subspace. Define an $(m+1) \times m$ matrix $\tilde{H}_m$ and initialize all its entries $h_{i,j}$ to zero

2. Arnoldi process:

(a) Compute $r_0 = b - Ax_0$, $\beta = \|x_0\|$ and $v_1 = r_0/\beta$

(b) For $j = 1, ..., m$ do

- Compute $z_j := M^{-1}v_j$
- Compute $w := Az_j$
- For $i = 1, ..., j$ do
  $h_{i,j} := (w, v_i)$
  $w := w - h_{i,j}v_i$
- Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$

(c) Define $V_m := [v_1, ..., v_m]$

3. Form the approximate solution: Compute $x_m = x_0 + M^{-1}V_m y_m$ where $y_m = \text{argmin}_y \|\beta e_1 - \tilde{H}_m y_m\|_2$ and $e_1 = [1, 0, ..., 0]^T$

4. Restart: If satisfied, stop. Else set $x_0 \leftarrow x_m$ and go to 2

The Arnoldi loop constructs an orthogonal basis of the preconditioned Krylov subspace: $spab\{r_0, AM^{-1}r_0, ..., (AM^{-1})^{m-1}r_0\}$ by a modified Gram-Schmidt process, in which the new vector to be orthogonalized is obtained from the previous

vector of the process. The last step in the above algorithm forms the solution as a linear combination of the preconditioned vectors $z_i = M^{-1}v_i$ with $i = 1, ..., m$. Because these vectors are all obtained by applying the same preconditioning matrix M^{-1} to the $v's$, we need not save them. We only need to apply M^{-1} to the linear combination of the $v's$, for example to $V_m y_m$.

In the case we allow the change for the preconditioner at every step such that:

$$z_j = M_j^{-1} v_j$$

and we save these vectors to use them in updating x_m in step 3, we would obtain a 'modified' and 'flexible' version of the GMRES. Such algorithm is called *Flexible GMRES* and is shown below:

Algorithm 4 Flexible GMRES with variable preconditioning

1. Start: Choose x_0 and a dimension m of the Krylov subspace. Define an $(m+1) \times m$ matrix $\tilde{H}_m$ and initialize all its entries $h_{i,j}$ to zero

2. Arnoldi process:

(a) Compute $r_0 = b - Ax_0$, $\beta = \|x_0\|$ and $v_1 = r_0/\beta$

(b) For $j = 1, ..., m$ do

- Compute $z_j := M_j^{-1} v_j$
- Compute $w := Az_j$
- For $i = 1, ..., j$ do $\left| \begin{array}{l} h_{i,j} := (w, v_i) \\ w := w - h_{i,j} v_i \end{array} \right.$
- Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$

(c) Define $Z_m := [z_1, ..., z_m]$

3. Form the approximate solution: Compute $x_m = x_0 + M^{-1} Z_m y_m$ where $y_m = \text{argmin}_y \|\beta e_1 - \tilde{H}_m y_m\|_2$ and $e_1 = [1, 0, ..., 0]^T$

4. Restart: If satisfied, stop. Else set $x_0 \leftarrow x_m$ and go to 2

As the difference can be easily spotted in the step 2, the preconditioner matrix M can be varied from one iteration to another and that we now save the preconditioned vectors z_i and update the solution using these vectors. Besides, it should also be noted that neither preconditioned GMRES or Flexible GMRES requires explicit formation of $M^{-1}A$. As a result, the preconditioning task $z_j = M_j^{-1} v_j$ in Flexible GMRES can be solved by means of an iterative solver which differs from GMRES algorithm where M^{-1} needs to be explicit to compute $M^{-1} V_m y_m$. Consequently, more sophisticated matrices can be use as preconditioner for Flexible GMRES since the algorithm offers a possibility to *inverse* the preconditioner either if it is explicit or is in compact format.

3.2.2 Flexible GMRES in FM-SGBEM

The concept of using the Flexible GMRES preconditioned by $[Knear]$ fits naturally in the algorithm of FM-SGBEM because:

- $[Knear]$ is already computed and stored → No additional operations or space for the preconditioner

- $[Knear]$ possess similar spectral properties as the global matrix → Faster convergence

- $[Knear]^{-1}$ is not required explicitly in inner iterative solver → Simple task with low complexity and computational costs

Hence, we employ a simple algorithm of Flexible GMRES which is simply derived from GMRES in the FM-SGBEM. This Flexible GMRES utilizes another GMRES as an inner solver for the preconditioning task. So we basically have a solver which consists of 2 GMRES embedded in an inner-outer scheme: The outer GMRES solves the original system $[K]\{x\} = \{b\}$ and the inner GMRES performs the preconditioning of $[Knear]^{-1}$. The Algorithm of Flexible GMRES is described below in Algorithm 5:

Algorithm 5 Flexible GMRES in FM-SGBEM

1. **Build-up Process:** Compute known terms: vector $\{b\}$ and matrix $[Knear]$
2. **Iterative Solution:**
   → Call Outer GMRES
   - Initiate the first candidate vector $\{x_0\} = 0$
   - Matrix-vector product: $[K]\{z_j\} = [Knear]\{z_j\} + [K^{FMM}]\{z_j\}$
   - Preconditioning task: $\{z_j\} = [Knear]^{-1}\{v_j\}$
     → Call Inner GMRES
     - Inner Matrix-Vector product: $[Knear]\{w_j\}$
     - Inner Preconditioning: $\{w_j\} = [diag\_Knear]^{-1}\{\nu_j\}$
3. **Condition to converge:**
   - If $\|(\{b\} - [K]\{x_i\})\| \leq Precision$, stop.
   - Else, set $x_0 \leftarrow x_m$ and go back to Outer GMRES.

As one can notice, the main equation $[K]\{x\} = \{b\}$ is solved by Outer GMRES. This solver is right-preconditioned by $[Knear]_{global}$ and converges if the backward error is smaller than 10^{-3}. We also note a slight deviation from the the Flexible scheme presented in [2] where the inner GRMES functions without any preconditioner. In our work, we take advantage of the block diagonal of $[Knear]_{global}$ and use it as the preconditioner for the inner GMRES (the precision is set to 10^{-1}). By doing so, the number of inner iterations (normally around or even higher than 50) is reduced to less than 20 and the quality of vectors transferred to the outer operation is also well assured. Hence, the inner iteration is sped up as well and contributes to the overall improvement. The performance of Flexible GMRES over GMRES is analyzed in the numerical tests. The source codes of GMRES, Flexible GMRES and corresponding subroutines are downloaded from *http://www.cerfacs.fr*.

3.3 Numerical validations

3.3.1 Parameter choices

Before running the numerical calculations, we need to provide all the necessary input parameters for the program such as: the backward error of GMRES, number of gaussian points, the truncation threshold, the maximal number of elements in a *leaf*... Certain parameters are closely related to the nature of the FM-SGBEM method which play a crucial role in the functioning of the computational code. A slight change of them can affect considerably the overall performance and quality of a calculation. Hence, beside the optimization strategies, the algorithm can be already improved by the pertinent choices of parameters. For this purpose, the very first numerical step is addressed to the analysis of the input parameters. The decisive condition for all tests is the quality of results: a calculation is valid when the biggest relative error $< 1\%$. Once this condition is met, the computational time and resources-consummation are considered. The choices that lead to correct results with minimal requirements are retained.

Parameter of truncation

A generic Fast multipole analysis takes into account an infinite series of products of 2 variables. In a numerical calculation, this development needs to be truncated at a certain number: p. This number must be chosen such that the precision is sufficient and the cost of calculation should not be excessive. Two simple tests in elasticity and fracture mechanics are carried out to analyze the different outcomes due to the variation of different truncation parameters. The enhanced FM-SGBEM code has been utilized, max_elem is set to 30 and the backward error of Flexible GMRES is set to 10^{-3}.

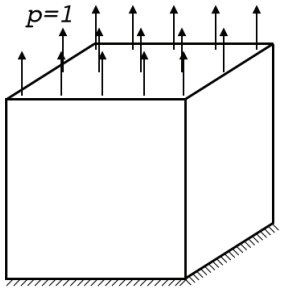


Figure 3.3: Clamped cube under tensile load

The first test consists of an elastic clamped cube being put under uniform load $\overline{p} = 1$ on the free surface (Fig.3.3). The reaction at the support surface and the displacement of all nodes are unknowns in this calculation. The code runs with 5 different parameter of truncation: $p = 4, 5, 6, 7, 8$ on 2 different mesh densities: M1, M2 which feature respectively 9.600 and 21.600 4-nodes quadilateral elements.

There is no exact solution for this problem, so a finite element analysis has been carried out to provide a reference solution. The FM-SGBEM's performance and results are observed. The table below shows the relative errors of different choices of p on 2 meshes (the displacement U_z of a vertical edge is compared with a finite elements calculation) :

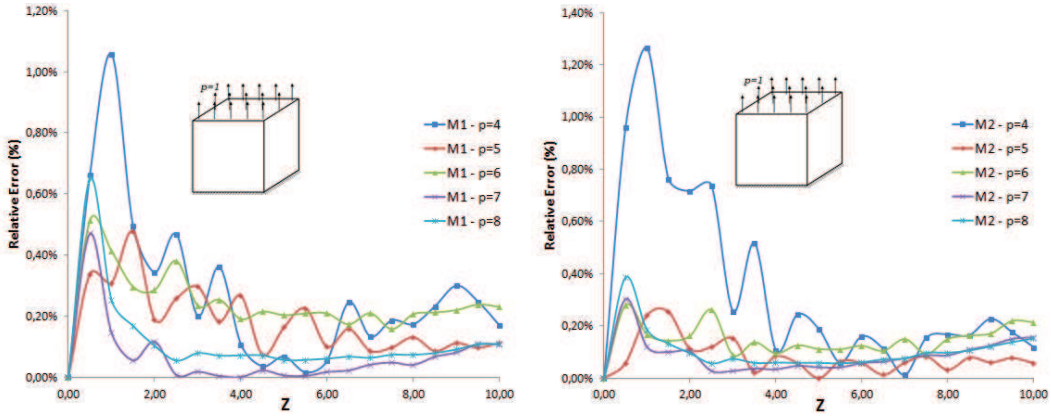
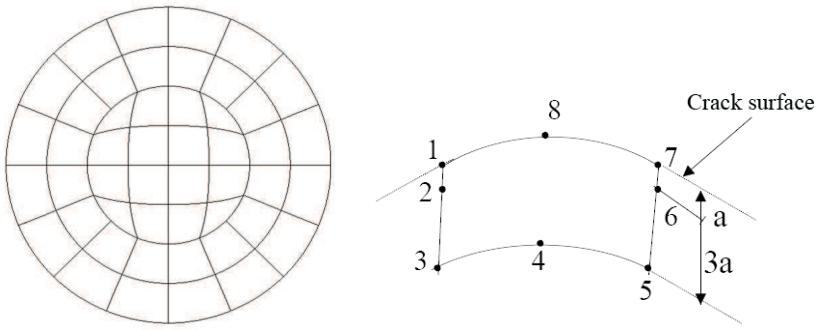


Figure 3.4: Relative error (%) of U_z on a vertical edge

Remark: from the diagram in Fig.3.4, except the case $p = 4$, all the results obtained by $p = 5, 6, 7, 8$ are satisfying with the biggest relative error does not exceed 1%. Therefore, $p = 5$ appears to be a better option since the computational time is the lowest whilst giving results of similar quality.



(a) (b)

Figure 3.5: (a) Illustration of a penny-shaped crack meshed from 48 quarter-point elements Q8 (b) Special elements near crack tip: intermediate points are pushed closer to the crack-tip by a quarter of the edge's length

The second test is run in the context of fracture mechanics. In this test, we compute the crack opening displacement of a penny-shaped crack (radius $a = 1$) in an unbounded domain, subjected to a remote tensile load $\sigma^0_{33} = 1$ (see Fig.3.5). Analogously to the first test, numerous calculations based on different truncation param-

eters $p = 4, 5, 6, 7, 8$ are carried out and compared. Two meshes are taken into consideration: M3 and M4 contain respectively 7.500 and 10.800 quarter-point 8-nodes elements (which are constituted from 22.701 and 32.641 nodes). In this case, there is an analytic solution of the crack opening displacement: $\Delta u = \dfrac{4(1-\nu)}{\pi\mu}\sqrt{a^2 - r^2}\sigma^0_{33}$ for comparison purposes. The quality of the computed crack opening displacements is shown in the Fig.3.6

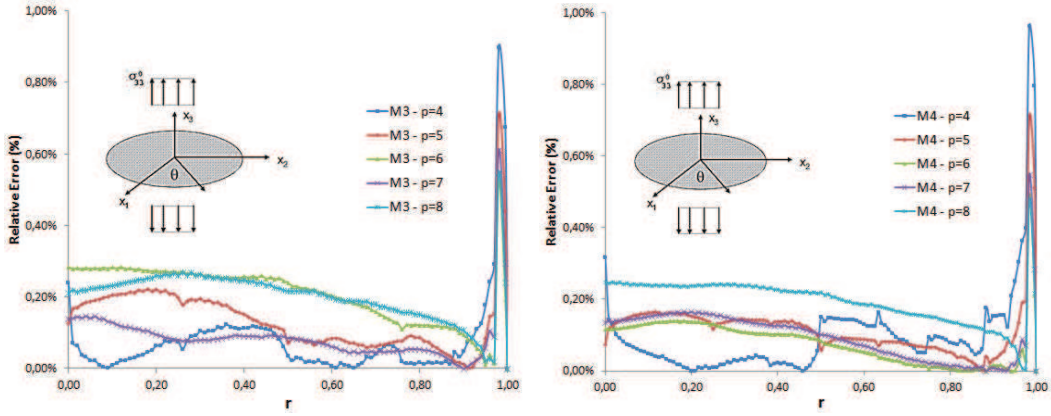


Figure 3.6: Relative error (%) of Δu_z on the radius

Remark: In this second test, all choices exhibit relatively good results. However, $p = 4$ and $p = 5$ give particularly rather big errors near the crack-front. This is not favorable for the study of crack propagation where the behavior near the crack-tip is of the utmost importance. Result of $p = 6$, however, is unstable: best on M4 but worst on M3, so it should not be used. Between the last two options $p = 7$ and $p = 8$, $p = 7$ is better because $p = 8$ would be too expensive.

Overall, we can deduce that the choice $p = 5$ would fit better for non-fracture configuration or for geometries of less interest. On the other hand, $p = 7$ is required on cracks or complex geometries for better precision.

Maximal number of elements in a *leaf*

The maximal number of boundary elements in each *leaf* (max_elem) has been, however, chosen differently for each particular example. The main reason for doing so is to balance the computational portions between the traditional SGBEM and the Fast Multipole algorithm. This parameter has turned out to be a very important index which can affect directly the efficiency of the code: If this parameter is large, the near interactions becomes majority and the traditional evaluations of double integrals will slow down the system solution. On the other hand, if this number is too small, the *octree*'s depth and the number of cells may grow important thus entails a relatively high number of operations but produces a very sparse matrix of near interactions. This matrix would therefore not be helpful as a preconditioner

in accelerating the convergence of the solver and the efficiency of the method is not assured. The study carried out by Chaillat in [2] has proved this fact and has given us some ideas on choosing the appropriate *max_elem* for each problem.

This phenomenon can be illustrated by the same test above. The crack configuration M3 is run now with different *max_elem*. Three options of *max_elem* have been taken as 10,30 and 100. By choosing so, the octree depth is 6,5 and 4 respectively. The outcome of these calculations is reported in the below table 3.3.1. The truncation parameter is set to 7 and the the backward error of Flexible GMRES is 10^{-3}.

max_elem	$\bar{l}$	*ncells*	pre_time(s)	CPU(s) iter	N-Iter	sol_time(s)	tot_time(s)
10	6	2.014	841	35	292	10.516	11.357
30	5	801	1.444	28	207	5.824	7.268
100	4	257	3.293	25	33	854	4.147

Table 3.1: Different outcomes due to different choices of *max_elem* on the mesh M3: $\bar{l}$ denotes octree depth, *ncells* is the total number of cells and N-Iter is the iteration counts. $pre\_time(s)$ denotes the computational time for $[Knear]$ and $\{vect\_y\}$. $sol\_time(s)$ indicates the time consumed by the iterative solver to converge. $tot\_time(s)$ is the total computational time.

As predicted, when *max_elem* is small, the *octree* gets deeper and generates better clustering of near and distant interactions. The computational cost for the preparation increases therefore from *max_elem* = 10 to 100. However, since the *octree* structure becomes more complex, the cost of each iteration increases as represented by the CPU(s) per iteration. Lastly, the trade-off between the preparation time and the quality of $[Knears]$ as a preconditioner is well shown: despite having the longest preparation time, the third test assumingly produces the best preconditioner thus speeds up drastically the convergence rate of Flexible GMRES. The number of iteration counts of the third run is 1/6 compared to the second and only 1/9 compared to the first, making *max_elem*=100 the most optimal choice in terms of overall speed (See Fig.3.7).

In general, this remark does not hold true every time and is very hard to predefine a rule for choosing *max_elem*. On different geometries, mesh densities and different material properties (different matrix spectral), we get different outcomes. For instance, similar tests have been run on mesh M4:

max_elem	$\bar{l}$	*ncells*	pre_time(s)	CPU(s) iter	N-Iter	sol_time(s)	tot_time(s)
10	7	3.637	1.029	57	178	10.503	11.532
30	6	1.089	2.018	41	87	3.564	5.582
100	5	305	5.555	36	71	2.639	8.194

Table 3.2: Different outcomes due to different choices of *max_elem* on the mesh M4

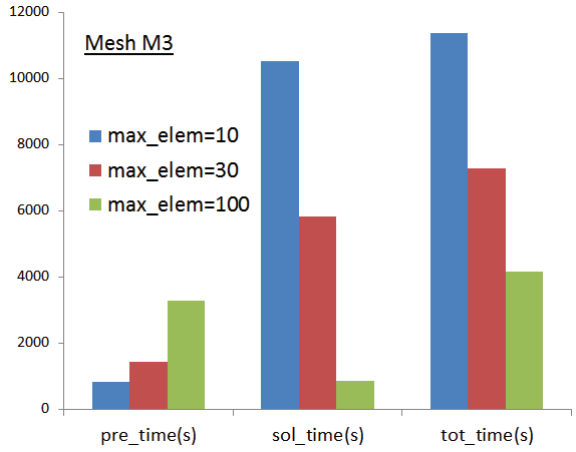


Figure 3.7: Computational times (s) by different choices of *max_elem* on mesh M3

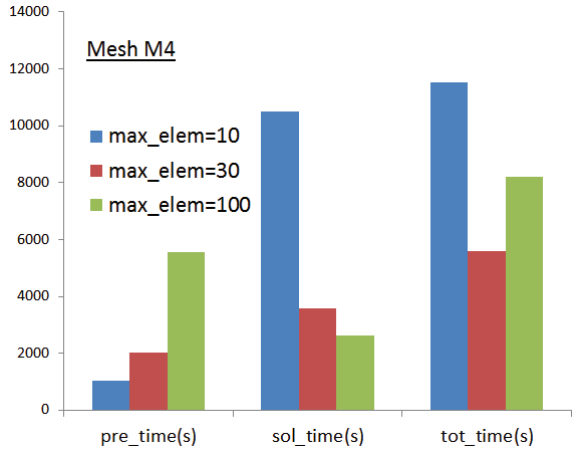


Figure 3.8: Computational times (s) by different choices of *max_elem* on mesh M4

In Fig.3.8, we can notice that the computational times follow the above principle: high *max_elem* is time-consuming for the preparation phase but is efficient in the solution phase and vice-versa. However, in this case, as *max_elem* = 100 takes too long in the near calculation, the overall computational time is not the fastest. Also, the memory usage must also be taken into consideration. In the above tests, *max_elem*=100 takes the most memory slots: 5 times more than the first choice. Therefore, throughout the numerical experiments, *max_elem*=30 has been retained in almost simulations as it can be stable and balances the overall speed over memory requirements. If different value of *max_elem* is chosen other than 30, it is because it particularly produces the most optimized performance.

Number of Gaussian points

In a boundary calculation, the number of gaussian points is also crucial to determine the trade-off between the computational speed and precision. The lower number will obviously lead to better speed but risks having considerable errors. As the surface integral equations deal with singularity $O(r^{-1})$, higher degree of precision should be applied for 'close' interactions and less points for 'further' interactions. An empirical scheme for selecting appropriate number of gaussian points has been introduced by Rezayal et al. in [1986]. As stated in the formula, the choice depends primarily on the ratio of element size to the distance between two elements. Here, for quadrilateral elements, the element size, H is taken as the length of the longest diagonal. The distance between the centers of 2 elements is denoted by d. The line between these 2 centers form an angle θ with the normal of the exterior element (see Fig.3.9).

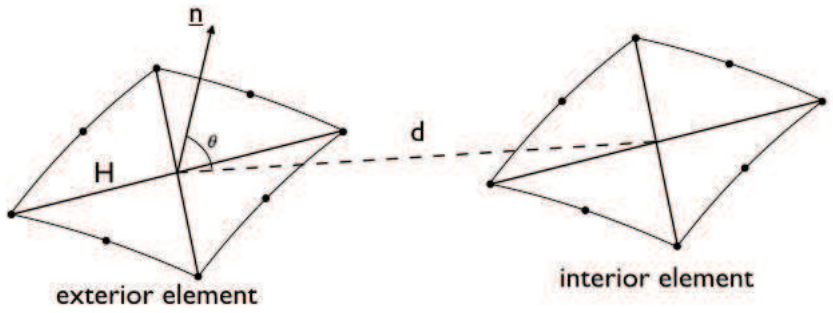


Figure 3.9: Geometry of a pair of elements

For convenience, an index of severity (IS) has been introduced to quantify the variation in the required degree of the integration formula:

$$IS = (2.37 + 0.424cos\theta)H/d \tag{3.3}$$

(IS is rounded off to the nearest integer). The correlation between IS and the number of Gaussian points is given in table 3.3.

A simple test has been carried out to investigate the quality of the result issued from this variation of gaussian points. Single penny-shaped crack (48 elements Q8) in an unbounded domain is put under tensile load $\sigma_{33} = 1$. Backward error of the iterative solver is set to 10^{-3}, max_elem chosen as 10. By adopting the criterion, the integrations are divided in 3 categories according to the distance and elements size and are shown in table 3.4. From the table, instead of using 4x4 gaussian points, the regular schemes are now computed with 2x2 or 3x3 points thus reduced considerably the operations counts. In terms of precision, comparing with the exact result (Fig.3.10), this scheme exhibits very good quality outputs, especially the displacement discontinuity at the vicinity of the crack front, the relative error is $0,5\%$.

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

Table 3.3: Table of serverity index

	near_16	regu_4	regu_9	regu_16
Integral counts without IS	210	0	0	390
Integral counts with IS	210	213	177	0

Table 3.4: Number of pairs of elements integration using different numbers of gaussian points: *near_16* denotes the near (singular) interactions count with 4x4 gaussian points. Analogously, *regu_4*, *regu_9*, *regu_16* show respectively the number of integrals with 2x2, 3x3 and 4x4 gauss points. Initianlly, regular double integrals with 4x4 gaussian points requires $390 \times 4 \times 4 \times 4 \times 4$ operations. With the IS implemented, the operation counts are $(213 \times 2 \times 2 \times 2 \times 2) + (177 \times 3 \times 3 \times 3 \times 3)$ which is equal to a 80% reduction of operations.

This technique is quickly implemented in large-scale tests and has obtained a great boost in terms of near computational time. The details are later reported in the performance test.

3.3.2 Performance test

In this numerical test, the enhanced performance of the FM-SGBEM algorithm is justified. All the clear improvements due to the proposed strategies are mentioned. Since the memory requirement is well-assured by the compressed sparse format, there leaves only one major concern: the computational speed. In order to provide a comparative basis, similar computations by GMRES (preconditioned by the block diagonal of $[Knear]$) have been performed along with Flexible GMRES. Some lateral tests have been carried out first hand to ensure that GMRES is also executed efficiently on the similar set of parameters as Flexible GMRES and the obtained results are of equal precision.

Regarding this performance test, the FM-SGBEM is applied in a large-scale fractured configuration: a network of n_c^3 penny-shaped cracks is introduced in an unbounded homogeneous domain ($E = 1, \nu = 0.3$). A tensile load $\sigma_{33}^0 = 1$ is put on the crack system and we compute the crack opening displacement. Each crack

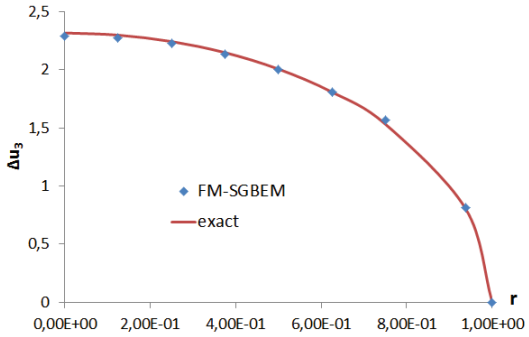


Figure 3.10: Selective number of gauss point: numerical and exact results on crack opening displacement

(radius $a = 1$) is modeled using 48 8-nodes quarter-point quadrilateral elements (see Fig.3.5). These cracks are generated regularly on a cubic grid with a regular distance of $4r$ in each coordinate direction and are randomly oriented in space (illustrated in Fig. 3.11). The number of crack in this system is therefore n^3 (n being the number of cracks in one direction).

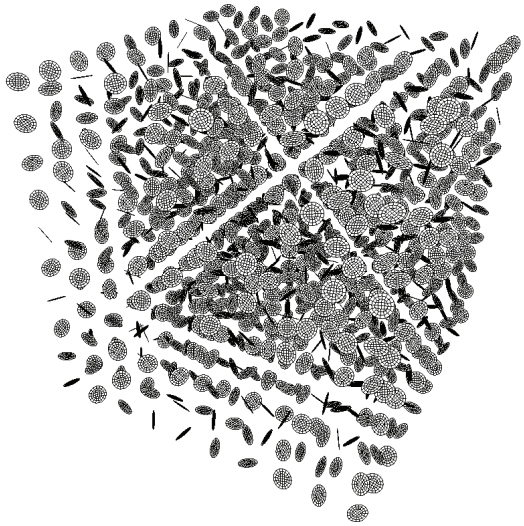


Figure 3.11: System of 1000 randomly-oriented penny-shaped cracks in an unbounded domain

We compute the crack opening displacement (COD) with help of the governing variational traction equation written on S_c:

$$\int_{S_c}\int_{S_c}[R\Delta u]_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})[R\Delta\tilde{u}]_{iq}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x = \int_{S_c} p_k(\mathbf{x})\Delta\tilde{u}(\mathbf{x})dS_x \qquad (3.4)$$

In order to test the performance of the 2 chosen solvers, we generate 4 meshes by varying $n_c = 14, 16, 18, 20$ which contain respectively arrays of 2.744, 4.096, 5.832 and 8.000 cracks. All of these meshes feature more than 1 millions unknowns. The calculations were carried out successively on these 4 configurations by Flexible GMRES and by GMRES. Similar input parameters are chosen for these 2 solvers: parameter of truncation is set to $p = 7$, precision is 10^{-3}. Information of these tests are shown in Table 3.5.

Mesh	N-Cracks	NEQ	max_elem	$\bar{l}$
1	2.744	1.061.928	15	7
2	4.096	1.585.152	30	6
3	5.832	2.256.984	30	7
4	8.000	3.096.000	30	7

Table 3.5: Information of 4 meshes: *NEQ* denotes the number of unknowns, *max_elem* is the maximum number of elements in an *octree* cell, $\bar{l}$ is the depth of the *octree* structure.

Near Computation: For simplicity purposes, the effect of different numbers of gaussian points for regular integration by the index of severity (IS) is investigated first hand. Simulations on 4 meshes are carried out twice, one time with constant 4x4 gaussian points for all integrations (both singular and regular) and one with the IS implemented. Table 3.6 shows the computational time for computing $[Knear]$ and $\{vect\_y\}$ in both cases.

Mesh	$pre\_time(s)$		Gain
	all 4x4	with IS	
1	17.561	9.995	43.08%
2	23.404	12.890	44.92%
3	45.825	22.753	50.3%
4	112.000	44.070	60.65%

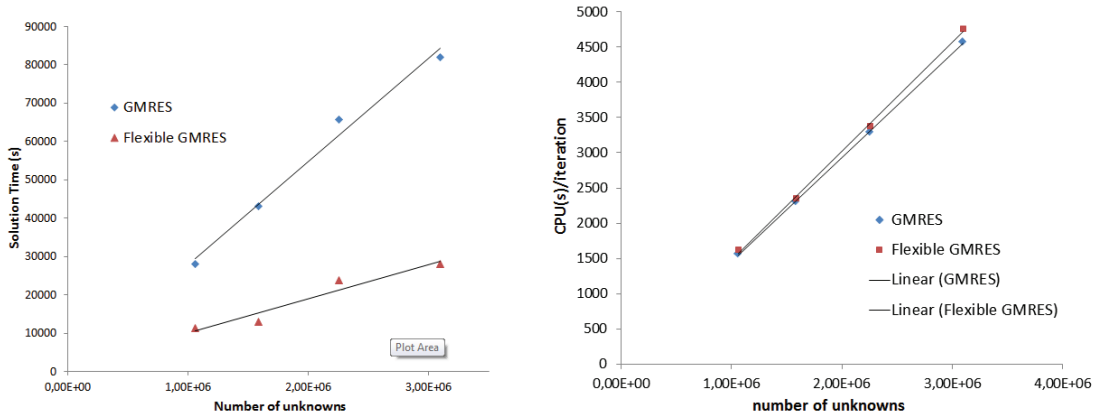
Table 3.6: Time (s) for computing $[Knear]$ and $\{vect\_y\}$ (pre_time) using respectively 4x4 gaussian points (second column) and using selective numbers of gaussian points by adopting the IS (third column). The last column indicates the gain of computational time.

Iterative Solution: Secondly, the performance of the iterative solution phase is observed. Instead of using the operation counts as a comparative basis, the solution times (defined equally as the convergence rate) are used to justify the performance of the solvers. Table 3.7 details all the computational times on the 4 meshes by GMRES and Flexible GMRES. The solution times are shown in Fig.3.12a. Fig.3.12b illustrates the CPU(s) of each iteration by these 2 solvers.

- **Remark 1:** The index of severity (IS) negates greatly the need of too 'precise'

Mesh	N-Iter		CPU(s)/iter		Sol_time (s)			Tot_time (s)		
	Flex	GMRES	Flex	GMRES	Flex	GMRES	Gain	Flex	GMRES	Gain
1	6	18	1.626	1.569	11.155	28.271	60,54%	21.087	38.952	44,02%
2	4	18	2.349	2.317	11.706	42.659	72,56%	26.053	57.067	54,35%
3	4	18	3.377	3.299	16.711	59.004	71,68%	42.424	84.904	50,03%
4	4	18	4.756	4.576	23.542	79.403	70,35%	73.383	128.828	43,04%

Table 3.7: Crack configurations and solution times respectively by Flexible GMRES and GMRES. *N-Iter* indicates the number of iterations; *CPU(s)/iter* is time (s) consumed per iteration; *sol_time* denotes the iterative solution time (s) and *tot_time* is the total computational time (s).



(a) (b)

Figure 3.12: Multicrack - (a) Solution time and (b) Time per iteration by GMRES and Flexible GMRES

integrals by replacing the 4x4 gauss scheme with 2x2 or 3x3 thus halves the computational time for $[Knear]$ and $\{vect\_y\}$.

- **Remark 2:** Flexible GMRES and GMRES both exhibit linear dependent relation between the CPU(s) per iteration and the number of unknowns (Fig.3.12b). One can easily notice that Flexible GMRES is more expensive than GMRES for each iteration. This is understandable because Flexible GMRES also features the inner solution for the preconditioning task.

- **Remark 3:** Globally, Flexible GMRES converges much faster than GMRES which results in a $60-70\%$ solution time cut off (Fig.3.12a). Consequently, $40-50\%$ of the total computational time can be reduced by using Flexible GMRES.

3.4 Conclusions

In this chapter, two majors improvements have been implemented for the algorithm of FM-SGBEM to compute the large-scale problems. The first consideration is the rational storage of the near interaction matrix $[Knears]$. For this purpose, the CSRSYM approach has been applied in the hierarchical algorithm of the FM-SGBEM. This work does not only divide the global $[Knears]$ into many cell-level matrices of insignificant size, but also compress each symmetric and sparse sub-matrix into arrays of non-zero indexes. The memory usage is therefore pushed to the minimal possible which helps computing successfully many test problems whose size goes up to 3 millions unknowns on a single processor PC.

The second concern of the FM-SGBEM algorithm is the high iteration counts in the solution phase by GMRES. The reduction of the solution time by fastening the convergence is met by the use of a more robust solver: Flexible GMRES, where the preconditioning task is occupied by a better matrix: $[Knears]$. The particularities and reasons for which Flexible GMRES is more favorable than GMRES are discussed. The algorithm and implementation of Flexible GMRES in the FM-SGBEM have also been presented.

Some of the important parameters (truncation, $max\_elem...$) have been chosen so that the program is computationally robust while being sufficiently correct. Nevertheless, the study did not include the influence of a great number of other factors: geometries, material properties, restart parameter etc.

Lastly, the enhanced performance of the code has been demonstrated through some numerical examples. It is effectively proved that in most cases, Flexible GMRES outperforms significantly GMRES in terms of speed and of quality of results. This optimized algorithm has been therefore utilized thorough out all the remaining numerical tests in this thesis.

In the following chapter, we will discuss about the extension of the method into more sophisticated problem where the heterogeneity is accounted for and the interfaces between sub-domains are presented.

Chapter 4

Extension of the FM-SGBEM to multizone problems

Contents

Multizone problems also 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... At the common boundary between two sub-domains (interface), the corresponding full matching behaviors have to be enforced. One important aspect in a multi-domain algorithm is to enforce the continuity and equilibrium conditions at interfaces, but there are no prescribed values.

Considering a generic fractured interface problem Ω containing 3 homogeneous sub-domains (Fig. 4.1). On an interface, both displacements and traction are unknowns that belong to all adjacent bodies. In a simple case where the interface is shared by 2 bodies. The continuity of displacement and equilibrium conditions are: $\mathbf{u}^a(\mathbf{x}) = \mathbf{u}^b(\mathbf{x})$ and $\mathbf{t}^a(\mathbf{x}) = -\mathbf{t}^b(\mathbf{x})$ (a, b being the name of two adjacent zones). The normal vector of an interface is chosen so that it is directed from the zone having the lower enumeration to the zone having the higher number.

In general, the BEM becomes expensive compared to the domain techniques when the studied problem exhibits a high ratio of surface to volume. This is true

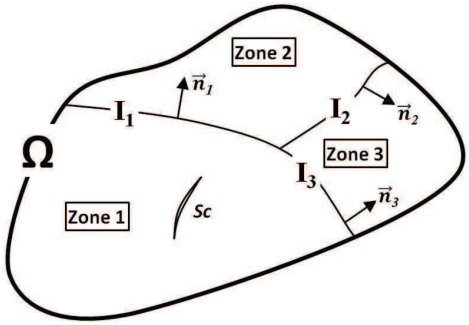


Figure 4.1: A multizone fractured domain

for multizone problems with the presence of interfaces (internal boundaries). Nevertheless, the BEM approach is still attractive for this class of problem because of the natural treatment of continuity conditions: For the displacement-based FEM, enforcing the continuity of traction is a difficult task, while this quantity appears directly in the boundary integral formulation.

SGBEM normally provides a symmetric system matrix but when applied to the sub-domains formulation, this property cannot be completely achieved. In order to conserve the global symmetry of the method, an appropriate technique must be adopted during the matrices construction. Layton et al. [24] 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 rests are symmetric. In [10], Gray and Paulino 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 [5] and fracture mechanics [23]. With the advantageous nature and symmetry in treating multizone problems, the SGBEM becomes therefore a formidable option. The need of solving practical multizone issues which feature high amounts of unknowns naturally leads to the application of the Fast Multipole Method. With the complexity of $O(Nlog^{\alpha}N)$, the multizone FM-SGBEM is expected to be a powerful alternative for many important realistic applications.

In this work, the approach described in [10] by Gray and Paulino has been exploited. Perfect bonding between sub-domains is assumed first, imposing the continuity of displacement and the equilibrium of traction across the interface. Via some appropriate terms rearrangement and sign adoptions, the symmetry of the global matrix can be achieved. Secondly, the Fast Multipole Method is introduced in the multizone SGBEM formulation. Some computational and efficiency issues of the algorithm is discussed in the later subsection. Lastly, numerical experiments on validation tests are given and one extension on practical material is reported, followed by some conclusions and perspectives.

4.1 SGBEM formulation for multizone problems

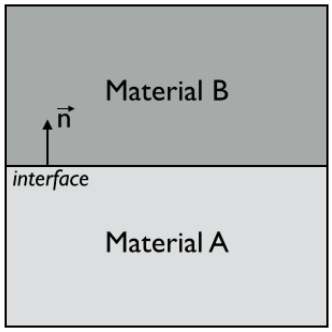


Figure 4.2: A bi-material interface problem

A simplest geometry would be preferable 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.4.2). The solid contains 2 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 2 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-interfacial and interfacial unknowns of A and B respectively.

The basic idea is to write the usual SG 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:

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

and zone B:

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

these terms are well detailed in chapter 2, the upper scripts indicate the surfaces on which the integrals are written, for instance:

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

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: replace the interface traction of the top region (B) by the negative of the bottom interface traction (A). The equation (4.3) is thus transformed to:

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

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

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

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) are also result of 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 unsymmetric. 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 from the interface is invoked.

Let us now consider a generic fractured sub-domain d sharing n interfaces $(I^1, I^2, ..., I^n)$ with its surrounding zones. S^d denotes the non-interfacial boundaries of this body. In a general case, S^d can be an union of prescribed traction, displacement and cracks surfaces: $S^d = S_t^d \cup S_u^d \cup S_c^d$. The SGBEM block matrix form for this zone can be written as:

$$[K^d]\{x^d\} = \{b^d\} \tag{4.6}$$

$[K^d]$ is the coefficient matrix that stores all the coefficients which are derived from the integral equations of SGBEM written for all surfaces in the zone d. This matrix can be expressed in term-wise manner as follow:

$$\begin{bmatrix}
B_{uu}^{S_tS_t} & B_{tu}^{S_uS_t} & B_{\Delta uu}^{S_cS_t} & B_{uu}^{I_1S_t} & \circ B_{tu}^{I_1S_t} & ... & B_{uu}^{I_nS_t} & \bullet B_{tu}^{I_nS_t} \\
B_{ut}^{S_tS_u} & B_{tt}^{S_tS_t} & B_{\Delta ut}^{S_cS_t} & B_{ut}^{I_1S_u} & \circ B_{tt}^{I_1S_u} & ... & B_{ut}^{I_nS_u} & \bullet B_{tt}^{I_nS_u} \\
B_{u\Delta u}^{S_tS_c} & B_{t\Delta u}^{S_uS_c} & B_{\Delta u\Delta u}^{S_cS_c} & B_{u\Delta u}^{I_1S_c} & \circ B_{t\Delta u}^{I_1S_c} & ... & B_{u\Delta u}^{I_nS_c} & \bullet B_{t\Delta u}^{Si_nS_c} \\
B_{uu}^{S_tI_1} & B_{tu}^{S_uI_1} & B_{\Delta uu}^{S_cI_1} & B_{uu}^{I_1I_1} & \circ(B_{tu}^{I_1I_1} + I_u^1) & ... & B_{uu}^{I_nI_1} & \bullet B_{tu}^{I_nI_1} \\
\circ B_{ut}^{S_tI_1} & \circ B_{tt}^{S_uI_1} & \circ B_{\Delta ut}^{S_cI_1} & \circ(B_{ut}^{I_1I_1} - I_t^1) & B_{tt}^{I_1I_1} & ... & B_{ut}^{I_nI_1} & \circ \bullet B_{tt}^{I_nI_1} \\
... & ... & ... & ... & ... & \ddots & ... & ... \\
B_{uu}^{S_tI_n} & B_{tu}^{S_uI_n} & B_{\Delta uu}^{S_cI_n} & B_{uu}^{I_1I_n} & B_{tu}^{I_1I_n} & ... & B_{uu}^{I_nI_n} & \bullet(B_{tu}^{I_nI_n} + I_u^n) \\
\bullet B_{ut}^{S_tI_n} & \bullet B_{tt}^{S_uI_n} & \bullet B_{\Delta ut}^{S_cI_n} & \bullet B_{ut}^{I_1I_n} & \bullet \circ B_{tt}^{I_1I_n} & ... & \bullet(B_{ut}^{I_nI_n} - I_t^n) & B_{tt}^{I_nI_n}
\end{bmatrix} \tag{4.7}$$

$\{x^d\}$ and $\{b^d\}$ are respectively the solution vector and local known vector. The solution vector $\{\mathbf{x}^d\}$, despite being written for the *body* $- d$, contains the global unknowns which are related to this body. The right-hand side vector $[\mathbf{b}^d]$ contains all the known values and can be expressed as follow:

$$\{\mathbf{x}^d\} = \begin{pmatrix} u^{S_t^d} \\ t^{S_u^d} \\ \Delta u \\ u^{I_1} \\ t^{I_1} \\ \vdots \\ u^{I_n} \\ t^{I_n} \end{pmatrix} \qquad \{\mathbf{b}^d\} = \begin{pmatrix} \mathcal{F}(\tilde{u}^{S_t}) \\ \mathcal{F}(\tilde{t}^{S_u}) \\ \mathcal{F}(\Delta\tilde{u}^{S_c}) \\ \mathcal{F}(\tilde{u}^{I_1}) \\ \circ\mathcal{F}(\tilde{t}^{I_1}) \\ \vdots \\ \mathcal{F}(\tilde{u}^{I_n}) \\ \bullet\mathcal{F}(\tilde{t}^{I_n}) \end{pmatrix} \tag{4.8}$$

The terms $[B_{**}]$ and $[I_*]$ denote double and single integrals of the SGBEM formulations, $\mathcal{F}(*)$ denote the known value related to the test-function. Detailed of these terms can be found in chapter 2. One important issue in the multizone problems is the sign adoption. Correct signs for the interfacial traction have to be accounted for where symbol $\circ$ or $\bullet$ is present. This is to incorporate the equilibrium of the traction vector from one zone to another across the interface. For example, if the normal vector of the studied interface is inward, the sign of the corresponding traction term and its transpose is negative and vice-versa [5].

As the previous remark, the single integral $[I_u^{Si}]$ is, in general, different with the transpose of $[I_t^{Si}]$, thus rendering the local coefficient matrix $[K^i]$ non-symmetric. Fortunately, the equation (4.6) is written for every zone which eventually gives rise to equal and sign-opposite single integral terms. Since they are not material-dependent, these single integrals simply disappear during the assembly process and the global system is fully symmetric. As is proven in [10], this multizone SGBEM algorithm is computationally very efficient: the global matrix is of reduced size because only unknowns on one side of the interface are considered; and the symmetry

can be used to reduce the matrices build-up cost or to couple with finite elements method.

4.2 Fast multipole SGBEM for multizone problem

In this work, we discuss directly the application and the solution of the Fast Multipole method in the multizone SGBEM without describing the solution of the multizone SGBEM since they follow similar principles. The same settings of the fast multipole algorithm in a single domain can be effectively extended to the case of multiple domains. The only distinct feature in a multizone problem is the gradual computation of n sub-problems (n being the number of bodies). Each sub-problem is represented as a sub-domain of a distinguishing material. Since an indispensable part of a sub-domain is the interface which contains unknown values, each sole sub-domain is not well-posed and thus cannot be solved separately. However, the coefficients matrix and the right-hand side vector of each zone can be easily obtained. By evaluating all the local sub-domains, we eventually gain access to the global system. This system is derived from the linear combination of the component matrices and known vectors of all sub-domains. As a result, the solution of a multizone problem is conducted to the solution of a combined system which is globally well-posed and solvable. This task appears to be trivial since the equation system is also linear and can be handled by iterative approaches as introduced in single-domain problems.

4.2.1 Multizone FM-SGBEM algorithm

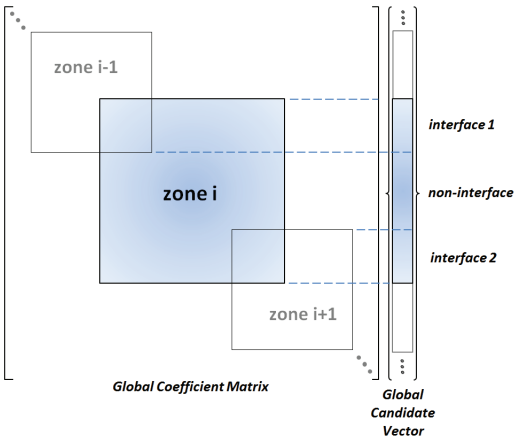


Figure 4.3: Block matrixes in Multizone problem

The multizone FM-SGBEM algorithm can be summarized in a few steps: (a) an *octree* structure is constructed first, covering the whole solid. (b) a loop on all bodies is called. Near interactions are computed then stored locally in $[Knears]$.

(c) an iterative solver (Eg. 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.4.3: (1) Zone-i sharing 2 interfaces with zone i-1 and zone i+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 obtain eventually the global matrix-vector product for GMRES. After the convergence is achieved, the post-processing does not differ from the case of single domain. The detailed algorithm of the multizone FM-SGBEM can be expressed as follows:

Algorithm 6 Multizone FM-SGBEM

1-Initialization

(a) Import geometries and parameters

(b) Create an **octree** for the complete multizone configuration

(c) Compute the right-hand side vector $\{b\}$

Initiate $\{b\} = 0$

Loop over $i = 1, nbody$

Compute $\{b\}$ for each subdomain i, set $\{b\} := \{b\} + \{b\}_i$

End

(d) For each $i = 1, nbody$, compute $[Knear]_i$

2-Iterative Solution

(a) Initiate family of Krylov vectors: $\{w\}_1 = \{b\}$

(b) GMRES main loop: set $k = 1$; while $\parallel [K]\{x\}_k - \{b\} \parallel \geq 10^{-3}$ (precision)

(i) Compute new Krylov vector $\{w\}_{k+1}$:

Initiate $\{y\}_k = 0$

Loop over $i = 1, nbody$

Compute $\{y\}_k^i := \{y\}_k^i + [K]^{i,FMM}\{w\}_k^i$ using FMM

Add near contribution: $\{y\}_k^i := \{y\}_k^i + [K]^{i,near}\{w\}_k^i$

End

Solve $[K]^{near}\{w\}_{k+1} = \{y\}_k$ using inner GMRES loop (preconditioning)

(stopping criterion: $\parallel [K]^{near}\{w\}_{k+1} = \{y\}_k \parallel \geq 10^{-1}$)

(ii) Find $\{x\}_k \in Vect(w_1, w_2, ..., w_k)$ such that $\parallel [K]\{x\}_k - \{f\} \parallel \rightarrow min$

(iii) Set $k := k + 1$

3-Post Processing

4.2.2 Multizone FM-SGBEM numerical implementation

The scheme of the multizone FM-SGBEM is based essentially from the single-domain FM-SGBEM scheme which is presented in chapter 2. Even though the principle conveys the impression of simplicity, the numerical implementation of the multizone FM-SGBEM has proven to be a more difficult task. A lot of compu-

tational efforts have been made for this purpose but due to the vast complexity of multi-domain geometries, it still appears impossible to model and compute efficiently any generic or random multizone problems. There are thus numerous adjustments in the numerical code in order to cope with each particular problem. Details of these changes will be clarified and discussed along with the problem description. In this section, we present the most general computation scheme (pre processing in Fig.4.4 and main processing in Fig.4.5) for a multizone FM-SGBEM.

4.2.2.1 Pre-Processing

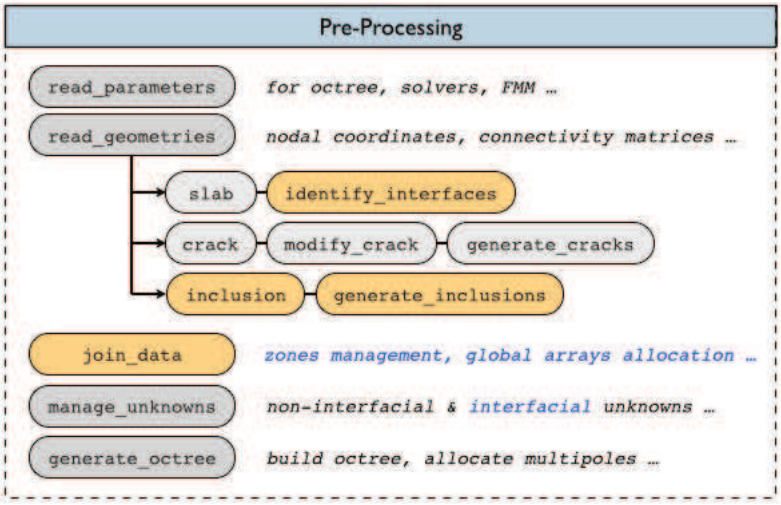


Figure 4.4: Multizone FM-SGBEM: Pre-Processing phase

The calculation begins with the import of parameters and geometries. Here, slab usually denotes the outer boundary (either fractured or not), crack denotes one single penny-shaped crack whilst inclusion is often represented as a sphere. These geometries are generated by **Gid** thus only 'raw' texts of nodal coordinates and connectivity matrices are provided. To fully define a particular multizone and multifracture problem, we usually need to declare more input information. In the spirit of minimizing the labor work for each test variety, the data provided by **Gid** needs to be complimented by some automatic codes so that the flexibility is not lost. For this purpose, a number of subroutines in the pre-processing phase is written: for instance, *generate_crack* and *generate_inclusion* can multiply the original crack or inclusion, adjust size, rotate and distribute them in space... Once the preparation is done, join_data regroups all component entities and form the expected global geometry (with distinguishing zones, global arrays of unknowns types and positions etc ...)

The second stage of the pre-processing is to construct the *octree* structure. To adapt with the multiple sub-regions feature, there are 2 alternatives: (1) Build only one *octree* structure for the entire geometry, (2) Build separate *octree* structures for

each body. For the first approach, the *octree* generation can be simple and quick (no real change from the mono-zone algorithm). However, stricter conditions should be applied in the latter steps to prevent useless operations. Contrarily, The second approach can handle local FMM-operations rather smoothly and elegantly since the *octree* does not possess 'exotic' elements (from unwanted zones). Nevertheless, the drawback of the second approach is the lack of flexibility and adaptability: when the difference of geometrical dimensions or mesh refinement between internal zones become considerable (Eg. matrix-inclusions problem) or when the number of internal bodies grows big, it becomes more sophisticated to correctly generate and manage the local *octrees* in an efficient way. Sometimes, it would be more desirable to consider a coarse zone/mesh by pure SGBEM approach rather than by FMM. In our work, we adopt the first approach to generate the *octree* structure as well as the related Fast algorithm, some adaptive changes are also included to replace the FMM operations by pure SGBEM to achieve the best efficiency.

4.2.2.2 Main-Processing

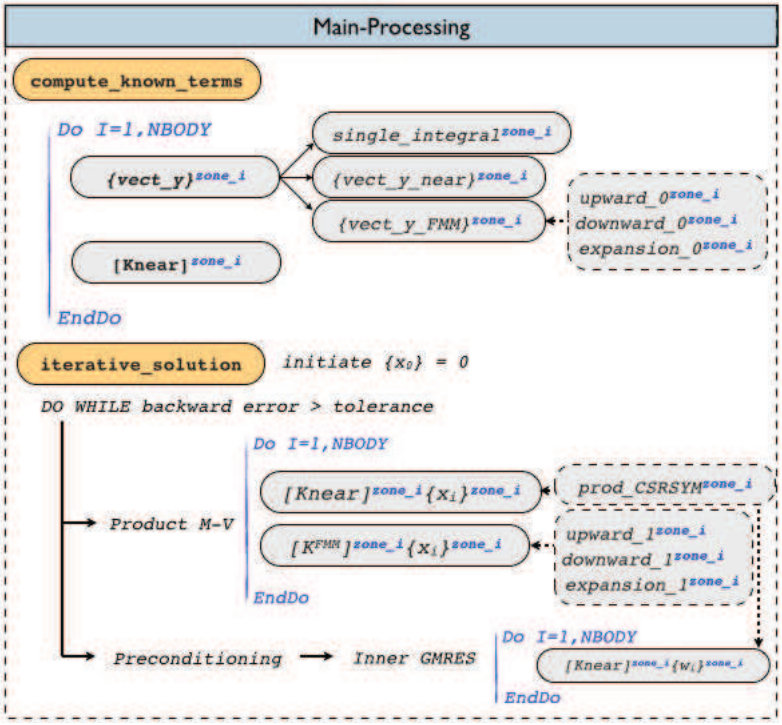


Figure 4.5: Multizone FM-SGBEM: Main-Processing phase

The main program consists of constructing and solving the linear system equation. The numerical implementation of the multizone follows closely the previously described algorithm: whenever a global-scale quantity is needed, a loop on all bodies (N_{body}) are called and all local computations are performed. At the end of each

local computation, the local components are added up to form the global quantities:

$$[Knear] = \sum_{i=1}^{N_{body}} [Knear]^i$$

$$\{vect\_y\_near\} = \sum_{i=1}^{N_{body}} \{vect\_y\_near\}^i$$

$$[Knear]\{x\} = \sum_{i=1}^{N_{body}} [Knear]^i\{x\}^i$$

Multizone sign adoption

Different sign adoptions for traction-related terms on different normal orientations is the reason for which the most adequate modifications and adjustments must take place to ensure the exactitude and efficiency of the multizone program. The first remark is addressed to the sign adoption of the SGBEM terms. The Fast algorithm breaks the double integrals into 2 parts: one in the multipole moments and one during the expansion process. Hence, if the correct sign for the interfacial traction is adopted in one process, it should not be repeated in another. Example for the term $B_{tu}^{S_i S_t}$, the multipole moment would be computed on $\int_{S_i} t^{S_i}...$ and the expansion is on $\int_{S_t} \tilde{u}...$, the sign is thus put in the multipole moment, not in the expansion (in case the normal vector of interface S_i is inward). Secondly, providing that the normal vector of a boundary surface in an integration is outward by convention, we also need to invert the nodal sequence of the elements on an inward-interface to keep it 'locally' outward.

Multizone FMM memory allocations

Even though a good majority of the memory is reserved for $[Knear]$, we should not underestimate the Fast Multipole memory management. In the multi-level FM-SGBEM, the multipole moments and local expansions are computed for every level, every cell and every degree of freedom. Hence, a large amount of memory slots is reserved to store these quantities (especially for multizone problems). A good management in this step is necessary to contribute to the overall efficiency. In a multizone problem, the 'local' computation is restricted to only one sub-domain at a time, so the cells in the *octree* structure which contain no element of the studied zone should be 'muted' to minimize the memory usage *(no memory allocation)* - otherwise they are 'active' - See Fig.4.6. Overall, this memory allocation is temporary. When the local calculation ends and the next zone is called, these slots are freed and another cycle repeats. The highest amount of memory in use is thus required by the zone that features the most cells. For this matter, the multizone FMM requires less memory than single-domain FMM of a same problem size.

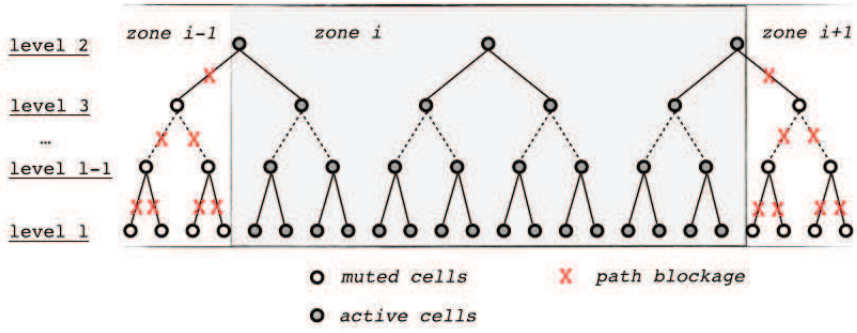


Figure 4.6: Multizone FM-SGBEM Memory management: the gray window represents the studied *zone i*, the cells which contain elements of *zone i* are all computationally active, the rests are muted

Regarding the active cells, despite having many more terms, the number of arrays to store the multizone FMM moments need not exceed the single-domain case. Let [MBTT] and [LBTT] denote respectively the arrays containing multipole moments and local expansion of $\int_{S_u} t$ for the terms $B_{tt}^{S_u S_u}$, $B_{tu}^{S_u S_t}$ and $B_{tu}^{S_u S_c}$. The presence of interfaces $\int_{S_i} t^{S_i}$ is accounted for by simply adding this contribution in these arrays (no further need to allocate new distinct memory slot such as [MBTT]_I or [MBTT]_C particularly for interface or crack surfaces). By expanding these arrays (multipole moments) on the corresponding surfaces S_u, S_t, S_c or S_i, the routines can also automatically compute the terms $B_{tt}^{S_u S_i}$, $B_{tt}^{S_i S_u}$, $B_{tt}^{S_i S_i}$, $B_{tu}^{S_u S_i}$, $B_{tu}^{S_i S_t}$, $B_{tu}^{S_i S_c}$ and $B_{tu}^{S_i S_i}$ which appear in the multizone formulation. Hence, two set arrays [MBTT] and [LBTT] are sufficient to compute all $\boxed{B_{tt}}$ and $\boxed{B_{tu}}$ terms in the system. Analogous concept is applied for the terms $\boxed{B_{ut}}$ and $\boxed{B_{uu}}$ and they are expressed below:

$$\begin{aligned}
&\text{[MBTT] and [LBTT]} \longleftarrow \int_{S_u} t \text{ and } \int_{S_i} t^{S_i} \\
&\text{[MBUT] and [LBUT]} \longleftarrow \int_{S_t} u \text{ and } \int_{S_c} \Delta u \text{ and } \int_{S_i} u^{S_i} \\
&\text{[MBUU] and [LBUU]} \longleftarrow \int_{S_t} (Ru) \text{ and } \int_{S_i} (Ru^{S_i}) \text{ and } \int_{S_c} (R\Delta u)
\end{aligned} \tag{4.9}$$

In terms of memory allocation, this confinement may appear fairly helpful: Skipping all the intermediate steps, a general cell would contain $40(p+1)(p+2)$ *double-precision* terms (p being the parameter of truncation). Hence, for an *octree* featuring about 20.000 cells and p is set to 7, the total amount of RAM for all the multipole moments introduced in (4.9) is approximately 60 Mb. If extra multipole arrays (Eg. [MBTT_I] for interfaces or [MBTT_C] for cracks) were used to better distinguish the different surfaces and to facilitate the programming, the extra RAM needed would

be only about 120 Mb. This amount of memory is inconsiderable in comparison with $[Knear]$. However, from the computational point of view, these additional arrays will triple the FMM operations because M2M, M2L and L2L must include the extra interface and crack arrays. This entails a process roughly 3 times more expensive than a normal Fast algorithm would need as consequence. Therefore, this confinement remark is overall advantageous and should not be underrated.

Multizone FMM passages

In the scenario where only one *octree* structure is built for multiple sub-domains, all cells are globally connected. The full set of Fast operations (comprising *M2M*, *M2L* and *L2L*) will therefore be applied to all these cells by default which entails excessive unneeded process. For an optimal multizone Fast multipole code, all the FMM paths (such as *M2L* and *L2L*) connected to 'muted' cells should be blocked. This consideration is not applied for *M2M* in the *upward* pass because the parent cell is never deemed as 'muted' since it contains the elements of the studied zone as its child does. Regarding the *downward pass* and *local expansion*, this blockage helps preventing the transfer of coefficients to unwanted destinations therefore reducing notably the wasteful operations (Fig.4.6).

Multizone [Knear] storage

With respect to the storage of the matrix $[Knear]$, the compressed sparse format used in the single domain is still effective but undergoes some adjustments. This is to anticipate a general circumstance where cell C contains multiple sub-domains: the contribution and internal interactions of elements in cell C should not be mixed from one zone to another. For the purposes of maintaining the generality, the new subroutine is designed so that it can recognize the number of zones in the studied cell and allocate accordingly distinguishing CSRSYM arrays for the coefficients of each sub-domain. These contributions are separated by some markers since there is no connection between non-interfacial elements of 2 adjacent zones. When needed, the implanted markers can distinguish the segments and extract only the necessary one for the multizone matrix-vector products. This concept is illustrated in Fig.4.7.

4.3 Computational Aspects

There are two most important benefits from the symmetry of the multizone SGBEM formulation: (1) Possibility of coupling boundary and finite elements (2) a computationally efficient algorithm. As mentioned in [5], the fully symmetric scheme is more efficient than both unsymmetrical Galerkin and collocations approaches for medium to large scales since the cost of build-up and storage is basically reduced by a half. The solution time should also be halved if a direct solver is adopted.

Additionally, the present approach is very suitable for the parallelization [24]. The equation (4.7) is written for each zone of distinct material parameters. Each

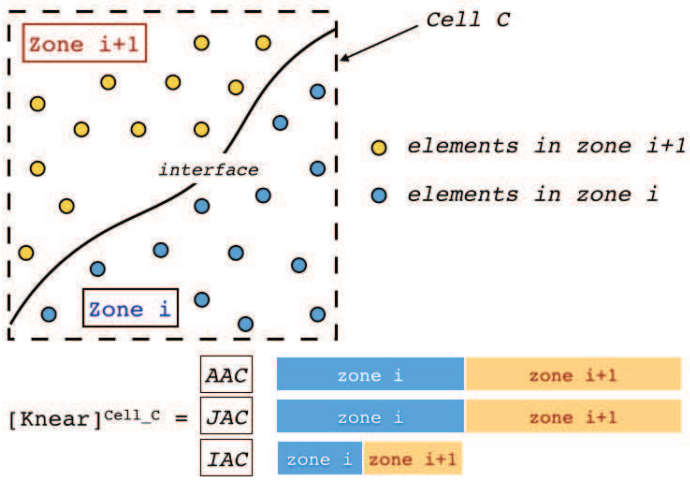


Figure 4.7: The storage of near-interactions in Cell C which contains 2 zones i and $i+1$: $[Knear]_c^{zone\ i}$ and $[Knear]_c^{zone\ i+1}$. These matrices are all concatenated in the CSRSYM arrays of cell C: AAC, JAC and IAC. In a general case, cell C can contains n zones and the CSRSYM arrays are composed of n separated segments

sub-matrix is independent from the other domains thus can be run on different processors of a parallel machine. However, this interesting feature is not investigated in this work but holds a promising perspective for further discussion.

Another technique can be adopted to improve the efficiency of the method is the rational choice of element types and the number of Gaussian points [5]: linear and coarser elements should be used to mesh the boundary portions where results are of less interest (for instance, in a fracture problem, the outer geometry should be meshed with only Q4 elements and computed with 2 gaussian points).

For certain multizone problems such as long, thin geometries, the multizone SGBEM can be very efficient because the integral equations are written over small pieces of the boundary and the resulting coefficient matrix is block-sparse (the sparsity is proportion to the connection of the internal zones). In more general multizone cases, the complexity of the problem (overall operation counts) is usually higher due to the presence of sophisticated interfaces. The portion of interface to the non-interfacial boundaries can in fact, affect considerably the efficiency of the multizone algorithm. Let us consider a simple example of 2 configurations (Fig.4.8): (a) mono-domain and (b) bi-domain. In the first geometry, the number of element in S_u and S_t are respectively n_{S_u} and n_{S_t}. In the second geometry, the boundary S_u, S_t is preserved, only the interface S_I is added (having n_{S_I} extra elements) and divides evenly the boundary S_t such that there are $n_{S_t}/2$ elements on S_t^{zone1} and $n_{S_t}/2$ elements on S_t^{zone2}.

This example is for the purpose of studying the complexity change when an interface is present in a body. The operations counts are assumed to be the element-element interactions in this analysis. Taking into account the symmetry of the

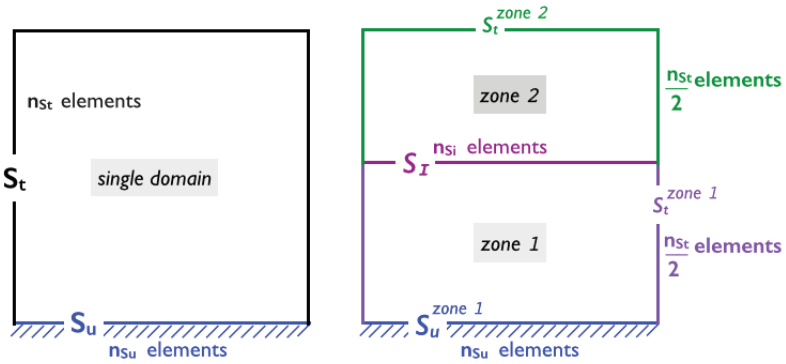


Figure 4.8: Single domain (left) and 2-zone domain (right)

SGBEM formulation, the complexity of these 2 cases is examined.

Single-Domain:

non-interface & non-interface	$B_{tt}^{S_uS_u}$	$\longrightarrow$	$n_{S_u}^2/2$
	$B_{uu}^{S_tS_t}$	$\longrightarrow$	$n_{S_t}^2/2$
	$B_{tu}^{S_uS_t}$ and $B_{ut}^{S_uS_t}$	$\longrightarrow$	$n_{S_u}n_{S_t}$

The number of operations in this first case is therefore:

$$n_{S_t}^2/2 + n_{S_u}^2/2 + n_{S_u}n_{S_t} \tag{4.10}$$

Considering the second configuration by scanning gradually over 2 zones, we get:

Zone 1:

non-interface & non-interface	$B_{tt}^{S_uS_u}$	$\longrightarrow$	$n_{S_u}^2/2$
	$B_{uu}^{S_tS_t}$	$\longrightarrow$	$n_{S_t}^2/8$
	$B_{tu}^{S_uS_t}$ and $B_{ut}^{S_uS_t}$	$\longrightarrow$	$n_{S_u}n_{S_t}/2$

interface & interface	$B_{tt}^{S_IS_I}$	$\longrightarrow$	$n_{S_I}^2/2$
	$B_{uu}^{S_IS_I}$	$\longrightarrow$	$n_{S_I}^2/2$
	$B_{tu}^{S_IS_I}$ and $B_{ut}^{S_IS_I}$	$\longrightarrow$	$n_{S_I}n_{S_I}$

non-interface & interface	$B_{tt}^{S_uS_I}$	$\longrightarrow$	$n_{S_u}n_{S_I}$
	$B_{uu}^{S_tS_I}$	$\longrightarrow$	$n_{S_t}n_{S_I}/2$
	$B_{tu}^{S_uS_I}$,$B_{ut}^{S_uS_I}$ and $B_{tu}^{S_IS_t}$,$B_{ut}^{S_IS_t}$	$\longrightarrow$	$n_{S_I}(n_{S_u}+n_{S_t}/2)$

The total operation counts in this second case (zone 1 + zone 2) is:

$$n_{S_u}^2/2 + n_{S_t}^2/4 + n_{S_u}n_{S_t}/2 + 2n_{S_I}(n_{S_u}+n_{S_t}+2n_{S_I}) \tag{4.11}$$

From (4.10) and (4.11), it is clear that with the presence of interface, the difference in operation counts is $2n_{S_I}(n_{S_u}+n_{S_t}+2n_{S_I}) - n_{S_t}^2/4 - n_{S_u}n_{S_t}/2$. Hence, in

Zone 2:

non-interface & non-interface	$B_{uu}^{S_tS_t}$	$\longrightarrow$	$n_{S_t}^2/8$
interface & interface	$B_{tt}^{S_IS_I}$	$\longrightarrow$	$n_{S_I}^2/2$
	$B_{uu}^{S_IS_I}$	$\longrightarrow$	$n_{S_I}^2/2$
	$B_{tu}^{S_IS_I}$ and $B_{ut}^{S_IS_I}$	$\longrightarrow$	$n_{S_I}n_{S_I}$
non-interface & interface	$B_{uu}^{S_tS_I}$	$\longrightarrow$	$n_{S_t}n_{S_I}/2$
	$B_{tu}^{S_IS_t}$ and $B_{ut}^{S_IS_t}$	$\longrightarrow$	$n_{S_t}n_{S_I}/2$

this example, the threshold for the multizone computation to be as efficient as the mono-domain case is when $n_{S_I} \simeq 1/10$ of $(n_{S_u} + n_{S_t})$, which denotes a very small portion of interface over non-interfacial boundaries.

In general, the above fact holds true for almost all multizone configurations: If more zones or cracks are involved, the additional complexities become far more substantial. Furthermore, the presence of internal boundaries also increases the near-interactions in the Fast Multipole algorithm which are computationally expensive. Consequently, these drawbacks slow down significantly the calculation of a multizone problem in comparison with a mono-domain problem of similar number of unknowns.

There are, however, a number of alternatives to alleviate this disadvantage: Either applying the method on long thin domains - where the contribution of 'artificial interfaces' is insignificant and the block-sparsity of the coefficient matrix can compensate for the presence of the additional boundaries [75] or using a relatively coarse mesh on interfaces of less importance. Also, special treatments could be invoked on certain circumstances to take advantages of the geometry: for instance, in a matrix-inclusion problem where each inclusion is considered as a body/interface, we exclude the inclusions from the FM algorithm and compute them by the conventional SGBEM. The combination is performed later during matrix-vector product and the algorithm has proven to be very efficient.

4.4 Numerical examples

4.4.1 Cube subjected to its body-weight

The first example involves a clamped cube subjected to its own body-weight (Fig. 4.9). In this problem, the right-handed side of the traction and displacement equations contains additional contributions due to gravitational loads:

$$\mathcal{F}_{bwt}(\tilde{\mathbf{t}}) = \int_{S_u}\int_{S} P_i(\tilde{\mathbf{x}},\mathbf{x})\tilde{t}_i(\tilde{\mathbf{x}})dS_x dS_{\tilde{x}} \tag{4.12}$$

$$\mathcal{F}_{bwt}(\tilde{\mathbf{u}}) = -\int_{S_t}\int_{S} Q_{ij}(\tilde{\mathbf{x}},\mathbf{x})\tilde{u}_i(\tilde{\mathbf{x}})dS_x dS_{\tilde{x}} \tag{4.13}$$

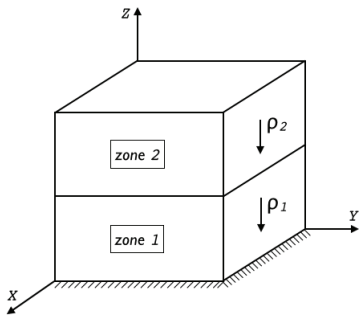


Figure 4.9: Clamped cube under gravitational load

where

$$
\begin{aligned}
P_i(\tilde{\mathbf{x}},\mathbf{x}) &= \frac{1}{8\pi\mu}[b_i n_k(\mathbf{x}) r_{,k} - \frac{1}{2(1-\nu)} b_k r_{,k} n_i(\mathbf{x})] \\
Q_{ij}(\tilde{\mathbf{x}},\mathbf{x}) &= \frac{1}{8\pi r}[n_m r_{,m}(b_i r_{,j} + b_j r_{,i}) + \frac{\nu}{1-\nu}\delta_{ij}(n_m r_{,m} b_s r_{,s} + b_m n_{,m}) \\
&\quad - \frac{1}{2(1-\nu)}[b_m r_m(n_i r_{,j} + n_j r_{,i}) + (1-2\nu)(b_i n_j + b_j n_i)]]
\end{aligned}
\tag{4.14}
$$

the gravitational load is contained in vector $b = (0, 0, -g\rho)^T$

The cube, dimension 100x100x100, is composed of two bodies which have identical material properties (for validation purposes): $E_1 = E_2 = 33$, $\nu_1 = \nu_2 = 0.1$ and $\rho_1 = \rho_2 = 0.16$. The mesh consists of 176 Q4 quadrilateral elements. Since there is no exact solution available, a finite element analysis has been carried out to obtain a reference solution.

The implemented FM-SGBEM for multi-zone problems is carried out using similar parameters input as in the mono-domain case: max_elem is 30, truncation is set to 7, the Flexible GMRES converges if the backward error is smaller than 10^{-3}. The displacement component along the vertical edge of cube is compared with the FEM reference (see Fig. 4.10). Despite the coarseness of the chosen mesh, the numerical solution gives a very good agreement with the FEM result.

4.4.2 Bi-material cantilever beam

In this example, a bi-material cantilever beam is studied. The dimensions of the beam are 40×40×200, the interface divides equally the beam into two layers. Identical materials $E_1 = E_2 = 1$ and $\nu_1 = \nu_2 = 0,3$ are chosen for these two layers to test the multizone algorithm. The beam is fixed on one end and subjected to uniform pression $p = 1$ on the top face (See Fig.4.11).

The beam is modeled with 432 Q4 elements. The solution is obtained by the multizone FM-SGBEM using similar input parameters as described in the first validation test. The results of displacement Uz on the red line is compared with

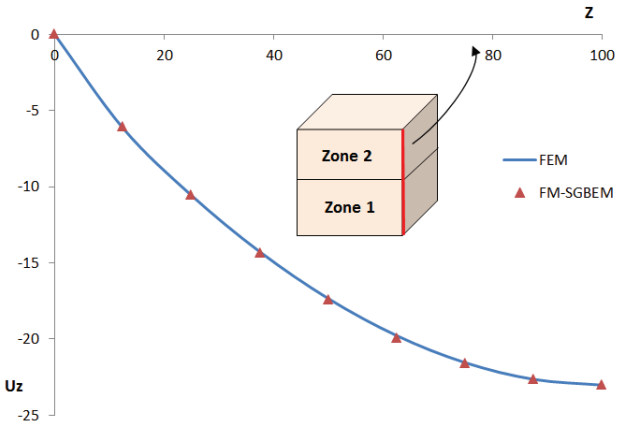


Figure 4.10: Displacement Uz of the vertical bold edge of the cube

the finite elements solution. The agreements between the BE and FE approaches are very good (See Fig.4.12).

4.4.3 Spherical envelope under internal pressure

Let us consider a spherical homogeneous envelope of internal radius a and external radius b. The constitutive material is elastic isotropic ($E = 1, \nu = 0.3$). The internal surface is subjected to a normal uniform pressure $p = 1$ (Fig.4.13):

This is a simple test in elastostaticity where we have the exact solution of radial displacement u_r:

$$u_r = \frac{a^3}{b^3 - a^3}\left[(1-2\nu)r + (1+\nu)\frac{b^3}{2r^2}\right]\frac{p}{E} \tag{4.15}$$

and the stress σ_{rr}:

$$\sigma_{rr} = \frac{a^3}{b^3 - a^3}\left[\frac{b^3}{r^3} - 1\right]p \tag{4.16}$$

In order to test the multizone code, we proceed therefore to model a spherical envelope which is composed of 3 layers. To take advantage of the exact solution and to test the validity of the program, we choose identical material properties for all 3 layers. However, the algorithm considers them as separate bodies and use the multizone scheme to solve the problem. The geometry and boundary conditions is found on Fig.4.14:

In this example, 1.047 Q8 elements have been used, constituting 13.689 unknowns (9.447 in displacements and 4.242 in traction). The FM-SGBEM program converges after 25 iterations (about 30' calculation to reach the precision of 10^{-3}). The mean values $\overline{u}_r$ and $\overline{t}_r$ are computed respectively from the radial displacement and traction of all nodes on different radius. The relative error between the numerical code and the exact solution are reported in the below table. It can be seen that

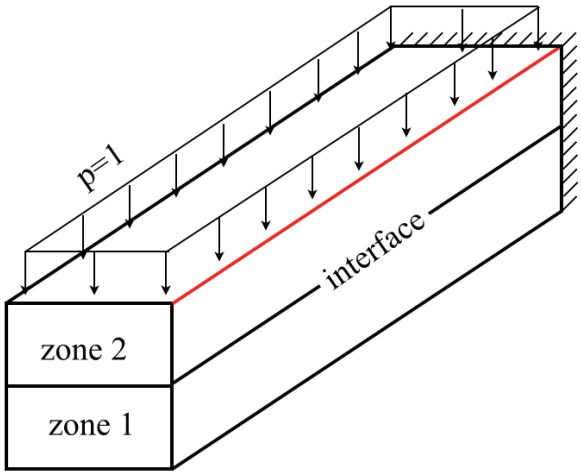


Figure 4.11: Bi-material cantilever beam

the multizone FM-SGBEM algorithm is correct and the agreement is good despite the coarseness of the chosen mesh.

position	u_r theorical	$\overline{u}_r$ numerical	relative error (%)
r_1	0.66667	0.6632	0.52 %
r_2	0.17778	0.176338	0.81 %
r_3	0.092416	0.0916	0.86 %
r_4	0.06667	0.0661	0.83 %
position	t_r theorical	$\overline{t}_r$ numerical	relative error (%)
r_2	0.111111	0.113082	2,6%
r_3	0,021752	0.022988	5,67%

Table 4.1: Displacement u_r and traction t_r on different layers

4.4.4 Fractured cylinder under tensile load

A simple bi-materials cylinder (176 Q4-elements) was used to represent a multi-domain solid in this example. The boundary conditions as well as the materials properties are shown in the Fig.4.16a. This cylinder contains in zone 2 an internal crack. In this test, we compute the displacement discontinuity on the crack. In order to do so, the fracture SGBEM formulation in chapter 2 is invoked in the multizone scheme. Two forms of crack are solved in this example: penny-shaped crack and elliptical crack. The exact solution of these two crack types are available.

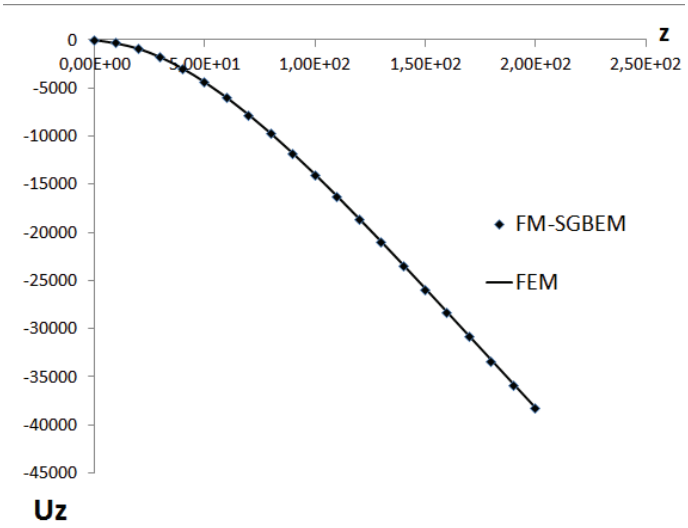


Figure 4.12: Vertical displacements on an edge of the cantilever beam

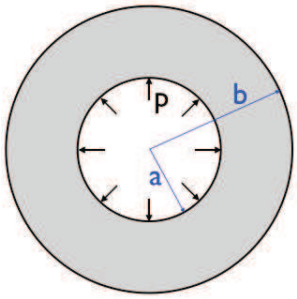


Figure 4.13: Spherical envelope under internal uniform pressure

Penny-Shaped crack

The penny-shaped crack (as is presented in chapter 3) is constituted from 48 Q8 elements, the elements adjacent to the front are modified according to the quarter-point scheme. The crack (radius $a = 1$) is put on the plane Oxy at the center of zone 2. The numerical crack opening displacement (COD) is compared with the analytical solution: $\Delta u_3 = \frac{4(1-\nu)}{\pi\mu}\sqrt{a^2 - r^2}\sigma^0_{33}$. The results are shown in Fig.4.16b. Similar tests where the crack is moved to zone 1 or oriented randomly in space are also carried out. The relative errors of these tests' results are all inferior to 1%.

The Stress Intensity Factors (SIFs) can be evaluated from the nodal displacement discontinuity of 2 nodes at the vicinity of the crack-front:

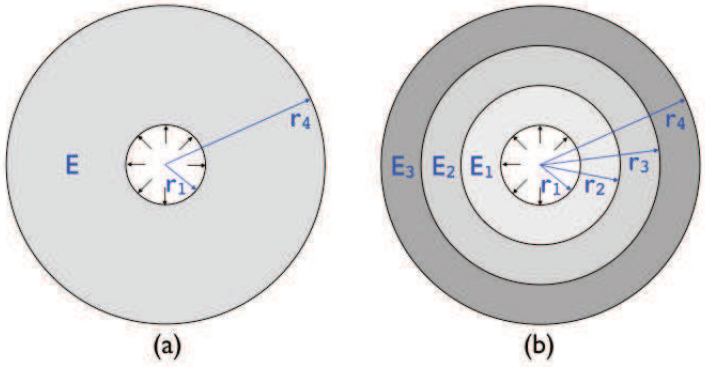
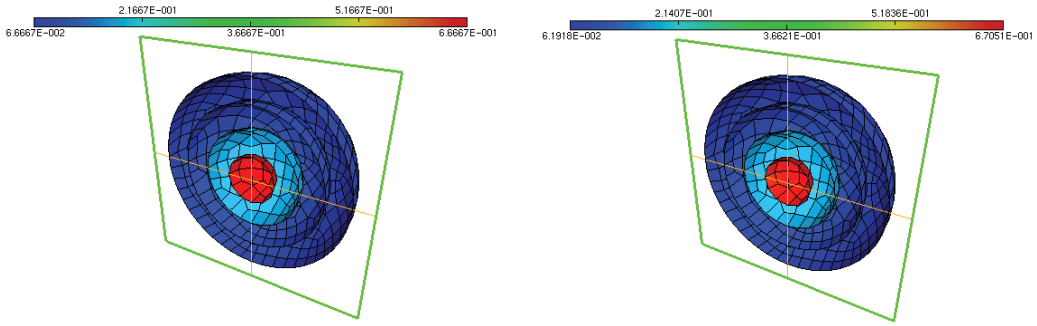


Figure 4.14: Spherical Envelope under internal pressure (a) 1-layer body $E = 1, \nu = 0.3$ (b) 3-layer body $(r_1 = 1, r_2 = 2, r_3 = 3, r_4 = 4)$ of identical properties: $E_1 = E_2 = E_3 = 1, \nu_1 = \nu_2 = \nu_3 = 0.3$



(a) (b)

Figure 4.15: Radial displacement of a internal-pressurized spherical envelope (a) exact results (b) numerical results

$$K_I^{node\ 1} = \frac{\mu}{4(1-\nu)} \left(\frac{2\pi}{a}\right)^{1/2} \left[2\Delta u_3^{node\ 2} - \frac{1}{2}\Delta u_3^{node\ 3}\right] \tag{4.17}$$

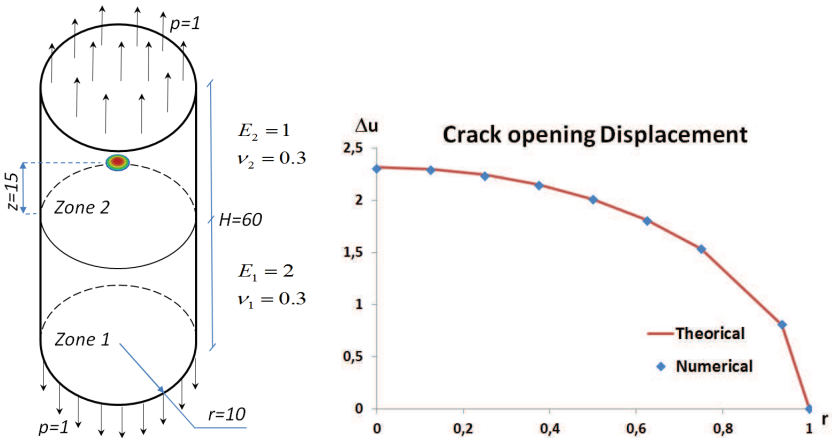
Compared with the exact SIFs $K_I = \frac{2}{\pi}\sqrt{\pi a}\sigma_{33}^0$, the numerical results exhibit a relative error of $1,65\%$

Elliptical crack

Considering now an elliptical crack of major semi-axis a and minor semi-axis b (Fig.4.17):

The analytic expression of the crack opening displacement (see [15]) is:

$$\Delta u_3 = \frac{2(1-\nu)\sigma_{33}^0}{\mu} \frac{b}{E(k)} \sqrt{1 - \frac{x^2}{a^2} - \frac{y^2}{b^2}} \tag{4.18}$$



(a) (b)

Figure 4.16: (a) bi-material cylinder (b) crack opening displacement (Δu_3)

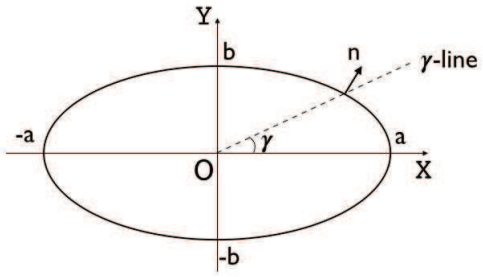


Figure 4.17: Elliptical crack geometry

where $E(k)$ is the complete elliptic integral of the second kind:

$$E(k) = \int_0^{\pi/2} \sqrt{1 - k^2 sin^2\alpha} d\alpha \qquad k^2 = 1 - \frac{b^2}{a^2} \tag{4.19}$$

$E(k)$ can be approximated either by:

$$E(k) = \left[1 + 1,464(cos\alpha)^{1,65}\right]^{1/2} \qquad k = sin\alpha \qquad \text{(accuracy 0.1\%)} \tag{4.20}$$

or

$$E(k) = \frac{\pi}{2} \frac{1}{1+\lambda} \frac{4 - 0,18\lambda^4}{4 - \lambda^2} \qquad \lambda = tan^2 \frac{\alpha}{2} \qquad \text{(accuracy 0.05\%)} \tag{4.21}$$

Equation (4.20) has been chosen in this work. The above fractured geometry is retaken into computation but the penny-shaped crack is replaced by an elliptical crack. The mesh for this elliptical crack is obtained from the same penny-shaped

crack mesh simply by contraction of the y nodal coordinates. In Fig.4.18, a comparison between exact and numerical crack opening displacement of nodes on Ox (left) and Oy (right) is shown.

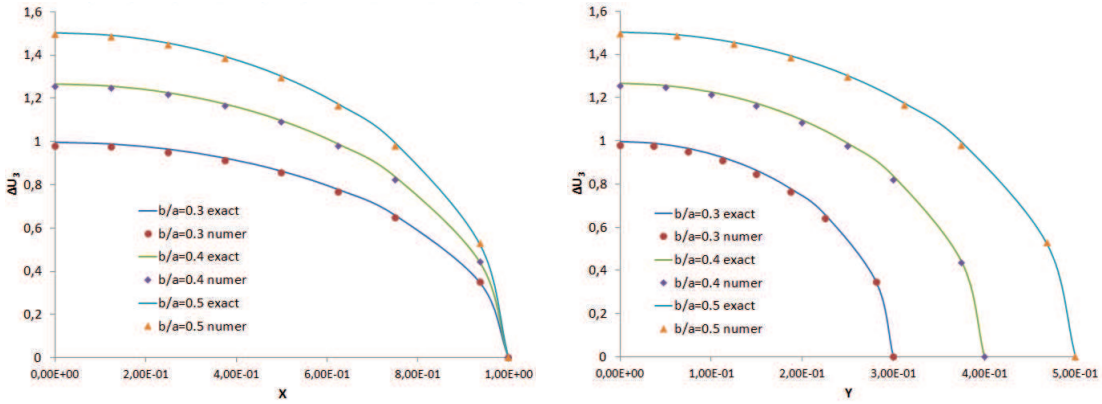


Figure 4.18: Δu_3 of an elliptical crack embedded in a bi-material cylinder

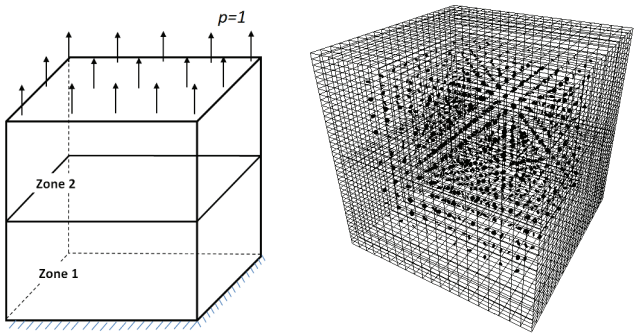
4.4.5 Mulicrack in a bounded multizone domain

In this test, we study a clamped bi-material cube which contains many cracks and is put under an uniform tensile load $p = 1$. The cube, dimension 100x100x100, is modelled with 4400 elements Q4. The interface divides the cube evenly at the altitude of $h = 50$ (see Fig.4.19a). The same material properties ($E_1 = E_2 = 2000, \nu_1 = \nu_2 = 0.3$) are defined for these two bodies but the multi-zone computation is carried out as if they were two different materials. The feature and configuration of the cracks systems are similar to those in the previous tests. The center of the crack system coincides with the cube's. We compute 3 meshes in which the system contains respectively: 1.000, 1.728, 2.744 cracks. The number of Gaussian points is chosen as 2, the parameter of truncation is set to 7. The maximal number of elements in a leaf is 30, the restart parameter is fixed at 50 and the stopping criterion for the solver Flexible GMRES is 10^{-3}.

Mesh	NEQ	*max_elem*	$\bar{l}$	Pre_Time(s)	N-iter	CPU(s)/iter	Tot_Time (s)
1	401.412	30	7	5.457	79	561	50.986
2	683.148	30	7	12.197	66	1.204	95.584
3	1.061.928	30	7	11.903	102	1.872	206.114

Table 4.2: Details of the numerical tests: $\bar{l}$ denotes the depth of the *octree* structure, NEQ is the problem size; N-iter is the iteration counts; Pre_Time and Tot_Time are respectively the preparation and total computational times.

Table 4.2 shows the details of the calculation on these 3 meshes. The CPU times per iteration are linear with the number of DOFs which corresponds very



(a) (b)

Figure 4.19: (a) Bi-material clamped cube under uniform tensile load (b) Mesh 1: cube containing 10x10x10 cracks

well to the expected performance of a FM-SGBEM algorithm. Nevertheless, by comparing the results in the unbounded problem (mono-zone) with this one in terms of speed, one can notice a considerable difference in the computation times between these two problems. This can be explained as the latter problem has to take into consideration many extra terms due to interactions between the outer and interfaces boundaries. In contrast to the unbounded fractured geometry where we compute only the term $B_{uu}^{S_c S_c}$. In the bounded configuration, all other terms in equation (4.7) including $B_{uu}^{S_c S_c}$ must also be calculated. Furthermore, with poor distribution of cracks in space, there can occur cells with highly concentrated cracks; the near interactions with these cells can cause major slow down and memory overflow to the code performance.

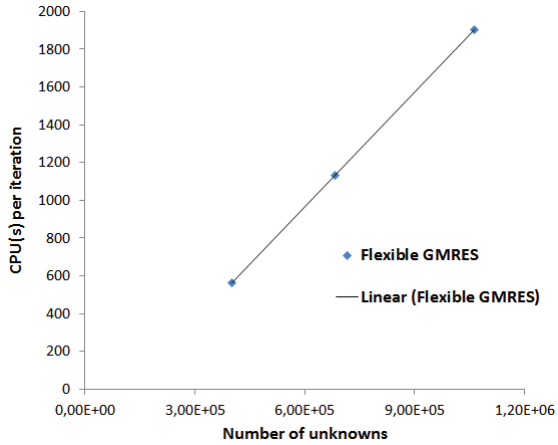


Figure 4.20: Time CPU (s) consumed per iteration

4.4.6 Extension to Matrix-Inclusion materials

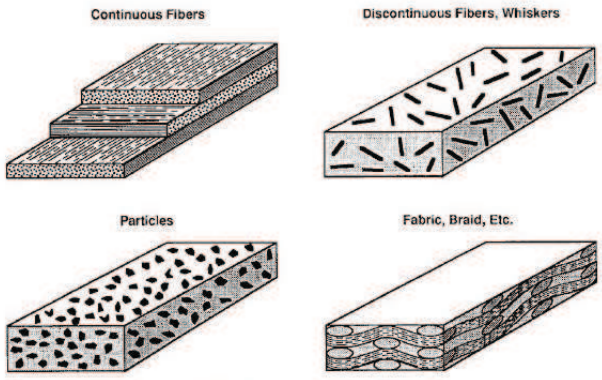
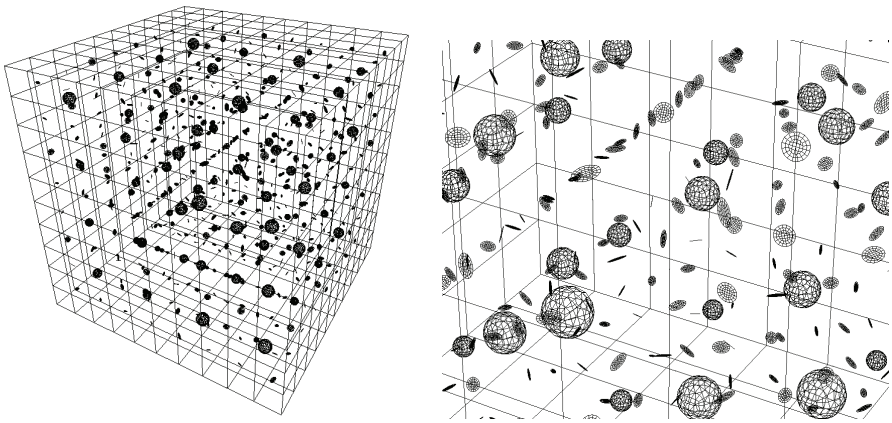


Figure 4.21: Composite materials

In this section, we discuss about the extension of the method into the matrix-inclusion materials (as known as composites - Fig.4.21). Composite materials have proven to be the subject of great interest as these materials possess better characteristics than the original components. Composite materials are seen very often in all engineering structures: from the widely-used concrete in buildings to the highly-cost fibre-reinforced polymers in the spacecrafts. The literature concerning the fracture of composite materials is rather limited and restricted to extremely idealized models ([48]- [50]). In these configurations, the model's parameters must rely on the macroscopic behaviors of the equivalent medium. Nevertheless, in certain scales and circumstances, the response of the system should not be approximated by the macroscopic parameters. To provide a better analysis on a fractured composite material, one should take the heterogeneity of the sub-domains into consideration. Exhaustive studies on the fracture composite by different numerical approaches can be found in ([51]- [58]). In the present paper, we employ the Fast Multipole-SGBEM to simply investigate the behavior of fractures (the crack opening displacement/stress intensity factors) in a model of composite. The authors aim to extend this study further to simulate a crack propagation in more sophisticated configurations.

Considering a simple configuration of a composite material (Fig.4.22): the outer geometry is a clamped cube of size a^3 under uniaxial tensile load, contains a system of n_i^3 spherical inclusions of radius r_i. These inclusions are located regularly on a cubic grid of step d_i. The solid also has an array of n_c^3 penny-shaped cracks inside. Having a unique radius of r_c, these cracks are oriented randomly in space and are also distributed on a regular cubic grid of step d_c. In a generalized manner, the program recognizes each inclusion as an independent zone and employs the equation (4.7) as well as the multizone scheme to construct and to solve the global equations system. The distances d_i and d_c are chosen such that the cracks are sufficiently far (at least four times the radius of crack) from the outer boundary and from the



(a) (b)

Figure 4.22: (a) Model of fractured composite material (4x4x4 spherical inclusions & 8x8x8 cracks) (b) Interior view of cracks and inclusions

surfaces of the inclusions (interfaces).

To ensure a good variety for this problem, different sizes, shapes and material properties are also applied for the cracks and inclusions: Two type of cracks are considered: penny-shaped crack of radius r_c and elliptical cracks of major semi-axis a_c and minor semi-axis b_c; a scaling coefficient ranged randomly from 0.5 to 1 is applied to each crack and inclusion to vary the size of these entities. While the material of solid is fixed as $E_{solid} = 1, \nu_{solid} = 0,3$, these values on inclusions are varied: $E_{inclusion} = 1 - 10$, $\nu_{inclusion} = 0,1 - 0,4$. The large-scale computations consist of important numbers of inclusions and cracks. The dimensions are chosen as: $a = 80$, $r_c = 1$, $r_i = 2$; the number of inclusion $n_i = 4$; distance between inclusions $d_i = 20$ the uniform tensile load $p = 1$. The outer boundary and inclusion are made of 600 and 151 Q4 elements respectively. The crack is meshed with 48 Q8 elements. The table below shows the details of the number of components in the solid as well as the number of unknowns and the output results:

Mesh	n_c	d_c	*NEQ*	*max_elem*	$\bar{l}$	Pre_Time(s)	N-Iter	CPU(s)/iter	Tot_Time(s)
1	8	10	258.702	30	5	4.182	16	823	18.219
2	10	7	447.558	30	8	6.165	15	1.274	26.832
3	12	6	729.294	30	8	24.112	16	2.353	63.666
4	14	5	1.122.468	30	6	15.982	14	4.404	83.180

Table 4.3: Fractured composite numerical tests by Flexible FM-SGBEM: *NEQ* denotes the problem size, N-iter is the iteration counts; Pre_time, Tot_time are respectively the preparation times and total computational times.

In order to obtain an efficient performance, the FMM algorithm is partly modified in this example. Because the FMM is not advantageous on the scales inferior to 10^4, it should not be applied on inclusions which are locally considered as a

sub-domain. The pure SGBEM is utilized on inclusions instead. Being composed of a small number of elements, the computation on each inclusion becomes thus instantaneous while having negligible additional SGBEM storage. Nevertheless, as the geometry contains more surface areas, the computation is still highly expensive. The dependence of the computational time per iteration is captured and shown in Fig.4.23.

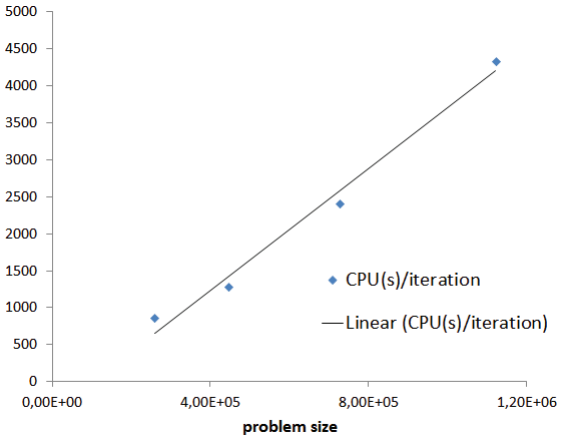


Figure 4.23: Time (s) consumed per iteration

4.5 Conclusion

In this chapter, the multizone SGBEM and the application of the Fast algorithm in this context have been presented. Firstly, the symmetric multizone formulation for fracture mechanics has been introduced for simple and general cases. Secondly, the implementation of the Fast Multipole Method for multizone SGBEM has been reported. All the important concerns regarding the *octree* generation, memory management, complexity counts, FMM operations ... have been covered in this chapter. It is effectively shown that, despite having greater complexities, the multizone FM-SGBEM is still a very attractive alternative for studying heterogeneity and interface problems because of its natural treatment of tractions continuity and its ability to couple with finite method or parallelization.

In the numerical experiments, the program has successfully solved some validation tests in elastostaticity and in fracture mechanics with excellent accuracy. The extension of the optimized algorithm to large-scale multizone fractured problems has also been introduced and reported. The performance of the code has shown that the FM-SGBEM is also a viable and efficient alternative for the solution of practical multizone problems.

Future developments of this multizone algorithm could be possibly aimed for the study of interfacial cracks or cross-interfacial cracks. Further investigations to enhance the efficiency of the method such as parallelization or improvements in

the Fast Multipole scheme should also be considered. In the next chapter, we will discuss about the crack propagation and some applications of the numerical code.

Chapter 5

Fatigue Crack Propagation

Contents

5.1 Introduction

The three-dimensional numerical modeling of fracture propagation remains a challenging issue. Generally, due to the lack of efficiency during the re-meshing phase, completely robust and automated routines have yet to be developed. This obstacle appears particularly difficult for domain-based approaches where 3D elements are employed [82]. Using a boundary-only discretisation, the remeshing task takes place only along the crack front or on the intersection of the crack and the external body, thus provides a process straightforward and more elegant.

Many developments have been devoted to improve the capacity of the boundary elements in simulating the crack-propagation. See, for example, Li and Keer, 1992 [16] where pure Mode I behavior is assumed or the subregion approach is employed. Mi and Aliabadi, 1994 [17], Mi, 1996 [14] adopted the dual approach to enforce the traction equation in addition to the traditional Somigliana displacement identity at points of the fracture surface. These approaches are limited by a strong continuity requirement set on the displacement field (C^1) at the collocation points. A more efficient treatment can be achieved with the Galerkin approach, see Yoshida et al. [4], Frangi et al. [11]. This variational method yields symmetric coefficient matrices and generally provides high accuracy and better convergence rate when iterative solvers are used. Frangi, 2002 [12] utilized the SGBEM to simulate a simple fatigue crack growth. Similar method can be found in the work of Roberts et al. [26] and Kitey et al. [27] where the crack growth occurred in particulate composites. Xu et al., 2004 [28] also applied the SGBEM to investigate the 2D

crack propagation. Tavara, on the other hand, modeled cohesive crack growth in homogeneous media [29].

The main drawback caused by the fully-populated matrix in SGBEM can be circumvented, as suggested by the authors of these previous studies, by coupling with the Fast Multipole Method. Starting from these experiences, the FM-SGBEM is here implemented to simulate a simple fatigue-crack propagation governed by Paris law. The first and second sections of this chapter introduce the criterion of the fatigue crack propagation and the subsequent remeshing strategy especially designed for a boundary element analysis. Lastly, several numerical tests for validation purposes have been performed. These tests include many fractured configurations such as the propagation of one or multiple crack(s) in homogeneous or piece-wise homogeneous domain.

5.2 Propagation criterion

Since the accurate modeling of cracks and cracks growth in three dimensional linear elastic fracture mechanics remains an open issue, a widely accepted advancement law for cracks is still elusive. In this work, a simple fatigue crack growth governed by the Paris law and independent of the fatigue ratio R has been considered. The configuration of the crack growth is shown in Fig.5.1.

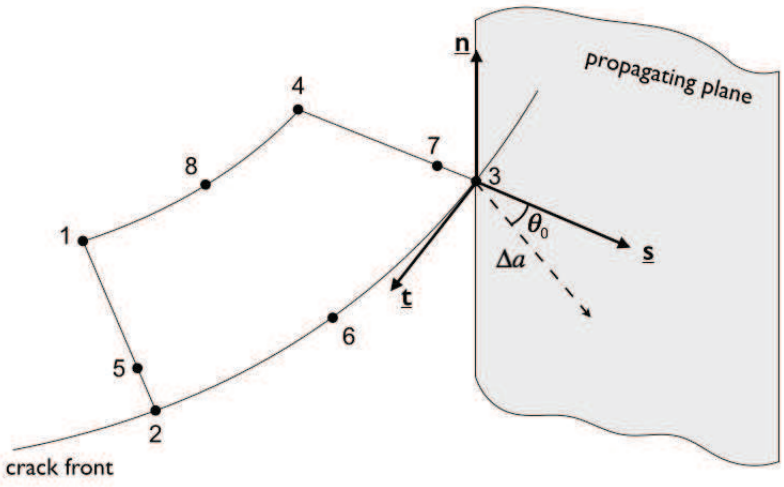


Figure 5.1: Generic local crack extension at a frontal element. Vectors **s**, **t** and **n** denote the local coordinates of node **3**. The vectors **s** and **n** compose the plane of propagation. The advancement takes place in this plane and form an angle θ_0 with the direction **s**. Δa is the length of the extension.

A few steps are required to simulate this crack growth:

(i) The displacement discontinuity field can be expressed in a local coordinate system as $\Delta\mathbf{u} = \Delta u_s\mathbf{s} + \Delta u_t\mathbf{t} + \Delta u_n\mathbf{n}$. The components Δu_n, Δu_s and Δu_t

correspond respectively to 3 modes: opening, sliding and tearing. Based on these values, the SIFs can be evaluated accordingly through extrapolation:

$$\begin{aligned} K_I &= \Delta u_n \ \frac{\mu}{4(1-\nu)}\sqrt{\frac{2\pi}{\rho}} \\ K_{II} &= \Delta u_s \ \frac{\mu}{4(1-\nu)}\sqrt{\frac{2\pi}{\rho}} \\ K_{III} &= \Delta u_t \ \frac{\mu}{4}\sqrt{\frac{2\pi}{\rho}} \end{aligned} \tag{5.1}$$

where ρ is the arc-length distance from the crack front along the $\mathbf{s}$ direction.

(ii) The frontal nodes will move in the plane perpendicular to the crack front and the propagating angle in this plane is determined by:

$$tan\frac{\theta_0}{2} = \frac{1}{4}\left[\frac{K_{Ieff}}{K_{II}} - sign(K_{II})\sqrt{\left(\frac{K_{Ieff}}{K_{II}}\right)^2 + 8}\right] \tag{5.2}$$

where $K_{Ieff} = K_I + B \mid K_{III} \mid$ is an 'effective' or 'equivalent' mode I stress intensity factor which accounts for the presence of the non-zero K_{III}; B is a material parameter.

(iii) The advancement length Δa is deduced from an application of Paris law:

$$\frac{\Delta a}{\Delta n} = C(\Delta K_{eff})^m \tag{5.3}$$

where n denotes the number of intervals; C and m are material parameters.

(iv) The propagation stops when the increments number attains a given threshold or the maximal extension size is reached.

5.3 Remeshing phase

As the nature of the method concerns only the boundary elements and no interaction between cracks and the outer boundary is present, the re-meshing task is relatively simple.

At each cycle, after determining the angle and length of the extension, a set of new nodes is added ahead of the crack front. We purposely applied the lengths Δa and $3\Delta a/4$ to generate new frontal and quarter nodes. The position of the common nodes between 2 elements is averaged to ensure the continuity from one element to another. From the new nodes, new quarter-point elements can be generated and become the new frontal elements. The quarter points of the former frontal elements are moved back to the middle of the crack edge, reverting them to standard Q8-element.

Minor attentions are expected during the renumbering of new nodes and new elements to prevent repetition. Additionally, the nodal sequence of new elements should be similar to that of the old elements to avoid conflicts in the later double integral evaluations.

5.4 Numerical examples

The algorithm of the fracture FM-SGBEM has been further developed in order to simulate the fatigue crack-propagation. There are generally two most important ingredients in an analysis of crack propagation: one is the solution of the governing equations for a fractured configuration and the other is the stress analysis and the extension of the crack geometry. While the first concern is already well taken care of by the existing FM-SGBEM, the other proves to be a trivial matter since the adopted propagation law is relatively simple. Exploiting the strategies presented in the previous sections, necessary yet straightforward steps can be designed and implemented. The general procedure on each cycle of computation can therefore be summarized in a few steps (Fig.5.2): (i) The fractured system is introduced first. (ii) The system is solved by the FM-SGBEM algorithm, producing the important results on the crack displacement discontinuities $\Delta\mathbf{u}$. (iii) Stress analysis is carried out to find θ_0 and Δa and eventually the crack extension is performed. (iv) Lastly, the newly generated geometry is added to the old system. Once all the information is updated, a new cycle can commence.

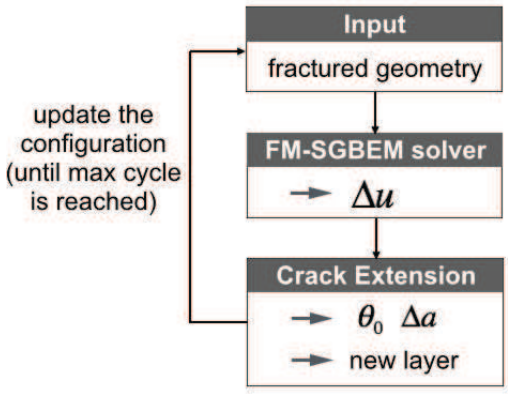


Figure 5.2: Each cycle of the fatigue-crack propagation: resolution scheme by FM-SGBEM

Similarly to the previous fracture tests, the studied crack is modeled with 48 Q8 elements. The frontal elements are quarter-point elements. In these tests, the material is elastic and isotropic. The parameters are chosen such that: the fatigue ration $R = 0$, the Young modulus $E = 10^3\ kN/cm^2$, the Poisson coefficient $\nu = 0.3$. At each cycle, the maximal crack advancement Δa_{max} is set equal to the edge length. Flexible GMRES has been used for the system resolution. The stopping-criterion is set to 10^{-3}.

5.4.1 Inclined penny-shaped crack in a bounded domain

In this first example, we consider an inclined penny-shaped crack ($r = 1\ cm$) embedded in the center of an homogeneous cylinder of dimensions $H = 60\ cm$, $R = 10\ cm$.

The cylinder is subjected to uniform tensile load $p = 10\ kN/mm^2$ at 2 extremities. The crack forms an angle $\alpha = \pi/4$ with axis-y (Fig.5.3).

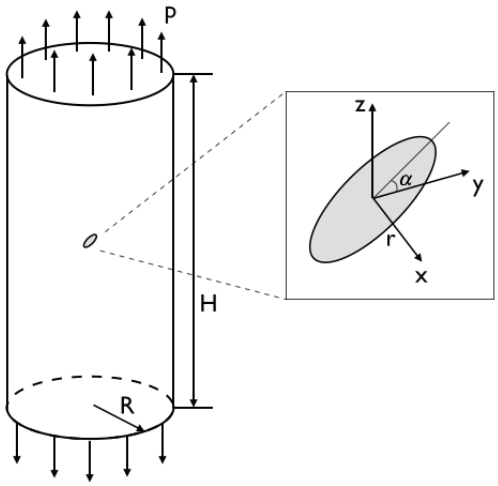


Figure 5.3: Inclined penny-shaped crack embedded in an homogeneous cylinder

The system size goes from 1.065 unknowns (initial state) to 2.361 unknowns (interval-10). The time needed for each cycle of computation increases from 7,7s (cycle-1) to 53s (cycle-10). After 10 intervals, as the natural consequence of the particular configuration, the crack evolves quickly towards the plane xOy which is perpendicular to the applied force - See Fig.5.4:

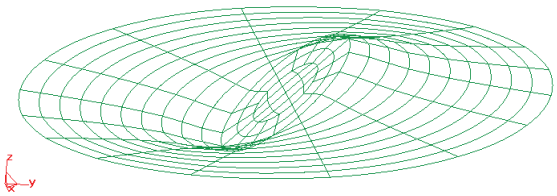


Figure 5.4: Propagation of a bounded and inclined penny-shaped crack ($\alpha = \pi/4$) after 10 intervals

To better analyze the output results, the evolution of the SIFs and the propagating angle of all nodes on the crack front over 10 increments are considered (depicted in Fig.5.5). During the course of propagation, as the crack shape approaches the horizontal plane (xOy), K_I becomes predominant, K_{II} rapidly tends to zero and K_{III} decays.

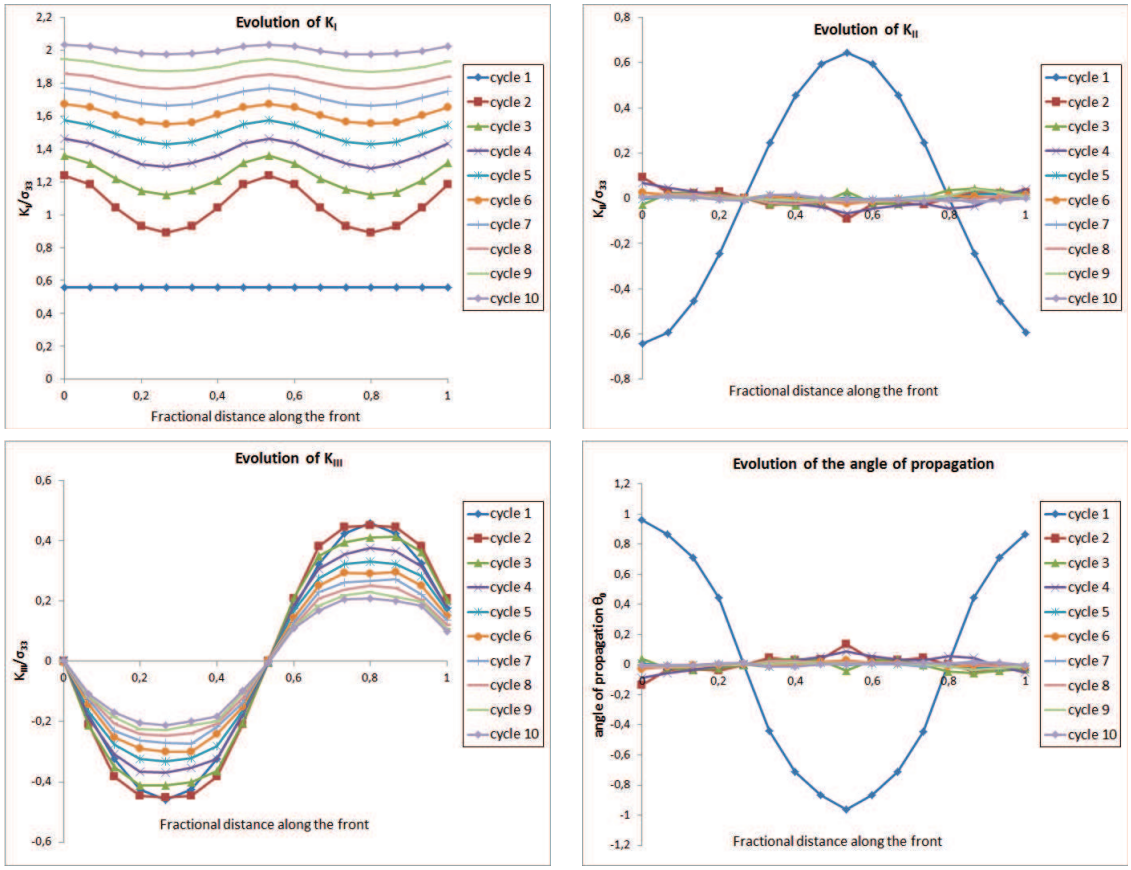


Figure 5.5: Evolution of the SIFs and θ_0 of an inclined penny-shaped crack ($\alpha = \pi/4$) during 10 cycles of propagation. The highest node on the crack-front is taken as the starting node of the sequence

5.4.2 Multiple Inclined penny-shaped cracks in a bounded domain

The second example concerns the modeling of multiple crack propagation. The outer object in this case is a clamped cube of dimensions 80x80x80, subjected to uniform tensile load $p = 10$ at the top face. The crack array, as described in the previous tests, contains n_c^3 randomly-oriented penny-shaped cracks ($r_c = 1$) 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 big to avoid influences between cracks.

The parameter $max\_elem = 30$ has been chosen constant for all cycles. The mesh density varies from 4.902 unknowns (initial state) to 16.038 unknowns (last increment). The time for each cycle of computation changes from 241s (initial state) to 1.335s (last increment). From the first to the last cycle, the convergence rate of the GMRES solver is relatively fast: 11 iterations for the initial state and increases gradually to 17 in the last increment. Fig.5.6 shows the crack array after 10 increments of propagation. As they are put under vertical tensile load, the cracks

all evolve toward the horizontal plane.

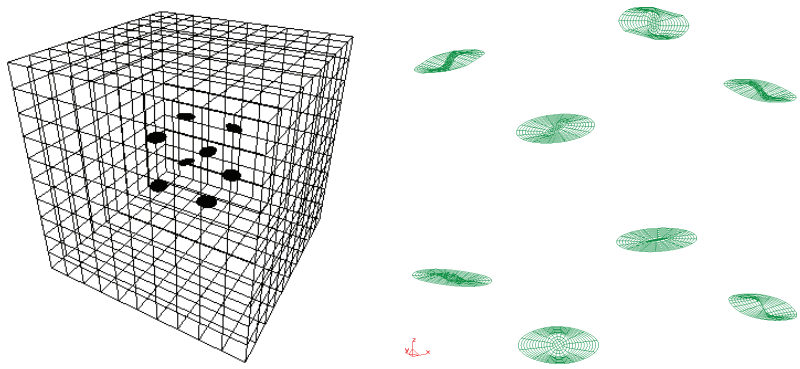


Figure 5.6: Propagation of the 2x2x2 crack-network after 10 intervals in a homogeneous clamped cube.

5.4.3 Multiple Cracks propagation in multizone configurations

This last example incorporates all the developments of the numerical code (multizone and crack propagation) to simulate the growth of several cracks in a 2-layers cubic domain. The interface divides evenly the cube into 2 horizontal layers. The dimensions and boundary conditions of the cube are taken from the cube in the previous test. From bottom to top, the material properties of the subdomains are respectively: $E_1 = 1000\ kN/cm^2, \nu_1 = 0,3$, $E_2 = 2000\ kN/cm^2, \nu_2 = 0,3$. Centered inside the solid, a system of $2 \times 2 \times 2$ penny-shaped cracks ($r_c = 1\ cm$) is generated on a cubic grid of steps $dx = dy = dz = 40\ cm$.

Fig.5.7 shows the details of the configuration and the state of the cracks after 10 intervals. Similarly to the homogeneous case, all the cracks tend to propagate toward the horizontal plane under the tensile load.

5.5 Conclusions

Several numerical examples of crack growth in the context of 3D linear elastic fracture mechanics by the FM-SGBEM have been presented. The implemented code proves reliable and very efficient. While a relatively simple crack growth law was adopted, the resulting crack trajectories showed a good correlation with the experimental outcomes. Due to time constraint, more complex fracture configurations or more sophisticated propagation criterion have not been taken into account. However, this work could very well contribute a solid basis for many areas of future researches regarding the modeling of fracture propagation, on either moderate or large-scale simulations.

Regarding the overall complexity and code performance, despite the fact that the Fast algorithm has dealt with almost all the bottlenecks of a boundary analysis, the

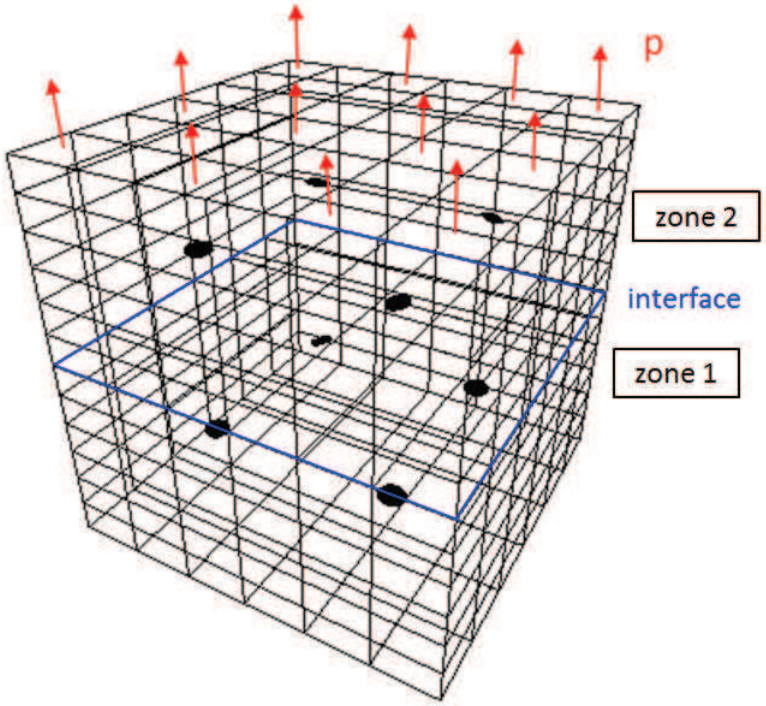


Figure 5.7: Propagation of the array of 2x2x2 cracks after 10 intervals in a 2-layered cube.

normal cost of a computation is still high. For instance, each increments featuring 10^6 unknowns can take a matter of day to finish. Realistic simulations may therefore last up to a number of weeks if higher incremental numbers are expected. Further optimization and calibration should be investigated to improve the overall efficiency of the method.

One of the first thoughts to enhance the code performance may be addressed to the construction of the coefficients matrix. During every increment of propagation, a layer of new elements is added to the geometry, the system needs to be updated and recomputed. If one rebuilds the coefficient matrix, the interaction between pairs of old elements will be repeated and will require wasteful operations. Therefore, during an increment, the old parts of the matrix should be kept constant, only the parts of the matrix that are related to the newly added elements should be computed. Once the computation of this increment terminates, these parts become the constant part of the next increment and so on (see Fig.5.8). By doing so, the cost of re-constructing the coefficient matrix should be greatly reduced and therefore accelerate the near-field evaluations by SGBEM. This idea concerns not only the Symmetric Galerkin approach but also the Fast Multipole SGBEM with similar application to the near-field coefficient matrix $[Knear]$.

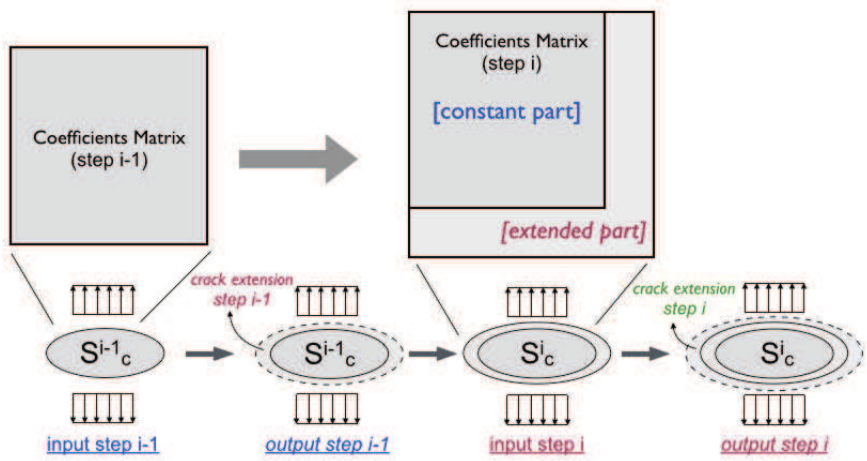


Figure 5.8: The coefficients matrix of an unbounded crack S_c computed at *step i*: the part coresponding to the input state of the crack at *step i-1* are kept constant, only the extended geometry (from *step i-1*) needs to be computed anew. These 2 parts consititute the coefficients matrix with which the system of *step i* can be solved to find the next geometrical extension.

Chapter 6

Application to Road Structures

Contents

6.1 Introduction

It is well-known that pavement systems are vital elements in the infrastructure network for all societies. Besides, they also raise technical and economical issues related to high material consumption, energy input and capital investments. An elaborate, effective and prudent pavement design is therefore composed of two factors: (1) the enhancement of the sustainability of the transportation network and (2) the most economical combination of layer thickness and material type. One of the critical elements in any pavement designs is linked to a realistic modeling of the pavement under various interior and exterior conditions. In the past, pavement design had to rely on the empirical approach that has not been able to predict performance very accurately. This is indeed a very difficult task since the pavement system is multilayered, three-dimensionnal and is composed of many materials which are nonlinear, elastic, viscous, anisotropic etc; the loadings are not usually circular or uniformly distributed, and so on. Thus, to improve pavement modeling, it is necessary to use numerical methods such as Finite Difference Method, Boundary Elements method and Finite Elements Method. As depicted in the first chapter, the BEM is well-suited to model the semi-infinite boundaries associated with layered pavement systems and has the benefit of the dimension reduction. Besides, the method is also capable of accounting for the presence of cracks and crack(s) propagation with less computational effort. The boundary discretization-based approach

therefore provides a very attractive alternative beside the 'conventional' FEM to simulate and analyze the pavement systems.

The study presented in this chapter has focused on investigating the applicability of the FM-SGBEM to predict or simulate the structural degradation of typical road structures with the presence of cracks. In order to generalize the findings of this study, large pavement examples have been modeled to include wide-base tires, different pavement structures and material properties. The analysis was carried out using a static load. The numerical results are also compared with the observation of the performance of existing pavements.

6.2 Overview of Pavement Structures

6.2.1 Pavement definition and types

Pavement is the actual travel surface especially made sustainable and serviceable to withstand the traffic load. Pavement also grants friction for the vehicles thus providing comfort to the driver and transfers the traffic load from the upper surface to the natural soil.

In earlier times before the traffic became most regular, stone paths were much familiar for carts and foot traffic load.

Nowadays, pavements are primarily used by vehicles and pedestrians. All hard road pavements usually fall into two broad categories namely

- **Flexible pavements** - Theses are pavements which leads to the deformation of subgrade and the subsequent layers to the surface. A flexible material, usually asphalt, is laid with no reinforcement or with a specialized fabric reinforcement that permits flow or repositioning of the road-bed under ground changes.

- **Rigid pavements** - The rigid characteristic of the pavement are associated with rigid or flexural strength or slab action so the load is distributed over a wide area of subgrade soil. Rigid pavement is laid in slabs with steel reinforcement.

6.2.2 Pavement constitution

In general, road structures are composed of several layers of material. Each layer receives the loads from the above layer, spreads them out, then passes on these loads to the next layer below. Thus, the further down in the pavement structure a particular layer is, the less load (in term of force per unit area) it must carry. In order to take maximum advantage of this property, material layers are usually arranged in order of descending load bearing capacity, with the highest-capacity (and most expensive) material on the top and the lowest load-bearing capacity (and least expensive) material on the bottom.

A typical pavement structure (Fig.6.1) consists of:

- **Surface course** - This is the top layer which is in contact with traffic. It provides characteristics such as friction, smoothness, noise control, rut resistance and drainage. In addition, it serves to prevent the entrance of excessive quantities of surface water into the underlying base and subbase layers.

- **Base course** - lays directly under the surface course and generally consists of aggregate (either stabilized or unstabilized). It provides additional load distribution and contributes to the drainage and frost resistance.

- **Subbase course** - intermediate layer, acts primarily as structural support

- **Subgrade** - the existing soil

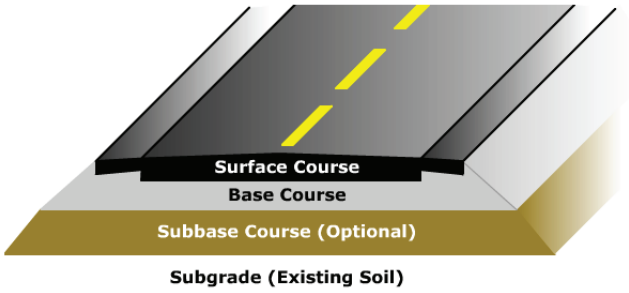


Figure 6.1: A general road structure

6.2.3 Cracking in pavements

As time passes, asphalt pavements may encounter problems that need to be addressed. Three types of degradation that an asphalt pavement may develop are cracking, distortion and disintegration. Cracking has certainly the highest influence on the service life of the pavement since it leads to water penetration, thereby weakening the bearing capacity of the pavement structure. There are various causes of cracks in pavement that include stresses from axle loads, temperature changes in the asphalt layer, or moisture and temperature changes in an underlying layer. A solid understanding of the cracking phenomenon in pavements is essential for the pavement design, performance prediction and reparation.

It has been accepted since a long time that cracking of the asphalt layer is a major mode of premature failure. Many studies have verified that pavement cracking not only occur in fatigue cracking in which a crack initiates from the bottom of the asphalt layer but also in other modes such as low temperature cracking or top-down cracking. This chapter focus on generalizing and studying the pavement structure with the presence of the transverse cracks in the asphalt concrete.

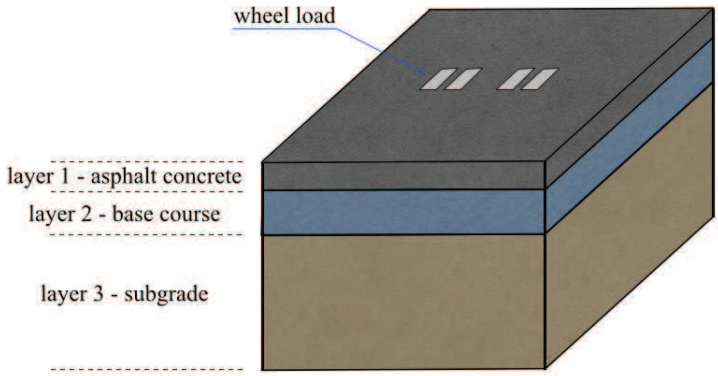


Figure 6.2: Modeling of a 3-layer pavement

6.2.4 Pavement Modeling

A 3-layered pavement structure has been analyzed in this work. This pavement is one of the four different low traffic pavement sections (from S1 to S4), with unbound granular bases, which have been tested with the LCPC accelerated pavement testing facility in Nantes which is an outdoor installation dedicated to full-scale pavement experiments. The structure S4 has been studied in this chapter. The characteristics of the layers are shown in table 6.1 and can be visualized in Fig.6.2. The mechanical characteristics of the asphalt concrete layer have been determined with laboratory tests. In *situ* measurement of the frequency (12,5 Hz) and temperature ($23°C$) at the bottom of the asphalt concrete layer lead to the proposed modulus. The unbound layers mechanical parameters have been back calculated to fit the *in situ* deflection performed at 65 KN and 43,2 km/h speed (see [84]).

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

Table 6.1: Pavement characteristics

For simplicity purposes, the contact area of the wheel and the road surface is supposed to be a rectangle of dimensions 180×300 mm, as depicted in the Fig.6.3.

6.3 Numerical Test

6.3.1 Deflection of pavement under static axle loading

This simple test in elastostatics takes advantage of the numerical solution produced by the finite element model (in code CAST3M) of Chazallon [83] to verify the exactitude of the boundary elements code before simulating the real fractured pavement

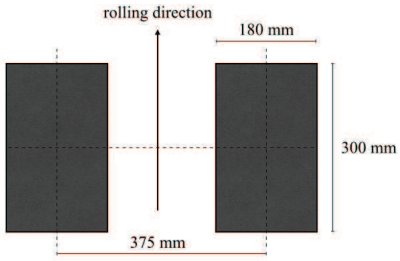
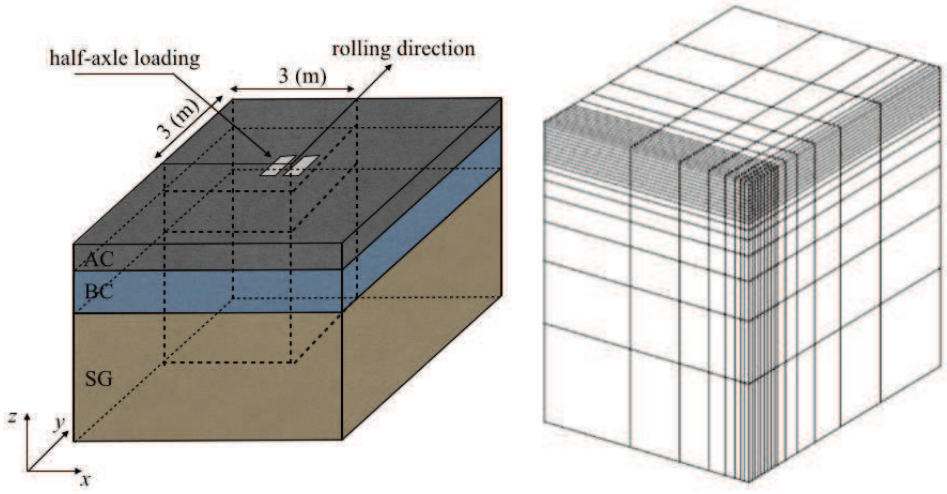


Figure 6.3: Modeling of the contact area of a half-axle loading as 2 rectangles of dimensions 180 × 300 mm (LCPC pavement testing facility). The vertical load of these wheels is 65 kN which is equal to the distributed load of $p = 0,6\ MPa$

structure. The 3-layer pavement structure previously introduced (comprising the depth and characteristics of each layer) is thus studied in this section.

The finite element model in CAST3M is of dimensions: 6000 × 6000 × 2786 mm. For efficiency purpose, the finite element mesh constitutes only a quarter of the model as to take into account the symmetry of the configuration along 2 directions x and y. The mesh contains 1000 20-nodes cubic elements, see Fig.6.4). The pavement deflection subjected to half-axle is computed.

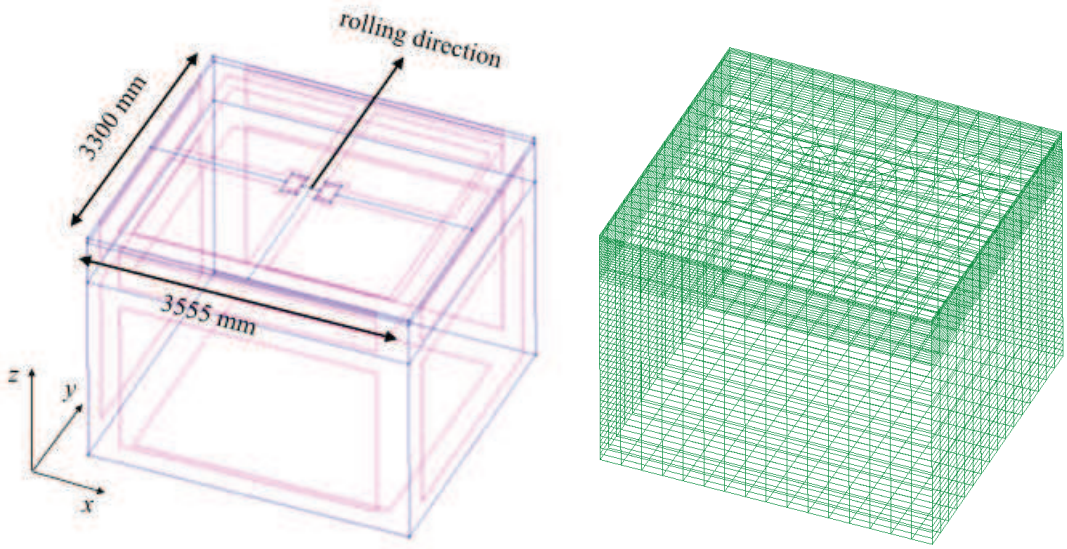


(a) (b)

Figure 6.4: Pavement simulation: (a) Full model (b) The 3D finite element mesh (a quarter of the model)

In parallel, a boundary mesh is generated to test the validity of the boundary code. Since the boundary methods treats preferably massive structures of important sizes, the symmetry of the geometry and the loading is not taken into consideration here. There is, however, an approach to take into account the geometrical symmetry which can be found in the work of Bonnet [7]. The dimensions of the boundary

element mesh are chosen as $3555 \times 3300 \times 2786$ mm (following respectively 3 directions x,y and z) which now represents the entire model (loaded by one half-axle), see Fig.6.5.This boundary mesh is composed of 4.276 four-nodes quadrilateral elements which generates overall about 11.000 unknowns in displacement and in traction.



(a) (b)

Figure 6.5: (a) BEM model of the 3-layer pavement under half-axle loading (b) Boundary mesh

Details of the computation are shown in the table 6.2 below. The computation by the FM-SGBEM code converged after 190 iterations to reach the desired precision.

Model	Solver	p	precision	num_crack	N_iter
2220_500_66	Flexible GMRES	7	10^{-3}	0	190

Table 6.2: Detail of the pavement (without crack) calculation. The model is named after the thickness (in mm) of the constittutive layers from the bottom to top (Eg. 2220_500_66 is the previously introduced structure S4). The number of cracks in the structure is described by *num_crack*; p is the truncation parameter and *N_iter* denotes the number of iterations.

The following diagrams (Fig.6.6 and 6.7) show the deflection of the models respectively across and along the rolling direction of the studied models under the effect of half-axle loading. Despite the coarseness of the boundary mesh, the exhibited results from the boundary analysis correspond very well with the output from CAST3M (the region of most important deflection - left half of the diagrams). There is, however, a noticeable dispersion between the BEM and FEM results in the right half of these diagrams. It is understandable since the boundary mesh is

chosen to be nearly twice smaller than the finite mesh.

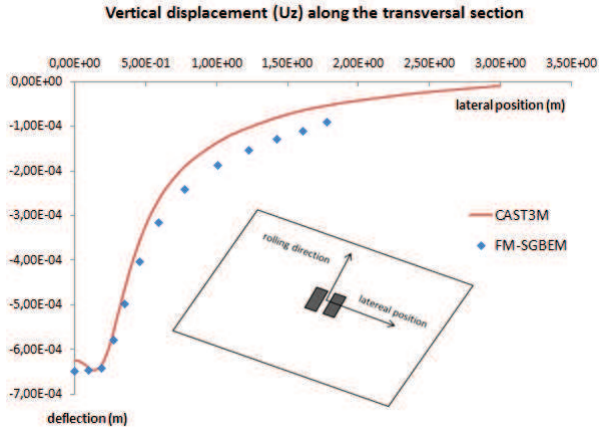


Figure 6.6: Calculated deflection of the model under the effect of half-axle loading: transverse section

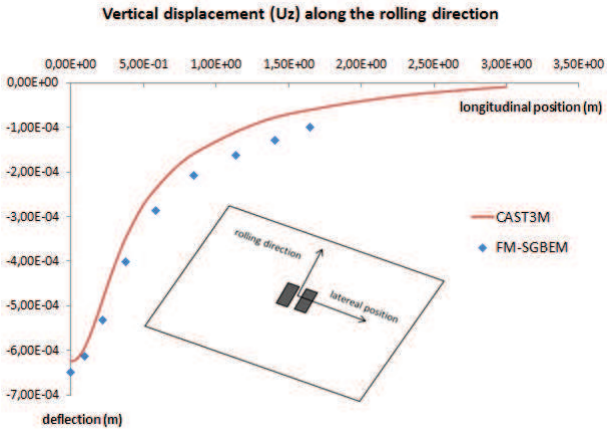


Figure 6.7: Calculated deflection of the model under the effect of half-axle loading: longitudinal section

6.3.2 Deflection of fractured pavement

In this simulation, a boundary mesh of dimensions $5355 \times 3300 \times 2786$ mm, subjected to an axle loading, has been taken into account (see Fig.6.8) to model the real 3-layer fractured pavement.

In order to account for the presence of the cracks in the surface course (Asphalt Concrete), a system of transverse cracks is generated and introduced here. Each crack is modeled as a rectangle of width h and length L (see Fig.6.9). Since the cause of transverse cracks are mainly due to traffic loads and thermal constraints,

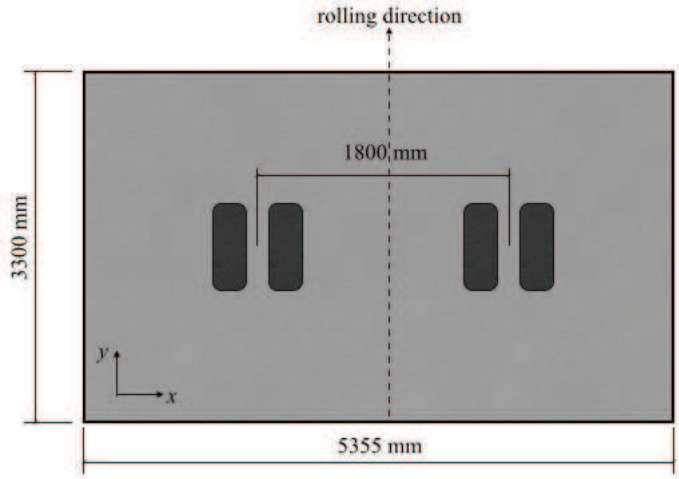


Figure 6.8: Dimensions of the pavement model on the plane xOy. The axle is located at the center of the surface course

these cracks are centered at the middle of the wheel-road contact and are distributed along the rolling direction.

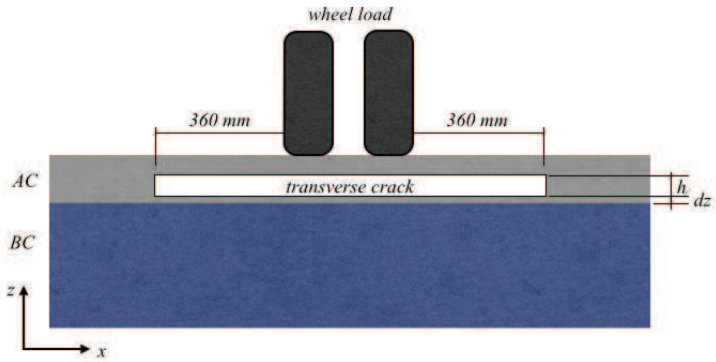


Figure 6.9: Transverse cracks in the asphalt concrete under the wheel track. h is the crack width, varies from 10 to 40 mm; $dz = 10$ mm is the distance of the cracks to the interface. The crack length is chosen as $L = 1275$ mm.

The cracks are located symmetrically under the wheel-road contact and are positioned progressively further from the contact line center (see Fig.6.10). Let $dx(i)$ denote the distance of the crack i to crack $i-1$, we have a crack distribution as in the following table 6.3. The cracks are modeled with eight-node quadrilateral elements which contain 2.000 nodes in average depending on the input crack mesh ($h = 10$-40 mm). By varying the number of cracks in the surface course, we can have different fractured states of the pavement. For instance, if 15 cracks are present (as shown in Fig.6.10), we have a highly fractured pavement.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	...
dx (mm)	40	90	150	220	300	390	490	600	720	850	990	1140	1300	...

Table 6.3: Crack steps from the center of the wheel-road contact (following the y direction).

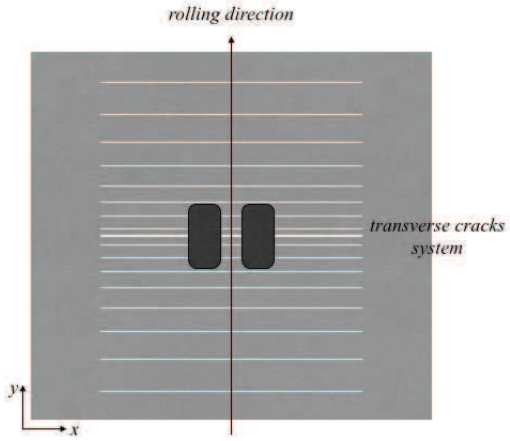


Figure 6.10: Planar distribution of the transversal crack system under half-axle loading. The bold white line denotes the transverse crack at the center of the contact surface. All the cracks are positioned symmetrically to this line. This configuration is identical for the other half-axle.

The model of a highly-fractured pavement which features more than 50 transverse cracks (about 4×10^5 unknowns) can be visualized in Fig.6.11. In this test, the parameter *max_elem* is set to 100 instead of 30 to prevent memory exhaustion due to the poor distribution of elements. We have attempted to solve this problem with the FM-SGBEM code but the calculations could not converge with the set of input parameters introduced in the previous chapters. To be precise, the maximum number of iteration is reached ($max\_N\_iter = 1.000$) before the backward error is smaller than the stopping criterion of 10^{-3}. The convergence could have been achieved if the *max_N_iter* had been set higher (Eg. 3.000 or more) but in that case, the calculation would take a large computational time and therefore would loose its applicability of calculations with larger degree of freedom.

In this configuration, there are a number of unfavorable factors that generate ill-conditioned matrix and subsequently limit the convergence rate of the code. Some of the investigations are therefore performed on the simple pavements (without crack or with a few cracks). The outcomes are discussed in the following:

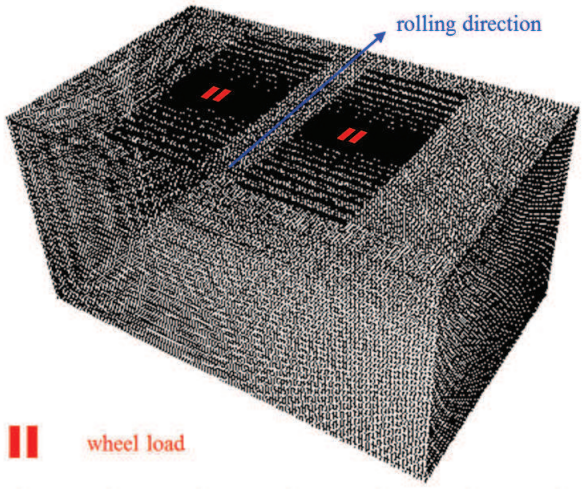


Figure 6.11: 3D view of the pavement model with the static truck load and the transverse crack systems

Thickness of the surface course

The first observed factor is the contrast of the dimensions of the layers. In the mentioned pavement structure, the surface course is the thinnest and has the largest difference in dimensions (3300 mm to 66 mm). A few tests with different thickness of the AC have been performed to see the influence of this thickness to the change in the convergence rate of the FM-SGBEM. These 3 thickness are respectively 66 mm, 120 mm and 240 mm. As shown in the table 6.4, with similar input parameters, thicker AC layer will result in a slightly faster convergence:

Model	AC thickness (mm)	N_iter
2220_500_66	66	190
2220_500_120	120	185
2220_500_240	240	180

Table 6.4: Different outcomes by different thicknesses of the AC - pavement without crack.

Contrast of the materials' stiffness

The stiffness of the layers is also believed to be one of the unfavorable factor that have caused the ill-conditioned matrix. Similar tests on different ratios of stiffness has been performed. The largest ratio is the case of the real structure, this quantity is decreased progressively toward the case of identical materials. Table 6.5 shows the results of these tests and indicates that smaller ratios lead to faster convergences.

Model	$E_3 - E_2 - E_1$	N_iter
2220_500_66	80-180-6110	190
2220_500_66	300-600-4000	148
2220_500_66	180-180-180	96
2220_500_66	6110-6110-6110	49

Table 6.5: Different outcomes by different ratios of the layers' stiffness - pavement without crack.

High concentration of elements

In the fractured model, all the cracks are located inside a very thin layer and therefore create a region of very high concentration of entities (elements/nodes/degrees of freedom). We have tried to reduce this effect by either decreasing the number of cracks or increasing the distance between them. Models featuring thicker AC (presented above) are also included in this investigation. Different outcomes are obtained. With a few number of cracks, the thin AC structure (2220_500_66) still could not converge while convergences have been reached with the thicker AC layers. For example, the structure 2220_500_120 containing one rectangle crack ($h = 40, L = 1275\ mm$) at $dz = 10\ mm$ converged after 96 iterations (27s/iteration) in 3.061 s, the structure 2220_500_240 featuring up to 3 penny-shaped crack (radius r=30 mm, distance between cracks 30 mm) at the center of the AC layer converged after 183 iterations (18s/iterations) in 3.770 s.

Remarks

From all the investigations, we can deduce that the input parameters (model dimensions, materials stiffness, crack distribution ...) affect clearly the convergence rate of the FM-SGBEM code and consequently limit our simulations to a small scale (about 15×10^3 unknowns) which feature relatively simple fractured state. Due to time constraint, further investigations could not be carried out and we have not reached the optimal solution for this test.

6.4 Conclusions

In this chapter, the Symmetric Galerkin approach coupled with the Fast Multipole Method has been applied to pavement study. We have attempted to compute the deflection of the pavement under axle-load. The small-scale simulations (without or with a small number of cracks) have provided satisfactory results whereas larger-scale configurations could not be achieved. Some investigations have proved that the input configurations dispose of unfavorable factors for the numerical calculations. The performance and stability of the FM-SGBEM code require therefore further investigations combined with some optimization techniques to overcome the convergence difficulty to reach the goal of simulating large-scale fractured pavements. Also, more realistic factors should be embedded in the code to render the model

closer to the reality.

Chapter 7

Conclusions and Future Research

Contents

7.1 Conclusions and Discussions

The extension and application of the Fast Multipole SGBEM to the context of multizone, multicrack in linear fracture mechanics have been successfully carried out in this thesis. In order to solve efficiently some practical issues of engineering interests, a great effort has been devoted first hand in *chapter 3* to the optimization of the algorithm. Multiple strategies have been proposed and implemented. The most considerable improvements are the compressed storage of the near-field coefficient matrix $[Knear]$ and the concept using this compressed matrix as the preconditioner of the nested solver namely Flexible GMRES. The enhanced code has been run on various large-scale tests ($N = O(10^6)$) and has proved to be very robust and of excellent accuracy.

As the code has become more efficient and reliable, more complex fractured configurations should be considered. In *chapter 4*, the FM-SGBEM has been extended to multi-region problems. By adopting a technique of appropriate terms-arrangement, the global matrix is rendered symmetric during the assembling phase thus conserves well the efficiency feature. Different numerical aspects and attentions during the implementation of the multizone SGBEM and multizone FM-SGBEM are well discussed. The good precision of the developed code has been shown in a few numerical verification tests incorporating a number of practical solicitations such as *bending* or *body-weights*. The successful implementation of the FM-SGBEM in treating multi-region problems has opened many interesting areas of studies such as the application in multilayered road structure or the study of composite materials ... The later case has been generalized and presented in this chapter as a fractured matrix-inclusion material. Even though the large-scale test converges rather rapidly, it still requires considerable overall computational times. Future developments are expected to boost further the performance and make the FM-SGBEM an formidable option in treating specific problems for composite materials.

Chapter 5 introduces another interesting application of the FM-SGBEM in the fracture context concerning the simulations of crack-growth. For this matter, the boundary analysis shows clearly its flexibility and versatility during the process of remeshing. The newly added elements are generated and simply added to the crack-front following the computed angle without affecting the initial mesh thus can avoid immense mesh-data modifications. Beside the remeshing, the modeling of a propagation involves also the recomputation of the system at every step which requires imperatively a robust and efficient algorithm. Unlike the limited ordinary BEM, the FM-SGBEM is qualified in every aspect: accurate, straightforward and efficient. Three-dimensional simulation of non-planar crack(s)-growth has been presented here. Good correlations have been found between the obtained numerical results and the established references. Due to the time constraint, only simple configurations of crack propagation under fatigue *Paris* law have been taken into account. However, the correct numerical results encourages the developments and extensions of the algorithm in more complex geometries with more sophisticated criteria.

Chapter 6 presents the application of the FM-SGBEM in simulating road structures (pavements). Since the structure of pavements is similar to all the studied models in this work which are multi-layered and multi-fractured, the implementation and calculation have been carried out rather simply. The behaviors (deformations) of the asphalt surface (either fractured or not) under vehicle loads are computed. A great number of tests have been carried out and the simulations are reported to be unstable and have poor convergence rate. This event may happen due to the significant contrast in the stiffness of the constitutive materials. Besides, the extreme thinness of the surface course compared to its other two dimensions is also believed to be the cause of the ill-conditioned matrix which slow down considerably the iterative solution. Large-scale pavement simulations have therefore not been accomplished in this work. Nonetheless, early results on moderate size pavement meshes have been achieved and exhibit very good agreements with the provided references. Further investigations and refinements should be able to run large-scale pavement simulations by the FM-SGBEM

7.2 Directions for future works

Along with the development of the Boundary analysis, the models are getting more complex and featuring greater sophisticated issues. Ordinary and simple algorithms can no longer answer to such rising demands. The numerical work developed during this thesis has somehow provided a robust algorithm in treating various large-scale problems of multizone and multicrack in the context of linear fracture mechanics. This is but a small piece in the large picture of the study of fractures. There are a great number of another aspects and factors that must be accounted for if one wants the numerical approach to get closer to the real-behavior of this phenomenon.

As mentioned many times during the thesis, great efforts should always be spent

to enhance the efficiency of the method. Similarly in the work of many other authors, the improvement of the method performance stays at their top priority. The adaptation and extension of the method to cope with some particular circumstances might come at the second place. Some thoughts on these concerns are discussed below:

Parallelization

The algorithm of the FM-SGBEM in this thesis has been implemented for single processor platform. The adaptation of the code in a multi-core environment is expected to boost greatly its performance. This idea comes from observing the hierarchical structure of the FMM: the system is solved by looping on all the *octree* cells, instead of putting all the computations on one processor, different groups of cells can be associated with different and independent processors. The same concept could also be applied in the multizone FM-SGBEM: different zones can also be computed simultaneously thus reduces significantly the total computational time. Even though the parallelization has been adopted in a few publications concerning the FMM, the actual task is deemed rather difficult and considerable effort should be spent along this line.

Other refined opitimizations

The preconditioning strategy introduced in thesis has somewhat reduced greatly the iterative solution phase thus has contributed to the overall efficiency. However, the cost of calculating the near coefficients is still very high. Works on the reformulation of the integral operators or different approaches that take into account the mathematical properties of the continuous operators are expected to enhance further the algorithm performance. Also, some special techniques which make use of the direct solver without exhausting the computational cost have also been reported recently to achieve considerable improvements.

Coupling with other numerical methods

Another interesting alternative for the future work is the coupling of the FM-SGBEM with the well-known FEM. One viable application of such coupling should be the model of soil-structure interactions. The coupled method has the advantage of dealing with all distinct difficulties of each method. One such BEM-FEM model, in example, can use the BEM mesh to deal with the singularities and the FEM mesh to correctly represent the non-linearity of materials.

For final remark, the author hope that this thesis can somehow contribute to the growth of the BEM community and that in the near future, the FM-SGBEM in particular can become a practical tool for solving greater ranges of engineering applications.

Appendix A

Integration techniques

A.1 Gaussian quadrature

In numerical analysis, a quadrature rule is an approximation of the definite integral of a function, usually stated as a weighted sum of function values at specified points within the domain of integration. The rule is constructed to yield an exact result for polynominals of degree $2n-1$ or less by a suitable choice of point x_i and weights w_i for $i = 1, 2, ..., n$. For a simple domain taken as $[-1, 1]$, the rule is written as:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{A.1}$$

The abscissas and weights of different numbers of gaussian points are:

n	Abscissa (x_i)	Weight (w_i)
1	0	2
2	-0,577350269	1
	0,577350269	1
3	-0,77459666	0,555555556
	0	0,888888889
	0,77459666	0,555555556
4	-0,861136311	0,347854845
	-0,339981043	0,652145154
	0,339981043	0,652145154
	0,861136311	0,347854845
5	-0,906179846	0,236926885
	-0,53846931	0,478628670
	0	0,568888889
	0,53846931	0,478628670
	0,906179846	0,236926885

Change of interval

An integral over $[a, b]$ must be change into an integral over $[-1, 1]$ in order to apply the Gaussian quadrature rule. This change can be done in this way:

$$\int_{a}^{b} f(x)dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}z + \frac{b+a}{2}\right) dz \tag{A.2}$$

After applying the gaussian quadrature rule, the following approximation is:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}z_i + \frac{b+a}{2}\right) \tag{A.3}$$

Gaussian quadrature rule in 2D

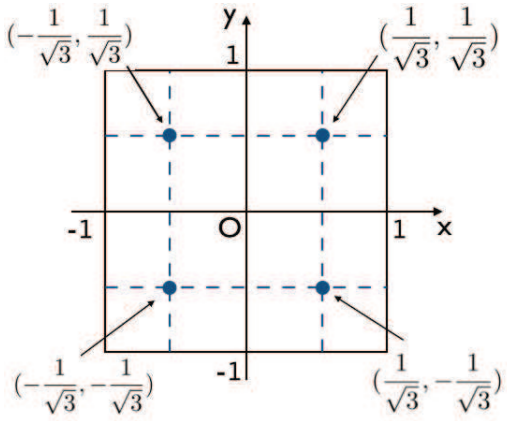


Figure A.1: Gauss points in the intervals $[-1,1] \times [-1,1]$

The 2D integration is simply computed as repeated one-dimensional integrals:

$$\begin{aligned} I &= \int_{-1}^{1}\int_{-1}^{1} f(x,y)dxdy \\ &\approx \int_{-1}^{1}\left(\sum_{i=1}^{n} w_i f(x_i,y)\right)dy \\ &\approx \sum_{j=1}^{n}\sum_{i=1}^{n} w_j w_i f(x_i,y_j) \end{aligned} \tag{A.4}$$

Example in 2D with the number of gaussian points $n = 2$. As we know the abscissas and weighted in 1D for $n = 2$ are respectively $x_i = \pm\frac{1}{\sqrt{3}}$, $w_i = 1$, the gaussian quadrature rule for $f(x,y)$ is:

$$f(x,y) \approx 1.1.f(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}) + 1.1.f(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}) + 1.1.f(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}) + 1.1.f(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$$

A.2 Singular Integration

An efficient approach to double area integration of weakly singular kernels is based on coordinate transformations, which produce a jacobian that can cancel the weak singularity of the kernel. Such technique has been developed and reported in [Frangi]. Singularity occurs when 2 elements share one or more nodes. There are

3 cases respectively: *common vertex*, *common edge* and *coincident*. To deal with the singularity, the four-dimensional integration $0 \leq \psi_1, \psi_2, \eta_1, \eta_2 \leq 1$ is divided into several integration sub-domains. In each sub-domain, a special coordinate transformation is introduced accordingly.

The numerical computation of a general singular integration is expressed as follow:

$$I = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \sum_{isub=1}^{nsub} f(\mathbf{x}(\xi_1, \xi_2)) K(\mathbf{x}(\xi_1, \xi_2), \mathbf{y}(\eta_1, \eta_2)) g(\mathbf{y}(\eta_1, \eta_2)) J_{isub} d\omega_1 d\omega_2 d\omega_3 d\omega_4 \tag{A.5}$$

where $nsub$ is the number of integration sub-domains; $\xi_1, \xi_2, \eta_1, \eta_2$ are local coordinates in sub-domain $isub$ which are expressed through integration variables $0 \leq \omega_1, \omega_2, \omega_3, \omega_4 \leq 1$ and J_{isub} is the sub-domain transformation Jacobian. Three cases of singularity are presented below:

Common vertex

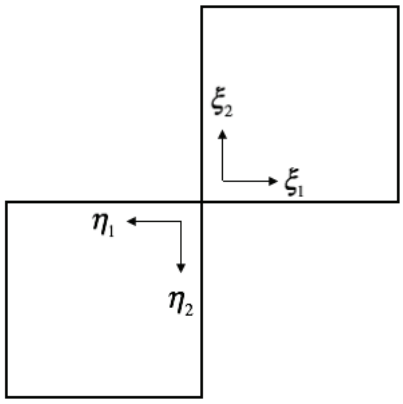


Figure A.2: Elements with common vertex

Number of sub-domains $nsub = 4$

$isub$	ξ_1	ξ_2	η_1	η_2
1	v_2	v_3	v_4	v_1
2	v_3	v_4	v_1	v_2
3	v_4	v_1	v_2	v_3
4	v_1	v_2	v_3	v_4

Variables transformations and Jacobian:

$$
\begin{aligned}
v_1 &= \omega_1 \\
v_2 &= \omega_1.\omega_2 \\
v_3 &= \omega_1.\omega_3 \\
v_4 &= \omega_1.\omega_4 \\
J_{isub} &= \omega_1^3
\end{aligned}
$$

A.2.0.1 Common Edge

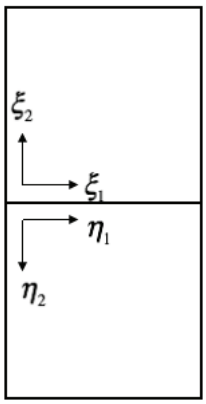


Figure A.3: Elements with common edge

Number of sub-domains $nsub = 6$

$isub$	ξ_1	ξ_2	η_1	η_2
1	v_4	v_2	$v_1 + v_4$	v_3
2	$v_1 + v_4$	v_2	v_4	v_3
3	v_5	v_1	$v_2 + v_5$	v_3
4	$v_2 + v_5$	v_1	v_5	v_3
5	v_5	v_3	$v_2 + v_5$	v_1
6	$v_2 + v_5$	v_3	v_5	v_1

Variables transformations and Jacobian:

$$
\begin{aligned}
v_1 &= \omega_1 \\
v_2 &= \omega_1.\omega_2 \\
v_3 &= \omega_1.\omega_3 \\
v_4 &= \omega_4.(1-\omega_1) \\
v_5 &= \omega_4.(1-\omega_1\omega_2) \\
J_{1,2} &= \omega_1^2(1-\omega_1) \\
J_{3,4,5,6} &= \omega_1^2(1-\omega_1\omega_2)
\end{aligned}
$$

A.2.0.2 Coincident

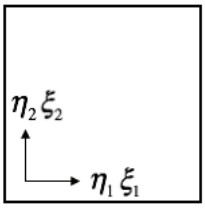


Figure A.4: Coincident elements

Number of sub-domains $nsub = 8$

$isub$	ξ_1	ξ_2	η_1	η_2
1	v_3	v_4	v_1+v_3	v_2+v_4
2	v_3	v_2+v_4	v_1+v_3	v_4
3	v_1+v_3	v_4	v_3	v_2+v_4
4	v_1+v_3	v_2+v_4	v_3	v_4
5	v_4	v_3	v_2+v_4	v_1+v_3
6	v_4	v_1+v_5	v_2+v_4	v_3
7	v_2+v_4	v_3	v_4	v_1+v_3
8	v_2+v_4	v_1+v_3	v_4	v_3

Variables transformations and Jacobian:

$$
\begin{aligned}
v_1 &= \omega_1 \\
v_2 &= \omega_1.\omega_2 \\
v_3 &= \omega_3(1-\omega_1) \\
v_4 &= \omega_4.(1-\omega_1\omega_2) \\
J_{isub} &= \omega_1(1-\omega_1)(1-\omega_1\omega_2)
\end{aligned}
$$

Appendix B

FMM developments for SGBEM terms

B.1 Terms $\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}})$ and $\mathbf{B}_{uu}(\mathbf{u}^{\mathbf{D}},\tilde{\mathbf{u}})$

Paying attention to 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 $\mathbf{B}_{uu}(\mathbf{u}^{\mathbf{D}},\tilde{\mathbf{u}})$ can be written as:

$$\begin{aligned}\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}}) &= \int_{S_t}\int_{S_t}(Ru)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \\ &= \int_{S_t}\int_{S_t}(R\tilde{u})_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(Ru)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \qquad \text{(B.1)}\\ \mathbf{B}_{uu}(\mathbf{u}^{\mathbf{D}},\tilde{\mathbf{u}}) &= \int_{S_t}\int_{S_t}(Ru^D)_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(R\tilde{u})_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \\ &= \int_{S_t}\int_{S_t}(R\tilde{u})_{iq}(\mathbf{x})B_{ikqs}(\mathbf{r})(Ru^D)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}}dS_x \qquad \text{(B.2)}\end{aligned}$$

The term $\mathbf{B}_{uu}(\mathbf{u},\tilde{\mathbf{u}})$ is chosen to evaluate. The formula 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 formulas (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 formula 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:

$$M^{1uu}_{ks,n,m}(O) = \int_{S_t} R_{n,m}(\overrightarrow{O\tilde{x}})(Ru)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}} \tag{B.12}$$

$$M^{2uu}_{sn,m}(O) = \int_{S_t} R_{n,m}(\overrightarrow{O\tilde{x}})(\overrightarrow{O\tilde{x}})_k(Ru)_{ks}(\tilde{\mathbf{x}})dS_{\tilde{x}} \tag{B.13}$$

Analogously 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 similar 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^{th} 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:

$$\begin{aligned}\mathbf{B}_{ut2}(\mathbf{u^D}, \tilde{\mathbf{t}}) =& -\int_{S_u}\int_{\partial\Omega}\tilde{t}_k(\tilde{\mathbf{x}})u^D_i(\tilde{\mathbf{x}})\frac{1}{8\pi\mu}C_{ijab}\frac{\partial}{\partial x_b}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left(\overline{F^{Stt}_{ka,n,m}(\overrightarrow{\mathbf{Ox}})}\right.\\ &\left.+\overline{G^{Stt}_{k,n,m}(\overrightarrow{\mathbf{Ox}})}(\overrightarrow{\mathbf{Ox}_a})\right)R_{n,m}(\overrightarrow{\mathbf{Ox}})n_j(\mathbf{x})dS_x dS_{\tilde{x}}\\ =& \frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_k(\tilde{\mathbf{x}})u^D_i(\tilde{\mathbf{x}})\left(\overline{F^{Stt}_{ka,n,m}(\overrightarrow{\mathbf{Ox}})}M^{1ut2}_{ia,n,m}(\mathbf{O})\right.\\ &\left.+\overline{G^{Stt}_{k,n,m}(\overrightarrow{\mathbf{Ox}})}M^{2ut2}_{in,m}(\mathbf{O})\right)dS_{\tilde{x}}\end{aligned} \tag{B.55}$$

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 help of $L^{1ut2}_{ia,n,m}$ and $L^{2ut2}_{i,n,m}$:

$$\mathbf{B}_{ut2}(\mathbf{u^D},\tilde{\mathbf{t}}) = -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{S_u}\tilde{t}_k(\tilde{\mathbf{x}})u^D_i(\tilde{\mathbf{x}})\left[F^R_{ka,n,m}(\overrightarrow{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}})L^{1ut2}_{ia,n,m}(\tilde{\mathbf{x}_1}) + G^R_{k,n,m}(\overrightarrow{\tilde{\mathbf{x}}_1\tilde{\mathbf{x}}})L^{2ut2}_{in,m}(\tilde{\mathbf{x}_1})\right]dS_{\tilde{x}} \tag{B.64}$$

Appendix C

List of subroutines in the Program

In this section, the name and functionality of the subroutines in the Fortran program is given. This is to facilitate the comprehension of the thesis in case a subroutine is mentioned. Most of these are developed in-house by the authors.

1. **Fracture_FMSGBEM** main program, containing all the subroutines
2. **read_parameters** import the parameters for the Octree generation (*max_elem*) and for the Solver (*truncation*, *precision*, *restart_parameter*, *maximum_iteration*, *number of Gaussian points*)
3. **read_slab** import the geometry of slab (nodal coordinates and connectivity matrix)
4. **read_crack** import the geometry of crack (nodal coordinates and connectivity matrix)
5. **read_inclusion** import the geometry of inclusion (nodal coordinates and connectivity matrix)
6. **conditionlimit** modify the crack following the quarter-point scheme
7. **multicrack** multiply, orientate and distribute the cracks in space
8. **generate_inclusion** multiply, orientate and distribute the inclusions in space
9. **join_data** concatenate the geometries and form the global arrays; allocate some arrays that control the zones and element type
10. **unkn_manager** allocate arrays for the known and unknown vectors; estimate the problem size
11. **generate_gauss** compute the coordinates and weights of the gaussian points
12. **generate_octree** build the *octree* structure
13. **find_adjac_elem** create a list of adjacent elements
14. **eval_baric** compute the center of all elements

15. **allocate_multipole** allocate the arrays for multipole moments and local expansions

16. **clean_multipole** set all the values of the multipole moments and local expansions to zero

17. **int_direct_single** compute the single integral

18. **int_lu** compute the single integral

19. **upward_0** execute the *upward pass* of the FMM operation (input by known values on boundaries)

20. **int_mpl_0** compute the multipole moments

21. **downward_0** execute the *downward pass* of the FMM operation

22. **int_direct_0** loop on all cells, compute matrix $[Knear]$ and right-hand side vector $\{b^{near}\}$; expand the far-away interaction to the leaf-cells and compute the portion $\{b^{FMM}\}$; store matrices $[Knears]$ under CSRSYM format

23. **csrsym** convert a normal matrix to the compressed sparse format

24. **solve_gmres** activate the solution by the iterative solver (either by GMRES or Flexible GMRES)

25. **upward_1** execute the *upward pass* of the FMM operation (input by the candidate vector)

26. **int_mpl_1** compute the multipole moments

27. **downward_1** execute the *downward pass* of the FMM operation

28. **int_direct_1** loop on all cells, compute a part of the product by the far-interactions $[K^{FMM}]$; invoke the matrix $[Knear]$ and proceed to multiply it with a candidate vector; finalize the matrix-vector product

29. **prod_knear** multiply $[Knear]$ and the candidate vector

30. **csrsymv** multiply a compressed sparse matrix with a vector

31. **post_proc** post-processing subroutine, export the results under text files: traction, displacement, crack opening displacement

32. **Q8_Q4_slab** create geometry and nodal solution (*.mesh and *.bb) of slab for MEDIT

33. **Q8_Q4_crack** create geometry and nodal solution (*.mesh and *.bb) of cracks for MEDIT

34. **Q8_Q4_inclusion** create geometry and nodal solution (*.mesh and *.bb) of inclusions for MEDIT

35. **Q8_Q4_exact** create geometry and exact nodal solution if it exists (*.mesh and *.bb) of the problem for MEDIT

36. **write_echo** create a memo file (*log.txt*) which contains all the input information of the calculation: date and time, number of unknowns in traction, in displacement, in crack opening displacement, number of zones/inclusions, material properties of each zone, number of gaussian points for each geometry, and parameter for octree and GMRES

37. **write_octree** create a memo file (*octree.txt*) which contains the *octree* structure: name of each cell, its neighbors, children, position about its branch, level; scan on each level and show name of cells in each level...

38. **write_report** create a memo file (*report.txt*) which summarizes the output information of the calculation: number of outer iteration, time per iteration, pre-processing time, solution time, total computational time ...

39. **int_buu** consider the relative position between 2 elements and warp to the appropriate B_{uu} integration scheme

40. **int_btt** consider the relative position between 2 elements and warp to the appropriate B_{tt} integration scheme

41. **int_btu** consider the relative position between 2 elements and warp to the appropriate B_{tu} integration scheme

42. **int_buu_2** consider the relative position between 2 elements and warp to the appropriate B_{tu_2} integration scheme

43. **int_regu_buu** compute the regular B_{uu} integral

44. **int_regu_btu** compute the regular B_{tu} integral

45. **int_regu_btt2** compute the regular B_{tt} integral

46. **int_regu_btu_2** compute the regular B_{tu}_2 integral

47. **int_coin_buu** compute the common vertex B_{uu} integral

48. **int_coin_btu** compute the common vertex B_{tu} integral

49. **int_coin_btt2** compute the common vertex B_{tt} integral

50. **int_coin_btu_2** compute the common vertex B_{tu}_2 integral

51. **int_conf_buu** compute the coincidence B_{uu} integral

52. **int_conf_btu** compute the coincidence B_{tu} integral

53. **int_conf_btt2** compute the coincidence B_{tt} integral
54. **int_conf_btu_2** compute the coincidence B_{tu}_2 integral
55. **int_cote_buu** compute the common edge B_{uu} integral
56. **int_cote_btu** compute the common edge B_{tu} integral
57. **int_cote_btt2** compute the common edge B_{tt} integral
58. **int_cote_btu_2** compute the common edge B_{tu}_2 integral
59. **eval_Uik** evaluate the kernel U_i^k
60. **eval_Tik** evaluate the kernel T_i^k
61. **eval_Bikqs** evaluate the kernel B_{ikqs}
62. **coor_elem** recuperate the cartesian coordinates of nodes in the given element
63. **eval_PF** evaluate the real cartesian coordinates of a given point
64. **eval_SHP** evaluate the shape function of a given gaussian point
65. **eval_DSHP** evaluate the derivative of the shape function of a given gaussian point
66. **eval_NORM** evaluate the normal vector of an element at given point, also compute the jacobian of the transformation
67. **eval_Cijhk** evaluate the forth order tensor of elasticity
68. **eval_Rnm** evaluate the harmonic function R_{nm}
69. **eval_Snm** evaluate the harmonic function S_{nm}
70. **eval_Fiknm** evaluate the term $F_{ij,n,m}$
71. **transl_M2M** execute the M2M transformation
72. **transl_M2L** execute the M2L transformation
73. **transl_L2L** execute the L2L transformation

Appendix D

FMM - Useful formulaes

This appendix 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 readers are referred to Nishimura et al. [31] for further details. The brief description of the recursive formulae can be summarized as follow:
(a) The $R_{n,m}(\mathbf{y})$ are computed recursively by setting $R_{0,0}(\mathbf{y}) = 1$ and using:

$$R_{n+1,n+1}(\mathbf{y}) = \frac{y_1 + iy_2}{2(n+1)} R_{n,n}(\mathbf{y}) \tag{D.1}$$

$$\left((n+1)^2 - m^2\right) R_{n+1,m}(\mathbf{y}) - (2n+1) y_3 R_{n,m}(\mathbf{y}) + \|\mathbf{y}\|^2 R_{n-1,m}(\mathbf{y}) = 0 \tag{D.2}$$

(b) The $S_{n,m}(\mathbf{x})$ are computed recursively by setting $S_{0,0}(\mathbf{x}) = \dfrac{1}{\|\mathbf{x}\|}$ and using:

$$S_{n+1,n+1}(\mathbf{x}) = \frac{(2n+1)(x_1 + ix_2)}{\|\mathbf{x}\|^2} S_{n,n}(\mathbf{x}) \tag{D.3}$$

$$\|\mathbf{x}\|^2 S_{n+1,m}(\mathbf{x}) - (2n+1) x_3 S_{n,m}(\mathbf{x}) + (n^2 - m^2) S_{n-1,m}(\mathbf{x}) = 0 \tag{D.4}$$

(c) Finally, the negative terms are computed ($n > m$) following the properties:

$$R_{n,-m}(\mathbf{y}) = (-1)^m \overline{R_{n,m}(\mathbf{y})} \tag{D.5}$$

$$S_{n,-m}(\mathbf{x}) = (-1)^m \overline{S_{n,m}(\mathbf{x})} \tag{D.6}$$

The first order derivatives of $R_{n,m}$ and $S_{n,m}$ are computed via:

$$\frac{\partial}{\partial y_1} R_{n,m} = \frac{1}{2}(R_{n-1,m-1} - R_{n-1,m+1}) \tag{D.7}$$

$$\frac{\partial}{\partial y_2} R_{n,m} = \frac{i}{2}(R_{n-1,m-1} - R_{n-1,m+1}) \tag{D.8}$$

$$\frac{\partial}{\partial y_3} R_{n,m} = R_{n-1,m} \tag{D.9}$$

$$\frac{\partial}{\partial x_1} S_{n,m} = \frac{1}{2}(S_{n+1,m-1} - R_{n+1,m+1}) \tag{D.10}$$

$$\frac{\partial}{\partial x_2} S_{n,m} = \frac{i}{2}(S_{n+1,m+1} - S_{n+1,m-1}) \tag{D.11}$$

$$\frac{\partial}{\partial x_3} S_{n,m} = -S_{n+1,m} \tag{D.12}$$

And the second order derivatives:

$$\frac{\partial}{\partial y_1}\frac{\partial}{\partial y_1}R_{n,m} = \frac{1}{4}(R_{n-2,m-2} - 2R_{n-2,m} + R_{n-2,m+2}) \tag{D.13}$$

$$\frac{\partial}{\partial y_1}\frac{\partial}{\partial y_2}R_{n,m} = \frac{\partial}{\partial y_2}\frac{\partial}{\partial y_1}R_{n,m} = \frac{i}{4}(R_{n-2,m-2} - R_{n-2,m+2}) \tag{D.14}$$

$$\frac{\partial}{\partial y_1}\frac{\partial}{\partial y_3}R_{n,m} = \frac{\partial}{\partial y_3}\frac{\partial}{\partial y_1}R_{n,m} = \frac{1}{2}(R_{n-2,m-1} - R_{n-2,m+1}) \tag{D.15}$$

$$\frac{\partial}{\partial y_2}\frac{\partial}{\partial y_2}R_{n,m} = -\frac{1}{4}(R_{n-2,m-2} + 2R_{n-2,m} + R_{n-2,m+2}) \tag{D.16}$$

$$\frac{\partial}{\partial y_2}\frac{\partial}{\partial y_3}R_{n,m} = \frac{\partial}{\partial y_3}\frac{\partial}{\partial y_2}R_{n,m} = \frac{i}{2}(R_{n-2,m-1} + R_{n-2,m+1}) \tag{D.17}$$

$$\frac{\partial}{\partial y_3}\frac{\partial}{\partial y_3}R_{n,m} = R_{n-2,m} \tag{D.18}$$

$$\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_1}S_{n,m} = \frac{1}{4}(S_{n+2,m-2} - 2S_{n+2,m} + S_{n+2,m+2}) \tag{D.19}$$

$$\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}S_{n,m} = \frac{\partial}{\partial x_2}\frac{\partial}{\partial x_1}S_{n,m} = \frac{i}{4}(S_{n+2,m-2} - S_{n+2,m+2}) \tag{D.20}$$

$$\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_3}S_{n,m} = \frac{\partial}{\partial x_3}\frac{\partial}{\partial x_1}S_{n,m} = -\frac{1}{2}(S_{n+2,m-1} - S_{n+2,m+1}) \tag{D.21}$$

$$\frac{\partial}{\partial x_2}\frac{\partial}{\partial x_2}S_{n,m} = -\frac{1}{4}(S_{n+2,m-2} + 2S_{n+2,m} + S_{n+2,m+2}) \tag{D.22}$$

$$\frac{\partial}{\partial x_2}\frac{\partial}{\partial x_3}S_{n,m} = \frac{\partial}{\partial x_3}\frac{\partial}{\partial x_2}S_{n,m} = -\frac{i}{2}(S_{n+2,m-1} + S_{n+2,m+1}) \tag{D.23}$$

$$\frac{\partial}{\partial x_3}\frac{\partial}{\partial x_3}S_{n,m} = S_{n+2,m} \tag{D.24}$$

Bibliography

[1] D. Pham, Fast multipole method applied to Symmetric Galerkin boundary element method for 3D elasticity and fracture problems. Eng. Anal. Bound. Elem. 2012 (Cited on pages 2, 22, 25 and 43.)

[2] S. Chaillat, Methode multipole rapide pour les equations integrales de frontieres en elastodynamique 3D. Application a la propagation d'ondes sismiques. Phd thesis 2008. (Cited on pages 6, 43, 44, 47 and 51.)

[3] S. Mouhoubi, Couplage Symetrique Elements Finis-Elements de Frontiere en Mecanique: Formulation et Implantation dans un code element finis. Phd thesis 2000. (Cited on pages 2 and 22.)

[4] Yoshida. K.I, Application of fast multipole to boundary integral equation method. Phd thesis 2001. (Cited on pages 2, 6, 26 and 87.)

[5] M. Margonari, Boundary element techniques for three dimensional problems in Elastostatics. Phd thesis 2004. (Cited on pages 2, 22, 43, 60, 63, 70 and 71.)

[6] M. Bonnet, Boundary integral equation method for Solid and Fluids, Wiley 1999. (Cited on pages xiv, 6, 10, 17 and 20.)

[7] M. Bonnet, Exploiting partial or complete geometrical symmetry in 3D symmetric Galerkin indirect BEM formulations, Int. J. Num. Meth. Engng., Vol. 57, issue 8, p.1053-1083, 2003. (Cited on page 101.)

[8] Y.Saad, A flexible inner-outer preconditioned GMRES algorithm. (Cited on page 44.)

[9] Y. Saad, H. M. Schulz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Compt., 7:856-869, 1986. (Cited on pages 21 and 44.)

[10] L. J. Gray, Glaucio H. Paulino, Symmetric Galerkin Boundary Integral Formulation for Interface and Multi-zone problems. Int. J. Numer. Meth. Engng., vol. 40:pp. 3085-3101, 1997. (Cited on pages 6, 60 and 63.)

[11] A. Frangi, G. Novati, R.Springhetti, M. Rovizzi, 3D fracture analysis by symmetric Galerkin BEM. Compt. Mech. 28 (2002) p.220-232. (Cited on pages 1, 2 and 87.)

[12] A. Frangi, Fracture propagation in 3D by the symmetric Galerkin boundary element method. Int. J. Fract. vol.116, p.313-330, 2002. (Cited on page 87.)

[13] S. Sitory, General stress analysis method by means of integral equations and boundary elements. Meccanica 14:210-218, 1997. (Cited on page 1.)

[14] Y. Mi, Three-dimensional Analysis of Crack Growth, Compt. Mech. Publication, Southampton, 1999. (Cited on page 87.)

[15] S. Tada, P. Paris, G. Irwin, The Stress Analysis of Cracks handbook. Dell Research Corporation, St. Louis, 1985. (Cited on page 78.)

[16] X. Li, L. M. Keer, A direct method for solving crack growth problems. Int. J. Solids. Struct. vol.29, p.2735-2747, 1992. (Cited on page 87.)

[17] Y. Mi, M. H. Aliabadi, Three-dimensional crack growth simulation using BEM. Comp. Struct. vol.52, p.871-878, 1994. (Cited on page 87.)

[18] D.Z.Ding,R.S.Chen,Z.H.Fan, SSOR preconditoned inner-outer flexible GMRES method for MLFMM analysis of scattering of open objects. Progress in Electromagnetics Research, PIER 89,339-357, 2009. (Cited on page 44.)

[19] F. Hartmann, Introduction to boundary elements - theory and applications. Berlin: Springer Verlag, 1989. (Cited on page 5.)

[20] J.M.Song, C.C. Lu, W.C.Chew, Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. IEEE transactions on Antennas and Propagation. Vol. 45, No.10, 1488-1493, Oct.1997. (Cited on page 43.)

[21] K.Sertel, J.L. Volakis, Incomplete LU preconditioner for FMM implementation. Microwave and Optical Technology Letters, Vol.26, No.7, 265-267, 2000. (Cited on page 43.)

[22] E.Chow, Y.Saad, Experimental study of ILU precontioners for indefinite matrices. Journal of Computational and Applied Mathematics, Vol.86, 387-414, 1997. (Cited on page 43.)

[23] T. Chen, A symmetric Galerkin multi-zone boundary element method for cohesive crack growth. Engineering Fracture Mechanics, Vol.63, p.591-609, 1999. (Cited on page 60.)

[24] S. Ganguly, J.G. Layton, A fully symmetric multi-zone Galerkin boundary element method. Int. J. Meth. Num. Engng., Vol.44, p.991-1009, 1999. (Cited on pages 60 and 70.)

[25] R.C.Williams, A.V.Phan, SGBEM analysis of crack-particle(s) interactions due to elastic constants mismatch. Eng.Frac.Mech., Vol.74, p.314-331, 2007. (Cited on page 6.)

[26] D. J. Roberts, A. V. Phan, H. V. Tippur, L. J. Gray, T. Kaplan, SGBEM modeling of fatigue crack in particulate composites, Arch. Appl. Mech., vol. 80, p307-322, 2010. (Cited on page 87.)

[27] R. Kitey, A. V. Phan, H. V. Tippur, T. Kaplan, Modeling of crack growth through particulate clusters in brittle matrix by symmtric Galerkin boundary element method, Int. J. Frac, vol. 141, p11-25, 2006. (Cited on page 87.)

[28] K. Xu, S. T. Lie, Z. Cen, Crack propagation analysis with Galerkin boundary element method, Int. J. Num. Anal. Geomechanics, vol.28, p421-435, 2004. (Cited on page 87.)

[29] L. Tavara, V. Mantic, A. Salvadori, L. J. Gray, F. Paris, SGBEM for cohesive cracks in homogeneous media, Key. Eng. Mat, vol. 454, p1-10, 2011. (Cited on page 88.)

[30] Compressed Sparse Row (CSR) *https://en.wikipedia.org/wiki/Sparse_matrix* (Cited on pages xvi and 40.)

[31] N. Nishimura, Fast Multipole accelerated boundary integral equation methods. Appl. Mech. Rev., vol. 55, pp. 299-324, 2002 (Cited on page 131.)

[32] L. Zhao, Z. Yoa, Fast Multipole BEM for 3D elastostatic problems with application for thin structures. Tshinghua science and technology, vol.10,67-75,2005. (Cited on page 2.)

[33] V. Rokhlin, Rapid solution of integral equations of classical potential theory. J. Comp. Phys., vol.60.pp. 187-207, 1985. (Cited on pages 2, 6 and 22.)

[34] M. S. Bapat, L. Shen, Y. J. Liu, Adaptive fast multipole boundary element method for three-dimensional half-space acoustic wave problems. Eng. Anal. Bound. Elem., 33:1113-1123, 2009. (Cited on page 22.)

[35] E. Becache, J. Nedelec, N. Nishimura, Regularization in 3D for anisotropic elastodynamic crack and obstacle problems. J. Elast., 31:25-46, 1993. (Cited on page 12.)

[36] Bonnet M., Regularized direct and indirect symmetric variational BIE formulation for three-dimensional elasticity. Eng. Anal. Bound. Elem., 15:93-102. 1995. (Cited on pages 11 and 12.)

[37] Bonnet M., Maier G., Polizzoto C., Symmetric Galerkin Boundary Element Method. Appl. Mech. Rev., 51:669-704. 1998. (Cited on page 11.)

[38] S. Chaillat, M. Bonnet, J. F. Semblat, A multi-level fast multipold BEM for 3D elastodynamics in the frequency domain Comp. Meth. Appl. Mech. Eng., 197:4233-4249, 2008. (Cited on page 22.)

[39] Y. H. Chen, W. C. Chew, S. Zeroug, Fast multipole method as an efficient solver for 2D elastic wave surface integral equations. Comp. Mech., 20:495-506, 1997. (Cited on page 22.)

[40] W. C. Chew et al., Fast integral equations solvers in computational electromagnetics of complex structures. Eng. Anal. Bound. Elem. 27:803-823, 2003. (Cited on page 22.)

[41] A. Frangi, A. Di Gioia, Multipole BEM for the evaluation of damping forces on MEMS. Comp. Mech. 37:24-31, 2005. (Cited on page 22.)

[42] J. E. Gomez, H. Power, A multipole direct and indirect BEM for 3D cavity flow at low Reynolds number. Eng. Anal. Bound. Elem. 19:17-31, 1997. (Cited on page 22.)

[43] Sitori S., Maoer G., Novati G., Miccoli S., A Galerkin symmetric boundary element method in elasticity: formulation and implementation. Int. J. Num. Meth. Eng., 35:255-282. 1992. (Cited on pages 10 and 11.)

[44] T. Takahashi, N. Nishimura, S. Kobayashi, A fast BIEM for three-dimensional elastodynamics in time domain. Eng. Anal. Bound. Elem. 28:165-180, 2004. (Cited on page 22.)

[45] Y. Yauda, T. Sakuma, A technique for plane-symmetry sound field analysis in the fast multipole boundary element method. J. Comp. Acoust. 13:71-85, 2005. (Cited on page 22.)

[46] Y. Fu and G. J. Rodin, Fast solution method for three-dimensional Stokesian many-particle problems. Comm. Numer. Meth. Engng., 16:145-149, 2000. (Cited on page 2.)

[47] G. Sylvand, La methode multipole rapide en electromagnetisme. Performance, parallelisationm application. Phd thesis, ENPC, 2002. (Cited on page 2.)

[48] O. Tamate, The effect of a circular inclusion on the stresses around a line crack in a sheet under tension. Int. J. Fract. 1968;4:257-65. (Cited on page 82.)

[49] GP. Sendeckyj, Interaction of cracks with rigid inclusions in longitudinal shear deformation. Int. J. Fract. 1974;10:45-52. (Not cited.)

[50] C. Hwu, YK. Liang, WJ. Yen, Interactions between inclusions and various types of cracks. Int. J . Fract. 1995;73:301-23. (Cited on page 82.)

[51] R. Li, A. Chudnovsky, A variation of the energy release rate as a crack approaches and passes through an elastic inclusion. Int. J. Fract. 1993; 59:R69-74. (Cited on page 82.)

[52] R. Li, A. Chudnovsky, Energy analysis of crack interaction with an elastic inclusion. Int.J.Fract. 1993;63:274-61. (Not cited.)

[53] P. Lipetzky, S. Schmauder, Crack-particle interaction in two-phase composites. Part I: Particle shape effects. Int.J.Fract. 1994;65:345-58. (Not cited.)

[54] P. Lipetzky, Z. Knesl, Crack-particle interaction in two-phase composites. Part II: Crack deflection. Int.J.Fract. 1995;73:81-92. (Not cited.)

[55] A. Haddi, D. Weichert, Three-dimensional interaction between a crack front and particles. Int.J.Numer.Meth.Eng. 1998;42:1463-76. (Not cited.)

[56] MB. Bush, THe interaction between a crack and a particle cluster. Int.J.Fract. 1997;88:215-32. (Not cited.)

[57] MG. Knight, LC. Wrobel, JL, Henshall, LA. De Lacerda, A study of the interaction between a propagating crack and an uncoated/coated elastic inclusion using the BE technique. Int.J.Fract. 2002;114:47-61. (Not cited.)

[58] C. Wang, W. Libardi, JB. Baldo, Analysis of crack e xtension paths and toughening in a two phase brittle particulate composite by the boundary element method. Int.J.Fract. 1998,94:177-88. (Cited on page 82.)

[59] Peirce AP, Napier JAL. A spectral multipole method for efficient solution of large-scale boundary element models in elastostatics. Int. J. Number. Methods. Eng. 1995; 38:4009-34. (Cited on page 22.)

[60] Gomez JE, Power H. A multipole direct and indirect BEM for 2D cavity flow at low Reynolds number. Eng. Anal. Bound. Elem. 1997;19:17-31. (Cited on page 22.)

[61] Fu Y, Klimkowski KJ, Rodin GJ, Berger E, Browne JC, Singer JK, et al. A fast solution method for three-dimensional many-particle problems of linear elasticity. Int. J. Numer. Methods Eng. 1998;42:1215-29. (Cited on page 22.)

[62] Nishimura N, Yoshida K, Kobayashi S. A fast multipole boundary integral equation method for crack problems in 3D. Eng. Anal. Bound. Elem. 1999;23:97-105. (Cited on page 22.)

[63] Mammoli AA, Ingber MS. Stokes flow around cylinders in a bounded two-dimensional domain using multipole-accelerated boundary element methods. Int. J. Numer. Meth. Eng. 1999;44:897-917. (Cited on page 22.)

[64] Nishimura N, Liu YJ. Thermal analysis of carbon-nanotube composites using a rigid-line inclusion model by the boundary integral equation method. Comput. Mech. 2004;35(1):1-10. (Cited on page 22.)

[65] Liu YJ, Nishimura N, Otani Y, Takahashi T, Chen XL, Munakata H. A fast boundary element method for the analysis of fiber-reinforced composites based on a rigid-inclusion model. J. Appl. Mech. 2005;72(1):115-28. (Cited on page 22.)

[66] Chew WC, Chao HY, Cui TJ, Lu CC, Ohnuki S, Pan YC, et al. Fast integral equation solvers in computational electromagnetics of complex structures. Eng. Anal. Bound. Elem. 2003;27(8):803-23. (Cited on page 22.)

[67] Leung C.Y., Wtalker S.P. Iterative solution of large three-dimensional BEM elastostatic analyses using the GMRES technique. Int. J. Numer. Meth. Engng., vol. 40:pp.22727-2236, 1997. (Cited on page 21.)

[68] H. Andra, Integration of singular integrals for the Galerkin-type boundary element method in 3D elasticity. Comp. Mech. Appl. Mech. Engng., 157:239-249, 1998. (Cited on page 1.)

[69] J. Balas, J. Sladek and V. Sladek, Stress Analysis by the Boundary Element Method. Elsevier, Amsterdam, 1989. (Cited on page 1.)

[70] P. K. Banerjee, The Boundary Element Methods in Engineering. McGraw Hill, London 1994. (Cited on page 1.)

[71] G. Beer, Programming the Boundary Element Method: In Introduction for Engineers. John Wileys & Sons, New Jersey, 2001. (Cited on page 1.)

[72] C. A. Brebbia and J. Dominguez, Boundary Elements: An Introductory Course. WIT Press, Boston, Southampton, 1992. (Cited on page 1.)

[73] D. Capuani, D. Bigoni and M. Brun, Integral representations at the boundary for stokes flow and related symmetric Galerkin formulation. Archives of Mechanics, 57(5):363-385, 2005. (Cited on page 2.)

[74] L. J. Gray and G. H. Paulino, Symmetric Galerkin boundary integral fracture analysis for plane orthotropic elasticity. Computational Mechanics, 20(1/2):26-33, 1997. (Cited on page 2.)

[75] J. H. Kane, Boundary Elements Analysis in Engineering Continuum Mechanics. Prentice Hall, New Jersey. 1994. (Cited on page 73.)

[76] G. Krishnasamy, L. W. Schmerr, T. J. Rudolphi and F. J. Rizzio. Hypersingular boundary integral equations: some applications in acoustic and elastic wave scattering. Journal of Applied Mechanics, 57:404-414, 1990. (Cited on page 1.)

[77] L. Lehmann and H. Antes, Dynamic structure-soil-structure interaction applying the symmetric Galerkin boundary element method (SGBEM). Mechanics Research Communications, 28(3):297-304, 2001. (Cited on page 2.)

[78] G. Maier, M. Diligenti and A. Carini, A variational approach to boundary element elastodynamic analysis and extension to multidomain problems. Comp. Meth. Appl. Engng., 92:193-213, 1991. (Cited on page 2.)

[79] D. Poljak, Foreword to EABE special issue on electromagnetics. Engineering Analysis with Boundary Elements, 27(4), 2003. (Cited on page 1.)

[80] B. Reidinger and O. Steinbach. A symmetric boundary element method for the Stokes problem in multiple connected domains. Mathematical Methods in the Applied Sciences, 26(1):77-93, 2003. (Cited on page 2.)

[81] M. Tanaka and T.A. Cruse. Boundary element methods in applied mechanics. Pergamon Press, Oxford, 1988. (Cited on page 1.)

[82] N. Sukumar, D. L. Chopp, E. Bechet, N. Moes, Three-dimensional non-planar crack growth by a coupled extended finite element and fast marching method. Int. J. Numer. Meth. Engng. p.1-39, 2007. (Cited on page 87.)

[83] C. Chazallon, G. Koval, P. Hornych, F. Allou, S. Mouhoubi, Modeling of rutting of two flexible pavements with the shakedown theory and the finite elements method. Comp. and Geotechnics, vol. 36 (2009), 798-809. (Cited on page 100.)

[84] P. Hornych, Development and validation of a method of prediction of rutting of unbound pavement layers. Sustainable and Advanced Materials for Road InfraStructure (SAMARIS) project (Report WP5). Retrieved from *http://samaris.zag.si/* (Cited on page 100.)

[85] M. Rezayal, On time-harmonic elastic-wave analysis by the boundary element method for moderate to high frequencies. Comp. Meth. Appl. Mech. Engng., vol.55, p.349-367, 1986. (Cited on page xvi.)