A hierarchical linear elastic boundary element solver for lenticular ore bodies

by

Christiaan Abraham Zietsman

Thesis presented in partial fulfilment of the requirements for the degree of Master of Natural Sciences at the University of Stellenbosch

Applied Mathematics Division, Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, 7602 Matieland, South Africa.

Supervisor: Prof J.P. du Plessis

December 2007

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: .....
C.A. Zietsman

Date: .....

Abstract

South Africa is involved in huge mining operations deep in the earth's crust. Stresses induced by these mining operations may cause seismic events or rockbursts to occur, which could damage infrastructure and put miners' lives at risk. The effect of different mining layouts are modelled and used by engineers to make design decisions. The frequency at which models are updated and integrated with the decision making process is not optimal. These large mining layouts can not be modelled adequately using domain methods, but they are particularly well suited for the boundary element method (BEM).

This work focuses on the theory and background needed for creating a linear elastic static stress boundary element solver suited to South African mining layouts. It starts with linear elastic theory and subsequently describes the physical continuum, governing equations and the fundamental solutions which are an integral part of the BEM. Kelvin's solution cannot be applied to crack-like excavations, therefore the displacement discontinuity kernels, which are very well suited to model fractures, are derived. The derivation is approached from both the direct and indirect BEM's perspectives. The problem is cast as a boundary integral equation which can be solved using the BEM. Some of the different specializations of the BEM are discussed. The major drawback of the BEM is that it produces a dense influence matrix which quickly becomes intractable on desktop computers. Generally a mining layout requires a large amount of boundary elements, even for coarse discretization, therefore different techniques of representing the influence matrix are discussed, which, combined with an iterative solver like GMRES or Bi-CG, allows solving linear elastic static stress models.

Ekserp

Suid-Afrika het baie groot myne wat diep ondergronds is. Kragte wat deur hierdie myn-aktiwiteite veroorsaak word kan seismisiteit en rotsstortings veroorsaak en dit kan infrastruktuur en mynwerkers se lewens in gevaar stel. Die invloed van verskillende mynuitlegte kan deur ingenieurs gebruik word tydens beplanning. Die frekwensie waarteen modelle opgedateer en met besluitneming geïntegreer word is nie optimaal nie. Die baie groot mynuitlegte word tans nie goed genoeg met gebiedsmetodes hanteer nie, maar is wel baie gepas vir die rand-element metode (BEM).

Hierdie werk fokus op die teorie en agtergrond wat nodig is om 'n staties linieêr elastiese rand-element oplosser vir Suid-Afrika se mynuitlegte te maak. Dit begin met lineêre elastisiteit, 'n beskrywing van die fisiese kontinuum, die bepalende vergelykings en die fundementele oplossings wat 'n integrale deel van die BEM vorm. Kelvin se oplossing is nie toepaslik om frakture mee te modelleer nie, dus word kerne vir verplasings diskontinuiteite afgelei wat wel goed gepas is. Die afleiding word van beide die direkte en indirekte BEM se perspektiewe benader. Die probleem word dan omskryf as 'n rand-integraal vergelyking wat met behulp van die BEM opgelos kan word. Van die verskillende BEMs word dan bespreek. Die grootste nadeel van die BEM is dat dit 'n digte invloed matriks genereer wat onhanteerbaar word op persoonlike rekenaars. In die algemeen be nodig mynuitlegte baie elemente, selfs vir 'n growe voorstelling. Verskillende maniere om die invloed matriks meer optimaal voor te stel word bespreek. Hierdie voorstellings kan dan saam met iteratiewe matriks oplossers soos GMRES en Bi-CG gebruik word om die staties lineêr elastiese modelle op te los.

Acknowledgements

Foremost, I would like to thank Integrated Seismic Systems International (ISSI), specifically Alex Mendecki, who envisions ISSI as a leader in integrating seismicity with numerical modelling and providing services to the mining community. His encouragement of employees to undertake post-graduate studies is acknowledged. If it wasn't for his persuasiveness I probably would have done a fulltime Masters in Mathematics rather than a part-time Masters on Continuum Mechanics. At ISSI, I would also like to thank my collogues: Renoir Sewjee, for hour long discussions and brainstorming sessions, Assen Ilchev who was always eager to share some of his vast mathematical and physics insights and Francois Malan for listening when Renoir wasn't around.

Thank you to Prof du Plessis for his eagerness to help and provide support on a topic which is somewhat vaguely related to his field of interest, for his willingness to listen, for providing me with a quiet office for the final push and for making many revisions to the manuscript.

Thanks to my friends for who continually kept me aware of the unfinished state of this work and Michelle for understanding and giving me enough time to finish. I would especially like to thank my parents for years of support, patience and encouragement.

Contents

List of Abbreviations

ACA Adaptive Cross Approximation

BEM Boundary Element Method

Bi-CG Bi-Conjugate Gradient

BIE Boundary Integral Equation

BVP Boundary Value Problem

CPV Cauchy Principal Value

DDM Displacement Discontinuity Method

FDM Finite Difference Method

FDM Force Discontinuity Method

FEM Finite Element Method

FFM Fictitious Force Method

FFT Fast Fourier Transform

FMM Fast Multipole Method

GMRES Generalised Minimal Residual

MGS Modified Gram-Schmidt

MPM Material Point Method

PBF Primitive Boundary Function

PIC Particle-in-cell

SPH Smoothed Particle Hydrodynamics

SVD Singular Value Decomposition

List of Figures

Chapter 1

Introduction

In research regarding mining operations there are many unknowns and numerous assumptions to be made. Considerable effort is made globally to gain more understanding of the subsurface environment. While the main aim of this understanding should be the improvement of safety, this is not always the case; rather improving profitability is usually considered first, due to legislation and severe monetary penalties for casualties and injuries these two aims frequently coincide. Even if more could be done to improve our understanding, using the information already obtained and preparing it to be used in modelling can be difficult.

Over the years the search for gold has taken us deeper and deeper, and with depth the dangers increase. Currently there are plans to go even further down and this will continue with the demand for gold. The same is happening for platinum mines, where mining is still being done at shallow depths relative to the depths at which gold extraction is taking place.

A lot of surveying is done to get a good picture of the geological structures, rock types and in situ stress due to prehistoric loading. Rock samples are taken and tested in laboratories to determine their properties which can then be used in numerical modelling. Certain mining practices like backfilling, support systems and layout design also need to be included in the modelling. It has become common to monitor seismic activity which could be incorporated to bring the modelling closer to reality.

1.1 Motivation

The initial aim of our project was to develop a tool to solve for the static stress state given the geometric layout of the mine using the boundary element method (BEM). The boundary element method is also known as the boundary integral equation method (BIE) or boundary integral method. The development of this tool contained many pitfalls and was a big learning experience; because the different parts were developed together it did

not make the job any easier. These parts include

- setting up the model,
- discretizing the model boundary,
- integration of kernels,
- constructing a linear system of equations,
- solving the linear system and
- computing results at field points.

The first two parts were separated into a preprocessing step to generate input for the numerical engine. The numerical engine consists of the remaining parts. The major driving factors for the engine were speed, storage requirements, accuracy, reliability and flexibility. This work does not try to explain exactly how such a project can be implemented, but rather tries to provide the background needed and give a general guideline on what considerations needs to be made when tackling such a difficult project.

1.2 Kernel Integration

The most crucial part of implementing the boundary element method is in computing the influence coefficients. The difficulties associated are probably why boundary element methods have taken a back seat to more popular and conceptually easier domain methods, like finite elements and finite differences. When tackling very large scale problems one quickly runs into the spatial limitations of domain methods. In these situations BEM is perfect for supplementing the domain method by providing accurate boundary conditions.

The kernels are derived from the fundamental solution for the linear static stress problem in an infinite homogenous domain also known as the Kelvin solution. These kernels are obtained by differentiating the Kelvin solution with respect to space. Integrating the resulting kernels varies in difficulty since they range from weakly singular through singular and up to hypersingular. The weakly singular integrals are amenable to simple numerical quadrature, the singular integrals are a little bit more difficult and exist as Cauchy principal value integrals, and lastly the hypersingular integrals can be interpreted in the Hadamard finite part sense.

A unified method was sought to compute the kernel integration. The continuation approach was found to be the most appealing method since it seems to be numerically robust and simultaneously allows for points which are on, near or far from the element integrated over. A wide range of integration schemes were tried out before settling on the

continuation approach. Some of the techniques tried were: analytical integration, subdivision based integration and Gauss quadrature. The continuation approach described in a later chapter was chosen for its accuracy, reliability and flexibility. Even though the continuation approach gave very accurate results, it was still too slow to use in the far-field where kernels are amenable to ordinary low order numerical quadrature. To remedy the speed problem, an adaptive Gauss quadrature scheme was introduced. This was done in such away as not to compromise the accuracy, reliability or flexibility of the continuation approach.

1.3 Construct and solve

Constructing the linear system and solving it can be a great and difficult art. Building the system requires the evaluation of $O(n^2)$ integrals where n is the number of elements used for describing the boundary. The system can be very large for simple practical problems and one quickly runs into storage and memory limitations. The choices of matrix solvers are relatively few, even though many variants exist. The easiest and most reliable are direct solvers like Gauss elimination which requires $O(n^3)$ operations and is not feasible for relatively small problems.

If one travels along the less reliable route, one arrives at the family of iterative solvers. The major advantage of iterative solvers is the huge speed improvement obtainable, even though the cost for this improvement comes at the price of reliability, since iterative solvers do not guarantee convergence for unsymmetric dense systems obtained from BEM. Iterative solvers are based on successively performing matrix-vector products and in general do not require the construction of the matrix.

Successive approximations of the solution vector are generated until a specified absolute or relative accuracy is reached. The core of any fast BEM solver exploits the fact that the influence matrix can be substituted with an approximation and then used to perform the matrix-vector product. The aim is to create such an approximant which does not need the usual $O(n^2)$ operations to create and more importantly, the $O(n^2)$ growth in storage space, but instead performs the matrix-vector product up to a prescribed accuracy requiring much less resources.

1.4 Compression

Huge primary and secondary storage for desktop computers have become common, but still it is not enough for even a small to average sized problem. The additional time required to swap from disk into main memory at the scales required by direct unoptimized BEM implementations would make using an iterative solver infeasible. Over the years

many ideas have been researched and applied to elasticity to overcome this obstacle. Examples are the Fast multipole method (FFM) [3], precorrected FFT methods [4], wavelets methods [5], panel clustering method [6] and algebraic methods based on the singular value decomposition (SVD) [7; 8; 9].

An in-house variant based primarily on IES^3, because of its simplicity, was implemented. Unfortunately getting it to work for linear elasticity was difficult and only raised more questions. The IES^3 method is an example of an algebraic method, since it only concerns itself with algebraically manipulating the influence matrix directly. Other methods are more complicated because they usually require details of the problem's kernel function. The advantage of IES^3 was that it constructed an approximation of the influence matrix with which a fast matrix-vector product could be performed. In this text a matrix-vector product will be considered fast if it can be performed in sub-quadratic time, but preferably in $O(n \log^{\delta}(n))$ operations.

A method to construct the approximation in sub-quadratic time was very briefly discussed in the initial [7, IES^3] paper, but many of the details were left out. Therefore the major bottleneck was $O(n^2)$ kernel evaluations to construct the approximation, which was not yet fast enough and could be improved.

The [2, Dual-MGS] method provided the breakthrough needed to get sub-quadratic construction time. It described a manner of constructing sub-matrices by deterministically sampling rows and columns using smoothness assumptions to select candidates and also to provide a stopping criterion. In the paper they applied Dual-MGS to sub-matrices for capacitance extraction. Dual-MGS was then applied to approximate sub-matrices for linear elasticity. Dual-MGS iteratively expands the approximation space and will give an accurate approximation if all the assumptions are applicable.

Later a kernel-independent variant of the FMM [10] proved to be even more general and successful at producing a compressed approximant of the influence matrix, but a discussion will not be included.

1.5 Preconditioning

Iterative solvers tend to be more robust for smaller problems and generally become unstable for larger ones. It has been shown that the number of iterations needed with an appropriately preconditioned iterative solver for BEM can be bounded by a logarithmic factor of the problem size [11]. Slow convergence rates do not really affect the total solution time too severely for small problem sizes. All the improvements made, to more efficiently construct the approximation, allows for much larger problem sizes. Therefore, slow convergence rates will severely affect the total solution time and for some problems the solver does not even converge.

The compressed structure from IES^3 is very similar to the structure explained in both [8, Sequentially semi-separable matrices] and [9, H-Matrices] and for both these representations, ideas on constructing preconditioners were given. In truth, both papers include algorithms to construct pseudo-inverses that should be applicable to the in-house variant. A very coarse preconditioner was implemented by borrowing ideas from these papers and has proved to do the job well enough. Using a preconditioner improved reliability and reduced the number of iterations needed for convergence by almost and order of magnitude. This was expected, because the same results have been reported in many other papers on the topic.

1.6 Numerical Methods

There exist many different methods which can be used to solve problems in the engineering sciences. These methods can be split into two categories, namely domain and boundary methods. Domain methods are generally flexible, powerful and simple to understand. They have many advantages, like the possibility to model non-linearity, very complex geometries and inhomogeneous materials. Their difficulty lies in setting up these properties, which can be very time consuming and difficult. The finite element (FEM), finite difference (FDM), spectral and particle methods are examples of domain methods, [4]. Their major disadvantages lie in mesh generation and description of the problem domain, which have spawned active research into meshless methods, such as smooth particle hydrodynamics (SPH), particle-in-cell (PIC) and the material point method (MPM).

Boundary methods are generally more complex, but very powerful. Active research is underway to make it more flexible and applicable to a wider range of problems. Boundary methods have been used to solve problems in electromagnetism, acoustics, viscous flow, elasticity and other fields. The complexity in generating a mesh for the boundary of the problem domain is much less than for the domain itself. This reduction in dimensionality is probably the main reason for research into boundary element methods. Describing the problem domain for boundary element methods require much less data because of this. In general the numerical accuracy that can be achieved is better than with domain methods, [12].

1.7 Modelling tools

Modelling tools for routine static stress analysis have many requirements to fulfil. They have to be easy to use, have a quick turn around time, must be feature rich and have to provide accurate results. The impression the average user will have of such a tool will be formed by interaction with the graphical user interface. In general though, not much

thought would be given to the underlying numerical engine. What they would expect are that the results are accurate and that they would not have to wait excessively long for them. The easier the user interface is, the larger the problem specification will inevitably become.

The only way to compete with other tools on the market is to implement algorithms that perform at optimal or near optimal complexity which can treat larger problem domains. The final choice on selecting a tool for routine analysis would probably be based on features which simplifies describing the problem domain and minimizes user input, rather than on how powerful the competing product's numerical engine is. An example of such a feature could be automatic adaptive discretization with error control. The experienced user might have a feel for where the model need to be refined, but usually the job of building the model is delegated to someone with less experience. The intelligence of ensuring that the discretization and solution accuracy is achieved should rather be taken away from the user and incorporated into the engine using strict error bounds and maybe some kind of re-analysis.

1.8 Layout

Chapter 1: Introduction
Introduces the reader to the problem domain and explains the motivation behind this work. An overview of the difficulties adresses within this work, namely:

- kernel integration,
- assembling the linear system,
- solving the linear system and
- compressing the linear system

is given. Mention is made of preconditioning although this is not adressed.

Chapter 2: Elasticity
Linear elasticity theory and the governing equations are discussed, covering the necessary background needed to model a linear elastic continuum.

Chapter 3: Fundamental Solutions
Introduces the fundamental solutions which form the basis of boundary integral equation methods. The chapter mainly focuses on Kelvin's solution which gives displacement produced by a point load in an infinite elastic continuum. Equations for strain, stress and traction are derived using elasticity theory. Together with Kelvin's solution, these equations form the kernels for the fictitious force method (FFM).

Chapter 4: Cracks
The displacement discontinuity kernels, with which crack-like structures can be modelled, are derived from Kelvin's solution. This is done for both the direct and indirect method, but the explicit form of the kernels are derived from the indirect method.

Chapter 5: Boundary Element Method
Discusses the boundary integral equations and approach form solving them numerically.

Chapter 6: Kernel Integration
Discusses the difficulties associated with integrating kernels over boundary elements and methods how these integrals can be performed.

Chapter 7: Assembly
Discusses how a system of linear equations can be assembled into a matrix and reviews techniques which can be used to reduce the $O(n^2)$ storage and computational requirements inherent with the BEM when tackling large scale problems.

Chapter 8: Summary and Conclusion
Gives an overview of what was achieved within this work.

Chapter 2

Elasticity

2.1 Introduction

Solids like rubber, rock, steel, etc., exhibit linear elastic behaviour while undergoing small deformations. The range of deformation for which a linear elastic description of the solid's behaviour holds, might be very small for a brittle piece of wood or very large for an elastic band. The deformation is the result of some force applied to the object and if this force is taken away the object should return to its undeformed shape. If the object fully returns to its undeformed shape, we say that the deformation was perfectly elastic; if not then some kind of plastic deformation occurred. If the force applied causes the body to deform past some critical point, then the material might either fail or become plastic.

There are many different causes for and ways in which a body may undergo deformations. Some examples are:

- applying pressure on the surface,
- inertial forces,
- inducing magnetic flux and
- temperature changes.

In continuum mechanics, of which linear elasticity is a branch, it is assumed that the effect of molecules, atoms and other particles are continuously spread over the body and that there exists no voids. This idealization is obviously not true, but when regarding processes from a macroscopic perspective it does give good results and simplifies the description of the medium. The scales at which we apply the theory is much larger than the atomic scale, therefore we can have confidence that in an average sense the response of the material should correspond well with this theoretical model, [13].

We will only concern ourselves with the linear elastic behaviour of the medium and aim at a description of the medium in its equilibrium state. This implies that the stress-strain

relationships are linear and time independent. We also assume that induced deformations will be small enough that we can ignore their effect on the boundary of the domain and that the equilibrium equations may refer to the undeformed boundary, [12].

2.2 Strain

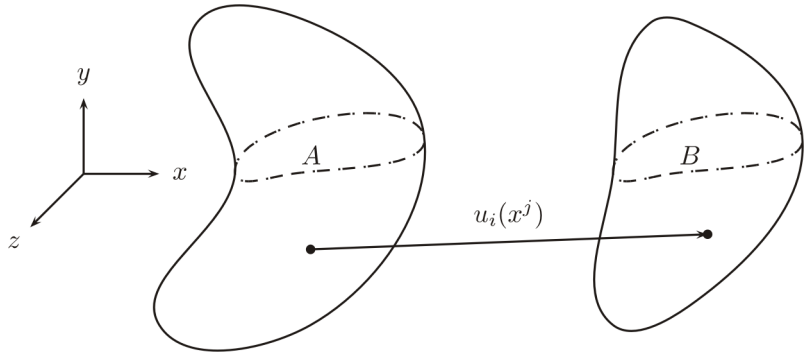


Figure 2.1: *The displacement vector $u_i(x^j)$ that joins the particle x^j in the underformed body A and the deformed body B.*

In [13] the term deformation refers to both the motion of particles within the body as well as motion of the body itself. Therefore loading a body may cause both rigid body deformation, like translation and/or rotation of the whole body, or cause straining within the body, which will make particles move relative to each other within the body and cause the body's shape to change.

We start off by studying the arbitrary body A in Figure 2.1 filled with a homogeneous material and no forces acting in on it. In the undeformed body, a coordinate system can be associated. To each point or particle in A we can fix a coordinate x^i. Applying a force on the body will cause the body to deform and we let the coordinate system respond in such a manner that each particle retains its coordinate. Therefore when we refer to the same particle in the deformed body B, we can refer to it using the same coordinate x^i. If the medium is described in this way it is said that we are using *particle or convected coordinates* and is popularly known as the Lagrange formalism. It is then possible to define the displacement vector as

$$\mathbf{u} = u_i(x^j). \tag{2.1}$$

The displacement vector is defined for every point inside the continuum and therefore defines a displacement field. The displacement field can be used to describe the strain experienced by the body. This relationship between the displacement experienced by the body and the strain is termed the displacement-strain relationship. The strain tensor

can be derived from the displacement field (see Appendix A) which results in the exact relation

$$\varepsilon_{ij} = \frac{1}{2}\left(u_i|_j + u_j|_i + u^k|_i u_k|_j\right). \tag{2.2}$$

This relation can be linearized by assuming that the deformations experienced by the body are small. Under this assumption the last term $u^k|_i u_k|_j$, which is quadratic in displacement can be neglected and equation (2.2) reduces to

$$\varepsilon_{ij} = \frac{1}{2}\left(u_i|_j + u_j|_i\right), \tag{2.3}$$

which is the linearized displacement-strain relation. The strain tensor in equation (2.3) is symmetrical

$$\varepsilon_{ij} = \varepsilon_{ji}, \tag{2.4}$$

since interchanging the subscripts yields the same equation. Obviously the same is not true about the general strain tensor in equation (2.2), since interchanging the subscripts would not yield the same equation for all coordinate systems, but if Cartesian tensors were used then $u^k|_i = u_k|_i$ therefore the quadratic term can be written as $u_k|_i u_k|_j$ from which it is clear that the general strain tensor would also be symmetric.

2.3 Stress

Imagine an arbitrarily orientated infinitesimal cut anywhere within the medium, depicted in Figure 2.2. It is easiest to think of the cut as a little disc, but the shape may be chosen arbitrarily. The cut therefore will have two sides. Both sides will be experiencing a force which would be the force exerted on it by the other side. The force per unit area experienced by the one face on the other would be the stress in the medium at that point. Strictly speaking it is not stress, but rather traction on the cut that we would measure.

If we wished to describe the stress that the material experiences at that point, then we would have to describe the tractions for all possible cuts. How to achieve such a description of stress in a material puzzled researchers, since there are infinitely many orientations. Luckily a very clever mathematician Augustine Cauchy showed in 1823 how to proceed. Cauchy showed that for a three dimensional continuum we only need three independent cuts to fully describe the stress state at any point within the medium, [13].

Mathematically we can represent the stress state with a second order tensor, the stress tensor σ^{ij}, which is related to traction t^j on a small patch with normal n_i by the relation

$$t^j = \sigma^{ij} n_i \tag{2.5}$$

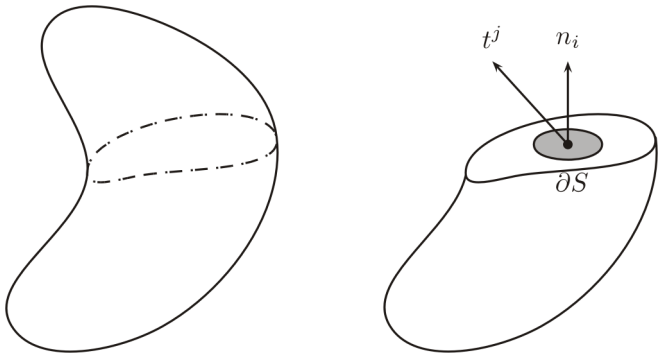


Figure 2.2: *Traction t^j experienced on a small patch ∂S with normal n_i inside the continuum*

In some texts the traction at a point is defined as the limit of the force Δf^j acting on an infinitely small patch ΔS, such that [14]

$$t^j = \lim_{\Delta S \to 0} \frac{\Delta f^j}{\Delta S}, \tag{2.6}$$

which is then termed traction on the surface ΔS. Similar arguments are used to relate traction to strain.

Cauchy also showed, using local equilibrium arguments, that the stress tensor is symmetric

$$\sigma^{ij} = \sigma^{ji}. \tag{2.7}$$

Throughout this text, unless explicitly mentioned, the compressive positive stress convention will be used. This is a matter of convenience since compressive normal stresses are more common that tensile ones in the mining environment. While in mechanical engineering when working with beams and bars it would be more convenient to use the compressive negative convention since there tensile stresses are more common.

2.4 Gravity

The gravitational body force will play a very important role in expressing the traction boundary conditions for the static stress modelling. The effects of the surface topology is relatively small in comparison to the overburden in deep mining applications, even explicitly taking into account the free surface would have little effect and therefore it will be ignored in the current work. A very simple model for the induced weight will be used that can only take into account different material zones.

The stress induced by the weight of soil is given by

$$\sigma_{zz}(z) = \int_0^z \rho(z) g dz, \tag{2.8}$$

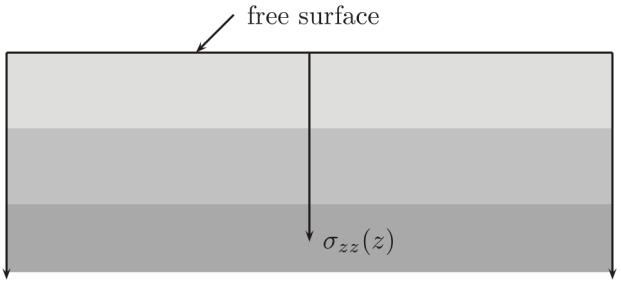


Figure 2.3: *Stress due to gravity*

where $\rho(z)$ is the soil density function above the point z as shown in Figure 2.3. The effect of the soil can be added to the stress at that point since the superposition principle applies to linear elasticity. If the topology was provided it would be possible to apply this model, since it only requires the computation of a line integral.

There are different ways of taking into account the free surface which does not require special kernels. One such method would be to model the free surface as a very large cave above the excavation. The modeller would have to make sure that the dimensions of the cave are much larger than the dimensions of the model. It would be prudent to test such a layout thoroughly by enlarging the cave and then testing how it influences the area where the model under analysis will be. The effect of such an enlargement would have to be negligible so that the error introduced does not compare to the expected accuracy of the solution.

Many numerical BEM applications model the in-situ stress state by specifying a locked-in stress $\sigma_{\text{locked-in}}$ and a depth variational component $\Delta\sigma_{\text{variational}}$. Therefore the stress state before any mining activity would be given as

$$\sigma_{\text{in-situ}}(z) = \sigma_{\text{locked-in}} + (z - d)\Delta\sigma_{\text{variational}} \tag{2.9}$$

where d is the datum at which the variational component should be zero. The z-coordinate is positive going down and therefore if the mines coordinate system is relative to sea level and the mine is located 500 meters above sea level, then $d = -500$. This formulation is useful when locked-in tectonic stresses are present and stress measurements are available with which to specify these two components.

2.5 Elastic Moduli

We will assume that tension is positive and compression negative to facilitate the description of the physical meaning of the elastic constants. Consider a homogeneous isotropic elastic rod, aligned with the x-axis and put under tension by a stress σ_{xx} as in Figure 2.4. The tension experienced by the rod will cause it to lengthen. The extensional strain ε_{xx}

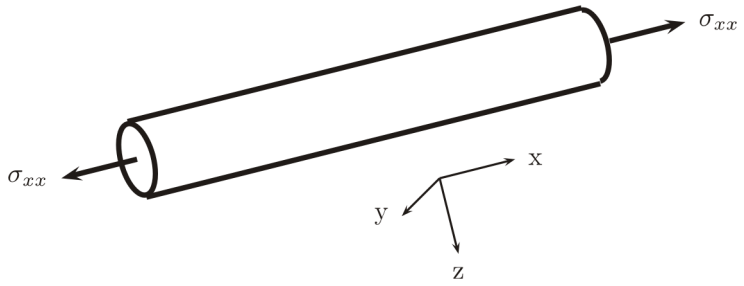


Figure 2.4: *Uniaxial tension applied to an elastic rod.*

is proportional to the stress and given by

$$\varepsilon_{xx} = \frac{1}{E}\sigma_{xx}. \tag{2.10}$$

The proportionality constant E is called the Young modulus and has dimensions of stress. When stretching a piece of rubber, which is the same as applying tension on it as in the case of the rod, then one will notice that the rubber also becomes thinner in the lateral directions relative to the direction in which the tension is applied. This thinning gives rise to equal lateral extensional strains ε_{yy} and ε_{zz} which is expressed in terms of the lengthening ε_{xx} through

$$\varepsilon_{yy} = \varepsilon_{zz} = -\nu\varepsilon_{xx} \tag{2.11}$$

$$= -\frac{\nu}{E}\sigma_{xx}, \tag{2.12}$$

with the dimensionless Poisson's ratio ν as the proportionality constant. Thus far we have only expressed the extensional strains in terms of the applied stress σ_{xx}.

The shear stresses σ_{ij} $(i \neq j)$ are related to the shear strains ε_{ij} $(i \neq j)$, via the shear modulus G, through the relationships

$$\sigma_{ij} = 2G\varepsilon_{ij} \qquad (i \neq j). \tag{2.13}$$

The shear modulus G can be expressed in terms of E and ν as

$$G = \frac{E}{2(1+\nu)}. \tag{2.14}$$

The Lamé parameters λ and μ are related to the Young modulus E and Poisson's ratio ν by the following set of equations

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \tag{2.15}$$

$$\mu = \frac{1}{2}\frac{E}{1+\nu}. \tag{2.16}$$

The first Lamé parameter λ is used to simplify the equations, but it does not carry any physical meaning like the other constants. The second Lamé parameter is μ and also called

the shear modlus G. The bulk modulus K relates the mean stress to the volumetric strain through

$$\frac{1}{3}\sigma_m^m = K\varepsilon_m^m. \tag{2.17}$$

A very useful Table is found in [13, p. 46] which shows the relationships between the elastic constants.

2.6 Hooke's Law

Hooke's law relates the components of the strain tensor to the components of the stress tensor using the stress-strain relations of the material given by a fourth order tensor E^{ijmn} which contains elastic moduli. Hooke's law for a general linear elastic anisotropic material can then be written as

$$\sigma^{ij} = E^{ijmn}\varepsilon_{mn}. \tag{2.18}$$

The number of entries that E^{ijmn} contains is limited due to the symmetry of both the stress and strain tensors. The symmetries of E^{ijmn} can compactly be expressed as

$$E^{ijmn} = E^{jimn} = E^{jinm} = E^{ijnm}. \tag{2.19}$$

The existence of a strain energy density [15, Flügge, p. 51] can be used to further reduce the number of independent moduli (see Appendix B), the result of which is another symmetry

$$E^{ijmn} = E^{mnij}, \tag{2.20}$$

which in turn brings the total of independent moduli down to 21.

If we impose the additional constraint of isotropy only two moduli are needed to describe the material. Using the Lamé elastic parameters λ and μ, the stress-strain relations can be written explicitly as

$$E^{ijmn} = \lambda g^{ij}g^{mn} + \mu(g^{im}g^{jn} + g^{in}g^{jm}) \tag{2.21}$$

and for Cartesian coordinates the metric tensors can be replaced by Kronecker deltas giving

$$E^{ijmn} = \lambda\delta^{ij}\delta^{mn} + \mu(\delta^{im}\delta^{jn} + \delta^{in}\delta^{jm}). \tag{2.22}$$

Inserting the general form of E^{ijmn} into Hooke's law and lowering the appropriate indices gives

$$\sigma_j^i = \lambda\varepsilon_m^m\delta_j^i + 2\mu\varepsilon_j^i. \tag{2.23}$$

Instead of expressing stress in terms of strain we can invert (2.23) and express strain in terms of stress as

$$\varepsilon^i_j = \frac{1+\nu}{E}\sigma^i_j - \frac{\nu}{E}\sigma^m_m\delta^i_j \tag{2.24}$$

in terms of Young's modulus and Poisson's ratio using the relations (2.15) and (2.16).

2.7 Compatibility Conditions

The compatibility conditions express the condition that the body stays continuous when deformed. This means that no cracks will form or overlaps will occur. If we wanted to compute the strain tensor at some point from the displacement field then, using equation (2.3), we would have to solve for each of the six strain components. This is very straightforward, but if we wished to solve for the displacement at some point we end up with three unknowns and six equations. The compatibility conditions exists to ensure that for a given set of six functions $\varepsilon_{ij}(x^k)$ there does exist a solution. The compatibility equations can be compactly expressed by

$$\varepsilon_{ij}|_{kl}\epsilon^{ikm}\epsilon^{jln} = 0, \tag{2.25}$$

which yields six different equations, with ϵ^{ijk} the permutation tensor [15]. In Cartesian coordinates the six compatibility equations, which can be derived from equation (2.25), are [13]

$$\begin{aligned}
2\frac{\partial^2\epsilon_{xy}}{\partial x\partial y} &= \frac{\partial^2\varepsilon_{xx}}{\partial y^2} + \frac{\partial^2\varepsilon_{yy}}{\partial x^2},\\
2\frac{\partial^2\epsilon_{xz}}{\partial x\partial z} &= \frac{\partial^2\varepsilon_{xx}}{\partial z^2} + \frac{\partial^2\varepsilon_{zz}}{\partial x^2},\\
2\frac{\partial^2\epsilon_{yz}}{\partial y\partial z} &= \frac{\partial^2\varepsilon_{yy}}{\partial z^2} + \frac{\partial^2\varepsilon_{zz}}{\partial y^2},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2\epsilon_{xx}}{\partial y\partial z} &= -\frac{\partial^2\varepsilon_{yz}}{\partial x^2} + \frac{\partial^2\varepsilon_{xz}}{\partial x\partial y} + \frac{\partial^2\varepsilon_{xy}}{\partial x\partial z},\\
\frac{\partial^2\epsilon_{yy}}{\partial x\partial z} &= -\frac{\partial^2\varepsilon_{xz}}{\partial y^2} + \frac{\partial^2\varepsilon_{xy}}{\partial y\partial z} + \frac{\partial^2\varepsilon_{yz}}{\partial x\partial y},\\
\frac{\partial^2\epsilon_{zz}}{\partial x\partial y} &= -\frac{\partial^2\varepsilon_{xy}}{\partial z^2} + \frac{\partial^2\varepsilon_{xz}}{\partial y\partial z} + \frac{\partial^2\varepsilon_{yz}}{\partial x\partial z}.
\end{aligned}$$

2.8 Equilibrium

Cauchy's equations of equilibrium for the stress tensor are given by [16; 17; 12]

$$0 = \sigma^{ij}|_i + b^j, \tag{2.26}$$

where b^j is the body force per unit volume. These equilibrium equations can be derived by considering the equilibrium of an infinitesimal material cube.

If a body is in equilibrium then the internal stresses has to equilibrate the surface tractions on the boundary. If the outward surface normal is n_i, the equilibrium condition leads to

$$t^j = \sigma^{ij} n_i, \tag{2.27}$$

where t^j are the surface tractions.

The Navier-Cauchy equations of equilibrium are then obtained by substituting Hooke's law equation (2.18), the displacement-strain relationships equation (2.3) and the homogeneous isotropic linear elastic stress-strain relationships equation (2.21) into equation (2.26)

$$\begin{aligned} 0 &= \sigma^{ij}|_i + b^j \\ &= (E^{ijmn}\varepsilon_{mn})|_i + b^j \\ &= \frac{1}{2}[E^{ijmn}(u_m|_n + u_n|_m)]|_i + b^j \\ &= \frac{1}{2}(\lambda g^{ij} g^{mn} + \mu(g^{im}g^{jn} + g^{in}g^{jm}))(u_m|_{ni} + u_n|_{mi}) + b^j \\ &= \mu u^j|^i_i + (\mu + \lambda) u^i|^j_i) + b^j. \end{aligned} \tag{2.28}$$

In Cartesian coordinates equation (2.28) can be written as

$$\mu u_{j,ii} + (\mu + \lambda) u_{i,ji} + b_j = 0. \tag{2.29}$$

2.9 Summary

In this chapter the basic theory describing the physics of a linear elastic continuum in equilibrium has been discussed. A general equation (2.2) for the strain tensor as a function of the displacement field in covariant form was given. This equation is valid for large deformations, but is non-linear. The small-strain assumption was used to obtain the linear displacement-strain relationship (2.3), which form the basis of linear elastic theory. The concepts of stress and tractions were then introduced, but no equations were given to relate them to strain. The effect of gravitational loading was discussed, because it is the major contributor to loading in mining. The many elastic constants, which describe the response of the elastic medium, and their relationships was given.

Then Hooke's law, which relates stress and strain for a general linear elastic anisotropic medium through the stress-strain relationships, was introduced. The stress-strain relationships were explicity written for a linear elastic isotropic medium in terms of the Lamé parameters in both general curvilinear coordinates (2.21) and Cartesian coordiantes (2.22).

The continuum has to stay continuous and the compatibility equations (2.25) express this constraint. Finally all the equations needed to describe a linear elastic continuum in equilibrium are put together. The constitutive law (2.18), the conservation of energy or equilibrium equations (2.26) and the continuity or compatibility conditions (2.25) are called the governing equations.

Chapter 3

Fundamental Solutions

3.1 Introduction

The building blocks of elasticity came from answering simple questions involving how loads within an elastic medium would cause it to respond, [13]. These solutions are referred to as fundamental solutions and named after the people who solved them. Fundamental solutions are solutions to the governing equations and are also called Green's functions or kernels, [18]. The governing equations are the constitutive law (2.18), the conservation of energy or equilibrium equations (2.26) and the continuity or compatibility conditions (2.25). The boundary element method uses these fundamental solutions, which can only be derived for linear homogeneous continua. This was initially thought to restrict the boundary element method to these kinds of media. Instead it is possible to model inhomogeneous media using the multi-region approach and non-linearity using internal cell division.

The solutions are for a point load within a linear homogeneous isotropic elastic continuum. In the framework of continuum mechanics this can be achieved, but in practice applying such a load to soil in an experiment would not be possible. Fundamental solutions are the core of the boundary element method. When applied in the boundary element method these fundamental solutions are smeared over elements describing the domain of interest within the continuum.

There also exist other solutions that do not involve point loads, for example the engineer Alfred Flamant used Boussinesq's solution to answer the question of how the continuum would react if a line load was applied normal to the surface of an elastic half-space. This was done by integrating Boussinesq's solution along an infinitely long line. These kinds of solutions are useful if plane strain conditions can be applied to reduce the dimensionality of the problem from three to two dimensions.

Kelvin's solution is only applicable for an infinite continuum, but is useful if the effect of the free surface can be ignored, such as for deep mining applications. Boussinesq's,

Cerrutti's and Mindlin's solutions are useful when the effect of the free surface is significant and has to be taken into account.

3.2 Kelvin's solution

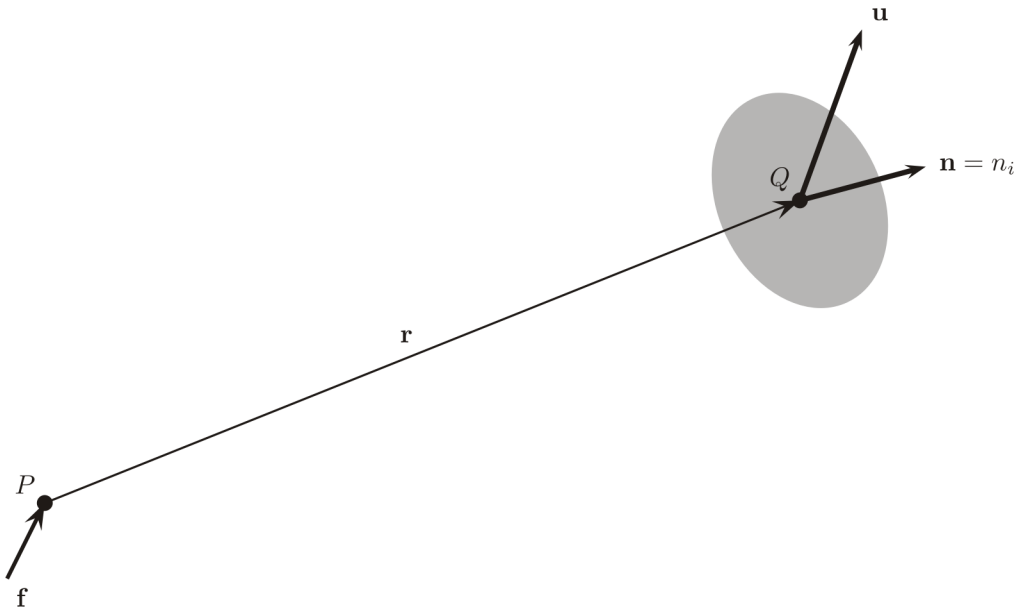


Figure 3.1: *Kelvin's solution for a point load* **f** *at the source point* P *and displacement* **u** *at the field point* Q.

The physicist William Thompson (Lord Kelvin) solved the problem, illustrated in Figure 3.1, of a point load acting within an infinite elastic continuum. Let P be the source point where the load $\mathbf{f}(P) = f^i(P)$ is applied and let Q be the field point where we wish to know the displacement $\mathbf{u}(Q) = u_j(Q)$. Kelvin's solution in kernel form describes the displacement field in Cartesian coordinates by the equation

$$u_j(Q) = U_{ij}(Q, P) f^i(P). \tag{3.1}$$

U_{ij} is known as the displacement kernel and relates the load at P with a displacement at Q and can be expressed in tensor notation as

$$U_{ij}(Q, P) = \frac{\kappa_u}{r}[B\delta_{ij} + r_{,i_Q} r_{,j_Q}] \tag{3.2}$$

where

$$\kappa_u = \frac{1+\nu}{8\pi E(1-\nu)}, \tag{3.3}$$

$$B = 3 - 4\nu. \tag{3.4}$$

The subscript i in U_{ij} indicates the direction of the load applied at P and the j subscript indicates the direction of the displacement at the field point Q. This is not so important for the U_{ij} kernel, which is symmetric about i and j, but it is important for the traction kernel which can be split into a symmetric and anti-symmetric tensor. The vector $r_i(P,Q)$ between the source point P and field point Q is given by

$$r_i = x_i(Q) - x_i(P). \tag{3.5}$$

The Euclidean distance r between P and Q will be expressed as

$$r = \sqrt{r_i r_i} \tag{3.6}$$

where the repeated subscript i implies summation over all the whole range (therefore, $r = \sqrt{r_1^2 + r_2^2 + r_3^2}$). Spatial derivatives are used very frequently in boundary element formulations and the comma convention used in tensor analysis will be used to indicate derivatives with respect to a coordinate. Therefore

$$r_{,i_Q} = \frac{\partial r}{\partial x_i(Q)} \tag{3.7}$$

is the spatial derivative of the distance with respect to the field point Q and using the chain rule of differentiation, we obtain

$$r_{,i_Q} = \frac{r_{i_Q}}{r}. \tag{3.8}$$

The spatial derivative with respect to the source point P is obtained in the same manner and

$$r_{,i_P} = -\frac{r_{i_P}}{r}, \tag{3.9}$$

but with the additional minus sign that comes from the definition of r_i in equation (3.5). The spatial derivative with respect to the source P will be used less often and therefore the subscript for the field point Q in equation (3.8) will be dropped, thus

$$r_{,i_Q} = r_{,i} = \frac{r_i}{r}. \tag{3.10}$$

These equations are easily verified, but very important, because they are used often in derivations and when implementing the kernel functions in code. The displacement kernel (3.2) can be expressed without the derivatives using (3.8) to obtain

$$U_{ij}(Q,P) = \kappa_u \left[B\frac{\delta_{ij}}{r} + \frac{r_i r_j}{r^3} \right], \tag{3.11}$$

which more distinctly shows the terms in the kernel.

The influence of the point load vanishes as the distance r between the field Q and source P point is made large. The decay of the kernel is inversely proportional to this distance

$$|u| \propto \frac{1}{r}|f|, \tag{3.12}$$

but as we come closer to the source point we see that the kernel becomes singular. If we were to integrate over a two dimensional element, we will see that the kernel is only weakly singular and we can easily remove the singularity. In the BEM the fundamental solutions exist solely to be used as kernel functions and will be integrated over elements used to discretize the boundary of some domain. When integrating over an element we can eliminate the weak singularity using some coordinate transformation like polar coordinates. The displacement kernel as given by equation (3.11) contains two terms ($1/r$ and $r_i r_j / r^3$) which are both inversely proportional to distance. Taking the limit as the distance goes to zero, we will see that each of the components r_i will also have to go to zero. Factors of the form

$$\lim_{r \to 0} \frac{r_i}{r} = 0 \tag{3.13}$$

will not cause a singularity at zero, but in both terms there remains a power of r in the denominator which produces a weak singularity.

The displacement kernel is the simplest kernel in linear elasticity. If we know displacements everywhere we would be able get strains by using the displacement-strain relationship (2.3), then apply Hooke's law to obtain stress and contract with a normal to get traction. This recipe can be applied to U_{ij} to derive the strain, stress and traction kernels which in turn relates a point load to strain, stress and traction everywhere in the continuum.

The strain kernel

$$\varepsilon_{jk}(Q) = \Xi_{ijk} f^i(P) \tag{3.14}$$

is obtained by applying the linearized displacement-strain relationship (2.3) on the displacement kernel (3.11), yielding

$$\Xi_{ijk} = \frac{1}{2}(U_{ij,k} + U_{ik,j}) \tag{3.15}$$

in terms of the displacement kernel. Next we focus on one of the two terms and expand the displacement kernel

$$U_{ij,k} = \kappa_u \left[B \frac{\delta_{ij}}{r} + \frac{r_i r_j}{r^3} \right]_{,k}, \tag{3.16}$$

now applying the spatial derivative and using the chain rule of differentiation we get

$$U_{ij,k} = \kappa_u \left[\frac{1}{r^3}(-B r_k \delta_{ij} + r_j \delta_{ik} + r_i \delta_{jk}) - \frac{3}{r^5} r_i r_j r_k \right]. \tag{3.17}$$

The second term in equation (3.15) is the same as the first term, except that the indices are swopped and therefore we get

$$\begin{aligned} U_{ik,j} &= \kappa_u \left[B\frac{\delta_{ik}}{r} + \frac{r_i r_k}{r^3} \right]_{,j} \\ &= \frac{\kappa_u}{r^3} \left[-B r_j \delta_{ik} + r_k \delta_{ij} + r_i \delta_{jk} - 3\frac{r_i r_j r_k}{r^2} \right]. \end{aligned} \tag{3.18}$$

Substituting equations (3.17) and (3.18) back into equation (3.15) and introducing a new constant

$$C = \frac{B-1}{2} = 1 - 2\nu, \tag{3.19}$$

yields

$$\begin{aligned} \Xi_{ijk} &= \frac{-\kappa_u}{r^3} \left[C(r_j \delta_{ik} + r_k \delta_{ij}) - r_i \delta_{jk} + 3\frac{r_i r_j r_k}{r^2} \right] \\ &= \frac{-\kappa_u}{r^2} \left[C(r_{,j} \delta_{ik} + r_{,k} \delta_{ij}) - r_{,i} \delta_{jk} + 3 r_{,i} r_{,j} r_{,k} \right]. \end{aligned} \tag{3.20}$$

The strain kernel is symmetric with respect to the indices

$$\Xi_{ijk} = \Xi_{ikj}, \tag{3.21}$$

because of the symmetry of the strain tensor.

The stress kernel

$$\sigma_{jk}(Q) = \Sigma_{ijk} f^i(P) \tag{3.22}$$

is obtained by applying Hooke's law (equation (2.18)) to the strain kernel equation (3.20), therefore

$$\Sigma_i^{jk} = E^{jkmn} \Xi_{lmn} \tag{3.23}$$

and substituting the stress-strain relationships yields

$$\begin{aligned} \Sigma_i^{jk} &= \left[\lambda \delta_{jk} \delta_{mn} + \mu(\delta_{jm}\delta_{kn} + \delta_{jn}\delta_{km}) \right] \Xi_{imn} \\ &= \lambda \delta_{jk} \Xi_{imm} + \mu(\Xi_{ijk} + \Xi_{ikj}). \end{aligned} \tag{3.24}$$

For the first term

$$\Xi_{imm} = \frac{-\kappa_u}{r^2} \left[C(r_{,m}\, \delta_{im} + r_{,m}\, \delta_{im}) - r_{,i} \delta_{mm} + 3 r_{,i}\, r_{,m}\, r_{,m} \right] \tag{3.25}$$

we use the identities $r_{,m}\, r_{,m} = 1$ and $\delta_{mm} = 3$ to obtain

$$\Xi_{imm} = \frac{-2\kappa_u C}{r^2} r_{,i}\,. \tag{3.26}$$

Back substituting equations (3.26) and (3.20) into (3.24), yields

$$\Sigma_i^{jk} = \frac{-\kappa_t}{r^2} \left[C(r_{,j}\delta_{ik} + r_{,k}\delta_{ij} - r_{,i}\delta_{jk}) + 3 r_{,i} r_{,j} r_{,k} \right]. \tag{3.27}$$

with

$$\begin{aligned}\kappa_t &= 2\mu\kappa_u \\ &= \frac{1}{8\pi(1-\nu)}.\end{aligned} \tag{3.28}$$

The traction kernel

$$t_j(Q) = T_{ij} f^i(P) \tag{3.29}$$

is then obtained by applying the equilibrium condition on the surface equation (2.27) with normal n_k to the stress kernel equation (3.27) which gives

$$T_{ij} = \Sigma_{ijk} n_k \tag{3.30}$$

where

$$T_{ij} = \frac{-\kappa_t}{r^2}\left[C(r_{,j}\delta_{ik} + r_{,k}\delta_{ij} - r_{,i}\delta_{jk}) + 3r_{,i}r_{,j}r_{,k}\right] n_k \tag{3.31}$$

and simplifying where possible

$$T_{ij} = \frac{-\kappa_t}{r^2}\left[(C\delta_{ij} + 3r_{,i}r_{,j})r_{,k}n_k + C(r_{,j}n_i - r_{,i}n_j)\right], \tag{3.32}$$

and introducing the normal derivative of the distance

$$T_{ij} = \frac{-\kappa_t}{r^2}\left[(C\delta_{ij} + 3r_{,i}r_{,j})\frac{\partial r}{\partial n} + C(r_{,j}n_i - r_{,i}n_j)\right], \tag{3.33}$$

where

$$\frac{\partial r}{\partial n} = r_{,k}n_k, \tag{3.34}$$

we get the same traction kernel as in [14, p. 15].

The strain, stress and traction kernels all exhibit the same behaviour with respect to distance between the source and field point. The traction influence kernel vanishes much faster than the displacement kernel at a rate proportional to the inverse of the square of the distance

$$\left.\begin{matrix}\varepsilon_{jk}(Q) \\ \sigma_{jk}(Q) \\ t_j(Q)\end{matrix}\right\} \propto \frac{1}{r^2} f^i(P). \tag{3.35}$$

They produce a strong singularity at the source point P which will give some trouble when integrating these kernels in the boundary element method.

Table 3.2 summarizes the influence functions for the contributions of the fictitious force (FF) components to the displacement and stress at the field point. This table is similar to the table provided by [19, p. 166] and is useful for computing and implementing the boundary element integrals.

	Force along x (f_1)	Force along y (f_2)	Force along z (f_3)
u_x	$\kappa_u(\frac{B}{r} + \frac{r_1^2}{r^3})$	$\kappa_u(\frac{r_1 r_2}{r^3})$	$\kappa_u(\frac{r_1 r_3}{r^3})$
u_y	$\kappa_u(\frac{r_1 r_2}{r^3})$	$\kappa_u(\frac{B}{r} + \frac{r_2^2}{r^3})$	$\kappa_u(\frac{r_2 r_3}{r^3})$
u_z	$\kappa_u(\frac{r_1 r_3}{r^3})$	$\kappa_u(\frac{r_2 r_3}{r^3})$	$\kappa_u(\frac{B}{r} + \frac{r_3^2}{r^3})$
σ_{xx}	$\kappa_t(\frac{Cr_1}{r^3} + \frac{3r_1^3}{r^5})$	$\kappa_t(-\frac{Cr_2}{r^3} + \frac{3r_1^2 r_2}{r^5})$	$\kappa_t(-\frac{Cr_3}{r^3} + \frac{3r_1^2 r_3}{r^5})$
σ_{yy}	$\kappa_t(-\frac{Cr_1}{r^3} - \frac{3r_1 r_2^2}{r^5})$	$\kappa_t(\frac{Cr_2}{r^3} + \frac{3r_2^3}{r^5})$	$\kappa_t(-\frac{Cr_3}{r^3} + \frac{3r_2^2 r_3}{r^5})$
σ_{zz}	$\kappa_t(-\frac{Cr_1}{r^3} + \frac{3r_1 r_3^2}{r^5})$	$\kappa_t(-\frac{Cr_2}{r^3} + \frac{3r_2 r_3^2}{r^5})$	$\kappa_t(\frac{Cr_3}{r^3} + \frac{3r_3^3}{r^5})$
σ_{xy}	$\kappa_t(\frac{Cr_2}{r^3} + \frac{3r_1^2 r_2}{r^5})$	$\kappa_t(\frac{Cr_1}{r^3} + \frac{3r_1 r_2^2}{r^5})$	$\kappa_t(\frac{3r_1 r_2 r_3}{r^5})$
σ_{yz}	$\kappa_t(\frac{3r_1 r_2 r_3}{r^5})$	$\kappa_t(\frac{Cr_3}{r^3} + \frac{3r_2^2 r_3}{r^5})$	$\kappa_t(\frac{Cr_2}{r^3} + \frac{3r_2 r_3^2}{r^5})$
σ_{zx}	$\kappa_t(\frac{Cr_3}{r^3} + \frac{3r_1^2 r_3}{r^5})$	$\kappa_t(\frac{3r_1 r_2 r_3}{r^5})$	$\kappa_t(\frac{Cr_1}{r^3} + \frac{3r_1 r_3^2}{r^5})$

Table 3.1: *Fictitious force kernel contributions for each component.* Constants: $\kappa_u = \frac{1+\nu}{8\pi E(1-\nu)}$, $\kappa_t = \frac{1}{8\pi(1-\nu)}$, $B = 3 - 4\nu$, $C = 1 - 2\nu$

3.3 Boussinesq's solution

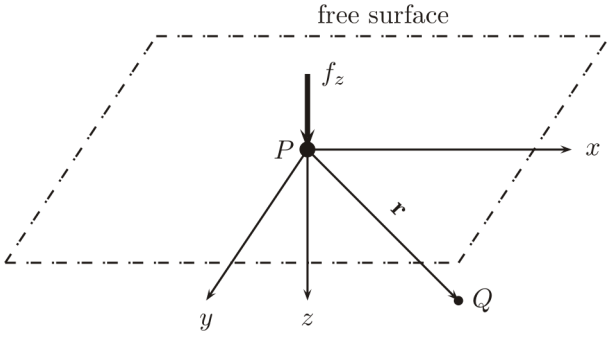


Figure 3.2: *Boussinesq's solution for u_i and σ^{ij} at the field point Q within the medium for a load f_z normal to the elastic half-space at the source point P.*

The mathematician Joseph Boussinesq solved the problem, illustrated in Figure 3.2, of a point load acting normal to the surface of an elastic half-space. This solution has many practical applications, especially in the civil engineering industry where the effect of piles and foundations are considered. The infinite space is divided into two half-spaces by the free surface. The free surface defines a boundary on which the traction is specified to be zero everywhere. Physically the free surface could be thought of the ground surface and the half-space underneath as the soil which leaves the half-space above to be the air. It is easiest to write Boussinesq's solution in cylindrical coordinates with the origin centered at the source point and using the compressive positive convention.

In cylindrical coordinates the position vector $\mathbf{r}$ and length $R = \|\mathbf{r}\|_2$ is given by

$$\begin{aligned} \mathbf{r} &= x_i(Q) - x_i(P) = \{r, \theta, z\}, \\ R &= \sqrt{(r^2 + z^2)}. \end{aligned}$$

The displacement vector is expressed as

$$\begin{aligned} u_r &= \frac{f_z}{4\pi\mu R}\left[\frac{rz}{R^2} - \frac{(1-2\nu)r}{R+z}\right], \\ u_\theta &= 0, \\ u_z &= \frac{f_z}{4\pi\mu R}\left[2(1-\nu) + \frac{z^2}{R^2}\right]. \end{aligned}$$

The strain tensor can then be derived from the displacement vector using equation (2.3). Thereafter the stress-strain relationships are applied using Hooke's law (2.18) to write the following equations for the six components of the symmetric stress tensor:

$$\begin{aligned} \sigma_{rr} &= -\frac{f_z}{2\pi}\left[\frac{(1-2\nu)}{R(R+z) - \frac{3r^2 z}{R^5}}\right], \\ \sigma_{\theta\theta} &= -\frac{P(1-2\nu)}{2\pi}\left[\frac{z}{R^3} - \frac{1}{R(R+z)}\right], \\ \sigma_{zz} &= \frac{f_z}{2\pi}\left[\frac{3z^3}{R^5}\right], \\ \sigma_{rz} &= \frac{f_z}{2\pi}\left[\frac{3rz^2}{R^5}\right], \\ \sigma_{r\theta} &= \sigma_{\theta z} = 0. \end{aligned}$$

3.4 Cerrutti's solution

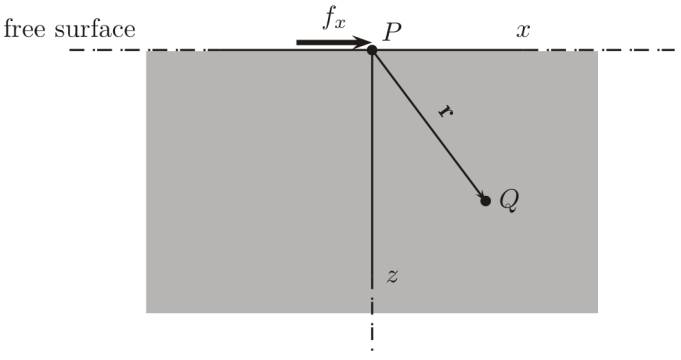


Figure 3.3: *Cerrutti's solution for a point load f_x along the surface at the source point P and displacement $\mathbf{u}$ at the field point Q.*

V. Cerrutti solved the problem, illustrated in Figure 3.3, of a point load $f_x(P)$ acting along the surface of an elastic half-space. The resulting solution does not exhibit the same

radial symmetry as either Boussinesq's or Kelvin's problems and have to be written using Cartesian coordinates.

The displacement field is given by

$$\begin{aligned}
u_x &= \frac{f_x}{4\pi\mu r}\left[1+(r_{,x})^2+(1-2\nu)\left(\frac{1}{1+r_{,z}}-\frac{(r_{,x})^2}{(1+r_{,z})^2}\right)\right],\\
u_y &= \frac{f_x}{4\pi\mu r}\left[r_{,x}r_{,y}-(1-2\nu)\frac{r_{,x}r_{,y}}{(1+r_{,z})^2}\right],\\
u_z &= \frac{f_x}{4\pi\mu r}\left[r_{,x}r_{,z}+(1-2\nu)\frac{r_{,x}}{1+r_{,z}}\right]
\end{aligned}$$

and the stress field

$$\begin{aligned}
\sigma_{xx} &= -\frac{f_x r_{,x}}{2\pi r^2}\left[-3(r_{,x})^2+\frac{1-2\nu}{(1+r_{,z})^2}\left(1-(r_{,y})^2-\frac{2(r_{,y})^2}{1+r_{,z}}\right)\right],\\
\sigma_{yy} &= -\frac{f_x r_{,x}}{2\pi r^2}\left[-3(r_{,y})^2+\frac{1-2\nu}{(1+r_{,z})^2}\left(1-(r_{,x})^2-\frac{2(r_{,x})^2}{1+r_{,z}}\right)\right],\\
\sigma_{zz} &= \frac{3f_x r_{,x}(r_{,z})^2}{2\pi r^2},\\
\sigma_{xy} &= -\frac{f_x r_{,y}}{2\pi r^2}\left[-3(r_{,x})^2+\frac{1-2\nu}{(1+r_{,z})^2}\left(-1+(r_{,x})^2+\frac{2(r_{,x})^2}{1+r_{,z}}\right)\right],\\
\sigma_{yz} &= \frac{3f_x r_{,x}r_{,y}r_{,z}}{2\pi r^2},\\
\sigma_{xz} &= \frac{3f_x r_{,z}(r_{,x})^2}{2\pi r^2}.
\end{aligned}$$

Cerrutti's solution together with Boussinesq's solution can be used to obtain the displacement due to a general load applied on the half-space surface.

3.5 Mindlin's solutions

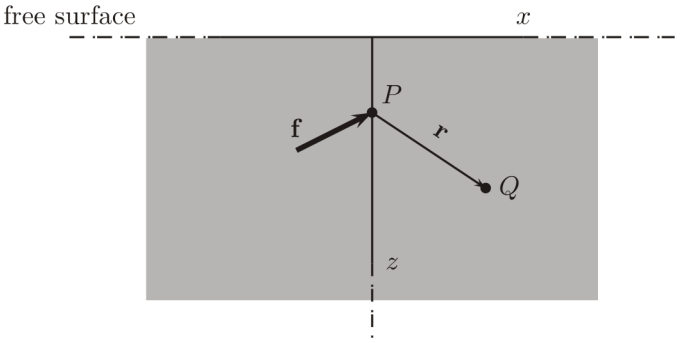


Figure 3.4: *Mindlin's solutions for a point load* **f** *acting with an elastic half-space at source point P and displacement* **u** *at the field point Q.*

Raymond Mindlin solved both the problems, illustrated in Figure 3.4, of a point load acting vertically and horizontally within an elastic half-space. These solutions are quite

complex and can be written as additional terms to be added to Boussinesq's and Cerrutti's solutions. For brevity these additional terms will not be given here, but an interested reader is referred to [13, p. 96].

3.6 Summary

This chapter gave an overview of fundamental solutions available in elasticity and mainly focused on Kelvin's solution. The fundamental solutions give the response of the continuum for a point load. Kelvin's solution also referred to as the displacement kernel (3.11) gives the displacement everywhere in an infinite homogeneous linear elastic medium. The elasticity theory discussed in the previous chapter was applied to produce kernels for strain (3.20), stress (3.27) and traction (3.33). A quick mention was given of Boussinesq's and Cerrutti's solutions with equations for displacements and stress and an even quicker mention of Mindlin's solution without any equations.

Chapter 4

Cracks

4.1 Introduction

A crack can be modelled as a dislocation in the continuum. Figure 4.1 depicts such a crack. The sides of the crack are named, top and bottom, respectively. This naming is dependent of the coordinate system and in rheology it is convenient to choose the z-coordinate to increase with depth. The other two coordinate axes might be chosen to point along compass directions to form a right-handed coordinate system. Given such a coordinate system the top of a crack is reached when taking the limit going down along increasing z and the bottom when taking the limit coming up along decreasing z. Let $\mathbf{n}$ denote the normal on the top. The bottom face of the crack is parallel to the top and therefore its normal on the bottom face is $-\mathbf{n}$. Consider a load free continuum without any cracks or fractures. In such a continuum every point in the continuum would have a unique coordinate. If a crack is magically created by making an incision without causing a stress field or deformation, then each point on the top will have a matching point on the bottom face. This means that the top and bottom faces would coincide and that between the two faces there would be zero dislocation.

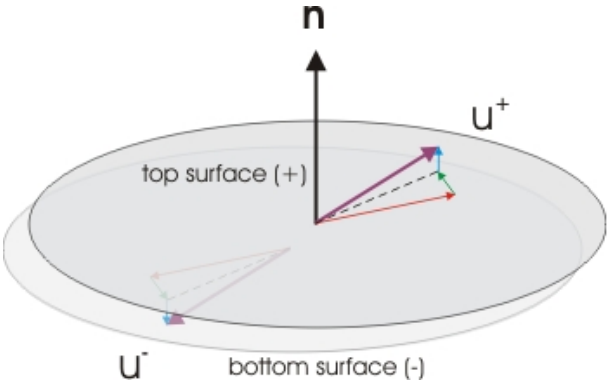


Figure 4.1: *Elementary crack element.*

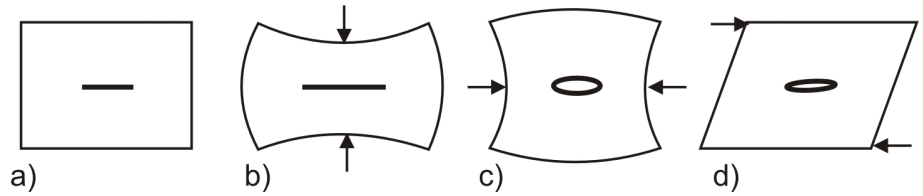


Figure 4.2: *Crack inside an elastic block which is a) unloaded, b) vertically loaded, c) horizontally loaded in plane and d) horizontally loaded off plane.*

In Figure 4.2 we examine a possible deformation of a crack within an elastic block under different loading scenarios. The figure represents four different loading situations:

a) no load at all,

b) a load applied vertically and normal to the face of the crack,

c) a load applied horizontally in the plane of the crack and lastly

d) a load applied horizontally causing the block to shear.

In the first two cases no dislocation between the two faces occurs. The relative location of points on the top and bottom faces remain the same, except that in the second case the crack might have become elongated. In the third case the crack is compressed and opening such that the top and bottom faces experiences a relative vertical dislocation. In the fourth case the crack experiences both opening and relative horizontal dislocation. In all cases except b), the crack faces experience zero traction boundary conditions, because they're not touching or pressing against the other side. In b) though, the two faces are pressing against each other and this crack is considered fully closed. Mathematically it is possible to allow these two faces to interpenetrate. This might be useful when modelling very thin cuts with finite thickness in an elastic continuum. The thickness of tabular mining depends on the thickness of the reef being mined, which might vary from place to place, but generally it is of the order of a meter. This thickness known as the stoping width is much smaller than the panels being mined. The whole mining layout which is on the order of a couple of kilometers contains many such panels. Therefore it is possible to approximate the stoping width by using cracks which have no thickness to start with, allowing interpenetration and then limiting the interpenetration so that it does not go beyond the stoping width.

The dislocation or displacement discontinuity (DD) will be given as

$$\Delta u = u^+ - u^-, \tag{4.1}$$

with u^+ the displacement on the top side of the crack and u^- the displacement on the bottom side of the crack. Notice that the DD is defined as the bottom face's displacement

minus the top face's displacement of the crack. On the boundary of the crack the regularity condition is imposed, this means that no discontinuity in the displacements are allowed on the boundary, therefore $\Delta u = 0$ on dS, which is required by the compatibility equations (2.25) to ensure a unique solution, [20].

4.2 Betti's reciprocal identity

The reciprocal theorem links the solutions to two different boundary value problems for the same body Ω bounded by a surface S. The theorem is a direct consequence of the linearity of the equilibrium equations (2.26) and of the generalized Hooke's law (2.18). Consider two equilibrium states in the region Ω. The stresses and strains for the two states are expressed as $(\sigma^{ij}, \varepsilon_{ij})$ and $(\tilde{\sigma}^{ij}, \tilde{\varepsilon}_{ij})$ [16]. Multiplying Hooke's law on both sides by $\tilde{\varepsilon}_{ij}$ gives

$$\begin{aligned} \sigma^{ij}\tilde{\varepsilon}_{ij} &= E^{ijmn}\varepsilon_{mn}\tilde{\varepsilon}_{ij} && (4.2)\\ &= (E^{mnij}\tilde{\varepsilon}_{mn})\varepsilon_{ij} && (4.3)\\ &= (E^{ijmn}\tilde{\varepsilon}_{mn})\varepsilon_{ij} && (4.4)\\ &= \tilde{\sigma}^{ij}\varepsilon_{ij}. && (4.5) \end{aligned}$$

First the indices are renamed to produce equation (4.3) and then the symmetry of the stress-strain relations (2.20), due to the existence of a strain energy function, is used to write (4.4). Finally, Hooke's law is substituted in the second equilibrium state to get equation (4.5). Integration over the domain Ω produces the integral statement

$$\int_\Omega \sigma^{ij}\tilde{\varepsilon}_{ij}d\Omega = \int_\Omega \tilde{\sigma}^{ij}\varepsilon_{ij}d\Omega. \tag{4.6}$$

The work done by the stresses of the first system on the strains of the second system is equal to the work done by the stresses of the second system on the strains of the first system [18]. Integrating the left hand side by parts produces

$$\begin{aligned} \int_\Omega \sigma^{ij}\tilde{\varepsilon}_{ij}d\Omega &= \int_\Omega \sigma^{ij}\tilde{u}_i|_j d\Omega && (4.7)\\ &= \int_\Omega \sigma^{ij}\tilde{u}_i n_j dS - \int_\Omega \tilde{u}_i\sigma^{ij}|_j d\Omega && (4.8)\\ &= \int_\Omega t^i\tilde{u}_i dS + \int_\Omega b^i\tilde{u}_i d\Omega, && (4.9) \end{aligned}$$

for which the linearized displacement-strain relations (2.3), traction on the boundary (2.27) and equilibrium equations (2.26) have been used. The same can be done to the right hand side to yield

$$\int_\Omega \tilde{\sigma}^{ij}\varepsilon_{ij}d\Omega = \int_\Omega \tilde{t}^i u_i dS + \int_\Omega \tilde{b}^i u_i d\Omega, \tag{4.10}$$

from which Betti's reciprocal identity is obtained by substituting back equations (4.9) and (4.10) into (4.6)

$$\int_\Omega t^i \tilde{u}_i dS + \int_\Omega b^i \tilde{u}_i d\Omega = \int_\Omega \tilde{t}^i u_i dS + \int_\Omega \tilde{b}^i u_i d\Omega. \tag{4.11}$$

4.3 Somigliana identity

The indices in Betti's identity (4.11) are renamed to obtain

$$\int_\Omega t^j \tilde{u}_j dS + \int_\Omega b^j \tilde{u}_j d\Omega = \int_\Omega \tilde{t}^j u_j dS + \int_\Omega \tilde{b}^j u_j d\Omega. \tag{4.12}$$

The tractions and displacements are unknown in those parts of the boundary for which they have not been prescribed as boundary conditions. Let the state (t^j, u_j) represent these unknowns. Let the other set $(\tilde{t}^j, \tilde{u}_j)$ represent the set of known displacements and traction that should be valid for any geometry in equilibrium [14]. The Kelvin solution represents such a possible set for displacements and tractions at any surface point due to a unit load applied at the interior in an infinite domain. It is important to point out that using the fundamental solution is not the only possible choice and it might be more convenient to use simple functions that satisfy the governing equations [12].

The unknown quantities for the first state can be expressed in terms of the quantities of the second state. Let

$$\begin{aligned} u_j &= u_j(Q), \\ t^j &= t^j(Q), \\ b^j &= b^j(Q), \\ \tilde{u}_j &= U_{ij}(Q,P)e^i, \\ \tilde{t}^j &= T_i^j(Q,P)e^i, \\ \tilde{b}^j &= \delta_i^j \delta(Q,P)e^i \end{aligned}$$

where Q is a field point on the boundary surface and a point load is applied at P for which e^i are the components of a unit vector $\hat{\mathbf{e}}$ in the direction of the point load. δ^{ij} is the Kronecker delta and $\delta(Q,P)$ the Dirac delta. Substituting these equations into (4.12) it can be seen that e^i is common to all the integrals. It is therefore possible to write equations for each component separately [16], yielding

$$\begin{aligned} &\int_\Omega t^j(Q) U_{ij}(Q,P) dS(Q) + \int_\Omega b^j(Q) U_{ij}(Q,P) d\Omega(Q) \\ = &\int_\Omega T_i^j(Q,P) u_j(Q) dS(Q) + \int_\Omega \delta_i^j \delta(Q,P) u_j(Q) d\Omega(Q). \end{aligned} \tag{4.13}$$

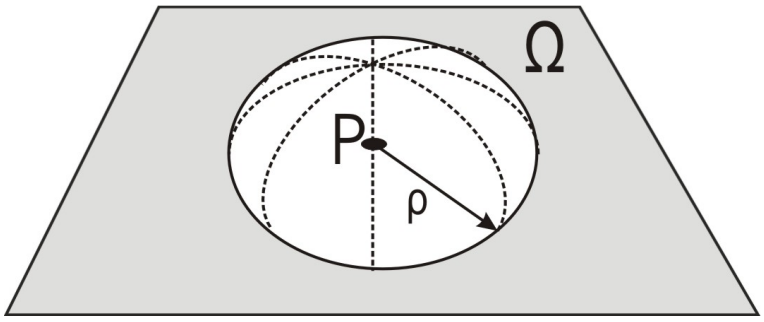


Figure 4.3: *Obtaining the BIE's using a limiting process on the boundary.*

Usage of the properties of the Kronecker and Dirac delta function and reshuffling then yields

$$u_i(P) = -\int_\Omega T_i^j(Q,P)u_j(Q)dS(Q) + \int_\Omega U_{ij}(Q,P)t^j(Q)dS(Q) \tag{4.14}$$

$$+\int_\Omega U_{ij}(Q,P)b^j(Q)d\Omega(Q). \tag{4.15}$$

This equation is called the *Somigliana identity for displacements* [14]. The equation can be used to compute a displacement anywhere inside the domain Ω, [21]. The fundamental solutions become singular on the boundary $d\Omega$ and therefore it can not directly be applied for points on the boundary $d\Omega$. Outside the body the integrals in equation (4.15) are zero. Integration is carried out with respect to the field point Q, the normal in the traction kernel is associated with the surface at the field point Q and that the summation is carried out over the subscript j and not i.

Boundary integral equations (BIE's) can be obtained by using a limiting process to let the point P coincide with the boundary surface as shown in Figure 4.3. The normal way of doing this, is to expand the domain Ω by a small region at the point of the boundary where the source point P would coincide with the boundary. This region is then shrunk in a symmetric way using a limiting process to obtain the BIE's. The BIE's can then be used to form a well posed boundary value problem and calculate the known boundary parameters [16].

4.4 Boundary Integral Equations For Cracks

Consider the finite region Ω with a crack inside as show in Figure 4.4. Write Somigliana's identity, using the stress kernel (3.22) instead of the traction kernel (3.29), assume that

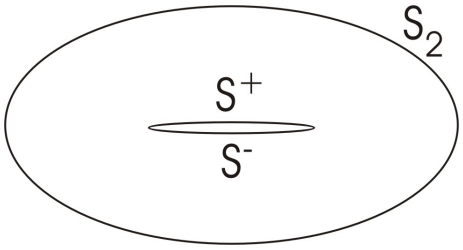


Figure 4.4: *Region Ω containing a crack $S_1 = S^+ \bigcap S^-$ and bounded by an outer surface S_2.*

body forces are zero and that the crack surface is free from traction, gives

$$\begin{aligned} u_i(P) &= \int_{S_1+S_2} (U_{ij}(Q,P)t^j(Q) - \Sigma_i^{jk}(Q,P)n_k(Q)u_j(Q))dS(Q) \\ &= \int_{S_2} (U_{ij}(Q,P)t^j(Q) - \Sigma_i^{jk}(Q,P)n_k(Q)u_j(Q))dS(Q) \\ &- \int_{S^+} \Sigma_i^{jk}(Q,P)n_k^+(Q)u_j^+(Q)dS(Q) \\ &- \int_{S^-} \Sigma_i^{jk}(Q,P)n_k^-(Q)u_j^-(Q)dS(Q). \end{aligned} \tag{4.16}$$

The two crack faces are assumed to be practically coinciding, therefore let $S = S^- = S^+$ and write the normals of the bottom in terms of the top surface, $n^+ = -n^- = n$ and substitute (4.1) into (4.16) to get

$$\begin{aligned} u_i(P) &= \int_{S_2} (U_{ij}(Q,P)t^j(Q) - T_i^j(Q,P)u_j(Q))dS(Q) \\ &- \int_{S^+} T_i^j(Q,P)\Delta u_j(Q)dS(Q). \end{aligned} \tag{4.17}$$

This equation is valid for any finite domain. Letting the outer surface expand to infinity produces the solution of a crack in an infinite elastic space,

$$u_i(P) = u_i^\infty(P) - \int_S T_i^j(Q,P)\Delta u_j(Q)dS(Q), \tag{4.18}$$

where $u^\infty(P)$ is the solution without the crack.

The traction kernel in equation (4.18) is not the same as the traction kernel in equation (3.33) because the spatial differentiation is carried out with respect to the field point Q and not the source point P, but they are closely related because differentiation of r at the source and field point only differ by a minus sign. Therefore equation (4.18) is rewritten as

$$u_i(P) = u_i^\infty(P) - \int_S V_i^j(Q,P)\Delta u_j(Q)dS(Q) \tag{4.19}$$

where

$$V_i^j(Q,P) = T_i^j(Q,P) \tag{4.20}$$

will be used for the DD displacement kernel. Strain at P can be computed from equation (4.19) using the displacement strain relationship and therefore

$$\varepsilon_{ij}(P) = \varepsilon_i^{\infty}(P) - \int_S \frac{1}{2}(V_i^k(Q,P)|_{j^P} + V_j^k(Q,P)|_{i^P})\Delta u_k(Q)dS(Q) \tag{4.21}$$

which expresses strain in terms of a DD. Stress is derived by applying Hooke's law to the strain, therefore

$$\sigma^{ij}(P) = (\sigma^{\infty}(P))^{ij} - \int_S W^{ijk}(Q,P)\Delta u_k(Q)dS(Q), \tag{4.22}$$

where

$$W^{ijk}(P) = E^{mnij}\frac{1}{2}(V_m^k(Q,P)|_{n^P} + V_n^k(Q,P)|_{m^P}) \tag{4.23}$$

is used to denote the DD stress kernel. To obtain traction, equation (4.22) is contracted with a normal at the source point, thus

$$t^i(P) = (t^{\infty}(P))^i - \int_S n_j(P)W^{ijk}(Q,P)\Delta u_k(Q)dS(Q). \tag{4.24}$$

4.5 Displacement Discontinuity Kernels

Kelvin's solution (3.11) gives the response of the continuum due to a concentrated unit FF. This fictitious load is also known as a force-discontinuity [19; 22], because the stress near the point of application jumps from tensile to compressive. The displacements throughout the continuum are continuous. Another kind of loading which is useful for crack-like structures are "displacement discontinuities". The term displacement discontinuity (DD) is due to [21, Crouch and Starfield], but these loads are also referred to as "double-layer potentials", "Volterra dislocations", dislocation densities and "double couples", depending on the field in which they are used. In contrast to force-discontinuities which are only fictitious forces, the DD represent the actual convergence and ride components of the material.

The DD kernels can be derived from Kelvin's solution by applying the traction operator at the source point. Applying the traction operator on a displacement field at a particular point yields the traction at that point. Therefore if $\mathbf{T_n}$ is the traction operator, applying it to a displacement field $u(x)$, where $x \in D$ is a point in the domain D on which the displacements are defined, then

$$t(x) = T_n u(x) \tag{4.25}$$

gives the traction at that point on the plane with normal n. The traction operator is the linearized displacement-strain relationship (2.3) applied to the displacement field u_i, followed by applying the stress-strain relationships (2.23) and then contracted with the

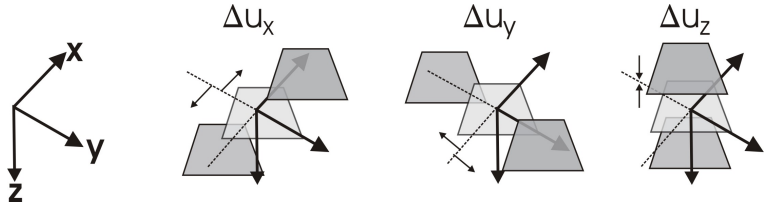


Figure 4.5: *Displacement discontinuity modes.*

normal n_k. Throughout the rest of this chapter Cartesian coordinates will be assumed to improve readability and therefore the upper and lower position of the indices is immaterial. Expanding the traction operator and applying it to a displacement field gives

$$t^l = E^{ijkl}\frac{1}{2}(u_{i,j} + u_{j,i})n_l. \tag{4.26}$$

The connection between the DD kernels and Kelvin's solution comes from recognizing that the FF could be obtained from such a displacement field. Therefore applying the traction operator on Kelvin's solution at the source point is the same as changing the unknown, the force-discontinuity, to a DD.

Kelvin's fundamental solution (equation (3.2)) gives the displacement due to a fictitious force f^i

$$u_j = U_{ij}f^i. \tag{4.27}$$

Figure 4.5 depicts the different modes of a DD in the local coordinate system for an infinitesimal plane. Δu_x and Δu_y is the extent of displacement between two points in the continuum on the infinitesimal plane and is also known as the ride components. The meaning of Δu_z depends whether the value is positive or negative. If positive it is the amount of convergence, closure or interpenetration between the two surfaces of the infinitesimal plane. If it is negative it is the amount of dilation or opening between the two surfaces of the infinitesimal plane. Now let us rewrite the previous equation and express the displacement as the result of a DD (Δu^k) and transform the kernel as described. The displacement is then given by

$$u_j = E^{ijkm}\frac{1}{2}(U_{ij,s} + U_{sj,i})n_m\Delta u^k, \tag{4.28}$$

where n_n is the normal of the plane across which the DD is applied. Remember that the derivatives are taken at the source point and therefore equation (3.9) should be used.

Next we expand the stress-strain relationships

$$u_j = \left[\lambda\delta^{is}\delta^{km} + \mu(\delta^{ik}\delta^{sm} + \delta^{im}\delta^{sk})\right]\frac{1}{2}(U_{ij,s} + U_{sj,i})n_m\Delta u^k, \tag{4.29}$$

and contract the appropriate terms to simplify the equation,

$$u_j = (\lambda U_{ij,i}\,\delta^{km} + \mu(U_{kj,m} + U_{mj,k}))n_m\Delta u^k. \tag{4.30}$$

The general form for derivatives of Kelvin's solution at the source point is

$$U_{ij,k} = \frac{1+\nu}{8\pi E(1-\nu)}\frac{1}{r^2}\left[((3-4\nu)r_{,k}\,\delta_{ij} - r_{,j}\,\delta_{ik} - r_{,i}\,\delta_{jk}) + 3r_{,i}\,r_{,j}\,r_{,k}\right]. \tag{4.31}$$

Substituting the subscripts we obtain

$$\begin{aligned} u_j &= \frac{1+\nu}{8\pi E(1-\nu)}\frac{1}{r^2}\Big(\lambda\left[((3-4\nu)r_{,i}\,\delta_{ij} - r_{,j}\,\delta_{ii} - r_{,i}\,\delta_{ji}) + 3r_{,i}\,r_{,i}\,r_{,j}\right]\delta_{km} + \\ &\quad \mu\left[((3-4\nu)r_{,m}\,\delta_{kj} - r_{,j}\,\delta_{km} - r_{,k}\,\delta_{jm}) + 3r_{,k}\,r_{,j}\,r_{,m}\right] + \\ &\quad \mu\left[((3-4\nu)r_{,k}\,\delta_{mj} - r_{,j}\,\delta_{mk} - r_{,n}\,\delta_{jk}) + 3r_{,m}\,r_{,j}\,r_{,k}\right]\Big)\, n_m \Delta u^k. \end{aligned} \tag{4.32}$$

Let us now use the identities $\delta_{ii} = 3$, $r_{,i}\,r_{,i} = 1$ to obtained a simplified equation

$$\begin{aligned} u_j &= \frac{1+\nu}{8\pi E(1-\nu)}\frac{1}{r^2}\Big[2\lambda(1-2\nu)r_{,j}\,\delta_{km} + \\ &\quad 2\mu\left(((1-2\nu)r_{,m}\,\delta_{jk} + (1-2\nu)r_{,k}\,\delta_{jm} - r_{,j}\,\delta_{km}) + 3r_{,j}\,r_{,k}\,r_{,m}\right)\Big]\, n_m \Delta u^k \end{aligned} \tag{4.33}$$

Now we substitute the Lamé parameters to obtain displacements as a function of a DD

$$u_j = \frac{1}{8\pi(1-\nu)}\frac{1}{r^2}\Big[(1-2\nu)(\delta_{jk}r_{,m} + \delta_{jm}r_{,k} - r_{,j}\,\delta_{km}) + 3r_{,j}\,r_{,k}\,r_{,m}\Big]\, n_m \Delta u^k. \tag{4.34}$$

The DD displacement kernel

$$V_{kj} = \frac{1}{8\pi(1-\nu)}\frac{1}{r^2}[(1-2\nu)(\delta_{jk}r_{,m}\,n_m + n_j r_{,k} - r_{,j}\,n_k) + 3r_{,j}\,r_{,k}\,r_{,m}\,n_m], \tag{4.35}$$

which is equal to the negative of the traction kernel (3.33) for fictitious forces.

The DD stress kernel can now be derived from the displacement kernel using the same technique as used for Kelvin's solution. Essentially we apply the traction operator to the displacement at the field point. We use the newly derived V_{kj} kernel to express the displacement at field point

$$u_j = V_{kj}\Delta u_k. \tag{4.36}$$

Using the displacement strain relationship we now express the strain due to the DD as

$$\epsilon_{ij} = \frac{1}{2}(V_{kj,i} + V_{ki,j})\Delta u_k, \tag{4.37}$$

and then get the stress using the stress-strain relationship

$$\sigma_{mn} = E^{ijmn}\epsilon_{ij}. \tag{4.38}$$

The traction can be obtained from the stress by contracting with a normal η at the field point

$$t_m = \sigma_{mn}\eta_n \tag{4.39}$$

Substituting the displacement kernel (4.35) into equation (4.38) we get

$$\sigma_{mn} = E^{kjmn}\frac{1}{2}(V_{kj,i}+V_{ki,j})\Delta u_k. \tag{4.40}$$

Next we substitute in the linearized stress-strain relationship to get

$$\sigma_{mn} = (\lambda\delta^{ij}\delta^{mn}+\mu(\delta^{im}\delta^{jn}+\delta^{in}\delta^{jm}))\frac{1}{2}(V_{ki,j}+V_{kj,i})\Delta u_k. \tag{4.41}$$

The Lamé parameter λ can be expressed in terms of the rigidity as

$$\lambda = \mu\frac{2\nu}{1-2\nu}. \tag{4.42}$$

Therefore we can factor out the rigidity and get

$$\sigma_{mn} = \mu\left(\frac{2\nu}{1-2\nu}V_{ki,i}\,\delta^{mn}+\frac{1}{2}(V_{kn,m}+V_{km,n})\right)\Delta u_k \tag{4.43}$$

The general form of the spatial derivative $V_{ki,j}$ is

$$\begin{aligned}\frac{r^3}{\kappa_u}V_{ki,j} &= C\,(n_j\delta_{ik}+\delta_{jk}n_i-\delta_{ij}n_k)-3C\,(\delta_{ki}r_{,j}\,r_{,m}\,n_m+r_{,k}\,r_{,j}\,n_i-r_{,j}\,r_{,i}\,n_k)\\ &+ 3\,(\delta_{jk}r_{,i}\,r_{,m}\,n_m+\delta_{ij}r_{,k}\,r_{,m}\,n_m+r_{,k}\,r_{,i}\,n_j)-15r_{,i}\,r_{,j}\,r_{,k}\,r_{,m}\,n_m,\end{aligned} \tag{4.44}$$

where $C = 1-2\nu$, therefore we get

$$\begin{aligned}\frac{r^3}{2\kappa_u}(V_{ki,j}+V_{kj,i}) &= C\,(n_i\delta_{jk}+\delta_{ik}n_j-\delta_{ij}n_k)+3\,(\delta_{ij}r_{,k}\,r_{,m}\,n_m+Cr_{,i}\,r_{,j}\,n_k)\\ &+ 3\nu\,(\delta_{ik}r_jr_mn_m+\delta_{jk}r_{,i}\,r_{,m}\,n_m+r_{,i}\,r_{,k}\,n_j+r_{,j}\,r_{,k}\,n_i)\\ &- 15r_{,i}\,r_{,j}\,r_{,k}\,r_{,m}\,n_m\end{aligned} \tag{4.45}$$

and

$$\frac{r^3}{(1-2\nu)\kappa_u}V_{ki,i} = n_k-3r_{,k}\,r_{,m}\,n_m \tag{4.46}$$

which we can substitute back into equation (4.43)

$$\begin{aligned}\sigma_{ij} &= \frac{1}{\kappa_t r^3}[2\nu(n_k-3r_{,k}\,r_{,m}\,n_m)\delta_{ij} &(4.47)\\ &+ (1-2\nu)\,(n_i\delta_{jk}+\delta_{ik}n_j-\delta_{ij}n_k)\\ &+ 3\,(\delta_{ij}r_{,k}\,r_{,m}\,n_m+(1-2\nu)r_{,i}\,r_{,j}\,n_k)\\ &+ 3\nu\,(\delta_{ik}r_{,j}\,r_{,m}\,n_m+\delta_{jk}r_{,i}\,r_{,m}\,n_m+r_{,i}\,r_{,k}\,n_j+r_{,j}\,r_{,k}\,n_i)\\ &- 15r_{,i}\,r_{,j}\,r_{,k}\,r_{,m}\,n_m]\Delta u_k, &(4.48)\end{aligned}$$

where

$$\kappa_t = \frac{E}{8\pi(1-v^2)}. \tag{4.49}$$

After some simplification we arrive at

$$\begin{aligned}\sigma_{ij} &= \frac{1}{\kappa_t r^3}[3r_{,m}\,n_m((1-2\nu)\delta_{ij}r_{,k}+\nu(\delta_{ik}r_{,j}+\delta_{jk}r_{,i}\,)-5r_{,i}\,r_{,j}\,r_{,k}\,)\\ &+ 3\nu(r_{,i}\,r_{,k}\,n_j+r_{,j}\,r_{,k}\,n_i)+(1-2\nu)(3r_{,i}\,r_{,j}\,n_k+\delta_{ik}n_j+\delta_{jk}n_i)\\ &- (1-4\nu)n_k\delta_{ij}]\Delta u_k \end{aligned} \tag{4.50}$$

from which the DD stress kernel can be isolated, such that

$$\begin{aligned}W_{ijk} &= \frac{1}{\kappa_t r^3}[3r_{,m}\,n_m(C\delta_{ij}r_{,k}+\nu(\delta_{ik}r_{,j}+\delta_{jk}r_{,i}\,)-5r_{,i}\,r_{,j}\,r_{,k}\,)\\ &+ 3\nu(r_{,i}\,r_{,k}\,n_j+r_{,j}\,r_{,k}\,n_i)+C(3r_{,i}\,r_{,j}\,n_k+\delta_{ik}n_j+\delta_{jk}n_i)\\ &- (1-4\nu)n_k\delta_{ij}]. \end{aligned} \tag{4.51}$$

The DD displacement kernel $V_{ij} \propto \frac{1}{r^2}$ has a pole at the loading point which will require a Cauchy-type integral. The stress kernel $W_{ijk} \propto \frac{1}{r^3}$ has a higher order pole at the loading point which will require an integral interpreted in the Hadamard finite part sense. These kernels correspond to the equations given on [16, p. 32]. Let η_i be a normal at the field point, then

$$\begin{aligned}W_{ik} &= W_{ijk}\eta_j\\ &= \frac{1}{\kappa_t r^3}[3(\nu\delta_{ik}-5r_{,i}\,r_{,k}\,)\frac{\partial r}{\partial n}\frac{\partial r}{\partial \eta}\\ &+ 3(C\eta_i r_{,k}+3\nu r_{,i}\,\eta_k)\frac{\partial r}{\partial n}+3(Cr_{,i}\,n_k+3\nu n_i r_{,k}\,)\frac{\partial r}{\partial \eta}\\ &+ (C\delta_{ik}+3\nu r_{,i}\,r_{,k}\,)n_m\eta_m+Cn_i\eta_k-(1-4\nu)n_k\eta_i] \end{aligned} \tag{4.52}$$

is the equation for the DD traction kernel.

Table 4.5 is obtained by rotating the coordinate system such that the normal $n_m = \delta_{m3}$ of the DD coincides with the z-coordinate. The DD displacement kernel

$$V_{kj} = \frac{1}{8\pi(1-\nu)}\frac{1}{r^2}[(1-2\nu)(\delta_{jk}r_{,3}+\delta_{j3}r_{,k}-r_{,j}\,\delta_{k3})+3r_{,j}\,r_{,k}\,r_{,3}\,] \tag{4.53}$$

and the DD stress kernel

$$\begin{aligned}W_{ijk} &= \frac{1}{\kappa_t r^3}[3r_{,3}\,(C\delta_{ij}r_{,k}+\nu(\delta_{ik}r_{,j}+\delta_{jk}r_{,i}\,)-5r_{,i}\,r_{,j}\,r_{,k}\,)\\ &+ 3\nu(r_{,i}\,r_{,k}\,n_j+r_{,j}\,r_{,k}\,n_i)+C(3r_{,i}\,r_{,j}\,n_k+\delta_{ik}n_j+\delta_{jk}\delta_{i3})\\ &- (1-4\nu)n_k\delta_{ij}]. \end{aligned} \tag{4.54}$$

and using the identity $(r_{,1}\,)^2+(r_{,2}\,)^2+(r_{,3}\,)^2=1$, the kernels can be tabulated as in [19].

	Ride along x (Δu_1)	Ride along y (Δu_2)	Convergence along z (Δu_3)
u_x	$\kappa_u\left(\frac{Cr_3}{r^3} + \frac{3r_3r_1^2}{r^5}\right)$	$\kappa_u\left(\frac{3r_1r_2r_3}{r^5}\right)$	$\kappa_u\left(-\frac{Cr_1}{r^3} + \frac{3r_1r_3^2}{r^5}\right)$
u_y	$\kappa_u\left(\frac{3r_1r_2r_3}{r^5}\right)$	$\kappa_u\left(\frac{Cr_3}{r^3} + \frac{3r_3r_2^2}{r^5}\right)$	$\kappa_u\left(-\frac{Cr_2}{r^3} + \frac{3r_2r_3^2}{r^5}\right)$
u_z	$\kappa_u\left(\frac{Cr_1}{r^3} + \frac{3r_1r_3^2}{r^5}\right)$	$\kappa_u\left(\frac{Cr_2}{r^3} + \frac{3r_2r_3^2}{r^5}\right)$	$\kappa_u\left(\frac{Cr_3}{r^3} + \frac{3r_3^3}{r^5}\right)$
σ_{xx}	$\kappa_t\left(\frac{3r_1r_3}{r^5} - \frac{15r_1^3r_3}{r^7}\right)$	$\kappa_t\left(\frac{3Cr_2r_3}{r^5} - \frac{15r_1^2r_2r_3}{r^7}\right)$	$\kappa_t\left(\frac{1+C}{r^3} - \frac{3Cr_2^2}{r^5} - \frac{15r_1^2r_3^2}{r^7}\right)$
σ_{yy}	$\kappa_t\left(\frac{3Cr_1r_3}{r^5} - \frac{15r_1r_2^2r_3}{r^7}\right)$	$\kappa_t\left(\frac{3r_2r_3}{r^5} - \frac{15r_2^3r_3}{r^7}\right)$	$\kappa_t\left(\frac{1+C}{r^3} - \frac{3Cr_1^2}{r^5} - \frac{15r_2^2r_3^2}{r^7}\right)$
σ_{zz}	$\kappa_t\left(\frac{3r_1r_3}{r^5} - \frac{15r_1r_3^3}{r^7}\right)$	$\kappa_t\left(\frac{3r_2r_3}{r^5} - \frac{15r_2r_3^3}{r^7}\right)$	$\kappa_t\left(\frac{1}{r^3} + \frac{6Cr_3^2}{r^5} - \frac{15r_3^4}{r^7}\right)$
σ_{xy}	$\kappa_t\left(\frac{3\nu r_2r_3}{r^5} - \frac{15r_1^2r_3^2}{r^7}\right)$	$\kappa_t\left(\frac{3\nu r_1r_3}{r^5} - \frac{15r_1r_2r_3^2}{r^7}\right)$	$\kappa_t\left(\frac{3Cr_1r_2}{r^5} - \frac{15r_1r_2r_3^2}{r^7}\right)$
σ_{yz}	$\kappa_t\left(\frac{3\nu r_1r_2}{r^5} - \frac{15r_1r_2r_3^2}{r^7}\right)$	$\kappa_t\left(\frac{1+\nu}{r^3} - \frac{3\nu r_1^2}{r^5} - \frac{15r_2^2r_3^2}{r^7}\right)$	$\kappa_t\left(\frac{3r_2r_3}{r^5} - \frac{15r_2r_3^3}{r^7}\right)$
σ_{zx}	$\kappa_t\left(\frac{1+\nu}{r^3} - \frac{3\nu r_2^2}{r^5} - \frac{15r_1^2r_3^2}{r^7}\right)$	$\kappa_t\left(\frac{3\nu r_1r_2}{r^5} - \frac{15r_1r_2r_3^2}{r^7}\right)$	$\kappa_t\left(\frac{3r_1r_3}{r^5} - \frac{15r_1r_3^3}{r^7}\right)$

Table 4.1: *Displacement discontinuity kernel contributions for each component.* Constants: $\kappa_u = \frac{1}{8\pi(1-\nu)}$, $\kappa_t = \frac{E}{8\pi(1-\nu^2)}$, $C = 1 - 2\nu$.

4.6 Summary

In this chapter, the boundary integral equations were derived using Betti's reciprocal work theorem and the Somigliana identity. These equations form the basis for the direct boundary element method. The boundary integral equations were applied to the crack model to obtain equations for the displacements (4.19), strain (4.21), stress (4.22) and traction (4.24). The explicit form of the kernels was not given. Section 4.5 derives the explicit form of the kernels using Kelvin's solution directly. The indirect boundary element method uses these kernels, but the aim was to highlight the link between the direct and indirect method.

Chapter 5

Boundary Element Method

5.1 Introduction

Linear elasticity and many other engineering problems are posed as boundary value problems [21]. There exists a region or body B of interest, bounded by a contour C in two-dimensions and a surface S in three-dimensions. Physics is used to describe the behaviour of the body and expressed as partial differential equations. Certain constraints or conditions are placed on the boundary of the domain and the solution of the relevant parameters are sought which will describe the state of the body. In a few simple situations it is possible to solve these partial differential equations analytically. In other cases we are forced to use numerical techniques. There are two classes of numerical methods, namely domain and boundary methods. The finite element method and finite difference method are examples of domain methods. The boundary element method is an example of a boundary method.

Work on the boundary element method was begun by Jaswon, Ponter and Symm [23] at the Department of Mathematics at Imperial College, London. This work was done on Laplace's equation in two dimensions using the direct formulation and the first papers on their work were published in 1963. The boundary element method was shown to be a promising alternative to the more widespread and simpler finite element and finite difference methods.

The boundary element method has the possibility of being computationally significantly less expensive than finite elements which require $O(n^3)$ variables within the body. In contrast the BEM only requires $O(n^2)$ variables on the boundary of the body, where n represents the resolution along a particular dimension under analysis [24].

The domain boundary S could be regarded as the boundary for two separate problems. The most natural would be the boundary for an interior problem where the body is bounded by the boundary surface. Instead it could also be the boundary for the exterior problem which encompasses the rest of the infinite space and for which the boundary

surfaces delineates cavities within the space. If the same problem was tackled using a domain method then a sufficiently large region around the cavity would have to be included. This region would have to be large enough so that the artificial boundary conditions did not influence the results around the cavity [21].

Recasting the problem and using the BEM reduces the problem dimensionality by one. Knowing the solution of the boundary value problem (BVP) on the boundary of the domain, uniquely determines the solution within the body. The solution to the BVP can then be used to accurately model the far-field with the use of influence functions, which finite element methods can not do [25]. In the case of elasticity, this could be the deformation or stress state everywhere inside and on the boundary of the body.

Even though large scale problems have to be tackled, there is still a need to quickly solve reasonably complicated problems on a routine basis. Better algorithms with less time and storage complexity allows the user to tackle these problems on common desktop computers without the need for parallel computing clusters which might involve lots of legwork by the user.

For large scale underground mining applications spanning kilometers, domain methods are not feasible, even if huge clusters for parallel computing are available. The popular method for such problems uses finite element discretization on adaptive unstructured meshes [3]. This requires construction of unstructured meshes in 3D, which for huge volumes of space with high complexity can be very difficult and computationally intensive. The boundary element method in 3D does not suffer from this level of difficulty, since only the boundary surface need to be discretized. It is still possible and common to couple the two methods. This might be done if it is necessary for a more fine grained analysis in certain areas.

5.2 Integral Equations

In contrast to the finite element method that solves partial differential equations, the BEM solves integral equations. Therefore we first have to formulate our problem as an integral equation. The form of an integral equation can be written as [7]

$$\phi(x) = \int_S G(x,y)\rho(y)dS(y) \tag{5.1}$$

with ϕ known everywhere on the surface S. $G(x,y)$ is the kernel function and ρ the unknown density function or loads on S.

The aim is to obtain a solution ρ which will satisfy equation (5.1) everywhere on the boundary S. The kernels we will consider are those of Kelvin's solution for displacement (3.11) and traction (3.33) depending whether the prescribed boundary conditions ϕ are displacements or tractions. These kernels are only applicable for volumetric excavations,

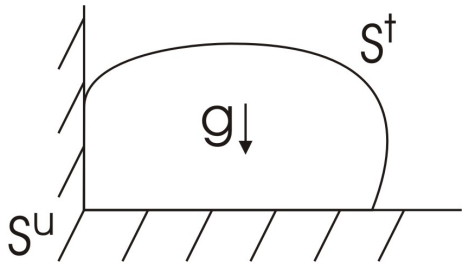


Figure 5.1: *Mixed boundary conditions.*

for crack-like geometries the displacement discontinuity kernels for displacement (4.35) and traction (4.52) will be used also depending on whether the prescribed boundary conditions ϕ are displacements or tractions.

The boundary surface can be expressed as the union $S = S^u \bigcup S^t$, such that the intersection $S^u \bigcap S^t = \oslash$, where S^u and S^t have got prescribed displacement and traction, respectively.

A mixed system is allowed, but, to guarantee unique solutions and a well posed BVP, boundary conditions have to be prescribed over the whole boundary. The boundary surface will have parts on which displacements have been prescribed and other parts on which tractions have been prescribed, but no part may have both displacement and traction prescribed. Displacement and traction correspond to Dirichlet and Neumann boundary conditions respectively for the BVP being solved. Figure 5.1 shows an elastic medium under gravitational loading with fixed displacements on the left and bottom half and zero tractions boundary conditions everywhere else.

5.3 Treffz method

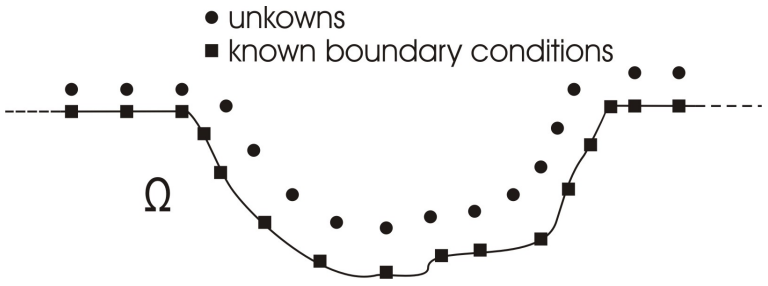


Figure 5.2: *Example of the Treffz method.*

The Treffz method places sources or loads around the object under investigation. These loads are placed far enough away from the boundary so that a linear system is well conditioned. Enough loads have to be used to ensure that the combined influence of

all the loads creates a smooth field which can reproduce the required physical quantities inside the body of the object. The loads must be outside the body, which implies that the problem must have enough room to place these loads. This is not true when studying cracks, since cracks have almost no space inside by definition.

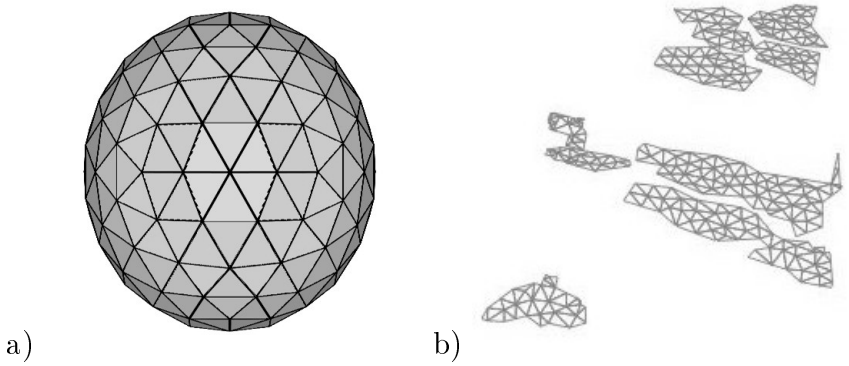
Figure 5.2 shows a side view of a pit where boundary conditions are known at squares on the boundary and the unknowns are at the circles in the air. The unknown loads can then be found by solving a linear system, such as

$$\phi(x_i) = \sum_j G(x_i, y_j)\rho(y_j) \tag{5.2}$$

where $\phi(x_i)$ are known boundary conditions at $x_i \in \partial\Omega$ on the surface of the domain Ω and $\rho(y_j) \notin \Omega$ are the unknown loads outside the domain. The Treffz method sounds very simple, but all numerical techniques have their own difficulties. The number of points to place, the distance to move loads away from boundary conditions, how to move the loads and the matrix solver are examples of some of the problems that will have to be addressed.

The BEM is essentially an extension to the Treffz method, but instead of placing the loads outside, the loads are placed on the boundary of the object. Therefore BEM does not need any room to distribute the loads. Unfortunately the kernel functions are singular and therefore integrating over them can be difficult. The Treffz method does not have this problem because it side stepped singular integration by moving the load points away from the boundary conditions.

5.4 Discretization



a) b)

Figure 5.3: *a) Volumetric and b) tabular discretization.*

To draw a sphere on a computer screen, the sphere, which is an infinitely smooth mathematical object, would have to be represented as a set of polygons. Today's computer hardware only works with triangles and quadrilaterals. Polygons and more complicated

geometries are all broken up into triangles or quadrilaterals and fed to the hardware. In some cases even the quadrilaterals are split into triangles. The act of partitioning the geometry into simpler treatable pieces is called discretization. To discretize something is to break it up into a countable set of elements.

A very coarse approximation of a sphere is a tetrahedron. The three vertices of the tetrahedron could have been chosen to lie on the sphere or instead they could have been chosen such that the tetrahedron has the same volume as the sphere. In both cases it would be difficult to convince anyone that the tetrahedron is indeed a sphere. The next step of refinement would be a cube and so forth. Ultimately using subdivision and remapping each vertex onto the sphere will produce something that will look like a perfect sphere on a users computer screen. For insanely sharp eyed people the dimensions of the largest triangle will have to be less than the size of a pixel. Therefore by using enough triangles one would be able to approximate almost any continuous surface to a predetermined accuracy.

The boundary surface S in a three dimensional space can be approximated in many ways using different kinds of geometrical elements. The aim of the discretization is twofold. It can be used to approximate the geometry of the problem. The problem geometry could be very intricate and it could be difficult to represent using flat triangles. It can be used to approximate the unknown density, which might be a fictitious force or displacement discontinuity. The geometric and functional discretization does not have to be the same. Flat elements might approximate the geometry well enough, but it might be a bad approximation of the unknown density. If constant flat elements are used, then the geometric and functional discretization matches. The geometry might have to be further subdivided or refined until the requirements for functional approximation is met.

The geometry being modelled in mines could be volumetric like caves, as shown in Figure 5.3 a), shafts and tunnels or tabular, as shown in Figure 5.3 b), like reefs and stopes. The maps used to build numerical models for mines are sometimes very inaccurate. Tunnels are winding with rocks jutting out and in no way smooth or straight as shown on the map. Stopes try to follow the reef, which in many cases are rolling and varies in thickness. This information is often unknown or ignored and the mine's average stoping is assumed instead. Therefore what is known on the maps are just an approximation of what is really underground. The reef and development tunnels are in many cases delineated by markers which are spaced relatively far appart. It is possible to use very high order elements which would fit almost any surface to approximate the mine layout, but most probably using flat or linear elements would fit the known data best. Using a discretization which interpolates these markers are probably the best that one can hope for to be able to assume that the model is a sufficient approximation of the actual layout in the mine. Many times the model is another approximation on top of the survey data.

Flat triangles are very useful to approximate three dimensional surfaces with. This is because one can approximate any three dimensional surface using just flat triangles. The only negative aspect is that in some situations one might require quite a lot of them. It might be advantageous to use curved elements instead, because one would need fewer elements to achieve the same accuracy. The number of elements used will determine how long it will take to solve the problem. Rectangular elements are simpler to integrate over than other kinds of elements.

5.5 Collocation

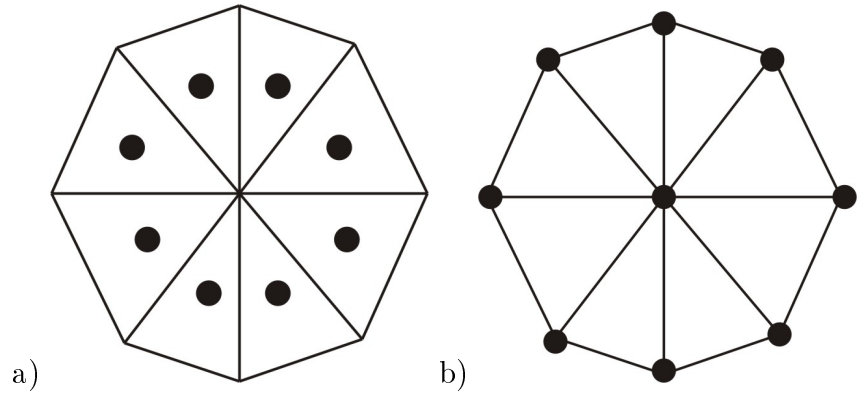


Figure 5.4: *Collocation points could be a) inside an element or b) shared between elements.*

The aim of the collocation method is to find a load distribution which will interpolate the boundary conditions. The boundary conditions are only specified at certain points on the boundary, called the collocation points. The loads and the boundary conditions are both put on the boundary, which is where the term collocation comes from. A collocation point might be contained within a boundary element or it might be shared by more than one element. The boundary elements associated with a collocation point is called the support of the collocation point.

Using a first-order collocation method it is assumed that the density $\rho(x_i) = \rho_i$ is piecewise constant over each of the boundary elements. For simplicity it is assumed that each collocation point is associated with only one boundary element and that the collocation point is located at the element's centroid. It is known that the solution to the integral equation is smooth. Therefore it is also known that there is an optimal position where to place the collocation point. Placing the collocation point inside the element away from its edges allows the development of integration schemes which can avoid the collocation point. It is preferable to avoid the collocation point, because when integrating over the element the kernel function has a pole at the collocation point.

Figure 5.4 depicts a discretization with constant and linear elements. The collocation points are the circles placed at the centroids of the constant elements and vertices of the linear elements. The support for the constant elements is much smaller than the support of the linear elements.

The collocation method tries to satisfy the boundary conditions exactly at the collocation points. The linear system can then be expressed as

$$\phi(x_i) = \sum_{j}^{N} \left(\int_{S_j} G(x_i, y) dS_j(y) \right) \rho(x_j) \qquad 1 \leq j \leq N. \tag{5.3}$$

The density $\rho(x_j)$ was taken out of the integral, because it is constant over the element. If $\rho(x_j)$ represented a displacement discontinuity, then $\rho(x_j)$ would have physical meaning and could be directly interpreted.

Define the influence matrix $A = \{a_{ij}\}$ as

$$a_{ij} = \int_{S_j} G(x_i, y) dS_j(y). \tag{5.4}$$

The linear system can then compactly be written as $A\rho = \phi$. A direct method like Gauss elimination would require the evaluation of all the kernel functions to construct a matrix with N^2 elements. This system would be dense and semi-diagonally dominant, but not enough to guarantee convergence using iterative solvers like Jacobi or Gauss-Seidel.

Constant elements have a big disadvantage, because quite a lot of elements are needed to obtain a decent solution for the boundary conditions. Instead of adding more elements, it is possible to use higher order elements to represent the density function. This can be done by introducing linear shape functions. If collocation points are placed at the vertices then the meshing strategy would have to be able to create discretization without free internal nodes. Vertices should be shared between elements, because otherwise artificial pillars and structures will be created which could make the linear system unstable. Linear elements with only internal nodes could also be used. It is preferable to place these internal nodes wisely, because choosing them at equally spaced points (in squares or triangles) or on a Gaussian product grid (for triangles) could lead to very bad results [26].

The linear system with shared vertices can then be expressed as

$$\phi(x_i) = \sum_{j}^{N} \sum_{k \in V_j} \left(\int_{S_j} G(x_i, y) \psi_{jk}(y) dS_j(y) \right) \rho(x_k) \qquad 1 \leq j \leq N, \tag{5.5}$$

where V_j is the set of vertices associated with j-th element and ψ_{jk} is the shape function associated with both the j-th element and the vertex x_k.

The influence matrix $A = \{a_{ij}\}$ for linear approximation of the density function would then be defined as

$$a_{ij} = \sum_{k \in E_j} \int_{S_j} G(x_i, y) \psi_{jk}(y) dS_j(y), \tag{5.6}$$

where E_j is the support of the j-th vertex, which is all the elements associated with the j-th vertex.

5.6 Galerkin

The Galerkin method or variational approach solves the integral equation in a least squares sense. It tries to minimize the solution over the whole boundary instead of interpolating the integral equation at points, which the collocation method aims to do. The Galerkin method incurs a penalty because it requires an additional surface integration. Even though the additional integration might be considered bad, it can be used to soften the singularities which makes the integration easier.

A drawback of the collocation method is that the influence matrix is non-symmetric [27]. The symmetric Galerkin method produces, as the name implies, a symmetric linear system which is useful for coupling BEM with FEM. The Galerkin method has drawn much attention and has been studied for a wide range of engineering fields.

The method tries to solve a system of equations of the form

$$\int_S \phi(x) dS(x) = \int_S \psi(x) \int_S G(x,y)\psi(y)\rho(y) dS(y) dS(x). \tag{5.7}$$

5.7 Indirect and direct BEM

The indirect approach tries to solve for unknown physical quantities or loads. In the case of linear elasticity these quantities are FF or displacement discontinuities. These loads are related to the constraints prescribed on the boundary surface using singular solutions and then solved for.

The unknown loads f are expressed in terms of the boundary conditions using the displacement kernel (3.11)

$$\int U_{ij}(x,y) f^i = u_j(Q), \forall Q \in S^u, \tag{5.8}$$

and traction kernel (3.33)

$$\int T_i^j(x,y) f^i = -t^j(Q), \forall Q \in S^t, \tag{5.9}$$

respectively and can then be used to calculate the remaining boundary parameters which was not specified. This approach is called the indirect method because it uses the indirectly determined loads to express the unspecified parameters on the boundary.

The direct BEM is based on fundamental integral theorems used to eliminate the intermediate step and express the boundary parameters directly in terms of the unspecified

parameters. Betti's reciprocal identity (4.2) is used to obtain the Somigliana identity (4.3) for elastostatic problems and given by

$$C_{ij}(P)u_j(P) = \int_S [U_{ij}(Q,P)t_j(Q) - T_{ij}(Q,P)u_j(Q)]dS(P), \quad \forall P \in S, \tag{5.10}$$

in which the u_j and t_j is the displacement and traction fields, respectively. It should be noted that the equation is written in terms of the source point P in stead of the field point Q as is the case for the indirect BEM method. The unknown displacement or traction field which is being solved for is the actual physical quantity, which is also different than the indirect method. The prescribed boundary condition is substituted into equation (5.10) and used to solve for the unknown quantity. If the boundary is smooth the free term coefficient C_{ij} which contains a solid-angle integral is given by $C_{ij} = (1/2)\delta_{ij}$, [28].

5.8 Fictitious Force Method

The fictitious force method (FFM) is also known as the force discontinuity method (FDM). The unknown quantity solved for is a distribution of forces which when applied to the boundary will balance the prescribed internal and external forces and produce the required deformation. This method is particularly suitable for treating volumetric excavation, but is also known for being numerically unstable for tabular excavations and fractures.

The FFM produces numerical instabilities, because the effect of elements placed along the top side of a crack surface is indistinguishable from the effect of elements placed along the bottom side of the crack surface, even if the two surfaces are separated by some small distance [29]. From a numerical point of view, it means that the influence matrix contains rows that are practically the same, which results in a badly conditioned influence matrix. Each crack in the system would have both a hanging and footwall element and would require an equation in the linear system which implies two equations per crack element. The effect of the surrounding elements on the paired elements is almost identical relatively small distances away which causes this numerical instability and gives rise to the bad condition number.

5.9 Displacement Discontinuity Method

The displacement discontinuity method (DDM) has become the mining industry's standard method for treating tabular excavation. The term crack-like element, also called slit-like openings and cracks by [21], refers to two identical elements separated in the normal direction by distances that are small compared to the scale of the elements themselves.

The DDM assumes that the two surfaces of a crack effectively coincides and models the discontinuity of the displacement between the two surfaces.

The FFM breaks down for these kinds of geometry, because the effects of elements placed along the two surfaces of the crack are indistinguishable which produces almost equal equations in the influence matrix. The DDM approximates a relative displacement distribution between the two surfaces over a crack and uses the DD displacement and traction kernels to relate the discretized distribution to the prescribed boundary conditions.

Ore seams in the South African mines, for instance, are very thin and it is common to have stoping widths of one meter. The total excavated regions are also generally very large, up to a kilometer in some of the larger mines, with pillars scattered throughout. Therefore, these kinds of structures can be well modelled by such crack like elements. Other structures that can also be modelled similarly would be faults and dykes.

5.10 Summary

In this chapter, a general overview of the application of the BEM to an integral equation (5.1) was given. The Treffz method was briefly discussed, because it is very similar to the BEM, but mainly because it is sometimes used to speedup the solution of the BEM. The discretization of the boundary was also very briefly discussed. Creating a high quality discretization of a general problem boundary can be difficult. This is a very wide field and an important aspect of the BEM. Without a proper discretization the method can not function and bad discretizations can lead to numerical instabilities. Many different methods can be used to solve integral equations, like collocation and Galerkin. These methods both have their merits, but in recent days more papers seem to use the Galerkin method. Variants of the BEM for elasticity include the indirect and direct method, which differ in the form of the equations and the physical quantities they solve for. The FFM is an indirect method, because the unknown quantity is a FF, but the DDM has roots in both direct and indirect BEM.

Chapter 6

Kernel Integration

6.1 Introduction

It is important to accurately and efficiently perform integration of the displacement and traction kernels for FF and DD over the surface elements of the problem domain. Strategies to compute kernels vary depending on the proximity of the result point and surface element over which the integration is performed. If the field point coincides with or is near the surface element the effect of the singularity might be strong and special techniques have to be employed to accurately perform the integration. Generally the effect of the singularity becomes easier to treat further away from the surface element.

6.2 Numerical Integration

Integrating over an arbitrary element can be difficult. It is possible to represent any flat polygonal shape with triangles.

There exist many numerical integration schemes, Netwon-Cotes, Gauss, Clenshaw-Curtis etc. The Gauss quadrature scheme has become popular, because it only requires n function evaluations to integrate polynomials of order $2n-1$ exactly, whereas Clenshaw-Curtis and Newton-Cotes methods at best integrate polynomials of the order n exactly. The Newton-Cotes methods use equally spaced knots which simplifies hand calculations. It has been pointed out in [26] that using equally spaced points over quadrilaterals or triangles near the singularity can lead to disaster.

It is simple to create an efficient adaptive integration algorithm which reuses previous function evaluations for equally spaced knots, and therefore it is sometimes preferred. The adaptive integration only needs to be calculated at new knots which are placed in between existing knots. The result of the integral for the two integration orders can then be compared to get an error estimate which can then be used as a stopping criterion. For Gauss quadrature increasing the integration order is not as profitable, since knots are

not reused in successive refinements of the integration order. Other methods which reuse function evaluations from Gauss quadrature to provide error estimates do exist.

The Gauss-Kronrod method estimates the error by inserting new points between the knots of a Gauss-Legendre result to calculate another estimate of the integral. If the estimate is not yet good enough then the Gauss-Patterson iterative method can be used to modify the Gauss-Kronrod method by inserting additional points and computing a new error estimate using revised weights. If the error is still not acceptable then another set of knots can be added until the error is acceptable. Instead of adding new knots it is also possible to drop knots to obtain another estimate of the integral. The Gauss-Bond method starts with the Gaussian quadrature and then for an odd integration order drops the middle knot, otherwise it drops the middle pair and then recalculates the weights to obtain the Bond estimate of the integral, [30; 31].

6.2.1 Numerical integration in one variable

A single variable function $f(x)$ can be integrated over the interval $[a, b]$ using numerical quadrature as

$$\begin{aligned} I &= \int_a^b f(x)dx && (6.1)\\ &= \int_{-1}^{1} f(x(\xi))J(x,\xi)d\xi && (6.2)\\ &= \sum_{k=1}^{n} w_k f(x(\xi_k))J(x,\xi_k) + E && (6.3) \end{aligned}$$

where

$$x(\xi) = \frac{1-\xi}{2}a + \frac{1+\xi}{2}b \quad (6.4)$$

maps the coordinate ξ to the interval $[a, b]$,

$$\begin{aligned} J(x,\xi) &= \frac{\partial x(\xi)}{\partial \xi}\\ &= \frac{1}{2}(b-a) && (6.5) \end{aligned}$$

is the Jacobian of the coordinate transformation, w_k are the weights and ξ_k the knots of the quadrature scheme and E the error.

6.2.2 Numerical integration in two variables

The one dimensional integration discussed in the previous section can be extended to surface integration using a tensor product over the two coordinates.

The surface integrals can then be written as a summation over the knots and weights of the quadrature scheme

$$\begin{aligned} I &= \int_S f(x)dS(x) && (6.6)\\ &= \int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x_1,x_2)dx_2dx_1 && (6.7)\\ &= \int_{-1}^{1}\int_{-1}^{1} f(x_1(\xi_1,\xi_2))J(x,\xi)d\xi_2d\xi_1 && (6.8)\\ &= \sum_{k_1=1}^{n_1}\sum_{k_2=1}^{n_2} w_{k_1}w_{k_2}f(x_1(\xi_1,\xi_2),x_2(\xi_1,\xi_2))J(x,\xi)+E_1+E_2. && (6.9) \end{aligned}$$

If the field point is not located on the element then an adaptive technique using approximation formulas for the Gaussian error terms can be employed [16]. The order m_i of the Gaussian quadrature required to obtain a prescribed accuracy is given by [16, p. 44]

$$m_i = p'[\frac{-1}{10}\ln(\frac{e_i}{2})]\left[\left(\frac{8L_i}{3R}\right)^{\frac{3}{4}}+1\right], \tag{6.10}$$

where $p' = \sqrt{\frac{2}{3}p+\frac{2}{5}}$ and p is the order of the singularity, $e_i = \frac{E_i}{I}$ is the relative error and L_i is the length of the element in the i-th direction.

The formula can be inverted for the element length to get another formula of interest which gives an estimate for length to use to reach the prescribed accuracy and is given by

$$L_i\frac{3}{8}R\left(\frac{-10m_i}{p'\ln(\frac{e_i}{2})}\right)^{\frac{4}{3}}. \tag{6.11}$$

This formula can be used to adaptively subdivide the element and therefore to achieve an efficient and accurate integration scheme.

6.3 Continuation Approach

The continuation approach, discussed in depth in [32; 33; 34; 35] and only briefly outlined here was chosen as the basis of integration, because it is a relatively simple and a well explained framework for computing a large class of singular and near-singular integrals which covers the integrals obtained from the FF and DD kernels. The singular integrals are seen as mere continuations of nonsingular integrals. The singularity of the integrand is placed outside the integration domain and then the limit is taken, allowing the singularity to approach the intergation domain.

The surface element integrals are of the form

$$I(q) = \int_\Omega G(q-p)\psi(p)d\Omega(p) \tag{6.12}$$

where Ω is an n-dimensional finite surface in $\Re^{n+1}$. In three dimensions the surface integrals therefore only have dimensionality $n = 2$. The boundary of Ω is piecewise continuously differentiable, because triangles and quadrilaterals are the basic elements used in the boundary discretization, but the continuation is not limited to these basic elements. The FF and DD kernels will be used to model the physical problem and therefore these kernels belong to C^{inf} for $p \neq q$ and is infinite when these points coincide as required by [35]. The loading function $\psi(p)$ is used to interpolate the unknown density over the element and will be given by a piecewise polynomial in the parametric coordinates of the element.

Singular integrals have to be evaluated when the field point q is inside the integration domain. This is needed when calculating self-influence integrals, which are required because it is not always possible to use rigid body motion to compute the self-influence integrals indirectly. These integrals have to be treated specially, because they can not be evaluated using regular numerical quadrature. Therefore a proper interpretation of these integrals is required, since the integrand is infinite on the surface of the element.

When the field point q is close to the domain of integration, which implies that the integration surfaces are near the singularity, ordinary numerical quadrature might fail. These kinds of integrals are termed near-singular integrals. Their evaluation can be avoided by subdividing the integration domain, which effectively increases the relative distance to the singularity as a function of the integration domain's dimensions. The continuation approach provides a unified framework wherein it is possible to integrate these near-singular integrals more effectively. Near-singular integrals might also be encountered when calculating result points close to the boundary or when variable discretization sizes or finely graded meshes are used, where large boundary elements are close to much smaller elements.

Weakly singular integrals, strongly singular integrals and hypersingular integrals are all encountered in elastostatics. The displacement kernel for FF produces a weakly singular integral since it contains a $\frac{1}{r}$ pole. Transforming the integral to polar coordinates it is possible to eliminate the pole and integrate using ordinary Gauss quadrature. The traction kernel for FF and the displacement kernel for DD both produce strongly singular integrals since they contain a $\frac{1}{r^2}$ pole. These integrals have to be interpreted in the Cauchy principal value (CPV) sense. This is done by excluding a small symmetric neighbourhood of the integration domain around the singularity and letting it vanish by taking the limit to produce the CPV of the integral. The traction kernel for DD produces an integral with an even higher singularity since it contains a $\frac{1}{r^3}$ pole, these integrals are called hypersingular integrals and should be treated in a finite part sense introduced by Hadamard, [36].

An alternative interpretation of singular integrals is to view them as continuations of nonsingular ones. The singularity is moved outside the domain of integration and then

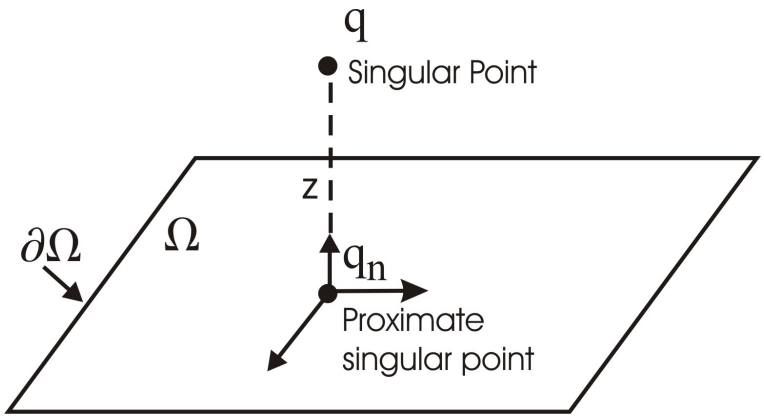


Figure 6.1: *Flat integration domain.*

the limit taken as the singularity is moved towards the integration domain. Integrals interpreted in this sense will be called continuation integrals, [35].

6.3.1 Continuation Theory

The singular integrals encountered in the BEM can be reduced to the following general form

$$I(\hat{z}) \quad = \quad \int_{\Omega} f_{\beta}(\mathbf{x}, \hat{z}) dV \tag{6.13}$$

where the boundary $d\Omega$ is piecewise continuously differentiable, $f_{\beta} : \Re^{n+1} \rightarrow \Re$ is a homogeneous function of degree β, $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and the volume element is given by $dV = dx_1 dx_2 \cdots dx_n$. The integration domain is a star convex n-flat polygon lying in the ambient space R^{n+1}, [32; 33].

In [33], the singular point is defined as a point in the ambient space where the integrand becomes singular. This would be the field point in the context of the indirect BEM and the source point in the direct BEM. The proximate singular point is defined as the point on the integration domain which is closest to the singular point, or the projection of the singular point on the element hyper-plane. In Figure 6.1, q is the singular and q_n is the proximate singular point. The coordinate system has been chosen with the origin on the element and oriented such that any point p on the element is given by the coordinate $p = (\mathbf{x}, 0)$, therefore $q = (\mathbf{x}, z)$ and $q_n = (\mathbf{x}, 0)$.

The homogeneity of the integrand is the basis on which the continuation theory rests. A homogeneous function of degree β satisfies the equation

$$f(\lambda \mathbf{x}, \lambda \hat{z}) \quad = \quad \lambda^{\beta} f(\mathbf{x}, \hat{z}). \tag{6.14}$$

This property can also be expressed by the Euler condition which is a differential equation

with respect to the formal derivatives $\frac{\partial f_\beta}{\partial x_i}$, therefore

$$\beta f(\mathbf{x}, \hat{z}) = \sum_i^n x_i \frac{\partial f_\beta}{\partial x_i} + \hat{z}\frac{\partial f_\beta}{\partial \hat{z}}. \tag{6.15}$$

This leads to the basic continuation linear differential equation for $I(\hat{z})$

$$\hat{z}\frac{\partial}{\partial \hat{z}} I(\hat{z}) - \alpha I(z) = -\int_{\partial\Omega} f(\mathbf{x}, \hat{z})\mathbf{x} \cdot d\mathbf{s} \tag{6.16}$$

where $\alpha = \beta + n$ and $d\mathbf{s}$ is the directed line segment element on the integration boundary and for elements in two-dimensional Euclidean space the term $\mathbf{x} \cdot d\mathbf{s} = x_1 dx_2 - x_2 dx_1$.

The prototype for integrands can be expressed by

$$f(\mathbf{x}, \hat{z}) = \frac{x_1^{l_1} x_2^{l_2} \cdots x_n^{l_n} \hat{z}^m}{r^k} \tag{6.17}$$

where r is the Euclidean distance function and the exponents $l_1, \cdots, l_n, m$ and k are all nonnegative integers and satisfies

$$\beta = \sum_i^n l_i + m - k. \tag{6.18}$$

The linear differential equation (6.16) can be solved as an initial value problem with initial condition $I(z_0)$ at z_0, therefore

$$I(\hat{z}) = -\hat{z}^\alpha \int_{z_0}^{\hat{z}} \left(\frac{1}{\eta^{\alpha+1}} \int_{\partial\Omega} f(\mathbf{x}, \eta)\mathbf{x} \cdot d\mathbf{s} \right) d\eta + \frac{\hat{z}^\alpha}{z_0^\alpha} I(z_0). \tag{6.19}$$

The integral $I(\hat{z})$ is independent of the initial condition z_0 and can be chosen arbitrarily far away from the integration domain. In [35], it is shown that if the initial condition is set at $z_0 = \pm\infty$, then for homogeneous integrands

$$\lim_{z_0 \to \pm\infty} \frac{I(z_0)}{z_0^\alpha} = 0 \tag{6.20}$$

for all α. Therefore the last term in equation (6.19) disappears. The order of integration can be interchanged to obtain the boundary-only continuation formula

$$I(\hat{z}) = \hat{z}^\alpha \int_{\partial\Omega} (F_{\pm\infty}(\mathbf{x}) - F(\mathbf{x}, z))\mathbf{x} \cdot ds. \tag{6.21}$$

where

$$F(\mathbf{x}, \hat{z}) = \int \frac{1}{\hat{z}^{\alpha+1}} f(\mathbf{x}, \hat{z}) d\hat{z} \tag{6.22}$$

and

$$F_{\pm\infty}(\mathbf{x}) = \lim_{\hat{z} \to \pm\infty} F(\mathbf{x}, \hat{z}). \tag{6.23}$$

$F(\mathbf{x}, \hat{z})$ is called the primitive boundary function (PBF). The surface integration can be converted into a contour integration using the PBF. These PBFs can be obtained analytically using a symbolic manipulation tool such as Mathematica.

6.3.2 Adaptive contour integration

The primitive boundary functions are computed using line integrals along each of the polygon's edges. The line integral

$$I_i(\hat{z}) = \int_{\partial\Omega_i} f(\mathbf{x}, \hat{z})\mathbf{x} \cdot ds \tag{6.24}$$

is computed for each edge $\partial\Omega_i$ separately. The total integral for the element is then given by the sum of

$$I(\hat{z}) = \sum_i I_i(\hat{z}) \tag{6.25}$$

the integrals for each of the edges.

An adaptive integration scheme is used which recursively subdivides the domain of integration. The integral over the interval before subdivision is computed. Then an integral over each of the subdivided intervals is computed. The sum of the integrals of the subdivided intervals should be equal to the integral before subdivision. In general numerical Gauss quadrature does not exactly integrate an arbitrary function, therefore this assumption does not necessarily hold.

Let $[a, b]$ be the domain of integration in the line segment's intrinsic coordinates. The relative integration error can then be expressed as

$$\begin{aligned} I_0 &= I([a, b]) \\ I_1 &= I([a, c]) + I([c, b]) \\ E &= \frac{|I_1 - I_0|}{\max\{|I_0|, |I_1|\})} \end{aligned} \tag{6.26}$$

where $c = \frac{a+b}{2}$ is the center of $[a, b]$. If the error is less than the prescribed relative tolerance or the absolute value of I_1 is less than the prescribed absolute tolerance then the integral over the interval $[a, b]$ is taken as I_1, since it should be a better approximation of the integral, otherwise the recursion continues on each of the subdivided intervals $[a, c]$ and $[c, b]$. It should be noted that the denominator in equation (6.26) could be zero, but in that case the absolute value test would be enough to stop the recursion.

In practice a line integral for each term of the kernel function has to be evaluated. It is better to compute them simultaneously since they share computations. Therefore the

relative error can be expressed in terms of all the terms as

$$\begin{aligned} I_0^i &= I^i([a,b]) \\ I_1^i &= I^i([a,c]) + I^i([c,b]) \\ E_{abs} &= \max_i |I_1^i - I_0^i| \qquad (6.27) \\ E_{rms} &= \sqrt{\sum_i (\max\{|I_0^i|, |I_1^i|\})^2} \qquad (6.28) \\ E &= \frac{E_{abs}}{E_{rms}} \qquad (6.29) \end{aligned}$$

where the index i indicates the term which was integrated, E_{abs} is the maximum absolute deviation and E_{rms} the root mean square of the maximum term between the two integrals I_0^i and I_1^i. Therefore the recursion would return if the new relative error is less than the prescribed relative tolerance or the maximum absolute value of all the terms $\max_i |I_1^i|$ is less than the prescribed absolute tolerance and the integral over the interval $[a,b]$ taken as I_1. Lastly to stop the recursion from subdividing too far, a maximum recursion depth is used to ensure that the algorithm is robust.

6.4 Summary

The evaluation of the influence of a surface element requires integration of the kernel function. If the singularity lies near or within the plane of the element, then ordinary quadrature can not always accurately compute these integrals. Many integration schemes exist, but the continuation approach is attractive, because it is a systematic approach applicable to all kernels in elastostatics. The scheme converts the surface integral into a contour integral, which can be integrated using a simple adaptive contour integration. It is possible to apply the continuation approach to curved elements [35] and collocating the unknowns at the element vertices [25; 37]. The continuation theory discussed only covered the basic near-singular integration formula. This can be used to compute influence integrals which are close to the boundary or surface of an integral by slightly moving the influence point away. It should be noted that the continuation theory also covers the proper computation of influence integrals on the element surface.

Chapter 7

Assembly

7.1 Introduction

The boundary element method leads to a system of linear equations

$$A\mathbf{x} = \mathbf{b} \tag{7.1}$$

where $\mathbf{x}$ are the unknowns, $\mathbf{b}$ are the prescribed boundary conditions and A is a linear operator expressed here as a matrix.

The naïve assembly of this system would express A as a matrix of dimension $n \times n$, where n is the number of unknowns and related to the number of boundary elements used to describe the problem boundary. If few boundary elements are used then this would not pose to be a big problem and a direct or iterative solver could be used to find x. Unfortunately Mother Nature does not work this way and the problem sizes considered in mining applications can grow to be very big, especially if the model was built to represent the whole mining operation. The direct construction of A requires $O(n^2)$ kernel evaluations, has storage requirement of size $O(n^2)$ and requires $O(n^2)$ operations for a matrix-vector multiplication. The aim of this chapter is to develop a better representation for A and algorithms to construct the representation and for performing matrix-vector multiplication.

There currently exist a couple of different representations which have gained popularity over the years. They include, but are not limited to:

- the panel clustering algorithm,
- fast multipole method (FMM),
- kernel independent FMM,
- precorrected-FFT,
- wavelet based methods and

- algebraic techniques (hierarchical matrices).

The panel clustering algorithm has been around for a long time. It constructs a tree-like representation of the boundary elements and then links clusters (or nodes) with in the tree to other clusters on the same level in the tree. An admissibility constraint, which determines whether the effect of one cluster on another may be ignored, is used to determine whether two clusters can be linked. This constraint determines whether the clusters can be treated as a simple boundary element or whether it is necessary to subdivide the clusters to meet the specified error tolerance. The method requires $O(nlog^{d+2}n)$ operations to construct the representation, needs $O(nlog^{d}n)$ of storage and a matrix-vector multiplication has an operation count of $O(nlog^{d+1}n)$, where d is the dimensionality of the problem being solved, [6].

The fast multipole method is another popular method and many variants exist. The mathematics used in constructing such a representation is quite involved and dependent on the kernel. The FMM's aim is to express the potentials or densities on the boundary elements using multipole expansions and local expansions at places far away from these elements. The method then applies translations between these expansions to efficiently evaluate the densities. The overall complexity is $O(n)$, but depends on the size of the expansions used and error tolerances. Even though the complexity is linear, it might still be more efficient to use a scheme with a slightly higher complexity, because it might out perform FMM for even very large cases. Constructing a FMM representation might only be profitable if many resolves are required.

Much work has been done to bypass the kernel dependence of the FMM. [3, Ying] replaced the expansions and translations in the FMM by an equivalent density representation which only requires kernel evaluations. The idea is akin to the Treffz method [18, Beer, p. 84] and used to isolate sections by solving local interior and exterior problems. The method has similar complexity to the FMM.

The precorrected-FFT method [4, Ding] uses a uniform grid to represent the long-range interactions. Short-range interactions are then treated specially and therefore the term "precorrected" in the name of the method. It is also kernel independent, but the reliance on a uniform grid is a big disadvantage. The methods complexity is due to the fast Fourier transform which is of $O(nlogn)$.

The wavelet based methods, which are not discussed here, has complexity of $O(n)$, [5].

The algebraic techniques are almost purely focused on compressing A, while possibly using some geometric information in preparing the problem description before and during assembly of the representation. The basis of these techniques is that for many problems involving coefficient matrices it might be possible to express blocks within the matrix by low-rank approximations. These low-rank approximations were termed pseudoskeletons in [38]. The aim of these methods is to find the optimal covering of pseudoskeletons for which

the storage requirement would be optimal. IES^3 [39; 7; 2; 40], solvers for sequentially semi-separable matrices [8; 41; 42] and hierarchical matrices [43; 44; 45; 46; 47] are all examples of algebraic techniques.

7.2 Ordering

The kernels used in the modelling all exhibit similar spatial behaviour, namely that they have a singularity at the collocation point and then decrease as the distance from the collocation point increases. The concept of first constructing a tree over the element list is core to hierarchical methods and is similar to multipole, panel clustering and wavelet-based approaches, [7; 48; 41; 9]. With each element there is associated, a spatial coordinate as well as a dimension, such as the area covered by the element. With some limited knowledge of the geometry a better tree can be constructed and some heuristics can be used to optimize this construction to give better compression results. Nodes and leaves in the tree are clusters of elements. If we associate a bounding shape with each of these clusters we can use Saint-Venant's principal as guide to develop some kind of algorithm which will give us the best compression.

Saint-Venant's Principal states [13, p. 180]:

> An equilibrated system of external forces applied to an elastic body, all of the points of application lying within a given sphere, produces deformations of negligible magnitude at distances from the sphere which are sufficiently large compared to its radius.

Physically this means that elements which are spatially close together influence each other more than far away elements. The input elements might not have been labelled to take into account this spatial relationship. The compression algorithm does not require that the kernel values go to zero, but only that groups separated by a reasonable distance from each other exhibit correlated behaviour which can be exploited to approximate their influences. The aim of ordering will be to increase the compressibility of the influence matrix, for the simple case of a one dimensional strip of elements, the natural ordering is usually most optimal, [7]. In two or three dimensions it might not be possible to achieve optimal ordering given an algorithm of reasonable complexity. A simple $O(n \log n)$ ordering scheme proposed in [7] is used and later adapted to generate additional spatial information [9], it is similar to the binary space partitioning, KD-tree and oct-tree algorithms which are very common in computer graphics.

Let T be the set of elements used to discretize the problem boundary. Find the closest fitting bounding box and transform the elements into the local coordinate system. Next apply the following recursion on T:

1. If T is empty or contains one element: stop the recursion.
2. Find the axis of maximal extent.
3. Order all the elements in T along the axis.
4. Split the list such that $T^{'} \subset T$, $T^{''} \subset T$, $T^{'} \cap T^{''} = \emptyset$ and $T^{'} \cup T^{''} = T$.
5. Apply the recursion on both $T^{'}$ and $T^{''}$.

The hierarchical structure is retained and for each level a bounding sphere is constructed within which all sub elements are fully contained. It is not necessary that bounding spheres between subsets be disjoint and in general lots will overlap. Orderings produced by this algorithm gives good results for tabular mining layouts that have been tested and it allows for other heuristics that uses intersection and separation distance between the bounding spheres to control error tolerances.

7.3 Algebraic Method

7.3.1 Review of the IES^3 algorithm

IES^3 [7] is an algorithm that was first developed to exploit the algebraic structure of matrices that arise from the linearization of integral equations. It is a purely algebraic method, does not require any knowledge of the model geometry and is kernel-independent. Compression is controlled by an error tolerance which is used when constructing the low rank decompositions.

The algorithm exploits the fact that there exists large parts in the matrix which is numerically low rank just like the mosaic methods [48]. In essense a hierarchical view is constructed for the matrix, which is analyzed to find where low rank decomposition can adequately approximate the blocks and save on storage and memory usage.

The representation is constructed by recursively splitting the matrix into smaller sub-matrices until it is not possible to split further or a prescribed minimum dimension is reached. These sub-matrices are then decomposed using some kind of rank revealing decomposition method. Then a bottom-up scheme is used to merge each of these sub-matrices. The aim is to determine whether a sub-matrix can be expressed more efficiently by a low rank decomposition which approximates the sub-matrix to a given tolerance. If the storage requirement of the low rank decomposition is less than the storage requirement of the sub-matrix then the decomposition is returned and the recursion continues upward. The most time consuming parts in this recursion are the computation of the kernels and construction of the decomposition. In theory the decomposition could be obtained

by applying a singular value decomposition and then truncating the representation by thresholding the singular values.

The algorithm promises storage complexity of $O(nlogn)$ [7] and with clever interpolation techniques, $O(nlogn)$ construction time [7; 2]. Another advantage is that matrix-vector and matrix-matrix multiplication in the IES^3 structure can be performed with the same $O(nlogn)$ complexity.

```
if minimum dimension of A is greater or equal to a prescribed number then
  split A into sub-matrices
  recurse over the sub-matrices
  if sub-matrices are all of low rank then
    merge sub-matrices
    if merged decomposition is more storage efficient than sub-matrices then
      return merged decomposition
    else
      store sub-matrices
      return nothing
    end
  else
    store returned sub-matrices
    return nothing
  end
else
  create rank revealing decomposition A
  if decomposition is more storage efficient then
    return decomposition
  else
    store A
    return nothing
  end
end
```

Figure 7.1: *IES^3 pseudo-code.*

Figure 7.1 shows the pseudo-code for IES^3. It is a recursive "divide and conquer" algorithm that starts of with the full coefficient matrix and subdivides A until the dimension of A is smaller than some prescribed number. The matrix might be split into sub-matrices

as follows

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}. \tag{7.2}$$

Each of these matrices are then processed recursively and effectively become the full coefficient matrix when processed. The divide and conquer method has the advantage of memory locality. When constructing the decomposition, the algorithm might be clever and know when the decomposition will not be storage efficient enough. Instead of continuing with the decomposition and wasting time, it could continue to construct the full matrix instead using the kernel evaluations that were already computed.

If the return value is a full matrix, it is stored, otherwise the decomposition would be returned as normal. When a dense matrix or group of decompositions are stored, then *nothing* is returned. Here *nothing* has meaning and is used in the algorithm to find out the state of particular sub-matrices. A *nothing* is associated with a sub-matrix and would indicate that the algorithm has finished with that sub-matrix.

The decomposition of a sub-matrix is given by

$$A_i^{n\times m} = U_i^{n\times r}(V_i^T)^{r\times m} \tag{7.3}$$

and has a rank r associated with it. The storage requirement for such a decomposition would be $r*(n+m)$ elements, while the storage requirement of the dense matrix would be $n*m$. The decomposition's rank would then have to obey

$$r < \frac{1}{2}\min\{m,n\}. \tag{7.4}$$

When the recursion completes processing all the sub-matrices it goes on to evaluate the returned sub-matrices. If not all sub-matrices are decompositions then the sub-matrices which were decompositions are stored and *nothing* is returned. This could represent the case where one of the sub-matrices was dense. In any event it will always happen if the list of returned sub-matrices contained a *nothing*. If the list contained only decompositions then the recursion calls a merging procedure. The merging procedure tries to further decompose the list of decompositions and construct a more efficient decomposition. If the new decomposition is more storage efficient then it is returned, otherwise the decompositions in the list is stored and a *nothing* is returned. This continues until recursion returns to where it started. In the odd event that the algorithm returns a decomposition it will have to be stored.

The IES^3 algorithm can be broken up into four parts:

- the hierarchical matrix traversal which basically controls when to subdivide the matrix,
- generation of low rank decompositions,

- merging low rank decompositions and
- storing the dense matrices and decompositions.

7.3.2 Compression

A compression technique is needed to optimize the amount of storage required to represent the linear system. The aim for the compression algorithm is also to transform the problem into a representation which facilitates fast matrix-vector multiplication. The algorithm developed is based on the algorithm described in IES^3 [39; 7] and used to construct the matrix. The compression algorithm was improved by including ideas from [8; 41; 43; 44] and others, which are all similar, but somewhat more advanced than IES^3 in their theoretical treatment.

The development of the algorithm was done in phases. First the focus was placed primarily on constructing the compressed representation of the matrix, disregarding the time it takes to evaluate the kernels. The resulting algorithm followed the IES^3 algorithm closely, the only differences being in the stopping criterion for splitting and the merging scheme.

A prescribed small number, the size of the smallest sub-matrix, was still present, but only for completeness. What this small number should be set to was not described in [7] and varying this number did effect IES^3's performance. Therefore the new algorithm was developed with this prescribed number set to two kernel elements. This means that the average block the algorithm would try to decompose would be a matrix of dimension 2×2.

The other difference was that the IES^3 merging algorithm was replaced by a merging algorithm based on Lanczos bidiagonalization. The Lanczos bidiagonalization process described in [49] requires only matrix vector products and exploits the structure of the already compressed sub-matrices, which in turn speeds up the process. An additional truncated SVD of the bidiagonal matrix was also done to further reduce the rank. The reason for using Lanczos bidiagonalization was to improve the compression, since it was noted in [7] that the modified Gram-Schmidt merging sometimes overestimated the rank of the decomposition. Where the IES^3 algorithm would stop as soon as one or more of the sub-matrices were not a decomposition, the new algorithm would try and continue until the depth of the sub-matrix representation exceeded a prescribed number.

The second stage of development focused on faster assembly of the representation and to reduce the number of kernel evaluations. [7] gave promise of an alternative faster interpolation scheme that did exactly that, but their description was too vague and only hinted at how this might be achieved. After much searching, two different algorithms looked promising, a method called Dual-MGS [2] and it's variant IE-QR [40] and another

called Adaptive Cross Approximation (ACA) [50] and it's variant ACA+ [45]. Even though ACA promises faster compression and does not require solving a least squares system, Dual-MGS's performance does not lag far behind and gives a somewhat better approximation.

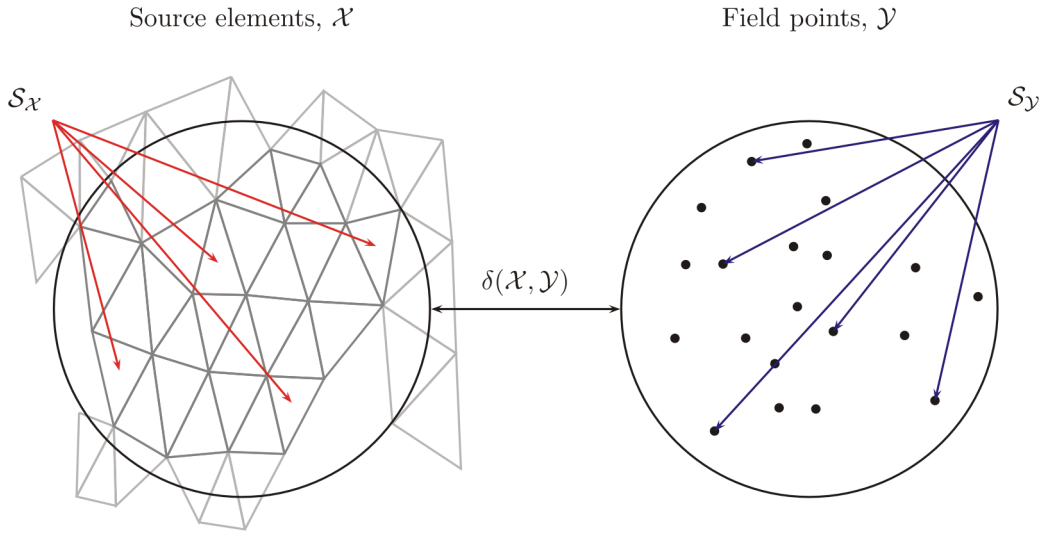


Figure 7.2: *Sampling the source elements and field points.*

Fast assembly can only be achieved if less kernel evaluations are performed. The Dual-MGS and ACA methods described later on does exactly this. The only problem is that if the traversal used in the IES^3 process is used, then the power of these methods would not be fully utilized. This is because both these methods construct a decomposition based on the assumption that the influence matrix between the set of source elements and field points are of low rank and most importantly, that these sets are not to small. If the prescribed number for the IES^3 method is too small then the number of evaluations which are saved might not be optimal.

Figure 7.2 shows two sets of source elements $\mathcal{X}$ and field points $\mathcal{Y}$. The aim is to build an approximation of the sub-matrix $\mathcal{X} \times \mathcal{Y}$ of interactions between the sets by using samples from $S_\mathcal{X} \subset \mathcal{X}$ and $S_\mathcal{Y} \subset \mathcal{Y}$. These fast incomplete decomposition algorithms use different selection strategies to grow their sample spaces and construct their decompositions. There exists a relationship between the spatial separation $\delta(\mathcal{X},\mathcal{Y})$ and spread of the source element and field point sets which determines how big these samples should be to achieve a given accuracy. These algorithms adaptively discover this and therefore are not reliant on defining such a relation.

7.3.3 Traversal

The resulting hierarchical structure can be used efficiently to perform matrix-vector multiplication. Let A be a general $m \times n$ matrix, then

$$\mathbf{b} = A\mathbf{x} \tag{7.5}$$

with $\mathbf{x}$ a vector of length n and $\mathbf{b}$ a vector of length m. Suppose A has the form given in equation (7.2), then

$$\begin{pmatrix} \mathbf{b}_1^{a\times 1} \\ \mathbf{b}_2^{c\times 1} \end{pmatrix} = \begin{pmatrix} A_1^{a\times b} & A_2^{a\times d} \\ A_3^{c\times b} & A_4^{c\times d} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1^{b\times 1} \\ \mathbf{x}_2^{d\times 1} \end{pmatrix} \tag{7.6}$$

where $a + c = m$ and $b + d = m$. Therefore the vector $\mathbf{x}$ is multiplied in such that

$$\mathbf{b}_1 = A_1\mathbf{x}_1 + A_2\mathbf{x}_2, \tag{7.7}$$

$$\mathbf{b}_2 = A_3\mathbf{x}_1 + A_4\mathbf{x}_2. \tag{7.8}$$

Matrix-vector multiplication of the larger matrix A can now be written as the combination of the result of the four smaller matrix-vector multiplications $A_1\mathbf{x}_1$, $A_2\mathbf{x}_2$, $A_3\mathbf{x}_1$ and $A_1\mathbf{x}_2$. If the sub-matrix $A_i = U_iV_i^T$ is in pseudoskeleton form then first $\mathbf{y} = V_i^T\mathbf{x}_j$ is performed followed by $\mathbf{b}_k = U_i\mathbf{y}$.

The hierarchical tree structure is traversed in a bottom-up fashion, multiplying sub-matrices as they are encountered at the leaves and combining the results as the traversal bubbles up.

7.3.4 Decompositions

The basis of our method requires some technique with which a storage efficient representation might be constructed which also facilitates faster matrix-vector multiplication of a matrix. In the algebraic sense, this can be achieved by a low rank decomposition, which has for a given matrix A the form

$$A = UV^T \tag{7.9}$$

with the dimensions of A being $m \times n$, U being $m \times r$ and V being $n \times r$, where r is the rank of the decomposition and

$$r < \frac{1}{2}\min\{m, n\}. \tag{7.10}$$

Given that equation (7.10) holds we can see that the storage of A requires $m*n$ values while for the decomposition UV^T only $r*(m+n)$ values are needed which guarantees

that $r*(m+n) < m*n$ and that the decomposition is more storage efficient. Multiplying a vector with the decomposition is also faster than multiplying it with A, for example

$$\begin{aligned} UV^T\mathbf{x} &= U(V^T\mathbf{x}) \\ &= U\mathbf{y} \\ &= \mathbf{z}. \end{aligned} \tag{7.11}$$

Multipling a vector $\mathbf{x}$ with a matrix A requires $m*n$ operations while multiplying with the decomposition only requires $r*(m+n)$ operations.

Nowhere, except maybe during the construction of the decomposition, does the algorithm require that the matrices U or V be orthonormal or orthogonal. The only constraint is that the decomposition should approximate the original matrix with respect to some error tolerance and a given norm. The matrix norm of preference and used throughout is the Frobenius norm. The Frobenius norm of a matrix is defined by [1] as:

$$\|A\|_F = \left(\sum_j^m \sum_i^n |a_{ij}|^2 \right)^{\frac{1}{2}}. \tag{7.12}$$

Singular Value Decomposition

The singular value decomposition is described by the following theorem [51]:

Theorem 7.3.1 *Any $N \times N$ matrix A can be written as a product of an $N \times N$ orthogonal matrix U, an $N \times N$ diagonal matrix S with positive or zero elements (the singular values) and the transpose of an $N \times N$ orthogonal matrix V, i.e.,*

$$A = U \cdot S \cdot V^T \tag{7.13}$$

where

$$U \cdot U^T = V \cdot V^T = I, \tag{7.14}$$

and I is the $N \times N$ identity matrix.

The singular values $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ of the matrix are usually ordered from largest to smallest. The rank of A is then equal to the number of positive singular values.

The truncated SVD can be obtained from A by applying a threshold on the singular values such that all $\sigma_i < \epsilon\sigma_1$, for a given small positive ϵ, are considered to be zero. The truncated SVD is given by

$$\tilde{A} = \tilde{U} \cdot \tilde{S} \cdot \tilde{V}^T, \tag{7.15}$$

where $\tilde{U}$ and $\tilde{V}$ has dimensions $N \times r$ and $\tilde{S}$ has dimensions $r \times r$, with $r \leq N$ the rank of $\tilde{A}$. $\tilde{A}$ is an approximation of A with the approximation error bounded by ϵ, such that

$$\left\| A - \tilde{A} \right\|_2 < \epsilon. \tag{7.16}$$

The SVD is rank revealing and can be used as a decomposition method in the compression algorithm. It is easy to get the SVDs into the desired form of equation (7.9). This can be done by either multiplying out the diagonal matrix containing the singular values into column basis U or row basis V, because equation (7.9) does not require either of the two matrices to be orthonormal or even orthogonal to each other. Unfortunately calculating an SVD is very expensive and is considered here only for theoretical completeness and to clarify the mechanism on which the following algorithms are based.

Modified Gram-Schmidt

The modified Gram-Schmidt method is the preferred practical method for computing a basis of orthogonal vectors instead of the standard Gram-Schmidt method, since it is numerically stable and does not suffer as much from loss of orthogonality. Figure 7.3 shows the pseudo-code for the modified Gram-Schmidt method which is used to generate an orthogonal basis from a set of vectors $A = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_r\}$. The resulting factorization is then expressed as

$$A = QR, \tag{7.17}$$

where Q is the orthogonal basis and R is an $r \times r$ upper triangular matrix and called a QR decomposition.

However, in some cases a second reorthogonalization step might well be required to recover orthogonality. Many modified Gram-Schmidt steps will need to be performed during merging in the IES^3 algorithm. If the needed corrections are not performed then the numerical error might grow too large and cause the approximation to deviate too much. It is common to also introduce pivoting into the Gram-Schmidt algorithms as was proposed in [7] for the merging scheme. This is done to bypass premature breakdown of the process which might occur, because of linearly dependent vectors in the set of vectors to be orthogonalized. Figure 7.4 shows such a method which combines the classical Gram-Schmidt with modified Gram-Schmidt and adds pivoting.

In Figure 7.5 the pseudo-code for another variant of the Modified Gram-Schmidt algorithm, based on Arnoldi's procedure is given. This method requires the multiplication of some vector with the matrix A for which we wish to construct the orthogonal basis, instead of constructing the basis directly by using the entries within A. The method constructs an orthogonal basis for the Krylov subspace $\mathcal{K}_m$

$$\mathcal{K}_m = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}. \tag{7.18}$$

The resulting decomposition is then expressed as

$$A = V_m H_m, \tag{7.19}$$

```
define r_11 := ||x_1|| .
if r_11 = 0 then
  stop
else
  q_1 := x_1/r_11
end
for j = 2, ..., r do
 define q̂ := x_j
 for i = 1, ..., j − 1 do
 r_ij := q̂ · q_i
   q̂ := q̂ − r_ij q_i
 end
 compute r_j,j := ||q̂||_2
 if r_j,j = 0 then
    stop
  else
   q_j = q̂_j/r_j,j
  end
end
```

Figure 7.3: *Modified Gram-Schmidt pseudo-code [1, p. 11].*

```
find the index of the column vector u_k with largest norm
keep the norm τ = τ_1 = ||u_k|| to determine when to stop
while τ > ετ_1 do
  chalk up one rank and save the column index
  orthogonalize against the previous set of orthogonal vectors and normalize
  for all the remaining vectors
    orthogonalize against the new orthogonal vector
  for the index k of the remaining vector with largest norm
  compute the norm τ = ||u_k|| of the newly selected vector
end
```

Figure 7.4: *Modified Gram-Schmidt with partial pivoting [2].*

where V_m is the orthogonal basis for $\mathcal{K}_m$ and H_m is the upper triangular Hessenberg matrix.

```
choose a vector v_1 such that ||v_1||_2 = 1
for j = 1, 2, ..., m do
 compute w_j := A v_j
 for i = 1, ..., j do
 h_ij := w_j · v_i
    w_j := w_j − h_ij v_i
 end
 h_{j+1,j} = ||w_j||_2
 if h_{j+1,j} = 0 then
    stop
  end
 v_{j+1} = w_j / h_{j+1,j}
end
```

Figure 7.5: *Arnoldi Modified Gram-Schmidt pseudo-code [1, p. 156].*

Lanczos bidiagonalization

The Lanczos bidiagonalization technique described in [49] tries to efficiently construct a good approximation $\tilde{A}$ of the original matrix A. It provides a recursive formula for the stopping criterion and orthogonality of the left and right Lanczos vectors and discusses different reorthogonalization methods. The approximation is constructed iteratively and therefore meets the requirement of a rank revealing decomposition, even though the computed rank might not be as tight as possible.

Where the merging scheme proposed in [39; 7] requires that the list of sub-matrices all be decompositions, the Lanczos method only requires that matrix-vector multiplication be defined on the sub-matrices. Fast matrix-vector multiplication was the whole aim of the compression algorithm and therefore the bidiagonalization procedure can take advantage of the structure in which the sub-matrices are cast. This extends the applicability of the merging scheme, because it does not require all the sub-matrices to be decompositions. IES^3 required some prescribed value which was used to control the recursion depth or resolution at which we analyzed the matrix. Using bidiagonalization for merging removes this requirement, because now it is possible to recurse until we reach the kernel resolution and merge up from there. In [2] it is mentioned that the Lanczos bidiagonalization was used to perform block-factorization, but no mention was made of using it for merging.

The aim of the method is to construct an approximation $\tilde{A}_k$ of A such that

$$\left\| A - \tilde{A}_k \right\| \leq \epsilon \tag{7.20}$$

where $\epsilon \in \Re^+$ controls the absolute accuracy of the approximation and k is the number of iterations needed to obtain the required accuracy. The approximation $\tilde{A}_k$ is constructed

in the form

$$\tilde{A}_k = U_k B_k V_k^T \tag{7.21}$$

where $U_k = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ is the orthogonal left Lanczos vectors, $V_k = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ is the orthogonal right Lanczos vectors and B_k is a bidiagonal matrix with entries

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix}. \tag{7.22}$$

The procedure is started with some arbitrary vector $\mathbf{u}_1$, such that $\|\mathbf{u}_1\| = 1$. Multiplying this vector with the transpose of A gives the first entry in B_1 as

$$\alpha_1 v_1 = A^T u_1. \tag{7.23}$$

The consecutive approximations $i = 1, 2, \ldots$ are then defined by

$$\beta_{i+1}\mathbf{u}_{i+1} = A\mathbf{v}_i - \alpha_i \mathbf{u}_i, \tag{7.24}$$

$$\alpha_{i+1}\mathbf{v}_{i+1} = A^T\mathbf{u}_{i+1} - \beta_i \mathbf{v}_i. \tag{7.25}$$

A full derivation of the error estimation is done in [49], here we will note that there exists a formula which recursively expresses the error in terms of α_{k+1} and β_{k+1}. Let $w_k = \left\|A - \tilde{A}_k\right\|_F$ be the error in the Frobenius norm after k iterations. Then the error for the next iteration can be expressed as

$$w_{k+1}^2 = w_k^2 - \alpha_{k+1}^2 - \beta_{k+1}^2. \tag{7.26}$$

The loss of orthogonality in either the left or right Lanczos vectors can cause the matrix B_k to become ill-conditioned. An efficient reorthogonalization is needed to monitor the loss of orthogonality.

Two methods are discussed in the paper namely, partial and one-sided reorthogonalization. The partial reorthogonalization method uses the recurrence formulae

$$\begin{aligned} \epsilon_{\text{left}} &= \epsilon_M \sqrt{m} \\ \epsilon_{\text{right}} &= \epsilon_M \sqrt{n} \\ \omega_{ii} &= \delta_{ii} = 1 \\ \omega_{i+1,k} &= (\alpha_k \delta_{i,k} + \beta_k \delta_{i,k-1} - \alpha_i \omega_{i,k} + \epsilon_{\text{left}})/\beta_{i+1} \\ \delta_{i+1,k} &= (\beta_{k+1}\omega_{i+1,k+1} + \alpha_k \omega_{i+1,k} - \beta_{i+1}\delta_{i,k} + \epsilon_{\text{right}})/\alpha_{i+1} \end{aligned} \tag{7.27}$$

where $k = 1, \ldots, i$, $\delta_i = 0$ and ϵ_M is the machine's precision, to determine when loss of orthogonality becomes too large. The algorithm checks that $\max_{1 \le j \le i} \omega_{i+1,j} < \sqrt{\epsilon_M}$, if not

then the last two of left Lanczos vectors are reorthogonalized against the previous vectors and the recurrence for $\omega_{i+1,j}$ reset to ϵ_{left}. Next it checks that $\max_{1 \leq j \leq i} \delta_{i+1,j} < \sqrt{\epsilon_M}$, if not then the last two of right Lanczos vectors are reorthogonalized against the previous vectors and the recurrence for $\delta_{i+1,j}$ reset to ϵ_{right}.

The one-sided reorthogonalization, reorthogonalizes either the left or right Lanczos vectors. From the numerical results it seems that due to the coupled two term recurrence keeping only one side orthogonal controls the orthogonality in the other. If the matrix being approximated is very skinny this result implies that only the smaller of the two Lanczos vectors need to be reothogonalizated.

The resulting bidiagonalization can cheaply be converted into a singular value decomposition, by factorizing the bidiagonal B matrix. The B matrix is small relative to the input matrix A, which we assume is of low rank and therefore applying the SVD on B is not too costly. There are methods which can be used to construct the SVD more cheaply than the conventional methods given a bi-diagonal matrix, but we have not implemented or researched that yet. In general a further reduction of one extra rank is achieved when applying and truncating the SVD.

Dual-MGS

The kernel in the far-field is smooth and it is expected that it should be possible to construct a low rank approximation which is similar to an interpolation of the kernel between the source and target (field points) elements. Kernel evaluations are computationally expensive and if it was possible to build a low rank decomposition by only sampling a few kernels then subquadratic construction time could be achieved. In [52] a proof is given that for the electrostatic problem, the reduction in rank between groups is a function of their separation distance. The same applies for elastostatic problems as can be judged by looking at numerical results.

In [2; 40] a block-QR-factorization algorithm is presented that samples the kernels and does not require all the entries of the sub-matrix to be computed. The method uses a dual-row-column-MGS approach which only require r columns and rows of the original matrix. It is very similar to the pivoted Gram-Schmidt algorithm in Figure 7.4, but selects rows and columns in such a way which does not require the full matrix.

Let A be the $n \times m$ matrix we wish to decompose and let the columns be given by $A = \{u_1, \ldots u_m\}$ and the rows by $A = \{v_1, \ldots v_n\}$. As in the previous algorithms we wish to construct the decomposition $A \approx QR$ with error tolerance given by $\epsilon \in \Re^+$ where $Q = \{q_1, \ldots q_r\}$ is an orthogonal $n \times r$ matrix, R is a $r \times m$ matrix and r the rank of the decomposition.

Figure 7.6 shows the pseudo-code for the Dual-MGS algorithm and the first part is on how to compute Q. The last two lines' description is very compact and shows how

to find the matrix R using a co-location approximation. Only the previously generated row vectors are used as well as the indices that were stored in rows(:). These are used to select only the rows in Q that correspond to the row vectors $V(:, \text{rows}(:)) = \{v_1, \ldots, v_r\}$ and used in the approximation of R. The inverse Q^{-1} can be found by applying a LU-decomposition on $Q(\text{rows}(:), :)$ and then solving for R as in the last line of the pseudo-code. The complexity of the LU-factorization is $O(r^3/3)$ and $O(nr^2)$ solving for R, [2].

```
let r = 1 compute the first row v_r
store the first row index in rows(r)
let p_r = v_r/||v_r||
find the index k of the largest absolute entry in p_r
compute the kth column u_k
save the column index k
let q_r = u_k/||u_k||
keep the norm τ = τ_1 = ||u_k|| to determine when to stop
while τ > ετ_1 do
  find the index k of the largest absolute unused row entry in q_r
  chalk up one rank and save the row index in rows(r)
  compute v_k
  let p_r be v_k orthogonalized against the previous row vectors
  normalize p_r
  find the index k of the largest absolute entry in p_r
  compute the kth column u_k
  save the column index k
  let q_r be v_k orthogonalized against the previous row vectors
 let tau = ||q_r|| and normalize q_r
end
reduce the rank by one
let Q^{-1} be the inverse of Q(rows(:), :)
let R = Q^{-1}V(:, rows(:))
```

Figure 7.6: *Dual-MGS pseudo-code.*

7.3.5 Merging

The merging scheme is the hart of the IES^3 method, but the same idea works well in our algorithm, since it can be used to improve the compression where heuristics, that focus on the geometric separation of the source elements and field points, might fail. A thorough time complexity analysis might show that there exists a problem dimension where kernel evaluations might be cheap enough to consider merging smaller decompositions.

IES^3 Merging

The merging scheme can be applied either horizontally or vertically first, the diagram shows the general idea where the matrices are merged horizontally first.

$$\begin{array}{|cc|} \hline U_1V_1^T & U_2V_2^T \\ U_3V_3^T & U_4V_4^T \\ \hline \end{array} \begin{array}{c} \rightarrow \\ \rightarrow \end{array} \begin{array}{|c|} \hline A_{12} = U_{12}V_{12}^T \\ A_{34} = U_{34}V_{34}^T \\ \hline \end{array} \begin{array}{c} \searrow \\ \nearrow \end{array} \begin{array}{|c|} \hline \\ A = UV^T \\ \\ \hline \end{array} \tag{7.28}$$

First we try to express the column space U_2 of A_2 in terms of the column space U_1 of A_1. The ideal is that U_2 is linearly dependent on all the vectors in U_1, because then $U_{12} = U_1$, but in general U_{12} would need to include those vectors which could not be represented up to a given error by the vectors in U_1. The algorithm described in [39] was to construct a matrix $X = [U_1U_2]$ and to obtain the new column basis U_{12} using the pivoted Gram-Schmidt method.

Then it is easy to obtain the row basis by $V_{12}^T = U_{12}^T \cdot [U_1V_1^TU_2V_2^T]$ with which we can express the new matrix A_{12}. The same is then done to get A_{34}. In the next step matrices A_{12} and A_{34} need to be merged vertically. To do this we need to construct the row basis for the matrix A. This is done similarly to the horizontal merge, except that now a matrix $Z = [V_{12}V_{34}]$ is constructed from the row basis vectors of A_{12} and A_{34} and pivoted Gram-Schmidt applied to obtain the row basis V for A.

From this the column basis U is computed using $U = [U_{12}V_{12}^TU_{34}V_{34}^T] \cdot V$. This completes the merge of the four sub-matrices. The major advantage of using the merge scheme is the reduced complexity of computing a low rank decomposition of a large matrix which is known to be of low rank compared to using a conventional Gram-Schmidt method. Suppose that all four sub-matrices were square and of dimension $k \times k$ and of rank r, then the merge scheme has complexity $O(kr^2)$, while Gram-Schmidt would have complexity $O(k^2r)$ [39].

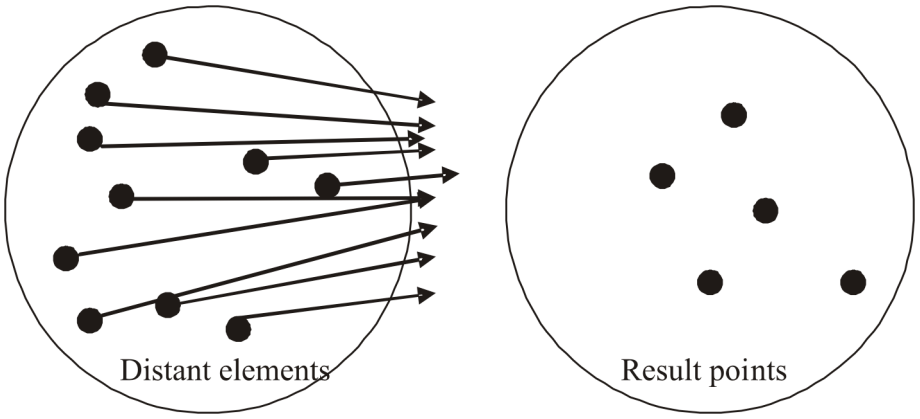


Figure 7.7: *Direct influence of distant elements used to evaluate at result points.*

Lanczos Merging

Let us now suppose that the sub-matrix we wish to merge does not only contain decompositions, but that it could be expressed by a tree or in hierarchical matrix form. It might be possible that even though a sub-matrix could not be expressed as a low rank decomposition on its own, it could still be linearly dependent on its neighbours. Merging it together with its neighbours might then give a more storage efficient decomposition. When using the IES^3 merging scheme the assumption is that sub-matrices are all decompositions and that fact is also used in the algorithm.

The matrix-vector and matrix-matrix multiplication operations are defined for the sub-matrices. Therefore it would be possible to either apply the Arnoldi MGS in Figure 7.5 or the Lanczos bidiagonalization to do the merge, since these methods only use the matrix-vector multiplication operation. The complexity of these techniques under the same assumptions used to determine the complexity of the IES^3 merging scheme also is of $O(kr^2)$ if it assumed that the most time is spent on the matrix-vector multiplication.

7.4 The Tree Method

The tree method will be described as the predecessor of the fast multipole method (FMM). The panel clustering method is essentially a variant of the tree method. The tree method (and FMM) requires the expansion of kernels in terms of spherical harmonics. The tree method only requires one type of expansion, called the multipole expansion, while FMM also uses another expansion called the local expansion to speed up the tree method. The construction of the multipole moment expansion often requires difficult mathematical manipulations to get analytical expressions for the expansion coefficients.

The expansion coefficients capturing the influences of elements are computed at the expansion point like in Figure 7.8. The number of coefficients used in the expansion de-

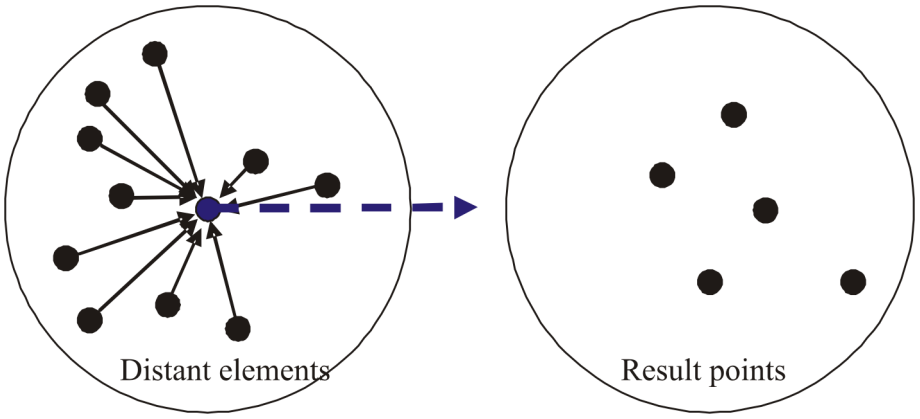


Figure 7.8: *Multipole moment expansion of distant elements used to evaluate at result points.*

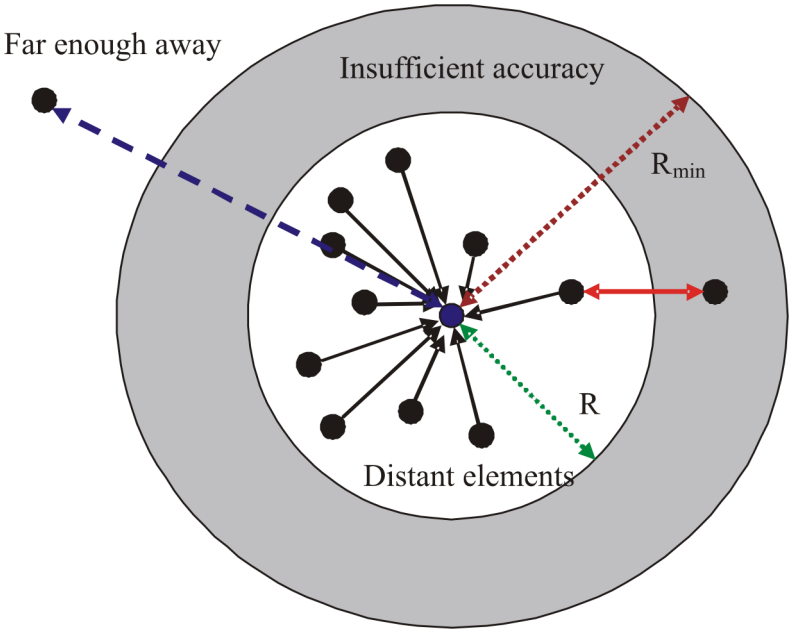


Figure 7.9: *Illustrating when multipole expansion and direct expansion will be used.*

termines how accurately the influences of the elements can be captured. This expansion is similar to the Taylor expansion. Formulae can be derived on the accuracy of the expansion based on the radius of the sphere enclosing all the elements used in the expansion, the distance to the result point and the number of coefficients used in the expansion. The required accuracy might not be reachable if the result point lies to close to where the expansion was computed. If that is the case the influences for each element will have to be computed directly, like in Figure 7.7. Figure 7.9 illustrates the categories used to determine whether to use direct influence or the multipole expansion coefficients to calculate results at the field points.

Sometimes the number of coefficients used to construct an expansion is fixed and for

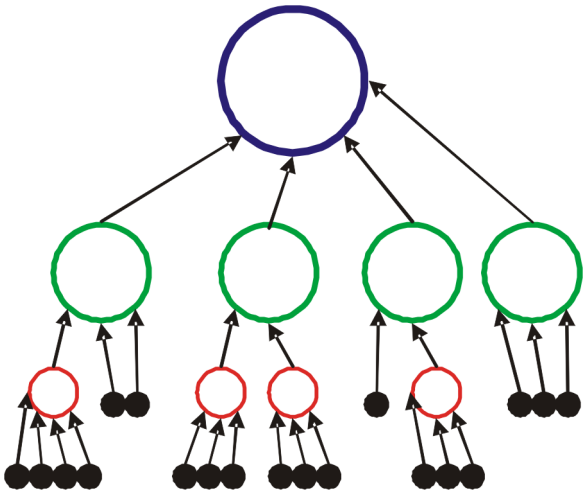


Figure 7.10: *Depicts a quad-tree of the elements in the problem domain.*

this discussion we will assume that this is the case. We are aiming to use the expansion rather than directly computing influences, because evaluation using the expansion is much faster than direct kernel evaluation. This is only true if the expansion represents a reasonable, but small, number of elements. We will aim to always have at least the optimal number of elements for each expansion. This will reduce the number of direct kernel evaluations which we assume to be the most expensive part of the solution.

The elements which describe the problem will then be grouped such that the radii of the closest fitting spheres around these groups are a minimum. In practice constructing such a grouping might be difficult. We will approximate such a grouping by using a quad-tree in two dimensions and an oct-tree in three dimensions, which is similar to the tree structured described in Section 7.2 and depicted in Figure 7.10. The tree will then represent a partitioning of the space enclosing the problem domain, grouping nearby elements together as shown in Figure 7.11. The leaves contain the element groupings and the expansion coefficients for each group will be computed at the enclosing sphere's centroid. There are many issues that will have to be resolved when constructing the tree. We will not focus on these issues, but encourage the reader to look at efficient ways of constructing, traversing and possible balancing the tree.

Evaluating these expansions and direct kernel evaluations where necessary, as shown in Figure 7.12, it is possible to accurately compute the influence of the problem domain at any point faster than just using direct evaluations. We deliberately use a tree to construct the element groups, because at each node at any level of the tree it is possible to find a sphere which will enclose all the elements contained in the children of that node.

Suppose that the result point is far enough away from the problem domain that using an expansion at the center of the whole problem domain we would achieve the desired accuracy or better. Then it would make sense to use that expansion rather than all the

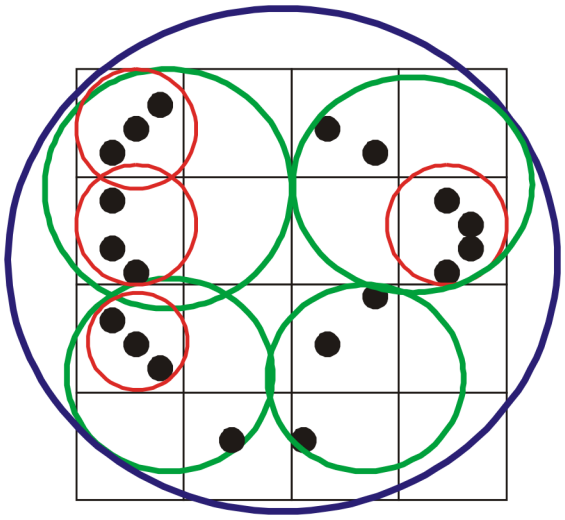


Figure 7.11: *Illustrating how elements are split using a quadtree and then grouped into overlapping spheres on different levels.*

expansions which was created at the leaves of the tree.

It would be much faster to calculate using only one set of expansion coefficients rather than all expansions at the leaves of the tree. For a two dimensional problem and for which we use a quad-tree each node will have four children. It would then be four times less work to use one set of coefficients centered at the node's center rather than each child's coefficient set.

The result point could be anywhere and generally we will want to place many result points almost everywhere. Therefore we would want to have coefficients available at almost every node in the tree to minimize the number of calculations we have to perform when calculating the influences at those points.

The tree method which is the father of FMM tackles this problem very elegantly. Instead of calculating the coefficients at every node in the tree separately, instead the work done to compute the coefficients at children in the tree is used to compute coefficients at the nodes in the tree. Formulae are derived which move the expansion coefficients at each child's expansion point to their parent expansion points. These translation formulae are much cheaper than directly constructing the expansions for the child elements, which become more the higher the node is in the tree.

When calculating at many result points we determine for each result point the least number of coefficients and direct evaluations by traversing the tree and using the first set of expansions for each branch with which we will achieve the desired accuracy. This is a huge gain compared to direct kernel evaluations. Still this is just how the tree method works.

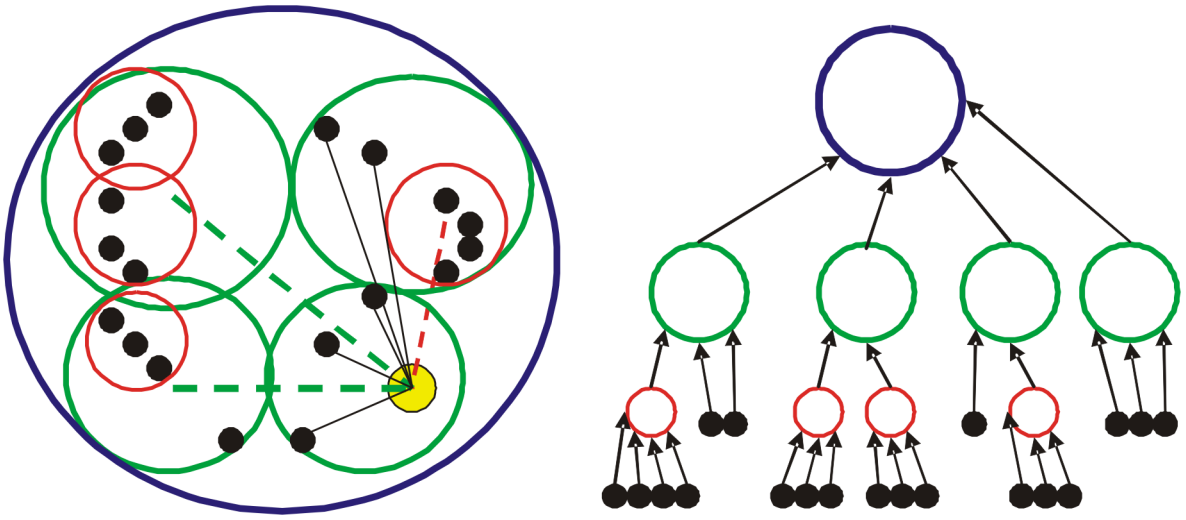


Figure 7.12: *Illustrating which multipole moments and direct evaluations might be used to evaluate at the yellow result point.*

7.5 Fast Multipole Method

FMM improves the tree method by recognizing the fact that not just one result point will be computed, but rather a set of result points which might be clustered in space. The multipole expansion works on the basis that the influence of everything within a sphere with a fixed radius can be expressed in terms of the multipole coefficients.

These coefficients can then be used to compute a result point to a prescribed accuracy if it is far enough away. We could have approached this differently by asking whether it is possible to have another expansion which is computed from elements far enough away, which can be used to locally compute their effect to result points locally to a prescribed accuracy. Fortunately such an expansion, called the local expansion, does exist.

The local expansion exhibits all the same properties as the multipole expansion except that now we are focusing on computing result points locally using coefficients which describe far away elements. The local expansions are linear and therefore we can sum the contribution of each far away element and produce the local expansion coefficients which describes the contributing elements' effect. The local expansion can be translated from the parent node to its children. This is opposite to the multipole expansion where multipole coefficients are translated upward from the children to their parents.

On each level of the tree we update the coefficients to take into account nearer elements which fall outside of the smaller local expansion enclosing sphere. When we reach the leaf we then evaluate the local expansion to compute the effects of all far away elements and add the effect of close elements which have to be computed directly. The local expansion could be used to compute a large set of result points contained within the local expansion's enclosing sphere from elements which are far enough away very effectively and accurately.

The last missing ingredient converts the multipole expansion into a local expansion. This completes the tree method and produces the FMM.

7.6 Summary

In this chapter, techniques for representing the influence matrix were reviewed. The importance of ordering the discretized elements was discussed. Elements which are in close proximity spatially should also have labels which are close together to ensure that the influence matrix contain influences between large clusters of spatially disjoint well separated elements. A quick description of the KD-tree structure was given, which can be used to construct such an ordering.

The IES^3 method, which was introduced by [7], was explained. The aim of compression and how such a compression can be obtained using pure algebraic methods, was described. Then, assuming that a compressed hierarchical matrix structure exists, a method of traversing the structure to compute a matrix-vector multiplication was given. Decomposing matrices efficiently and accurately is essential to the algebraic approach. Many different ways are available. They are all based on the SVD which is accurate, but not efficient at all.

The modified Gram-Schmidt method, is acceptably less accurate, but still very inefficient, because it requires the input matrix to exist a priori. Still the Gram-Schmidt method is useful, since it can be used to merge decompositions. The Lanczos bi-diagonalization technique is an iterative method of constructing a reduced SVD. Since it is an iterative method, it can be used to merge arbitrary matrix representations as long as matrix-vector multiplication is defined.

The Dual-MGS method is useful when constructing decompositions without evaluating the whole input matrix. Instead rows and columns are evaluated as needed, which allows this method to construct a decomposition much more efficiently than previously mentioned techniques. Algorithms of this kind include IE-QR [40], ACA [50] and ACA+ [45], but descriptions of these algorithms were not given. Merging decompositions is useful, because it can reduce the storage requirements and in turn improve the efficiency of the matrix-vector multiplication. Two methods, one using modified Gram-Schmidt and another using bi-diagonalization were discussed.

Lastly a quick description is given of the tree and fast multipole methods which have been succesfully applied to elastostatics, of which [20; 53; 54; 10] are only a few examples.

Chapter 8

Summary and Conclusion

The aim was to research and give an overview of the main concepts needed to create an efficient linear elastic static stress boundary element solver which can be used to solve problems in the South-African mining environment. The gold and platinum ore bodies are very much planar and thin, compared to the extent of the mining. These ore bodies are accessed by shafts and development tunnels which are very expensive to create and which does not in general contain ore. They provide access ways and much needed ventilation for the mine, therefore it is in the interest of the mines to know when mining activities or fault movement are causing excessive stress which could cause damage to these tunnels. The theory to handle the different kinds of geometries using the BEM had to be researched.

8.1 General Summary

An overview of the theory of linear elasticity was given; covering the important aspects needed. The first important item was how strain in an elastic body relates to the displacement field and how the linearized theory is obtained from the general displacement-strain relationship. This was followed by a brief description of stress and a discussion on representing the in situ stress state, which was later used to define boundary conditions. Then the elastic moduli which describes the continuum and used in Hooke's law, which relates strain and stress through the stress-strain relationships, was covered. Finally the compatibility equations, which ensure continuity in the continuum and the conditions for elastic equilibrium, followed.

The fundamental solutions, which form the cornerstone for solving problems in elasticity using the BEM was discussed. Even though only Kelvin's solution was used in subsequent chapters, similar techniques can be applied to the other solutions to create applications to solve useful engineering problems. A whole section was devoted to modelling thin excavations as fractures or cracks. This included the derivation of the displacement discontinuity kernels for use in the indirect BEM.

Integral equations are solved by the boundary element method. The BEM only required the boundary of the domain to be discretized and therefore reduces the dimensionality of the problem by one. There are many different ways of constructing and solving the BEM. The unknowns could be collocated with the boundary and solved for exactly (collocation) or solved for in a least squares sense (Galerkin). The unknowns could represent the actual physical quantities being solved for (direct BEM) or fictitious quantities which could be used to find the unknown physical quantities which were not prescribed (indirect BEM).

The BEM requires integration of the singular kernel functions over the boundary surface elements. These integrals are regular and tractable using ordinary numerical quadrature when the field point is far away from the integration domain, but when the field point is on or near the integration domain, then special care have to be taken to interpret these integrals in a meaningful way. The continuation approach provides a unified framework which can be used to integrate the kernels in an efficient, robust and accurate way.

The final step is to quickly build an efficient representation of the influence matrix which can then be fed to an iterative solver like Bi-CG or GMRES. The IES^3 method seemed to be a very understandable and easily implementable method and promised to deliver a quick, robust and efficient solver. The flow of the method was explained and various algorithms needed to implement each part of the method were discussed. Other methods like the tree and fast multipole methods were investigated and a quick outline of these methods was given.

8.2 Conclusion

The BEM can be used as a tool to model large mining layouts. The derivations of the elasticity kernels are useful in understanding the fundamentals needed to devise accurate integration algorithms to use in implementing the BEM. The FFM coupled with the DDM can be used to model both volumetric (development tunnels) and tabular (planar ore bodies) geometry. The BEM can be coupled to a domain method if more detailed analysis of a specific region in a mine is required. If such an analysis is required, then using the symmetric Galerkin method is the most common approach in recent articles, because it can be coupled with the FEM. The algebraic representation of the influence function contains many numerical pitfalls and it is difficult to handle all the special cases that might arise. A method based on the relatively recent article on a kernel-independent FMM [10] seems to give better results. The BEM field is actively being researched and promises exciting new developments and applications.

Bibliography

[1] Saad, Y.: *Iterative Methods for Sparse Linear Systems.* 2nd edn. SIAM, 2003.

[2] Burkholder, R.J. and Lee, J.-F.: Fast Dual-MGS Block-Factorization Algorithm for Dense MoM Matrices. *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 7, pp. 1693–1699, July 2004.

[3] Ying, L.: *An Efficient and High-Order Accurate Boundary Integral Solver for the Stokes Equations in Three Dimensional Complex Geometries.* PhD thesis, Department of Computer Science, New York University, 2004.

[4] Ding, J.: *Fast Boundary Element Method Solutions for Three Dimensional Large Scale Problems.* PhD thesis, Mechanical Engineering, Georgia Institute of Technology, 2004.

[5] Harbrecht, H. and Schneider, R.: Wavelets for the fast solutions of boundary integral equations. In: Mang, H.A., Rammerstorfer, F.G. and Eberhardsteiner, J. (eds.), *Fifth World Congress on Computational Mechanics.* July 2002.

[6] Hackbusch, W.: The Panel Clustering Algorithm . In: Whiteman, J.R. (ed.), *MAFELAP*, pp. 339–348. Academic Press, London, April 1990.

[7] Kapur, S. and Long, D.E.: IES^3: A Fast Integral Equation Solver for Efficient 3-Dimensional Extraction. *ICCAD*, pp. 448–455, 1997.

[8] Chandrasekaran, S., Dewilde, P., Gu, M., Pals, T. and van der Veen, A.J.: Fast Stable Solver for Sequentially Semi-separable Linear Systems of Equations. In: *HiPC '02: Proceedings of the 9th International Conference on High Performance Computing*, pp. 545–554. Springer-Verlag, London, UK, 2002.

[9] Börm, S., Grasedyck, L. and Hackbusch, W.: Hierarchical Matrices, 2005. Winterschool.

[10] Ying, L., Biros, G. and Zorin, D.: A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, vol. 196, pp. 591–626, 2004.

[11] Schmidlin, G.: *Fast Solution Algorithms for Integral Equations in R^3*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 2003.

[12] Brebbia, C.A.: *The Boundary Element Method for engineers*. Pentech Press Limited, London, 1978.

[13] Davis, R.O. and Selvadurai, A.P.S.: *Elasticity and Geomechanics*. Cambridge University Press, 1996.

[14] Miranda-Valenzuela, J.C. and Muci-Küchler, K.H.: *Adaptive Meshing with Boundary Elements*, vol. 41 of *Topics in Engineering*. WIT Press, 2002.

[15] Flügge, W.: *Tensor Analysis and Continuum Mechanics*. Springer-Verlag Berlin Heidelberg, 1972.

[16] Gao, X.-W. and Davies, T.G.: *Boundary Element Programming in Mechanics*. Cambridge University Press, 2002.

[17] du Plessis, J.P.: Kontinuum Modellering, 2002. Course Notes.

[18] Beer, G.: *Programming the Boundary Element Method: An Introduction for Engineers*. John Wiley and Sons, Limited, 2001.

[19] Ryder, J.A. and Jager, A.J.: *A textbook on rock mechanics for tabular hard rock mines*. 1st edn. The Safety in Mines Research Advisory Committee, 2002.

[20] Yoshida, K.-I.: *Applications of Fast Multipole Method to Boundary Integral Equation Method*. PhD thesis, Department of Global Environment Engineering, Kyoto University, Japan, March 2001.

[21] Crouch, S.L. and Starfield, A.M.: *Boundary Element Methods in Solid Mechanics*. George Allen and Unwin (Publishers) Ltd, London, 1983.

[22] Linkov, A.M.: *Boundary Integral Equations in Elastic Theory*, vol. 99 of *Solid mechanics and its applications*. Kluwer Academic Publishers, 2002.

[23] Watson, J.O.: Boundary Elements from 1960 to the Present Day. *Electronic Journal of Boundary Elements*, vol. 1, no. 1, pp. 34–46, 2003.

[24] Yuan-fang Wang, S.C.: Toward Real-Time, Physically-Correct Soft Tissue Behavior Simulation. 2003.
Available at: `http://citeseer.ist.psu.edu/707792.html`

[25] Yacoub, T.: *Three-dimensional analysis of lenticular ore bodies using displacement discontinuity elements*. PhD thesis, University of Toronto, 1999.

[26] Taylor, M.A., Wingate, B.A. and Vincent, R.E.: An Algorithm for Computing Fekete Points in the Triangle. *SIAM Journal on Numerical Analysis*, vol. 38, no. 5, pp. 1707–1720, 2001.

[27] Zhao, Z. and Yuan, W.: Evaluation of singular integrals in the symmetric Galerkin boundary element method. *Advances in Engineering Software*, vol. 35, pp. 781–789, 2004.

[28] Liu, Y.J.: On the simple-solution method and non-singular nature of the BIE/BEM - a review and some new results. *Engineering Analysis with Boundary Elements*, vol. 24, pp. 789–795, 2000.

[29] Mack, M.G.: *A three dimensional boundary element method for elastodynamics.* PhD thesis, University of Minnesota, June 1991.

[30] Bond, C.: A new integration method providing the accuracy of Gauss-Legendre with error estimation capability (rev.). 2002.
Available at: `http://www.crbond/papers/gbint.pdf`

[31] Birger, C.B.: *Gauss Bond Quadrature.* PhD thesis, South Dakota State University, 2005.

[32] Vijayakumar, S. and Cormack, D.E.: An invariant imbedding method for singular integral evaluation on finite domains. *SIAM Journal on Applied Mathematics*, vol. 48, no. 6, December 1988.

[33] Vijayakumar, S. and Cormack, D.E.: A new concept in near-singular integral evaluation: The continuation approach. *SIAM Journal on Applied Mathematics*, vol. 49, no. 5, pp. 1285–1295, October 1989.

[34] Rosen, D. and Cormack, D.E.: Singular and near singular integrals in the BEM: A global approach. *SIAM Journal on Applied Mathematics*, vol. 53, no. 2, pp. 340–357, April 1993.

[35] Rosen, D. and Cormack, D.E.: The continuation approach: A general framework for the analysis and evaluation of singular and near-singular integrals. *SIAM Journal on Applied Mathematics*, vol. 55, no. 3, pp. 723–762, June 1995.

[36] Hadamard, J.: *Lectures on Cauchy's Problem in Linear Partial Differential Equations.* Dover Publications, New York, 1952.

[37] Vijayakumar, S., Yacoub, T.E. and Curran, J.H.: A node-centric indirect boundary element method: three dimensional displacement discontinuities. *Computers and Structures*, vol. 74, pp. 687–703, 2000.

[38] Goreinov, S.A., Tyrtyshnikov, E.E. and Zamarashkin, N.L.: A Theory of Pseudoskeleton Approximations. *Linear Algebra Applications*, vol. 261, pp. 1–21, 1997.

[39] Kapur, S. and Zhao, J.: A Fast Method of Moments Solver for Efficient Parameter Extraction of MCMs. In: *Design Automation Conference*, pp. 141–146. 1997. Available at: `http://citeseer.ist.psu.edu/kapur97fast.html`

[40] Zhao, K. and Lee, J.-F.: A Single-Level Dual Rank IE-QR Algorithm to Model Large Microstrip Antenna Arrays. *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 10, pp. 2580–2585, October 2004.

[41] Chandrasekaran, S., Gu, M. and Pals, T.: Fast and Stable Algorithms for Hierarchically Semi-separable Representations, September 2002. Available at: `http://www.mirrorsky.com/papers/hss1.pdf`

[42] Chandrasekaran, S., Gu, M. and Lyons, W.: A Fast and Stable Adaptive Solver for Hierarchically Semi-separable Representations, May 2004. Available at: `http://www.mirrorsky.com/papers/hss.pdf`

[43] Hackbusch, W.: A Sparse Matrix Arithmetic based on $\mathcal{H}$-Matrices. Part I: Introduction to $\mathcal{H}$-Matrices. *Computing*, vol. 62, pp. 89–108, 1999.

[44] Grasedyck, L., Hackbusch, W. and Borne, S.L.: Adaptive Geometrically Balanced Clustering of $\mathcal{H}$-Matrices. *Computing*, vol. 73, pp. 1–23, 2004.

[45] Grasedyck, L.: Adaptive Recompression of $\mathcal{H}$-Matrices for BEM.

[46] Hackbusch, W. and Khoromskij, B.N.: A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. *Computing*, vol. 64, pp. 21–47, 2000.

[47] Hackbusch, W. and Khoromskij, B.N.: $\mathcal{H}$-matrix approximation on graded meshes. December 1999. Max-Planck-Insitut für *Mathematic in den Naturwissenschaften.*

[48] Tyrtyshnikov, E.E.: Incomplete Cross Approximation in the Mosaic-Skeleton Method. *Computing*, vol. 64, pp. 367–380, 2000.

[49] Simon, H.D. and Zha, H.: Low-Rank Matrix Approximation Using the Lanczos Bidiagonalization Process with Applications. *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 2257–2274, 2000. Available at: `http://citeseer.ist.psu.edu/simon00low.html`

[50] Vouvakis, M., Zhao, K., Seo, S.-M. and Lee, J.-F.: A domain decomposition approach for non-conformal couplings between finite and boundary elements for

unbounded electromagnetic problems in R^3. *J. Comput. Phys.*, vol. 225, no. 1, pp. 975–994, 2007. ISSN 0021-9991.

[51] Stoer, J. and Bulirsch, R.: *Introduction to Numerical Analysis.* Springer-Verlag, NY, NY, 1979.

[52] Sun, X. and Pitsianis, N.P.: A matrix version of the fast multipole method. *Proc. SIAM Rev.*, vol. 43, pp. 289–300, 2001.

[53] Wang, P.B. and Yao, Z.H.: Fast multipole DBEM analysis of fatigue growth. *Computational Mecahnics*, vol. 38, pp. 223–233, 2006.

[54] Fu, Y., Klimkowski, K.J., Rodin, G.J., Berger, E., Browne, J.C., Singer, J.K., van de Geijn, R.A. and Vemaganti, K.S.: A fast solution method for three-dimensional many-particle problems of linear elasticity. *International Journal for Numberical Methods in Engineering*, vol. 42, pp. 1215–1229, 1998.

Appendix A

Strain

This appendix supplements Section 2.2 and covers the derivation of the strain tensor using tensor notation and a similar approach as was taken in [15, Flügge, pp. 23-28]. Let us assume a convected coordinate system and describe the undeformed body with basis vectors $\hat{\mathbf{g}}_i$ and metric tensor $\hat{g}_{ij}$. A line element can then be written as

$$d\hat{\mathbf{s}} = \hat{\mathbf{g}}_i dx^i \tag{A.1}$$

and its metric, which is the square of equation A.1, by

$$d\hat{\mathbf{s}} \cdot d\hat{\mathbf{s}} = \hat{g}_{ij} dx^i dx^j. \tag{A.2}$$

After deformation the line element $d\mathbf{s}$ can have a different length or orientation and can be written as

$$d\mathbf{s} = \mathbf{g}_i dx^i \tag{A.3}$$

and its metric by

$$d\mathbf{s} \cdot d\mathbf{s} = g_{ij} dx^i dx^j, \tag{A.4}$$

where the basis vectors $\mathbf{g}_i$ are for the deformed body's coordinate system.
The measure of deformation of the body is given by the change in metric tensor γ_{ij},

$$\gamma_{ij} = g_{ij} - \hat{g}_{ij}. \tag{A.5}$$

To show that γ_{ij} is indeed a tensor we transform the line segments $d\hat{\mathbf{s}}$ and $d\mathbf{s}$ in the undeformed and deformed coordinate systems to another coordinate system

$$\hat{\mathbf{g}}_{i'} = \hat{\beta}^i_{i'} \hat{\mathbf{g}}_i$$
$$\mathbf{g}_{i'} = \beta^i_{i'} \mathbf{g}_i$$

Using the initial assumption of convected coordinates we have that

$$\hat{\beta}^i_{i'} = \frac{\partial \hat{x}^i}{\partial \hat{x}^{i'}} = \frac{\partial x^i}{\partial x^{i'}} = \beta^i_{i'} \tag{A.6}$$

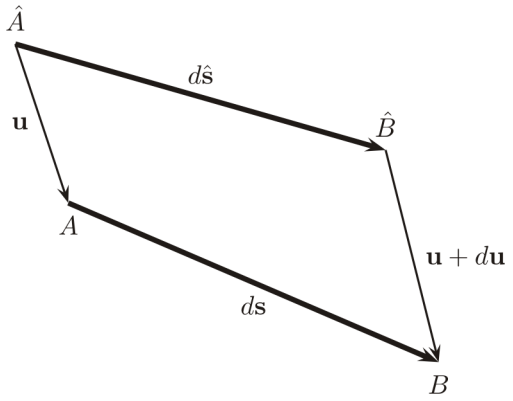


Figure A.1: *Displacement during deformation.*

from which it is clear that both $\hat{\mathbf{g}}_i$ and $\mathbf{g}_i$ transform similarly and therefore

$$\begin{aligned}\gamma_{i'j'} &= g_{i'j'} - \hat{g}_{i'j'} \\ &= \beta^m_{i'}\beta^n_{j'}g_{mn} - \hat{\beta}^m_{i'}\hat{\beta}^n_{j'}\hat{g}_{mn} \\ &= \beta^m_{i'}\beta^n_{j'}(g_{mn} - \hat{g}_{mn}) \\ &= \beta^m_{i'}\beta^n_{j'}\gamma_{mn}\end{aligned}$$

from which we can see that γ_{ij} is indeed a tensor. The metric tensor is symmetric which implies that the strain tensor is also symmetric, i.e.

$$\gamma_{ij} = \gamma_{ji}. \tag{A.7}$$

Next we wish to express this tensor in terms of the displacement vector $\mathbf{u} = \mathbf{g^i u_i}$ which connects particles in the undeformed body with particles in the deformed body. We will need to differentiate this vector

$$d\mathbf{u} = \mathbf{u}_{,j} = u_i|_j \mathbf{g}^i dx^j. \tag{A.8}$$

Taking the derivative of a general vector v with respect to some basis we may write

$$\mathbf{v}_{,j} = (v^i_j + v^k\Gamma^i_{jk})\mathbf{g}_i = v^i|_j\mathbf{g}_i \tag{A.9}$$

where Γ^i_{jk} is called the Christoffel symbols and for Cartesian coordinates $\Gamma^{ijk} = \Gamma^i_{jk} = 0$. Let $\hat{A}$ and $\hat{B}$ be two points inside of the undeformed material. When the material is deformed the points' relative distances changes as shown in Figure A.1. Applying vector arithmetic to get from A to $\hat{B}$ gives

$$\mathbf{u} + d\mathbf{s} = d\hat{\mathbf{s}} + \mathbf{u} + d\mathbf{u} \tag{A.10}$$

$$d\mathbf{s} = d\hat{\mathbf{s}} + u_i|_j\hat{\mathbf{g}}^i dx^j \tag{A.11}$$

$$
\begin{aligned}
d\mathbf{s}\cdot d\mathbf{s} &= (\hat{\mathbf{g}}_i dx^i + u_k|_i \hat{\mathbf{g}}^k dx^i)\cdot(\hat{\mathbf{g}}_j dx^j + u_l|_j \hat{\mathbf{g}}^l dx^j) \\
&= (\hat{\mathbf{g}}_i + u_k|_i \hat{\mathbf{g}}^k)\cdot(\hat{\mathbf{g}}_j + u_l|_j \hat{\mathbf{g}}^l) dx^i dx^j \\
&= (\hat{g}_{ij} + u_k|_i \hat{\mathbf{g}}_j\cdot\hat{\mathbf{g}}^k + u_l|_j \hat{\mathbf{g}}_i\cdot\hat{\mathbf{g}}^l + u_k|_i u_l|_j \hat{\mathbf{g}}^k\cdot\hat{\mathbf{g}}^l) dx^i dx^j \\
&= (\hat{g}_{ij} + u_j|_i + u_i|_j + \hat{g}^{kl} u_k|_i u_l|_j) dx^i dx^j \\
&= (\hat{g}_{ij} + u_i|_j + u_j|_i + u^l|_i u_l|_j) dx^i dx^j .
\end{aligned}
$$

Therefore from (A.4)

$$g_{ij} = \hat{g}_{ij} + u_i|_j + u_j|_i + u^l|_i u_l|_j . \tag{A.12}$$

Using equation (A.2) and (A.5) we get

$$\gamma_{ij} = u_j|_i + u_i|_j + u^l|_i u_l|_j . \tag{A.13}$$

For small displacements the last term, which is quadratic in displacement, can be ignored and thus (A.13) reduces to the linearized form

$$\gamma_{ij} = u_j|_i + u_i|_j . \tag{A.14}$$

These two equations have only to do with the geometry of the deformation and is called the *kinematic relations*.
The measure commonly used for strain is given by

$$\varepsilon_{ij} = \frac{1}{2}\gamma_{ij} \tag{A.15}$$

$$= \frac{1}{2}(u_i|_j + u_j|_i + u_i^k u_k|_j) \tag{A.16}$$

and for small deformations

$$\varepsilon_{ij} = \frac{1}{2}(u_i|_j + u_j|_i) \tag{A.17}$$

can be used.

Appendix B

Strain Energy Density

In [15, Flügge, p. 51] the existence of a strain energy density function is used to show that

$$E^{ijmn} = E^{mnij}. \tag{B.1}$$

Let the strain energy density a be expressed by

$$a = \frac{1}{2}\sigma^{ij}\varepsilon_{ij}. \tag{B.2}$$

The differential da is given by

$$da = \frac{\partial a}{\partial \sigma^{ij}} d\sigma^{ij} + \frac{\partial a}{\partial \varepsilon_{ij}} d\varepsilon_{ij}. \tag{B.3}$$

Since stress is a function of strain the differential can be written as

$$d\sigma^{ij} = \frac{\partial \sigma^{ij}}{\partial \varepsilon_{mn}} d\varepsilon_{mn}. \tag{B.4}$$

The chain rule is applied to the first term of equation (B.3), by renaming indices in equation (B.4) and substituting into equation (B.3), such that

$$da = \frac{\partial a}{\partial \sigma^{mn}} \frac{\partial \sigma^{mn}}{\partial \varepsilon_{ij}} d\varepsilon_{ij} + \frac{\partial a}{\partial \varepsilon_{ij}} d\varepsilon_{ij}. \tag{B.5}$$

Applying the partial differentiations with respect to equation (B.2) it follows that

$$da = \frac{1}{2}\left(\varepsilon_{mn}\frac{\partial \sigma^{mn}}{\partial \varepsilon_{ij}} + \sigma^{ij}\right) d\varepsilon_{ij}. \tag{B.6}$$

Differentiating the stress-strain relations

$$\sigma^{ij} = E^{ijmn}\varepsilon_{mn}, \tag{B.7}$$

which are constant with respect to stress and strain and substituting the result into equation (B.6) gives

$$da = \frac{1}{2}\left(\varepsilon_{mn}E^{mnij} + \sigma^{ij}\right) d\varepsilon_{ij}. \tag{B.8}$$

The strain energy increment da for a small increment of stress and strain can be approximated to first order accuracy by the work incremented for a small increment in strain done on the existing stress and expressed as,

$$da = \sigma^{ij} d\varepsilon_{ij}. \tag{B.9}$$

Equating (B.8) and (B.9) gives

$$\sigma^{ij} = \frac{1}{2}\left(\varepsilon_{mn} E^{mnij} + \sigma^{ij}\right) \tag{B.10}$$

whence

$$\sigma^{ij} = \varepsilon_{mn} E^{mnij}. \tag{B.11}$$

Comparing equation (B.11) with equation (B.7) shows that equation (B.1) holds.