Multilevel Multi-Integration Algorithm for Acoustics

Isaías Hernández Ramírez

TABLE OF CONTENTS

INTRODUCTION

1 CHAPTER

Acoustics is defined as the science of sound, including its production, transmission and effects [51]. Sound can be described as a succession of compressions and rarefactions propagating through a medium (solid or fluid) induced by a certain source. Sound is generally characterized by its frequency (in Herz) and its intensity (in dB).

Audible sound is the sound that humans are able to perceive such as the voice during speech, a car passing on the street, the ringing of (mobile) phones, and music. The frequency range of audible sound is between 20 Hz and 20 kHz. Sound with a frequency below 20 Hz is referred to as *infrasound*. It can not be detected by the human ear. Typical examples of infrasound are found in nature; the sound produced by earthquakes or sea waves. Infrasound can also be produced by devices used in daily life such woofers, vehicle structures, and fans. Research has indicated that prolonged exposure to infrasound forms a serious health risk [65]. On the other side of the spectrum, *ultrasound* encompasses all sound with a frequency larger than the upper limit of the human hearing (20 KHz). In nature, some animals like dogs, dolphins and bats, have an upper limit higher than the human ear and thus can hear ultrasound and even use it for navigation and communication. In our society ultrasound is often used as a diagnostic tool in industry and medicine. For example; ultrasonography enables visualization of muscles and soft tissues, making it a useful way to scan a human body and diagnose specific health problems. Probably the most well-known application of ultrasound is in obstetrics to monitor the foetal development and growth during pregnancy.

The field of acoustics ranges from fundamental physics to engineering, earth sciences, life sciences, and arts. Pierce [51] gives an overview chart indicating the scope and ramifications of acoustics. Clearly acoustics affects many aspects of our daily lives.

In engineering in the past decade the attention for acoustics has increased enormously. The main reason is the need to reduce unwanted sound, or noise. In our society the noise level to which people are exposed has increased so much that a quiet living and working environment is almost a luxury. The rapidly intensifying use of the urban environment including the intertwining of infrastructure, housing, and industry leads to increasingly restrictive demands on noise performance of installations, vehicles, and appliances. A low noise level is an essential quality and marketing advantage for products ranging from transport vehicles (airplanes, high-speed trains, trucks, cars) to personal appliances (air conditioners, computers, hair dryers, vacuum cleaners, shavers). For guidance in the design of silent products there is an urgent need for computational tools to help localize, identify, and accurately predict sources of noise.

1.1 Sources of sound

Sound is the result of a disturbance in a medium. Depending on the source of the disturbance, in acoustical studies in engineering the following classification can be made [74]:

1. *Structure induced sound*: Sound caused by mechanical vibrations; typical examples are the vibration of engine casing or the string-body of a piano.

2. *Flow induced sound*: Sound generated by a disturbed flow such as vortex shedding (von Karman vortex street) or gas jet.

3. *Thermally induced sound*: Sound caused by local high variations of temperature of the fluid. A typical example is the sound produced by the lightning during storms which is associated with an electric discharge causing a sudden local expansion of the medium.

A common way to study sound propagation is by considering the idealized situation of an harmonic sinusoidal acoustic disturbance produced by a device vibrating. Such a device spreads spherically symmetric acoustic waves outward from the source with a frequency f into an unbounded fluid medium with a speed c which depends on the physical properties of the medium (see Figure 1.1). Due to the vibration of the device, the fluid particles in the medium show small local displacements in a positive and negative direction relative to the propagation direction. This generates zones of compression and rarefaction with a wavelength λ which depends on the speed of propagation and the frequency of the vibration ($\lambda = c/f$). Since the net displacement of the fluid particles is zero, the disturbance may propagate over large distances, but the fluid particles themselves remain at all time close to their original position. Owing to the fact that the wave is propagated in radial direction from the acoustic source with an average power P_{av}, the acoustic intensity only has a radial component and its time average depends only on the radial distance r from the source. So, considering the acoustic-energy-principle with S taken as the spherical surface with radius r, the acoustic intensity is proportional to r^{-2} which is known as the spherical spreading law [51].

Sound generated by a disturbance in the medium travels as a wave through the medium away from the source with a speed c that depends of the physical characteristics of the medium (unlike electromagnetic and optical waves, sound needs a medium in order to be transmitted), for example; in air at 20°C the speed of sound is 343 m/s, and for water at 20°C (and atmospheric pressure) the value of the speed of sound is 1481 m/s. In a more general case the speed of sound can be estimated from thermodynamical relations [7].

1.2 Measurement of sound

A sound wave propagating in a medium causes pressure variations and particle velocity variations at each location it passes. The common way to measure sound is based on measuring the pressure variations, which is the principle on which a microphone is based. Microphones nowadays have an excellent amplitude response coupled to a wide dynamic range of frequencies in which accurate measurements can be done. However, recently a novel sensor was introduced [29] named *microflown* which measures the sound particle velocity. The sensor consists of two, extremely thin, heated wires. The principle is that a particle velocity

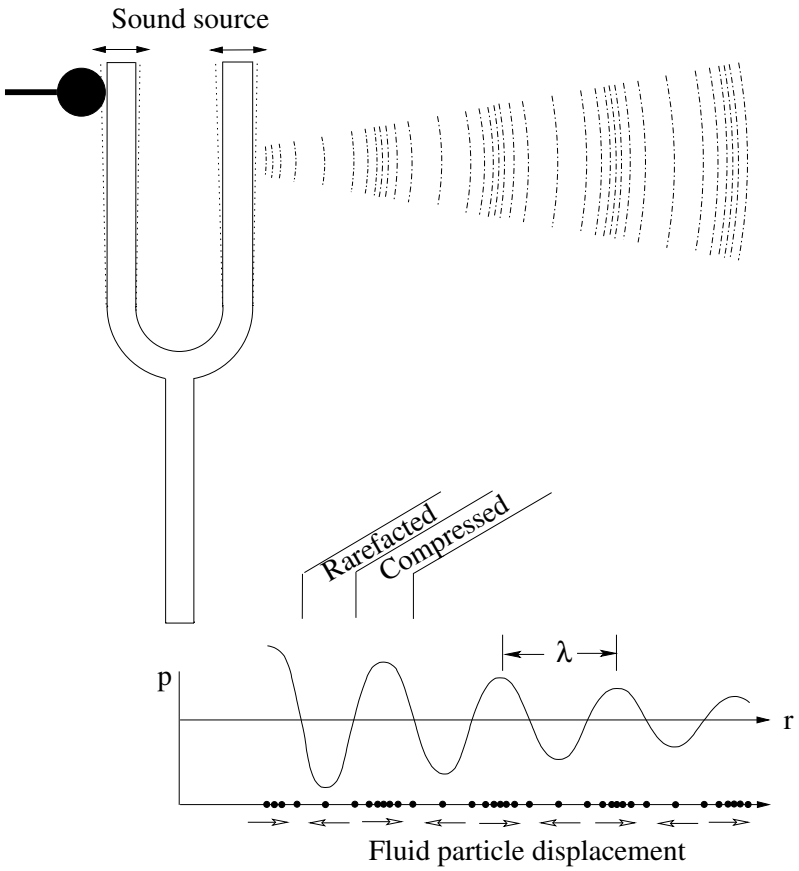


FIGURE 1.1: *Graphical representation of the sound wave propagation in a fluid. The dots represent the fluid particles with small local displacements in the direction indicated by the arrows below which produce a rarefaction and compression when the disturbance is propagated out of the source with a wevelength* λ.

variation in the direction perpendicular to the wires changes the temperature distribution instantaneously, because the upstream wire is cooled more by the airflow than the downstream wire. The resulting resistance difference provides a broadband signal that is proportional to the particle velocity, which is related to a particular sound level [29]. The sensor is illustrated in Figure (1.2).

At present the possible advantages of a particle velocity measurement over a pressure measurement are a subject of active research. The studies range from confirmation that it is really particle velocity that is measured to detailed studies of the properties of the sensor, its behavior in different situations, and possibilities for further improvement, see [57]. As sound particle velocity, by definition, is a vector quantity. In principle its measurement implies that more information becomes available than through a pressure measurement. This could be advantageous for source identification and reconstruction. For example, [73, 74] illustrate how the use of the sound particle velocity as input in an inverse numerical calculation to compute

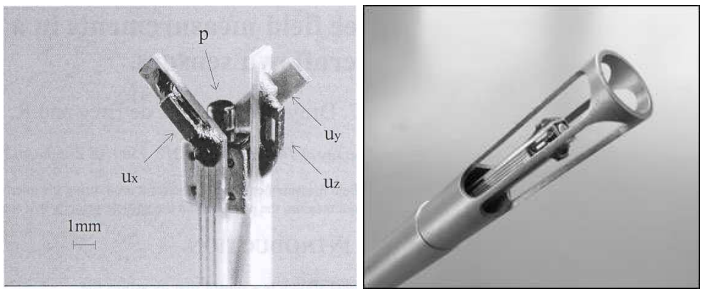


FIGURE 1.2: *Examples of 3D particle velocity sensor (microflown) combined with a pressure sensor (microphone). (Pictures reproduced with the permission of μicroflown Technologies)*

the source field at the surface generating the sound, leads to a more accurate reconstruction than the use of the measured pressure as input. Finally, a combination of sound pressure and particle velocity in a single sensor can give direct information about the acoustic intensity, which again can be useful in source reconstruction [74].

1.3 Problem Classification

In Figure 1.3 a classification of the types of acoustical problems appearing in engineering is given. The problems can be grouped using different criteria based on problem characteristics such as the domain of interest and/or information available or required. These characteristics strongly determine which method is most suited to study the problem and the phenomena that play a role. In the case of *interior* acoustics the domain is bounded by solid surfaces which play an important role in the reflection of acoustical waves (reverberant sound). Typical applications are the acoustic design of rooms, theaters, or aircraft cabins.

Characteristic for *Exterior* acoustics is that the the domain of interest is unbounded and the sound waves are propagated to the infinity or until they reach an object as in scattering cases.

Acoustical problems can also be classified by the type of input used and the information to be obtained using this input. *Forward* acoustics is refers to problems in which the source (e.g. vibration of a body) is known and the resulting sound field is unknown that can be measured or evaluated. This is important in the prediction of sound generated by machines or devices, as well as in the prediction of noise pollution. *Inverse* acoustical problems relate to the case that the disturbance of the field is known from e.g. measurements and the question to be answered is which (part of) a structure or machine is the major source of the noise. Both types of problems appear in vibro, aero, and thermal acoustics and generally numerical simulations based on a mathematical model are used. Of the two types of problems the inverse problem is the most difficult one to solve. Yet, when the objective is to use the models and computational algorithms as tools for quiet design or as diagnostic tools it is exactly the solution of the inverse problem that is most needed, namely given a product or device, identify the source of the noise. Examples of the development and use of computational algorithms in vibro acoustics are [74] and [63].

In many practical situations vibro- and aero-acoustics problems appear together, see [51], [7] and [58]. These coupled problems are generally very demanding to solve.

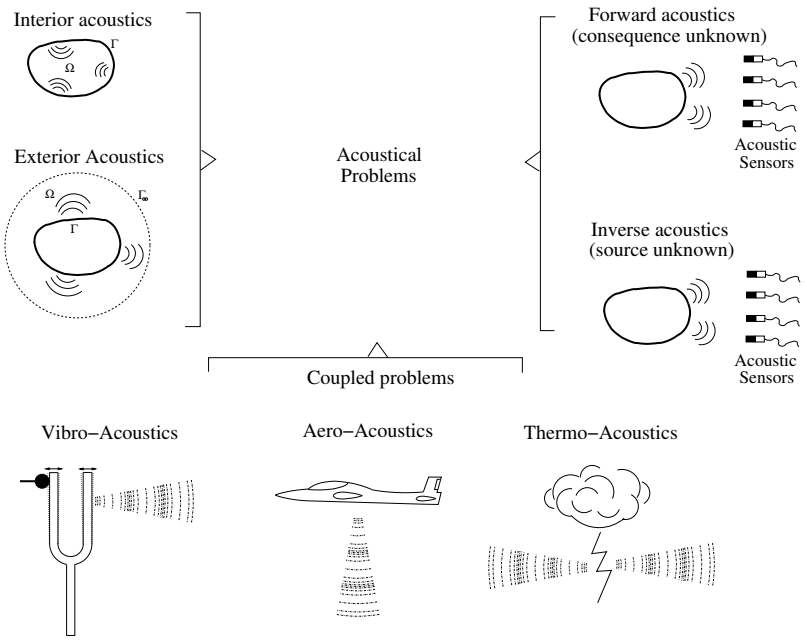


FIGURE 1.3: *A possible clasification of acoustical problems found in engineering applications.*

1.4 Numerical methods

In this thesis we restrict ourselves to vibro-acoustic problems and it is assumed that a model is used based on the Helmholtz equation with the associated boundary conditions, see chapter 2. Even then, only in much simplified cases the problem can be solved analytically. In general a numerical approach is necessary. During past decades different approaches have been proposed (see e.g. [22], [30], [41], [42], [46], [51] and [76]). Two approaches have become very common:

Firstly, the so-called *Finite Element Method* (FEM). In this approach the domain is covered with a mesh of volume elements. On each element of the mesh the field variables are approximated by interpolation polynomials using shape functions with a local basis. Subsequently using an equivalent (volume) integral formulation (weighted residual or variational) of the partial differential equation, an algebraic system of equation is constructed in terms of the values of the field variables at the nodal points of the mesh. This algebraic system of equations (usually a band matrix) is then solved using standard numerical procedures. The accuracy of the approximation depends on the order of the polynomials used and the volume element size in relation to the length and time scales of the acoustical field. Accurate approximation requires small elements. This applies particularly for cases in which the frequency is high, i.e. the wavelength of the phenomena to be resolved is small. Because of the use of volume elements, the Finite Element Method is most suited for the solution of problems in bounded domains, i.e. interior problems, see Figure 1.4.

The second approach widely used for the simulation and analysis of acoustic problems is

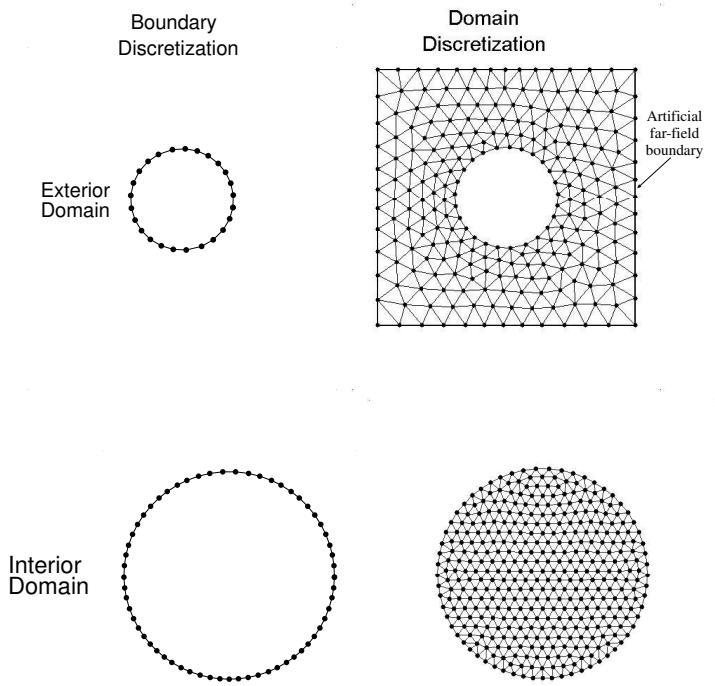


FIGURE 1.4: *Schematic representation of the discretization for a volumen and boundary domain.*

the *Boundary Element Method* (BEM), e.g. [22], [48], [43], and [44]). The method is based on the (direct or indirect) boundary integral formulation of the problem. Using elementary solutions of the governing partial differential equation and the boundary conditions the problem of solving the field variables in the domain is replaced by the solving an integral equation for the strength of the distribution of elementary solutions on the boundary of the domain, e.g. see [22]. Once this distribution is known, using the form of the elementary solution, the value of the field variables in any point in the domain can be obtained from an integral over the distribution on the boundary. Since the elementary solutions satisfy the boundary condition of infinity the boundary element method is very well suited for exterior problems. In the numerical approach, similar to the finite element method, in the boundary element method the problem of solving the continuous integral equation is replaced by a discrete formulation leading to an algebraic system that can be solved by computer. The way this is done is essentially the same as in the finite element method, i.e. the boundary is covered with a grid, see Figure 1.4. On each of the elements of the grid the boundary variable is approximated by an interpolation polynomial using shape functions with local support and using a variational or collocation formulation an algebraic system of equations is obtained that can be solved by standard numerical routines.

The Boundary Element Method is very well suited for problems on a large domain, i.e.

radiated sound problems. Instead of the need to cover the full domain with a mesh, only the boundary of the domain needs to be covered with a mesh. However, it also has some disadvantages. One difficulty is that it requires approximation of an integral equation with a singular kernel (Green's function). The second is that the resulting system of algebraic equations is generally a fully populated complex system which is very expensive to solve numerically. Typically the computing time will be $O(n^3)$, with n the number of unknowns (boundary elements). Finally, on top of that, to obtain the value of a field variable at point of the exterior domain a (discrete) integral transform has to be evaluated which requires a summation over all the boundary elements and thus $O(n)$ operations for each point at which the field is to be determined. This may lead to excessive computing times for large n or in case the field is to be computed at a large number of points. Often in practical problems indeed large n is required to have a sufficiently accurate approximation, particularly so for high frequencies.

At present impressive results can be obtained with both the Finite Element method as well as the Boundary Element method for acoustic problems. An illustration is given in Figure 1.5, Visser [74] used an inverse procedure for the determination of the source of annoying noise produced by a hairdryer. First, measurements were conducted in a range of 110-1100 Hz with a resolution of 5 Hz using microphone and microflown sensors. The data was used to determine the spectrum of sound pressure level (SPL) and particle velocity level (PVL). Subsequently, the measured data was supplied to a BEM program in order to reconstruct the source of the noise. From the results found it could be concluded that the sound radiation was produced by three different mechanisms clearly identified by the ranges of frequencies: air inlet of the hairdryer (low frequency), the structural vibration of the casing (medium frequency) and air outlet of the hairdryer (high frequency).

However, the computing time needed to solve the problem is often a serious bottleneck limiting the extent to which practical problems can be treated. Besides, often the problem must be solved multiple times for a range of frequencies. A fast turn-around time of the numerical computations is crucial in design processes. The continuous improvement in terms of hardware (faster computers) can only partly alleviate the computing time problems. Major steps forward can be made using more efficient computational algorithms.

1.5 Fast matrix multiplication

The need to numerically solve an integral equation or to evaluate an integral transform with a singular potential type or oscillatory Green's function arises in many fields in science. Its discrete version is also referred to as a multi-summation, discrete multi-integration, or evaluation of a discrete integral transform. The discrete task in fact is the multiplication of a full $n \times n$ matrix by a vector of n elements, which requires $O(n^2)$ operations. In this form the problem also appears in many body interaction problems in physics, e.g., to compute the long range Coulombic molecular interaction forces. If the matrix has arbitrary entries there is no fast way to carry out the multiplication. However, in many cases the matrix entries represent potential type influence coefficients with a value that decreases logarithmically or as $1/r$ with increasing distance r from the diagonal. Several methods have been proposed that, using different properties of the influence coefficients, enable a faster evaluation, e.g. Fast Fourier Transform (FFT)[24], the Fast Multipole Method (FMM)[36], and the Multi-Level

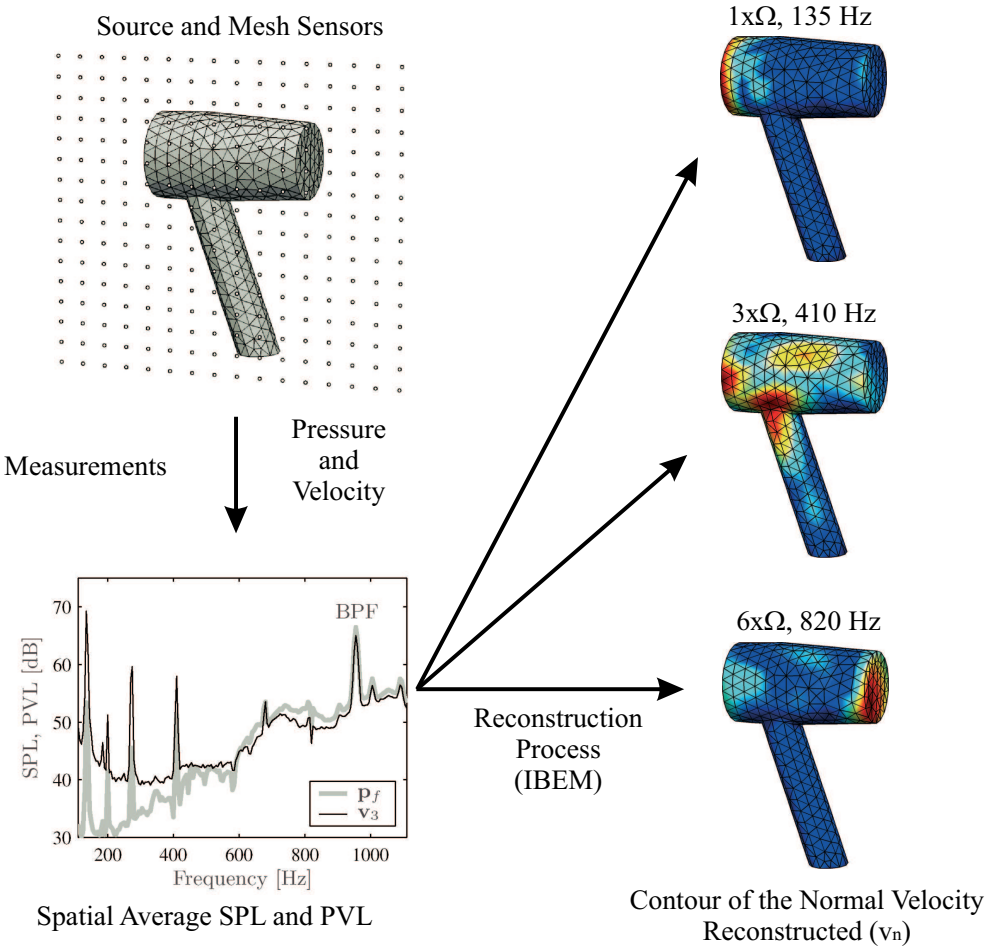


FIGURE 1.5: *Example of source sound reconstruction from an hairdryer reported by Visser [74]. (Graph and pictures reproduced with the permission of the author)*

Multi-Integration (MLMI) algorithm [16]. In most cases the fast evaluation is achieved at the expense of a certain controllable error. This is no restriction as in general since there is no need to exactly evaluate the transform as the matrix-vector multiplication by itself is an approximation to the continuous analytical result upto an error determined by the discretization. Any approximation to the discrete summation obtained with an error small compared to this discretization error is as good an approximation as the "exact" discrete transform itself.

Each of the algorithms has its own merits and uses a different approach to reduce the number of operations needed in the evaluation of the discrete transform or matrix-vector multiplication. For example; The Fast Fourier Transform uses the algebraic properties of the Fourier transform to construct a sparse factorization of the elements of the discrete Fourier transform which is ordered in a convenient way such that the number of operations in the matrix multiplication can be reduced from $O(n^2)$ to $O(n \log n)$ operations. Without doubt it

is one of the simplest and most efficient algorithms. However, its application is also limited to uniform non-curvilinear (Cartesian) grids.

The Fast Multipole Method uses a tree type of scheme and a set of polynomial expansions to represent the far-field and intermediate influence of the elements or particles. The central strategy used is that of clustering at various spatial lenghts and the compution of the interaction with other clusters which are sufficiently far away by means of multipole expansions. For elements in the near field, this assumption is no longer valid, so that, a direct computation is performed only in a small region. Using the Fast Multipole Method at the expense of an error controlled by the accuracy of the expansions and the size of the near field the computing time needed can be reduced from $O(n^2)$ to $O(n \log n)$ operations.

Multilevel Multi-Integration was introduced by Brandt and Lubrecht [16]. In this algorithm the smoothness properties of the Green's functions are exploited, this also determine the properties of the dense matrix. The key is that in regions where the (discrete) Green's function is smooth it can be represented accurately by interpolation from a reduced data set of the values of the Green's functions at a limited number of points. As a result the matrix-vector evaluation can be replaced by an equivalent matrix-vector multiplication involving the reduced data set and some transfer operators from the original to the reduced data set and back from the reduced problem to the full problem. For smooth Green's functions it allows a computing time reduction from $O(n^2)$ to $O(n)$. For singular smooth Green's functions computing time is reduced from $O(n^2)$ to $O(n \log n)$. An advantage of Multilevel Multi-Integration is that in its basic form it is straightforward to implement. Besides, it can be applied to any dense matrix vector multiplication without the need to make any assumption about its behavior. Also it can be applied to curved surfaces and extended to non-uniform grids. Besides, any error made in the fast evaluation can be corrected employing an a posteriori correction.

These methods have been applied successfully for particle interactions and for integral transforms related to problems governed by the Laplace equations for which Green's functions are asymptotically smooth (e.g. $\log|r|$, $1/r$). Computing time reductions have been reported up to orders of magnitude, see [36], [16], [50], [60], [70] and [13]. As a result of the reduced computing times larger and more practical problems could be considered leading to significant advances in the fields.

Numerical studies of boundary integral problems based on the Helmholtz equation can also significantly benefit from faster and more efficient algorithms. The implementation of more efficient techniques for such problems, for which the Green's function appearing in the transform is oscillatory, has indeed begun. In [50] has used the panel method for the discretization of the body surfaces and developed an approach based on the idea to represent the long-range part of the potential by point charges located on an uniform grid, which then allows fast evaluation using FFT, whereas the short-range interaction is still computed directly. The approach consists of two steps. First a transform is carried out of the boundary variable from the non-uniform curved boundary to a uniform non-curved grid. The second step is FFT to compute the integral transform represented on the grid. The final step is to interpolate the computed values of the discrete integral transform on the grid back to the non-uniform curved boundary followed by a local correction. Compared to simple direct summation to obtain the result large computing time reductions have been achieved.

The fast multipole method developed by Rokhlin and co-workers [60] has been applied to a variety of problems, including cases with oscillatory kernels [23], [27], [43] and [66]. Recently Drave [28] has given a set of guidelines for the numerical implementation of the

multipole method based on a physical interpretation of the method, which can be seen as a superposition of plane waves. In the method it is asssumed that a simple radiating source point $R(P)$ is located at the origin, and an observation point P is located in the field. When P is far enough from the origin R can locally be approximated by a single plane wave, with direction of propagation $P/|P|$. As P gets closer to the origin $R(P)$ will no longer be approximated well by a single plane wave but rather by a superposition of plane waves. Depending on the accuracy desired, the number of terms in the expansion needs to be increased, and the method can evaluate problems in $O(n^{3/2})$, $O(n^{4/3})$ or even $O(n \log n)$ operations.

The multilevel multi-integration method has not extensively been used for Helmholtz related boundary integral problems yet. Recently Grigoriev and Dargush [37] presented results for a two-dimensional boundary element method program based on the direct formulation. For the fast evaluation of the integral transforms related with the distributions on the boundary surface (contour in 2D) they used the standard multilevel multi-integration algorithm of [16]. Moreover, they implemented a conjugate gradient scheme for solving the discretized integral equation. The results confirmed the expectation that a significant speed up was attainable using the multilevel algorithm. However, their algorithm was only limited to moderate wave numbers. As mentioned above the Multilevel Multi-Integration algorithm exploits the smoothness properties of the Green's function. For high wavenumbers the oscillatory Green's function appearing in acoustic problems is obviously not smooth and when using the standard MLMI algorithm the attainable improvement in efficiency will strongly depend on the frequency. The generalization of the Multilevel Multi-Integration algorithm to the case of oscillatory kernels was already described in 1991 by Brandt [14]. It is claimed that independent of the frequency the computational effort of the discrete evaluation can be reduced from $O(n^2)$ to $O(np^d \log n)$ operations with p the order of polynomial and d the dimension of the problem (e.g. $d = 3$ for 3D). However, as far as known to the author no evidence that the algorithm really works has been published so far 2D.

1.6 Objective

The objective of the research presented in this thesis is the development and implementation of a Multilevel Multi-Integration algorithm along the lines suggested by Brandt in [14] for the fast evaluation of discrete integral transforms with oscillatory kernels as they appear in Boundary Element formulations of acoustic radiation problems. Eventually the algorithm should be combined with the Boundary Element Method proposed by [74] for source identification to alleviate the computing time problems and enable larger and more realistic problems to be solved. The project has been part of a larger project with the aim to develop efficient numerical and experimental tools for the determination of radiated noise and acoustic source identification.

1.7 Outline

In the chapter 2 a brief overview of relevant theory is given. This includes the derivation of the Helmholtz equation. It is tranformed in its integral version using the Green theorem. The more common boundary conditions are described and the fundamental solutions for acoustical problems are presented. Furthermore the chapter briefly describes the type of integral

transforms to be evaluated.

In chapter 3 the Multilevel Multi-Integration algorithm is explained taking a generic form of the integral transform as model problem. First the basic algorithm for smooth and asymptotically smooth algorithms is described. Subsequently the generalization of the algorithm to the case of oscillatory kernels is outlined.

In chapter 4 numerical results are presented for a number of representative one and two dimensional problems. It is illustrated that the generalized Multilevel Multi-Integration algorithm as proposed by Brandt in [14] really works and for large n yields substantial computing time reductions. For reference a comparison with results obtained using Fast Fourier Transform is given. Further improvement is even possible by further optimization

In chapter 5 results of the implementation of the algorithm for some acoustic problems analysed with a Boundary Element Method are given.

The thesis is concluded with concluding remarks and recommendations for further research.

FUNDAMENTAL SOLUTION FOR ACOUSTICS

CHAPTER 2

In this chapter a brief description is given of the theory related to acoustics. It starts with the linearization of the conservation equations of mass, momentum and energy. A general representation of the wave equation for acoustics is shown. Further, the more fundamental solutions (Green's function) of the wave equation is described and the integral formulation suitable for acoustic radiation problems is given. Attention is given to the treatment of the fundamental solutions for infinite domains and the conditions which have to be satisfied. Finally, the complications arising from the need to evaluate the resulting integrals in practical applications are discussed.

2.1 Conservation equations

In the continuum theory of fluids, a fluid (liquid or gas) is regarded as a continuous relatively dense distribution of particles, with uniform properties. Conservation laws are derived which describe the changes in space and time of the field variables: velocity, pressure, density, temperature, etc., of the fluid. The physical principles of mass, momentum and energy conservation are the starting point in the derivation of equations that describe acoustic phenomena. Their derivation can be found in standard text books, see [4] and [6]. A specific differential formulation of the basic conservation laws is:

- Mass conservation.

$$\frac{D\rho}{Dt} + \rho\nabla \cdot \mathbf{v} = 0, \tag{2.1}$$

- Momentum conservation, for constant μ and λ and neglecting any external force filed.

$$\rho\frac{D\mathbf{v}}{Dt} + \nabla p = (\lambda + 2\mu)\nabla(\nabla \cdot \mathbf{v}) - \mu\nabla \times (\nabla \times \mathbf{v}), \tag{2.2}$$

- Energy conservation, neglecting volumetric heat sources (adiabatic) and employing Fourier's low of heat conduction ($\mathbf{q} = -k\nabla T$.

$$\rho\frac{De}{Dt} + p\nabla \cdot \mathbf{v} = \Phi^{visc} + \kappa\nabla^2 T, \tag{2.3}$$

where $\frac{D}{Dt}$ represents the material derivative*, and the field variables of the fluid the velocity ($\mathbf{v}$), density (ρ), pressure (p), temperature (T) and internal energy (e). The term Φ^{visc} is defined as the three-dimensional viscous energy dissipation function (a nonlinear quantity) †. Alternatively the energy can be expressed as:

$$\rho\frac{Ds}{Dt} = \frac{1}{T}\left[\Phi^{visc} + \kappa\nabla^2 T\right] \tag{2.4}$$

The equations of conservation of mass, momentum and energy and the used relations for the viscous stress terms and the heat flux are not a completely closed system of equations. Additional relationships are required. These additional relations are the equations of state between thermodynamic variables. They follow from the law of thermodynamics an the thermodynamics state principle: *if the chemical composition of a fluid is fixed then the local thermodynamic state is fixed completely by two independent variables*. So one could choose the density ρ and the specific entropy.

$$T = T(s, \rho) \tag{2.5}$$

$$p = p(s, \rho) \tag{2.6}$$

Depending of the specific application the equations, (2.1)-(2.3), and (2.6), can be simplified [7]:

1. Incompressible flow. $\frac{D\rho}{Dt} = 0$. In this case the equation of continuity reduces to $\nabla\cdot\mathbf{v} = \mathbf{0}$, which means that also several terms in the other equations vanish. (Incompressible flows are of interest in fields as hydraulics, civil engineering, low-speed aerodynamics, etc.)

2. Time-independent or steady state flow. Here a considerable simplification occurs because the time derivative terms vanish. (Steady flows are of interest in aerodynamics, hydraulics, pipe flow, etc.)

3. Lossless flow. When effects of viscosity and heat conduction can be neglected (λ, μ, $\kappa = 0$), also many terms vanish. Important fluid flow problems, including sound propagation, can be approximated as lossless flow.

4. Small-perturbation flow. Linearization of the equations is helpful to deal with many kind of problems. Most sound waves are considered as small disturbances onto a main flow field.

As has been mentioned under 3) and 4), lossless and small perturbation flow are the main assumptions applied in the study of acoustics. Of course, effects of viscosity and heat conduction on acoustic waves can be taken into account for small perturbations in the conservation laws. However, this is beyond the scope of this thesis.

*The material derivative operator is defined by: $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v}\cdot\nabla$

†Φ^{visc} has been defined by some authors in tensor form as: $\Phi^{visc} = 2\mu d_{ij}d_{ji} + \lambda d_{kk}d_{ii}$, where the rate of deformation is $d_{ij} = \frac{1}{2}(v_{i,j} + v_{j,i})$, μ is the dynamic viscosity of the fluid, and λ the viscosity dilatation coefficient.

2.2 Acoustic wave equation

Acoustic disturbances can often be regarded as small-amplitude perturbations to an ambient state [51]. In that case the perturbations can be solved from a linearization of the conservation equations. The justification is that most acoustic disturbances are so small that the nonlinear terms in the conservation equations are not important [7]. Nonetheless, the exact form of the resulting wave equation depends on the assumptions made about the nature of the wave motion and the medium. Blackstock [7] and Pierce [51] give a linearization for the case a viscous and thermally conducting fluid is considered. Here the simplest and most common acoustic linearization is dicussed which applies for problems for which the medium may be characterized as inviscid and thermally nonconducting. In that case the continuity, momentum and energy equations can be written as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{2.7}$$

$$\rho \left[\frac{D\mathbf{v}}{Dt} \right] + \nabla p = 0 \tag{2.8}$$

$$\rho \frac{Ds}{Dt} = 0 \tag{2.9}$$

where (2.8) is known as Euler's equation of inviscid motion. Note that the energy equation indicates that for adiabatic flow of a inviscid non-heat-conducting fluid the entropy remains constant when moving with a fluid element.

2.2.1 The equation of state

For any fluid the equation of state relates physical quantities describing the thermodynamic behaviour of the fluid. For an adiabatic and reversible process (nearly isentropic) it is preferable to determine experimentally the relationship between pressure and density fluctuations. The relationship can be represented by a Taylor series expansion

$$p - p_0 = \left(\frac{\partial p}{\partial \rho} \right)_0 (\rho - \rho_0) + \frac{1}{2} \left(\frac{\partial^2 p}{\partial \rho^2} \right)_0 (\rho - \rho_0)^2 + ... \tag{2.10}$$

with the partial derivatives determined for isentropic compression and expansion of the fluid about its equilibrium density. If the fluctuations are small (assumption for linear acoustics) only the lowest order terms in $(\rho - \rho_0)$ need be retained, which gives a linear relationship between the pressure fluctuations and the change in the density.

$$p - p_0 \approx \beta \frac{(\rho - \rho_0)}{\rho_0} \tag{2.11}$$

where $\beta = \rho_0 (\frac{\partial p}{\partial \rho})_s$ is the so-called adiabatic bulk modulus[‡] and $(\rho - \rho_0)/\rho_0$ is known as the condensation term [41].

[‡]The adiabatic bulk modulus is defined as the change of the pressure as a function of the volume at constant entropy $\left(\beta = \rho \left(\frac{\partial p}{\partial \rho} \right)_s \right)$, and it has a direct relation with the speed of sound c.

2.2.2 The linear wave equation

A fluid is considered at rest when the field variables; pressure ($p = p_0$), density ($\rho = \rho_0$) and velocity ($\mathbf{v} = 0$) are absent of any perturbation. Furthermore, a medium is considered homogeneous when all ambient quantities are independent of the position, and when the medium is quiescent, they are independent of the time. In many cases, the idealization of a homogeneous quiesent medium is satisfactory for the description of acoustic phenomena [51].

The field variables of a fluid at rest satisfy the fluid dynamics equations (or conservation laws). However, when the fluid is disturbed for an acoustical perturbation, these variables can be represented by,

$$p(\mathbf{x},t) = p_0 + p'(\mathbf{x},t), \quad \rho(\mathbf{x},t) = \rho_0 + \rho'(\mathbf{x},t), \quad \mathbf{v}(\mathbf{x},t) = \mathbf{v}'(\mathbf{x},t), \tag{2.12}$$

where p', ρ' and $\mathbf{v}'$, all a function of space and time represent the acoustic perturbation of the fluid.

The mass conservation equation (2.7), the Euler equation (2.8), and the state equation with constant entropy ($p = p(\rho, s)$ and $s = s_0$) can be written in terms of the acoustic disturbances. Here $\mathbf{v} = \mathbf{v}'$; p_0 and ρ_0 are constants related by $p_0 = p(\rho_0, s_0)$.

$$\frac{\partial}{\partial t}(\rho_0 + \rho') + \nabla \cdot [(\rho_0 + \rho')\mathbf{v}'] = 0 \tag{2.13}$$

$$(\rho_0 + \rho')\left(\frac{\partial}{\partial t} + \mathbf{v}' \cdot \nabla\right)\mathbf{v}' = -\nabla(p_0 + p') \tag{2.14}$$

$$p_0 + p' = p(\rho_0 + \rho', s_0) \tag{2.15}$$

The terms in the equations (2.13)-(2.15) can be grouped into zero, first, and higher-order terms. In Equation (2.15), the grouping resulting from a Taylor-series expansion for small ρ' is,

$$p' = \left(\frac{\partial p}{\partial \rho}\right)_{\rho_0, s_0} \rho' + \frac{1}{2}\left(\frac{\partial^2 p}{\partial \rho^2}\right)_{\rho_0, s_0} (\rho')^2 + ... \tag{2.16}$$

Neglecting all terms of second order and higher results in the equations of linear acoustics in the case of a quiesent medium:

$$\frac{\partial \rho'}{\partial t} + \rho_0 \nabla \cdot \mathbf{v}' = 0 \tag{2.17}$$

$$\rho_0 \frac{\partial \mathbf{v}'}{\partial t} = -\nabla p' \tag{2.18}$$

$$p' = c^2 \rho', \quad \text{with} \quad c^2 = \left(\frac{\partial p}{\partial \rho}\right)_{s_0} \tag{2.19}$$

Thermodynamic considerations require that c^2 always be positive. Here c is defined as the speed of sound in the medium. Of course, in general c can be a function of position and

of time. This variation can depend on the specific properties of the fluid (e.g. when the thermodynamic properties are not constant in the domain, the speed of sound will also not be constant). When equations (2.17) and (2.18) are combined to eliminate the dependency of the velocity from (2.17), the resulting equation is:

$$\frac{\partial^2 \rho'}{\partial t^2} - \nabla^2 p' = 0 \tag{2.20}$$

Substitution of (2.19) in (2.20) leads to the wave equation for the acoustic pressure:

$$\nabla^2 p' - \frac{1}{c^2}\frac{\partial^2 p'}{\partial t^2} = 0 \tag{2.21}$$

This is the most fundamental equation in acoustics. It describes the properties of a sound field in space and time.

From (2.17)-(2.19) we can also derive that $p'(\mathbf{x}, t)$ and $\mathbf{v}'(\mathbf{x}, t)$ also satisfy the wave equation.

If the solution of (2.21) is assumed to be time harmonic; e.g.,

$$p'(\mathbf{x}, t) = \hat{p}(\mathbf{x})e^{i\omega t} \tag{2.22}$$

with ω the angular frequency ($\omega = 2\pi f$) and p the amplitude, equation (2.21) can be transformed from the time domain to the frequency domain. The resulting equation is known as the Helmholtz partial differential equation.

$$\nabla^2 \hat{p} + k^2 \hat{p} = 0 \tag{2.23}$$

where $k = \omega/c$, is known as the wave number. Assuming that $\mathbf{v}'$ and ρ' are also time harmonic, with the same frequency, also results in a Helmholtz equation for $\hat{\mathbf{v}}(\mathbf{x})$ and $\hat{\rho}(\mathbf{x})$. Furthermore, it follows directly from (2.18) that

$$\hat{\mathbf{v}}(\mathbf{x}) = \frac{i}{\rho_0 \omega}\nabla \hat{p} \tag{2.24}$$

and from (2.19): $\hat{\rho}(\mathbf{x}) = \frac{1}{c^2}\hat{p}(\mathbf{x})$. In the successive we drop the ($\hat{}$) on the variables.

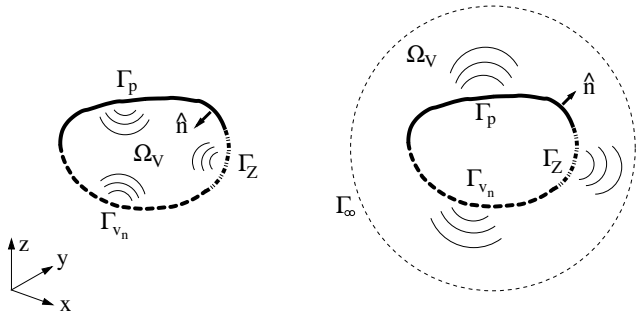
2.2.3 Boundary conditions

To determine the pressure field solving Helmholtz's equation (2.23) for domain Ω_V (see Figure 2.1), boundary conditions Γ_S must be specified at each position on the boundary $S = \partial\Omega_V$ of the domain. Depending on the type of fluid domain (bounded or unbounded), the following boundary conditions may occur.

Interior acoustic problem. The fluid domain is bounded by the surface S, and boundary conditions of the following three types may occur on the closed boundary surface $\Gamma_S = \Gamma_p \cup \Gamma_{v_n} \cup \Gamma_Z$:

- Dirichlet condition (imposed pressure)

$$p = p_S \quad \text{on} \quad \Gamma_p \tag{2.25}$$



a) Interior acoustics b) Exterior acoustics

FIGURE 2.1: *Sketch of the type of boundary conditions that appear in a) interior and b) exterior acoustical problems.*

- Neumann condition (imposed normal velocity)

$$\mathbf{v} \cdot \mathbf{n} = v_n \quad \text{on} \quad \Gamma_{v_n} = v_n \tag{2.26}$$

  which with (2.24) can be expressed as: $\frac{i}{\rho_0 \omega} \frac{\partial p}{\partial n}$

- Robin condition (imposed normal impedance)

$$p = Z v_n \quad \text{on} \quad \Gamma_Z \tag{2.27}$$

Exterior acoustic problem. The fluid domain Ω_V is unbounded and the same type of boundary conditions may apply as have been defined in (2.25)-(2.27) at the closed surface Γ_S of a vibrating body (In radiation problems a Neumann boundary condition is commonly used because data associated with the vibration is specified on the surface S)[63]. However, in addition to these boundary conditions a Sommerfeld radiation condition must be satisfied at the boundary surface Γ_∞ located at infinity, which ensures that the acoustic waves are propagated freely towards infinity and reflections at the far field boundary can not occur. In terms of the time-harmonic pressure perturbation, this can be expressed

$$\lim_{|\mathbf{r}-\mathbf{r}_S| \to \infty} |\mathbf{r} - \mathbf{r}_S| \left(\frac{\partial p}{\partial |\mathbf{r} - \mathbf{r}_S|} + ikp \right) = 0 \tag{2.28}$$

where $\mathbf{r}$ is the position of any point in the space of the domain Ω_V, and $\mathbf{r}_S$ is the position of any point on the surface S of the body (see Figure 2.2).

In practice a combination of these conditions can be used to solve different problems that appear in applications; for example, in the case of cavities (open boundary surface) or a combination of interior/exterior problems. However, the physical meaning of these conditions is always the same.

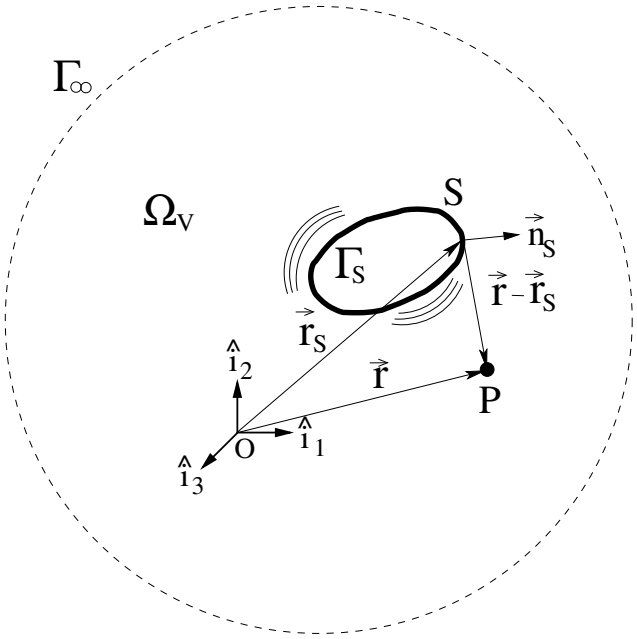


FIGURE 2.2: *Graphical representation of the point position in the radiation acoustical problems.*

2.3 The Helmholtz integral representation

A surface integral version of the Helmholtz equation (2.23) can be derived using Green's theorem (second identity)[§]. Consider a domain bounded by the surfaces S of a vibrating body and Γ_∞ being the border of the domain at infinity (see Figure 2.2). Now assume that $G(\mathbf{r}, \mathbf{r}_S)$ is a function that satisfies the homogeneous Helmholtz equation (2.23) for any point in the domain V and which is non-singular since the position $\mathbf{r}$ of the point P is located at different positions than $\mathbf{r}_S$, then applying the second identity of the Green theorem yields [51]:

$$c(p)p(\mathbf{r}) = \int_S \left(p(\mathbf{r}_S)\frac{\partial G(\mathbf{r}, \mathbf{r}_S)}{\partial \mathbf{n}_S} - G(\mathbf{r}, \mathbf{r}_S)\frac{\partial p(\mathbf{r}_S)}{\partial \mathbf{n}_S} \right) dS \tag{2.29}$$

In order to solve the Helmholtz integral equation (2.29) boundary conditions on S must be included. For this propose (2.26) can be used, which relate the pressure gradient and the normal velocity on the surface. Generally, in practical applications the normal velocity is known as it is associated with the vibration of the surface S. This formulation is the *direct* integral formulation due to the direct relation between the normal velocity and the acoustical pressure. Furthermore, since pressure and normal velocity are not independent, both can not

[§]For two functions φ and ψ, which are sufficiently smooth and non-singular in the domain V enclosed by a surface $S = \partial V$, Green's theorem (second identity) states that: $\int_S \left(\varphi \frac{\partial \psi}{\partial n} - \psi \frac{\partial \varphi}{\partial n} \right) dS = \int_V \left(\varphi \nabla^2 \psi - \psi \nabla^2 \varphi \right) dV$. The normal n to the surface S has positive orientation away from the domain V.

be prescribed on the surface S; this means that a prescribed normal velocity on the surface S causes a certain pressure on S, and vice versa [63].

The contribution from the far field boundary (Γ_∞) of the acoustic domain has been removed analytically by invoking the Sommerfeld radiation condition (2.28) which is automatically satisfied in the sense that the surface normal vector always points out of the domain. In this way, the characteristic impedance ($p/v_n = -\rho c$, when $|\mathbf{r} - \mathbf{r}_s| \to \infty$) is negative, which ensures that no reflecting waves occur. Equation (2.29) can be written as:

$$c(p)p(\mathbf{r}) = \int_S \left(p(\mathbf{r}_S) \frac{\partial G(\mathbf{r}, \mathbf{r}_S)}{\partial \mathbf{n}_S} + i\rho_0 \omega G(\mathbf{r}, \mathbf{r}_S) v_{\mathbf{n}_S} \right) dS \tag{2.30}$$

In (2.29) and (2.30) $c(p)$ is known as the geometry coefficient, which can be represented by the integral form [74]:

- When the normal vector on Γ_S is uniquely defined.

$$c(p) = \begin{cases} 1 & \text{for } \mathbf{x} \in \Omega_V \\ \frac{1}{2} & \text{for } \mathbf{x} \in \Gamma_S \\ 0 & \text{for } \mathbf{x} \notin \Omega_V \end{cases} \tag{2.31}$$

- When the normal vector on Γ_S is not uniquely defined: e.g at corners and edges.
  - Interior problems:

$$c(p) = \frac{1}{4\pi} \int \frac{\partial}{\partial \mathbf{n}} \frac{1}{\mathbf{r}} dS \tag{2.32}$$

  - Exterior problems:

$$c(p) = 1 + \frac{1}{4\pi} \int \frac{\partial}{\partial \mathbf{n}} \frac{1}{\mathbf{r}} dS \tag{2.33}$$

2.3.1 The fundamental solution for acoustics

The kernel $G(\mathbf{r}, \mathbf{r}_S)$ in Green's identity is defined as the fundamental solution that satifies the inhomogeneous Helmholtz equation.

$$(\nabla^2 + k^2) G(|\mathbf{r} - \mathbf{r}_S|) = -\delta(\mathbf{r} - \mathbf{r}_S) \tag{2.34}$$

and the Sommerfeld condition (2.25). Here $\delta(\mathbf{r} - \mathbf{r}_S)$ is the Dirac delta function defined by the value of its integral when it is integrated over a volume V. The Green function represents the free-field acoustic pressure at a point $\mathbf{r}$ in the domain due the acoustic point source at $\mathbf{r}_S$.

The Green's function can be represented by:
Two-dimensional problem[¶].

$$G(\mathbf{r}, \mathbf{r}_S) = -\frac{i}{4} H_0^{(2)}(k|\mathbf{r} - \mathbf{r}_S|) \tag{2.35}$$

Three-dimensional problem

$$G(\mathbf{r}, \mathbf{r}_S) = \frac{e^{-ik|\mathbf{r} - \mathbf{r}_S|}}{4\pi|\mathbf{r} - \mathbf{r}_S|} \tag{2.36}$$

[¶]The Hankel function of order zero and of the second kind of a variable ϕ is defined by: $H_0^{(2)}(\phi) = J_0(\phi) - iY_0(\phi)$. Here $J_0(\phi)$ and $Y_0(\phi)$ are the zero-order Bessel functions of first and second kind, respectively[1].

Note that the Green's function is a complex valued function, which becomes singular when the distance $|\mathbf{r} - \mathbf{r_S}|$ between the field position and the source position becomes zero.

2.4 Evaluation of the Helmholtz Integral Equation

Inlcuding the Neumann boundary conditions in the Equation (2.30) and reordering in terms of known and unknown quantities. The integral equation that represents the pressure on the surface of an vibrating body is given by:

$$c(p)p(\mathbf{r}) - \int_S p(\mathbf{r}_S)\frac{\partial G(\mathbf{r}, \mathbf{r}_S)}{\partial \mathbf{n}_S} dS = i\rho_0\omega \int_S G(\mathbf{r}, \mathbf{r}_S) v_{\mathbf{n}_S} dS \tag{2.37}$$

The evaluation of the direct Helmholtz integral equation (2.37) is generally performed in two steps [22][51][74]:

1. The pressure on the surface S must be evaluated solving (2.37) in order to introduce the boundary condition.

2. Once that the pressure is evaluated on the surface, the pressure in any point of the domain V is obtained be evaluationg (2.29).

In real applications it is generally not possible to obtain an exact solution of the integral equation for p (2.37), nor an exact evaluation of the integral in 2.29 and a numerical approximation has to be done.

The natural method for the numerical evaluation of the direct formulation of the Helmholtz integral equation is so-called *Boundary Element Method* (BEM) [63], which consists of the discretization of the integrals that appear in both steps above, and subsequently solving the system of equations by computer.

MULTI-LEVEL MULTI-INTEGRATION

CHAPTER 3

In this chapter Multilevel Multi-Integration (MLMI) is explained. This is an algorithm for the fast numerical evaluation of integral transforms as they appear in many fields in science, such as vision image analysis, contact mechanics, electromagnetics, and acoustics, etc. The complexity of the algorithm depends of the type of Green's function (kernel) in the transform.

First a description of the algorithm for integrals with smooth kernel is given. This case serves well to explain the basic principle of how to use the smoothness of the kernel to obtain a fast evaluation. In many real applications the kernel is asymptotically smooth. The extension of the algorithm to such cases only requires minor modifications and is explained next. Finally, the case of oscillatory kernels as they appear in acoustic and electromagnetic problems is discussed. Such kernels are by definition non-smooth. However, by introducing the concept of *separation of directions*, the task of evaluating a transform with oscillatory kernel can be rewritten as the task to evaluate a series of subtransforms each with an asymptotically smooth kernel which then facilitates fast evaluation.

3.1 Generic form of the task

The generic form of integral transforms appearing in problems as for example described by the Laplace or the Helmholtz equation is:

$$v(x) = \int_{\Omega} G(x,y)u(y)\,dy \quad x \in \bar{\Omega} \tag{3.1}$$

with,

$$G(x,y) = \begin{cases} G_{smo}(x,y) & \text{Smooth} \\ G_{asy}(x,y) & \text{Laplace} \\ G_{osc}(x,y) & \text{Helmholtz} \end{cases} \tag{3.2}$$

where $v(x)$ is the unknown function to be determined for each location of x ($x \in I\!R^d$; e.g. $x^1, x^2, ..., x^d$), where d is the dimension of the problem), $u(y)$ is a source function located at y ($y \in I\!R^d$; e.g. $y^1, y^2, ..., y^d$) which is generally related to the boundary conditions of the particular problem. $G(x,y)$ is the so-called kernel (Green's function) which is the fundamental solution of the governing equation. The kernel can be smooth, asymptotically

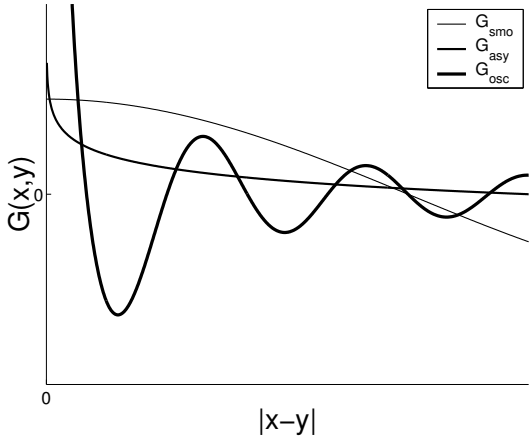


FIGURE 3.1: *Behaviour of the Green's function for the case of a smooth kernel (G_{smo}) and for problems governed by Laplace (G_{asy}) or Helmholtz (G_{osc}) equation.*

smooth or oscillatory as is illustrated in Figure 3.1. The evaluation of (3.1) can be part of the task to solve an integral equation such (2.37) in acoustical problems or it can be a task for itself as in the calculation of paticles interactions.
In practical problems it is generally not possible to analytically evaluate (3.1), so a numerical approximation has to be developed. Assume that $x_i = x_0 + ih$ are equidistant grid points of an evaluation grid defined on the domain $\bar{\Omega}$ where $x^d \in I\!R$ and $i = (i^1, i^2, ..., i^d)$, and h is the mesh size. In the same way an integration grid of points y_j is defined on the domain Ω. It is assumed that the value of u at the points y_j is given as $u^h(y_j)$. The domain Ω is subdivided in integration intervals. On each of these integration intervals the function u is approximated by a polynomial function $\hat{u}^h$ of degree $s-1$. The coefficients of the polynomial are obtained from requiring $\hat{u}^h(y_j) = u^h(y_j)$ at a set of points in or near the integration interval. The contribution of each integration interval to the integral transform at a point x is now approximated by the integration of the product $G(x,y)\hat{u}^h(y)$ over the interval. Summing up the contributions of all intervals yields a discrete approximation $v^h(x)$ to $v(x)$ to be evaluated for each point of the evaluation grid x_i:

$$v^h(x_i) \equiv \int_\Omega G(x,y)\hat{u}^h(y)dy = \sum_j G^{hh}(x_i, y_j)u^h(y_j) \tag{3.3}$$

In some cases the coefficients $G^{hh}(x,y)$ can be calculated analytically (for example; for the logarithmic kernel in problems of potential theory [16]). This is especially important for points near kernel singularities. Alternatively a numerical integration can be used, e.g. based on an adaptive quadrature rule [13]. The discretization error (τ) is $O(h^s||u^{(s)}||)$, where $||u^{(s)}||$ is an upper bound for the s-order derivative of u.

Note that (3.3) is the equivalent to the multiplication of a vector by a dense matrix. So, when there are n points x_i and $\bar{n}$ points y_j the evaluation of (3.3) for all x_i requires $O(n\bar{n})$ operations. Assuming $(n = \bar{n})$ this will lead to $O(n^2)$ operations. Since in practical problems often large n may be required for accuracy, this can lead to prohibitive computing times for

the evaluation of (3.3).

In [16] Brandt and Lubrecht have described Multilevel Multi-Integration as an alternative method for the fast evaluation of integral transforms and demonstrated its efficiency for smooth and asymptotically smooth kernels. They showed that the computational effort of the evaluation of the discretized integral transforms can be reduced from $O(n^2)$ to $O(n)$ for smooth kernels and to $O(n \log n)$ for asymptotically smooth kernels. The algorithm exploits the smoothness properties of the kernel and is explained below.

The integral transforms that appear in problems described by the Helmholtz equation are numerically more demanding to evaluate due to the oscillatory behaviour of the kernel (see Figure 3.2). For accuracy of representation the number of gridpoints per wavelength should be sufficiently large. Visser [74] and Schubmacher [63] state that at least 7 points per wavelenght (n_λ) are required to ensure that the phenomena can be adequately represented. So, with increasing wavenumber an increasing number of nodes is needed aggravating the problem of excessive computing times.

Due to the oscillatory behaviour the standard MLMI algorithm can only be used with full efficiency for cases with low and medium wave numbers. The general principle of a Multilevel Multi-Integration algorithm for the case of oscillatory kernels has been described by Brandt in [14]. It is claimed that the complexity of the numerical evaluation can be reduced from $O(n^2)$ to ($O(np^d \log n)$ operations, where p is the order of interpolation used in the method and d is the dimension of the problem (see below). In [14] no actual results were presented. Implementation and performance test of this algorithm has been one of the challenges of the research reported in this thesis.

3.2 Multilevel Multi-Integration: Basic Algorithm

In this section a short description is given of the multilevel multi-integration algorithm for the evaluation of integral transforms with smooth or asymptotically smooth kernels. For further details the reader is referred to [16]. The algorithm can also be used for the evaluation of integral transforms with oscillatory kernels as long as the wave number k is sufficiently small.

3.2.1 Notation

In the final algorithm to obtain the transform on a target grid with mesh size h a series of coarser grids is used recursively. However, for simplicity, the algorithm is explained using only two grids; i.e. a fine grid with mesh size h and an auxiliary coarse grid with mesh size H. For simplicity it can be assumed that $H = 2h$. The variables and indices on the fine grid will be represented by lower case characters, subscripts and superscripts; so that $u_j^h = u^h(y_j)$ stands for the value of u in grid point j on the fine grid h, at location $y_j = y_0 + jh$. The variables and the indices on the coarse grid will be represented by upper case letters such as V, U, I, J, etc; so that $U_J^H = U^H(Y_J)$ is the value of the function U at a coarse grid point J with location $Y_J = Y_0 + JH$. In some cases the dependence on the mesh size of variables is explicitly indicated for example when it concerns values of a variable at the same location at different grids. (i.e. $V_I^H = v_{2I}^h$, $Y_J^H = y_{2J}^h$, etc.).

In the algorithm operations occur between the grids h and H. For example to obtain the value of a variable at a location of the fine grid x_i^h from its values at coarse grid locations X_I^H

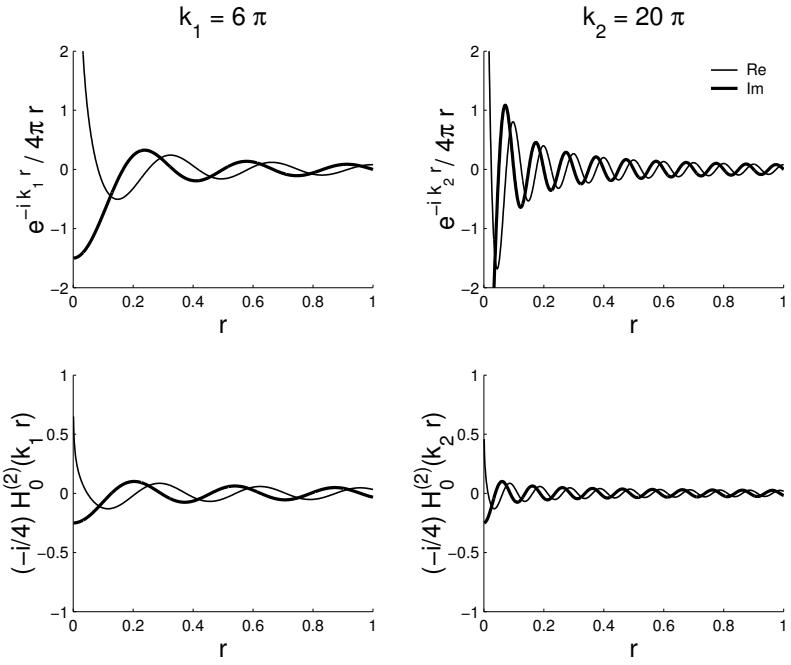


FIGURE 3.2: *Graphical representation of the behaviour of the Green functions for acoustical radiation problems (Eqs.(2.34) and 2.35).*

interpolation will be used. The interpolation coefficients are denoted by ω_{iI}^{hH}. In the same way ω_{jJ}^{hH} denotes the interpolation weight for the value of a variable in the point Y_J^H when interpolating to the fine grid location y_j^h. The order of the interpolation will be denoted as p. In this work only even orders of interpolation will be considered. (A formula to obtain these interpolation coefficients to different orders is given in Appendix A)

3.2.2 Smooth kernels

When $G(x, y)$ is by definition smooth as a function of y on the scale H its value at any point y_j can be obtained with $O(\epsilon)$ error using a $p = O(log(1/\epsilon))$ order interpolation from its values at points Y_J of a (coarse) grid, with mesh size H. In particular for any point y_j:

$$G^{hh}(x_i, y_j) = \sum_J \omega_{jJ}^{hH} G^{hH}(x_i, Y_J) + O(\epsilon) \tag{3.4}$$

where ϵ is the interpolation error ($\epsilon = (\gamma_1 H)^p |G^{(p)}|$). Here γ_1 is a constant in the range $1/2 \leq \gamma_1 \leq 1$ and depends on the kind of interpolation used ($\gamma_1 = 1/2$ for central interpolation), $|G^{(p)}|$ is the maximum of a p^{th} derivative of G. The summation over J in (3.4) usually extends over only p points around point y_j, so it is a local summation.
Substitution of (3.4) in (3.3) gives:

$$v^h(x_i) = \sum_j \sum_J \omega_{jJ}^{hH} G^{hH}(x_i, Y_J) u^h(y_j) + O(\epsilon) \tag{3.5}$$

Changing the order of summation yields:

$$v^h(x_i) = \sum_J G^{hH}(x_i, Y_J) U^H(Y_J) + O(\epsilon) \tag{3.6}$$

where

$$U^H(Y_J) = \sum_j \omega_{jJ}^{hH} u^h(y_j) \tag{3.7}$$

As the summation over j (for each J) in (3.5) is a local summation over $O(p)$ points also the summation over J for each j in (3.7) is also a local summation over $O(2p)$ points when $H = 2h$.

The change of order of summation going from (3.5) to (3.6) is the first crucial step in the algorithm. Its effect is that a *fine grid* summation over index j is replaced by a *coarse grid* summation over index J.

Summarizing, when $G^{hh}(x, y)$ is smooth as a function of y the computation of $v^h(x_i)$ by summation over all y_j at the expense of an $O(\epsilon)$ error can be replaced by a coarse grid summation (3.6) over all Y_J of "injected" values $G^{hh}(x_i, y_j)$ ($G^{hH}(x_i, Y_J) \equiv G^{hh}(x_i, y_{2J})$) multiplied with *collected charges* $U^H(Y_J)$ defined by (3.7). This latter operation is referred to as *anterpolation* of $u^h(y_j)$, since it is the *adjoint* of the *interpolation* of (3.4) [14].

Next, assume that $G^{hh}(x, y)$ is also a smooth function of x on the scale H. In that case for any x_i up to $O(\epsilon)$ error $G^{hh}(x_i, y)$ can be obtained by interpolation from its values on a coarse grid X_I with mesh size H.

$$G^{hh}(x_i, y_j) = \sum_I \omega_{iI}^{hH} G^{Hh}(X_I, y_j) + O(\bar{\epsilon}) \tag{3.8}$$

For a $\bar{p}$-order interpolation the error is defined by $\bar{\epsilon} = (\gamma_1 H)^{\bar{p}} |G^{(\bar{p})}|$. Often $\bar{p} = p$ can be used and $\bar{\epsilon} = \epsilon$. For each i the summation over I is a local summation involving only $\bar{p}$ terms. This is the second crucial step in the algorithm. This second step implies that it is not necessary to compute the transform in all points of the fine grid x_i. Permitting a certain error that can be controlled by the order of interpolation it is sufficient to only compute the summations in *coarse grid* points, and then obtain the value in all points of a finer grid by means of interpolation.

The result of both steps is that, up to $O(\epsilon)$ error, the computation of $v^h(x_i)$ in all evaluation grid points x_i involving a summation over all integration grid points y_j can be replaced by:

(i) *Anterpolation:*

$$U^H(Y_J) = \sum_j \omega_{jJ} u^h(y_j) \tag{3.9}$$

(ii) *Injection of the kernel:*

$$G^{HH}(X_I, Y_J) = G^{hh}(x_{2I}, y_{2J}) \tag{3.10}$$

(iii) *Coarse grid summation:*

$$V^H(X_I) = \sum_J G^{HH}(X_I, Y_J) U^H(Y_J) \tag{3.11}$$

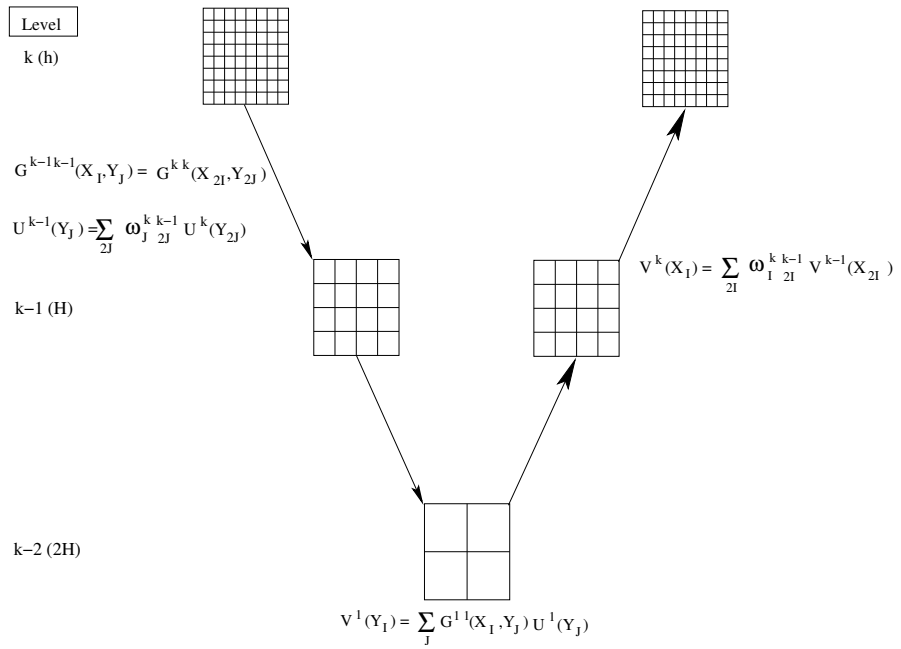


FIGURE 3.3: *Schematic representation of the Multilevel Multi-Integration algorithm for smooth kernels. Anterpolation (left): recursive operations are carried out until* $N = O(\sqrt{n})$ *is reached. Interpolation (right): recursive operations are carried out until* n *is reached.*

(iv) *Interpolation:*

$$v^h(x_i) = \sum_I \omega_{iI} V^H(X_I) \tag{3.12}$$

When the kernel is sufficiently smooth, the coarsening step from grid y_j to Y_J and x_i to X_I can be such that the evaluation of the coarse grid summation requires $O(n)$ operations. As the anterpolation and interpolation also require at most $O(n)$ operations, the total work needed to obtain the direct transform is reduced from $O(n^2)$ to $O(n)$ operations at the expense of an error $O(\epsilon)$. This error can be made small by the choice of the order p of interpolation, where $p = O(\log(1/\epsilon))$ is generally needed [70]. However, in practice it is only needed to ensure that the additional error introduced by the fast evaluation is small compared to the discretization error (τ) that is made anyway.

For convenience the algorithm can be programmed recursively using a sequence of grids with mesh size increasing by a factor 2 each coarsening. The grids are then referred to as levels and numbered. In that case the coarsening is repeated until a grid is reached with $N = O(\sqrt{n})$ points. A flow diagram of the algorithm is given in Figure 3.3. In the left side of the figure are represented the recursive operations of anterpolation for $u^h(y_j)$ and the injection of the kernel until the coarsest grid with $N = O(\sqrt{n})$ points is reached. In the right side of the figure is represented the recursive operation of interpolation for $V^H(X_I)$ from the coarsest grid to the target (finest) grid.

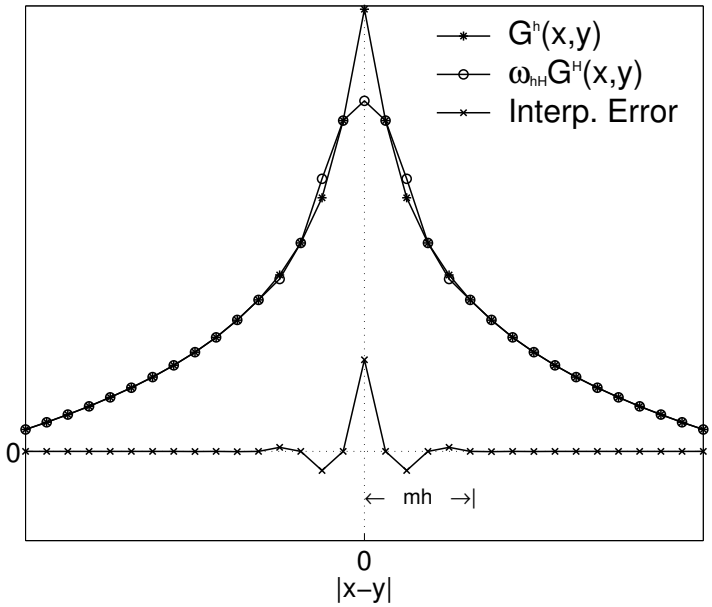


FIGURE 3.4: *A discrete representation of an asymptotically smooth kernel and the error introduced by interpolation. (Given as example:* $G(x,y) = -\log(|x-y|)$*)*

3.2.3 Asymptotically smooth kernels

In many practical problems the kernel is *asymptotically smooth* (or *singularly smooth*), i.e. it has a singularity at $x = y$. Obviously in such cases it is not possible to obtain the value of $G(x,y)$ at a given location x_i, y_j in the vicinity of the singularity from its values at coarse grid points X_I, Y_J by interpolation with sufficient accuracy (see Figure 3.1). However, with increasing distance from the singularity the smoothness increases and at a sufficient distance from the singularity again Equations (3.4) and (3.8) hold for an interpolation of $p = O(\log(1/\epsilon))$. The Multilevel Multi-Integration algorithm as described above can be adapted to the case of singular smooth kernels by introducing *posteriori* local corrections for the error made by the interpolation near the singularity. The effect of the interpolation on the result of the algorithm can be written explicitly as follows:

$$v^h(x_i) = \bar{v}^h(x_i) + \sum_j \left[G^{hh}(x_i, y_j) - \sum_I \sum_J \omega_{iI}^{hH} \omega_{jJ}^{hH} G^{HH}(X_I, Y_J) \right] u^h(y_j) \tag{3.13}$$

where $\bar{v}^h(x_i)$ is defined by

$$\bar{v}^h(x_i) = \sum_I \omega_{iI}^{hH} V^H(X_I) \tag{3.14}$$

and the summation terms in (3.13) represent the error introduced in (3.12) by (3.5) and (3.8).

This error satisfies,

$$G^{hh}(x_i, y_j) - \sum_I \sum_J \omega_{iI}^{hH} \omega_{jJ}^{hH} G^{HH}(X_I, Y_J) = \begin{cases} 0 & x^h = X^H \text{ and } y^h = Y^H \\ O(\epsilon) & |x^h - y^h| \geq mh \end{cases} \tag{3.15}$$

Choosing m sufficiently large the error due to the interpolation of the kernel (in both x and y) can be made small compared to the discretization error. The error introduced in the region $|x - y| \leq mh$ can be computed and added as a local correction, i.e. the summation over j in (3.13) extending over the range $|x - y| \leq mh$.

Summarizing, (3.12) up to an error $O(\epsilon)$ can be written as,

$$v^h(x_i) = \bar{v}^h(x_i) + \tilde{v}^h(x_i) \tag{3.16}$$

where the correction term $\tilde{v}^h(x_i)$ is defined by:

$$\tilde{v}^h(x_i) = \sum_{|i-j| \leq m} \left[G^{hh}(x_i, y_j) - \sum_I \sum_J \omega_{iI}^{hH} \omega_{jJ}^{hH} G^{HH}(X_I, Y_J) \right] u^h(y_j) \tag{3.17}$$

For a given interpolation the correction kernel coefficients needed, see (3.15), can be precomputed and stored. The total work of the algorithm now depends on m. The value of m can be determined from a work-accuracy optimization: Setting the permissible error equal to the discretization error τ find m such that the total work is minimized. An example of such an analysis and details for the logarithm kernel $G(x, y) = \log|x - y|$ are given in [16] and [17].

The Multilevel Multi-Integration algorithm for an asymptotically smooth kernel is thus given by (3.9)-(3.12) steps (i) to (iv) followed by:

(v) *Correction* (3.16):
$$v^h(x_i) = \bar{v}^h(x_i) + \tilde{v}^h(x_i)$$

with $\tilde{v}_h(x_i)$ given by (3.17).

Figure 3.5 shows a schematic of the algorithm for the evaluation of integral transforms with asymptotically smooth kernel. Note that when the algorithm is used with a series of grids the corrections are applied between each set of a coarser and a finer grid.

3.3 MLMI algorithm for Oscillatory kernels

As was shown above the standard MLMI approach exploits the smoothness properties of the kernel of (3.3) to reduce the complexity of its numerical evaluation. However, for integrals with oscillatory kernel, as they appear in the solution of problems governed by the Helmholtz equation, the smoothness depends on the value of the wavenumber. The algorithm as explained above can still be used but the degree to which the coarsening can be repeated and thereby the attainable reduction in complexity depends on the wavenumber. When the wavenumber is small full efficiency can be obtained. However, with increasing wavenumber the gain in computing time will be smaller as the coarsening cannot be repeated to a grid with $O(\sqrt{n})$ points. To efficiently evaluate integral transforms with an oscillatory kernel requires

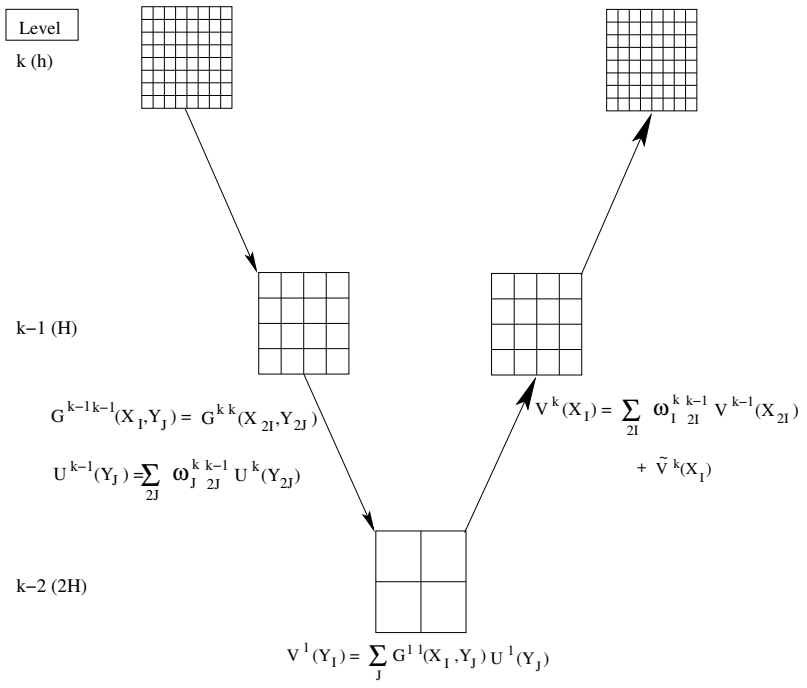


FIGURE 3.5: *Schematic of the Standard Multilevel Multi-Integration (MLMI) algorithm for the evaluation of integral transforms with asymptotically smooth kernels.*

a generalization of the algorithm. Assume the following generic form of the transform to be evaluated:

$$v(x) = \int_{\Omega} G_{asy}(x,y) e^{ik|x-y|} u(y) dy \qquad x \in \bar{\Omega} \tag{3.18}$$

In this generic form the oscillatory kernel is represented as the product of a smooth or an asymptotically smooth function $G_{asy}(x,y)$ and a complex exponential (oscillatory) function $e^{ik|x-y|}$.

The advantage of the representation (3.18) is that the oscillatory behaviour appears explicitly. In applied problems the oscillation may be implicit in the kernel, for example when it is the Hankel function. However, in such cases it can always be rewritten in the form (3.18). So assuming (3.18) does not imply any loss of generality. The generic form of the discrete integral transform (3.18) can now be written as:

$$v^h(x_i) = \sum_j G^{hh}(x_i, y_j) e^{ik|x_i - y_j|} u^h(y_j) \tag{3.19}$$

to be evaluated for all points x_i of an evaluation grid, given the value of u^h at points of an integration grid y_j. Note that the i preceding the wave number k always denotes the imaginary constant ($\sqrt{-1}$). It is not related to the grid index that appears elsewhere as a subscript.

Obviously for the case of an oscillatory kernel, the total kernel itself is not smooth. However, the argument of the oscillatory part is a smooth function of x and y. Using this smoothness is the key idea in the generalization of the Multilevel Multi-Integration algorithm to the case of oscillatory kernels.

Using the concept *separation of directions* the discrete transform is written as a series of discrete subtransforms for a number of predefined directions. These subtransforms can be rewritten in such a way that a kernel which is asymptotically smooth as a function of x and y appears. Its smoothness can then be used to obtain a fast evaluation.

3.3.1 Separation of directions

Equation (3.19) can be seen as a special case of a more general problem. This observation will lead to the fast evaluation algorithm. Let x_i be a point of the evaluation grid and a y_j a point of the integration grid where u^h is given. The direction "vector" between x_i and y_j is now determined by $e_{ij} = (y_j - x_i)/|y_j - x_i|$. Here $|x_i - y_j|e_{ij}$ denotes the vector from the *sender* point y_j to the *receiver* point x_i. Let θ_{ij} denote the angle of this vector, for example; in two dimensional problems $e_{ij} = \{\cos\theta_{ij}, \sin\theta_{ij}\}$. In (3.19) obviously the contribution of a point y_j to the transform at x_i no depends on the direction between x_i and y_j. However assume now that there were such an angular dependence and that it could be represented by defining an angular radiation filter $\mathcal{U}(\theta)$ and an angular reception filter $\mathcal{G}(\theta)$, such that the generalized version of (3.19) is:

$$v^h(x) = \sum_j G(x_i, y_j)\mathcal{G}(\theta_{ij})e^{ik|x_i - y_j|}\mathcal{U}(\theta_{ij})u^h(y_j) \tag{3.20}$$

The original task of evaluating (3.19) is exactly the same as equation (3.20) with $\mathcal{U}(\theta) \equiv 1$ and $\mathcal{G}(\theta) \equiv 1$. We now define a *grid of directions* $e_\beta \in \sigma^d = \{e \in I\!R^d : |e| = 1\}$ with σ^d the unit sphere (circle), and e_β the associated base vectors, e.g. $e_\beta = \{\cos\theta_\beta, \sin\theta_\beta\}$ for the two dimensional case (see Figure 3.6). It is assumed that the number of directions on the grid of directions is λ and the mesh size on this grid $\delta\theta = 2\pi/\lambda$.

Now assume that $\mathcal{U}$ and $\mathcal{G}$ are smooth as a function of θ. This is obviously true for our problem. For any function that is smooth on the scale of the mesh size on the grid of directions its value for a given direction θ can be obtained accurately by a p^{th}-order interpolation from its value at the directions on the grid. In particular,

$$\mathcal{U}(\theta) = \sum_{m \in s(e)} w_m^{s(e)}(e)\mathcal{U}(\theta_m) + O(\epsilon) \tag{3.21}$$

$$\mathcal{G}(\theta) = \sum_{l \in s(e)} w_l^{s(e)}(e)\mathcal{U}(\theta_l) + O(\epsilon) \tag{3.22}$$

where $w_l^{s(e)}(e)$ and $w_m^{s(e)}(e)$ represent the interpolation weights of the value at a given base vector l and m, respectively, into the result for a given e (see Appendix A). $s(e)$ denotes the set of p base directions involved in the p^{th}-order interpolation to the direction e. Preferably it should be chosen such that e is central to the set. Because of periodicity the interpolation is well defined even if the number of directions is smaller than p. Even for the case $\lambda = 1$ of a single direction it is well-defined as then $w_\lambda^{s(e)}(e) \equiv 1$. Finally, the error ϵ is clearly zero for the case of a constant function.

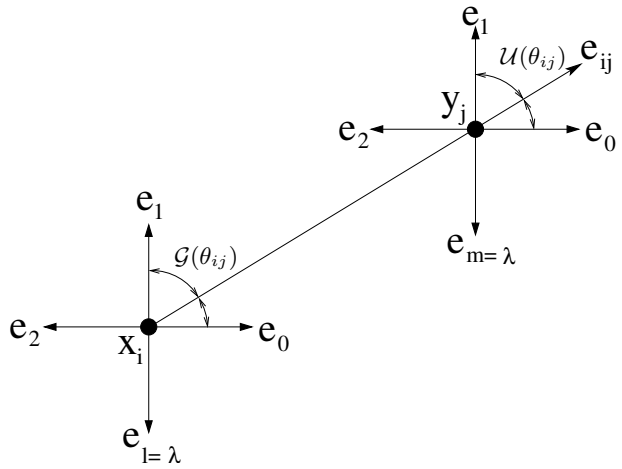


FIGURE 3.6: *Schematic representation of the grid of directions for* $u_m(y_j)$ *and* $v_l(x_i)$.

Substitution of (3.21) and (3.22) to determine the value of $\mathcal{G}(\theta_{ij})$ and $\mathcal{U}(\theta_{ij})$ in Equation (3.20) gives:

$$v^h(x_i) = \sum_j G(x_i, y_j) \sum_{l \in s_{ij}} w_l^{s_{ij}}(e_{ij}) \mathcal{G}(\theta_l) e^{ik|x_i - y_j|} \sum_{m \in s_{ij}} w_m^{s_{ij}}(e_{ij}) \mathcal{U}(\theta_m) u^h(y_j) \quad (3.23)$$

By defining

$$u_m^h(y_j) = \mathcal{U}(\theta_m) u^h(y_j) \quad (3.24)$$

and defining that $w_l^{s(e)}(e) = 0$ for $l \notin s(e)$ the original task of evaluation of (3.19) can now be written as the evaluation of:

$$v^h(x_i) = \sum_l \mathcal{G}(\theta_l) v_l^h(x_i) \quad (3.25)$$

with

$$v_l^h(x_i) = \sum_j G^{hh}(x_i, y_j) e^{ik|x_i - y_j|} w_l^{s_{ij}}(e_{ij}) \sum_{m \in s_{ij}} w_m^{s_{ij}}(e_{ij}) u_m^h(y_j) \quad (3.26)$$

for $1 \leq \lambda$. So the generalized task is now to the evaluate (3.26) for $l = 1, ..., \lambda$. Note that the original task (3.19) corresponds to the special case $\lambda = 1$.

3.3.2 One-dimensional scheme

To illustrate how the separation of directions introduced above can lead to a fast evaluation algorithm, first consider a one-dimensional case. The general representation (3.26) can now be written in the form of only two directions (positive and negative) from x_i.

$$e_{ij} = \begin{cases} +1 & y_j \geq x_i \\ -1 & y_j < x_i \end{cases} \tag{3.27}$$

The set of directions s_{ij} used in the interpolation to e_{ij} by definition consists of only one direction and the interpolation weights are given by:

$$w_l^{s_{ij}}(e_{ij}) = \begin{cases} 1 & e_{ij} = s_{ij} \\ 0 & e_{ij} \neq s_{ij} \end{cases} \quad \text{and} \quad w_m^{s_{ij}}(e_{ij}) = \begin{cases} 1 & e_{ij} = s_{ij} \\ 0 & e_{ij} \neq s_{ij} \end{cases} \tag{3.28}$$

Substituting (3.27) and (3.28) into (3.26), the generalization of the integral transform for one dimensional problems can be rewritten as:

$$v_l^h(x_i) = \begin{cases} \sum\limits_j G^{hh}(x_i, y_j) e^{ik|x_i - y_j|} u_m^h(y_j) & l = m \\ 0 & \text{otherwise} \end{cases} \tag{3.29}$$

Because (3.29) is defined for only two directions, we can introduce a simplified notation (+) and (-) to denote the directions that l and m can take: when $x - y \geq 0$ the direction is defined (+) when $x - y < 0$ it is defined (-). The function $e^{ik|x-y|}$ in (3.29) can be split in terms of x and y. The task (3.19) can now be rewritten as the computation of:

$$v^h(x_i) = e^{-ikx_i} v_+^h(x_i) + e^{ikx_i} v_-^h(x_i) \tag{3.30}$$

where,

$$v_+^h(x_i) = \sum_j G_+^{hh}(x_i, y_j) u_+^h(y_j) \tag{3.31}$$

$$v_-^h(x_i) = \sum_j G_-^{hh}(x_i, y_j) u_-^h(y_j), \tag{3.32}$$

with

$$G_+(x, y) = \begin{cases} G(x, y) & y \geq x, \\ 0 & y < x, \end{cases} \tag{3.33}$$

$$G_-(x, y) = \begin{cases} 0 & y > x, \\ G(x, y) & y \leq x, \end{cases} \tag{3.34}$$

and,

$$u_+^h(y_j) = e^{iky_j} u^h(y_j) \tag{3.35}$$

$$u_-^h(y_j) = e^{-iky_j} u^h(y_j) \tag{3.36}$$

The result is that the evaluation of the discrete transform with oscillatory kernel is now rewritten into the task of evaluating two subtransforms, the summations (3.31) and (3.32),

with each an asymptotically smooth kernel. Each of these summations can be evaluated separately using the algorithm described in Section 3.2.3, so that the evaluation of (3.19) can be carried out by the following steps:

(i) *Separate u^h in (+) and (-) directions*: (3.35) and (3.36)

(ii) *Perform the MLMI algorithm for (+) and (-) directions*: (3.31) and (3.32)

(iii) *Combine v^h_+ and v^h_- to v^h*: (3.30)

So, for a one dimensional problem the oscillatory part of the kernel can be fully incorporated in the function v and u and no special algorithm is needed for the fast evaluation. The total work involved in the fast evaluation for the oscillatory kernels is now twice the total work involved in the fast evaluation with a singular smooth kernel. So, for n points x_i and n points y_j the total work is $O(n \log n)$.

3.3.3 Two and Three-dimensional scheme

The one-dimensional case illustrated the principle of separation of directions very clearly. The separation of directions leads to 2 subtransforms which each have a smooth or singular smooth kernel. The principle for higher dimensional cases is the same. A series of subtranforms appears. However, the separation of directions cannot be made so explicit that the oscillatory behaviour in x can be taken outside the summation. As a result the higher dimensional algorithm appears much more complex. In this case for each combination of directions l, m a singular smooth kernel is obtained from the original kernel by multiplying it with an oscillatory function. The principle is as follows: Consider the function $e^{ik|x-y|}$ in the one-dimensional case and assume $x - y > 0$. When multiplied by $e^{-ikx}e^{iky}$, it can be written as $e^{ik(|x-y|-x+y)}e^{iky}e^{-ikx}$. Obviously the function $e^{ik(|x-y|-x+y)}$ for the region $(x - y) > 0$ is smooth in terms of x and y.

Now consider the two-dimensional case. The oscillatory term is written as:

$$e^{ik|x-y|} = e^{ik(|x-y|+e_l x-e_m y)}e^{-ike_l x}e^{ike_m y} \tag{3.37}$$

where e_m and e_l are a set of *base vectors* from the grid of directions. Substitution of (3.37) in (3.26) gives:

$$v^h_l(x_i) = \sum_j \sum_{m \in s_{ij}} G^{hh}_{ijlm}(x_i, y_j)e^{ik(e_m y_j - e_l x_i)}u^h_m(y_j) \tag{3.38}$$

with:

$$G_{ijlm}(x, y) = G(x, y)e^{ik(|x-y|+e_l x-e_m y)} \times w^{s_{ij}}_l(x, y)w^{s_{ij}}_m(x, y) \tag{3.39}$$

Now, for any combination of l and m, (3.39) is an asymptotically smooth function in terms of x and y. Here, the interpolation weights plays a crucial role too. For given directions m and l the exponential function is smooth in a region of the domain. The smoothness in the remaining part of the domain turns from the product of the interpolation weights $w^{s_{ij}}_l(x, y)w^{s_{ij}}_m(x, y)$ vanishing.

In the Figure 3.7 an illustration is given of the behaviour of the discrete function (3.39) for the two-dimensional case, fixed index $l = 0$ ($\theta_l = 0$) and $m = 0$ ($\theta_m = 0$), the variable x_i

a) $I\!Re(G_{ijlm}(x,y))$ b) $I\!Im(G_{ijlm}(x,y))$

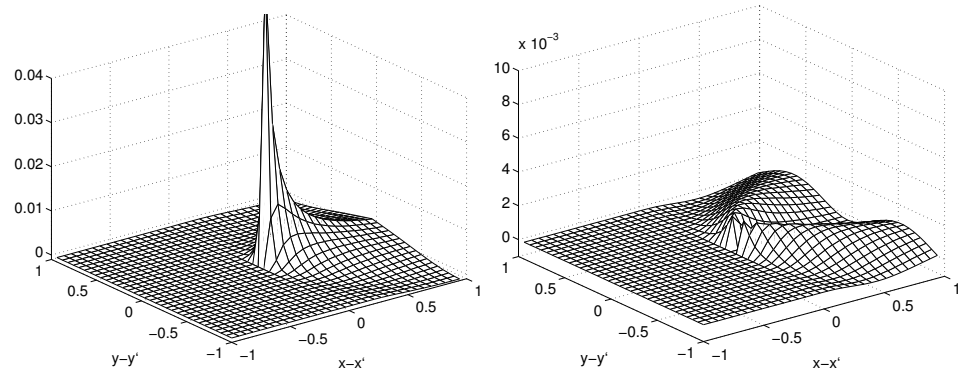


FIGURE 3.7: *Representation of the real and imaginary part of the discrete function* $G^{hh}_{ijlm}(x_i, y_j)$ *with* $G(x_i, y_j) = 1/|x_i - y_j|$, $k = 2\pi$, *and second order for* $w_m^{s_{ij}}(x_i, y_j)$ *and* $w_l^{s_{ij}}(x_i, y_j)$.

is fixed at position $(0, 0)$ and y_j runs over the domain. Notice that locally near $x = y$ the function is not smooth but outside this region the smoothness increases rapidly. So, except for the region around the singularity the values of the function at a given grid point can be obtained accurately by an interpolation of order $p = O(\log(1/\epsilon))$ from its values on a coarser grid (the order of angular (p^{th}_{θ}) and spatial (p^{th}_{xy}) interpolation can be chosen independent).

Coarsening

Having introduced the auxillary kernel G^{hh}_{ijlm} the next step is now to show how using its smoothness the original task of evaluation of (3.26) for $1 \leq l \leq \lambda$ can be replaced by a similar task but defined on a coarser scale. For simplicity again two grids will be assumed. A fine uniform grid with mesh size h further the variables and the indices are denoted by lower-case letters (e.g. v, u, x, y, i and j) and directions $0 \leq l \leq \lambda$ and $0 \leq m \leq \lambda$. Quantities defined on the coarse grid will all be defined by upper-case characters and superscripts.

Since a set of directions has been included to generalize (3.19), it is also required that the distance between directions (δ) be included in the criteria of discretization of (3.26), such as was shown in [14]. This criteria must be fulfilled to ensure that the discretization at least satisfy $p = O(\log(1/\epsilon))$. Furthermore, this requirement also must be satisfied for each coarse level, which implies that when the mesh size increases then the distance between directions must be reduced proportionately. Commonly the number of points for the coarse level is given by the ratio $1 : 2^d$ of the fine level (d is the dimension of the problem), so that, in order to mantain the criteria of the discretization fullfilled, the number of directions for the coarse level must be increased in proportion $2^{d-1} : 1$ relative to the fine level, see section 3.3.3.

Let the kernel (3.39) be suitable smooth on the scale of a coarse grid with mesh size H. Its value for any given x and y a fine grid with mesh size h can be obtained by interpolation from the values on the coarse grid with $O(\epsilon)$ error. In that case (3.38) can be approximated up to $O(\epsilon)$ error by:

$$\tilde{v}_l^h(x_i) = \sum_j \sum_{m\in s_{ij}} \sum_{I\in\bar{\sigma}_i} \sum_{J\in\sigma_j} \omega_{iI}^{hH}\omega_{jJ}^{hH} G_{ijlm}^{HH}(X_I, Y_J) e^{ik(e_m y_j - e_l x_i)} u_m^h(y_j) \tag{3.40}$$

where $\bar{\sigma}_i$ and σ_j are the sets of points in the neighbourhood of x_i and y_j used by the interpolation. This equation will now be re-written into a coarse grid representation. For the case of a smooth or asymptotically smooth kernel, this was quite straightforward since it only involved a change in the order of the summations (see Equation (3.6)). In this case it is a little more complex because exponential terms appear as well as their relation with the directions m and l. Using (3.39) and re-arranging terms, (3.40) can be written as:

$$\begin{aligned}\tilde{v}_l^h(x_i) = \sum_{I\in\bar{\sigma}_i} \omega_{iI}^{hH} e^{ike_l(X_I - x_i)} \sum_j \sum_{J\in\sigma_j} \omega_{jJ}^{hH} G^{HH}(X_I, Y_J) e^{ik|X_I - Y_J|} w_l^{s_{ij}}(X_I, Y_J) \\ \sum_{m\in s_{ij}} w_m^{s_{ij}}(X_I, Y_J) e^{ike_m(y_j - Y_J)} u_m^h(y_j)\end{aligned} \tag{3.41}$$

Taking a close look at this equation it can be seen that the last summation $\sum_{m\in s_{ij}}$ involving the weights $w_m^{s_{ij}}(X_I, Y_J)$ is in fact an interpolation taking place from the fine grid of directions to the direction E_{IJ} of the product of the functions $u_m^h(y_j)$ and $e^{ike_m(y_j - Y_J)}$. As shown in Brandt [14], both $u_m^h(y_j)$ and $e^{ike_m(y_j - Y_J)}$ are smooth on the grid of directions (i.e. as a function of m). As a result their interpolation from $\{e_m\}_{m\in s_{ij}}$ can be replaced, to $O(\epsilon)$ accuracy, by an interpolation from any other close subset of $\{e_m\}$, so that in (3.41) the summation $\sum_{m\in s_{ij}} w_m^{s_{ij}}(X_I, Y_J)$ could for example be replaced by the $\sum_{m\in s_{E_{ij}}} w_m^{s(E_{ij})}(E_{IJ})$. So as well the interpolation could be replaced by an interpolation from the set $\{E_M\}_{M\in S_{IJ}}$, provided $u_m^h(y_j)$ were defined on $\{E_M\}$ instead of on $\{e_m\}$. However, using the fact that $u_m^h(y_j)$ is suitably smooth on $\{e_m\}$ it can be interpolated to the coarse grid of directions $\{E_M\}$ by defining $\tilde{u}_M^h(y_j)$,

$$\tilde{u}_M^h(y_j) = \sum_{m\in s_{E_M}} w_m^{s(E_M)}(E_M) u_m^h(y_j) \tag{3.42}$$

As a reminder, the number of directions on the coarse grid is 2^{d-1} as large as on the fine grid, so this is indeed interpolation. As a result (3.41) can now be replaced by

$$\begin{aligned}\tilde{v}_l^h(x_i) = \sum_{I\in\bar{\sigma}_i} \omega_{iI}^{hH} e^{ike_l(X_I - x_i)} \sum_j \sum_{J\in\sigma_j} \omega_{jJ}^{hH} G^{HH}(X_I, Y_J) e^{ik|X_I - Y_J|} w_l^{s_{ij}}(E_{IJ}) \\ \sum_{M\in S_{IJ}} W_M^{S_{IJ}}(E_{IJ}) e^{ike_m(y_j - Y_J)} \tilde{u}_M^h(y_j)\end{aligned} \tag{3.43}$$

In a similar way (see Brandt [14]), the final result of the algorithm should, up to a permissible $O(\epsilon)$ error, not change when the interpolation coefficients $w_l^{s_{ij}}(E_{IJ})$ in (3.43) are replaced by p^{th}-order interpolation coefficients to the same E_{IJ} from any (other) close set. In particular they can be replaced by coefficients of p^{th}-order interpolation using the set $\{E_L\}$; i.e. a p^{th}-order interpolation from $\{e_l\}$ to $\{E_L\}$ followed by a p^{th}-order interpolation from $\{E_L\}$ to E_{IJ}. This implies that the term $w_l^{s_{ij}}(E_{IJ})$ in (3.43) can be replaced by $\sum_L w_l^{s(E_L)}(E_L) W_L^{S(E_{IJ})}(E_{IJ})$, where the summation is over all *L such that(s.t.)* $l \in s(E_L)$. Introducing this replacement one obtains:

$$\tilde{v}_l^h(x_i) = \sum_{I \in \tilde{\sigma}} \omega_{iI}^{hH} e^{ike_l(X_I - x_i)} \tilde{V}_l^H(X_I) \tag{3.44}$$

where

$$\tilde{V}_l^H(X_I) = \sum_{\substack{L \quad s.t. \\ l \in s(E_L)}} w_l^{s(E_L)}(E_L) V_L^H(X_I) \tag{3.45}$$

$$V_L^H(X_I) = \sum_J G^{HH}(X_I, Y_J) e^{ik|X_I - Y_J|} W_L^{S_{IJ}}(E_{IJ}) \sum_{M \in S_{IJ}} W_M^{S_{IJ}}(E_{IJ}) U_M^H(Y_J) \tag{3.46}$$

$$U_M^H(Y_J) = \sum_{\substack{j \quad s.t. \\ J \in \sigma_j}} e^{ikE_M(y_j - Y_J)} \tilde{u}_M^h(y_j) \tag{3.47}$$

(3.46) is exactly a coarse-grid version of (3.26). So, the coarsening process is complete. The task of the evaluation of (3.26) on the target grid h can thus be replaced by the evaluation of the coarse grid problem (3.46) and a number of intergrid operations in space and directions. The different steps are described below and the entire process is summarized in Figure 3.8 for a two-dimensional problems.

(i) Angular density interpolation:

Calculate $\tilde{u}_M^h(y_j)$ in all points y_j for all coarse grid directions $M = 1, ..., \Lambda$ using Eq. (3.47).

(ii) Density oscillatory anterpolation:

Calculate $U_M^H(Y_J)$ in all coarse grid points for all coarse grid directions $M = 1, ..., \Lambda$, using Eq. (3.42).

(iii) Coarse grid summation:

Compute $V_L^H(X_I)$ defined by (3.46) for all coarse grid points X_I for all coarse grid directions $L = 1, ..., \Lambda$.

(iv) Angular field anterpolation:

Calculate $V_l^H(X_I)$ for each fine grid direction $l = 1, ..., \lambda$ by anterpolation on the grid of directions as defined by Eq. (3.45).

(v) Field oscillatory interpolation:

Calculate $\tilde{v}_l^h(x_i)$ for $l = 1, ..., \lambda$ in all fine grid points x_i using Eq. (3.44)

Notice, that as the number of directions on the coarser grid is 2^{d-1} as large, but the number of nodes 2^d times smaller, when applied recursively, the algorithm will certainly lead to a reduction of the total amount of work when the number of points on the finest grid is sufficiently large.

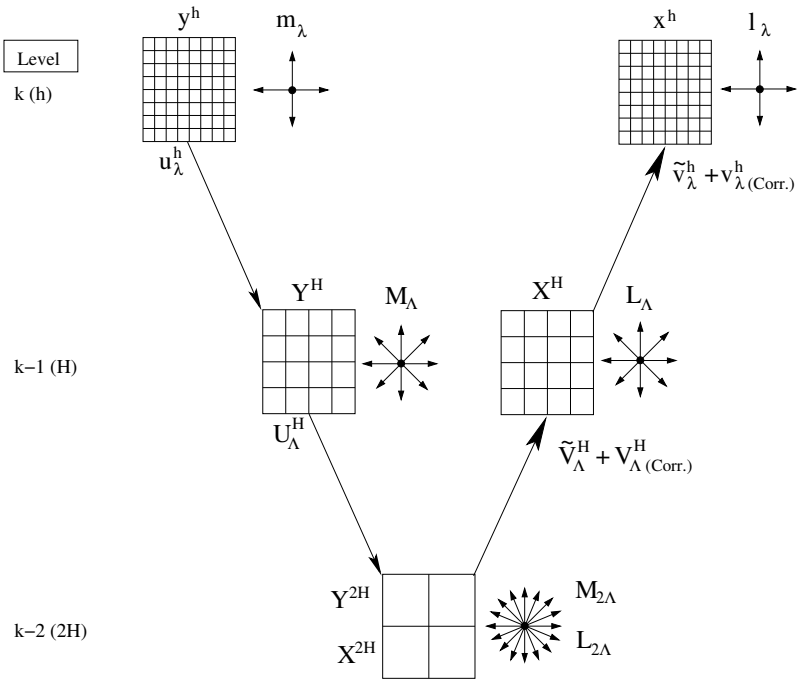


FIGURE 3.8: *Schematic representatition of the process for the evaluation of integral transforms with an oscillatory kernel using the MLMIO algorithm.*

3.3.4 Error Control

Since the function $G(x, y)$ was assumed to be asymptotically smooth in terms of x and y in (3.18), it implies that $G_{ijlm}(x, y)$ must also be asymptotically smooth. However, the function $G_{ijlm}(x, y)$ vanishes outside the region defined by s_{ij}. Unlike the cases described in Section 3.2, the kernel $G_{ijlm}(x, y)$ for each of the sub-summations $v_l^h(x_i)$ has smooth but not symmetrical properties. In this case, the error introduced by the interpolation of $G_{ijlm}(x, y)$ in (3.40) needs to be corrected considering this issue.

From the assumption that $G_{ijlm}(x, y)$ is asymptotically smooth in the angular region bounded by s_{ij}, and taking in to account that the smoothness of it increases as a function of the distance $|x - y|$, the largest errors introduced by the interpolation of $G_{ijlm}(x, y)$ from the coarse grid in (3.40) are located in the region close of $x = y$. In the same way as for asymptotically smooth kernels a local correction is needed to be able to ensure that the error can be kept small compared to the discretization error that is made anyway. Assuming that a correction is computed for all points within a distance r_c from from the singular point $x = y$ and that this distance is chosen such that the contribution of the remaining terms to the error is at most comparable to the discretization error, (3.38) can be written as:

$$v_l^h(x_i) = \tilde{v}_l^h(x_i) + v_l^h(x_i)_{corr} \tag{3.48}$$

where the correction term is given by:

$$v_l^h(x_i)_{corr} = \sum_{j \leq r_c} \sum_{m \in s_{ij}} \left[G_{ijlm}^{hh}(x_i, y_j) - \sum_{J \in \sigma_j} \sum_{I \in \bar{\sigma}_i} \omega_{iI}\omega_{jJ} G_{ijlm}^{hh}(X_I, Y_J) \right] \times e^{ik(e_m y_j - e_l x_i)} u_m^h(y_j) \tag{3.49}$$

Equation (3.49) in general can be seen as the exact definition of the task of corrections. However, the term of the error $G_{ijlm}^{hh}(x_i, y_j) - \sum_{J \in \sigma_j} \sum_{I \in \bar{\sigma}_i} \omega_{iI}\omega_{jJ} G_{ijlm}^{hh}(X_I, Y_J) e^{ik(e_m y_j - e_l x_i)}$ is not suited for simple evaluation as an a posteriori correction. Therefore, an alternative and very robust approach is proposed to obtain the corrections. Venner [71], namely, computed correction tables by using the algorithm itself. For each combination of directions l and m and grids h and H the correction matrix to be used can be obtained by applying the algorithm to the case of $u_m^h(y_j) = \delta(y_j)$ where $\delta(y_j)$ is the delta function taking value 1 at $y = y_j$ and zero otherwise. Subsequently the result obtained with the algorithm ($G_{mlmi}^{hh}(x_i, \delta)$) is substracted from the contribution that this point has in (3.38) (indicated by $G_{ds}^{hh}(x_i, \delta)$). In this way automatically all possible sources of error in the algorithm are taken into account in the correction. The alternative for the evaluation of the term of correction can now be represented by:

$$v_l^h(x_i)_{corr} = \sum_{j \leq r_c} \sum_{m \in s_{ij}} \tilde{G}_\epsilon^{hh}(x_i, \delta) u_m^h(y_j) \tag{3.50}$$

where,

$$\tilde{G}_\epsilon^{hh}(x_i, \delta) = G_{ds}^{hh}(x_i, \delta) - G_{mlmi}^{hh}(x_i, \delta) \tag{3.51}$$

Finally,

(vi) Field corrections: For each x_i correct the contribution to $\tilde{v}_l^h(x_i)$ from $u_m^h(y_j)$ for all y_j into the region bounded by $r_c h$

The resulting algorithm has the prospect of evaluation of (3.38) much faster with a controllable error $O(\epsilon)$ that can be kept small comparable with the discretization error $O(\tau)$ by the choice of the appropriate correction region and order of interpolation in space and in angle. Since the $G_\epsilon^{hh}(x_i, \delta)$ vanish out of the region defined by the relation between directions l and m as a function of the order of the angular interpolation p_θ, an alternative way to perform the corrections faster could be ordering the data in a convenient way to only consider the no vanish values of $G_\epsilon^{hh}(x_i, \delta)$ as a function of the directions l and m in teh correction. The total work will depend on the orders of the different interpolations and the radius of the correction region. Below some details of the effects of the different parameters on the total work are discussed. In the following chapter the algorithm is put to the test and numerical results are presented.

3.4 Estimation of complexity

Since that the algorithm for oscillatory kernels above was derived using two kinds of auxiliary grids (spatial and angular), it is possible to take independent p the order of interpolation on each grid such that they can be represented as p_θ for the angular and p_s for the spatial order of interpolation. So an estimation of the computational work can be done as a function of these two parameters and the correction parameter r_c.

Considering that one operation is defined as one addition and one multiplication, then the work done by the algorithm can be determined by:

1. *Angular interpolation and anterpolation* (steps *i* and *iv*): $O(\lambda n p_\theta)$
2. *Spatial anterpolation and interpolation* (steps *ii* and *v*): $O(\lambda n p_s)$ when the anterpolation/interpolation can be done one dimension at a time, and $O(\lambda n p_s^d)$ otherwise.
3. *Summation at the coarsest level* (step *iii*): This work can be neglected due to the evaluation of (3.46) on the coarsest level is performed in only $O(2\sqrt{n}p_\theta)$ operations.
4. *Correction* (step *vi*): A estimation of the work for the corrections is given by $O(\lambda n r_c^d p_\theta)$ since that for each i and l involves p_θ directions of m and a region bounded by r_c in the vicinity of the singularity ($x = y$).

Notice that the most expensive computational task above is related with the operation of correction due to the error introduced by the interpolation close to the singularity. In order to minimize this work an optimization as a function of the parameters p_θ, p_{xy} and r_c is desirable. Examples of optimization can be found in [16] and [17]. It should be an item for further research.

NUMERICAL RESULTS MLMIO

CHAPTER 4

In this chapter results obtained with the Multilevel Multi-Integration algorithm are presented for a number of one- and two-dimensional model problems of integral transforms with oscillatory kernel. The performance of the algorithm is measured by means of comparing the approximation to the discrete transform that is obtained with the exact analytical solution and with the result obtained using single grid simple matrix multiplication. In various ways it is shown that the generalization of the MLMI algorithm using separation of directions indeed works and enables fast evaluation of discrete integral transform with oscillatory kernel with performance independent of the wavenumber. Finally for reference the performance of the algorithm for some of the model problems is compared with the performance of an FFT algorithm.

4.1 One dimension ($G(x, y)$ smooth)

The first model problem is the evaluation of the integral transform (3.18) for one a dimensional case with $G(x, y)$ a smooth function of x and y:

$$v(x) = \int_{-1}^{1} G(x, y) e^{ik|x-y|} u(y) dy, \quad x \in [-1, 1] \tag{4.1}$$

where,

$$G(x, y) = \cos(y - x) \quad \text{and} \quad u(y) = 1 - y^2, \tag{4.2}$$

for different values of the wave number (k).

For this case the analytical solution is known, see Appendix B, and will be used to evaluate the accuracy of the numerical results obtained.

4.1.1 Discretization

The integral transform (4.1) was discretized as descibed in section 3.3. Here a constant meshsize h is assumed. $u^h(y_j)$ and $v^h(x_i)$ are the discrete approximations to u and v on the grid. The numerical approximation of (4.1) was done in two ways:

i) Single Matrix Multiplication

In this case, the integral transform (4.1) was discretized as follows: In each interval $[y_j - \frac{h}{2}, y_j + \frac{h}{2}]$ the function is approximated by a piecewise constant function with the value $u^h(y_j)$. Subsequently, the contribution of the interval to $v^h(x)$ is defined as:

$$\delta_j v^h(x) = \left[\int_{y_j - \frac{h}{2}}^{y_j + \frac{h}{2}} G(x,y) e^{ik|x-y|} dy \right] u^h(y_j). \tag{4.3}$$

Summing up the contributions of all integration intervals one obtains a discrete approximation to v given by the following *single matrix multiplication*:

$$v^h(x_i) = \sum_j G^{hh}_{osc}(x_i, y_j)\ u^h(y_j) \tag{4.4}$$

where the coefficients of the matrix (discrete kernel) G^{hh}_{osc} are given by:

$$G^{hh}_{osc}(x_i, y_j) = \int_{y_j - \frac{h}{2}}^{y_j + \frac{h}{2}} G(x_i, y) e^{ik|x_i - y|} dy \tag{4.5}$$

These coefficients were computed numerically using a Gauss adaptive quadrature approach with predefined accuracy $\epsilon = O(10^{-8})$. The discretization is of second order and the error should be insensitive to the wave number k.

ii) Separation of Directions (MLMIO)

The second approach uses the *separation of directions* in MLMIO algorithm as described in the previous chapter. The discrete representation of v is:

$$v^h(x_i) = e^{-ikx_i} v^h_+(x_i) + e^{ikx_i} v^h_-(x_i) \tag{4.6}$$

where

$$v^h_+(x_i) = \sum_j G^{hh}_+(x_i, y_j) u^h_+(y_j) \tag{4.7}$$

$$v^h_-(x_i) = \sum_j G^{hh}_-(x_i, y_j) u^h_-(y_j) \tag{4.8}$$

with,

$$G^{hh}_+(x_i, y_j) = \begin{cases} (A^+ + B^+) G^{hh}(x_i, y_j) & j > i, \\ A^+\ G^{hh}(x_i, y_j) & j = i, \\ 0 & j < i, \end{cases} \tag{4.9}$$

$$G^{hh}_-(x_i, y_j) = \begin{cases} 0 & j > i, \\ B^-\ G^{hh}(x_i, y_j) & j = i, \\ (A^- + B^-) G^{hh}(x_i, y_j) & j < i, \end{cases} \tag{4.10}$$

and

$$u_+^h(y_j) = e^{iky_j}u^h(y_j) \quad \text{and} \quad u_-^h(y_j) = e^{-iky_j}u^h(y_j) \tag{4.11}$$

The coefficients G_+^{hh} and G_-^{hh} depend on the specifics of the discretization. Here a second order discretization was used based on a piecewise linear approximation of $u^h(y_j)$ at integration intervals $[y_j, y_{j+1}]$. The coefficients were computed analytically as illustrated in appendix B where also the constants A^+, A^-, B^+ and B^- are given.

4.1.2 Performance

Hereafter the target grid with meshsize h satisfies the constraint $h = \lambda/n_\lambda$, where n_λ is the number of points per wavelength (λ), in this case n_λ was fixed to at least in 8 points. Notice that using the MLMIO algorithm this constraint is not needed in the coarsening procedure since that using the discretization of separation of directions smoothness of the kernel for each subtransform is introduced.

Results for (4.4) with (4.5) and for (4.6)-(4.11) on a series of grids with decreasing meshsize were computed. The grids are numbered with a grid index *fl* and each next grid has a meshsize that is two times smaller. The number of intervals on the grid is n. Results for (4.6)-(4.11) were obtained in two ways. Firstly, using single grid multisummation to obtain the two discrete subtransforms. Secondly using a MLMI algorithm for the fast evaluation of these discrete subtransforms.

The error of the approximations and its dependence on the wavenumber (k) are compared. The accuracy of the discrete approximations was measured using the Euclidean norm of the absolute error in $v^h(x)$:

$$||\epsilon||_2 = \frac{1}{n+1}\sum_{i=0}^{n}|v(x_i) - v^h(x_i)| \tag{4.12}$$

where $v(x)$ is the exact analytical solution (see Appendix B) and $(n+1)$ the number of discrete grid points. Notice that the $v(x_i)$ and $v^h(x_i)$ are complex.

$k = 64\pi$					
		Single Matrix Multiplication		Separation of directions	
No.	Mesh	$\|\|\epsilon\|\|$	cpu time [sec.]	$\|\|\epsilon\|\|$	cpu time [sec.]
8	512	1.12e-06	<1.00e-02	3.90e-08	<1.00e-02
9	1024	2.78e-07	1.00e-02	9.68e-09	1.00e-02
10	2048	6.93e-08	3.00e-02	2.42e-09	3.00e-02
11	4096	1.73e-08	1.30e-01	6.03e-10	1.40e-01
12	8192	4.33e-09	5.30e-01	1.51e-10	5.40e-01
13	16384	1.08e-09	2.13e+00	3.77e-11	3.00e+00
14	32768	2.71e-10	1.09e+01	9.42e-12	1.61e+01
15	65536	6.76e-11	4.83e+01	2.35e-12	6.42e+01
16	131072	1.69e-11	1.99e+02	6.14e-13	2.57e+02
17	262144	4.23e-12	8.12e+02	1.10e-13	1.02e+03

TABLE 4.1: *Average norm of the error and cpu time for both discretizations Single Matrix Multiplication and Separation of Directions)($k = 64\pi$).*

The evaluation was done for three values of k that represent a low, medium and a high wave number: ($k = 8\pi$, 32π and $k = 64\pi$). In this section the results for $k = 64\pi$ are shown. The other results can be found in Appendix B.

In table 4.1 the results are shown for single grid evaluation. The first column gives a grid index. The second column gives the number of intervals n on the grid. In the third and fourth column the error (4.12) and the computing time needed for the summation (4.4) are shown. In the fifth and sixth column similar results are shown for the summation (4.6).

It can seen that the evolution of the error follows the expected second order behaviour when the number of intervals is doubled, the norm of the error reduces by a factor 4.

From table 4.1 and additional results in Appendix B (tables B.1 and B.2), it can be seen that for both schemes the error is independent of the wavenumber k. This is due to the fact that in the single matrix scheme a Gauss quadrature is used for the kernel coefficients with a predefined accuracy and in the second scheme they were obtained analytically.

Finally the computing time used for the evaluation increases by a factor 4 each time the mesh size is halved which clearly shows the $O(n^2)$ work involved in the direct summation.

$\|\|\epsilon\|\|, k = 64\pi$							
fl	Mesh	cl=fl	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
8	512	3.90e-08	3.90e-08	3.90e-08	3.90e-08	3.90e-08	3.90e-08
9	1024	9.68e-09	9.68e-09	9.68e-09	9.68e-09	9.68e-09	9.68e-09*
10	2048	2.42e-09	2.42e-09	2.42e-09	2.42e-09	2.42e-09	2.42e-09
11	4096	6.03e-10	6.03e-10	6.03e-10	6.03e-10	6.03e-10	6.03e-10
12	8192	1.51e-10	1.51e-10	1.51e-10	1.51e-10	1.51e-10	1.51e-10
13	16384	3.77e-11	3.77e-11	3.77e-11	3.77e-11	3.77e-11	3.77e-11
14	32768	9.42e-12	9.42e-12	9.42e-12	9.42e-12	9.42e-12	9.42e-12
15	65536	2.35e-12	2.35e-12	2.35e-12	2.35e-12	2.35e-12	2.35e-12
16	131072	6.14e-13	6.14e-13	6.14e-13	6.14e-13	6.14e-13	6.14e-13
17	262144	1.10e-13	1.10e-13	1.10e-13	1.10e-13	1.10e-13	1.10e-13

TABLE 4.2: *Average norm of the error in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 64\pi$)*

$\|\|\epsilon\|\|, k = 64\pi$							
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9	cl=fl-10	cl=fl-11
8	512	3.90e-08	3.90e-08				
9	1024	9.68e-09	9.68e-09	9.68e-09			
10	2048	2.42e-09	2.41e-09	2.41e-09	2.42e-09		
11	4096	6.03e-10*	6.03e-10	6.03e-10	6.03e-10	6.04e-10	
12	8192	1.51e-10	1.51e-10	1.51e-10	1.50e-10	1.50e-10	1.51e-10
13	16384	3.77e-11	3.77e-11*	3.77e-11	3.76e-11	3.71e-11	3.72e-11
14	32768	9.42e-12	9.42e-12	9.42e-12	9.41e-12	9.34e-12	8.84e-12
15	65536	2.35e-12	2.35e-12	2.35e-12*	2.35e-12	2.34e-12	2.30e-12
16	131072	6.14e-13	6.14e-13	6.14e-13	6.14e-13	6.14e-13	6.18e-13
17	262144	1.10e-13	1.10e-13	1.10e-13	1.10e-13*	1.11e-13	1.15e-13

TABLE 4.3: *Average norm of the error in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ transfers. ($k = 64\pi$) (Continue)*

Tables 4.2 and 4.3 show the norm of the error (4.12) when (4.6) is evaluated with the

MLMIO algorithm, i.e. using a MLMI algorithm to evaluate each of the discrete subtransforms in (4.6). Note that due to discontinuity created in $G_+(x,y)$ and $G_-(x,y)$ to carry out the fast evaluation at the expense of an error that is small compared to the discretization error, a correction is needed for the contribution of p points around the discontinuity. The correction is applied posteriori, see section 3.2.3.

The tables are organized as follows; the first column (fl) indicates the index of the target grid on which the evaluation is needed. The second column (mesh) shows the number of intervals on this grid ($n = 2^{fl+1}$). The other columns give the error defined as (4.12) in the final result on the target grid fl when the MLMIO algorithm is used with the actual coarse grid summation carried out on the grid with index cl. So, the column with $cl = cf$ displays the results of single grid summation as shows in table 4.1. The column $cl = fl - 1$ gives the error in the final results when one coarse grid is used, and so on. The results marked with an asterisk (*) indicate results obtained using a coarsest grid with $N = \sqrt{n}$ points. This is the maximum useful degree of coarsening.

As can be seen from the tables in all the cases the norm of the error is comparable to the result obtained with single grid summation. This implies that even when coarsening is continued to $O(\sqrt{n})$ intervals the additional error in the result obtained on the target grid is small compared to the discretization error.

In the graphs on the left side of the Figures 4.1, 4.2 and 4.3 the behaviour of the norm of the error for single matrix multiplication (SMM), for the discretization by separation of directions using single grid summation, and for the MLMIO algorithm is shown graphically for the three wave numbers considered. The results again show that for all cases the accuracy of the result is second order. Moreover, one can not see a significative difference between the single grid separation of direction results and the MLMIO results, which again shows that the error resulting from the fast evaluation is negligable compared to the discretization error.

On the right side of the Figures 4.1, 4.2 and 4.3 the overall performance of the single grid summation and MLMIO algorithm is shown by means of the error in the result as a function of the cpu time invested to obtain the result. The efficiency gain is clear from the fact that using the MLMIO algorithm the cpu time invested to obtain a given accuracy is several orders of magnitude smaller.

This is further detailed in the tables 4.4 and 4.5 where the computing times used in the evaluation of the simple matrix multiplication (4.4) and separation of directions (4.6) using both, single matrix multiplication ($cl = fl$) and the MLMIO algorithm are shown. The results were obtained in a Pentium IV PC. It can clearly be seen that the evaluation of (4.6) using the MLMIO algorithm gives significant reductions in the cpu time compared to the evaluation using single matrix multiplication. These reductions should be computer intependent. Moreover, it can be seen from the results marked with an asterisk (*) that the computing time is really reduced from $O(n^2)$ to $O(n)$ operations. This is further illustred in the Figure 4.4.

From comparing the results given here with the auxiliary results in tables B.2-B.6 in appendix B, it is clear that the algorithms performance is independent of the wavenumber.

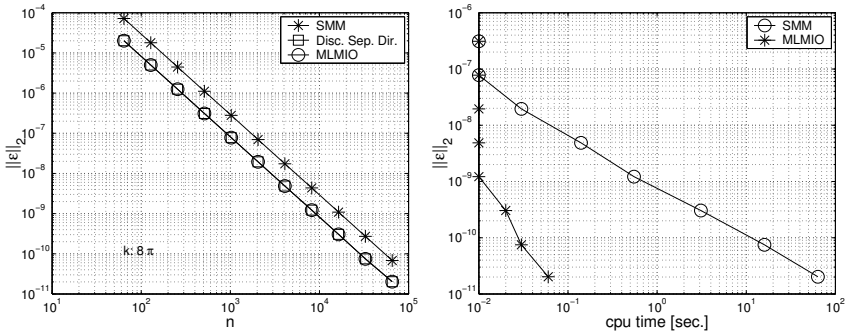


FIGURE 4.1: *Left: Norm of the error from the numerical evaluation of (4.1) using single matrix multiplication, discretization by separation of directions and the MLMIO algorithm. Right: Overall performance for both single matrix multiplication and the MLMIO algorithm. ($k = 8\pi$ and $p = 8$)*

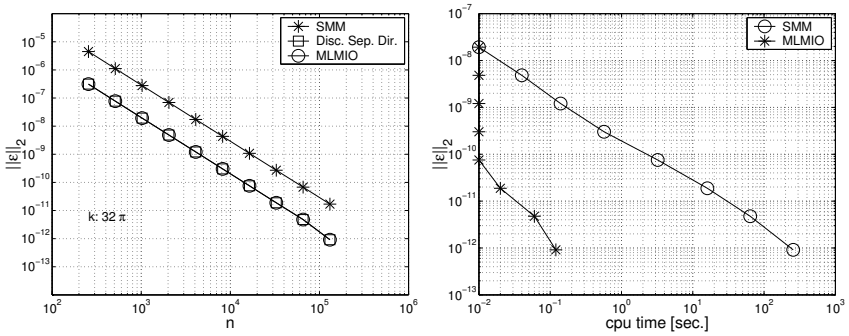


FIGURE 4.2: *Left: Norm of the error from the numerical evaluation of (4.1) using single matrix multiplication, discretization by separation of directions and the MLMIO algorithm. Right: Overall performance for both single matrix multiplication and the MLMIO algorithm. ($k = 32\pi$ and $p = 8$)*

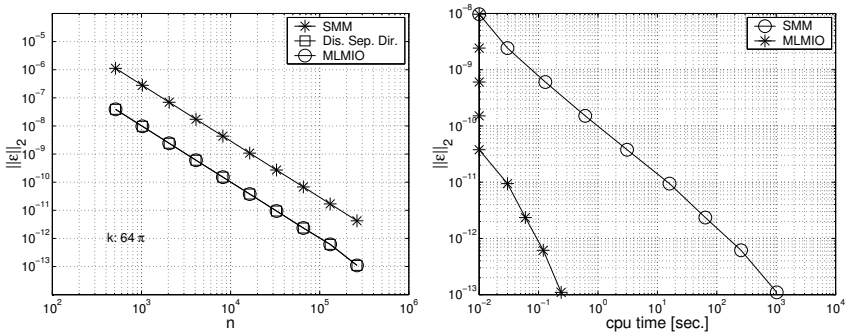


FIGURE 4.3: *Left: Norm of the error from the numerical evaluation of (4.1) using single matrix multiplication, discretization by separation of directions and the MLMIO algorithm. Right: Overall performance for both single matrix multiplication and the MLMIO algorithm.($k = 64\pi$ and $p = 8$)*

cpu time [sec.], $k = 64\pi$							
fl	Mesh	cl=fl	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
8	512	<1.0e-02	∼	∼	∼	∼	∼
9	1024	1.00e-02	<1.0e-02	∼	∼	∼	∼*
10	2048	3.00e-02	1.00e-02	<1.0e-02	∼	∼	∼
11	4096	1.40e-01	3.00e-02	1.00e-02	<1.0e-02	∼	∼
12	8192	5.40e-01	1.40e-01	4.00e-02	1.00e-02	1.00e-02	1.00e-02
13	16384	3.00e+00	5.40e-01	1.40e-01	5.00e-02	2.00e-02	2.00e-02
14	32768	1.61e+01	2.57e+00	5.60e-01	1.50e-01	6.00e-02	3.00e-02
15	65536	6.42e+01	1.56e+01	2.62e+00	5.80e-01	1.90e-01	8.00e-02
16	131072	2.57e+02	6.46e+01	1.59e+01	2.69e+00	6.40e-01	2.30e-01
17	262144	1.02e+03	2.56e+02	6.44e+01	1.64e+01	2.77e+00	7.30e-01

TABLE 4.4: *Cpu time (sec.) invested in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ order transfers.*

cpu time [sec.], $k = 64\pi$						
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9	cl=fl-10
8	512	∼	∼			
9	1024	∼	∼	∼		
10	2048	∼	∼	∼	∼	
11	4096	∼*	∼	∼	∼	∼
12	8192	<1.0e-02	∼	∼	∼	∼
13	16384	1.00e-02	<1.0e-02*	∼	∼	∼
14	32768	3.00e-02	2.00e-02	2.00e-02	2.00e-02	2.00e-02
15	65536	6.00e-02	5.00e-02	5.00e-02*	5.00e-02	5.00e-02
16	131072	1.30e-01	1.10e-01	1.00e-01	1.00e-01	1.00e-01
17	262144	3.30e-01	2.40e-01	2.20e-01	2.00e-01*	2.00e-01

TABLE 4.5: *Cpu time (sec.) invested in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ order transfers.(Continue)*

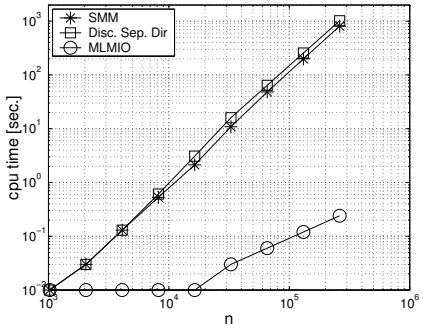


FIGURE 4.4: *Cpu time (sec.) invested in the numerical evaluation of (4.1) using direct matrix multiplication and the MLMIO algorithm with $p = 8$ order transferes.*

4.2 One dimension ($G(x, y)$ asymptotically smooth)

In the second example considered $G(x, y)$ is taken as an asymptotically smooth function, which is more realistic in view of real applications. In this case, $G(x, y)$ often has a singularity at the location $x = y$.

In the description of the MLMIO algorithm it was explained that larger errors are introduced by the interpolation of the kernel in the region close to the singularity (see Sec. 3.3.2), which need to be corrected to ensure that the fast evaluation at most introduces an error comparable to the discretization error. The correction factors for a local region near the singularity can be computed easily and the corrections applied a posteriori, see [70].

Consider the evaluation of the integral transform defined by:

$$v(x_i) = \int_{-1}^{1} G(x, y) e^{ik|x-y|} u(y) dy, \quad x \in [-1, 1] \tag{4.13}$$

where,

$$G(x, y) = \ln |x - y| \quad \text{and} \quad u(y) = 1 - y^2 \tag{4.14}$$

for different (k). As for the previous problem the analytical solution can be obtained and used for performance check. It is given in Appendix B.

4.2.1 Discretization

The discrete approximation of u and v on a grid with mesh size h is defined by $u^h(y_j)$ and $v^h(x_i)$ respectively.

i) Single Matrix Multiplication

The integral transform is discretized as follows. On each integration interval $[y_j - \frac{h}{2}, y_j + \frac{h}{2}]$ the product $e^{ik|x-y|}\, u$ is approximated by a piecewise constant with the value $e^{ik|x_i - y_j|}\, u^h(y_j)$. Subsequently the contribution of the integration interval to the value of the transform at a point x is approximated by:

$$\delta_j v^h(x) = \left[\int_{y_j-\frac{h}{2}}^{y_j+\frac{h}{2}} G(x,y)dy\right] e^{ik|x-y_j|}u^h(y_j). \tag{4.15}$$

The integral in 4.15 can be computed analytically. Summing up the contributions of all intervals yields the second order approximation:

$$v^h(x_i) = \sum_j G^{hh}_{osc}(x_i,y_j)\, u^h(y_j) \tag{4.16}$$

where

$$G^{hh}_{osc}(x_i,y_j) = G^{hh}(x_i,y_j)e^{ik|x_i-y_j|} \tag{4.17}$$

with,

$$\begin{aligned} G^{hh}(x_i,y_j) = (x_i-y_j+\frac{h}{2})\ln|x_i-y_j+\frac{h}{2}| - h \\ -(x_i-y_j-\frac{h}{2})\ln|x_i-y_j-\frac{h}{2}| - h \end{aligned} \tag{4.18}$$

ii) Separation of Directions (MLMIO)

First the transform is split into two directions. Subsequently each subtransform is discretized by approximating $u^h(y)$ by a piecewise constant function on each interval $[y_j - h/2, y_j + h/2]$. In the end, the contribution of all intervals are added up giving a second order approximation to the continuous transform:

$$v^h(x_i) = e^{-ikx_i}v^h_+(x_i) + e^{ikx_i}v^h_-(x_i) \tag{4.19}$$

where

$$v^h_+(x_i) = \sum_j G^{hh}_+(x_i,y_j)u^h_+(y_j) \tag{4.20}$$

$$v^h_-(x_i) = \sum_j G^{hh}_-(x_i,y_j)u^h_-(y_j) \tag{4.21}$$

with,

$$G^{hh}_+(x_i,y_j) = \begin{cases} G^{hh}_p(x_i,y_j) & j \geq i, \\ 0 & j < i, \end{cases} \tag{4.22}$$

$$G^{hh}_-(x_i,y_j) = \begin{cases} 0 & j > i, \\ G^{hh}_n(x_i,y_j) & j \leq i, \end{cases} \tag{4.23}$$

and

$$u^h_+(y_j) = e^{iky_j}u^h(y_j) \quad \text{and} \quad u^h_-(y_j) = e^{-iky_j}u^h(y_j) \tag{4.24}$$

The coefficients G^{hh}_p and G^{hh}_n can be computed analytically, see Appendix B.

4.2.2 Performance

The results for this model problem will be presented in a same way as for the previous problem. First the dependence of the accuracy on the mesh size and the wave number ($k = 64\pi$ for this case) is given in the third column ($cl = fl$) of the table 4.6, which actually is the discretization error. Next, the results for the MLMIO algorithm are presented in the tables 4.6 and 4.7 and Figures 4.5, 4.6 and 4.7. Additional results are given in the Appendix B.

Since in both cases, single matrix multiplication and separation of directions, the function u is approximated by a piecewise constant and the coefficients G, G_+ and G_- are evaluated analytically, there is no difference between the results obtained on a single grid with the two approaches.

In table 4.6 and B.7 and B.10 the discretization error (column $cl = fl$) is defined as:

$$||\epsilon||_2 = \frac{1}{n+1}\sum_{i=0}^{n} |v(x_i) - \tilde{v}^h(x_i)| \tag{4.25}$$

which is given as a function of the mesh size for the cases with $k = 8\pi$, 32π and 64π respectively. From the results it can be seen that the discretization error is indeed $O(h^2)$ as expected.

$\|\epsilon\|, k = 64\pi$							
fl	Mesh	cl=fl	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
8	512	2.36e-03	2.36e-03	2.36e-03	2.36e-03	2.36e-03	2.36e-03
9	1024	6.18e-04	6.18e-04	6.18e-04	6.18e-04	6.18e-04	6.18e-04*
10	2048	1.62e-04	1.62e-04	1.62e-04	1.62e-04	1.62e-04	1.62e-04
11	4096	4.23e-05	4.23e-05	4.23e-05	4.23e-05	4.22e-05	4.23e-05
12	8192	1.10e-05	1.10e-05	1.10e-05	1.10e-05	1.10e-05	1.10e-05
13	16384	2.87e-06	2.87e-06	2.87e-06	2.88e-06	2.88e-06	2.87e-06
14	32768	7.47e-07	7.47e-07	7.47e-07	7.48e-07	7.49e-07	7.49e-07
15	65536	1.94e-07	1.94e-07	1.94e-07	1.94e-07	1.95e-07	1.96e-07
16	131072	5.03e-08	5.03e-08	5.03e-08	5.03e-08	5.05e-08	5.10e-08
17	262144	1.30e-08	1.30e-08	1.30e-08	1.30e-08	1.31e-08	1.31e-08

TABLE 4.6: *Average norm of the error in the numerical evaluation of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 64\pi$)*

To illustrate the performance of the MLMIO algorithm the *fast evaluation error* will be used which is defined as:

$$||FE\epsilon||_2 = \frac{1}{n+1}\sum_{i=0}^{n} |v_h(x_i)^{fl} - v_h(x_i)^{cl}| \tag{4.26}$$

i.e. the difference between the result obtained on the target grid using no coarse grids at all and the results obtained on the target grid using the MLMIO algorithm in which the actual summation take place on grid *cl*.

In the fast evaluation, as a result of the singularity a local correction is needed for the error made by the interpolation near the singularity, see section 3.2.3. In this case a correction was introduced for the contribution of $m = 3 + \ln(n)$ points around ($x_i = y_j$).

$\|\|\epsilon\|\|, k = 64\pi$						
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9	cl=fl-10
8	512	2.36e-03				
9	1024	6.18e-04				
10	2048	1.62e-04	1.62e-04			
11	4096	4.23e-05*	4.23e-05			
12	8192	1.10e-05	1.10e-05	1.10e-05		
13	16384	2.85e-06	2.87e-06*	2.87e-06		
14	32768	7.49e-07	7.44e-07	7.37e-07	7.38e-07	
15	65536	1.96e-07	1.96e-07	1.95e-07*	1.89e-07	
16	131072	5.24e-08	5.20e-08	5.24e-08	5.15e-08	4.65e-08
17	262144	1.34e-08	1.40e-08	1.35e-08	1.48e-08*	1.70e-08

TABLE 4.7: *Average norm of the error in the numerical evaluation of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 64\pi$) (Continue)*

$\|\|FE\epsilon\|\|, k = 64\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
8	512	2.36e-03	1.65e-07	2.35e-07	1.79e-07	1.79e-07	1.79e-07
9	1024	6.18e-04	1.03e-07	1.36e-07	2.78e-07	2.23e-07	2.23e-07*
10	2048	1.62e-04	3.60e-08	4.95e-08	1.25e-07	1.38e-07	1.55e-07
11	4096	4.23e-05	6.20e-09	1.45e-08	6.33e-09	3.44e-08	5.34e-09
12	8192	1.10e-05	2.23e-09	6.21e-09	1.01e-08	4.74e-09	2.25e-08
13	16384	2.87e-06	1.12e-09	3.27e-09	6.98e-09	9.96e-09	5.58e-09
14	32768	7.47e-07	2.32e-10	6.86e-10	1.55e-09	2.95e-09	3.46e-09
15	65536	1.94e-07	8.76e-11	2.61e-10	6.02e-10	1.25e-09	2.25e-09
16	131072	5.03e-08	4.38e-11	1.30e-10	3.03e-10	6.43e-10	1.28e-09
17	262144	1.30e-08	1.04e-11	3.10e-11	7.20e-11	1.54e-10	3.14e-10

TABLE 4.8: *Norm of the fast evaluation error of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 64\pi$)*

In table 4.8 and B.11-B.14 the fast evaluation error is given as a function of the number of nodes (mesh size) on the target grid and the coarsest grid used in the MLMIO algorithm. For reference in the third column the discretization error is given. The MLMIO algorithm was used with 8^{th} order transfers. Note that in the tables only results are presented for target grids on which the mesh size is sufficiently small to resolve the oscillation with at least $n_\lambda = 8$ points. The results shown in the tables 4.8 and B.11-B.14 show that in all cases the error made by the fast evaluation is smaller or at most comparable than the discretization error, independently of the wavenumber k. The results marked with (*) indicate that the coarsening can be carried out until a grid is reached with $\sqrt{n}$ points where the actual summation can be carried out in $O(n)$ operations. In many cases this implies that the coarsening is carried out until a grid at which the kernel itself is not longer smooth. The fact that it can be clearly illustrated the real value of the MLMIO algorithm. Unsing the standard algorithm would not be possible.

$\|\|FE\epsilon\|\|, k = 64\pi$						
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9	cl=fl-10
8	512	1.79e-07				
9	1024	2.23e-07				
10	2048	1.55e-07	1.55e-07			
11	4096	1.09e-08*	1.09e-08			
12	8192	9.68e-09	1.40e-08	1.40e-08		
13	16384	2.20e-08	8.62e-09*	1.29e-08		
14	32768	3.81e-09	1.01e-08	1.05e-08	1.15e-08	
15	65536	2.40e-09	3.29e-09	8.48e-09*	1.16e-08	
16	131072	2.27e-09	2.38e-09	3.32e-09	8.52e-09	1.17e-08
17	262144	6.12e-10	1.04e-09	7.98e-10	1.84e-09*	4.43e-09

TABLE 4.9: *Norm of the fast evaluation error of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 64\pi$) (Continue)*

This is illustrated graphically on left side of the Figure 4.5. In this figure the development of the evaluation error with increasing coarsening is shown for $k = 8\pi$ and the target grids, $fl = 9$ and $fl = 15$ respectively. The level of the discretization error is indicated by the circles. It can be seen that each additional coarser grid used in the fast evaluation adds to the fast evaluation error but the total accumulated error is still at most comparable to the discretization error.

Finally, in table 4.8 and 4.9 the computing time needed to obtain the results presented in tables 4.6 and 4.7 are shown. Note that the cpu time should be $O(n \log n)$ because of the work invested in the corrections. However as can be seen from comparing the results marked with (*) that the logarithmic factor is hardly noticeable.

The overall performance of the MLMIO algorithm for each value of k is illustrated in the right side of the Figures 4.5, 4.6 and 4.7 where the error in the final results is shown as a function of the work invested. For reference also the results for the single matrix multiplication are shown. The MLMIO algorithm reduces the amount of work to be invested for a given accuracy significantly. In fact the product of work times accuracy $WE = O(1)$ for single grid summation and $WE = O(1/h)$ for MLMIO.

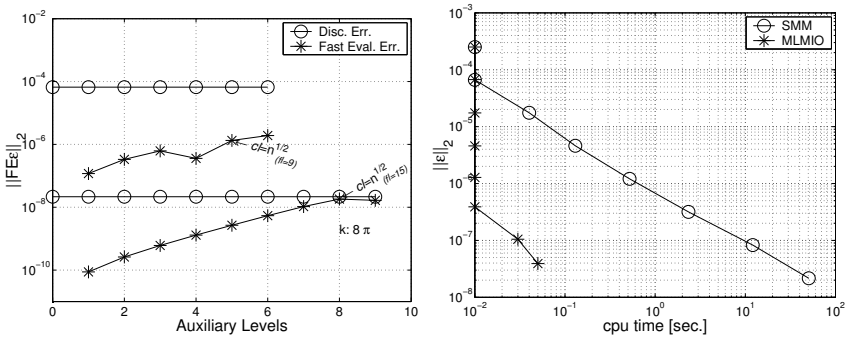


FIGURE 4.5: *Left: Norm of the fast evaluation error using the MLMIO in the evaluation of (4.13) Right: Overall performance for both single matrix multiplication and the MLMIO algorithm.($k = 8\pi$ and $p = 8$)*

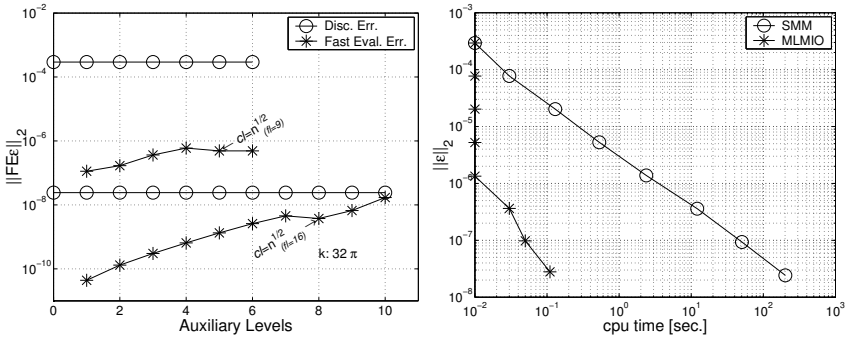


FIGURE 4.6: *Left: Norm of the fast evaluation error using the MLMIO in the evaluation of (4.13) Right: Overall performance for both single matrix multiplication and the MLMIO algorithm.($k = 32\pi$ and $p = 8$)*

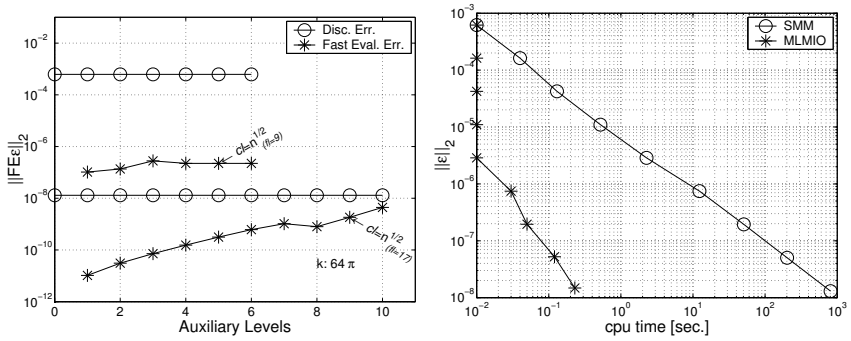


FIGURE 4.7: *Left: Norm of the fast evaluation error using the MLMIO in the evaluation of (4.13) Right: Overall performance for both single matrix multiplication and the MLMIO algorithm.($k = 64\pi$ and $p = 8$)*

4.3 Two dimensional problem: $G(x, y)$ explicit oscillatory

In the one dimensional case the separation of directions isolates the effect of the oscillatory behaviour of the kernel into functions of x and y that can be taken outside the integral and incorporated in the $u(y)$ respectively. As a result the algorithm for oscillatory kernels differs only little from the regular algorithm. The required computational effort for the fast evaluation of a discrete integral transform with an oscillatory kernel is only twice the effort needed that is needed for an asymptotically smooth kernel. The real test is off course the two dimensional case where the algorithm is much more involved as the separation of directions is implicit, and the number of directions increases each additional coarsening step.

The first two dimensional model problem considered is the evaluation of an integral transform with asymptotically smooth *explicit* oscillatory kernel. The total kernel is exactly of the form assumed in the generic problem used to describe the algorithm in chapter 3.

The general notation introduced in section 3.1 is used with $d = 2$ and $\bar{d} = d$, x and $y \in I\!R^2$, so that $x = (x^1, x^2)$ and $y = (y^1, y^2)$ with $|x - y| = [(x^1 - y^1)^2 + (x^2 - y^2)^2]^{\frac{1}{2}}$. The model problem is defined as the evaluation of:

$$v(x) = \int_\Omega G(x, y)\, e^{ik|x-y|}\, u(y)\, dy, \qquad x \in \Omega \tag{4.27}$$

with $\Omega \in [-1, 1] \times [-1, 1]$, where

$$G(x, y) = \frac{1}{|x - y|} \tag{4.28}$$

and

$$u(y) \equiv u(y^1, y^2) = \sin(ky^1)\, \sin(ky^2), \quad y \in \Omega \tag{4.29}$$

for different values of k. Note that for this problem the wavenumber k is introduced also in the function $u(y)$ with the aim to represent modal behaviour as is often the case in practical

applications. However, note that the behaviour or $u(y)$ plays no role at all in the algorithm. In fact, a more oscillatory $u(y)$ makes fast evaluation easier. This is because the discretization error for an oscillatory $u(y)$ is larger so the permissible error in the fast evaluation algorithm is larger and lower order transfers are already sufficient to achieve the result.

4.3.1 Discretization

As for the one dimensional model problems the performance of the MLMIO algorithm is illustrated by comparing results for two cases.

The notation below is as defined in section 3.1, so that $y_j \in I\!R^2$ (i.e. $y_j = (y_j^1, y_j^2)$) is a point of the integration grid and at this location the value of $u(y_j)$ is assumed to be given. The integral transform $v(x)$ is to be evaluated in all points $x_i \in I\!R^2$ (i.e. $x_i = (x_i^1, x_i^2)$) of an evaluation grid is the location of a receiver point $v(x_i)$. The distance between two points x_i and y_j is $|x_i - y_j| = [(x_i^1 - y_j^1)^2 + (x_i^2 - y_j^2)^2]^{\frac{1}{2}}$.

i) Single Matrix Multiplication

The product $e^{ik|x-y|}u(y)$ is approximated by a piecewise constant function on each integration interval of size $h \times h$ centered around of the point y_j. Subsequently the contribution of each interval is computed. Summing up over all integration intervals then yields the following second order approximation to (4.27):

$$v^h(x_i) = \sum_j G^{hh}_{osc}(x_i, y_j)\; u^h(y_j) \tag{4.30}$$

where

$$G^{hh}_{osc}(x_i, y_j) = G^{hh}(x_i, y_j)\; e^{ik|x_i - y_j|} \tag{4.31}$$

with

$$G^{hh}(x_i, y_j) \equiv G^{hh}(x_i^1, x_i^2, y_j^1, y_j^2) = \int_{y_j^1 - \frac{h}{2}}^{y_j^1 + \frac{h}{2}} \int_{y_j^2 - \frac{h}{2}}^{y_j^2 + \frac{h}{2}} \frac{dy^1 dy^2}{[(x_i^1 - y^1)^2 + (x_i^2 - y^2)^2]^{\frac{1}{2}}} \tag{4.32}$$

and

$$u^h(y_j) \equiv u^h(y_j^1, y_j^2) = \sin(k y_j^1)\; \sin(k y_j^2) \tag{4.33}$$

The integrals in (4.32) can be evaluated analytically, see Appendix B. Note that for an uniform grid these coefficients only depend on the distance between the point x_i and y_j and can be precomputed and stored in $O(n)$ operations.

ii) Separation of Directions (MLMIO)

The second formulation is obtained using the description given in Section 3.3 in terms of *separation of directions*. The approximation to (4.27) is:

$$v^h(x_i) = \sum_l v_l^h(x_i) \tag{4.34}$$

where

$$v_l^h(x_i) = \sum_j G^{hh}(x_i, y_j)\, e^{ik|x_i - y_j|}\, w_l^{s_{ij}}(e_{ij}) \sum_{m \in s_{ij}} w_m^{s_{ij}}(e_{ij})\, u_m^h(y_j) \tag{4.35}$$

with

$$u_m^h(y_j) = u^h(y_j) \tag{4.36}$$

where $v_l^h(x_i)$ represents a discrete subtransforms as defined by (3.26). Each discrete subtransform has been obtained from the continuous problem using the same procedure. The evaluation of the coefficients $G^{hh}(x_i, y_j)$ was done using (4.32) as well. This implies that using a single target grids both results should be exactly the same.

4.3.2 Performance

The objective is to show that with the MLMIO algorithm the discrete transform can be evaluated faster at the expense of an error that is at most comparable to the discretization error. However, for this problem, as for most problems in practical applications the exact analytical solution is not known. An estimate of the discretization error ($E\epsilon$) can be obtained from comparing the results obtained on two grids with different mesh size h, and $H = 2h$:

$$||E\epsilon||_2 = \frac{1}{N+1} \sum_{i=0}^{N} |v^h(x_i) - v^H(x_i)| \tag{4.37}$$

where N is the the number of nodes on the coarsest of the two grids and x_i its grid points which also appear on the fine grid. The error made by the fast evaluation is measured by (4.26), i.e. the difference between the result obtained on the target grid (fl) using simple summation and the result obtained using the MLMIO algorithm with the coarsening carried out until a coarser grid with index cl.

In the tables 4.10, 4.11 and 4.12 is presented the fast evaluation error as an function of fl and cl for the cases $k = 2\pi$, 8π and 32π. The MLMIO algorithm was used with angular transfers ($p_\theta = 2$) and spatial transfers ($p_{xy} = 6$). To compensate for the error introduced by interpolations near the singularity a correction was applied over a distance of $r_c = 3 + \ln(1/h)$ points near $x = y$. The correction factors involved can be precomputed and stored. However, unlike the one dimensional algorithm now this is not a trivial task as matrices for each combination of directions are needed for each combination of a fine and coarse grid appearing in the algorithm. This implies that the corrections will be the computationally most expensive part of the fast evaluation algorithm as will be shown later.

The results presented in tables 4.10, 4.11 and 4.12 show that in all the cases the evaluation can be done at expense of an error that is at most comparable to the discretization error. The

$\|FE\epsilon\|, k = 2\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
4	32	3.21e-02	2.69e-04	1.09e-03	1.09e-03		
5	64	7.87e-03	1.65e-05	1.21e-04	2.60e-04*		
6	128	1.81e-03	1.64e-06	1.32e-05	4.01e-05	7.77e-05	
7	256	4.07e-04	2.51e-06	2.50e-06	6.52e-06	1.96e-05*	
8	512	9.22e-05	5.12e-06	5.06e-06	5.07e-06	5.77e-06	1.07e-05
9	1024	2.20e-05	—	—	5.18e-06	5.21e-06	6.41e-06*
10	2048	5.80e-06	—	—	5.29e-06	5.30e-06	6.30e-06

TABLE 4.10: *Norm of the fast summation error for the numerical evaluation of (4.27) for $k = 2\pi$ using the MLMIO algorithm with $p_\theta = 2$ and $p_{xy} = 6$ order transfers. The total number of points in the finest grid is determined by $(2^{fl+1}+1)^2$. ($k = 2\pi$)*

$\|FE\epsilon\|_2, k = 8\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
5	64	2.38e-02	1.29e-03	1.76e-03	1.71e-03*		
6	128	7.09e-03	1.29e-04	1.30e-03	1.57e-03	1.52e-03	
7	256	1.66e-03	5.28e-06	7.14e-05	5.86e-04	6.14e-04*	
8	512	3.67e-04	1.02e-06	2.82e-06	2.19e-05	1.19e-04	8.27e-05
9	1024	7.96e-05	—	—	1.76e-06	5.67e-06	2.41e-05*
10	2048	1.76e-05	—	—	1.45e-06	1.54e-06	2.07e-06

TABLE 4.11: *Norm of the fast summation error for the numerical evaluation of (4.27) for $k = 8\pi$ using the MLMIO algorithm with $p_\theta = 2$ and $p_{xy} = 6$ order transfers. The total number of points in the finest grid is determined by $(2^{fl+1}+1)^2$. ($k = 8\pi$)*

$\|FE\epsilon\|_2, k = 32\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
7	256	5.511e-03	4.14e-04	2.23e-04	3.36e-04	4.39e-04*	
8	512	1.669e-03	3.11e-05	3.59e-04	2.57e-04	2.52e-04	2.58e-04
9	1024	3.899e-04	—	—	2.07e-04	1.83e-04	1.81e-04*
10	2048	8.488e-05	—	—	7.29e-06	6.41e-05	3.30e-05

TABLE 4.12: *Norm of the fast summation error for the numerical evaluation of (4.27) for $k = 32\pi$ using the MLMIO algorithm with $p_\theta = 2$ and $p_{xy} = 6$ order transfers. The total number of points in the finest grid is determined by $(2^{fl+1}+1)^2$. ($k = 32\pi$)*

Cpu time, $k = 2\pi$							
fl	Mesh	Dir. Summ.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
4	32	2.00e-02	2.80e-01	2.10e-01	2.30e-01		
5	64	1.50e-01	1.84e+00	9.40e-01	9.60e-01*		
6	128	2.32e+00	1.51e+01	4.21e+00	3.71e+00	3.84e+00	
7	256	5.19e+01	1.76e+02	2.90e+01	1.85e+01	1.89e+01*	
8	512	8.65e+02	2.38e+03	2.44e+02	1.10e+02	9.58e+01	9.94e+01
9	1024	1.61e+04	—	—	5.13e+02	3.88e+02	3.88e+02*
10	2048	2.54e+05	—	—	4.01e+03	2.09e+03	2.00e+03

TABLE 4.13: *Cpu time for the numerical evaluation of (4.27) using the MLMIO algorithm with $p_\theta = 2$ and $p_{xy} = 6$ order transfers. The total number of points in the finest grid is determined by $(2^{fl+1} + 1)^2$.*

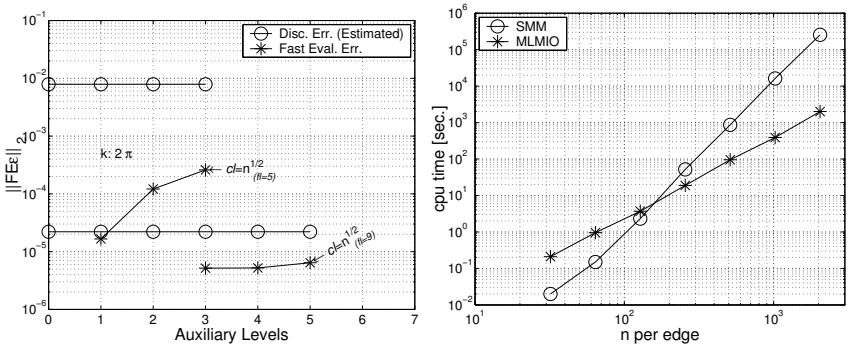


FIGURE 4.8: *Left: Fast summation error as a function of the number of auxiliary levels used in MLMIO algorithm for the evaluation of (4.27). Right: Cpu time invested in the numerical evaluation. ($k = 2\pi$)*

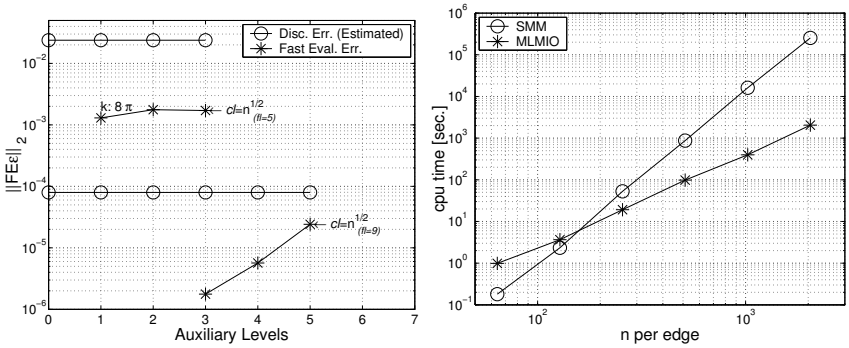


FIGURE 4.9: *Left: Fast summation error as a function of the number of auxiliary levels used in MLMIO algorithm for the evaluation of (4.27). Right: Cpu time invested in the numerical evaluation. ($k = 8\pi$)*

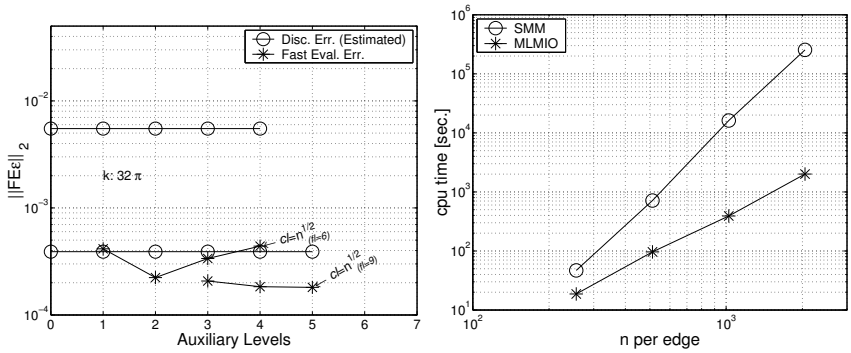


FIGURE 4.10: *Left: Fast summation error as a function of the number of auxiliary levels used in MLMIO algorithm for the evaluation of (4.27). Right: Cpu time invested in the numerical evaluation. ($k = 32\pi$)*

results marked with (*) indicate the cases where the coarsening is continued to a coarse grid with $\sqrt{n}$ points.

In table 4.13 the computing time needed to obtain the results is shown. From the first column of this table it can be seen that with direct summation each mesh refinement leads to a 16 times larger cpu time. Using the MLMIO algorithm a cpu time reduction by more than 2 orders of magnitude is obtained for the finest grids. From comparing the results for different wavenumbers it is clear that the computing time reductions obtained are independent of the wavenumber.

From comparing the results marked with an (*) it can be seen that the invested cpu time is not yet linear in the number of nodes. In [14] it is estimated that an $O(np^d \log n)$ complexity should be obtainable. This may indeed be the limit for very fine grids but the cpu time is strongly influenced by the corrections that have to be applied. This is an expensive task as each additional coarsening the number of directions increases by a factor 2 and corrections have to be computed for each combination of directions and coarse and fine grid. For efficiency it is crucial that only those correction coefficients are computed that are actually used, and that coefficients for directions that do not contribute are not computed.

As mentioned in chapter 3 in the algorithm three parameters are appear, the order of angular interpolation, the order of interpolation in space and the number of correction points. In table 4.14 the computing time of used by the algorithm when no corrections are applied is given. From comparing these computing time results with the results presented in table 4.13 it is clear that to further reduce the cpu time it is most profitable to invest in a less robust but equally accurate correction procedure.

Nevertheless, from the results presented here it is clear that these are the first results showing that the generalized Multilevel Multi-Integration algorithm using the concept of separation of directions as proposed by Brandt [14] indeed works for a two dimensional problem. The discrete integral transform can be obtained much faster at the expense of a controllable error. Moreover, the efficiency gain is independent of the wavenumber. In the following section results of its application to a model problem in which the oscillatory behaviour appears implicitly in the kernel will be presented.

Cpu time, $k = 2\pi$							
fl	Mesh	Dir. Summ.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
4	32	2.00e-02	1.80e-01	3.00e-02	2.00e-02		
5	64	1.50e-01	1.33e+00	2.60e-01	5.00e-02*		
6	128	2.32e+00	1.33e+01	1.35e+00	2.30e-01	1.10e-01	
7	256	5.19e+01	1.67e+02	1.34e+01	1.52e+00	4.30e-01*	
8	512	8.65e+02	2.33e+03	1.70e+02	1.42e+01	2.30e-01	1.34e+00
9	1024	1.61e+04	—	—	1.65e+02	1.66e+01	4.95e+00*
10	2048	2.54e+05	—	—	2.25e+03	1.85e+02	3.36e+01

TABLE 4.14: *Cpu time for the numerical evaluation of (4.27) using the MLMIO algorithm with $p_\theta = 2$ and $p_{xy} = 6$ order transfers with the time invested for the corrections subtracted. The total number of points in the finest grid is determined by $(2^{fl+1} + 1)^2$.*

4.4 Implicit Asymptotically-oscillatory 2D problem

In problems of practical interest (e.g. acoustics and electromagnetics) integral transforms with kernels in which the asymptotically smooth and oscillatory behaviour are implicit, i.e. the function can not straightforwardly be written as the product of two functions each representing one aspect of the behaviour. However, the MLMIO algorithm can easily be made to work for such cases too as will be shown below.

The model problem is defined as the evaluation of:

$$v(x) = \int_\Omega H_0^2(k|x-y|)\, u(y)\, dy, \qquad x \in \Omega \tag{4.38}$$

with $\Omega \in [-1,1] \times [-1,1]$, and

$$u(y) \equiv u(y^1, y^2) = \sin(2\pi\, y^1)\ \sin(2\pi\, y^2), \quad y \in \Omega \tag{4.39}$$

where $H_0^2(k|x-y|)$ represents the Hankel function of the second kind and order zero.

The Hankel function (4.38) is an implicit combination of an oscillatory and asymptotically smooth function. The problem (4.38) can be written in the generic form (3.28) by multiplying the original kernel with $e^{ik|x-y|}\ e^{-ik|x-y|}$ such that,

$$v(x) = \int_\Omega \left[H_0^2(k|x-y|)\, e^{ik|x-y|}\right] e^{-ik|x-y|}\, u(y)\, dy \tag{4.40}$$

where the meaning of the original integral transform (4.38) does not change. The original task can now be replaced by:

$$v(x) = \int_\Omega G(x,y) e^{-ik|x-y|}\, u(y)\, dy \tag{4.41}$$

with

$$G(x,y) = H_0^2(k|x-y|)\, e^{ik|x-y|} \tag{4.42}$$

and $u(y)$ represented by (4.39), for different values of k.

Like the previous example, in the results presented below, values of $k = 2\pi$, 8π and 32π were used for the evaluation of the model problem.

4.4.1 Discretization

i) Single Matrix Multiplication

Assuming the product $e^{-ik|x-y|}u(y)$ be constant on each integration interval of the mesh size $h \times h$ centered around of the point y_j one obtains a discrete second order approximation to (4.41) given as:

$$v^h(x_i) = \sum_j G^{hh}(x_i, y_j)\, e^{-ik|x_i - y_j|}\, u^h(y_j) \tag{4.43}$$

where

$$\begin{aligned} G^{hh}(x_i, y_j) \equiv G^{hh}(x_i^1, x_i^2, y_j^1, y_j^2) \quad = \quad & \int_{y_j^1 - \frac{h}{2}}^{y_j^1 + \frac{h}{2}} \int_{y_j^2 - \frac{h}{2}}^{y_j^2 + \frac{h}{2}} H_0^2 \left(k[(x_i^1 - y^1)^2 + (x_i^2 - y^2)^2]^{\frac{1}{2}} \right) \\ & \times e^{ik[(x_i^1 - y^1)^2 + (x_i^2 - y^2)^2]^{\frac{1}{2}}} dy^1 dy^2 \end{aligned} \tag{4.44}$$

and

$$u(y_j) \equiv u(y_j^1, y_j^2) = \sin(2\pi\, y_j^1)\ \sin(2\pi\, y_j^2) \tag{4.45}$$

The coefficients $G^{hh}(x_i, y_j)$ were computed using a Gauss adapative quadrature with predefined accuracy $\epsilon = O(10^{-8})$.

ii) Separation of Directions (MLMIO)

Using the description from the Section 3.3, the second discrete formulation of (4.41) is obtained in terms of *separation of directions* and it is represented by:

$$v^h(x_i) = \sum_l v_l^h(x_i) \tag{4.46}$$

where

$$v_l^h(x_i) = \sum_j G^{hh}(x_i, y_j)\, e^{-ik|x_i - y_j|}\, w_l^{s_{ij}}(e_{ij}) \sum_{m \in s_{ij}} w_m^{s_{ij}}(e_{ij})\, u_m^h(y_j) \tag{4.47}$$

with

$$u_m^h(y_j) = u^h(y_j) \tag{4.48}$$

where $v_l^h(x_i)$ represents each of the subtransforms defined by (3.37). Note that because the kernel of the original integral transform (4.38) was replaced by an explicit product of the both asymptotically smooth and oscillatory functions the evaluation of the coefficients $G^{hh}(x_i, y_j)$ can now be done using (4.44).

4.4.2 Performance

The performance is illustrated in the same way as for the previous example. As the exact analytical solution is not known the fast evaluation error defined by (4.26) is compared with the estimate of the discretization error as defined by (4.37). As a reminder, the fast evaluation error is the difference between the result obtained on the target grid (fl) using simple summation and the result obtained on this grid using the MLMIO algorithm with the coarsening carried out until a coarser grid with index cl.

In the tables 4.15, 4.16 and 4.17 the fast evaluation error is presented as a function of fl and cl for the case $k = 2\pi$, 8π and 32π. The MLMIO algorithm was used with angular transfers ($p_\theta = 2$) and spatial transfers ($p_{xy} = 6$). The correction was carried out over a region of $r_c = 3 + \ln(1/h)$ points around the singularity at $x = y$.

Tables 4.15, 4.16 and 4.17 show the fast evaluation error ontained with the MLMIO algorithm, where for all the cases the error is at most comparable to the discretization error. The results marked with (*) indicate the cases where the coarsest auxiliary grid reached has $\sqrt{n}$ points.

$\|\|FE\epsilon\|\|_2, k = 2\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
4	32	3.085e-03	8.70e-05	4.16e-04	4.16e-04		
5	64	7.017e-04	3.87e-06	3.94e-05	9.96e-05*		
6	128	1.658e-04	2.54e-07	2.69e-06	1.14e-05	2.76e-05	
7	256	4.053e-05	4.19e-07	4.11e-07	1.27e-06	5.24e-06*	
8	512	1.009e-05	8.95e-07	8.56e-07	8.56e-07	1.05e-06	2.74e-06
9	1024	2.669e-06	—	—	8.57e-07	8.61e-07	1.05e-06*
10	2048	7.276e-07	—	—	—	8.94e-07	9.01e-07

TABLE 4.15: *Norm of the fast summation error for 2D problem with Hankel function as kernel. The total number of points on the finest grid is determined by* $(2^{fl+1}+1)^2$. *(MLMIO,* $k = 2\pi$*)*

$\|\|FE\epsilon\|\|_2, k = 8\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
5	64	5.906e-04	2.60e-05	6.88e-05	9.55e-05*		
6	128	9.683e-05	7.12e-07	5.90e-06	3.91e-05	6.77e-05	
7	256	1.692e-05	5.67e-08	1.05e-07	4.21e-06	1.11e-05*	
8	512	3.512e-06	4.05e-08	6.86e-08	9.77e-08	5.47e-07	1.12e-05
9	1024	8.434e-07	—	—	4.67e-08	7.15e-08	7.40e-07*
10	2048	2.156e-07	—	—	—	2.64e-08	5.26e-08

TABLE 4.16: *Norm of the fast summation error for 2D problem with Hankel function as kernel. The total number of points on the finest grid is determined by* $(2^{fl+1}+1)^2$. *(MLMIO,* $k = 8\pi$*)*

$\|\|FE\epsilon\|\|_2, k = 32\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
7	256	3.813e-05	2.45e-06	3.54e-06	8.08e-06	8.301e-06*	
8	512	6.014e-06	8.78e-08	5.67e-07	9.96e-07	3.483e-06	3.680e-06
9	1024	9.695e-07	—	—	1.91e-07	7.324e-07	3.670e-06*
10	2048	1.752e-07	—	—	—	9.362e-08	3.347e-07

TABLE 4.17: *Norm of the fast summation error for 2D problem with Hankel function as kernel. The total number of points on the finest grid is determined by* $(2^{fl+1}+1)^2$. *(MLMIO,* $k = 32\pi$*)*

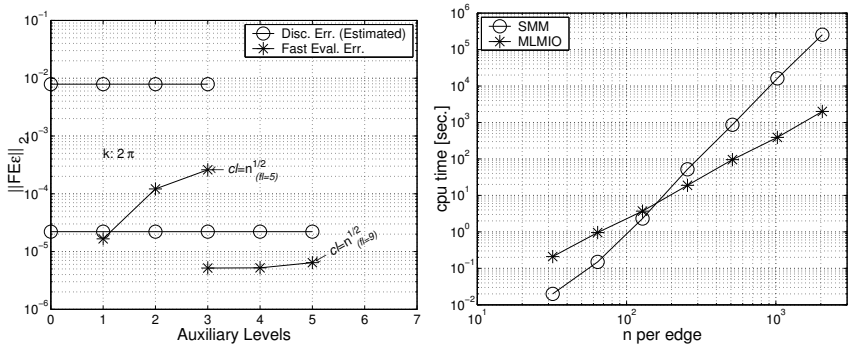


FIGURE 4.11: *Left: Fast summation error in the numerical evalaution of (4.41) using the MLMIO algorithm with* $p_\theta = 2$ *and* $p_{xy} = 6$ *order transfers. Right: Cpu time for the numerical evaluation of the 2D problem with Hankel functionas kernel. (*$k = 2\pi$*)*

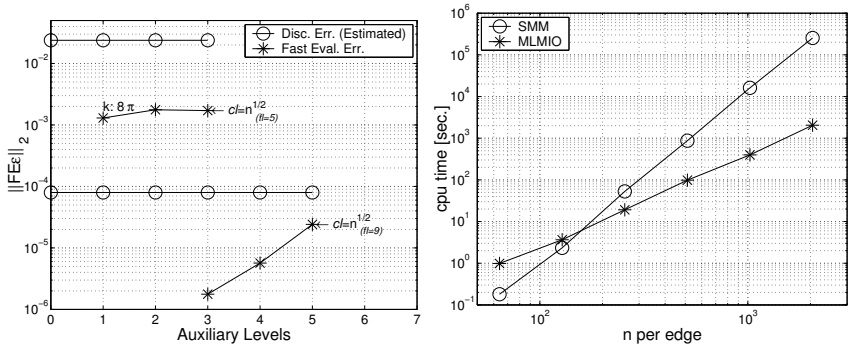


FIGURE 4.12: *Left: Fast summation error in the numerical evalaution of (4.41) using the MLMIO algorithm with* $p_\theta = 2$ *and* $p_{xy} = 6$ *order transfers. Right: Cpu time for the numerical evaluation of the 2D problem with Hankel functionas kernel. (*$k = 8\pi$*)*

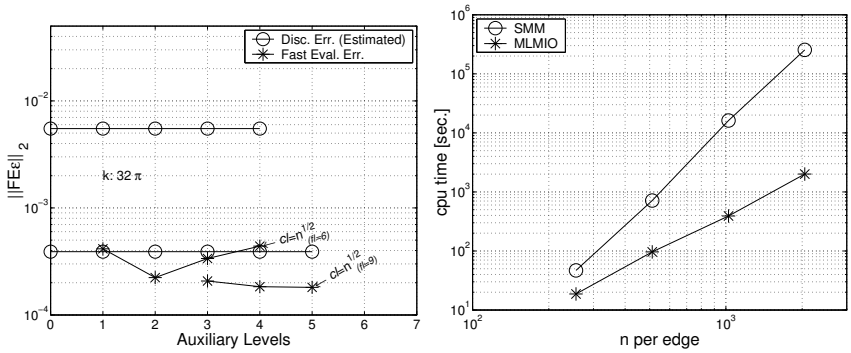


FIGURE 4.13: *Left: Fast summation error in the numerical evalaution of (4.41) using the MLMIO algorithm with $p_\theta = 2$ and $p_{xy} = 6$ order transfers. Right: Cpu time for the numerical evaluation of the 2D problem with Hankel functionas kernel. ($k = 32\pi$)*

The computing time needed for the single matrix multiplication and the MLMIO algorithm in the numerical evaluation of (4.41) is the same as shown in the table 4.13, and it is graphically illustrated by the graphs on the right side of the Figures 4.11, 4.12 and 4.13.

From the graphs one can see that the MLMIO yields significant cpu time reductions for large n. However, as was mentioned above, it is not immediately clear if it follows a linear tendency as a function of the number of nodes. As it was mentioned in [14] and shown in section 3.4, the most computationally expensive stage in the algorithm is to perform the corrections. Since the corrections are a function of: the number of directions on each coarsening level, the p_θ (order of the angular transfers), and the p_{xy} (order of the spatial transfers), an optimal combination of them should be found in order to use most efficiently the cpu time needed in the numerical evaluation. This issue is a topic of ongoing and future research.

4.5 FFT and MLMIO

As explained in chapter 1 several methods are in use for the fast evalutation of integral transforms. In particular FFT due to its simplicity and the availability of actual codes [55] is very popular. A commonly cited disadvantage of FFT in relation with boundary element methods and integral transforms is that it can only be used on flat domains covered with a uniform grid. Multilevel Multi-Integration is more generally applicable. Nevertheless it is interesting to see how its performance compares with the performance of FFT for a uniform grid. For a model problem from contact mechanics described by an integral transform with an asymptotically smooth kernel results were presented by Lubrecht et al. [25]. In this section some results for the case of an oscillatory kernel will be presented. The FFT code used was taken from [55].

Results are presented for each of the problems presented in this chapter, e.g. (4.1) and (4.13) for the one dimensional case, and (4.27) and (4.38) for the two dimensional case respectively. For each problem results for a set of values of the wavenumber are shown in the tables 4.18-4.21.

In each table in the first column the wave number is shown. In the second column of

the tables 4.18 and 4.19 the discretization error for the case of single matrix multiplication is shown. In the third and fourth column are presented the norm of the error in the result obtained using the FFT and MLMIO algorithm respectively. In the bottom row of each table is given the computing time invested to obtain the results.

For tables 4.20 and 4.21 the second column shows the estimate of the discretization error defined by (4.37). The third column represents the estimate of the error obtained in the evaluation using the FFT and the fourth column the error of the fast summation when MLMIO is used. Finally, in the bottom of the tables 4.20 and 4.21 is given the computing time invested in the evaluation of each case using the three methods.

From the tables described above one can see that in all cases using the FFT gives exactly the same result as the single matrix multiplication. This is due to the fact that FFT uses the algebraic properties of the Fourier transform to construct a sparse factorization of the elements of the discrete Fourier transform so there is no loss of information compared to the original entries (matrix elements). For the one dimensional case the fast evaluation error in the MLMIO result (tables 4.18 and 4.19) is also almost identical to the single matrix multiplication result, which shows that the fast evaluation error is small compared to the discretization error. From the last line of the tables it can be seen that both algorithms are equally efficient. In both cases the computing time is $O(n\log(n))$ [20])[25].

The results for the two dimensional cases (tables 4.20 and 4.21) show the same behaviour as for the one dimensional problem. The FFT algorithm exactly yields the single grid direct summation result, and using the MLMIO algorithm the discrete transform is obtained at the expense of a *fast evaluation* error (see section 4.2) that can be kept small compared to the discretization error. Again the performance is independent of the wavenumber. However, in this case clearly the FFT algorithm is faster, e.g. roughly by a factor three. One difference between the two algorithms is that the FFT scales $O(n\log(n))$ independently of the dimension of the problem. For the MLMI algorithm this is not entirely so due to the corrections that are applied, see [25]. However, when a more optimized correction procedure is used it is anticipated that the computing time can be further reduced to the same level of FFT (see table 4.14). Finally, as was explained before, the results of the comparison should not be given too much weight as the MLMIO algorithm is potentially more generally applicable than FFT.

	$\|\|\epsilon\|\|_2$		
k	SMM	FFT	MLMIO
2π	1.182e-09	1.182e-09	1.36e-09
4π	1.109e-09	1.109e-09	6.20e-10
8π	1.089e-09	1.089e-09	3.04e-10
16π	1.084e-09	1.084e-09	1.51e-10
32π	1.083e-09	1.083e-09	7.54e-11
64π	1.082e-09	1.082e-09	3.77e-11
128π	1.082e-09	1.082e-09	1.88e-11
Av. Cpu time:	2.20e-00	2.01e-02	2.00e-02

TABLE 4.18: *Norm of the error for the numerical approximation of (4.1) using: Single Matrix Multiplication (SMM), Fast Fourier Transform (FFT) and Multilevel Multi-Integration Algorithm for Oscillatory kernels (MLMIO). The domain was divided in (*16384*) constant intervals.*

$\|\|\epsilon\|\|_2$			
k	SMM	FFT	MLMIO
2π	7.43e-08	7.43e-08	1.85e-07
4π	1.51e-07	1.51e-07	2.64e-07
8π	3.16e-07	3.16e-07	3.89e-07
16π	6.61e-07	6.61e-07	6.58e-07
32π	1.38e-06	1.38e-06	1.34e-06
64π	2.87e-06	2.87e-06	2.87e-06
128π	5.97e-06	5.97e-06	5.97e-06
Av. Cpu time:	2.30e+00	1.98e-02	2.01e-02

TABLE 4.19: *Norm of the error for the numerical approximation of (4.13) using: Single Matrix Multiplication (SMM), Fast Fourier Transform (FFT) and Multilevel Multi-Integration Algorithm for Oscillatory kernels (MLMIO). The domain was divided in* $(16384 + 1)$ *constant intervals.*

	$\|\|E\epsilon\|\|_2$		$\|\|FE\epsilon\|\|_2$
k	SMM	FFT	MLMIO
2π	4.07e-04	4.07e-04	1.97e-05
4π	7.96e-04	7.96e-04	7.58e-05
8π	1.66e-03	1.66e-03	6.14e-04
16π	3.42e-03	3.42e-03	5.21e-04
32π	5.51e-03	5.51e-03	4.39e-04
Av. Cpu time:	5.19e+01	6.85e+00	1.89e+01

TABLE 4.20: *Estimation of the norm of the discretization error (*$||E\epsilon||_2$*) for the numerical approximation of (4.27) using: Single Matrix Multiplication (SMM) and Fast Fourier Transform (FFT). Norm of the fast summation error (*$||FE\epsilon||_2$*) using Multilevel Multi-Integration Algorithm for Oscillatory kernels (MLMIO). The domain was divided in* $(256)^2$ *squared constant pieces.*

	$\|\|E\epsilon\|\|_2$		$\|\|FE\epsilon\|\|_2$
k	SMM	FFT	MLMIO
2π	1.00e-05	1.00e-05	1.05e-06
4π	4.71e-06	4.71e-06	8.02e-07
8π	3.51e-06	3.51e-06	5.45e-07
16π	3.98e-06	3.98e-06	7.87e-06
32π	6.01e-06	6.01e-06	3.48e-06
Av. Cpu time:	8.65e+02	2.21e+00	9.58e+01

TABLE 4.21: *Estimation of the norm of the discretization error (*$||E\epsilon||_2$*) for the numerical approximation of (4.38) using: Single Matrix Multiplication (SMM) and Fast Fourier Transform (FFT). Norm of the fast summation error (*$||FE\epsilon||_2$*) using Multilevel Multi-Integration Algorithm for Oscillatory kernels (MLMIO). The domain was divided in* $(512)^2$ *squared constant pieces.*

APPLICATIONS

CHAPTER 5

As an illustration of an application in this chapter results are presented for two model problems of radiation acoustics in two dimensions where, given the velocity of a vibrating object the pressure field is computed from the Helmholtz integral equation using a Boundary Element Method and where the MLMIO algorithm for the two dimensional case is used for the evaluation of the integral transform.

5.1 Helmholtz integral equation

As was shown in chapter 2 the pressure on the surface and in the fluid around of a vibrating body can be solved from the so-called Helmholtz integral equation (2.37) with a Neumann boundary condition which can rewritten as:

$$c(x)p(x) = \int_S p(y)\frac{\partial G(x,y)}{\partial n_y}dS + i\rho_0\omega\int_S G(x,y)v(y)dS \tag{5.1}$$

with

$$c(x) = \begin{cases} 1 & , x \text{ exterior to S} \\ \frac{1}{2} & , x \text{ on S} \\ 0 & , x \text{ interior to S} \end{cases} \tag{5.2}$$

where $p(x)$ is the acoustical pressure, $v(y)$ the normal velocity on S, and $G(x,y)$ the free space fundamental solution (Green's function) which for two dimensional cases is defined by:

$$G(x,y) = \frac{i}{4}H_0^2(k|x-y|) \tag{5.3}$$

with normal derivative,

$$\frac{\partial G(x,y)}{\partial n_y} = -\frac{ik(x_\beta - y_\beta)n_\beta}{4|x-y|}H_1^2(k|x-y|) \tag{5.4}$$

$H_1^2(k|x-y|)$ is the Hankel function of first order and second kind. β represents the index of the coordinate system and n_β the corresponding normal component on S. Note that these functions are finite as $r \to 0$, and they lead to singularities of order $\ln(r)$ for $H_0^2(kr)$ and $1/r$ in case of $H_1^2(kr)$ with $r = |x-y|$.

In section 2.3 it was mentioned that in order to solve p in the domain two steps are required:

1. Solve the equation for pressure $p(x)$ on the surface using,

$$c(x)p(x) - \int_S p(y)\frac{\partial G(x,y)}{\partial n_y}dS = i\rho_0\omega \int_S G(x,y)v(y)dS \tag{5.5}$$

2. Given the pressure $p(y)$ on the surface, evaluate the pressure $p(x)$ at any location in the field using (5.1).

As was shown by Schenck[62] solving the surface Helmholtz integral equation non-uniqueness problems of the solution have to be overcome related to the natural frequencies of the associated interior Dirichlet problem. Several methods have been proposed to overcome these problems:
The Combined Helmholtz Integral Equation Formulation (CHIEF)

Schenck [62] proposes to remove the nonuniquense by introducing additional Helmholtz integral relations evaluated in the interior of the domain. Applying a discretization of the radiating surface to the CHIEF equations leads to an over determined system of the equations for the surface pressure which can be solved in the last squares sense. However, a potential complication of this method is the choice of the interior points at which the supplemental equations are defined such that to any possible distortion of the solution due to the interior critical frequencies is avoided.

The hypersingular formulation

This method was introduced by Burton and Miller [21] and has been widely studied. The method is based in the generation of an integral equation valid for all wavenumbers by forming a linear combination of the Helmholtz integral relations used by CHIEF and its normal derivative. However, a complication in this approach is the evaluation of the hyper-singular integrals involving a double normal derivative of the free space Green's function. Most recently efforts to improve the approach have been investigated developing efficient methods to evaluate the hyper singular integrals, see [5].
Visser [74] proposes a combination of the CHIEF and the hyper-singular formulation, referring to it as *Combined Interior Burton-Miller Formulation* (CIBMF).

In the cases shown in this chapter the CHIEF method was used.

5.2 Direct BEM discretization

The Boundary Element Method is a most suitable numerical approach for the evaluation of exterior acoustical problems using the direct formulation of the Helmholtz integral equation (5.1), e.g. see [22].

In the direct BEM method, the first step is to derive from the continuous equation a discrete system of equations from which the unknowns at specific points at the boundary can be solved. The boundary of the body is divided into N segments where the variables p (pressure) and v (normal velocity) are approximated on the surface by locally based functions $p(y)$ and $v(y)$ defined on each surface element (panel).

So, (5.1) can be now represented as a summation over all surface elements yielding the following discrete formulation:

$$c(x)p(x_i) = \sum_j \int_{S_j} p(y) \frac{\partial G(x,y)}{\partial n(y)} dS_j + i\rho_0\omega \sum_j \int_S G(x,y)v(y)dS_j \tag{5.6}$$

where i represents nodal points and S_j is the surface of the element j on the boundary, and $c(x) = 1/2$ due that the nodes are placed on a smooth surface.

Here on each element of the surface p and v are assumed to be constant. In this case they can be taken out of the integral so the (5.1) can be rewritten as:

$$\frac{1}{2}p(x_i) = \sum_j p(y_j) \left(\int_{S_j} \frac{\partial G(x_i, y_j)}{\partial n_j} dS_j \right) + i\rho_0\omega \sum_j v(y_j) \int_{S_j} G(x_i, y_j) dS_j \tag{5.7}$$

For the solution of the pressure at the surface (5.1) is applied to each point i which is also at the surface. This yields a full (densely populated) system of equations. The elements of the influence matrices can be defined as:

$$G_{ij} = \int_{S_j} G(x_i, y_j) dS_j \qquad H_{ij} = \int_{S_j} \frac{\partial G(x_i, y_j)}{\partial n_j} dS_j \tag{5.8}$$

So, the discrete represetation of (5.1) is given by:

$$\frac{1}{2}p(x_i) - \sum_j H_{ij} p(y_j) = \sum_j G_{ij} v(y_j) \tag{5.9}$$

Since the discretization is done with p and v constant at the elements, $H_{ij} = 0$ for $i = j$ because the normal is always perpendicular to the orientation of the element. So that the diagonal matrix can be filled only with the value of the constant $c(x)$ which in this case is $\frac{1}{2}$.

The linear system of equations approximating the continous integral equation is given by:

$$\mathbf{Hp} = \mathbf{Gv_n} \tag{5.10}$$

where $\mathbf{v_n}$ is the vector containing the values of the normal velocity on the surface. The coefficients of the influence matrices $\mathbf{G}$ and $\mathbf{H}$ are known and follow from the evaluation of the discrete fundamental solution (Green's function).

Introducing the boundary condition, i.e. Neumann condition when the particle velocity is known on the surface of the vibrating body, reordering with respect to known and unknown values enables us to write this system as:

$$\mathbf{Ap} = \mathbf{b} \tag{5.11}$$

where the sound pressure $\mathbf{p}$ is the unknown. The vector $\mathbf{b}$ is known from the boundary conditions. So, the values of p can be determined by solving the system of equations (5.11). For the model cases presented in this thesis a standard LU decomposition procedure was used in order to solve the system of the equations (5.11) [55]. The only purpose here is to test the fast evaluation algorithm. However, the computing time required for the solution using

LU decomposition is $O(n^3)$ if n is the number of elements. Efficient solution techniques are obviously important for practical applications. So far efficient iterative techniques as the Conjugate Gradient Method (CGM) and generalized Minimal Residual (GMRes) [43] have been widely studied and can significantly alleviate the intensive computational work needed to solve (5.11) in practical application with large number of unknowns. One of the aims in future research is also the application of Multgrid/Multilevel techniques.

5.2.1 Fast evaluation of the integral transforms using BEM with MLMIO

The evaluation of the right hand of (5.10) requires mutiplying a fully populated matrix by a vector, which using single matrix-vector multiplication involves $O(n^2)$ operations. This leads to large computing times for large n. The MLMIO algorithm was developed exactly for the purpose of faster evaluation of this task. Its prospect was illustrated for model problems in chapter 4. In the next sections results for two model problems of acoustic radiation in two dimensions will be shown.

5.3 Numerical cases

Two model problems were used to investigate the performance of the MLMIO algorithm for the evaluation of the discrete integral transforms that appear in a BEM formulation of an acoustic radiation problem. These cases serve to obtain an indication of the performance of the MLMIO algorithm when it is part of a BEM formulation.

In the model problems shown below two quantities are considered to evaluate the performance of the evaluation, the error of the approximation and the computing time invested for both, the single matrix-vector multiplication and using the MLMIO algorithm in the evaluation of the matrix-vector multiplication $\mathbf{Gv_n}$ from (5.10).

The error is evaluated using a similar definition as (4.17):

$$||\epsilon||_2 = \frac{1}{n}\sum_{i=1}^{n}|p(x_i) - p^h(x_i)| \tag{5.12}$$

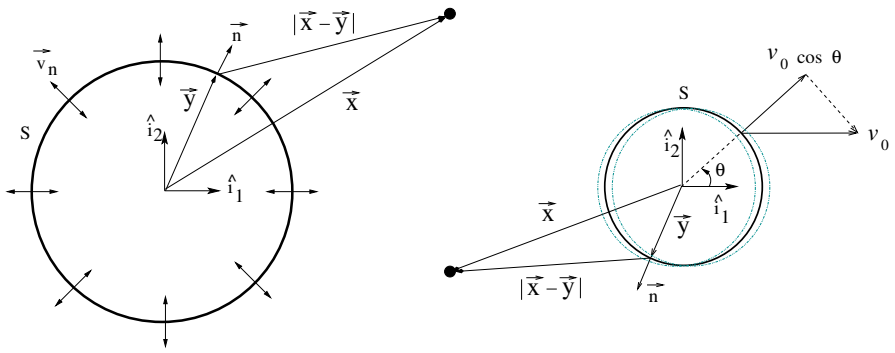
where $p(x_i)$ and $p^h(x_i)$ are the exact analytical solution and the approximation of the solution using BEM respectively, at the point x_i. n is the number of constant elements used in the boundary discretization.

5.3.1 Infinite pulsating cylinder

Consider the task of evaluating the pressure around an infinity long pulsating cylinder with radial a normal velocity (v_n) as showing in the Figure 5.1-a. The sound field satisfies the radiation condition which means that no incoming waves can be generated from infinity.
The exact analytical solution is given as [7]:

$$p(x) = i\rho c v_n \frac{H_0^2(kx)}{H_1^2(ky)} \tag{5.13}$$

where the H_1^2 is defined as the Hankel function of the second kind and first order. x is the distance from the center of the source to any point located in the open domain, y is the radius of the cylinder, v_n is the normal velocity on the surface of the cylinder, i is the imaginary



a) Uniformly pulsating cylinder. b) Vibrating wire.

FIGURE 5.1: *Schematic representation of two physical acoustical radiation problems.*

constant ($\sqrt{-1}$), ρ and c are the properties of the fluid. In this case they are assumed to be unity.

5.3.2 Infinite vibrating wire

This test case consists of the evaluation of the pressure field generated by a vibrating wire with velocity v_0 as shown in Figure 5.1-b. This problem also satisfies the radiation condition for unbounded domain.
The exact analytical solution is represented by [7]:

$$p(x) = -i\rho_0 c v_0 \cos\theta \frac{H_1^2(kx)}{H_1^2(ka)} \tag{5.14}$$

again the properties of the fluid are assumed to be one.

CONCLUSIONS AND RECOMMENDATIONS

CHAPTER 6

6.1 Conclusion

A low noise level is a strategic quality for products ranging from personal appliances to aeroplanes. Reliable and efficient methods and computational tools to design silent products and to identify sources of noise are urgently needed. Impressive results have been obtained with models and computer simulations to predict noise for specific problems. However, it is still a long way towards efficient and reliable computational tools for the design of silent products in many practical applications. A crucial bottleneck is that the required computational effort is often far too large for extensive use of the computer simulations in design.

In this thesis an alternative algorithm for the fast numerical evaluation of integral transforms with oscillatory Green's functions as they appear in acoustic radiation problems has been developed. The algorithm is the generalization of the Multilevel Multi-Integration algorithm which was introduced and proven to be very efficient for transforms with smooth and asymptotically smooth Green's functions by Brandt and Lubrecht [16]. In the generalized algorithm, using the concept of separation of directions and the smoothness of the oscillatory part as a function of the phase angle the original transform is rewritten as a special case of the more generalized task of the evaluation of a series of transforms each with an asymptotically smooth Green's function. The smoothness of these functions can then be used to replace the original task on the target grid by a series of intergrid transfers and a coarse grid discrete transform. This concept was outlined by Brandt [14] over a decade ago but no numerical results were presented so far. In this thesis it has been demonstrated that the concept really works. A careful step by step approach was adopted starting with one dimensional model problems for which the concept is straightforward. Subsequently, the much more involved two dimensional problem was tackled. Also for this case results were presented for transforms with different Green's functions including the Hankel function which appears in many practical applications. It has been shown that the algorithm yields an accurate approximation to the discrete integral transform with an error that can be kept comparable to the discretization error that is made anyway, in a computing time that is much smaller. The performance is independent of the wavenumber which is crucial as in practice the computing time problems are most urgent for high frequencies where large numbers of nodes are required. The required cpu time is reduced from $O(n^2)$ to $O(np^d \log n)$ operations. So far only uniform grids were used. The algorithm was benchmarked against the popular Fast Fourier Transform which also has a $O(n \log n)$ complexity but is restricted to uniform grids on flat domains. For

the two dimensional problem the required cpu time is about three times larger than used by an FFT algorithm. However, the main goal of this thesis was to prove that the concept works and yields a fast alternative algorithm. Optimization of the computation of the a posteriori corrections will most certainly lead to an algorithm that for uniform grids is fully competitive with FFT. Besides, the uniform grid case is only for testing. The algorithm has the prospect that it can be used for non-uniform grids and grids on curved surfaces.

As a first step towards implementation in Boundary Element Methods for acoustic radiation problems in chapter 5 some first results obtained with a BEM algorithm obtained for some accoustic model problems were presented.

6.2 Recommandations for further research

The evaluation of integral transforms with oscillatory kernels using the multilevel multi-integration algorithm involves the following tasks: coarsening (spatial anterpolation and angular interpolation), fast summation (multisummation on the coarsest level), refining (spatial interpolation and angular anterpolation), and correction (involving directions, angular and spatial interpolations).

As was shown in the chapters three and four the computationally most expensive task in the algorithm is the computation of the a posteriori corrections related to the error introduced in the region around the singularity where the auxiliary kernels are not smooth. A detailed optimization of the different parameters in the algorithm ,i.e. the angular interpolation (p_θ), the spatial interpolation (p_{xy}) and the size of the correction r_c, e.g. by carrying out an optimization analysis as done in [16] and [17] can be carried out to further optimize the algorithm and to reduce the size of the correction region. Also, in the present implementation the correction matrices are computed by applying the algorithm itself in a local form. The advantage of this approach is that it is very robust as it allows correction for any approximation made in the algorithm, including some of the assumptions of angular smoothness of variables in the coarsening procedure. It thus allows correction to zero error which is important in a development stage. However, various cheaper alternatives should be investigated such as correction based on the spatial interpolation error in the auxiliary kernels only or the use alternative of kernel softening around the singularity. This will lead to further reductions of computing time and the use of memory involved in storing the correction coefficients.

Another option for further development is the following. In the algorithm as described in chapter 3 the spatial coarsening is the same for each direction y^1 and y^2 (or x^1 and x^2). However, as can be seen from the auxiliary kernels $G_{ijlm}(x, y)$ for a given l this kernel is much smoother in the direction m on the grid than in the direction perpendicular to m. Hence, when represented on a grid aligned with this direction it can be represented with the same accuracy using a larger meshsize in the direction of alignment. The idea is now to introduce spatial grids aligned with the different directions on the angular grid. Subsequently in the coarsening these grids can be coarsened faster in the direction of alignment than in the cross direction. As a result the total number of nodes on the all coarser grids will decrease faster than 2^{d-1} which can lead to an even more efficient algorithm. A similar approach has proven successful in the development of a multigrid solver for the Helmholtz equation in differential form developed, see [15].

However, before exploiting such possibilities it is important that the present algorithm finds

its way towards applications to help alleviate computing time problems. In this respect it is important that the extension to three dimensional problems is implemented first as well as the extension to curved surfaces.

As shown in chapter 5 the evaluation of the integral transform is usually a subtask in the process of solving the integral equation. In principle the algorithm presented here can already be used for this purpose in combination with any type of solver. However, as has been shown in research in contact mechanics and lubrication problems there is an enormous prospect of efficiency increase when also for the solution of the integral equation multigrid techniques are used, see [70]. Multigrid techniques provide many options to further reduce the computational effort in acoustic radiation problems also for the inverse problem where complications arising from regularization may be overcome by appropriate use of the different scales. In this respect the research presented in this thesis is a crucial first step.

REFERENCES

[1] M. Abramowitz and I.A. Segun, *Handbook of mathematical functions*, Dover Publications, Inc., New-York, USA, 1968.

[2] S. Amini and S.M. Kirkup, Solution of the Helmholtz equation in the exterior domain by elementary boundary integral methods, *Journal of Computational Physics*, 118 (1995) 208-221.

[3] L. Baker, *C mathematical function handbook*, McGraw-Hill, USA, 1992.

[4] G.K. Batchelor, *An introduction to fluid dynamics*, Cambridge University Press, New York, USA, 2000.

[5] W. Benthien and A. Schenck, Nonexistence and nonuniqueness problems associated with integral equation methods in acoustics, *Computers and Structures*, 65 (1997) 295-305.

[6] R.B. Bird, W.E. Stewart, and E.N. Lightfoot, *Transport Phenomena*, John Wiley and Sons, New York, USA, 1960.

[7] D.T. Blackstock, *Fundamentals of physical acoustics*, John Wiley and Sons, New York, USA, 2000.

[8] J. Barnes and P. Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, *Nature*, 324 (1986) 446-449.

[9] A. Boag, E. Michielssen, and A. Brandt, Nonuniform polar grid algorithm for fast field evaluation, *IEEE Antennas and Wireless Propagation Letters*, 1 (2002) 142-145.

[10] A. Boag, A fast multilevel domain decomposition algorithm for radar imaging, *IEEE Transaction on Antennas and Propagation*, 49 (2001) 666-671

[11] A. Boag, A fast iterative physical optics (FIPO) algorithm based on non-uniform polar grid interpolation, *Microwave and Optical Technology Letters*, 35 (2002) 240-244.

[12] J. Board and K. Schulten, The fast multipole algorithm, *Computing in Science and Engineering*, 2 (2000) 76-79.

[13] J. Bos, *Frictional heating of tribological contacts*, University of Twente, PhD thesis, Enschede, The Netherlands, 1995.

[14] A. Brandt, Multilevel computations of integral transforms and particle interactions with oscillatory kernels, *Computer Physics Communications*, 65 (1991) 24-38.

[15] A. Brandt and I. Livshits, Wave-ray multigrid method for standing wave equations, *Electronic Transactions on Numerical Analysis*, 6 (1997) 162-181.

[16] A. Brandt and A.A. Lubrecht, Multilevel matrix multiplication and fast solution of integral equations, *Journal of Compuational Physics*, 90 (1990) 348-370.

[17] A. Brandt and C.H. Venner, Multilevel evaluation of integral transforms with asymptotically smooth kernels, *SIAM Journal on Scientific Computing*, 19 (1998) 468-492.

[18] A. Brandt and C.H. Venner, Multilevel evaluation of integral transforms on adaptive grids, *Gauss Center Report WI/GC-5*, Rehovot, Israel, 1996.

[19] W.L. Briggs, V.E. Henson, and S.F. McCormick, *A multigrid tutorial*, SIAM, Philadelphia, USA, 2000.

[20] E.O. Brigham, *The fast Fourier transform and its applications*, Prentice Hall, New Jersey, USA, 1988.

[21] A.J. Burton and G.F. Miller, The application of integral equation methods to the numerical solution of some exterior boundary-value problems, *Proceedings of the Royal Society of London. Serie A, Mathematical and Physical Sciences*, 323 (1971) 201-210.

[22] R.D. Ciskowski and C.A. Brebbia, *Boundary element method in acoustics*, Wessex Institue of Technology, Southapton, UK, 1990.

[23] R. Coifman, V. Rokhlin, and S. Wandzura, The fast multipole method for the wave equation: A Pedestrian Prescription, *IEEE Antennas and Propagation*, 35 (1993) 7-12.

[24] J.W. Cooley and J.W. Tukey, An algorithm for the machine computation of complex Fourier series, *Mathematics of Computations*, 19 (1965) 297-301.

[25] F. Colin and A.A. Lubrecht, Comparison of FFT-MLMI for elastic deformation calculations, *Journal of Tribology, ASME*, 123 (2001) 884-887.

[26] G. Dahlquist and A. Björck, *Numerical methods*, Printice Hall, New Jersey, USA, 2001.

[27] E. Darve, *Méthodes multipôles rapides: résolution des équations de Maxwell par formulations intégrales*, l'Univerité Paris 6, thèse de Doctorat, Paris, France, 1999.

[28] E. Darve, The fast multipole method: numerical implementation, *Journal of Computational Physics*, 160 (2000) 195-240.

[29] H.E. de Bree, *The Microflown*, ISBN 90-365-1579-3, 2001.

[30] J.E. Dowling and J.E. Williams, *Sound and sources of sound*, John Wiley and Sons, New York, USA, 1983.

[31] D.G. Duffy, *Green's functions with applications*, Chapman and Hall/CRC, Florida, USA, 2001.

[32] O. Estorff and O. Zaleski, Efficient acoustic calculation by the BEM and frequency interpolates transfer functions, *Engineering Analysis with Boundary Elements*, 27 (2003) 683-694.

[33] G. Evans, *Practical numerical analysis*, John Wiley and Sons, New York, 1995.

[34] M. Fischer, U. Gauger, and L. Gaul, A multipole Galerkin boundary element method for acoustics, *Engineering Analysis with Boundary Elements*, 28 (2004) 155-162.

[35] L. Greengard, A fast algorithm for classical physics, *Science*, 265 (1994) 909-914.

[36] L. Greengard and V. Rokhlin, A fast algorithm for particle simultions, *Journal of Computational Physics*, 73 (1987) 325-348.

[37] M.M. Grigoriev and G.F. Dargush, A fast multi-level boundary element method for the Helmholtz equation, *Computer Methods in Applied Mechanics and Engineering*, 193 (2004) 165-203.

[38] M.M. Grigoriev and G.F. Dargush, A multi-level boundary-element method for two-dimensional steady heat diffusion, *Numerical Heat Transfer*, 46 (2004) 329-356.

[39] I. Hernández-Ramírez and C.H. Venner, Multilevel algorithms for the fast evaluation of integral transform in acoustics, *Third International Conference on Acoustics 2003: Modelling and Experimental Measurements in Acoustics*, Cadiz, Spain, June 16-18, 2003.

[40] D. Kincaid, W. Cheney, *Numerical Analysis: Mathematics of scientific computing*, Brooks/Cole, California, USA, 2002.

[41] L.E. Kinsler, A.R. Frey, A.B. Coppens, and J.V. Sanders, *Fundamentals of Acoustics*, John Wiley and Sons, New York, USA, 2000.

[42] R.H. Lyon and R.G. DeJong, *Theory and applications of statistical energy analysis*, Butterworth-Heinemann, 1995.

[43] S. Merburg and S. Schneider, Performance of iterative solvers for acoustic problems. Part I. Solvers and effect of diagonal preconditioning, *Engineering Analysis with Boundary Elements*, 27 (2003) 727-750.

[44] S. Merburg and S. Schneider, Performance of iterative solvers for acoustic problems. Part II. Acceleration by ILU-type preconditioner, *Engineering Analysis with Boundary Elements*, 27 (2003) 751-757.

[45] P.M. Morse and K.U. Ingard, *Theoretical Acoustics*, McGraw Hill, New York, USA, 1968.

[46] J.C. Nédélec, *Acoustic and electromagnetics equations: integral representation for harmonic problems*, Springer, New York, USA, 2001.

[47] J. Oliver, A doubly-adaptive Clenshaw-Curtis quadrature method, *The Computer Journal*, 15 (1972) 141-147.

[48] E. Perrey-Debain, J. Trevelyan, and P. Bettess, Plane wave interpolation in direct collocation boundary element method for radiation and wave scattering: numerical aspects and application, *Journal of Sound and Vibration*, 261 (2003) 839-858.

[49] P.O. Persson, *Mesh generation for implicit geometries*, PhD Thesis of Massachusetts Institute of Technology, Massachusetts, USA, 2005.

[50] J.R. Phillips, *Rapid solution of potential integral equations in complicated 3-dimensional geometries*, PhD Thesis of Massachusetts Institute of Technology, Massachusetts, USA, 1997.

[51] A.D. Pierce, *Acoustics: An introduction to its physical principles and applications*, Acoustical Society of America, New York, USA, 1991.

[52] I.A. Polonsky and L.M. Keer, A numerical method for solving rough contact problems based on the multi-level multi-summation and conjugate gradient techniques, *Wear*, 231 (1999) 206-219.

[53] I.A. Polonsky and L.M. Keer, Fast methods for solving rough contact problems: a comparative study, *Journal of Tribology*, 122 (2000) 36-41.

[54] I.A. Polonsky and L.M. Keer, Stress analysis of layered elastic solid with cracks using fast fourier transform and conjugate gradient techinques, *Journal of Applied Mechanics*, 68 (2001) 708-714.

[55] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, Numerical recipes in C: The art of scientific computing, *Cambridge University Press*, New York, USA, 1992.

[56] Ch. Provatidis and N. Zafiropoulos, On the 'interior Helmholtz integral equation formulation' in sound radiation problems, *Engineering Analysis with Boundary Elements*, 26 (2002) 29-40.

[57] R. Raangs, W.F. Druyvesteyn and H.E. de Bree, A novel two-dimensional sound particle velocity for source localization and free field measurements in a diffuse field, *Inter-noise 2001*, The Hague, The Netherlands, August 27-30, 2001.

[58] S.W. Reinstra, A. Hirschberg, *An introduction to acoustics*, Report IWDE 01-03, Technical University of Eindhoven, Eindhoven, The Netherlands, 2001.

[59] D.N. Rockmore, The FFT: an algorithm the whole family can use, *Computing in Science and Engineering*, 2 (2000) 60-64.

[60] V. Rokhlin, Rapid Solution of Integral Equations of Classical Potential Theory, *Journal of Computational Physics*, 60 (1985) 187-207.

[61] B. Sandak, Multiscale Fast Summation of Long-Range Charge and Dipolar Interactions, *Journal of Computational Chemistry*, 22 (2001) 717-731.

[62] H.A. Schenck, Improved integral formulation for acoustic radiation problems, *Journal of the Acoustical Society of America*, 44 (1968) 41-58.

[63] A.P. Schubmacher, *Sound source reconstruction using inverse sound field calculations*,Technical University of Denmark, PhD thesis, Denmark, 2000.

[64] S. Schneider, Applications of fast methods for acoustic scattering and radiation problems, *Journal of Computational Acoustics*, 11 (2003) 387-401.

[65] Y. Takahashi, Y. Yonekawa and K. Kanada, A new approach to assess low frequency noise in the working environment, *Indusrtial Health*, 39 (2001) 281-286.

[66] M.A. Tournour and N. Atalla, A novel acceleration method for the variational boundary element approach based on multipole expansion, *International Journal for Numerical Methods in Engineering* , 42 (1998) 1199-1214.

[67] M.A. Tournour and N. Atalla, Efficient evaluation of the acoustic radiation using multipole expansion, *International Journal for Numerical Methods in Engineering* , 46 (1999) 825-837.

[68] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, California, USA, 2001.

[69] H.A. van der Vorst, Kyrlov subspace iteration, *Computing in Science and Engineering*, 2 (2000) 32-37.

[70] C.H. Venner and A.A. Lubrecht, *Multilevel Methods in Lubrication*, Elseiver, Tribology Series, 37, Amsterdam, The Netherlands, 2000.

[71] C.H. Venner, *Personal communication*, 2004.

[72] O. von Estorff and O. Zaleski, Efficient acoustic calculations by the BEM and frequency interpolated transfer functions, *Engineering Analysis with Boundary Elements*, 27 (2003) 683-694.

[73] R. Visser, Acoustic source localization based on pressure and particle velocity measurements, *Inter-noise2003*, Seogwipo, Korea, August 25-28, 2003.

[74] R. Visser, *A boundary element approach to acoustic radiation and source identification*, University of Twente, PhD thesis, Enschede, The Netherlands, 2004.

[75] S. Zhang and J. Jin, *Computation of special functions*, John Wiley and Sons, New York, USA, 1996.

[76] O.Z. Zienkiewicz, *The finite element method*, McGraw Hill, New York, USA, 1979.

[77] O.Z. Zienkiewicz and R.L. Taylor, *The finite element method, Volume 1: The basis*, Butterworth-Heinnemann, 2000.

SPATIAL AND ANGULAR TRANSFERS

The schematical representation of how the intergrid operators work in the MLMIO algortihm are showed in Figures A.1 and A.2. It is well known that the value of an arbitrary function $\phi(x)$ at location x can be approximated by the values known of the function ϕ at the locations x_β around of it, such as:

$$\phi(x_i) = \sum_{\beta=-p}^{p} \omega_\beta(x_i)\phi(x_\beta) \tag{A.1}$$

where p is the order of interpolation. So, the integrid coefficients can be evaluated by:

$$\omega_\beta(x_i) = \begin{cases} 1 & \text{if } i = 2I \text{ and } \beta = 0 \\ \prod\limits_{\substack{m(even)=-p \\ m \neq \beta}}^{p} \frac{(x_i - x_m)}{(x_\beta - x_m)} & \text{if } i = 2I+1 \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

So, the *interpolation* intergrid operation can be schematized as:

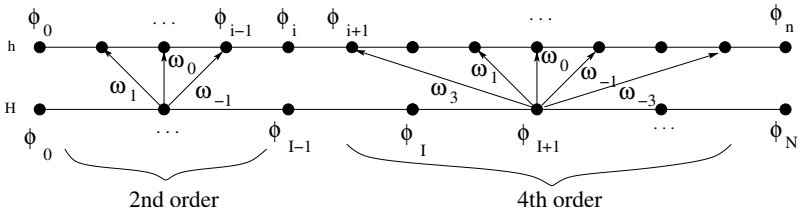


FIGURE A.1: *Schematic representation of the interpolation from a spatial coarse to fine grid, it can be seen as the distribution of the values of ϕ into the spatial domain.*

In another hand, the *anterpolation* is defined as the *adjoint of the interpolation* which in terms of the full weighting the coefficients can be defined by $2^{-d}\omega^T$ [16] [70]. It can be seen as the collection of the values of ϕ from a p-order region. The *anterpolation* intergrid operation is represented as:

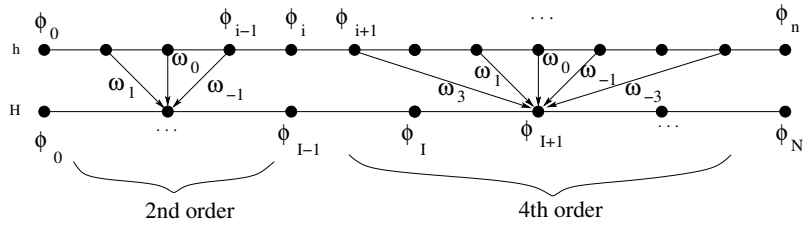


FIGURE A.2: *Schematic representation of the anterpolation from a spatial fine to coarse grid, it can be seen as the collection of the values of ϕ into the spatial domain.*

The angular case of *interpolation* and *anterpolation* follows the similar scheme that for spatial integrid operations, however, the $\mathrm{mod}\ 2\pi$ (or $\mathrm{mod}\ \lambda$, where λ is the number of directions in which is divided the angular domain) is used to determinate the direction that participate in the distribution or collection of values on the angular grid. This scheme is represented by Figures A.3 and A.4.

Angular interpolation:

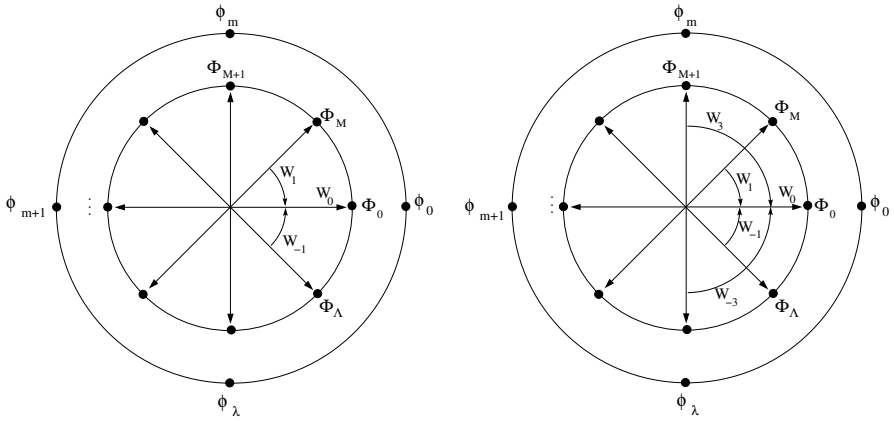


FIGURE A.3: *Schematic representation of the interpolation from a angular coarse to fine grid, it can be seen as the distribution of the values of ϕ into the angular domain.*

Angular anterpolation:

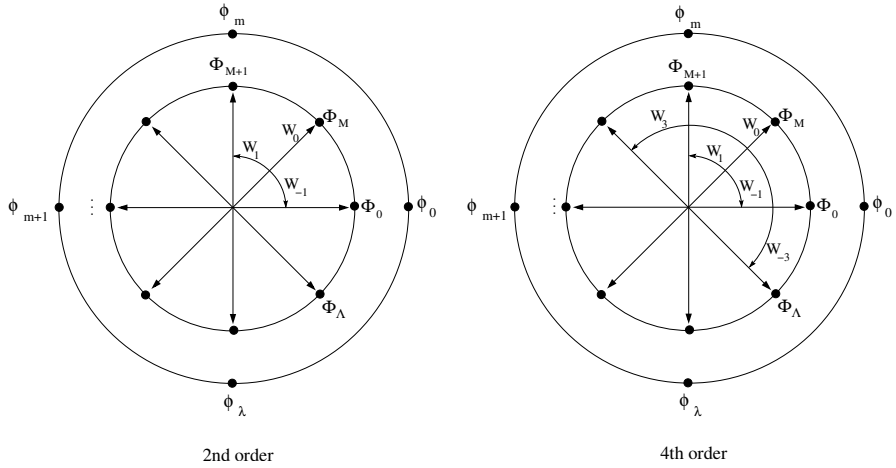


FIGURE A.4: *Schematic representation of the anterpolation from a angular fine to coarse grid, it can be seen as the collection of the values of ϕ into the angular domain.*

RESULTS OF THE MLMIO ALGORITHM

APPENDIX B

In this appendix are shown the way in that the discrete coefficients of the function $G(x, y)$ are determinated for each one of the problems showed in the chpater four. Furthermore, anlaytical solution for the first and second model problems of is provided. Additionally, tables with the norm of the error, computational time and fast evaluation error for the model problems showed in the chaper four are presented here.

B.1 First model problem

B.1-A) Exact analytical solution of (4.1)
The exact analytical solution of (4.1) is defined as:

$$v(x) = v_p(x) + v_n(x) \tag{B.1}$$

with,

$$\begin{aligned} v_p(x) = \quad & e^{-ikx}\{ \quad \left[(1-b^2)c_2 - c_3 + c_4 b\right] e^{ikb}\cos(x-b) \\ & - \quad \left[(1-x^2)c_2 - c_3 + c_4 x\right] e^{ikx} \\ & + \quad \left[(b^2-1)c_1 + c_5 - c_6 b\right] e^{ikb}\sin(x-b)\} \end{aligned} \tag{B.2}$$

$$\begin{aligned} v_n(x) = \quad & e^{ikx}\{ \quad \left[(x^2-1)c_2 + c_3 + c_4 x\right] e^{-ikx} \\ & - \quad \left[(a^2-1)c_2 + c_3 + c_4 a\right] e^{-ika}\cos(x-a) \\ & - \quad \left[(a^2-1)c_1 + c_5 + c_6 a\right] e^{-ika}\sin(x-a)\} \end{aligned} \tag{B.3}$$

where the constants are detfined as:

$$\begin{aligned} c_1 &= \frac{1}{1-k^2} & c_2 &= ikc_1 \\ c_3 &= 2i(-k^3-3k)c_1 & c_4 &= 2(-1-k^2)c_1^2 \\ c_5 &= 2(-1-3k^2)c_1^3 & c_6 &= 4ikc_1^2 \end{aligned} \tag{B.4}$$

B.1-B) Coefficients of the kernel for the discretization of separation of directions
From Equation (4.1), it can be represented by the separation of directions as:

$$v_+(x) = \int_x^b G(x,y)u(y)e^{-iky}dy \tag{B.5}$$

$$v_-(x) = \int_a^x G(x,y)u(y)e^{iky}dy \tag{B.6}$$

$G(x,y)u(y)$ was approximated by a polynomial piecewise $w(y)$ as:

$$\tilde{w}(y) = w_j + (w_{j+1} - w_j)(\frac{y - y_j}{h}) \tag{B.7}$$

Where, w_j is defined in $y = y_j$, and w_{j+1} in $y = y_{j+1}$, so the integrals $v_+(x)$ and $v_-(x)$ were carried out in an exact way between $j+1$ and j, Thus, the two next expressions were obtained to this scheme of discretization,

$$v_+(x_i) = \sum_j G_+(x_i, y_j)u_+(y_j) \tag{B.8}$$

$$v_-(x_i) = \sum_j G_-(x_i, y_j)u_-(y_j) \tag{B.9}$$

where,

$$G_+(x,y) = \begin{cases} (A^+ + B^+)G(x,y) & y > x, \\ A^+\, G(x,y) & y = x, \\ 0 & y < x, \end{cases} \tag{B.10}$$

$$G_-(x,y) = \begin{cases} 0 & y > x, \\ B^-\, G(x,y) & y = x, \\ (A^- + B^-)G(x,y) & y < x, \end{cases} \tag{B.11}$$

and

$$u_+(y_j) = e^{iky_j}u(y_j) \tag{B.12}$$

$$u_-(y_j) = e^{-iky_j}u(y_j) \tag{B.13}$$

and, the constants A^+, B^+, A^- and B^+ are defined as:

$$A^+ = \frac{i}{k} + \frac{1}{hk^2}(1 - e^{iky_j}), \quad B^+ = -\tfrac{i}{k} + \tfrac{1}{hk^2}(1 - e^{-iky_j}), \tag{B.14}$$

$$A^- = -\frac{i}{k} + \frac{1}{hk^2}(1 - e^{-iky_j}), \quad B^- = \tfrac{i}{k} + \tfrac{1}{hk^2}(1 - e^{iky_j}), \tag{B.15}$$

B.1-C) Additional results of the numerical evaluation of (4.1).

$k = 8\pi$					
		1st Scheme of Discretization		2nd Scheme of Discretization	
fl	Mesh	$\|\|\epsilon\|\|_2$	cpu time [sec.]	$\|\|\epsilon\|\|_2$	cpu time [sec.]
5	64	7.19e-05	<1.00e-02	2.01e-05	<1.00e-02
6	128	1.79e-05	<1.00e-02	5.00e-06	<1.00e-02
7	256	4.46e-06	<1.00e-02	1.25e-06	<1.00e-02
8	512	1.11e-06	<1.00e-02	3.11e-07	1.00e-02
9	1024	2.79e-07	1.00e-02	7.78e-08	1.00e-02
10	2048	6.97e-08	4.00e-02	1.95e-08	3.00e-02
11	4096	1.74e-08	1.30e-01	4.86e-09	1.40e-01
12	8192	4.36e-09	5.40e-01	1.22e-09	5.50e-01
13	16384	1.09e-09	2.25e+00	3.04e-10	3.10e+00
14	32768	2.72e-10	1.10e+01	7.48e-11	1.60e+01
15	65536	6.81e-11	4.75e+01	2.02e-11	6.37e+01

TABLE B.1: *Average norm of the error and cpu time for both discretizations (1st discretization: Single Matrix Multiplication. 2nd discretization: Separation of Directions)($k = 8\pi$).*

$k = 32\pi$					
		1st Scheme of Discretization		2nd Scheme of Discretization	
No.	Mesh	$\|\|\epsilon\|\|$	cpu time [sec.]	$\|\|\epsilon\|\|$	cpu time [sec.]
7	256	4.49e-06	<1.00e-02	3.12e-07	<1.00e-02
8	512	1.11e-06	<1.00e-02	7.75e-08	<1.00e-02
9	1024	2.77e-07	1.00e-02	1.93e-08	1.00e-02
10	2048	6.93e-08	4.00e-02	4.83e-09	3.00e-02
11	4096	1.73e-08	1.40e-01	1.21e-09	1.40e-01
12	8192	4.33e-09	5.30e-01	3.02e-10	5.50e-01
13	16384	1.08e-09	2.15e+00	7.54e-11	3.19e+00
14	32768	2.71e-10	1.11e+01	1.88e-11	1.61e+01
15	65536	6.77e-11	4.83e+01	4.77e-12	6.44e+01
16	131072	1.69e-11	1.98e+02	9.13e-13	2.56e+02

TABLE B.2: *Average norm of the error and cpu time for both discretizations (1st discretization: Single Matrix Multiplication. 2nd discretization: Separation of Directions)($k = 32\pi$).*

$\|\|\epsilon\|\|_2, k = 8\pi$							
fl	Mesh	cl=fl	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
5	64	2.01e-05	2.01e-05	2.01e-05	2.01e-05*	2.01e-05	
6	128	5.00e-06	5.00e-06	5.00e-06	5.00e-06	5.00e-06	5.00e-06
7	256	1.25e-06	1.25e-06	1.25e-06	1.25e-06	1.25e-06*	1.25e-06
8	512	3.11e-07	3.11e-07	3.11e-07	3.11e-07	3.11e-07	3.11e-07
9	1024	7.78e-08	7.78e-08	7.78e-08	7.78e-08	7.78e-08	7.78e-08*
10	2048	1.95e-08	1.95e-08	1.95e-08	1.95e-08	1.95e-08	1.95e-08
11	4096	4.86e-09	4.86e-09	4.86e-09	4.86e-09	4.86e-09	4.86e-09
12	8192	1.22e-09	1.22e-09	1.22e-09	1.22e-09	1.22e-09	1.22e-09
13	16384	3.04e-10	3.04e-10	3.04e-10	3.04e-10	3.04e-10	3.04e-10
14	32768	7.48e-11	7.48e-11	7.48e-11	7.48e-11	7.48e-11	7.48e-11
15	65536	2.02e-11	2.02e-11	2.02e-11	2.02e-11	2.02e-11	2.02e-11

TABLE B.3: *Average norm of the error in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 8\pi$)*

$\|\|\epsilon\|\|_2$, $k = 8\pi$					
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9
5	64				
6	128				
7	256	1.25e-06			
8	512	3.11e-07	3.12e-07		
9	1024	7.78e-08	7.79e-08	7.81e-08	
10	2048	1.95e-08	1.95e-08	1.95e-08	1.98e-08
11	4096	4.86e-09*	4.86e-09	4.87e-09	4.91e-09
12	8192	1.22e-09	1.22e-09	1.22e-09	1.22e-09
13	16384	3.04e-10	3.04e-10*	3.04e-10	3.04e-10
14	32768	7.48e-11	7.48e-11	7.48e-11	7.48e-11
15	65536	2.02e-11	2.02e-11	2.02e-11*	2.02e-11

TABLE B.4: *Average norm of the error in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 8\pi$) (Continue)*

$\|\|\epsilon\|\|$, $k = 32\pi$							
fl	Mesh	cl=fl	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
7	256	3.12e-07	3.12e-07	3.12e-07	3.12e-07	3.12e-07*	3.12e-07
8	512	7.75e-08	7.75e-08	7.75e-08	7.75e-08	7.75e-08	7.75e-08
9	1024	1.93e-08	1.93e-08	1.93e-08	1.93e-08	1.93e-08	1.93e-08*
10	2048	4.83e-09	4.83e-09	4.83e-09	4.83e-09	4.83e-09	4.83e-09
11	4096	1.21e-09	1.21e-09	1.21e-09	1.21e-09	1.21e-09	1.21e-09
12	8192	3.02e-10	3.02e-10	3.02e-10	3.02e-10	3.02e-10	3.02e-10
13	16384	7.54e-11	7.54e-11	7.54e-11	7.54e-11	7.54e-11	7.54e-11
14	32768	1.88e-11	1.88e-11	1.88e-11	1.88e-11	1.88e-11	1.88e-11
15	65536	4.77e-12	4.77e-12	4.77e-12	4.77e-12	4.77e-12	4.77e-12
16	131072	9.13e-13	9.13e-13	9.13e-13	9.13e-13	9.13e-13	9.13e-13

TABLE B.5: *Average norm of the error in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 32\pi$)*

$\|\|\epsilon\|\|$, $k = 32\pi$						
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9	cl=fl-10
7	256	3.12e-07				
8	512	7.75e-08	7.75e-08			
9	1024	1.93e-08	1.93e-08	1.93e-08		
10	2048	4.83e-09	4.83e-09	4.83e-09	4.83e-09	
11	4096	1.21e-09*	1.21e-09	1.21e-09	1.20e-09	1.21e-09
12	8192	3.02e-10	3.02e-10	3.02e-10	3.01e-10	2.99e-10
13	16384	7.54e-11	7.54e-11*	7.54e-11	7.54e-11	7.50e-11
14	32768	1.88e-11	1.88e-11	1.88e-11	1.88e-11	1.88e-11
15	65536	4.77e-12	4.77e-12	4.77e-12*	4.77e-12	4.76e-12
16	131072	9.13e-13	9.13e-13	9.13e-13	9.13e-13	9.13e-13

TABLE B.6: *Average norm of the error in the numerical evaluation of (4.1) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 32\pi$)(Continue)*

B.2 Second model problem

B.2-A) Exact analytical solution of (4.13).
The analytical solution for (4.13) is represented by:

$$v(x) = \begin{cases} v_p(x), & \text{if } x = a \\ v_n(x), & \text{if } x = b \\ v_p(x) + v_n(x), & \text{otherwise.} \end{cases} \tag{B.16}$$

with,

$$\begin{aligned} v_p(x) = \quad & \tfrac{1}{k^3} \quad \{\pi[-1 - \frac{1}{2}k^2(1-x^2)] + 2kx(c_1 - 1) + ke^{ik(b-x)}[b + x + 2a\ln(b-x)] \\ & -2kxE_1(ik(x-b)) + i[3(1 - e^{ik(b-x)}) + c_1(k^2x^2 + c_2) + \pi kx \\ & +c_4 \ln(b-x)e^{ik(b-x)} + (k^2x^2 + c_2)E_1(ik(x-b))]\} \end{aligned} \tag{B.17}$$

$$\begin{aligned} v_n(x) = \quad & \tfrac{1}{k^3} \quad \{\pi[-1 - \frac{1}{2}k^2(1-x^2)] + 2kx(c_1 - 1) + ke^{ik(x-a)}[a - x + 2a\ln(x-a)] \\ & +2kxE_1(ik(a-x)) + i[3(1 - e^{ik(x-a)}) + c_1(k^2x^2 + c_2) - \pi kx \\ & +c_3 \ln(x-a)e^{ik(x-a)} + (k^2x^2 + c_2)E_1(ik(a-x))]\} \end{aligned} \tag{B.18}$$

where the constants are defined as:

$$\begin{aligned} c_1 = \gamma \ln(k) \quad & c_2 = -k^2 - 2 \\ c_3 = k^2a^2 + c_2 \quad & c_4 = k^2b^2 + c_2 \end{aligned} \tag{B.19}$$

Here $\gamma = 0.577215664901532...$ which is the Euler constant and $E_1(z)$ represents the exponential integral defined as[1]:

$$E_1(z) = \int_1^\infty \frac{e^{-zt}}{t}dt = \int_z^\infty \frac{e^{-t}}{t}dt \quad (|\arg z| < \pi) \tag{B.20}$$

Numerical evaluation of the exponential integral is suggested to be performed [75] using series expansions for ($|z| \leq 1$):

$$E_1(z) = -\gamma - \ln(z) - \sum_{n=1}^{\infty}(-1)^n \frac{z^n}{nn!} \quad (|\arg z| < \pi)$$

and continued fractions for the arguments ($|z| > 1$) :

$$E_n(z) = e^{-z}\left(\frac{1}{z+}\frac{n}{1+}\frac{1}{z+}\frac{n+1}{1+}\frac{2}{z+}...\right) \quad (|\arg z| < \pi)$$

with $n = 1$.

B.2-B) Coefficients of the kernel for the discretization of separation of directions
The coefficients of the function $G(x, y)$ for the discretization of separation of directions are determined by:

Positive direction
$x_i > y_j$:

$$\begin{aligned} G_p^{hh}(x_i, y_j) &= \int_{y_j-\frac{h}{2}}^{y_j+\frac{h}{2}} \ln(x_i - y)dy \\ &= (x_i - y_j + \frac{h}{2}) \ln(x_i - y_j + \frac{h}{2}) \\ &\quad -(x_i - y_j - \frac{h}{2}) \ln(x_i - y_j - \frac{h}{2}) - h \end{aligned} \tag{B.21}$$

$x_i = y_j$

$$\begin{aligned} G_p^{hh}(x_i, y_j) &= \lim_{y_j \to x_i} \int_{y_j-\frac{h}{2}}^{y_j} \ln(x_i - y)dy \\ &= \frac{h}{2}(\ln(\frac{h}{2}) - 1) \end{aligned} \tag{B.22}$$

Negative direction
$x_i < y_j$

$$\begin{aligned} G_n^{hh}(x_i, y_j) &= \int_{y_j-\frac{h}{2}}^{y_j+\frac{h}{2}} \ln(x_i - y)dy \\ &= (y_j - x_i + \frac{h}{2}) \ln(y_j - x_i + \frac{h}{2}) \\ &\quad -(y_j - x_i - \frac{h}{2}) \ln(y_j - x_i - \frac{h}{2}) - h \end{aligned} \tag{B.23}$$

$x_i = y_j$

$$\begin{aligned} G_n^{hh}(x_i, y_j) &= \lim_{y_j \to x_i} \int_{y_j}^{y_j+\frac{h}{2}} \ln(x_i - y)dy \\ &= \frac{h}{2}(\ln(\frac{h}{2}) - 1) \end{aligned} \tag{B.24}$$

B.2-C) Additional results of the numerical evaluation of (4.13).

$\|\|\epsilon\|\|, k = 8\pi$							
fl	Mesh	cl=fl	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
5	64	1.31e-02	1.31e-02	1.31e-02	1.31e-02*	1.31e-02	
6	128	3.52e-03	3.52e-03	3.52e-03	3.53e-03	3.53e-03	3.53e-03
7	256	9.41e-04	9.43e-04	9.40e-04	9.46e-04	9.51e-04*	9.51e-04
8	512	2.50e-04	2.50e-04	2.51e-04	2.50e-04	2.51e-04	2.50e-04
9	1024	6.63e-05	6.64e-05	6.66e-05	6.69e-05	6.61e-05	6.70e-05*
10	2048	1.75e-05	1.75e-05	1.76e-05	1.77e-05	1.79e-05	1.74e-05
11	4096	4.60e-06	4.61e-06	4.61e-06	4.63e-06	4.68e-06	4.71e-06
12	8192	1.21e-06	1.21e-06	1.21e-06	1.21e-06	1.23e-06	1.26e-06
13	16384	3.17e-07	3.17e-07	3.17e-07	3.18e-07	3.23e-07	3.36e-07
14	32768	8.27e-08	8.27e-08	8.28e-08	8.30e-08	8.36e-08	8.55e-08
15	65536	2.16e-08	2.16e-08	2.16e-08	2.17e-08	2.18e-08	2.24e-08

TABLE B.7: *Average norm of the error in the numerical evaluation of (4.13 using the MLMIO algorithm with $p = 8$ order transfers. ($k = 8\pi$)*

$\|\|\epsilon\|\|, k = 8\pi$					
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9
5	64				
6	128				
7	256				
8	512	2.50e-04			
9	1024	6.63e-05			
10	2048	1.77e-05	1.71e-05		
11	4096	4.58e-06*	4.42e-06		
12	8192	1.28e-06	1.21e-06	1.06e-06	
13	16384	3.70e-07	3.89e-07*	3.18e-07	
14	32768	9.15e-08	1.05e-07	1.06e-07	1.04e-07
15	65536	2.41e-08	2.91e-08	3.91e-08*	3.82e-08

TABLE B.8: *Average norm of the error in the numerical evaluation of (4.13 using the MLMIO algorithm with $p = 8$ order transfers. ($k = 8\pi$) (Continue)*

$\|\|\epsilon\|\|, k = 32\pi$							
fl	Mesh	cl=fl	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
7	256	4.25e-03	4.25e-03	4.25e-03	4.25e-03	4.25e-03*	4.25e-03
8	512	1.12e-03	1.12e-03	1.12e-03	1.12e-03	1.12e-03	1.12e-03
9	1024	2.94e-04	2.94e-04	2.94e-04	2.94e-04	2.94e-04	2.94e-04*
10	2048	7.72e-05	7.72e-05	7.73e-05	7.71e-05	7.72e-05	7.70e-05
11	4096	2.02e-05	2.02e-05	2.02e-05	2.02e-05	2.02e-05	2.02e-05
12	8192	5.28e-06	5.29e-06	5.29e-06	5.30e-06	5.30e-06	5.28e-06
13	16384	1.38e-06	1.38e-06	1.38e-06	1.38e-06	1.39e-06	1.40e-06
14	32768	3.59e-07	3.59e-07	3.59e-07	3.60e-07	3.61e-07	3.65e-07
15	65536	9.34e-08	9.34e-08	9.34e-08	9.35e-08	9.38e-08	9.49e-08
16	131072	2.42e-08	2.42e-08	2.43e-08	2.43e-08	2.44e-08	2.47e-08

TABLE B.9: *Average norm of the error in the numerical evaluation of (4.13 using the MLMIO algorithm with $p = 8$ order transfers. ($k = 32\pi$)*

$\|\|\epsilon\|\|, k = 32\pi$						
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9	cl=fl-10
7	256					
8	512	1.12e-03				
9	1024	2.94e-04				
10	2048	7.70e-05	7.70e-05			
11	4096	2.02e-05*	2.02e-05			
12	8192	5.24e-06	5.28e-06	5.28e-06		
13	16384	1.38e-06	1.34e-06*	1.38e-06		
14	32768	3.64e-07	3.63e-07	3.53e-07	3.41e-07	
15	65536	9.76e-08	9.69e-08	9.77e-08*	9.61e-08	
16	131072	2.59e-08	2.85e-08	2.79e-08	2.90e-08	3.07e-08

TABLE B.10: *Average norm of the error in the numerical evaluation of (4.13 using the MLMIO algorithm with $p = 8$ order transfers. ($k = 32\pi$) (Continue)*

$\|\|FE\epsilon\|\|, k = 8\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
5	64	1.31e-02	1.38e-05	2.85e-05	3.14e-05*	3.14e-05	
6	128	3.52e-03	8.29e-06	4.70e-06	1.90e-05	2.18e-05	2.18e-05
7	256	9.41e-04	2.03e-06	3.66e-06	5.04e-06	1.07e-05*	1.13e-05
8	512	2.50e-04	2.34e-07	5.85e-07	4.71e-07	1.38e-06	2.03e-06
9	1024	6.63e-05	1.18e-07	3.34e-07	6.22e-07	3.60e-07	1.34e-06*
10	2048	1.75e-05	3.77e-08	1.10e-07	2.40e-07	3.89e-07	1.46e-07
11	4096	4.60e-06	6.31e-09	1.87e-08	4.25e-08	8.32e-08	1.16e-07
12	8192	1.21e-06	2.24e-09	6.66e-09	1.54e-08	3.21e-08	5.99e-08
13	16384	3.17e-07	1.12e-09	3.33e-09	7.73e-09	1.64e-08	3.31e-08
14	32768	8.27e-08	2.32e-10	6.90e-10	1.60e-09	3.42e-09	7.01e-09
15	65536	2.16e-08	8.76e-11	2.61e-10	6.07e-10	1.30e-09	2.67e-09

TABLE B.11: *Norm of the fast evaluation error of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 8\pi$)*

$\|\|FE\epsilon\|\|, k = 8\pi$					
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9
5	64				
6	128				
7	256				
8	512	1.86e-06			
9	1024	1.91e-06			
10	2048	8.05e-07	7.75e-07		
11	4096	4.98e-08*	2.10e-07		
12	8192	7.76e-08	4.81e-08	1.52e-07	
13	16384	6.06e-08	7.48e-08*	4.91e-08	
14	32768	1.38e-08	2.43e-08	2.27e-08	3.07e-08
15	65536	5.39e-09	1.05e-08	1.82e-08*	1.66e-08

TABLE B.12: *Norm of the fast evaluation error of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 8\pi$) (Continue)*

$\|\|FE\epsilon\|\|, k = 32\pi$							
fl	Mesh	Disc. Err.	cl=fl-1	cl=fl-2	cl=fl-3	cl=fl-4	cl=fl-5
7	256	4.25e-03	1.69e-06	3.22e-06	3.18e-06	3.18e-06	3.18e-06
8	512	1.12e-03	2.06e-07	2.72e-07	5.52e-07	4.47e-07	4.47e-07
9	1024	2.94e-04	1.13e-07	1.69e-07	3.64e-07	5.98e-07	4.89e-07*
10	2048	7.72e-05	3.72e-08	9.19e-08	6.43e-08	2.38e-07	2.41e-07
11	4096	2.02e-05	6.28e-09	1.76e-08	2.96e-08	1.09e-08	6.25e-08
12	8192	5.28e-06	2.24e-09	6.55e-09	1.40e-08	1.99e-08	1.11e-08
13	16384	1.38e-06	1.12e-09	3.32e-09	7.54e-09	1.47e-08	1.97e-08
14	32768	3.59e-07	2.32e-10	6.89e-10	1.59e-09	3.30e-09	6.01e-09
15	65536	9.34e-08	8.76e-11	2.61e-10	6.06e-10	1.29e-09	2.57e-09
16	131072	2.42e-08	4.38e-11	1.31e-10	3.03e-10	6.48e-10	1.33e-09

TABLE B.13: *Norm of the fast evaluation error of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 32\pi$)*

$\|\|FE\epsilon\|\|, k = 32\pi$						
fl	Mesh	cl=fl-6	cl=fl-7	cl=fl-8	cl=fl-9	cl=fl-10
7	256					
8	512	4.47e-07				
9	1024	4.89e-07				
10	2048	2.73e-07	2.73e-07			
11	4096	9.10e-09*	2.36e-08			
12	8192	4.37e-08	1.69e-08	2.48e-08		
13	16384	1.21e-08	4.34e-08*	1.57e-08		
14	32768	6.80e-09	7.81e-09	2.03e-08	2.07e-08	
15	65536	4.53e-09	4.75e-09	6.63e-09*	1.69e-08	
16	131072	2.60e-09	4.55e-09	3.76e-09	6.66e-09	1.70e-08

TABLE B.14: *Norm of the fast evaluation error of (4.13) using the MLMIO algorithm with $p = 8$ order transfers. ($k = 32\pi$) (Continue)*

B.3 Third model problem

Here are shown the exact analytical solution for the evaluation of the discrete coefficients $G(x_i, y_j)$ for the two dimensional model problem (4.27.

$$G^{hh}(x_i, y_j) \equiv G^{hh}(x_i^1, x_i^2, y_j^1, y_j^2) = \int_{y_j^1 - \frac{h}{2}}^{y_j^1 + \frac{h}{2}} \int_{y_j^2 - \frac{h}{2}}^{y_j^2 + \frac{h}{2}} \frac{dy^1 dy^2}{[(x_i^1 - y^1)^2 + (x_i^2 - y^2)^2]^{\frac{1}{2}}} \quad \text{(B.25)}$$

where the algabraic representation is given by:

$$\begin{aligned} G^{hh}(x_i^1, x_i^2, y_j^1, y_j^2) &= |x_p| sinh^{-1}\left(\frac{y_p}{x_p}\right) + |y_p| sinh^{-1}\left(\frac{x_p}{y_p}\right) \\ &- |x_m| sinh^{-1}\left(\frac{y_p}{x_m}\right) - |y_p| sinh^{-1}\left(\frac{x_m}{y_p}\right) \\ &- |x_p| sinh^{-1}\left(\frac{y_m}{x_p}\right) - |y_m| sinh^{-1}\left(\frac{x_p}{y_m}\right) \\ &+ |x_m| sinh^{-1}\left(\frac{y_m}{x_m}\right) + |y_m| sinh^{-1}\left(\frac{x_m}{y_m}\right) \end{aligned} \quad \text{(B.26)}$$

with

$$\begin{aligned} x_p &= x_i^1 - y_j^1 + h/2 \qquad & x_m &= x_i^1 - y_j^1 - h/2 \\ y_p &= x_i^2 - y_j^2 + h/2 \qquad & y_m &= x_i^2 - y_j^2 - h/2 \end{aligned} \quad \text{(B.27)}$$