Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Science and Engineering

Master's Thesis

Parallel Algorithms for microstrip Antennas Modeling by the Method of Moments

Dinara Giniyatova

Supervisor: Ing. Miroslav Bures, PhD

Study Program: Open Informatics

Field of Study: Software Engineering

May 24, 2018

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Giniyatova** Jméno: **Dinara** Osobní číslo: **474702**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Parallel algorithms for microstrip antennas modeling by the method of moments

Název diplomové práce anglicky:

Parallel algorithms for microstrip antennas modeling by the method of moments

Pokyny pro vypracování:

In this thesis, the problem of modeling and calculating the characteristics of microstrip antennas by the method of moments is considered, while the so-called RWG functions (as basic and testing functions in the method of moments) are widely used. In addition, we consider the possibility of parallelizing the method of moments using CUDA and OPEN MP technologies. Comparative analysis of the performance for CPU and GPU has been carried out.

Seznam doporučené literatury:

1. S.M. Rao, D.R. Wilton, A.W. Glisson : Electromagnetic Scattering by Surfaces of Arbitrary Shape. - IEEE Transactions on antennas and propagation. - V. AP-30. - No.3. - 1982.
2. D.H. Schaubert, D.R. Wilton, A.W. Glisson : A Tetrahedral Modeling Method for electromagnetic Scattering by Arbitrarily Shaped Inhomogeneous Dielectric Bodies. - IEEE Transactions on antennas and propagation. ? V. P-32. - No.1. -1984.
3. Walton C. Gibson : The method of moments in Electromagnetics. - hapman & Hall/CRC, Taylor & Francis Group. - N.-Y.- 2008.
4. Constantine A. Balanis : Antenna Theory. Analysis and design. - John Wiley & Sons. New Jersey - 2005.
5. T.K. Sarkar, A.R. Djordjevic, B.M. Kolundzija : Method of Moments applied to Antennas. - 2000.
6. W.L. Stutzman, G.A. Thiele : Antenna Theory and Design - John Wiley & Sons. USA. - 2013.
7. Jian Guan : OPEN MP, CUDA implementation of the moment method and multilevel fast multipole algorithm on multi-GPU computing systems. - Thesis. Urbana, Illinois. ? 2013.
8. I.P. Molostov, V.V. Sherbinin :Application of NVIDIA CUDA Thechnology for numerical Simulation of Electromagnetic Pulses Propagation. - IEEE Xplore Digital Library.2015.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Miroslav Bureš, Ph.D., laboratoř inteligentního testování softwaru FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **16.04.2018** Termín odevzdání diplomové práce: **25.05.2018**

Platnost zadání diplomové práce: **30.09.2019**

Ing. Miroslav Bureš, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studentky

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Ing. Miroslav Bures (CVUT) and Ing. Dmitrii Tumakov (KFU) for guidance and immense to help on my thesis. I want to especially thank my family, friends and colleagues who prop up me morale throughout my thesis work. Also I want to appreciate Turilova Ekaterina, Enikeev Arslan all Czech team for the opportunity to participate in the Double Degree program (CVUT, KFU).

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

Prague, May, 2018 . . . .

Abstract

Microstrip antennas (MA) are widely used as component in various devices for mobile communication systems: phased antenna arrays, base stations with sectorial radiation patterns, space-diversity reception, etc. Powerful computational algorithms are required to calculate the characteristics of (MA). In thesis we consider the problem of plane electromagnetic wave diffraction by a thin metal plate, which is a special case of the radiating element of a microstrip antenna. Parallel implementation of the numerical solving via CUDA technology is proposed.

Table of Content

Introduction

Microstrip antennas (see Fig.1) are widely used as component in various devices for mobile communication systems: phased antenna arrays, base stations with sectorial radiation patterns, space-diversity reception, etc. Microstrip antennas have several advantages in terms of weight, overall dimensions, cost, simplicity of manufacture, but represent electrodynamic systems, which are difficult to study theoretically. The most complete information on their characteristics can be obtained by using rigorous numerical-analytical methods of applied electrodynamics [3-5]. The calculation for characteristics of microstrip antennas of arbitrary configuration based on strict methods is very often complicated and cumbersome, and at times even impossible because of the difficulties of a fundamental nature. The way out of this situation is the use of powerful computational algorithms, such as the method of moments and technologies of distributed (parallel) computations, such as, for example, CUDA, OpenMP etc.

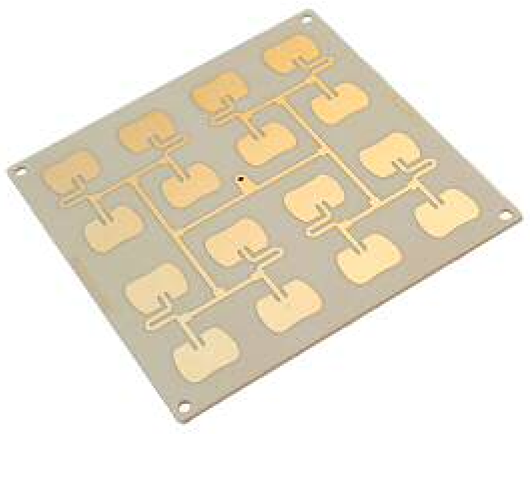


Fig. 1: Microwave antenna 5,8 GHz NP-A44-5800

Assignment

The aim of this work is to solve the problem of a plane electromagnetic wave diffraction by a thin metal plate, which represents a special case of the radiating element of a microstrip antenna and to propose parallel implementation of the numerical solving.

To achieve this goal, it was necessary to solve the following tasks:

1. To study the existing methods and algorithms for solving electrodynamics problems in the field of antenna's design.

2. To Apply the method of moments to solve the diffraction problem formulated in the EFIE form for a rectangular metal plate.

3. Use as a basis and testing functions piecewise linear RWG functions.

4. Program implementation of the developed algorithm (make a triangulation of domain, calculate the potentials and construct a matrix of moments, that is, prepare system of algebraic equations)

5. Within the program, to choose a method for solving a system of linear algebraic equations and to implement its parallel version via parallel programming technologies (CUDA)

The structure of the work corresponds to its goals and objectives: the work consists of an introduction, three chapters, conclusion, a list of used sources, bibliography and applications.

To solve the problems of scattering and radiation of an electromagnetic field, analysis of existing numerical methods and solutions based on them, as well as software, is necessary.

Thesis Outlines

The first chapter is an introduction to the theory of microstrip antennas (MA). It consist of an overview of MA. We examine the canonical forms of MA and the mathematical apparatus of antenna theory. We overview the main numerical methods for analyzing the characteristics of patch antennas, and also we provide the theoretical basis for the method of moments.

In the second chapter, we provide an overview of parallel programming technologies. Based on the comparative analysis, for the further solution of System of Linear Algebraic Equations (SLAE) in the method of moments, the CUDA technology was chosen.

Among the considered numerical methods for solving the problem of scattering EM waves by surfaces of arbitrary shape, the method of moments is chosen. For the discretization of models, it is proposed to use the method of dividing the surface into triangular elements, since it makes it possible to obtain an exact correspondence of any surface or boundary.

In the third chapter, a study of the algorithm for a solving by the method of moments based on the paper of S.M. Rao, D.R. Wilton and A.W.Glisson is made. Based on the C++ programming language, we implemented a computer program which calculate the elements of the moment matrix. The biconjugate gradient stabilized method is chosen for the numerical solution of SLAE, based on the moment matrix. The parallel version of the method is implemented via CUDA Toolkit. In conclusion, the results of the work are formulated.

Chapter 1

Background from Antennas Design

1.1 Overview of microstrip antennas

One of the main trends in the development of modern radio electronics is the microminiaturization of radio electronic equipment (REE). Significant progress in this direction was obtained with the widest use of microelectronics achievements both in the part of low-frequency REE units and its microwave modules. It is known that the qualitative characteristics of REE are largely determined by the properties and design-electrical parameters of its antenna-feeder device (AFD). Particularly noticeable gain in the mass-size parameters of the REA is achieved when moving in microwave modules from the planar integrated circuits (IC) of microwave to volume integrated circuits (VIC). The use of integrated technology makes it possible to successfully solve the problems of creating an AFD under very rigid and contradictory requirements for electrodynamic, aerodynamic, overall, weight, cost, constructive and other parameters.

The concept of a microstrip antenna (MSA) was first introduced in 1950. However, only 20 years later, in 1970 the development of the printed circuit board technology (PCB - Printed Circuit Board) began to be realized. And with the emergence of modern photolithographic technology and low-cost solid-state microwave sources, microstrip antennas have become the most widely used. Since then, microstrip antennas are the most common types of antennas with a wide range of applications due to their obvious advantages:

- light weight and small size;
- simple manufacturing of printed circuit board technology;
- simplified integration with electronic components (placement of microwave devices with conventional components on one board);
- the possibility of making an antenna on a curved plane, which allows you to "fit" the antenna into the carrier unit;
- simplified creation of antenna arrays;
- high repeatability of characteristics;
- ability to radiate energy with linear, circular and elliptical polarization, which leads to convenient design solutions and provides operation in two- or multi-frequency modes.

These antennas are widely used for civil and military systems, such as radio frequency identification (RFID), broadcast radio, mobile systems, global positioning systems (GPS), TVs, multi-input and output (MIMO) systems, vehicles, systems collision avoidance, satellite communications, surveillance systems, radar systems, remote sensing, missile guidance, and so on. Despite their versatility, such antennas have a number of limitations. The disadvantage of such antennas is that an approximate equality of their overall dimensions and working wavelength, which will be determined resonance mode of operation. At high frequencies, for example, at 915 MHz, the dimensions of the microstrip antennas can reach 35 cm.

Fig. 1.1 shows the structure of such antennas. Typically, they consist of three layers: (1) on top – a conductive plate (patch), (2) in the middle - a dielectric layer (substrate), (3) at the base - a substrate (ground plane). Dielectric in MSA is needed to reduce the overall dimensions of the radiator itself. Such microstrip antennas with a good dielectric substrate allow the creation of miniature emitters.The width of the frequency band in the MSA is

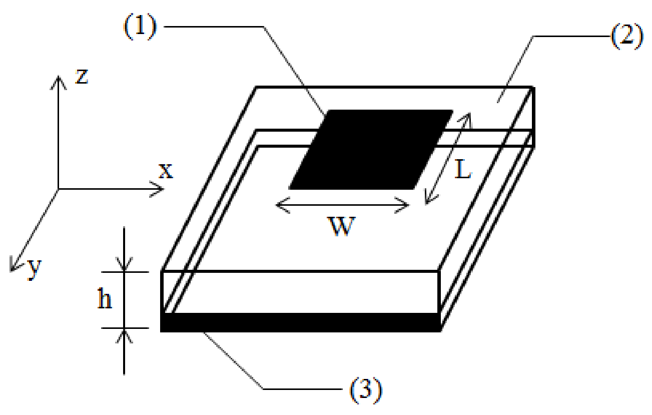


Fig. 1.1: Microstrip antenna model

approximately equal to one to two percent. Dielectric can affect the width of the strip, and how to expand, and narrow down. For example, if to increase the thickness of the dielectric substrate, the bandwidth of the antenna will increase. If to take a dielectric with a greater dielectric constant, then the bandwidth, on the contrary, narrowed.

It must also be remembered that the dielectric parameters affect other characteristics microstrip antennas. A dielectric must have the following properties to meet MPA's high technical requirements and ensure functionality: high temperature stability, minimal energy losses, homogeneity of the structure and unchangeability overall dimensions. Such qualities are possessed by dielectric materials, which are mainly are used in the manufacture of MSA: fluoroplastic, polycorb etc. The energy supply of the conductive plate can be provided in three standard ways: aperture, due to short transmission lines and directly by electromagnetic coupling. Each the method of supplying energy has its advantages and disadvantages. However, all methods affect the characteristics of the antenna as well as the different forms of the conductive plate.

The conductive plate of the MSA has linear dimensions of approximately 0.1 to 0.2 operating wavelengths. In form, they can be of different types. Such diversity contributes to the creation of antennas with a different set of certain characteristics, for example, the input impedance, directivity and polarization of the radiation.

As noted above, the use of the highest-quality dielectrics in the MSA contributes to the improvement of its characteristics, as well as to a reduction in the dimensions of the emitters. However, on the other hand, the presence of a dielectric in the antenna structure makes it difficult to find the necessary parameters of the MSA, since it will be necessary to calculate the effect of surface waves. Therefore, accurate calculation methods will be required to determine the characteristics of the radiators. At most of these methods are based on strict theorems and laws that allow further design and production accuracy to know the parameters of the antennas. There are many methods for calculating the characteristics of the radiation of MPA. In the next section, consider some of them.

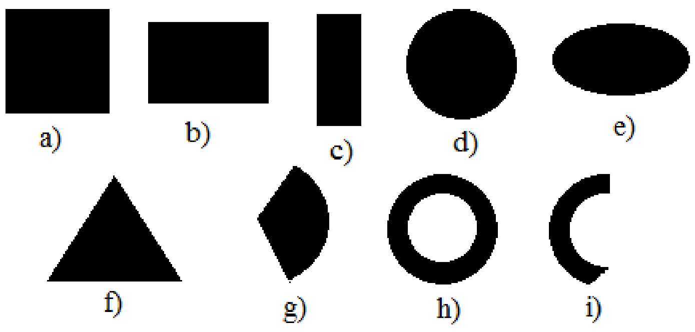


Fig. 1.2: Different shapes of patch

In Fig. 1.2 shows the variants of the geometric shapes of the patch: a – square, b – rectangle, c – dipole, d – circle, e – ellipse, f – triangle, g – disk sector, h – ring, i – sector of the ring.

The choice of the shape is mainly determined by the required radiative characteristics of the antenna. For example, rectangular, square and round radiators of the antenna have very good radiating characteristics. However, when a wide frequency band is required, dipole emitters are used. The metal base in its dimensions is slightly larger than the plate. All considered radiators can be formed practically on any metal surface, planar or non-planar, which leads to the possibility of constructing a large number of diverse antennas that can be used in wireless communication systems. The variety of forms testifies to the great difficulties of theoretical analysis of such electrodynamic structures, the difficulties of its adequate description, compilation convenient mathematical or electrical model.

1.2 Review and comparison of methods for EM modeling

The main problem of electromagnetic modeling is to find an approximate solution of the Maxwell's equations, which corresponds to the given boundary and for nonharmonic waves - to initial conditions. At present, for three-dimensional EM simulation, many methods for calculating, designing and analyzing objects of varying complexity have been developed. In modern software, the following three methods or their combination are more used: the *method of moments* (MoM), the *finite element method* (FEM), and the *finite time difference method* (FDTD). Below we briefly describe the essence and application fields for each of them.

Method of Moments. The method of moments is used to calculate the currents in metallic and dielectric structures during radiation in the free space. It is usually used for the analysis of electrically small structures are made of metals, but the extensions allow the inclusion of dielectrics or layered dielectrics of finite form. In fact, the method of moments is the solution of the Maxwell's equations in integral form in the frequency domain. The main practical advantage of this method is that it is necessary to discretize (i.e. divide into patches) only metal compounds of the modeling structure, since the current distribution on metal surfaces acts as an unknown quantity. This distinguishes MoM from other methods, where free space is discretized, as when we solve equations for finding a electric or magnetic field in a volume. As a result, the "planar" MoM grid proves to be much simpler and smaller than the equivalent "bulk" mesh needed for modeling using the FEM and FDTD methods. The grid turns out to be homogeneous and consists usually of rectangular, triangular or square cells. A reduced number of grid cells improves simulation efficiency. Therefore MoM is well suited for analysis of complex (multilayer) structures. But the MoM method has its drawbacks. It is not suitable for arbitrary three-dimensional structures. The simulated structures must be "planar" in nature and form a multi-layered structure (located in the x-y plane) or be planar objects located in the x-y plane, but elongated along the vertical (along the z-axis) through several superimposed layers . For many RF / microwave devices, this limitation does not play a significant role, since they often have a planar structure.

Finite Element Method. The finite element method is currently one of the main methods for solving variational problems, including the problems of calculating the stress-strain state of structures. The finite element method is used in modeling large or inhomogeneous dielectric objects that can not be effectively modeled by another method. It is FEM that is truly 3D (3D), favorably different from MoM in that it can be used to analyze arbitrary volumetric structures, not limited to multi-layer topologies. It requires placing the simulated object in the "box", which limits the space and defines the modeling area. The entire volume of the modeling domain is discretized using a grid with tetrahedral cells, and a denser grid is created closer to the modeled object. The main unknown quantity in the FEM method is usually the field size. The field is divided into tetrahedrons (elements), the dielectric properties of which vary from element to element. The size of the elements can be changed, reducing it near the area of interest, and increasing - to reduce the cost of CPU time. In each of the elements, the form of the approximating function is arbitrarily chosen. In the simplest case this is a polynomial of the first degree. Outside of its element, the approximating function is zero. The values of the function at the boundaries of the elements are the solution of the problem and are unknown in advance. The coefficients of the approximating functions are usually sought from the condition that the values of the neighboring functions on the boundaries between the elements are equal. Then these coefficients are expressed in terms of the values of the functions at the nodes of the elements. A system of linear algebraic equations is compiled. The number of equations is equal to the number of unknown values at the nodes on which the solution of the original system is sought, and is directly proportional to the number of elements. Since each of the elements is associated with a limited number of neighboring ones, the system of linear algebraic equations has a sparse form, which essentially simplifies its solution. Perhaps, FEM can be recognized as the most flexible method of EM analysis, allowing to model most volume structures. However, for geometrically complex structures a grid with a large number of tetrahedral cells is required. This, in turn, leads to the use of huge matrices, which can require large amounts of computer memory.

Finite Difference Time Domain method. Like the FEM method, the FDTD simulation method is truly three-dimensional and can be used to analyze structures of arbitrary shape. But if the MoM and FEM algorithms solve Maxwell's equations indirectly through matrices, the FDTD algorithms do this explicitly. The entire volume of the modeling domain in the FDTD method is discretized, usually using a grid with hexahedral cells (which are called Yee cells). FDTD uses a step-by-step integration algorithm over time that step-by-step updates the field values in the grid cell, explicitly tracking the progress of electromagnetic waves through the modeled structure. One of the significant advantages of this method over FEM is that it does not require the use of matrices, and as a result, complex problems can often be solved with very little computer memory consumption. In addition, when choosing the FDTD method to accelerate simulation, it is convenient to use the capabilities of modern graphics processors (GPUs). However, along with the advantages of the FDTD method, there are some drawbacks. So, for each one present in the port topology, you need to perform a separate simulation, as a result, a scheme with N ports will require N simulations. This makes the FDTD method not very attractive for analyzing circuits with a large number of ports. A typical application of the FDTD method is the determination of the characteristics of the antenna built into the mobile phone. The antenna may lose its setting when it is embedded in the phone or when the phone approaches the human body. Early detection of such effects can be very useful.

Thus, for each class of EM modeling problems, there are effective methods of solution. Below we will describe in detail the method of moments, which in the future will be applied to the analysis of microstrip antennas.

1.3 The Method of moments

The method of moments (MoM) is most fully described in Harrington's works and in his monograph, published in English. The consistent description and analysis of MoM is also presented, for example, in book of Nikolsky V.V. Note that the author of this book does not use the term MoM, but, nevertheless, it's about this method. From a mathematical point of view, MoM is a method for solving the following operator problem:

$$L(f) = g, \tag{1.1}$$

where $L(f)$ is the linear operator defined in a certain function space, and g is a known function. Under the operator in mathematics understands the action, which assigns a function to a function. The simplest operator example can be differentiation, which the original function puts in correspondence another function - its derivative. Another operators class are integrals. Among them, the best known is the Fourier transformation:

$$L(f) = \int_{\infty}^{\infty} f(x)^{i\omega x} dx$$

It is easy to see that the operator from this formula puts the original function $f(x)$ to a new function, which is called the Fourier transformation and is determined by the right-hand side.

In electrodynamics integral operators are more often used. About how they derive we will talk below. Now we consider the main idea of MoM. In this thesis we omit purely mathematical questions about the region definition of the operator $L(f)$ and convergence of MoM, considering the proof of the MoM known and talk about only its constructive part.

For a compact presentation of the MoM scheme, we need to use some definitions from functional analysis. Among such term refers to the inner product. Suppose we have two functions f and g. Their inner product $\langle f, g\rangle$ must satisfy the following properties:

$$\langle f, g\rangle = \langle g, f\rangle \tag{1.2}$$

$$\langle \alpha f + \beta g, h\rangle = \alpha\langle f, h\rangle + \beta\langle g, h\rangle \tag{1.3}$$

$$\langle \overline{f}, f \rangle > 0, \quad \text{if} \quad f > 0 \tag{1.4}$$

$$\langle \overline{f}, f \rangle = 0, \quad \text{if} \quad f = 0 \tag{1.5}$$

where α, β are constants and $\overline{f}$ is a complex conjugation of f. The definition of an inner product is ambiguous. It can be define by different ways. Let f and g be two functions defined in the open or close domain S, then inner product can be present as follows

$$\langle f, g \rangle = \int\limits_S fg dS. \tag{1.6}$$

It is easy to see that the definition (1.6) satisfies all the conditions (1.2) - (1.5). However, nothing prevents us from adding to (1.6) some known function called the weight function w. Then the inner product will change as follows:

$$\langle f, g \rangle = \int\limits_S wfg dS. \tag{1.7}$$

It still satisfies all the conditions (1.2) - (1.5) and has right to be. Because of the freedom to choose the type of the inner product, we can build more effective algorithms for solving equation (1.1). Now, we will describe the scheme of MoM. First of all, we represent the unknown function f as the expansion of the following type:

$$f = \sum_n \alpha_n f_n, \tag{1.8}$$

where f_n are known functions, called *basis functions*, and α_n – unknown coefficients. Let a function f be defined in some domain S. Equation (1.1) can be written as

$$\sum_n \alpha_n L(f_n) = g. \tag{1.9}$$

The basic functions f_n are chosen in such a way as to correctly model the expected properties of the unknown function f. Now, we introduce one more system of functions g_m, called *testing functions* and define the inner product, or the *moment* between the basis functions f_n and the test functions g_m as

$$\sum_n \alpha_n \langle g_m, L(f_n) \rangle = \langle g_m, g \rangle. \tag{1.10}$$

Thus, the formula (1.10) is a system of linear algebraic equations with respect to the unknown coefficients α_n. If we solve this system of linear equations and find α_n, then we automatically find the desired function f. Strictly speaking, this is the main idea of method of moments, which allows us to reduce the initial operator problem to the system of linear equations, which effectively solved on the computer.

Rewrite (1.10) in matrix form as **Z**a=**b**, where

$$\mathbf{Z} = \begin{pmatrix} \langle g_1, L(f_1)\rangle & \langle g_1, L(f_2)\rangle & ... & \langle g_1, L(f_N)\rangle & ... \\ \langle g_2, L(f_1)\rangle & \langle g_2, L(f_2)\rangle & ... & \langle g_2, L(f_N)\rangle & ... \\ ... & ... & & & \\ \langle g_N, L(f_1)\rangle & \langle g_N, L(f_2)\rangle & ... & \langle g_N, L(f_N)\rangle & ... \\ ... & ... & & & \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} \langle g_1, g\rangle \\ \langle g_2, g\rangle \\ ... \\ \langle g_N, g\rangle \\ ... \end{pmatrix}, \quad a = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_N \\ ... \end{pmatrix}. \tag{1.11}$$

Note, that resulting matrix in MoM is dense in contrast, for example, from the FEM, where the system matrix is symmetric and sparse. Using(1.11) we derive the expression for desired function f:

$$f = \langle \overline{f}, \mathbf{Z}^{-1}\mathbf{b}\rangle, \quad \overline{f} = (f_1, f_2, ...f_N, ..)^T.$$

In formulas (1.8) - (1.11), we deliberately omitted the summation limits. The point is that, strictly speaking, for an accurate description an unknown function f requires an infinite set of basis functions. On practice it is necessary to be limited to the final sum:

$$f_N = \sum_{n=1}^{N} \alpha_n f_n.$$

It is assumed that the following relationship is filled

$$\lim_{N\to\infty} f_N = F, \tag{1.12}$$

where F – exact solution of the equation (1.1). Formula (1.12) implies that the limit exists and is equal to the solution of (1.1). In this case, it is said that the method converges. Since the function F is unknown, relation (1.12) can not be used as a convergence criterion. On practice the following relationship is often used:

$$\lim_{N\to\infty} |f_{N+1} - f_N| = 0,$$

which asserts only that the iterative process converges and the function f change less and less with increasing N, which does not always mean that it converges to the exact solution of equation (1.1). Justification of equality (1.12) refers to the proof part of the MoM, which we shall not consider. To stop the iterative process, that is, to select N, the following relationship can be used:

$$\int\limits_S |f_{N+1} - f_N| dS < \epsilon \tag{1.13}$$

where ϵ is some preassigned number, called the stopping criterion. The formula (1.13) is not the only possible rule for the choice of N. Moreover, other criteria are often used in practice, but their meaning is the same: some value at the N step should become less than some fixed value.

1.3.1 Basis and testing functions

. Let's consider what functions can be used as basis and testing functions. There is a wide variety of basis functions. The fact is that MoM does not impose such strict requirements on the system of basis functions as orthogonality. It suffices to satisfy the requirement of completeness of a system of functions, which means that an infinite series of the form (1.8) converges in the functional space in which the function f is defined. There are two approaches to choose the basis functions. The first approach is focused on the solution of a specific problem. In this case, the basis functions are chosen so as to be maximally similar to the exact solution of equation (1.1). To do this, we have respect to a priori information about the properties of the solution. This reduces the amount of work, but limits the scope of the tasks to be solved. Consider the basis functions used in practice.

Waveguide modes. For example, as the basis functions the eigenfunctions of the solution of the Sturm-Liouville problem for the Helmholtz equation in case of rectangular waveguides are chosen

$$\varphi^x_{m,n}(x,y) = \sin\left(\frac{m\pi}{a}\left(x+\frac{a}{2}\right)\right)\cos\left(\frac{m\pi}{b}\left(y+\frac{b}{2}\right)\right)$$
$$\varphi^y_{m,n}(x,y) = \cos\left(\frac{m\pi}{a}\left(x+\frac{a}{2}\right)\right)\sin\left(\frac{m\pi}{b}\left(y+\frac{b}{2}\right)\right),$$

where m, n are the numbers of the basis functions,a, b – the geometric dimensions of the conductor. For lines tangent to electric walls, the basis functions are modified in such a way as to satisfy the boundary conditions. The possibility of an analytic calculation of the inner products of basis functions is a great advantage of this set. However, only a limited number of devices can be calculated using these basis functions, when the device topology is a set of disjoint rectangular transmission lines.

We can also note a class of finite functions. Such basis functions are functions with a finite support, which are invariant with respect to translational shift. Finite functions can be represented as a product:

$$\varphi(x,y) = \varphi_x(x)\varphi_y(y).$$

An example of such class can be **piecewise sinusoidal basis functions (PWS)**:

$$\varphi_x(x) = \begin{cases} \frac{\sin(k(\Delta x+x))}{\sin(k\Delta x)}, & -\Delta x \le x \le 0 \\ \frac{\sin(k(\Delta x-x))}{\sin(k\Delta x)}, & 0 \le x \le \Delta x \\ 0, & \text{in other cases,} \end{cases} \qquad \varphi_y(y) = \begin{cases} 1, & -\frac{\Delta y}{2} \le y \le \frac{\Delta y}{2} \\ 0, & \text{in other cases.} \end{cases}$$

Such functions are most convenient when calculations are made of configurations with inhomogeneities such as "open end", "capacitive gap" and so on. That is, such devices, where the value of the propagation constant k along the line can be reliably estimated using 2D analysis. Unfortunately, usually the pay for fast convergence of MoM in the above cases is the loss of versatility. Indeed, in each concrete structure there will be suitable basis functions. Searching for them is an informal process that can not be turned into a computer program. Therefore, software developers give preference to basis functions capable of solving a wider set of tasks, even to the detriment of the convergence of the solution. But it does not mean that accounting for a priori information has not found its application in real design systems. However, when switching to more complex structures, it is necessary to use other, more universal system of basis functions. In MoM widely used so called **rooftop functions**:

$$\varphi_x(x) = \begin{cases} 1 - |x|, & -\Delta x \le x \le \Delta x \\ 0, & \text{in other cases,} \end{cases} \qquad \varphi_y(y) = \begin{cases} 1, & -\frac{\Delta y}{2} \le y \le \frac{\Delta y}{2} \\ 0, & \text{in other cases.} \end{cases}$$

Such functions can be used to calculate devices with arbitrary configuration of conductors with rectangular boundaries. To model and analyse the devices with arbitrary boundary it is suitable to use so called RWG functions. Details about these functions will be presented in the next chapter.

For the testing procedure, we can, generally speaking, use any functions. However, their choice for a specific task is crucial to obtaining a "good" solution. One of the most effective methods is the *Galerkin's method*, where the same basis functions are used as testing functions, which are used in the expansion of the required function f. This ensures that the boundary conditions are satisfied in the entire solution domain, and not only at discrete points. Everywhere below, to solve problems, we will use the Galerkin's method.

Chapter 2

Background from Parallel Programming

Parallel computing is one of the most urgent and priority research topics for today. In all top IT companies (Microsoft, Intel, AMD, NVIDIA, Google, etc.) for a long time already there are departments and laboratories engaged in the design and development of high-performance systems based on parallel processing algorithms. With such close attention, parallel computations are due to the rapid growth of data volumes that need processing. The range of problems under study is extremely wide. This includes image analysis and processing, simulation of physical processes, forecasting of various processes, analysis of satellite data, financial calculations, electromagnetic calculations, neural networks and much more. Some of these tasks can be solved by using distributed computations in the cloud (Cloud Computing), the other part can be solved in an acceptable time only by running on a supercomputer with a performance of hundreds of teraflops (TeraFLoating point Operations Per Second, TFLOPS).

Follow an upward trend in the calculations on a big data, special parallel programming technologies are being developed. At the heart of each of them lies its own principle and for effective work the programmer needs the deep knowledge not only in the field of parallel algorithms, but also understanding of the features of the chosen technology. Below we will discuss Open MP and NVIDIA CUDA technologies.

2.1 Comparison of OpenMP and CUDA technologies

As mentioned above, technologies that allow to automate the distribution of tasks between nodes of the computer system use different approaches to solving the problem of the distribution of computations [7,8]. Therefore, it makes sense to compare how effective, universal and easy to use each of them.

OpenMP (Open Multiprocessing) is an application programming interface (API) intended for parallel programming using shared memory. C, C ++ and Fortran programming languages are supported, Solaris, AIX, HP-UX, Linux, Max OS X, and Windows operating systems. OpenMP development is conducted with the participation of large IT companies, such as AMD, Intel, IBM, Cray, and others.

OpenMP is a set of directives for the compiler, library calls and environment variables. So, to use OpenMP we need a compiler that can take into account the OpenMP directives. How to use the parallelism rests with the programmer, who must specify the runtime environment and the compiler what they should do. OpenMP does not detect data conflicts, search for concurrency states, etc. Thus, OpenMP allows you to create portable programs in which the programmer is responsible for the correctness of the parallel algorithm.

OpenMP uses the fork-join model. The API allows writing programs that work correctly, executing in parallel, and sequentially. If a sequential execution of the program is requested, the OpenMP library replaces the real functions with stubs, that is, the measures for the OpenMP operation are still spent. To create a new thread, you must specify the parallel directive. This directive leads to the creation of a team from the thread that processed the directive and any number of new threads (including none), and the thread that processed the directive becomes the main one in the set. Inside the parallel block, there may be several tasks, each job will be assigned to a separate thread. At the end of the parallel block is an implicit barrier that allows you to synchronize threads that are working on tasks. The entire program itself is already in an implicitly defined parallel block.

Thus, OpenMP allows the programmer to specify blocks that can be parallelized and tasks that must be run in parallel. OpenMP can not use the GPGPU tools. The library also includes synchronization primitives other than the barrier. Since the thread model is used for execution, the memory is common to all the generated threads. OpenMP can be considered as a standardized tool for creating portable parallel programs executed on the central processor (or processors), which allows the programmer do not think about the features of the parallel programming tools on each platform.

CUDA (Compute Unified Device Architecture) is a hardware-software platform for parallel computing that uses the resources of the NVidia graphics processor for non-graphical computing [2]. The development of CUDA was started in 2006. Supported programming languages C, C++ and Fortran, Windows 7, Windows XP, Windows Vista, Linux, Mac OS X. All supported nVidia GPUs are divided into several classes depending on hardware and supported operations, backward compatibility is supported. CUDA is available both on home video adapters, and in the form of specialized coprocessors Tesla and Fermi.

CUDA uses the SIMD model, which imposes limitations on algorithms that can be effectively implemented on such a platform. So, for example, when executing a conditional statement, there is a problem with processing an alternative branch - you have to perform operations, the result of which will be discarded. However, CUDA masks this problem, the programmer does not have to solve it himself. Because of the features of the processor, CUDA can only work with a limited set of data - one-dimensional, two-dimensional and three-dimensional blocks with data of the same type. Each block can be processed by several threads using shared memory. CUDA allows you to perform calculations simultaneously on the central and graphics processors due to the asynchrony of the call to the computational core. With GPUDirect technology, it is possible to directly access the memory (DMA) of the GPU. From a programmer's perspective, a program using CUDA consists of several phases, each running on a central or graphics processor. Functions that can be executed on the GPU are marked with special keywords (extension of the language) and are called kernels. The data structures involved are also marked with keywords. When compiling a special preprocessor splits the phases into two parts, these parts are compiled separately, that is, the modification of the compiler of the source programming language is not required. The address spaces of the central and graphic processors are independent of each other, which causes a delay when copying data between them.

Thus, CUDA allows the programmer to accelerate the execution of code sections that fit well on the SIMD execution model, with the help of graphics processor resources. You do not need to modify the development tools already in use, you just need to set up a connection with the preprocessor and the compiler from NVidia. Desadvantages are the need for a graphics processor from NVIDIA (which, incidentally, is corrected using the generalized OpenCL library) and the need to copy data between address spaces, as well as the complexity of the memory model on the graphics core.

In work [10], a comparison of OpenMP and CUDA technologies is presented for calculating matrix multiplication. According to the analysis carried out by the author, it can be established the following:

1. If the platform does not have a graphics processor supporting acceleration of calculations (nVidia CUDA / AMD Fusion / OpenCL), but has a multi-core (or multiple) processor, the use of OpenMP is an easy way to achieve improved performance and reduce power consumption in proportion to the number of processors / cores;

2. If the platform has a graphics processor, the task is SIMD-realizable, but it is a set of consecutive segments of the parallel SIMD code, then the application of the graphics processor gives an increase in performance and a reduction in energy costs by an order of magnitude.

Since in our problem matrix calculations are widely used, and we will develop a program on a machine with a graphics card NVIDIA Geforce 920M, then we chose CUDA technology to use them in our calculations. Therefore, below we will describe in more detail the CUDA program model.

2.2 CUDA program model

The first ideas about the possibility of using their own video adapters in tasks other than simple image rendering were born in the minds of NVIDIA developers in 2004. It was then that NVIDIA Corporation released its first book on graphics programming - GPU Gems. Most of the book consisted of articles devoted to programming game graphics, as well as creating high-quality visual effects. Of particular interest are the last chapters of this book, entitled "Beyond Triangles", which contained the first reflections on what we now call GPGPU (General-Purpose Computing on Graphics Processing Units). But the real interest is the second book, from the GPU Gems series released a year later, in the spring of 2005, in which almost a quarter of the chapters were devoted to the possibility of programming the GPU. But with the tools and environments that were available at that time, this task remained extremely complex and almost inaccessible to ordinary programmers. To solve this problem, NVIDIA assembled a group of developers engaged in the creation of special software and hardware. This direction was named Compute Unified Device Architecture or simply CUDA. The development of CUDA was announced together with the G80 chip at the end of 2006, and in early 2007 the developers were presented with the first beta version of the CUDA SDK. The first official version of CUDA was released in June 2007.

2.2.1 The CUDA memory model

Perhaps, one of the most important features for CUDA developers is free memory access (support for scatter and gather operations) with the possibility of byte addressing. Each of the threads executed on the GPU has access to the following memory types:

1. *Global memory* is the main type of memory, which has the largest volume and is available for all multiprocessors on a video chip. The size of global memory directly depends on the model of the video card and varies from 256 megabytes to 4 gigabytes on Tesla. Has a high bandwidth (more than 100 gigabytes / s on the latest solutions from

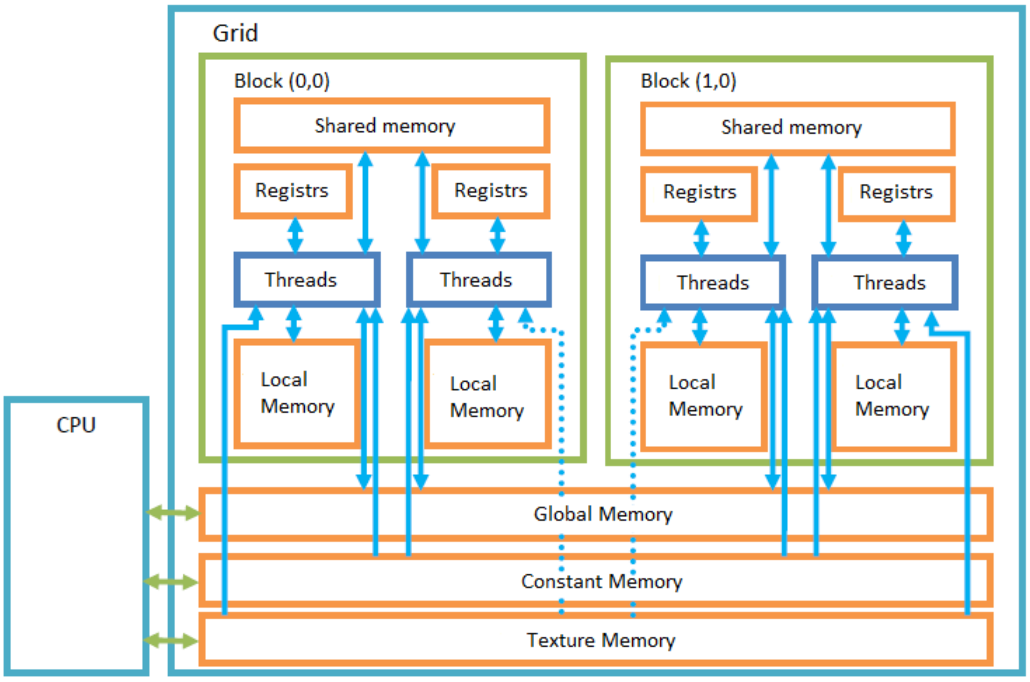


Fig. 2.1: Program model of multiprocessor memory

NVIDIA), but work with this type of memory is associated with significant time delays (several hundred clock cycles). Global memory is not cached, supports linear addressing and conventional pointers.

2. *Local memory* is a type of memory in which all variables declared inside the CUDA program are placed by default. Just like global memory, it is very slow and does not support caching.

3. *Shared memory* is a 16-kilobyte block of memory, available for writing and reading from all stream processors in a multiprocessor. In terms of speed, this type of memory is comparable with registers. The main purpose of shared memory is to ensure interaction between threads. Also shared memory can be used in the role of a programmable cache, which helps to reduce delays when working with global memory.

4. *Constant memory* is a 64 KB memory area used to store the program's constant values. This type of memory is cached (8 kilobytes per multiprocessor). And if there is no necessary data in the cache, reading from the constant memory is performed with delays of several hundred clock cycles.

5. *Texture memory* is a block of memory accessible only for reading by all multiprocessors. Sampling from this type of memory occurs using the texture blocks of the video chip. Due to this, it is possible to perform linear interpolation at no additional costs. This type of memory is cached.

   Fig. 2.1 shows the software model of the video card's memory. In fact, global, local, textural and constant memory are represented on the chip in the form of the local memory of the video card. The CPU can only access global, constant and texture memory.

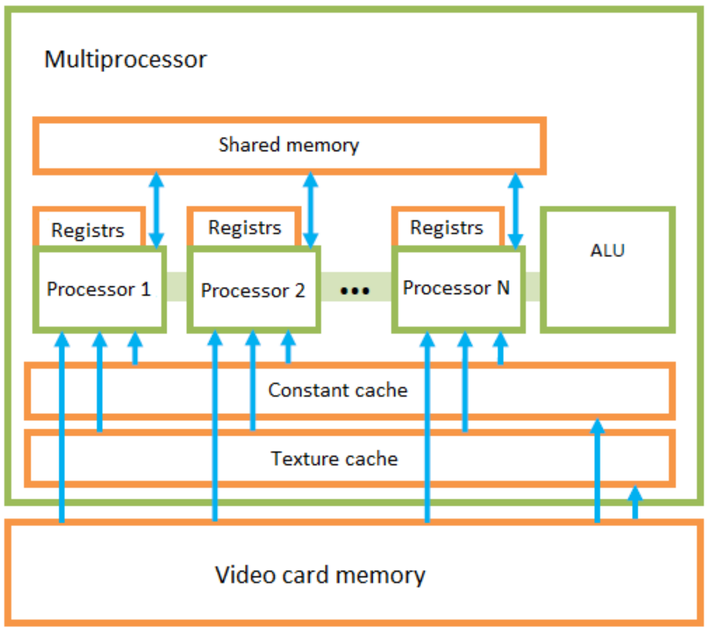


Fig. 2.2: NVIDIA Multiprocessor Model

2.2.2 Multiprocessors

Video chips from NVIDIA consist of several clusters of texture units (Texture Processing Cluster, TPC). Each cluster in turn consists of a block of texture samples and several multiprocessors. The multiprocessor consists of 8 computational devices and two superfunctional units. All instructions in the GPU are performed according to the SIMD principle, i.e. one instruction applies to all threads in the wrap (a group of 32 threads). This way of doing so called SIMT (Single Instruction Multiple Threads) - one instruction and many threads. Each of the multiprocessors is allocated from 8192 to 16384 registers (depending on the model of the video card), which are common to all the threads that are executed on it. Also on each processor there is 16 kilobytes of fast shared memory, which is write access and read from any stream within one block. Multiprocessors also have access to video memory, but it is necessary to be extremely cautious, since access to it is associated with significant time delays. To speed up access and reduce the number of accesses to video memory, all multiprocessors are equipped with a small (8 kilobytes) cache for constants and textures. Multiprocessors GPU can perform up to eight blocks and up to 24 wraps, each of which includes 32 threads. In other words, the multiprocessor has up to 768 threads.

Knowledge of the architecture of the GPU is extremely important in the compilation of the algorithm, since it allows to optimize it for the available resources of the video card.

2.2.3 Programming on CUDA

As mentioned earlier, the graphics transporter used to render the image is a set of multiple processing steps. With traditional APIs, the programmer, regardless of the complexity of the algorithm, is always required to configure all parts of the graphics transporter. This fact significantly complicates the use of the GPU for solving general-purpose problems, since even a simple addition of two matrices requires execution of a number of commands for preparing and drawing images in the off-screen buffer. As a result, several lines of the shader program account for hundreds of lines of additional code. When solving problems with small dimensions, these additional costs can negate the entire gain from using the GPU.

The programming model used in CUDA differs from the traditional API in that it completely hides the graphical transporter from the programmer, allowing it to write programs in more familiar terms for the extended variation of the C language. In addition, CUDA provides the programmer with a more convenient model of working with memory. There is no longer any need to store data in 128-bit textures, since CUDA allows you to read data directly from the memory of the video card.

NVIDIA CUDA includes two APIs: high level (CUDA Runtime API) and low (CUDA Driver API). If you need to use the low-level functions of the graphics processor, the programmer can always abandon the Runtime API in favor of the Driver API. It is worth noting that the use of both APIs in one program is not possible.

The first step when moving an existing algorithm to CUDA is certainly its analysis, the purpose of which is to find a "bottleneck" that needs to be parallelized. As a rule, in the algorithm for CPU such areas are enclosed in a cycle or recursion. Full transfer of the algorithm to the GPU is not possible, because the GPU does not have access to either computer memory or I / O devices (except for the frame buffer, which can be displayed as a picture on the computer screen). While executing the program, the CPU is still responsible for the preparation and postprocessing of the data, while the laborious work itself falls on the GPU. A set of instructions executed on the GPU is called the kernel. The kernel, in fact, is the development of the concept of shaders. The CPU is responsible for the formation and compilation of the kernels. The video chip simply accepts the already compiled kernel and creates its copies for each data item. Each of the kernels is executed in its own thread. Threads in the GPU can be executed only in groups of 32 copies (wrap). In this case, the total number of threads needed to solve a problem can exceed the maximum allowable for the current video card. Therefore, each clockcycle hardware chooses which of the wraps will be executed. But if in the CPU such a switch would take hundreds of clockcycles, the GPU does this almost instantly. The data in the kernel is scalar, unlike shaders, where all data are represented as four-component vectors. Such a representation is more natural for most non-graphic tasks. The programming model of CUDA involves grouping flows into blocks - one-, two- or three-dimensional matrices. Interaction between them is carried out with the help of shared memory. Also there are synchronization points that allow data in all threads to be brought to the current state. Each of the kernels is executed over the grid of the blocks (see Fig. 2.3) . At any one time, only one grid can be executed on the GPU. This grouping allows you to achieve maximum scalability. If the GPU

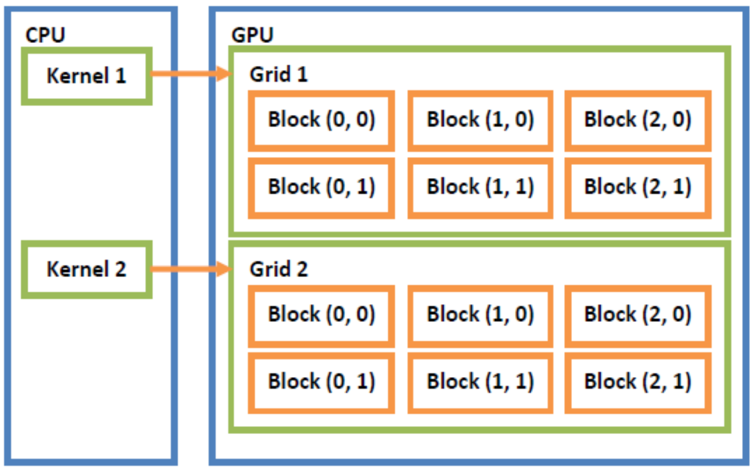


Fig. 2.3: The scheme of grouping threads into blocks and grids

does not have enough resources to run all the blocks - they will be executed sequentially, one by one. This allows the developer not to think about the power of the device on which the application will be launched.

Programming CPU on CUDA

There are a number of common steps in every application written on NVIDIA CUDA, regardless of its purpose: Preparing the memory. Since the GPU does not have access to RAM, the programmer needs to take care in advance that all the resources necessary for executing the application kernel are in the memory of the video card. For these purposes, three main functions are used from the CUDA SDK: cudaMalloc, cudaMemcpy, and cudaFree. These functions have the same purpose as standard malloc, memcpy and free, but, of course, all operations are performed on video memory. It is also worth noting that the function cudaMemcpy has an additional parameter, indicating the direction of copying the information (from the CPU to the GPU or vice versa). The configuration of the grid and blocks. The configuration process is extremely simple and consists in setting the size of the grid and blocks. The main task of the programmer at this step is to find the optimal balance between the size and the number of blocks. By increasing the number of threads in the block, it is possible to reduce the number of calls to the global memory by increasing the intensity of data exchange between the threads via fast shared memory. On the other hand, the number of registers allocated to the block is fixed and if the number of threads is much larger, the GPU will begin to place data in slow local memory, which will significantly increase the execution time of the kernel. NVIDIA recommends that programmers use blocks of 128 or 256 threads. In most tasks, the number of threads in the block allows to achieve optimal delays and the number of registers. Starting the kernel. The kernel is called as a regular function in the C language. The only significant difference is that when you call the kernel, you must transfer the previously defined grid and block dimensions. Retrieving results and freeing memory. After executing the kernel, you must copy the results of the program back to the main memory using the cudaMemcpy function, specifying the reverse copy direction (from the GPU to the CPU). Just like in any C-program, you need to release all allocated resources, to prevent memory leaks.

Programming GPU on CUDA

Writing code for the GPU on CUDA and Shader Model is very different. When using CUDA, there is no need to learn a new language, the developer has the familiar C with a number of additional extensions, with which you can access the hardware capabilities of the GPU and the context of the current executable thread. The functions that make up the kernel are placed in a file with the extension cu, which is compiled using the NVCC program. NVCC, in turn, is a wrapper over other tools, and calls them: cudacc, g ++, cl, etc. As a result of the compilation, all the code is divided into 2 parts: the first part is intended for execution on the CPU, and the second contains the PTX object code for the GPU. To determine the type of device (GPU or CPU) on which the function will be executed, a number of qualifiers have been introduced in CUDA:

1. __*host*__ – functions marked with this qualifier are executed on the CPU. They can be called in the same way, only with the CPU.

2. __*global*__ – runs on the GPU, it is called from the CPU.

3. __*device*__ – runs on the GPU, it is called from the GPU.

CUDA also provides a set of specifiers that define the type of memory for placing variables:

1. __*device*__ – this specifies that the variable is in the global memory of the device. This type of memory is used for data exchange between CPU and GPU, as well as for the interaction of kernel from different blocks.

2. __*constant*__ – specifies a variable in the constant memory. This type of memory is cached.

3. __*shared*__ – specifies a variable in the shared memory of the block.

A function executed on the GPU can have arbitrary arguments. In most tasks, references to memory locations are transferred as parameters to which all the necessary information was previously copied, as well as variables containing the dimensionality of the data. That is, the parameters are the same for all kernel instances running on the GPU. In order to determine the piece of data to be processed, the blockDim and blockIdx variables are used, which contain the block size and the index corresponding to the current kernel instance.

Chapter 3

Solution the problem of EM wave diffraction on metallic plate by the method of moments

In this section we will consider the problem of electromagnetic scattering for metallic plate and its solution by the method of moments. The rectangular metallic plate is the special case of microstrip antenna radiating element.

Following [1,2], we obtain analytic representations for the elements of the moment matrix and construct the resulting system of linear algebraic equations. All about the numerical solution of this problem will be presented below, in the next section.

3.1 Integral equation of electric field (EFIE) ant its solution

It will be recalled that the equation for the surface current induced on a conducting scatterer is obtained from Maxwell's equations and boundary conditions on the electric field (see, for example, [3-6]). Let S – be an open or closed surface of an ideally conducting body with unit normal $\mathbf{n}$. We denote by $\mathbf{E}^i$ the electric field generated by the source in the absence of the body (antenna). It induces a surface current $\mathbf{J}$ on S. If S is an open surface, then we consider $\mathbf{J}$ as the sum of surface currents on opposite sides of S, and hence the normal component $\mathbf{J}$ should disappear on S boundaries. The scattered field $\mathbf{E}^s$ can then be calculated from the formula (see [])

$$\mathbf{E}^s = -i\omega\mathbf{A} - \nabla\Phi, \tag{3.1}$$

where $\mathbf{A}$ and Φ – are vector and scalar potentials, respectively. From electromagnetic theory it is known that the potentials are related to the exciting current through the Green's function. The following formulas are valid in free space:

$$\mathbf{A}(r) = \mu \int\limits_S \mathbf{J}(r')G(r,r')dS', \tag{3.2}$$

$$\Phi(r) = \frac{1}{\varepsilon} \int\limits_S \sigma G(r,r')dS', \tag{3.3}$$

where the Green's function

$$G(r,r') = \frac{e^{-ik|r-r'|}}{4\pi|r-r'|},$$

$k = \omega\sqrt{\mu\varepsilon} = 2\pi/\lambda$ (λ - is the wavelength) and $|r - r'|$ – is the distance between an arbitrarily localized observation point r and the starting point r' on S. The charge density σ is related to the surface current density by the continuity equation

$$\nabla_s \cdot \mathbf{J} = -i\omega\sigma. \tag{3.4}$$

The boundary condition for the electric field in the case of an ideally conducting surface has the form

$$\mathbf{n} \times (\mathbf{E}^i + \mathbf{E}^s) = 0, \tag{3.5}$$

whence, using (3.1) we obtain an integro-differential equation with respect to $\mathbf{J}$

$$-\mathbf{E}^i_{tan} = (-i\omega\mathbf{A} - \nabla\Phi)_{tan} \tag{3.6}$$

Together with (3.2) - (3.4), equation (3.6) is the so-called integral equation for the electric field (EFIE). In the literature, the classical formulation of

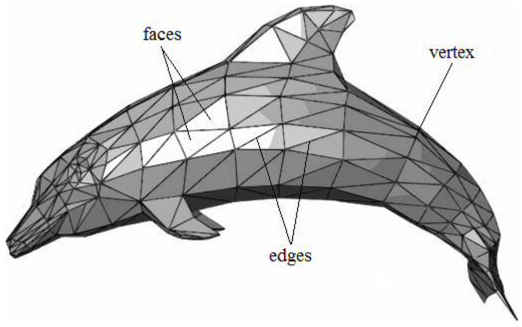


Fig. 3.1: Triangulation of an arbitrary surface

EFIE for ideally conducting surfaces is written as

$$\mathbf{n} \times \int_S G(r,r') \left[\mathbf{J}(r') + \frac{1}{k^2}\nabla(\nabla \cdot \mathbf{J}(r'))\right] dS' = \frac{1}{ik\eta_0}\mathbf{n} \times \mathbf{E}^i(r),$$

and equation (3.6) is called an equation in terms of mixed potentials (Mixed Potential Integral Equation). Nevertheless, everywhere below, we will adhere to the term EFIE, implying equation (3.6), taking into account (3.2) - (3.4). We note that the presence of the derived quantities $\mathbf{J}$ in (3.4) and Φ in (3.5) means that it is necessary to select with special care the basis and test functions when solving the problem by the method of moments.

3.1.1 RWG basis functions

One of the most popular basic functions used in calculating antenna parameters are the so-called RWG functions proposed in [1]. It is convenient to use them to search for an approximate solution of EFIE, when the surface of an ideally conducting body is divided into elementary triangular areas. That is, we assume that the surface S is "well" approximated by a triangular grid. We will use standard terms such as a face to indicate the surface of an elementary triangular area, an edge (boundary edge) - to indicate one of its sides and a vertex - to indicate the vertex of a triangle (see Fig. 3.1) First of all, note that each basic RWG function is associated with one inner edge and vanishes everywhere on S, except for a pair of triangles adjacent to the given edge. Figure 3.2 shows two such triangles, T_n^+ and T_n^-, adjacent to the n-th edge. The points belonging to the triangle T_n^+ can be described both in global coordinates by the radius vector $\mathbf{r}$ and in local coordinates using the

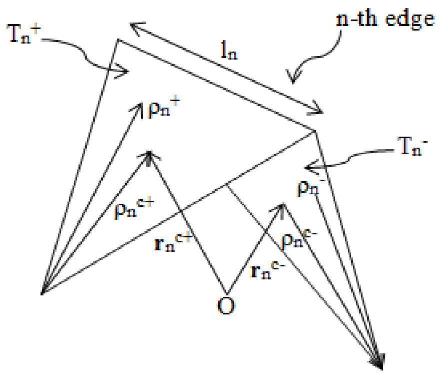


Fig. 3.2: A pair of triangles with a common inner edge n

radius of the vector ρ_n^+ given with respect to the free vertex of the triangle T_n^+. A similar remark is also true for the triangle T_n^- with the only difference that the vector ρ_n^- is directed from the point belonging to the triangle to the free vertex T_n^-. The choice of "positive" and "negative" triangles is arbitrary, taking into account that for the entire cycle of calculating the surface current, it will not change.

The base function associated with the n-th inner edge has the form:

$$f_n(\mathbf{r}) = \begin{cases} \frac{l_n}{2A_n^+}\rho_n^+, & \mathbf{r} \in T_n^+ \\ \frac{l_n}{2A_n^-}\rho_n^-, & \mathbf{r} \in T_n^- \\ 0, & \text{in other cases,} \end{cases} \tag{3.7}$$

where l_n is the length of the n-th edge, A_n^+ and A_n^- – areas of the triangles T_n^+ and T_n^-, respectively. List the main properties of these functions [1], which make it possible to use them as an approximation for the surface current

1. The current density does not have a normal component to the boundary edges of two triangles, except for the common n-th edge, i.e. the current does not flow through the external boundary of the given area and, consequently, there is no current density discontinuity and associated linear charges on the boundary of the sections of two triangular elements.

2. The current component normal to the n-th edge is constant and continuous along this edge and represents the height of the triangle $T_n^{\pm}$

with the edge n as the base. Numerically, it is expressed by the formula $h^{\pm} = 2A_n^{\pm}/l_n$. This coefficient normalizes f_n in (3.7) so that its thread density through the edge n is unit value and continuous when passing from T_n^+ T_n^-. Thus, linear charges on the n-th edge are also absent.

3. The surface divergence f_n, which is proportional to the surface charge density σ, is expressed by the formula

$$\nabla_s \cdot f_n = \begin{cases} \frac{l_n}{A_n^+}, & \mathbf{r} \in T_n^+ \\ -\frac{l_n}{A_n^-}, & \mathbf{r} \in T_n^- \\ 0, & . \end{cases} \tag{3.8}$$

Thus, the charge density is a constant in each of the triangular elements, and the total charge density is zero (on a pair of adjacent triangles).

4. The integral $R_n = \int\limits_{T_n^+ + T_n^-} f_n dS$ is called the moment of the function f_n. By the mean value theorem, we have

$$R_n = (A_n^+ + A_n^-) f_n^{avg} = \frac{l_n}{2}(\rho_n^{c+} + \rho_n^{c-}) = l_n(r_n^{c+} + r_n^{c-}), \tag{3.9}$$

where f_n^{avg} – is the average value of f_n on a pair of triangular elements, $\rho_n^{c\pm}$ – is the radius vector of the center of mass of triangles in local coordinates, and $r_n^{c\pm}$ – is the radius vector of the center mass relative to the global origin.

Following the method of moments, we represent the surface current everywhere on S in the form of an approximate formula

$$\mathbf{J} \approx \sum_{n=1}^{N} \alpha_n f_n(r), \tag{3.10}$$

where N – is the number of inner edges. Since each basis function is associated with one inner edge that is common to two triangular elements, accordingly, up to three-basis functions can have non-zero values inside each triangular face. But for a given edge only the basic function associated with it has a current component normal to this edge. The current of the remaining basis functions in adjacent faces is parallel to this edge. In addition, since the normal component f_n to the n-th edge is unit, the coefficient α_n can be interpreted as the normal component of the current density flowing through the edge n. According to the above properties, all the basis functions inside a given triangle are linearly independent. On the boundary edges of the sur-

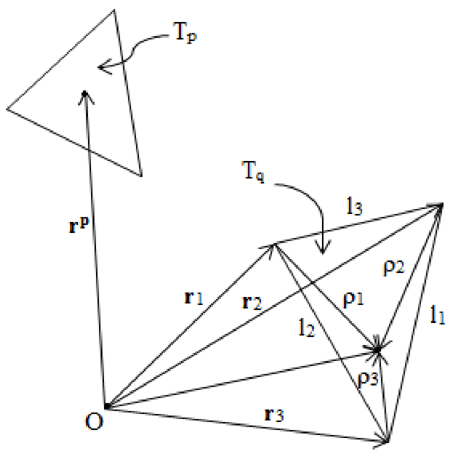


Fig. 3.3: Local coordinates in the triangle T_q with the observation point in the triangle T_p

face S, the sum of the normal components of the current flowing through the opposite sides of S is equal to zero because of the continuity equation. Therefore, it is not necessary to determine and take into account in (3.10) the contributions from the basis functions associated with the boundary edges. Note also that due to a significant change in the directions of the streamlines f_n in the triangle, it is not obvious that such functions can describe, for example, a current flowing in one direction over the entire area of the given triangle. Show this with the aid of Fig. Consider a triangle T_q with sides l_1, l_2 and l_3. The basic functions defined in this triangle have the form $f_i = (l_i/2A_q)p_i, i = 1, 2, 3$. It follows from Fig. 3 and the definition of the functions f_i that the linear combinations $l_2 f_1 - l_1 f_2$ and $l_3 f_1 - l_1 f_3$ are constant vectors for each point r in T_q that are parallel to the edges l_3 and l_2, respectively. Indeed, consider, for example,

$$l_2 f_1 - l_1 f_2 = \frac{l_2 l_1}{2A_q} p_1 - \frac{l_1 l_2}{2A_q} = \frac{l_1 l_2}{2A_q}(p_1 - p_2).$$

Since the two forms are linearly independent (not parallel), any constant vector of arbitrary magnitude and direction can be expressed by a symmetric linear combination of these forms in the triangle T_q.

3.1.2 Testing procedure

The next step in the method of moments is the procedure of testing or multiplying the initial equation for trial (test) functions. As trial functions, we take the expansion functions f_n. Define scalar multiplication as

$$\langle f, g \rangle = \int\limits_S f \cdot g dS$$

and will test the equation (3.6) with the RWG functions. We get

$$\langle \mathbf{E}^i, f_m \rangle = i\omega \langle \mathbf{A}, f_m \rangle + \langle \nabla\Phi, f_m \rangle \tag{3.11}$$

Using the methods of calculating the surface integral and the properties of f_m on the boundaries of S, we rewrite the last term in (11) in the form

$$\langle \nabla\Phi, f_m \rangle = -\int\limits_S \Phi \nabla_s \cdot f_m dS. \tag{3.12}$$

Then, taking into account (3.8), the integral in (3.13) can be approximated as follows

$$\int\limits_S \Phi \nabla_s \cdot f_m dS = l_m \left(\frac{1}{A_m^+} \int\limits_{T_m^+} \Phi dS - \frac{1}{A_m^-} \int\limits_{T_m^-} \Phi dS \right) \cong l_m [\Phi(r_m^{c+}) - \Phi(r_m^{c-})]. \tag{3.13}$$

In (3.13), the average value of Φ for each triangle was replaced by Φ at the center of mass of the triangles. Using similar arguments, we can approximate the terms in (3.11) containing the vector potential and the incident field. Show this with the example of the term $\langle \mathbf{E}^i, f_m \rangle$:

$$\langle \mathbf{E}^i, f_m \rangle = \int\limits_S \mathbf{E}^i \cdot f_m dS = \frac{l_m}{2} \left(\frac{1}{A_m^+} \int\limits_{T_m^+} \mathbf{E}^i \cdot \rho_m^+ dS + \frac{1}{A_m^-} \int\limits_{T_m^-} \mathbf{E}^i \cdot \rho_m^- dS \right) \cong$$
$$\cong \frac{l_m}{2} \left(\mathbf{E}^i(r_m^{c+}) \rho_m^{c+} + \mathbf{E}^i(r_m^{c-}) \rho_m^{c-} \right) \tag{3.14}$$

Thus, applying the procedure for testing EFIE RWG functions, taking into account (3.12) - (3.14), we obtain the equation

$$\begin{aligned} i\omega l_m \left[\mathbf{A}(r_m^{c+})\frac{\rho_m^{c+}}{2} + \mathbf{A}(r_m^{c-})\frac{\rho_m^{c-}}{2}\right] + l_m[\Phi(r_m^{c+}) - \Phi(r_m^{c-})] = \\ = l_m \left[\mathbf{E}^i(r_m^{c+})\frac{\rho_m^{c+}}{2} + \mathbf{E}^i(r_m^{c-})\frac{\rho_m^{c-}}{2}\right], \end{aligned} \tag{3.15}$$

which is written for each inner edge, $m = 1, 2, ..., N$. In [1] it is noted that another interpretation of the testing procedure is possible, leading to equation (3.15). We can equate linear integrals of the form $\int\limits_{C_m} F dr$, where F denotes the right and left sides of equation (3.6), and C_m – is a piecewise linear path from the point r_m^{c+} up to the middle of the edge m, and from it to the point r_m^{c-}. The values of $\mathbf{E}^i$ and $\mathbf{A}$ can be approximated along each part of the path by the corresponding values at the centers of mass of the triangles. The resulting equality, which we obtain in this case – is the equation (3.15) without the factor l_m. Within any interpretation, the testing procedure reduces the requirements for the differentiability of the scalar potential Φ in (3.6) due to the fact that $\nabla\Phi$ is first integrated. The aim of the approximations (3.13), (3.14) is to get rid of the surface integrals from the potential values, which allows us to approximate the double integral over the surface by using of a single integral. This, in turn, greatly simplifies the calculation of the elements of the moments matrix, which will be discussed below. On the other hand, to solve equation (3.11), we can use any other quadrature formulas suitable for calculating double integrals.

3.1.3 The matrix equation and the elements of the moment matrix evaluation

After substituting the expansion for the surface current with respect to the basis functions (3.10) into equation (3.15), we obtain a system of linear algebraic equations (SLAE) of size $N \times N$, which can be represented in the form

$$ZI = V, \tag{3.16}$$

where $Z = [Z_{mn}] - N \times N$ matrix, $I = [\alpha_n]$ – is the column of unknown coefficients, $V = [V_m]$ – is the column of the known right-hand side. Elements of the matrix Z and the column V are determined by the following formulas:

$$Z_{mn} = l_m \left[i\omega \left(\mathbf{A}^{+}_{mn} \cdot \frac{\rho^{c+}_{m}}{2} + \mathbf{A}^{-}_{mn} \cdot \frac{\rho^{c-}_{m}}{2} \right) + \Phi^{-}_{mn} - \Phi^{+}_{mn} \right] \tag{3.17}$$

$$V_m = l_m \left(\mathbf{E}^{+}_{m} \cdot \frac{\rho^{c+}_{m}}{2} + \mathbf{E}^{-}_{m} \cdot \frac{\rho^{c-}_{m}}{2} \right), \tag{3.18}$$

where

$$\overset{\pm}{\mathbf{A}}_{mn} = \frac{\mu}{4\pi} \int\limits_{S} f_n(r') \frac{e^{-ik|\overset{c\pm}{r_m} - r'|}}{|\overset{c\pm}{r_m} - r'|} dS', \tag{3.19}$$

$$\overset{\pm}{\Phi}_{mn} = -\frac{1}{4\pi\epsilon i\omega} \int\limits_{S} \nabla'_s f_n(r') \frac{e^{-ik|\overset{c\pm}{r_m} - r'|}}{|\overset{c\pm}{r_m} - r'|} dS', \tag{3.20}$$

$$\overset{\pm}{\mathbf{E}}_{m} = \mathbf{E}^{i}(\overset{c\pm}{r_m}). \tag{3.21}$$

After the elements of the moments matrix Z and the vector V are determined, we can solve the system (3.16) with respect to the vector of unknown coefficients α_n by one of the known methods for solving SLAE.

Elements of Z_{mn} can be calculated directly using formula (3.17) (taking into account (3.19) and (3.20)) for each combination of indices m and n. However, it will be shown below that such an approach is of little effect and a variant is suggested that allows us to reduce the calculations necessary to obtain the moments matrix. The calculation of each element of the matrix Z_{mn} associated with the edges m and n includes the integration over the triangles $\overset{\pm}{T_n}$ with the observation point at the center of mass of the triangles $\overset{\pm}{T_m}$. It is not difficult to verify (see Fig.3.4) that some of the same integrals required for calculating the element of the matrix Z_{mn} are also necessary for calculating the element $Z_{rs}(r \neq m, s \neq n)$ if the edge r is an edge of the triangle T^{+}_{m} or T^{-}_{m}, and the edge s – is an edge of the triangle T^{+}_{n} T^{-}_{n}. Indeed, if we focus on a pair of triangular faces and not on a pair of edges, we note that the integrals computed for the source surface with the vector and scalar potentials observed at the center of mass of the other triangular face are included in all elements of Z_{mn}, including an edge n as an edge of a triangle-source, and an edge m - as an edge of a triangle-observation. Therefore, it is

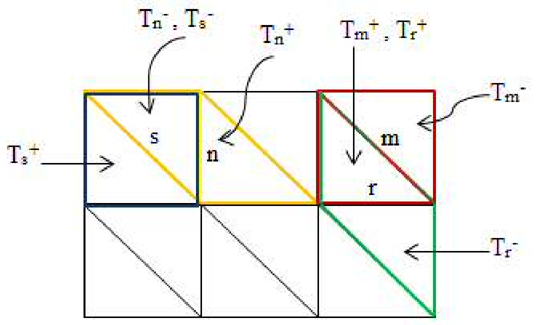


Fig. 3.4: The scheme for calculating the elements of the Z_{mn} matrix of moments

more efficient to calculate the required integrals of potentials by combinations of pairs of faces, rather than directly counting all single elements of Z by combinations of pairs of edges. For each combination of pairs of faces, the integrals of the potentials can be multiplied by the corresponding coefficients in (3.17) and their contributions accumulated in the corresponding elements of Z as they are calculated. In accordance with the above, we consider the calculation of the integral of the vector and scalar potentials for a given pair of faces of the source and observation. Figure 3 shows a pair of faces with an observation point at the center of mass of the triangle T_q and a current source in the triangle T_p. Each of the three-basis functions, which can simultaneously exist in T^q, is proportional to one of the vectors ρ_1, ρ_2 and ρ_3 shown in the figure. Each vector ρ_i, $i = 1, 2, 3$ is depicted directed from the corresponding vertex, but can be directed to the vertex, depending on the direction of the vector in the face adjacent to the edge. We express ρ_i in terms of global coordinates

$$\rho_i = \pm(r' - r_i), i = 1, 2, 3. \tag{3.22}$$

Further, it follows from (3.19) and (3.20) that we need to calculate the following integrals:

$$\mathbf{A}_i^{pq} = \frac{\mu}{4\pi} \int\limits_{T_q} \left(\frac{l_i}{2A^q}\right) \rho_i \frac{e^{-ik|r^{cp}-r'|}}{|r^{cp} - r'|} dS', \tag{3.23}$$

$$\Phi_i^{pq} = \frac{1}{4\pi\epsilon i\omega} \int\limits_{T_q} \left(\frac{l_i}{A^q}\right) \frac{e^{-ik|r^{cp}-r'|}}{|r^{cp} - r'|} dS' \tag{3.24}$$

associated with the i-th basis function on the face i and the observation point at the center of mass of the face p. To calculate the integrals (3.25), (3.26), we can use different quadrature formulas using a triangular master element that uses barycentric coordinates. More information about this can be found in [1,3]. It should also be noted that when the regions of the source and the observation region coincide ($p = q$) or overlap one another, a singularity arises in the integrals, which makes numerical integration impossible and presents a separate difficult problem in the method of moments. In these cases a singularity is usually distinguished in the Green's function and integration is performed analytically. The integrals that arise in this case are computed in [3]. The obtained expressions do not depend on the basis functions and, accordingly, it is sufficient to count them once for each triangular element of the surface.

3.2 Implementation

In this section, we consider the realization of a numerical solution of the problem of electromagnetic wave diffraction on a rectangular metal plate. The implementation consists of the following steps:

1. Surface triangulation and analysis of the resulting geometric structure for RWG elements construction

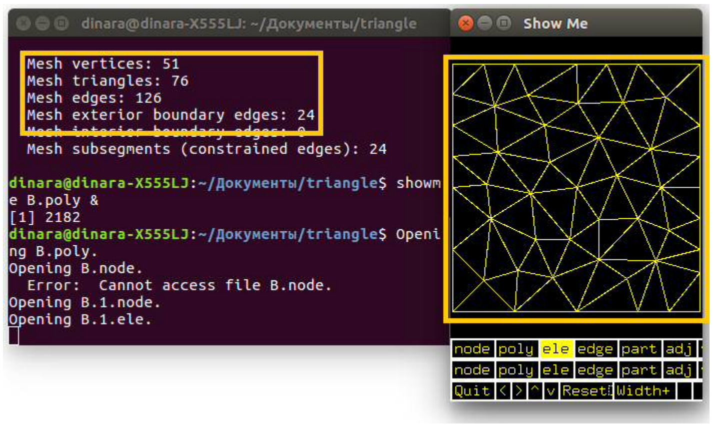
2. Approximate calculation of the vector and scalar potentials, filling the matrix of moments

3. Parallel implementation of the stabilized method of bi-conjugate gradients for the subsequent solution of SLA

3.2.1 Triangulation

The existing methods for constructing a triangular mesh can be divided into two groups according to the method of constructing the grid: direct and iterative. All direct methods have two main distinctive features:

- The construction of the desired grid is carried out in one step.



```
Mesh vertices: 51
Mesh triangles: 76
Mesh edges: 126
Mesh exterior boundary edges: 24
Mesh interior boundary edges: 0
Mesh subsegments (constrained edges): 24

dinara@dinara-X555LJ:~/Документы/triangle$ showm
e B.poly &
[1] 2182
dinara@dinara-X555LJ:~/Документы/triangle$ Openi
ng B.poly.
Opening B.node.
  Error:  Cannot access file B.node.
Opening B.1.node.
Opening B.1.ele.
```

dinara@dinara-X555LJ: ~/Документы/triangle
Show Me
node poly ele edge part adj
node poly ele edge part adj
Quit < > ^ v Reset Width+

Fig. 3.5: Triangulation with "Triangle" library

- The resulting grid is structured, that is, the topology (the link graph between nodes) and the coordinates of all grid nodes are known initially. The most common methods from this group are the methods for constructing a triangular grid based on templates. These methods are intended for triangulation of simple domains of a given type, such as a rectangle, circle, etc. in the two-dimensional case, and a parallelepiped, a cylinder and a prism, etc. - in the three-dimensional case. For each such area, a template is used, that is, a scheme for allocating nodes and establishing links between them.

There are also more universal iterative methods for constructing triangular grids. These methods allow you to automate the construction of triangular meshes in fairly complex areas. That is, there is no need to analytically analyze the area before building a grid. Unlike direct methods, these methods build a triangular grid sequentially, adding one or more elements on each iteration, and neither the node coordinates nor the link graph of these nodes are known in advance. Therefore, the grids constructed by these methods are unstructured. However, this seems to be a small payoff for the universality that has been acquired. Among the iterative methods, we can again distinguish several groups of methods, namely: methods of boundary correction, exhaustion methods, methods based on the Delaunay criterion.

Fig. 3.5 is the result of triangulation using the library "Triangle" developed by J.R.Shewchuk. This is a conformal Delaunay triangulation with a number of specific settings. For the design of complex surfaces, this library

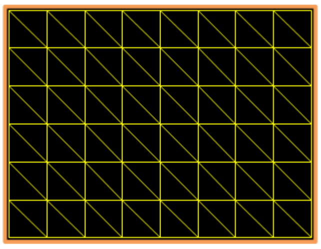


Fig. 3.6: Triangulation with right triangles

shows good results.

In our particular case, when the area is a rectangle, we built our triangulation by the method from the first group. This made it possible to simplify the calculation and analysis of geometry (the construction of a mesh of vertices, edges, triangles and RWG elements), see Fig.3.6.

We briefly describe the triangulation scheme and the analysis of the geometric structure, see Fig.3.7. The input is given the linear dimensions of the plate - width and height and two numbers that discretize these dimensions (they determine how small the triangular mesh will be).

First, we form an array of points that form a triangulation. On the basis of this array, we build an array of triangles, which are given by the numbers of the three points that make up these triangles. For the triangle, the area and centroid (center of mass) are calculated, which will be needed in the subsequent calculations.

Finally, at the last stage, we form the RWG structure. They consist of an inner edge and two adjacent triangles (we indicate their "+" and "-", i.e. 1 or 0), we additionally calculate the points of the beginning and the end of the edge, its length, and also the numbers of free vertices, for calculating the vector $rho(\mathbf{r})$. At the output, we get files "*nodes.txt*", "*triangles.txt*", "*inneredge.txt*" (for RWG structure).

3.2.2 Moment Matrix Calculation

After analysis the geometric structure, we proceed to calculate potential integrals $\mathbf{A}_{mn}^{\pm}$ and $\Phi_{mn}^{\pm}$. To calculate such integrals, we use the following

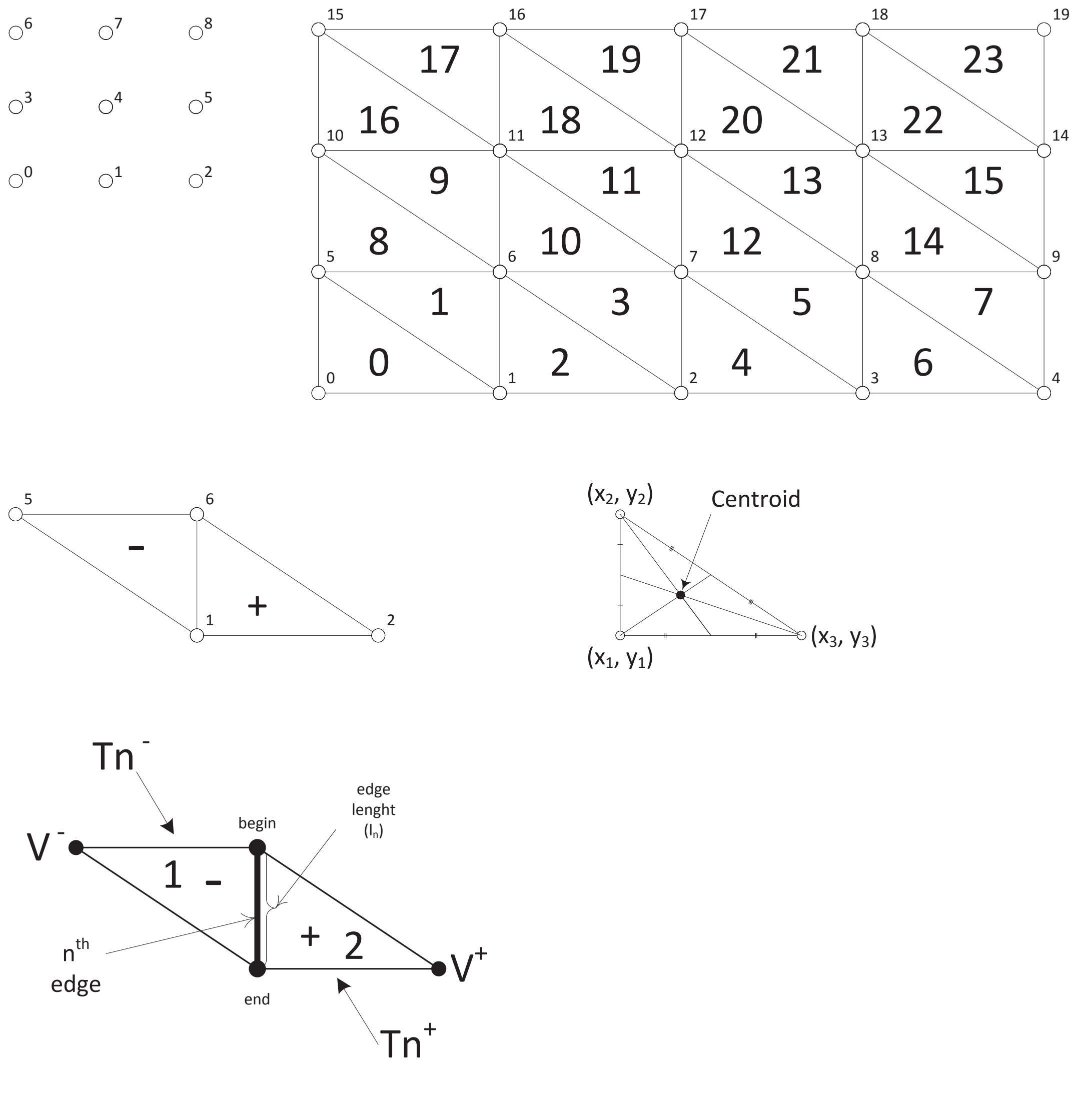


Fig. 3.7: Triangulation with right triangles

quadrature formula:

$$\int\limits_{T_n^+ + T_n^-} g(\mathbf{r})d\mathbf{r} = A_n^+ g(\mathbf{r}_c^+) + A_n^g(\mathbf{r}_c^-),$$

where A_n^{mn} – triangles area, r_c^{mn} – centroid of triangles T_n^{mn}, respectively. Then, the potential integrals are transformed to the form

$$\mathbf{A}_{mn}^{\pm} = \frac{\mu}{8\pi} l_n \left[(r_n^{c+} - v_n^+) \cdot \frac{e^{-ikR_{mn^+}}}{R_{mn^+}} + (v_n^- - r_n^{c-}) \cdot \frac{e^{-ikR_{mn^-}}}{R_{mn^-}} \right]$$

$$\Phi_{mn}^{\pm} = -\frac{1}{4\pi i\omega\epsilon} l_n \left[\frac{e^{-ikR_{mn^+}}}{R_{mn^+}} - \frac{e^{-ikR_{mn^-}}}{R_{mn^-}} \right]$$

Since we integrate a vector function, we separately consider the components $\mathbf{A}_{mn}^x$ and $\mathbf{A}_{mn}^y$ (Φ_{mn}^x and Φ_{mn}^y)

One of the most difficult stages of the method of moments is the computation of potential integrals in cases where the pvr elements coincide or overlap by one triangular face. In this case, numerical integration is impossible and we must perform an accurate calculation of such integrals analytically.

One way to overcome such difficulties is to distinguish a feature in the Green's function. Briefly describe the essence of the used technique. We present the Green function as follow:

$$\frac{e^{-ikr}}{r} = \left[\frac{e^{-ikr}}{r} - \frac{1}{r} \right] + \frac{1}{r}.$$

The first term on the right-hand side when r tends to zero has a constant limit equals $-ik$, and therefore terms with such an integral kernel can be calculated using the approximate formulas described above. As for the distinguished singularity, the new obtained integrals can be written in the form

$$\iint\limits_{T_n} \iint\limits_{T_m} \lambda_i \lambda_j^{'} \frac{1}{|r - r'|} d\lambda_1^{'} \lambda_2^{'} d\lambda_1 d\lambda_2,$$

where λ_i–barycentric coordinate are used for triangular domains. After all the elements of the matrix are calculated, we can only write down the right-hand side of the SLAE, which depends on the type of the incident wave. In our example, we consider a incident field that is directed along the normal to the surface. In this case

$$\mathbf{E}^i(x, y, z) = A_0 e^{-i(k_x x + k_y y + k_z z)}, \quad k_x = k_y = 0 \Rightarrow E^i = A_0 e^{-ikz},$$

and we can write different types of the right side for SLAE, for example, $E_x^i = 0, E_y^i = A_0$ or overwise.

3.3 Parallel implenetation of biconjugate gradient stabilized method

To solve real problems in which SLAE with asymmetric matrices arise, a bi-conjugate gradient stabilized method (BiCGStab) can be used. This iterative method, developed by Van Dur East [9], converges faster than the conventional method of bi-conjugate gradients and is used more often in practice. According to [9], this method has proven itself for solving SLAE with dense, complex and asymmetric matrices, which arise, for example, in the analysis of wire antennas.

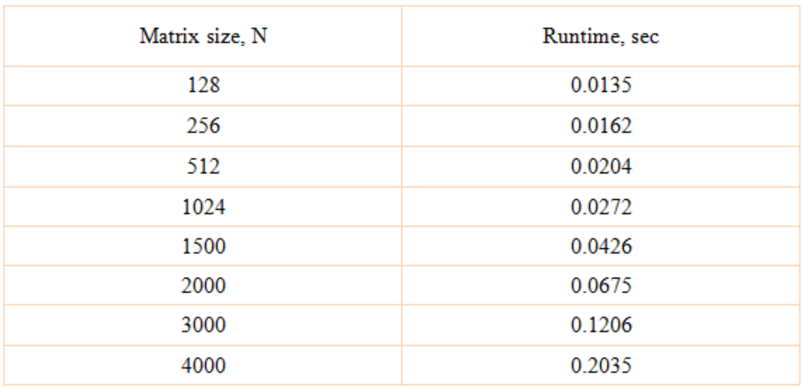
For the solving of the SLAE of the form $Ax = b$, where A is a complex matrix, the following algorithm can be used to stabilize the method of bi-conjugate gradients:

- **preparation before iterative process**

1. We choose the initial approximation x^0, for example, $x^0 = \vec{0} = \{0, 0, ...0\}$
2. We calculate the residual vector $r^0 = b - Ax^0$
3. $\widetilde{r} = r^0$
4. $\rho^0 = \alpha^0 = \omega^0 = 1$
5. $v^0 = p^0 = \vec{0}$

- ***k*th method iteration**

1. $\rho^k = (\widetilde{r}, r^{k-1})$
2. $\beta = \frac{\rho^k}{\rho^{k-1}} \frac{\alpha^{k-1}}{\omega^{k-1}}$
3. $p^k = r^{k-1} + \beta(p^{k-1} - \omega^{k-1} v^{k-1})$
4. $v^k = Ap^k$
5. $\alpha^k = \frac{\rho^k}{(\widetilde{r}, v^k)}$
6. $s^k = r^{k-1} - \alpha^k v^k$
7. $t^k = As^k$
8. $\omega^k = \frac{[t^k, s^k]}{[t^k, t^k]}$



Matrix size, N	Runtime, sec
128	0.0135
256	0.0162
512	0.0204
1024	0.0272
1500	0.0426
2000	0.0675
3000	0.1206
4000	0.2035

Fig. 3.8: Runtime, sec

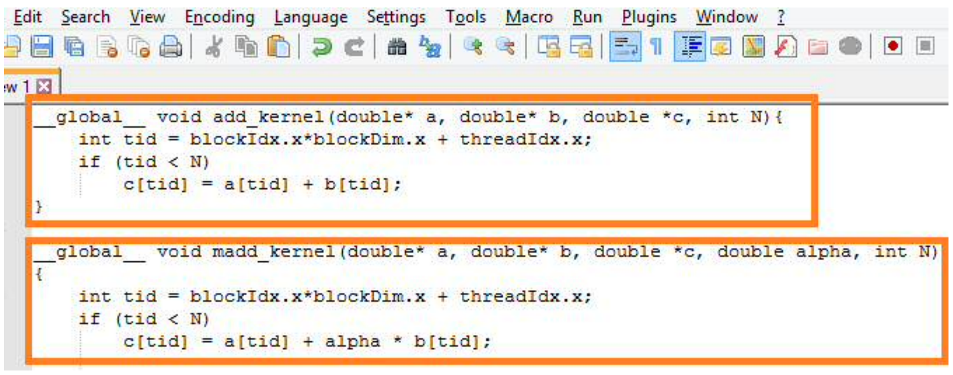
9. $x^k = x^{k-1} + \omega^k s^k + \alpha^k p^k$

10. $r^k = s^k - \omega^k t^k$

Textit Remark. For complex SLAE in the method are used two scalar product $(u, v) = \sum\limits_{i=1}^{n} \overline{u_i} v_i$ and $[u, v] = \sum\limits_{i=1}^{n} u_i v_i$. They coincide in the case of SLAE with real elements.

The stopping criterion for the iterative process can be the number of iterations $k < k_{max}$ or a given discrepancy $(\|r^k\|/\|b\|) < \varepsilon$. In addition, the stopping of the algorithm can be made when the value $|\omega_k|$ is less than some preassigned number.

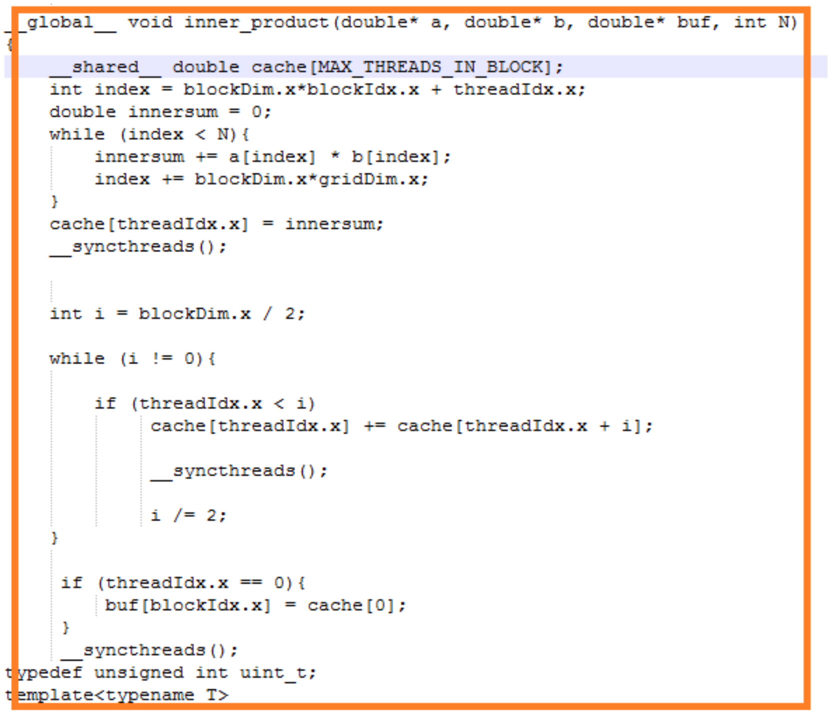
The main performance gain of the parallel version of the algorithm is provided by the use of parallel operations to calculate the scalar product, the product of the matrix by the vector, and also the linear combination of vectors. The algorithm running time and the graph of the dependence of the running time of the parallel BiCGStab algorithm on the SLAE dimension is presented in table (see Fig.3.8). Also we present as example the code of some computing kernels implementing the above operations on the GPU (see Fig.3.9, 3.10).


Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ew 1

```
__global__ void add_kernel(double* a, double* b, double *c, int N){
    int tid = blockIdx.x*blockDim.x + threadIdx.x;
    if (tid < N)
        c[tid] = a[tid] + b[tid];
}

__global__ void madd_kernel(double* a, double* b, double *c, double alpha, int N)
{
    int tid = blockIdx.x*blockDim.x + threadIdx.x;
    if (tid < N)
        c[tid] = a[tid] + alpha * b[tid];
```

Fig. 3.9: Addition operation



```
__global__ void inner_product(double* a, double* b, double* buf, int N)
{
    __shared__ double cache[MAX_THREADS_IN_BLOCK];
    int index = blockDim.x*blockIdx.x + threadIdx.x;
    double innersum = 0;
    while (index < N){
        innersum += a[index] * b[index];
        index += blockDim.x*gridDim.x;
    }
    cache[threadIdx.x] = innersum;
    __syncthreads();

    int i = blockDim.x / 2;

    while (i != 0){

        if (threadIdx.x < i)
            cache[threadIdx.x] += cache[threadIdx.x + i];

            __syncthreads();

            i /= 2;
    }

     if (threadIdx.x == 0){
         buf[blockIdx.x] = cache[0];
     }
     __syncthreads();
typedef unsigned int uint_t;
template<typename T>
```

Fig. 3.10: Scalar product

Conclusion

The method of moments is extremely convenient for modeling microwave devices due to simplicity, mathematical rigor and a sufficiently high computational speed. However, as the pitch of the triangulation decreases, the dimension of the SLAE increases. Therefore, for the solution of such systems, the technologies of distributed computing are attracted. In this paper, we considered the problem of the diffraction of a plane EM wave by a metal rectangular plate. To solve this problem,

1. We studied the existing methods and algorithms for solving electrodynamics problems in the field of antenna's design.

2. We applied the method of moments to solve the diffraction problem formulated in the EFIE form for a rectangular metal plate.

3. We studied and used RWG functions as a basis and testing functions in the method of moments.

4. We made a program implementation of the developed algorithm, included triangulation, numerical integration and construction the matrix of moments.

5. Within the program, we chose the biconjugate gradient stabilized method for solving a system of linear algebraic equations and implemented its parallel version using parallel programming technologies (CUDA).

As for further research, there are several possible directions.

The first is a change the geometry of the radiating element of a microstrip antenna. We plan to consider more complex forms. This will require new methods of triangulation and the using of special libraries, for example CGAL.

Further, to analyze the frequency characteristics of microstrip antennas, it is necessary to take into account more parameters, such as the method of feeding, the characteristics of the substrate and the dielectric, and also the effects that arise during the radiation process.

In the field of computing acceleration, we plan to optimize the calculations on the host using OpenMP technology and try to create some combined method (OpenMP+CUDA).

Bibliography

[1] S.M. Rao, D.R. Wilton, A.W. Glisson “Electromagnetic Scattering by Surfaces of Arbitrary Shape”. – IEEE Transactions on antennas and propagation. – V. AP-30. – No.3. – 1982.

[2] D.H. Schaubert, D.R. Wilton, A.W. Glisson “A Tetrahedral Modeling Method for electromagnetic Scattering by Arbitrarily Shaped Inhomogeneous Dielectric Bodies” ”. – IEEE Transactions on antennas and propagation. – V. AP-32. – No.1. – 1984.

[3] Walton C. Gibson “The method of moments in Electromagnetics”. – Chapman Hall/CRC, Taylor Francis Group. – N.-Y. – 2008.

[4] Constantine A. Balanis “Antenna Theory. Analysis and design” ”. – John Wiley Sons. – New Jersey – 2005.

[5] T.K. Sarkar, A.R. Djordjevic, B.M. Kolundzija “Method of Moments applied to Antennas”. – 2000.

[6] W.L. Stutzman, G.A. Thiele “Antenna Theory and Design” – John Wiley Sons. – USA. – 2013.

[7] Jian Guan “OPEN MP – CUDA implementation of the moment method and multilevel fast multipole algorithm on multi-GPU computing systems”. – Thesis. – Urbana, Illinois. – 2013.

[8] I.P. Molostov, V.V. Sherbinin “Application of NVIDIA CUDA Thechnology for numerical Simulation of Electromagnetic Pulses Propagation”. – IEEE Xplore Digital Library. – 2015.

[9] van der Vorst A. H. *Iterative Krylov Methods for Large Linear System.* – Cambridge Uneversity Press., 2003. – 221 p.

[10] EFFICIENCY COMPARISON OF OpenMP, nVidia CUDA AND StarPU TECHNOLOGIES BY THE EXAMPLE OF MATRIX MULTIPLICATION. – https://cyberleninka.ru/article/n/sravnenie-effektivnosti-tehnologiy-openmp-nvidia-cuda-i-starpu-na-primere-zadachi-umnozheniya-matrits