Libya

Libyan Academy

Janzur-Tripoli-Libya

School of Basic Sciences

Physics Department

Application Of Monte Carlo Method In Solving Poisson's Equation In Comparison With Analytical And Numerical Solutions

By

Ibrahim Ali Daw Ali AlHaj

A dissertation submitted

In conformity for the Department Requirement as Partial Fulfillment for the Degree of Master Physics

Supervised By:

Dr. Taher Sadeq Sherif

Edditted by: Mr. Khalifa S. Ahmad

بسم الله الرحمن الرحيم

وقل رب زدني علماً

صدق الله العظيم

من الآية 114 من سورة طه

DEDICATION

إلى كل محب للعلم وأهله وإلى كل محب أن يعم الأمن و الأمان والتقدم والرقي في بلادنا الحبيبة (ليـبـيـا)

To scientists and everyone who loves science and to everyone who seeks safety, security, development, superiority to our lovely land(Libya).

الشكر والتقدير

أشكر الله وأحمده أولاً، ثم أشكر كل من ساعدني في دراستي أو رسالتي و أخص بالذكر والشكر مشرفي د.الطاهر الصادق الشريف ،ورئيس قسم الفيزياء بالأكاديمية د.محمد سالم الليد ،و د.محمد عبد العزيز، وأخي عبد الخالق وأصدقائي خالد رعدان وعمر السوكني ورمضان الشاملي لمساعدتهم لي، كما و أشكر والدي ووالدتي وزوجتي وأخوي محمد و عبد الباسط لوقوفهم إلى جانبي.

ACKNOWLEDGMENTS

First of all my thanks goes to Allah, whom without him nothing can be done. Then, I thank everyone who supported me through my study and thesis research; especially my supervisor Dr. Taher Sadeq Sherif, and the head of the physics department in the academy Dr. Mohamed Salem Ellid,.Also for their help I thank Dr. Muhammad Abdulaziz, my brother Abdulkhalig, and my friends Khalid Radan, Omar Esokniand Ramadan Eshmli. My deep thanks goes also to my parents , my wife and my brothers Muhammad and AbdAlbaset for their support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

List of tables

ملخص البحث

طريقة المونتي كارلو تم استخدامها بجزئيها (المشي العشوائي الثابت)،(المشي العشوائي المتغير)،في هذا البحث لحل بعض المعادلات التفاضلية الجزئية (لابلاس و بواسون)في تطبيقات مجال الكهرومغناطيسية، و قد تم الحصول على نتائج حل تلك المعادلات باستخدام برنامج الماتكاد، ومن ثم قورنت النتائج المتحصل عليها بطريقة المونتي كارلو ببعض نتائج الطرق التحليلية أو العددية الأخرى، وكانت النتائج المتحصل عليها متقاربة، مما يؤكد فعالية هذه الطريقة.

Abstract

Monte Carlo method has been used with both ways (Fixed Random Walk, Floating Random Walk) in this research to solve some of PDEs (Laplace's equation and Poisson's equation) in the electromagnetic fields, results have been produced by MathCAD software and then these results that used the Monte Carlo method were compared to another analytical or numerical methods results, the comparison showed very closed results which confirm the reliability of this method.

Chapter: 1

Monte Carlo Methods(MCM)

1.1 Introduction

1.2 The advantage of Monte Carlo methods

1.3 The origin of the Monte Carlo methods

1.4 Kinds of Monte Carlo methods

1.5 Applications of Monte Carlo methods

1.6 Distribution functions of probability function

1.1 Introduction

Monte Carlo(MC)simulation is a method that involves random sampling for the mathematical simulation of physical systems. MC calculations are applied to problems that can be formulated in terms of probability and are usually carried out by computers. such calculations have been applied to study the behavior of nuclei, atoms, molecules, solids, liquid, and nuclear reactors.

Monte Carlo Method(MCM) is a numerical method for solving mathematical models by the simulation technique.

1.2 The Advantage of Monte Carlo Methods

Until the 1940s , physical problems were classified into two categories theoretical and experimental. with the advent of the modern digital computer. A third category has emerged namely computer simulation, which complements both theoretical and experimental analysis. The fact that a high degree of ingenuity, skill, and effort are required for theoretical or analytic solutions, only a narrow range of practical problems can be investigated, owing to the complex geometries defining the problems. Also, theoretical analysis often involves some mathematical approximations that are not explainable from first principles. On the other hand, experiments are expensive, time consuming, sometimes hazardous, and do not often allow much flexibility in parameter variation. Computer simulation offers the freedom and flexibility to adjust various parameters, of the system. It has the advantage of allowing the actual work to be carried out by operators without a knowledge of higher mathematics or physics, of course, the three approaches (theoretical, experimental or computer simulation) can be used in solving a given physical problem. [1,2]

Computer simulation of physical problems has led to the development of various numerical techniques. These techniques have become the standard modern tools for solving practical, complex physical problems, as well complicated electromagnetic calculation.

Popular numerical techniques that are commonly used in electromagnetic field include the moment method, finite difference methods, finite element methods, the transmission-line modeling matrix and Monte Carlo methods (MCMs).

The system modeled in Monte Carlo simulation may or may not be deterministic .

It is not true that Monte Carlo methods (such as fixed and floating random walks) generally do not provide accurate solutions. Although Monte Carlo simulation is a slow and costly technique, it has some advantages and is often used when all else fails.

The major advantages of MCMs include the following:

1.Themethods are easy to understand and apply.

2.The MCM Algorithms are simple to develop, allowing one to solve problems with complicated solution regions.

3.Computer code does not require large memory even for multidimensional problems

4.It is possible to obtain the solution directly at an arbitrary point without the whole field computation.

5. It enables one to solve problems with stochastic parameters.

6. Statistical error can be provided simultaneously with the solution.[2]

1.3 The origin of the MCM

The general accepted appearance of the MCM is 1949, when an article entitled “The MCM” by Metropolis and Ulam appeared.[3]

The American mathematicians John Von Neumann and StanisLav Ulam, are considered its main originators . In the Soviet Union, the first papers on the MCM were published in 1955 and 1956 by V.V. Chavchanidze, Yu . A .Shreider and V .S. Vladimirov.

The theoretical foundation of the method had been known long before the Neumann-Ulam article was published. Furthermore, before 1945certainproblems in statistics were sometimes solved by means of random sampling that is, in fact, by the MCM .

Because the simulation of random variables by hand is a laborious process, using the MCM as a universal numerical technique became practical only with the advent of computers .

MCM is applied in two ways namely simulation and sampling.[4]

Simulation refers to the method of providing mathematical imitation of real random phenomena. Atypical example is the simulation of a neutron's motion into a reactor wall, its zigzag path being simulated by a random walk. Sampling refers to the methods of deducing properties of a large set of elements by studying only a small, random subset. For example, the average value of $f(x)$ over $a < x < b$ can be estimated from its average over a finite number of points selected randomly in the interval.

MCM has been applied successfully for solving differential and integral equations, for finding eigen values, inverting matrices, and particularly for evaluation multiple integrals.

MCM has now found numerous applications in many branches of science, engineering, business, and other disciplines.

MCMs also find widespread applications in boundary value problems, semiconductor devices, statistical physics, and quantum field theory.

System models are more precisely characterized in Monte Carlo simulation than in theoretical analysis. (in general). Thus, Monte Carlo simulation are often used to check the accuracy and range of validity of some approximations made in the analytic treatment of a model. The Monte Carlo solution of a problem is closer in spirit to physical experiments than to classical numerical techniques. the Monte Carlo methods developed for the analysis of heat transfer in mechanical engineering have become useful for Monte Carlo solution of electromagnetic problems.

1.4 Kinds of Monte Carlo methods

There are many kinds of Monte Carlo Methods. In this chapter, we will briefly introduce some of these kinds such as Markov chain, kinetic Monte Carlo, Green function, the path integral, The fixed and floating random walks, and Metropolis. The Markov Chain Monte Carlo method is a generic method for approximate sampling from an arbitrary distribution. The main idea is to generate a Markov chain whose limiting distribution is equal to the desired distribution. The Kinetic Monte Carlo techniques a way to study Time-dependent phenomena in systems, where various successive states of the system transit a built-in random walk to simulate the dynamics of multi-particle system. The Green's Function Monte Carlo method is a useful tool for situations in which an exact answer to quantum mechanical problem is desired,[25]. The Fixed and Floating Random Walk methods, are mentioned and illustrated detail in chapter 3. And for the Metropolis Monte Carlo Method, it can be explained as follows: The name of the method back to the name of the mathematician Metropolis Nicholas Constantine. This method is used to simulate the random movement in the space to obtain information about that space, because all random movements are possible with no restriction on these movements. Therefore it is used to know the possibility of something to happen.

1.5 Applications of Monte Carlo methods

The MCM has many tremendous applications in the field of science. It can be used to solve a problem in statistical mechanics, quantum mechanics, simulation of random phenomenon, solving multiple integral and solving the partial differential equations. We will illustrate in this section two of these applications, the solving of the multiple integrals and the simulation of a probability function. The solution of PDE will be discussed in more detail in chapter 2.

1.5.1 Monte Carlo Method in Multiple Integrals

The MCM can be used to solve the integral problem with higher dimensions. It is very powerful method in solving such integrals, where many numerical methods field to solve such integrals. The errors from the MCM reduced with higher dimension. For example the following integral:

$$I_1=\int_0^{0.5}\int_0^{0.7}\int_0^{2.x}\int_0^{2.y}\int_0^{2.x}\frac{x^2 \ +y^3 \ -z}{w^2 \ +1}\ .dwdudzdydx \qquad (1.1)$$

Can be solved by the MCM with the help of Mathcad program as illustrate in Appendix(A1). By MCM the results of integral (1.1) is given as:

$$I_1= -6.211\times 10^{-3} \qquad (1.2)$$

A result from other accurate numerical method is given as:

$$I_1= -6.004\times 10^{-3} \qquad (1.3)$$

Comparing result in (1.2) and (1.3) we find 3.456 % error in case of MCM. For other example:

$$I_2 = \int_0^{0.7}\int_0^{0.8}\int_0^{0.9}\int_0^{1}\int_0^{1.1}\sqrt{x^2 - y^2 - z^2 - u^2 - w^2}\ .dwdudzdydx \qquad (1.4)$$

Can be solved by the MCM with the help of Mathcad program as illustrate in Appendix(A2). By MCM the results of integral (1.4) is given as:

$$I_2=1.18856588 \qquad (1.5)$$

A result from other accurate numerical method is given as:

$$I_2= 1.18878329 \qquad (1.6)$$

Comparing result in (1.5) and (1.6) we find 0,002 % error in case of MCM.

As we can see above, how MCM is effective and efficient in the evaluation of multiple integrals. However, the results obtained from Monte Carlo Method approaches accurate values with higher number of random numbers. Hence, we cannot ignore the importance of the Computer to generate such large random numbers and make the calculations precisely and in very short time.

Note that previous research shows that both classical numerical integration methods and the MCMs yield approximate answers whose accuracy depends on the number of intervals or on the number of trials respectively. The error for all MCM integration methods decreases as $\frac{1}{\sqrt{n}}$ independently of the integral.[17-22,24] Because the computational time is roughly proportional to n in both the classical and MCMs, where for low dimensions, the classical numerical methods such as Simpson's rule are preferable to MCMs unless the domain of integration is very complicated. However, the error in the conventional numerical methods increases with dimension, and MCMs, are essential for higher dimensional integrals where the errors decreases.

1.5.2. Simulation of probability distribution function

Monte Carlo Method is used to Simulate some complex probability distribution functions.

Suppose that we have a group of random numbers r' which have a uniform random distribution probability function p(r') in the range [0,1]. knowing that the function has uniformly random distribution function p(r') for simplicity is given by the following equation:

$$p(r') = 1 \qquad (1.7)$$

Then p(r') can be represented asin Figure 1.1

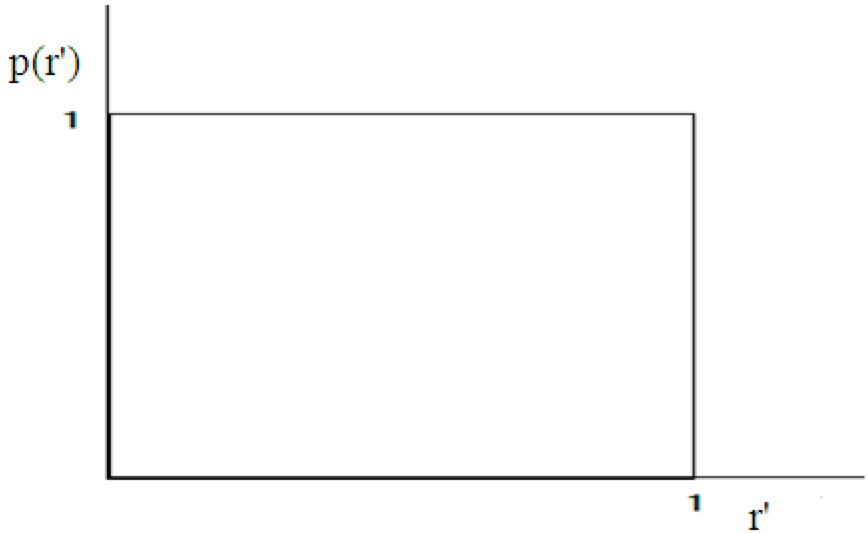

Figure 1.1 shows a uniformly distributed Probability function p(r')

If we want to Simulate a probability function with a non-uniformly distribution, for example a powered probability distribution function which is given by the following equation:

$$p(t') = e^{(-t')}, \text{where t' is a parameter.} \quad (1.8)$$

This powered probability distribution function can be shown as in Figure 1.2

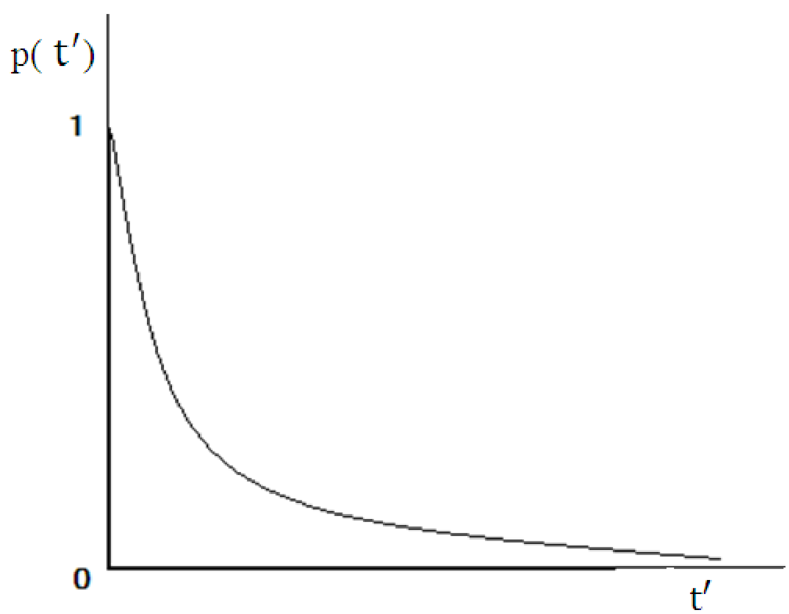

Figure1.2 shows a distribution of a powered probability distribution function p(t').

To perform the Simulate to the powered probability distribution function, we equalize the areas between Figure 1.1 and Figure 1.2 as shown in Figure 1.3:

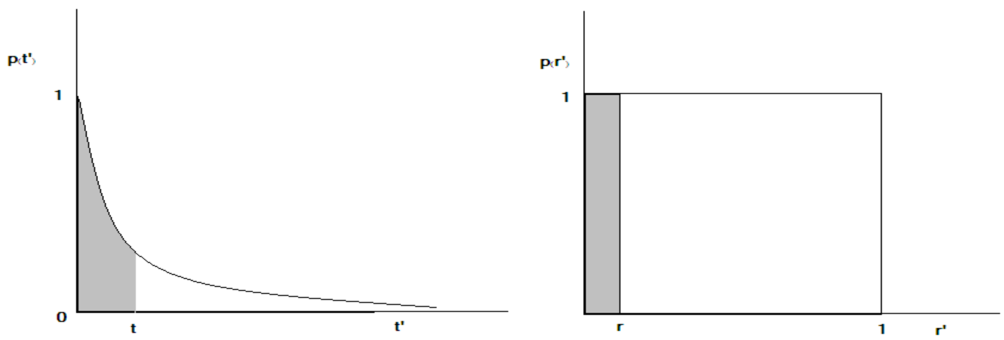

Fig. 1.3 shows how to equalize the area between the two figures.

From the equalization of the two areas we obtain the following relation

$$\int_0^r \mathrm{p}(\mathrm{r}')\, d\mathrm{r}' = \int_o^t \mathrm{p}(\mathrm{t}')\, d\mathrm{t}' \qquad (1.9)$$

By substitution equation (1.7) and equation (1.8) in equation (1.9) we get:

$$\int_0^r d\mathrm{r}' = \int_o^t e^{(-\mathrm{t}')} d\mathrm{t}' \qquad (1.10)$$

By perform the integration we get:

$$\mathrm{r} = 1 - \mathrm{e}^{-\mathrm{t}} \qquad (1.11)$$

Solve for t we get:

$$\mathrm{t} = -\ln(1-\mathrm{r}) \qquad (1.12)$$

where equation (1.9) shows an equal areas between a powered probability distribution function and a uniform probability distribution function.

Where r is a random number with a systematical distribution, t is a variable in relation with the powered distribution function, and the distribution of the variable t can be determined from the gradual repeat, which is obtained from equation (1.12) and by the computational simulation with the help of Mathcad software, as illustrated in Appendix (1C). The obtained result is illustrated in Figure 1.4 which shows the powered distribution.

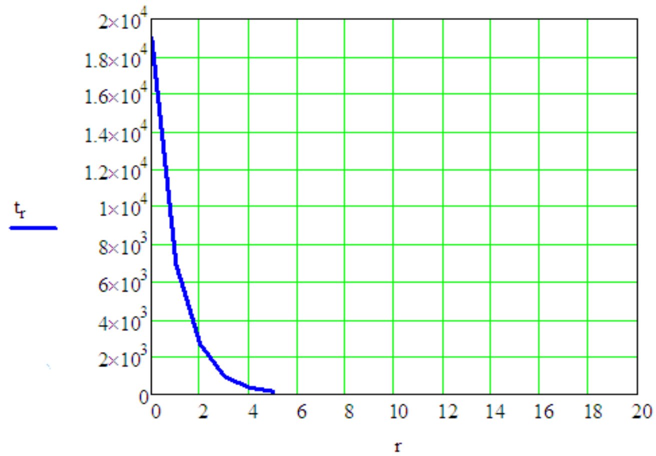

Figure 1.4 which shows the powered distribution.

For another example, if we want to simulate a probability function with a non- uniformly distribution, such as sin probability distribution function which is given by the following equation:

$$P(\theta') = \frac{1}{2}(sin\theta') \text{ , where } \theta' \text{ is a parameter.} \quad (1.13)$$

This sin probability distribution function can be shown as in Figure 1.5

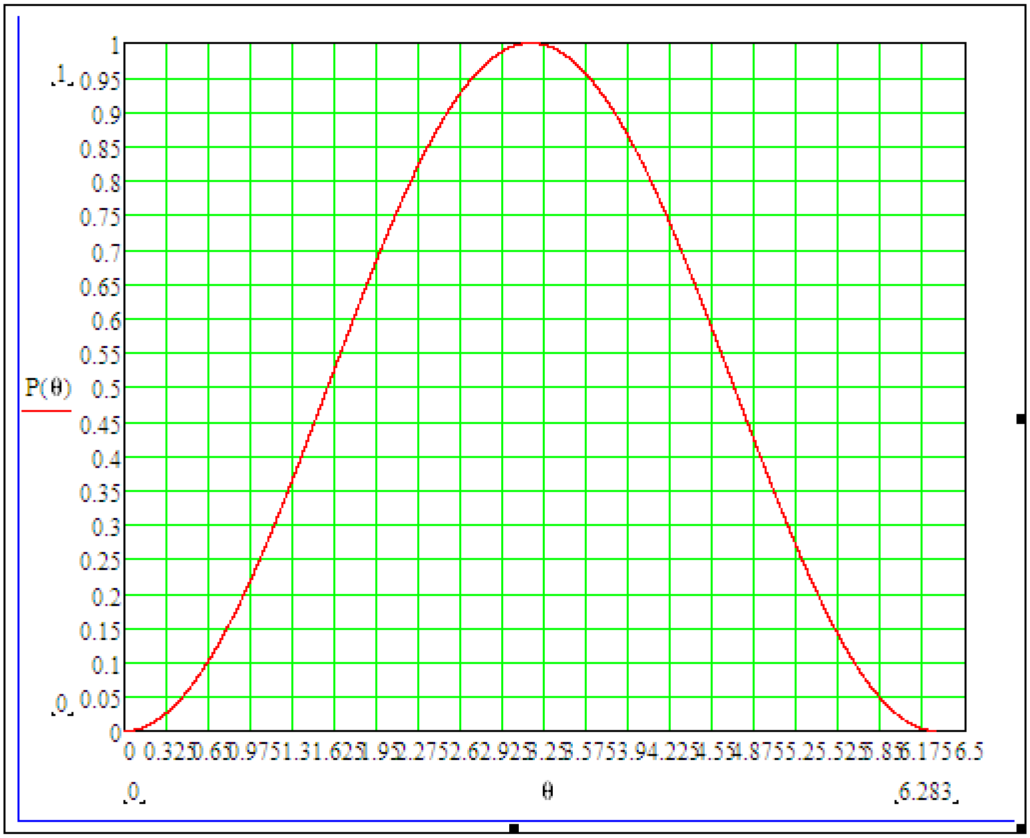

Figure1.5 shows a distribution of a sin probability distribution function p(θ′).

To perform the Simulate to the sin probability distribution function, we equalize the areas between Figure 1.1 and Figure 1.5 as shown in Figure 1.6:

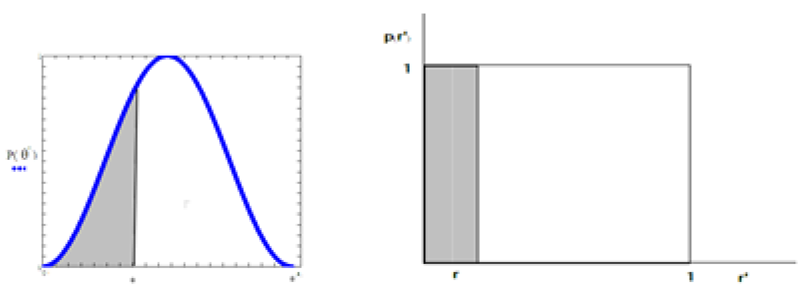

Fig. 1.6 shows how to equalize the area between the two figures.

From the equalization of the two areas we obtain the following relation

$$\int_0^r p(r')\, dr' = \int_o^\theta p(\theta')\, d\theta' \qquad (1.14)$$

By substitution equation (1.7) and equation (1.13) in equation (1.14) we get:

$$\int_0^r dr' = \int_o^\theta \frac{1}{2}(sin\theta')d\theta' \qquad (1.15)$$

By perform the integration we get:

$$r = \frac{1}{2}(1 - \cos\theta) \qquad (1.16)$$

Solve for θ we get:

$$\theta = \cos^{-1}(1 - 2r) \qquad (1.17)$$

where equation (1.14) shows an equal areas between a sin probability distribution function and a uniform probability distribution function.

Where r is a random number with a uniform distribution, θ is a variable in relation with the sin distribution function,

and the distribution of the variable θ can be determined from the gradual repeat, which is obtained from equation (1.17) and by the computational simulation with the help of Mathcad software, as illustrated in Appendix (1d). The obtained result is illustrated in Figure 1.7 which shows the sin distribution.

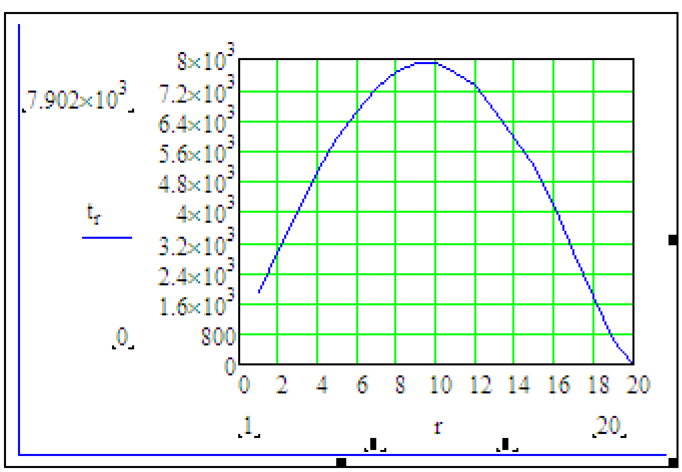

Figure 1.7 shows the sin distribution.

We note similarities between the two Figures, Figure1.2, Figure 1.4,and also the two Figures,Figure1.5,Figure1.7,probability distribution functions and generation random number distribution which using in the Simulation.

CHAPTER: 2

PARTIAL DIFFERENTIAL EQUATIONS

2.1 Introduction

2.2 Classification Of Partial Differential Equations

2.3 Importance Of Partial Differential Equations

2.4 The Methods Of Solution Of Partial Differential Equations

2.1 Introduction

In general, differential equations are very interesting subject in physics, where a differential Equation with boundary conditions or initial conditions can describe a physical problems. The solution of the differential equation explains large number of physical phenomena, and engineering applications.

An equation containing an unknown function with its derivatives form a differential equation. If the unknown function depend on only one independent variable then the differential equation called ordinary differential equation(ODE). But if the unknown function depends on more than one independent variable then the differential equation called a partial differential equation(PDE).

Most physical phenomena such as waves, fluid motion, magnetic field, electric field, heat flow, and quantum physical phenomena, can be described in general by partial differential equations. The most famous PDEs in physics are the classical wave equation used in classical physics, the Maxwell's equations used in describing classical electromagnetic phenomena, and the Schrodinger's equation, which used to describe the quantum physical phenomena.

2.2 Classification Of Partial Differential Equations

Most physical problems can be described by Partial differential equations. Partial differential equations are classified according to many things. Classification is an important concept because it simplify the solution of the PDE.

There are six basic classifications to the PDEs as follows:

- Order of the PDE. The order of a PDE is the order of the highest partial derivative in the equation.
- The number of independent Variables.
- Partial differential equations are either linear or nonlinear. In the linear case, the dependent variable u and all its derivatives appear in a linear fashion (they are not multiplied together or squared, for example). More precisely, a second-order linear equation in two variables can be `written as:

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G(x, y) \qquad (2.1)$$

Where A, B, C, D, E, F, and G can be constants or given functions of x and y.

- Homogeneity. Equation (2.1) is called homogeneous if right-hand side is identically zero for all x and y. If $G(x, y)$ is not identically zero, then the equation is called nonhomogeneous.
- Kinds of Coefficients. If the coefficients A, B, C, D, E, and F in equation (2.1) are constants, then (2.1) is side to have constant coefficients (otherwise, variable coefficients).

There are three Basic Types of linear Equation. All linear PDEs like equation (2.1) are either

(a) Parabolic PDE: Where the property $B^2 - 4AC = 0$ is satisfied.

Equation with this property are the heat flow and the diffusion equation.

(b) Hyperbolic PDE: Where the property$B^2 - 4AC > 0$ is satisfied.

Equation with this property are the vibrating system and the wave equation.

(c) Elliptic PDE: Where the property $B^2 - 4AC < 0$ is satisfied, and the steady-state phenomena can be described.

In Table (2.1) we summarize the mathematical forms of the most famous PDEs which used in physics and engineering science.

Table 2.1 some important physical partial differential equations

Name of the equation	The form of the equation
The classical wave equation Used in classical physics	$\nabla^2\phi(r,t) - \frac{1}{v^2}\frac{\partial^2}{\partial t^2}\phi(r,t) = 0$ Where v is velocity of the wave.
Helmholtz equation	$\nabla^2\phi(r) + k^2\phi(r) = 0$
The diffusion equation	$\nabla^2\psi(r,t) = \frac{1}{a^2}\frac{\partial}{\partial t}\psi(r,t)$
The time-dependent Schrödinger Equation Used in quantum mechanics	$-\frac{\hbar^2}{m}\nabla^2\psi(r,t) + V\psi(r,t) = i\hbar\frac{\partial\psi(r,t)}{\partial t}$
The time-independent Schrödinger Equation Used in quantum mechanics	$-\frac{\hbar^2}{m}\nabla^2\psi(r) + V\psi(r) = E\psi(r)$
Poisson's Equation Used in electromagnetic	$\nabla^2 u(x,y) = -g(x,y)$
Laplace's Equation Used in electromagnetic	$\nabla^2 u(x,y) = 0$

2.3 Importance Of Partial Differential Equations

In addition to what is stated in the introduction, here we will state the most important PDEs such as Diffusion Equation, Wave Equation, Helmholtz Equation, time-independent diffusion equation, Schrödinger Equation, Poisson's Equation and Laplace’s Equation.

The Poisson's Equation describes (for example) the steady-state temperature distribution in a solid in which there are heat sources, i.e. in regions containing matter, charges or sources of heat or fluid. In this case we can write this equation as:

$$\nabla^2 u(x, y) = f(x, y) \qquad (2.2)$$

Where f is a function that depends only on the space variables.

In Chapter 4, we will solve Physical problems in relation to Laplace and Poisson by using Monte Carlo Methods.

2.4 Solution Methods for Partial Differential Equations

There are many solution methods for PDEs, the following are part of them:

1- Separation of Variables: this technique reduces a PDE in n variables to n ODEs (Ordinary differential Equations).

2- Integral Transforms: this procedure reduces a PDE in (n) independent variables to one in $(n-1)$ variables; hence, a PDE in two variables could be changed to an ODE.

3- Change of Coordinates: this method changes the original PDE to an ODE or else another PDE (an easier one) by changing the coordinates of the problem.

4- Transformation of the Dependent Variable: this method transforms the unknown of a PDE into a new unknown that is easier to find.

5- Numerical Methods: these methods change a PDE to a system of difference equations that can be solved by means of iterative techniques on a computer.

6- Perturbation Methods: this method changes a nonlinear problem into a sequence of linear ones that approximates the nonlinear one.

7- Integral Equations: this technique changes a PDE to an integral equation (an equation where the unknown is inside the integral). The integral equation is then solved by various techniques.

8- Calculation of Variation Methods: these methods find the solution to PDEs by reformulating the equation as minimization problem. It turns out that the minimum of a certain expression (very likely the expression will stand for total energy)is also the solution to the PDE.

2.5 The solution of Poisson's equation by analytic method

We can use Green's function technique to solve the PDE. The Green's function method is used to solve the nonhomogeneous PDE with homogenous boundary condition or homogenous PDE with nonhomogeneous boundary condition.

Consider The nonhomogeneous PDE (Poisson's Equation) in two dimensions

$$\nabla^2 u(x,y) = f(x,y) \qquad (2.3)$$

With interval

$$0 \leq x \leq a \text{ ,and } \quad 0 \leq y \leq b \qquad (2.4)$$

Then the Green's function for Poisson's equation Eq(2.3) can be written as:

$$\nabla^2 G(x,y,x0,y0) = -\delta(\mathrm{x}-\mathrm{x0})\delta(y - y0) \qquad (2.5)$$

The B. C. for Eq(2.5) is given by:

$$u(0,y) = 0,\ u(a,y) = 0, u(x,0) = 0, \text{ and } u(x,b) = 0 \qquad (2.6)$$

For this case the Green's function for Eq(2.5) is given by:

$$G(x,y;x0,y0) = 4\frac{ab}{\pi^2}\sum_{nm}\frac{1}{(nb)^2+(ma)^2} sin\left(\frac{n\pi x}{a}\right)\times$$

$$sin\left(\frac{m\pi y}{b}\right) sin\left(\frac{n\pi x0}{a}\right)\ sin\left(\frac{m\pi y0}{b}\right) \qquad (2.7)$$

Then the solution for equation (2.3) can be written as:

$$u(x0,y0) = \int_0^a \int_0^b f(x,y)G(x,y;x0,y0)\ dxdy \qquad (2.8)$$

The detail solution for the Equation (2.3) is written in Appendix (2.A)

CHAPTER: 3

Random Walks of Monte Carlo Methods

3.1 Introduction

3.2 Fixed Random Walk

3.3 Floating Random Walk

3.1 Introduction

In this chapter, the random walk method will be introduced. Random walk is the underlying concept of Monte Carlo solution of differential equations. Different types of random walks lead to different Monte Carlo methods. Random walks in our study will contain the Fixed and Floating walk types. the main difference is that the fixed random walk consume a lot of time comparing to the floating one. The reason is that, in fixed random walk the step size is fixed and the steps of walks are constrained to lie parallel to the coordinate axes. As a result, to obtain an accurate results, fixed random walk required large number of walks with a relatively small step size, and consequently takes a considerable amount of time. Floating random walk has a floating steps and requiring a few number of steps per walk, therefore it can remove the time consumption drawback of using fixed random walk.

3.2 Fixed Random Walk

In this section we will introduce the procedure in the fixed random walk Monte Carlo method to solve Laplace's equation and Poisson's Equation. For solving Laplace's Equation

$$\nabla^2 V(x,y)=0 \qquad \text{in region R} \tag{3-1}$$

Subject to the Dirichlet boundary condition

$$V(x,y)=V_b \qquad \text{on the boundary B} \tag{3-2}$$

We consider a rectangular shaped, homogeneous, two dimensional finite region subjected to prescribed potential at the boundaries (i. e., Dirichlet condition).

To determine the potential at any point in the region, we apply Taylor's series for the first and second derivatives of V with respect to x and y, (as in appendix 3.A),and we obtain on

$$\frac{\partial^2 V}{\partial x^2} = \frac{V(x+\Delta,y)-2V(x,y)+V(x-\Delta,y)}{\Delta^2} \tag{3-3}$$

And $$\frac{\partial^2 V}{\partial y^2} = \frac{V(x,y+\Delta)-2V(x,y)+V(x,y-\Delta)}{\Delta^2} \tag{3-4}$$

Substituting (3-3) and (3-4) into (3-1) gives the finite difference representation of Laplace's equation, i.e.,

$$0 = \frac{V(x+\Delta,y)-4V(x,y)+V(x-\Delta,y)+V(x,y+\Delta)+V(x,y-\Delta)}{\Delta^2} \qquad (3\text{-}5)$$

Solving for V(x,y) yields,

$$V(x,y) = p_{x+}V(x+\Delta,y) + p_{x-}V(x-\Delta,y) + p_{y+}V(x,y+\Delta) + p_{y-}V(x,y-\Delta) \qquad (3\text{-}6)$$

Where $p_{x+} = p_{x-} = p_{y+} = p_{y-} = \frac{1}{4}$ (3-7)

A probabilistic interpretation of the finite difference equation (3-6) and equation (3-7) is as follows. When we desire the potential at the point (x,y)consider a random walking particle instantaneously situated at the point (x,y), and prepared to step to one of the four neighboring nodal points, it has probabilities p_{x+}, p_{x-}, p_{y+}, and p_{y-} of moving from (x,y), to $(x+\Delta,y)$,$(x-\Delta,y)$,$(x,y+\Delta)$, and $(x,y-\Delta)$, respectively. A means of determining which way the particle should move is to generate a random number r, $0 < r < 1$, and instruct the particle to walk as follows

If $0 \le r < 0.25$ particle steps from (x,y) to $(x+\Delta,y)$

If $0.25 \le r < 0.50$ particle steps from (x,y) to $(x-\Delta,y)$

If $0.50 \le r < 0.75$ particle steps from (x,y) to $(x,y+\Delta)$

If $0.75 \le r < 1$ particle steps from (x,y) to $(x,y-\Delta)$ (3-8)

If a rectangular grid rather than a square grid is employed, then $p_{x+} = p_{x-}$and $p_{y+} = p_{y-}$, but $p_x \neq p_y$. Also, for a three-dimensional problems in which cubical cells are used, then$p_{x+} = p_{x-} = p_{y+} = p_{y-} = p_{z+} = p_{z-} = \frac{1}{6}$. In both cases, the random number r is subdivided according to the probabilities.

To calculate the potential at (x,y) a random walking particle is instructed to start at the point. The particle proceeds to wander from node to another in the grid until it reaches the boundary. When it does, the walk is terminated and the prescribed potential V_b at that boundary is recorded. Let the value of V_b at the end of the first walk be denoted by V_b (1), as illustrated in Fig. 3.1. Then a second particle is released from (x,y) and allowed to wander until it reaches a boundary point where the walk is terminated and the corresponding value of V_b is recorded as $V_b(2)$. This procedure is repeated for the third, fourth, …, and Nth Particle released from (x,y) and the corresponding prescribed potential $V_b(3)$, $V_b(4)$, …., $V_b(N)$ are noted. the solution of the Dirichlet problem at (x,y), i.e.,

$$V(x\,,y) = \frac{1}{N}\sum_{i=1}^{N} V_b(i) \qquad (3\text{-}9)$$

Where N, the total number of walks, is large. The rate of convergence varies as the square root of N so that many random walks are required to ensure accurate result.

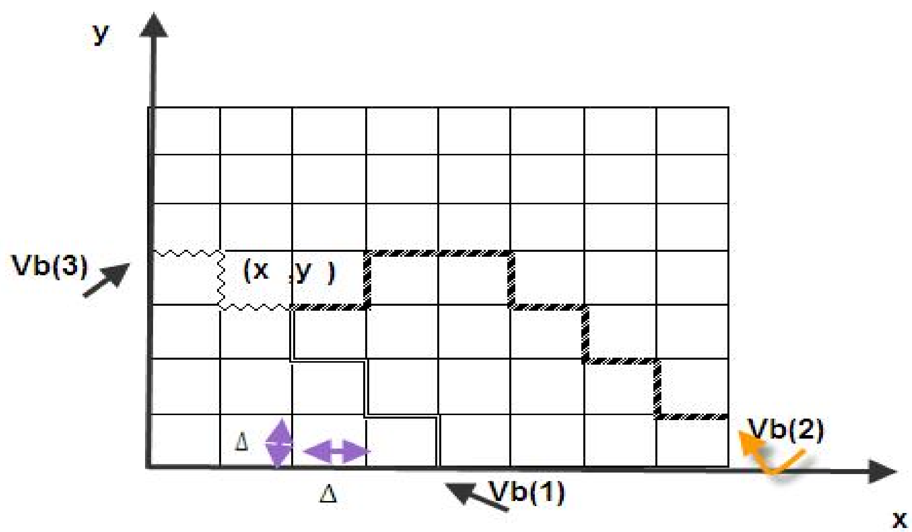

Fig 3.1configuration for fixed random walk in two dimensions

If it is desired to solve Poisson's equation

$$\nabla^2 V(x,y)=-g(x,y) \qquad \text{in region } R \qquad (3\text{-}10)$$

Subject to $V(x,y)=V_b$ on the boundary B (3-11)

then the finite difference representation becomes

$$V(x,y) = p_{x+}V(x+\Delta,y) + p_{x-}V(x-\Delta,y) + p_{y+}V(x,y+\Delta) +$$

$$p_{y-}V(x,y-\Delta) + \frac{\Delta^2 g}{4} \qquad (3\text{-}12)$$

Where the probabilities remain as stated in (3-8)and (3-9).

The probabilistic interpretation of (3-12) is similar to that for (3- 8). However, the term $\Delta^2 g/4$ in (3-12) must be recorded at each step of random walk. If g is constant and mi steps are required for the ith random walk originating at (x, y) to reach the boundary, then one records

$$V_b(i) + m_i\left(\frac{\Delta^2 g}{4}\right) \qquad (3\text{-}13)$$

Thus, the Monte Carlo result for V(x, y) is

$$V(x,y) = \frac{1}{N}\sum_{i=1}^{N} V_b(i) + \frac{\Delta^2 g}{4}.\frac{1}{N}\sum_{i=1}^{N} m_i \qquad (3\text{-}14)$$

This MCM is called random walk type since the step size Δ is fixed and the steps of the walks are constrained to lie parallel to the coordinate axes. Unlike in the finite difference method (FDM) where the potential at all mesh points is determined simultaneously, MCM is able to solve for the potential at any isolated point in the solution region. One disadvantage of this MCM is that it is slow, and is therefore recommended for solving problems for which only a few potentials are required. It shares a common difficulty with FDM in connection with irregularly shaped bodies having Neumann boundary conditions. This drawback is fully removed by employing MCM with floating random walk.

2.3 Floating Random Walk

The mathematical basis of the floating random walk method is the mean value theorem of potential theory. To apply Floating random walk in solving poison's equation

$$\nabla^2 V(x,y) = -g(x,y) \quad \text{in region R} \qquad (3\text{-}15)$$

Subject to Dirichlet boundary condition.

$$V(x,y) = V_b \quad \text{on the boundary B} \qquad (3\text{-}16)$$

If S is a sphere of Radius a, centered at (x, y, z), which lies wholly within region R, then to apply the floating random walk in solving Poisson’s equation

$$V(x,y,z) = \frac{1}{4\pi a^2}\int_s V(\grave{r})dS` + \frac{1}{4\pi}\int_v \frac{g(\grave{r})}{|r-\grave{r}|}d\grave{v} \qquad (3\text{-}17)$$

(the first part of the equation is to solve Laplace's equation)

Equation (3-17) can be given a probabilistic interpretation. The value V(x,y,z) at the center of the sphere is the average of g with respect to green's function plus the uniform average of the boundary values on the surface of the sphere. When the potential varies in two dimensions, V(x,y) is given by

$$V(x,y) = \frac{1}{4\pi a}\oint_l V(\grave{\rho})d\grave{\rho} + \frac{1}{2\pi}\int_s g(\grave{\rho})\ln|\rho - \grave{\rho}|d\grave{s} \qquad (3\text{-}18)$$

Where the integration is around a circle of radius a centered at (x, y). Alternatively to (3-17) and (3-18) are :

$$V(x,y,z) = \int_0^1\int_0^1 [V(a,\theta,\emptyset) + a^2g/6]\,dFdT \qquad (3\text{-}19)$$

$$V(x,y) = \int_0^1 [V(a,\emptyset) + a^2g/4]\,dF \qquad (3\text{-}20)$$

Where

$$F = \frac{\emptyset}{2\pi}\ ,T = \frac{1}{2}(1 - \cos\theta) \qquad (3\text{-}21)$$

And θ and ∅ are regular spherical coordinate variables. The functions F and T may be interpreted as the probability distributions corresponding to θ and ∅. While dF/d∅ = constant. dT/dθ = ½ sin θ. i.e. all angles ∅ are equally probable, but the same is not true for θ.

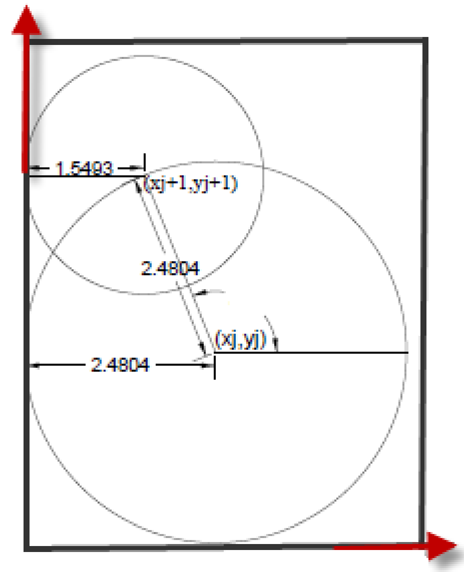

Fig. 3.2 Calculation of next step in each walk

The floating random-walk MCM depends on the application of (3-17) and (3-18) in a statistical sense. For a two-dimensional problem, suppose that a random-walking particle is at some point (x_j,y_j) after j steps in the ith walk. The next (j+1)th step is taken as follows. First, a circle is constructed with center at (x_j,y_j) and radius r_j which is equal to the short distance between (x_j,y_j) and the boundary. The $\emptyset$ coordinate is generated as a random variable uniformly distributed over $(0,2\pi)$, i.e. $\emptyset = 2\pi U$, where $0 < U < 1$. Thus the location of the random walking particle after the jth step is illustrated in Fig. 3.2 and gives as:

$$x_{j+1} = x_j + r_j \cos\emptyset_j$$

$$y_{j+1} = y_j + r_j \sin\emptyset_j \qquad (3\text{-}22)$$

The value of $[g(x,y).\ r^2]$ is recorded at this old location. The next random walk is executed by constructing a circle centered at $(x_{j+1},\ y_{j+1})$ and of radius r_{j+1}which is the shortest distance between$(x_{j+1},\ y_{j+1})$ and the boundary. This procedure is repeated several times and the walk is terminated when the walk approaches some prescribed small distance δ from the boundary. The potential $V_b(i)$ at the closest boundary point at the end of this ith walk is recorded as in the fixed random-walk MCM and the potential at $(x0\ ,y0)$ is eventually determined after N walks using

$$V(x0\ , y0) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{V_b}\ (i) + \frac{1}{4N} . \sum_{i=1}^{N}\sum_{j=1}^{m_i} \mathrm{g}(x_j, y_j)\ \mathrm{r_j^2} \qquad (3\text{-}23)$$

The floating random walk MCM can be applied to a three-dimensional problem involving Poisson's equation by proceeding along lines similar to those outlined above. A random-walking particle at (x_j, y_j, z_j) will step to a new location on the surface of a sphere whose radius r_j is equal to the shortest distance between point (x_j, y_j, z_j) and the boundary. The $\emptyset$ coordinate is selected as a random number U between 0 and 1, and solving another random number U between 0 and 1, and solving for $\theta = \cos^{-1}(1\text{-}2U)$. Thus the location of the particle after (j+1)th step in the ith walk is:

$$x_{j+1} = x_j + r_j \cos\emptyset_j \sin\theta_j$$

$$y_{j+1} = y_j + r_j \sin\emptyset_j \sin\theta_j$$

$$z_{j+1} = z_j + r_j \cos\theta_j \qquad (3\text{-}24)$$

Finally, the Potential at (x_0, y_0, z_0) is determined as:

$$V(x_0, y_0, z_0) = \frac{1}{N}\sum_{i=1}^{N} V_b(i) + \frac{1}{6N} . \sum_{i=1}^{N}\sum_{j=1}^{m_i}[g(x_j, y_j) r_j^2] \qquad (3\text{-}25)$$

Chapter 4:

Application of MCM in Solving Some Elliptic Partial Differential Equations

4.1 Applying Monte Carlo Method in 2 Dimensions (2D)

4.2 Applying Monte Carlo Method in 3 Dimensions (3D)

4.1 Applying Monte Carlo Method in 2 Dimensions (2D)

This chapter introduces a practical application to the usage of Monte Carlo Methods to solve elliptic partial differential equations under Dirichlet boundary condition value (Poisson and Laplace equations). For explanation we will use a number of problems, in 2 and 3 dimensions, and then compare the results obtained through one MCM with another MCM (Fixed with Floating) or with analytical solution (Exact) or with another numerical solution.

If we have a square sheet piece with two dimensions as shown in Figure 4.1 with a length of 1 unit in the direction of Y axis and 1 unit in the direction of X axis and under the Dirichlet boundary conditions V(0,y) = v(1,y) = v(x,1) = 0, v(x,0) = 100 volts and fits Laplace equation, and with the requirement of determining the voltage at any point within this metal piece. This problem has an exact solution[19]by separation of variables complying with the equation:

$$V(x,y) = -\frac{4V_0}{\pi}\sum_{n=1}^{\infty}\frac{\sin((2n-1)\pi x)\,\sinh(\frac{(2n-1)\pi(y-b)}{a})}{(2n-1)\,\sinh(\frac{(2n-1)\pi b}{a})}, \qquad (4.1)$$

Where a = b =1

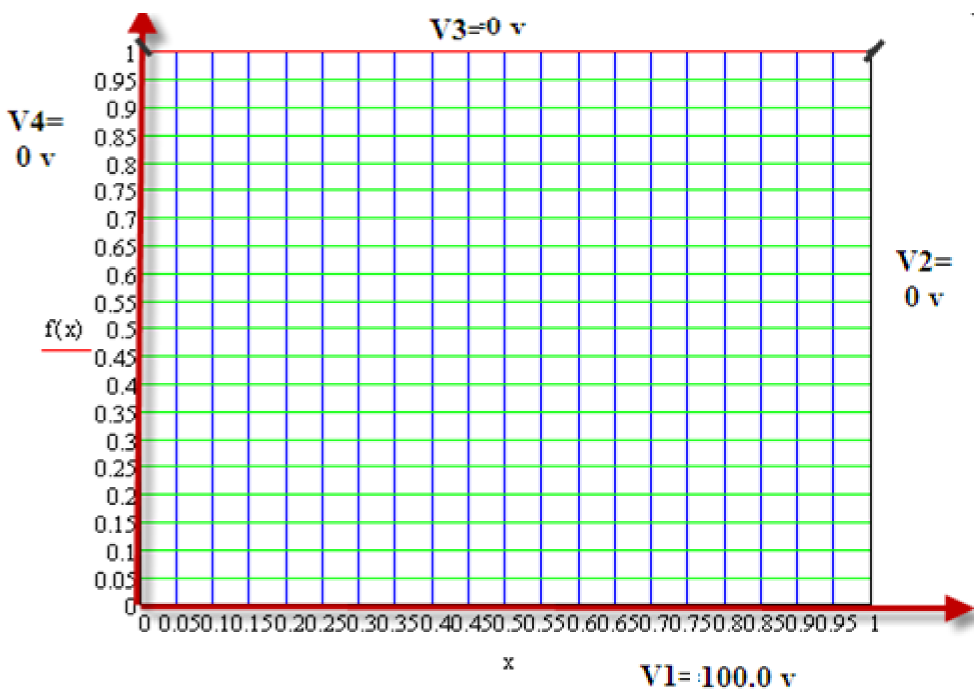

Figure 4.1 Square metal piece with 2 dimensions

We used this equation that has exact solution, to compare the results, which we determine through the fixed and Floating methods of Monte Carlo, with the Exact solution by applying fixed method to solve this problem using a prepared Mathcad program for this purpose (refer to Appendix 4.A), we obtained a group of results. These results were compared with the Exact Solution (see Table 4.1) and we found that the error was small and overall comparison result was good. Knowing that the obtained results are built on a random numbers and it is under the mean value Theorem of voltage or work, we considered the usage of the Mean for the results to be more settled with a variation in the number of inserted random numbers into the computation program.

Table 4.1 Exact solution and MCM with Fixed Random walk results

V(x,y)	N	mean steps(m)	Fixed random walk	Exact solution	Errors(%)
V(0.5,0.5)	1500	113	25.16	25.0	0.6
V(0.5,0.5)	5000	120.86	24.91	25.0	0.03
V(0.25,0.5)	5000	89.08	18.1	18.203	0.5
V(0.25,0.5)	1000	90	18.375	18.203	0.94
V(0.25,0.5)	2700	90	18.59	18.203	2.14
V(0.25,0.25)	1500	70	44.15	43.203	2.19
V(0.5,0.75)	1500	95	9.48	9.475	0.6
V(0.5,0.75)	5000	94.18	9.54	9.475	0.7

Furthermore, when we were solving the same previous problem under the same boundary conditions and the same potential value by the insertion of floating method (As shown in Figure 4.2), we could obtain a good results and which is close to the exact solution with a small errors (See Table 4.2). We obtained these results using the computer program (MathCAD) and attached in (Appendix 4.B). Also by comparing the results of Fixed method to those of Floating to the same problem we can notice that the floating random method provides a more accurate solution in less time compared with the fixed random walks.

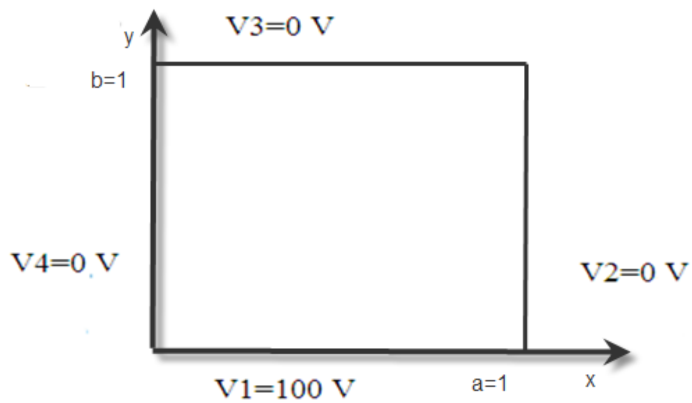

Fig.4.2 Usage of Floating Method

Table.4.2 Comparing the Results of Floating Method with Exact solution

V(x,y)	N	mean steps(m)	Floating random walk	Exact solution	Errors(%)
V(0.25,0.25)	5000	6.15	42.8	43.2	0.93
V(0.25,0.25)	2500	6.37	43.28	43.2	0.18
V(0.75,0.75)	500	6.5	6.6	6.8	2.9
V(0.5,0.5)	5000	5.46	24.72	25	1.12
V(0.75,0.25)	2000	6.44	42.96	43.2	0.56
V(0.75,0.25)	10000	6.42	43.18	43.2	0.05
V(0.5,0.75)	5000	6.93	9.52	9.54	0.21

Comparing these results (Floating random walk)with the existed results of Fixed Random Walk method as in Table 4.1, it is noticeable that the results was good and more stable as different values (N) (number used random values) are taken.

To solve the Laplace equation in the square 0< x, y $<\boldsymbol{\pi}$ subject to the Dirichlet condition u(x,0) = 1984, u(x,$\boldsymbol{\pi}$) = u(0,y) = u($\boldsymbol{\pi}$,y)= 0

We know this problem involves homogeneous boundary conditions on the two boundaries parallel to the y axis, and a single non homogeneous condition on the boundary y=0, as in Figure 4.3

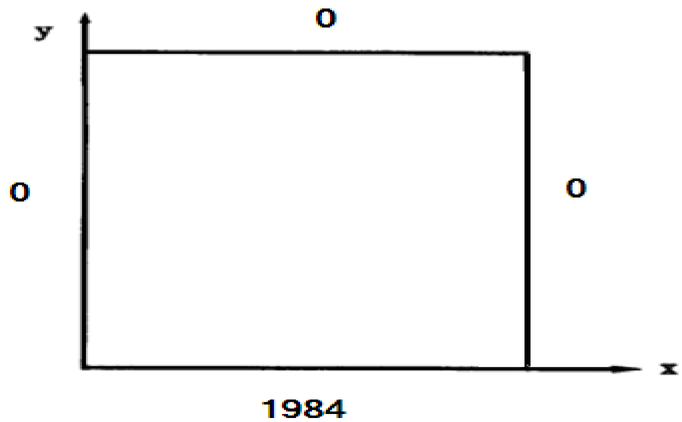

Figure 4.3.the Laplace equation in the square 0< x, y $<\boldsymbol{\pi}$

The analytic solution to the problem is given formally by [8]:

$$u(x, y) = 7936 \sum_{n=1}^{\infty} \frac{\sin((2n-1)x)\sinh((2n-1)(\pi - y))}{(2n-1)\sinh((2n-1)\pi)},$$

We use Fixed method through Mathcad program (See appendix 4.C) with a small difference over the program that used to solve the square sheet using Fixed method in the previous problem(with Fixed Random Walk).

Table 4.3 Comparing Fixed Random Method to Exact Solution

V(x,y)	N	mean steps(m)	Fixed random walk	Exact solution	Errors(%)
V(0.25,0.50)	3000	600	568.31	558.65	1.73
V(0.5,0.5)	1500	800	943.22	937.06	0.66
V(0.75,0.75)	5000	1700	857.09	868.90	0.47
V(0.25,0.25)	3000	745	982.92	978.26	0.48

In the following problem (as shown in Figure 4.4), We want to solve Laplace's equation $\nabla^2 u = 0$ in the rectangle R=[(x , y):0≤ x ≤ 4 , 0≤ y ≤ 4], where u(x,y) denotes the temperature at the point (x,y) and the boundary values are

u(x,0) = 20 and u(x,4) =180 for 0< x <4

and

u(0,y) = 80 and u(4,y) =0 for 0<y <4

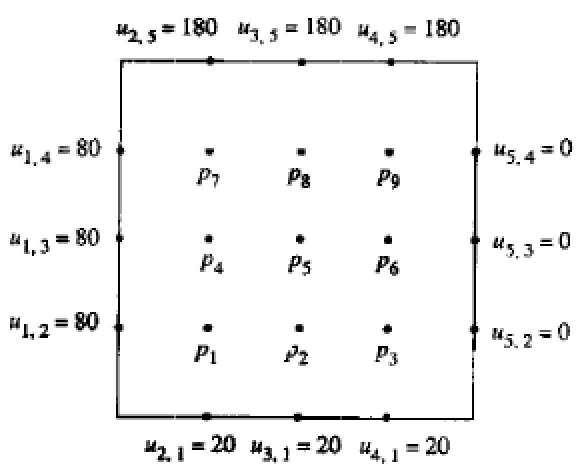

Figure 4.4 The 5× 5 grid

This problem has an approximate solution by Gaussian elimination [16] and when we solve it by Fixed Random Walk we obtained the results in Table 4.4

Table 4.4 The results from Fixed random walk and Gaussian elimination

V(x,y)	N	mean steps(m)	Fixed random walk	Gaussian elimination	Error(%)
V(1,1)	3000	1160	54.64	55.7	1.90
V(1,2)	3000	1400	79.3	79.64	0.43
V(1,3)	1000	1150	114	112.9	0.10
V(2,1)	1000	1420	43.30	43.20	0.20
V(2,2)	1000	1860	70.76	70.0	1.09
V(2,3)	1000	1550	110.20	111.80	1.40
V(3,1)	1000	1150	27.20	27.10	0.40
V(3,2)	1000	1440	45.52	45.36	0.35
V(3,3)	1000	1130	84.20	84.30	0.10

By Comparing these results Fixed with Gaussian elimination, we can see that the results are good.

We will try another problem (elliptic partial differential equation)

To determine the steady-state heat distribution in a thin square plate with dimensions 0.5 m by 0.5 m. Two adjacent boundaries are held at 0 ºC, and the heat on the other boundaries increases linearly from 0 ºC at one corner to 100 ºC where the sides meet

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0 \quad 0< x <0.5,\ 0< y <0.5$$

under the boundary conditions

$u(0,y) = 0$, $u(0.5,y) = 200y$, $0 \le y \le 2$

$u(x,0) = 0$, $u(x,0.5) = 200x$, $0 \le x \le 1$ as in the Figure 4.6

We choose this equation, which has the numerical solution with the Gauss-Seidel method[10], to verify the results we do obtain and the efficiency of the used methods.

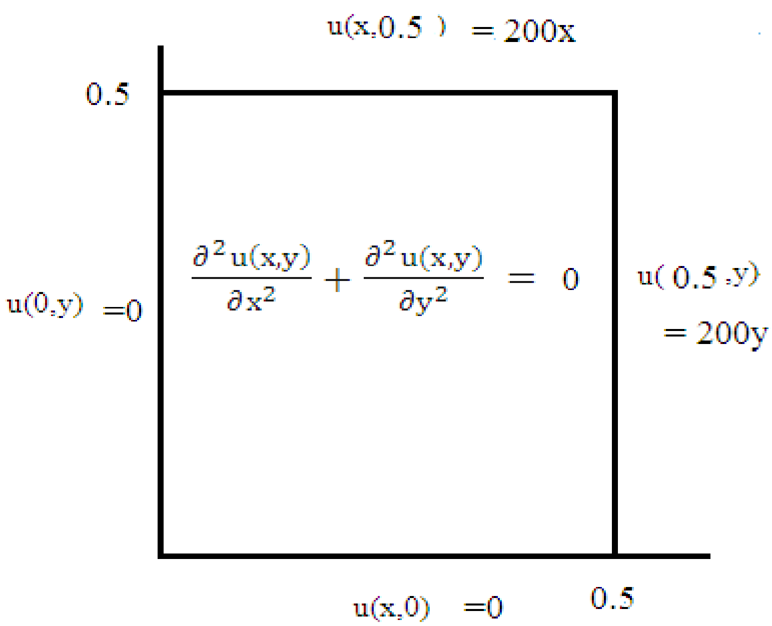

Figure4.5 Laplace’s equation with boundary conditions

When solving this problem using Floating method and MathCAD program (as in Appendix 4.E) the obtained results are good. As shown in Table 4.5, we put different values (N), almost in each case, to calculate these results.

Table 4.5 The results Floating RW with Gauss-Seidel method for Laplace's equation

V(x,y)	N	Floating random walk	Gauss-Seidel method	exact solution	error %
V(1/8,1/8)	1400	6.26	6.25	6.25	-0.16
V(1/8,2/8)	400	12.57	12.5	12.5	-0.56
V(1/8,3/8)	1400	18.72	18.75	18.75	0.16
V(2/8,1/8)	1400	12.55	12.5	12.5	-0.40
V(2/8,2/8)	1400	25	25	25	0.00
V(2/8,3/8)	1700	37.5	37.5	37.5	0.00
V(3/8,1/8)	1700	18.75	18.75	18.75	0.00
V(3/8,2/8)	1400	37.4	37.5	37.5	0.27
V(3/8,3/8)	1400	56.8	56.25	56.25	-0.98

In the following problem as shown in Figure 4.7 and which represents Poisson's equation.

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 1 \qquad 0< x <0,\ 0<y<1$$

Subject to the boundary condition u =0 on all four sides of the square

$u(0,y) = 0,\ u(1,y) = 0,\ 0 \le y \le 1$

$u(x,0) = 0$, $u(x,1) = 0$, $0 \leq x \leq 1$

This Problem has an exact solution and approximate solution by the finite method at nine points [13], and When solving this problem using Floating method and Math-cad program (as in Appendix 4.F) the obtained results are good. As shown in Table- 4.6.

Table 4.6. Comparing Floating random walk results with exact solution and Finite method

V(x,y)	n	Floating random walk	Exact solution	Finite method
V(1/4,1/4)	700	-0.045	-0.045	-0.043
V(1/4,2/4)	900	-0.056	-0.057	-0.055
V(1/4,3/4)	500	-0.046	-0.045	-0.043
V(2/4,1/4)	700	-0.057	-0.057	-0.055
V(2/4,2/4)	500	-0.73	-0.073	-0.07
V(2/4,3/4)	700	-0.57	-0.057	-0.055
V(3/4,1/4)	350	-0.45	-0.045	-0.043
V(3/4,2/4)	350	-0.56	-0.057	-0.055
V(3/4,3/4)	350	-0.045	-0.045	-0.043

In the next problem we have Poisson's equation

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 4 \quad 0 < x < 1,\ 0 < y < 2$$

Subject to the boundary conditions

$u(0,y) = y^2$, $u(1,y) = (y-1)^2$, $0 \leq y \leq 2$

$u(x,0) = x^2$, $u(x,2) = (x-2)^2$, $0 \leq x \leq 1$ as in Figure 4.6

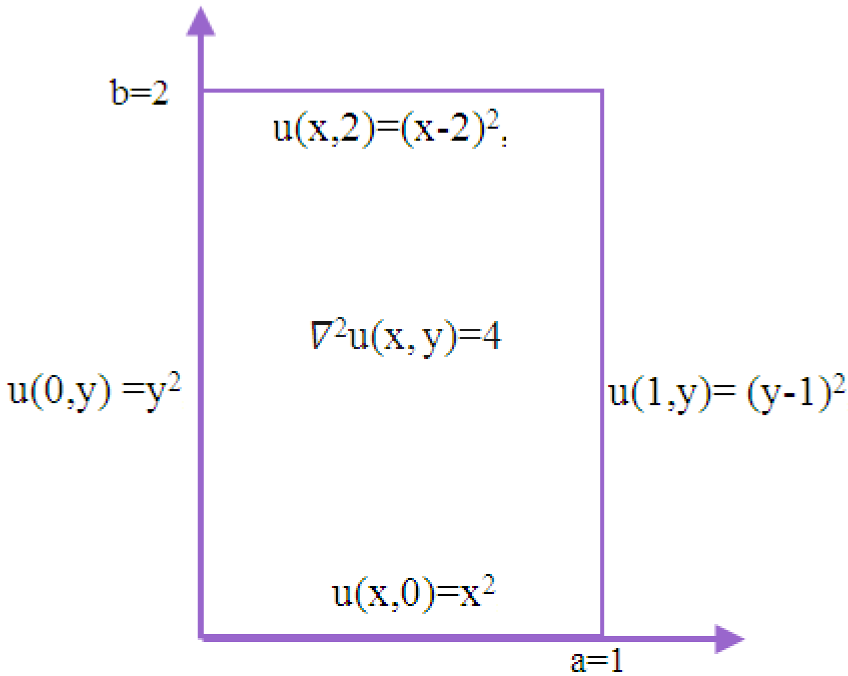

Figure 4.6 for the problem $\nabla^2 u(x, y)=4$ and its B.Cs

This Problem has exact solution $u(x,y) = (x - y)^2$, and When we solve this problem using Floating method and Mathcad program (as in Appendix 4.G) the obtained results are good. As shown in Table 4.7.

Table 4.7 the results for Floating random walk and exact solution

	x	Y	N	Floating random walk	exact solution	error(%)
u(x,y)	0.25	0.5	700	0.064	0.063	-2.40
u(x,y)	0.5	0.75	900	0.060	0.063	4.00
u(x,y)	0.75	1	500	0.060	0.063	4.00
u(x,y)	0.75	1.5	700	0.545	0.563	3.11
u(x,y)	0.25	1.5	700	1.575	1.563	-0.80
u(x,y)	0.5	1.75	400	1.496	1.563	4.26

u(x,y)	0.5	0.25	700	0.064	0.063	-2.40

In the other problem for Poisson's Equation

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = (x^2 + y^2)e^{xy}, \; 0< x <2, \; 0< y <1$$

Subject to the boundary conditions

$u(0,y) = 1$, $u(2,y) = e^{2y}$, $0 \leq y \leq 1$

$u(x,0)=1$, $u(x,1)=e^x$, $0 \leq x \leq 2$ as in Figure 4.7

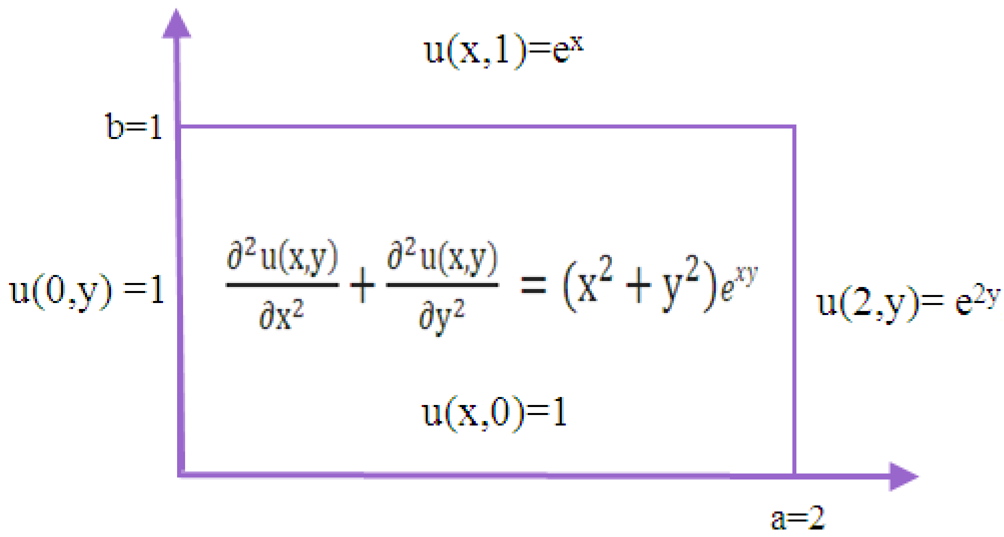

Figure 4.7.Poisson's Equation

we solve this problem using Floating random walk method and Mathcad program (as in Appendix 4. H), and we compare the results with exact solution

$u(x, y) = e^{xy}$. As in the Table 4.8 The results is good.

Table 4.8 the results for Floating random walk and exact solution

	x	Y	N	Floating random walk	exact solution	error(%)
u(x,y)	0.25	0.25	700	1.060	1.064	0.42
u(x,y)	0.25	0.50	700	1.110	1.133	2.04
u(x,y)	0.25	0.75	700	1.205	1.206	0.10
u(x,y)	0.75	0.25	700	1.180	1.206	2.17
u(x,y)	0.75	0.50	700	1.450	1.455	0.34
u(x,y)	0.75	0.75	700	1.750	1.755	0.29
u(x,y)	1.00	0.25	700	1.280	1.284	0.31
u(x,y)	1.00	0.50	700	1.560	1.649	5.38
u(x,y)	1.00	0.75	700	2.070	2.117	2.22
u(x,y)	1.25	0.25	700	1.260	1.367	7.82

u(x,y)	1.50	0.50	700	1.980	2.117	6.47
u(x,y)	1.75	0.75	700	3.660	3.715	1.49

In the following problem

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = \frac{x}{y} + \frac{y}{x} \qquad 1< x <2,\ 1< y <2$$

Subject to the Dirichlet boundary conditions

$u(1,y) = y\ln(y)$, $u(2,y) = 2y\ln(2y)$, $1 \le y \le 2$

$u(x,1) = x\ln(x)$, $u(x,2) = x\ln(4x^2)$, $1 \le x \le 2$ as in Figure 4.9

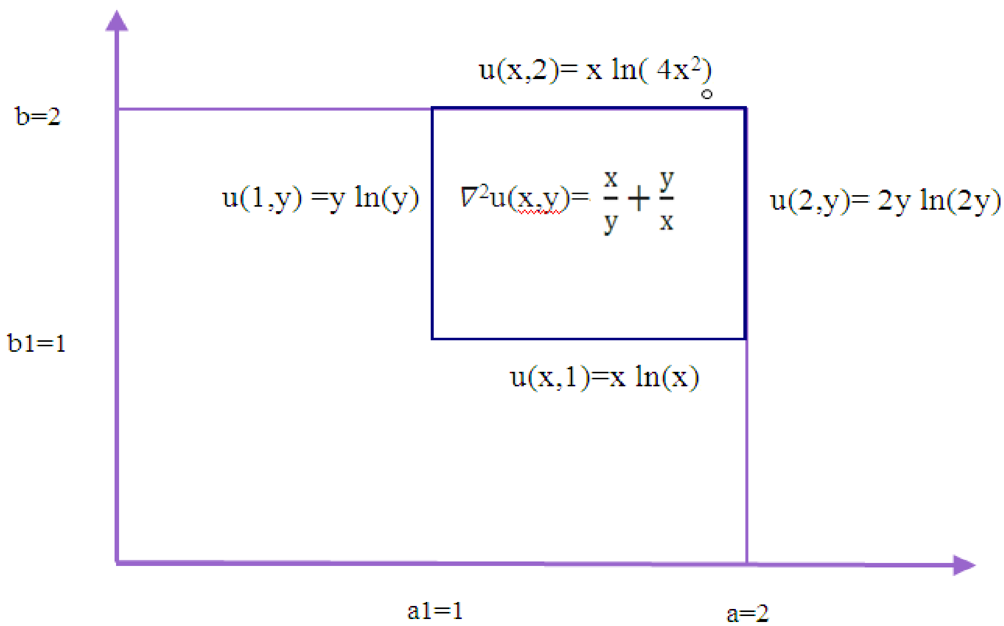

Figure 4.9.Poisson's Equation

We use Floating random walk to solve this problem(as in Appendix 4. I) and compare the resuls with exact solution, and the our results are good, as shown in Table 4.9.

Table 4.9 the results for Floating random walk and exact solution

x	y	N	Floating random walk	exact solution	error(%)
1.25	1.25	700	0.69	0.70	1.05
1.25	1.50	700	1.17	1.18	0.73
1.25	1.75	700	1.69	1.71	1.30
1.50	1.25	700	1.17	1.18	0.73
1.50	1.50	700	1.81	1.82	1.07
1.50	1.75	700	2.60	2.53	-2.63
1.75	1.25	700	1.67	1.71	2.29
1.75	1.50	700	2.53	2.53	0.33
1.75	1.75	700	3.47	3.43	-1.24

4.2 Applying Monte Carlo Method in 3 Dimensions (3D)

To Apply Monte Carlo Method in solving Laplace equation or Poisson's Equation in 3 dimensions, we choose a cubical box (sides of length a) consists of five metal plates, which are welded together and grounded (as in Figure 4.12). The top is made of separate sheet of metal, insulated from the others, and held a constant potential V0=1.0 volt. It is required to find the potential at any point inside the box.

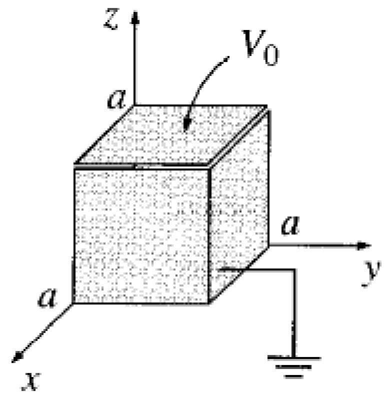

Figure 4.12.a cubical box (sides of length a)

This Problem, when it satisfy Laplace equation and the same boundary conditions, has the solution (EXACT) as in the equation

$$VV(x,y,z) := \frac{16}{\pi^2}\cdot V0\cdot\left[\sum_n\left[\sum_m\left[\left(\frac{1}{m\cdot n}\cdot\sin\left(\frac{n\cdot\pi\cdot x}{a}\right)\cdot\sin\left(\frac{n\cdot\pi\cdot x}{a}\right)\right)\cdot\frac{\sinh(J_{m,n}\cdot z)}{\sinh(J_{m,n}\cdot c)}\right]\right]\right]$$

We used Mathcad program to find the solution at a group of points attached to appendix4.J and it gave the results shown in Table 4.10.

$$V(x,y,z) := \sum_n\sum_m\left[\frac{16}{n\cdot m\cdot\pi^2}\sin(n\cdot x)\cdot\sin(m\cdot y)\cdot\frac{\sinh\left[\sqrt{(n)^2+(m)^2}\cdot z\right]}{\sinh\left[\sqrt{(n)^2+(m)^2}\cdot\pi\right]}\right]$$

Where a= b =c = π

Table 4.10.a cubical box (sides of length a)and V=1v

V(x,y,z)	N	Floating random walk	Exact solution	erorr
V(1.75,1,1.75)	900	0.18	0.1823	1.26%
V(1.75,0.75,1.75)	900	0.16	0.1517	-5.47%
V(1.75,0.5,1.75)	500	0.111	0.1097	-1.19%
V(1.75,0.25,1.75)	500	0.057	0.0578	1.38%
V(1.75,1,1.75)	700	0.18	0.182	1.10%

V(1.5,1,1.75)	700	0.181	0.184	1.63%
V(1.25,1,1.75)	700	0.171	0.1771	3.44%
V(1,1,1.75)	500	0.156	0.16	2.50%
V(0.75,1,1.75)	900	0.14	0.134	-4.48%
V(0.5,1,1.75)	900	0.0975	0.0968	-0.72%
V(0.25,1,1.75)	900	0.051	0.0511	0.20%
V(1.75,1.5,1.75)	900	0.203	0.209	2.87%
V(1.75,1.75,1.75)	700	0.19	0.207	8.21%
V(1.75,1.75,1.25)	500	0.111	0.103	-7.77%
V(1.75,1.75,1.5)	500	0.14	0.147	4.76%
V(1.75,1.75,1)	500	0.069	0.07	1.43%
V(1.75,1,0.75)	900	0.046	0.0465	1.08%
V(1.75,1,0.5)	500	0.027	0.028	3.57%

When this problem is solved by (Floating) method using the Mathcad program, it provided a good results to some extend comparing to (EXACT). And when we solved this problem at 100 volts by using MathCAD program, which is attached to Appendix 4.K, we obtained an acceptable results comparing to both (Floating) and (EXACT) as shown in Table 4.11.

Table 4.11.a cubical box (sides of length a)and V=100v

V(x,y,z)	**N**	**Floating random walk**	**Exact solution**	**erorr**
V(1.75,1,1.75)	750	18.07	18.225	0.85%
V(1.75,0.75,1.75)	900	15.429	15.17	-1.71%
V(1.75,0.5,1.75)	900	10.429	10.97	4.93%

V(1.75,0.25,1.75)	900	5.43	5.78	6.06%
V(1.75,2,1.75)	900	19.7	19.44	-1.34%
V(1.5,1,1.75)	900	18.286	18.422	0.74%
V(1.25,1,1.75)	900	17.708	17.07	-3.74%
V(1,1,1.75)	700	16.285	16.044	-1.50%
V(0.75,1,1.75)	700	14	13.373	-4.69%
V(0.5,1,1.75)	750	9.429	9.68	2.59%

Noting that, the results were good when we solved this problem by (Floating) and (exact) methods. And we put the structure with more generality in Mathcad program to solve the problem with (Floating) method, by considering the dimension of the shape (it might be a parallelograms), the boundary conditions, and the source (F(x,y,z)). We begin with the function F(x,y,z) =0, the problem will be changed to Laplace equation. And if we change the value of the function to a constant (excluding 0), then its solution become the solution of Poisson Equation. And if the function changed to a complex function with known integral limit values, then we can solve it by using a Mathcad program as it is calculated in the specified section of Chapter 1 for the multiple integrals with Monte Carlo.

In the next problem we have Laplace's equation in three dimensions, with F(x,y,z)=0,

And Dirichlet B.Cs $u(x,y,z)=\sin(\pi y)\sin(\pi z)$, at x=0

$u(x,y,z)=2\sin(\pi y)\sin(\pi z)$, at x=1

$u(x,0,z)=0$, $u(x,1,z)=0$, $u(x,y,0)=0$, $u(x,y,1)=0$.

This problem has an exact solution

$$u_2(x,y,z) := \frac{\sin(\pi \cdot y)\cdot\sin(\pi \cdot z)}{\sinh(\pi\cdot\sqrt{2})}\cdot\left[2\cdot\sinh(\pi\cdot\sqrt{2}\cdot x) + \sinh\left[\pi\cdot\sqrt{2}\cdot(1-x)\right]\right]$$

When we solve the problem using Floating random walk we obtain a good results, as in Table4.12,with a program as shown in Appendix4.L.

Table4.12.Laplace's equation

V(x,y,z)	**N**	**Floating random walk**	**Exact solution**	**erorr**
V(0.4,0.4,0.7)	450	0.226	0.233	3.00%
V(0.5,0.75,.5)	1450	0.216	0.227	4.85%
V(0.5,0.6,.5)	450	0.299	0.306	2.29%
V(0.25,0.5,0.25)	700	0.265	0.278	4.68%
V(0.5,0.5,0.4)	700	0.299	0.306	2.29%
V(0.35,0.35,0.75)	550	0.2	0.205	2.44%
V(0.25,0.5,0.75)	720	0.289	0.278	-3.96%
V(0.3,0.6,0.75)	450	0.225	0.233	3.43%
V(0.5,0.25,0.75)	550	0.167	0.161	-3.73%
V(0.5,0.5,0.8)	470	0.178	0.189	5.82%
V(0.8,0.5,0.8)	750	0.497	0.4971	0.02%

Chapter: 5

Conclusion& Future Work

5.1 Conclusion

In this thesis, Monte Carlo method with its components have been used to solve some of Partial Differential Equations such as Laplace and Poisson equations with their applications in electromagnetic areas. First, The method is used to solve these equations in 2 Dimensions with various problems, which are originally solved in other analytical and numerical methods. Then we added to our research the usage of the 3 dimensions and applied that; through MathCAD program (14, and 15), on Laplace's Equation as a special case of Poisson's equation. Hence, we could compare the obtained results by using Monte Carlo Method with other results of other Numerical and analytical methods. the comparison shows that all results are very close to each other, which prove that our method is effective. Noticing that Monte Carlo method is one of the new and effective methods, and it can introduce a solutions to problems which is difficult to solve through other Numerical and Classical methods, especially when there are many variables in the Problem in hand

5.2 Future Work

Monte Carlo Method is one of the strong numerical methods in solving Partial differential equations, therefore we recommended to generalize this study to contain other geometric designs such as spherical and cylindrical shapes, and contain under Dirichlet boundary condition, and also solving Poisson's equation under Neumann boundary condition.

References

[1] J. M. Hammersley, D .C. Handscomb, “Monte Carlo Methods”,4th edition, John Wiley &Sons Inc, New York, 1975, p[21,52]

[2] William L. Dunn, J. Kenneth Shultis, Exploring Monte Carlo Methods, an imprint of Elsevier, New York, 2010, p[2-7,35,39]

[3]. Ilya M. Sobol, A primer for the Monte Carlo method, CRC press,Inc, Florida,1994, p [20,66]

[4].Francis Scheid, Schaum's Outline of Theory and Problems of Numerical Analysis, McGraw-Hill book com, New York, 1968, p[401]

[5]. M. N. 0. Sadiku, Numerical Techniques in Electromagnetics, CRC Press, Boca Raton, 1992.

[6]. Brent Maxfield , Engineering With Mathcad, an imprint of Elsevier, New York, 2006

[7]. Joseph A. Edminister, Schaum's outline of theory and problems of electromagnetic, 2nd edition, McGraw-Hill book com, New York, 1993

[8]. Yehuda Pinchover and Jacob Rubinstein, An introduction to partial differential equations, Cambridge University Press, Cambridge, 2005,p[191,192]

[9]. Stanley J. Farlow, Partial Differential Equations, Dover publications, Inc, New York, 1982.

[10]. Richard L. Burden, J. Douglas Faires, Numerical Analysis,9th edition, Richard Stratton, New York, 2011.

[11]. Sergey E. Lyshevski , Engineering and Scientific Computations Using MATLAB, John Wiley &Sons Inc, New Jersey, 2003.

[12]. Ivan T. Dimov, Monte Carlo Methods For Applied Scientists, World Scientific Publishing Co, New Jersey,2008, p[28-30]

[13]. Nakhle H. Asmar, Partial Differential Equations with Fourier series and boundary value problems, 2nd edition, Pearson prentice Hall, New Jersey, 2004.

[14]. R. W. Hamming , Numerical Methods for Scientists and Engineers, Mcgraw-Hill book com, Inc, New York , 1962.

[15]. George B. Arfken, Hans J. Weber, Frank E. Harris, Mathematical methods for physics, 7th edition, Elsevier Inc, New York, 2013.

[16]. John. H. Mathews, Kurtis D. Fink, Numerical Methods Using MATLAB,3rd edition, Prentice Hall, 1998,TX,USA, p[540-541]

[17].George Levy, Computational Finance using C and C#, 1st edition, Elsevier Ltd, London, 2008, p[183]
[18]. Peter Jackel, Monte Carlo methods in Finance, John Wiley and Sons Inc, London, 2002, p[19-20]
[19]. Philip M. Morse, Herman Feshbach, Methods of theoretical physics, McGraw-Hill Book Company, Inc, New York,1953 p[197]
[20]. M. N. O. Sadiku, Monte Carlo methods for electromagnetics, 2009, p[554-557]
[21]. Tao Pang, Introduction to computational physics, 2ndedition,Cambridge University Press, New York, 2006, p[285-287]
[22].F. Reif, Fundamentals of Statistical And Thermal Physics,Waveland Press, Inc, 1965, p[16]
[23]. J. D. Jackson, classical electrodynamics 2ndedition, John Wiley & Sons, Inc, New York, 1962.
[24].Harvey Gould, Jan Tobochnik, Wolfgang Christian, An Introduction to Computer Simulation Methods: Applications to Physical Systems, 3rdedition, 2006, [Ch11]
[25]. Malvin H. Kalos, Paula A. Whitlock, Monte Carlo Methods,2ndedition, WILEY-VCH Verlag GmbH & Co.
KGaA, Germany, 2008, p[28-30]

[26].M. N. O. Sadiku, R. C. Garcia, "Monte Carlo Floating Random Walk Solution of Poisson's Equation," Southeastcon, Proceedings, IEEE , 1993, Page(s): [1-4].
[27]. M. N. 0. Sadiku, "Monte Carlo methods in an introductory electromagnetic course," IEEE Trans. on Educ., vol. 33, No.1, pp. 73- 80, 1990.

Appendixes

Appendix 1.A

$m := 1000000$

$i := 0..m$ $\quad fx_i := rnd(1) \quad fy_i := rnd(1) \quad fz_i := rnd(1) \quad fu_i := rnd(1) \quad fw_i := rnd(1)$

area :=

```
sum ← 0
for i ∈ 0..m
    Dx_i ← 1  if  0 ≤ fx_i ≤ 0.5
    Dx_i ← 0  otherwise
    Dy_i ← 1  if  0 ≤ fy_i ≤ 0.7
    Dy_i ← 0  otherwise
    Dz_i ← 1  if  0 ≤ fz_i ≤ 2·fx_i
    Dz_i ← 0  otherwise
    Du_i ← 1  if  0 ≤ fu_i ≤ 2·fy_i
    Du_i ← 0  otherwise
    Dw_i ← 1  if  0 ≤ fw_i ≤ 2·fx_i
    Dw_i ← 0  otherwise
    sum ← sum + [((fx_i)^2 + (fy_i)^3 − fz_i) / ((fw_i)^2 + 1)]·Dw_i·Du_i·Dz_i·Dy_i·Dx_i
sum
```

$$sum \leftarrow sum + \left[\frac{(fx_i)^2 + (fy_i)^3 - fz_i}{(fw_i)^2 + 1}\right]\cdot Dw_i\cdot Du_i\cdot Dz_i\cdot Dy_i\cdot Dx_i$$

Appendix (1.B)

$$A22(N) := \left| \begin{array}{l} sum \leftarrow 0.0 \\ \text{for } i \in 1..N \\ \quad \left| \begin{array}{l} x \leftarrow 0.7 \cdot rnd(1.0) \\ y \leftarrow 0.8 \cdot rnd(1.0) \\ z \leftarrow 0.9 \cdot rnd(1.0) \\ u \leftarrow rnd(1.0) \\ w \leftarrow 1.1 \cdot rnd(1.0) \\ sum \leftarrow sum + \sqrt{6 - x^2 - y^2 - z^2 - u^2 - w^2} \cdot 0.7 \cdot 0.8 \cdot 0.9 \cdot 1.1 \end{array} \right. \\ \dfrac{sum}{N} \end{array} \right.$$

Appendix (1.c)

```
F(m, n) := | for  i ∈ 0.. m
           |   d_i ← 0
           | for  i ∈ 0.. n
           |   | r ← rnd(1)
           |   | x ← −ln(1 − r)
           |   | i ← floor(x)
           |   | d_i ← d_i + 1  if  i ≤ n
           | d
```

$t := F(20, 20000)$

$r := 0..20$

Appendix (1.d)

$\Delta\theta := \frac{\pi}{20}$ $\quad$ $i := 1..20$

```
dd(n,m) := | dtetha ← π/n
           | for i ∈ 0..n
           |   y_i ← 0
           | for j ∈ 1..m
           |   | r ← rnd(1.0)
           |   | tetha ← acos(1 − 2·r)
           |   | k ← floor(tetha / dtetha)
           |   | y_k ← y_k + 1
           | y
```

Appendix (2.A)

In This appendix we will discuss the analytic solution of Poisson's equation Using Green's function technique in two dimensions. Consider the non-homogenous Poisson's Equation in two dimensions which takes the following form: $\nabla^2 u(x,y) = f(x,y)$
(1)

With interval given by:

$$0 \leq \mathrm{x} \leq \mathrm{a} \text{ ,and } \quad 0 \leq \mathrm{y} \leq \mathrm{b} \qquad (2\ )$$

Then the Green's function for Poisson's equation Eq(1) can be written as:

$$\nabla^2 G(x,y;x0,y0) = -\,\delta(\mathrm{x}-\mathrm{x}0)\delta(y\ -y0) \qquad (3)$$

Where $G(x,y;x0,y0)$ is define as the Green's function, and $\delta(\mathrm{x}-\mathrm{x}0)$ is delta function.

The boundary condition. for Eq(1) is given by:

$$u(0,b) = 0, \;\; u(a,0) = 0, \quad u(x,b) = 0, \;\; \text{and } u(a,y) = 0 \qquad (4)$$

Then the solution for Eq(1) is given by:

$$u(x0,y0) = \int_0^a \int_0^b f(x,y) G(x,y;x0,y0)\, dxdy \qquad (5)$$

To construct the Green's function for Poisson's equation, we can use the eigenfunction and eigenvalue of the Laplace equation with same boundary condition Eq(4).

Where the eigenfunction of Laplace equation is written as:

$$u(x,y) = \sqrt{\frac{2}{a}}\, sin\left(\frac{n\pi x}{a}\right).\sqrt{\frac{2}{b}}\, sin(m\pi y/b) \qquad (6)$$

And the eigenvalue takes the following form:

$$\lambda_{\mathrm{nm}} = \left(\frac{n\pi}{a}\right)^2 + \left(\frac{m\pi}{b}\right)^2 \qquad (7)$$

Then the Green’s function is written as:

$$G(x,y\ \ |,x0\ ,y0) = 4\,\frac{ab}{\pi^2}\sum\nolimits_{nm} \frac{1}{(nb)^2+(ma)^2}\times$$

$$sin\left(\frac{n\pi x}{a}\right)\ sin\left(\frac{m\pi y}{b}\right) sin\left(\frac{n\pi x0}{a}\right)\ sin\left(\frac{m\pi y0}{b}\right) \qquad (8)$$

The solution of Poisson's equation takes this form

$$u(x0, y0) = \int_0^a \int_0^b f(x, y) G(x, y \mid, x0 , y0) dxdy \tag{9}$$

Appendix 3. A

Taylor series

Taylor series for a function f(x) is written as:

$$\mathrm{f}(x+h) = f(x) + \frac{hf'(x)}{1!} + \frac{h^2 f''(x)}{2!} + \frac{h^3 f'''(x)}{3!} + \cdots \qquad (1)$$

Provided all derivatives of f(x) exist and are continuous between x and x+h

from Eq(1) we can find the first order derivative as:

$$f'(x) = \frac{\mathrm{f}(x+h) - \mathrm{f(x)}}{h} + \mathrm{O(h)} \qquad (2)$$

Also we can write Taylor series for a function f(x) as:

$$\mathrm{f}(x-h) = f(x) - \frac{hf'(x)}{1!} + \frac{h^2 f''(x)}{2!} - \frac{h^3 f'''(x)}{3!} + \cdots \qquad (3)$$

from Eq(3) we can find the first order derivative as:

$$f'(x) = \frac{\mathrm{f}(x) - \mathrm{f(x-h)}}{h} + \mathrm{O(h^2)} \qquad (4)$$

Adding Eq(1)and Eq(2) we get the following:

$$\mathrm{f}(x+h) + \mathrm{f}(x-h) = 2f(x) + \frac{h^2 f''(x)}{2!} \times 2 \qquad (5)$$

Then the second order derivative is taken this form:

$$f''(x) = \frac{\mathrm{f}(x+h) - 2f(x) + \mathrm{f}(x-h)}{h^2} + \mathrm{O(h^3)} \qquad (6)$$

For a function f(x,y) of two independent variables, it is convenient to analyse the rate of change of f(x,y) with respect to x and y separately. This involves the use of a

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{\mathrm{f}(x+h,y) - \mathrm{f}(x,y)}{h} \qquad (7)$$

Similarly we obtain $\frac{\partial \mathrm{f}}{\partial \mathrm{y}}$ on differentiating f(x,y) with respect to y, treating x as a constant.

$$\frac{\partial \mathrm{f}}{\partial \mathrm{y}} = \lim_{\mathrm{h} \to 0} \frac{\mathrm{f(x,y+h) - f(x,y)}}{\mathrm{h}} \qquad (8)$$

Then the second order derivative in two variables is taken this form:

$$\frac{\partial^2 \mathrm{f(x,y)}}{\partial^2 \mathrm{x}} \approx \frac{\mathrm{f(x+h,y) - 2f(x,y) + f(x-h,y)}}{\mathrm{h^2}} \qquad (9)$$

and

$$\frac{\partial^2 f(x,y)}{\partial^2 y} \approx \frac{f(x,y+h)-2f(x,y)+f(x,y-h)}{h^2} \quad (10)$$

The Laplace's equation in two dimensions can be written as:

$$\frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y} = 0 \quad (11)$$

Using Eq(9) and Eq(10) we can write Eq(11)as:

$$\frac{f(x+h,y)-2f(x,y)+f(x-h,y)}{h^2} + \frac{f(x,y+h)-2f(x,y)+f(x,y-h)}{h^2} = 0 \quad (12)$$

Which can be more simplified as:

$$-4f(x,y) - f(x+h,y) - f(x-h,y) - (x,y+h) - f(x,y-h) = 0 \quad (13)$$

Then we rewrite Eq(13) as:

$$f(x,y) = \frac{1}{4}f(x+h,y) + \frac{1}{4}f(x-h,y) + \frac{1}{4}(x,y+h) + \frac{1}{4}f(x,y-h) \quad (14)$$

Appendix 4.A

$V1 := 100.0 \quad V2 := 0.0 \quad V3 := 0.0 \quad V4 := 0.0$

$delt := 0.05 \quad a := 1 \quad b := 1 \quad IM := \frac{a}{delt} \quad JM := \frac{b}{delt}$

```
LapSol(x0, y0, N) := | I0 ← floor(x0/delt)
                     | J0 ← floor(y0/delt)
                     | sum ← 0
                     | ns ← 0
                     | jj ← 0
                     | m1 ← 0
                     | m2 ← 0
                     | m3 ← 0
                     | m4 ← 0
                     | for k ∈ 1..N
                     |    | I ← I0
                     |    | J ← J0
                     |    | M ← 1
                     |    | while M
                     |    |    | r ← rnd(1)
                     |    |    | ns ← ns + 1
                     |    |    | d ← floor(4·r)
                     |    |    | if d = 0
                     |    |    |    | I ← I + 1
                     |    |    |    | if I = IM
                     |    |    |    |    | m2 ← m2 + 1
                     |    |    |    |    | sum ← sum + V2
                     |    |    |    |    | M ← 0
                     |    |    | if d = 1
                     |    |    |    | I ← I − 1
                     |    |    |    | if I = 0
                     |    |    |    |    | m4 ← m4 + 1
                     |    |    |    |    | sum ← sum + V4
                     |    |    |    |    | M ← 0
                     |    |    | if d = 2
                     |    |    |    | J ← J + 1
                     |    |    |    | if J = JM
                     |    |    |    |    | m3 ← m3 + 1
                     |    |    |    |    | sum ← sum + V3
                     |    |    |    |    | M ← 0
                     |    |    | if d = 3
                     |    |    |    | J ← J − 1
                     |    |    |    | if J = 0
                     |    |    |    |    | m1 ← m1 + 1
                     |    |    |    |    | sum ← sum + V1
                     |    |    |    |    | M ← 0
                     |    | if mod(k, 250) = 0
                     |    |    | z_jj ← sum/k
                     |    |    | z_jj+1 ← ns/k
```

Appendix 4.B

```
Ex1float(x0, y0, N) := | sum ← 0.0
                       | jj ← 0
                       | ms ← 0
                       | TOL ← 0.005
                       | m ← 0
                       | for k ∈ 1.. N
                       |    | x ← x0
                       |    | y ← y0
                       |    | M ← 1
                       |    | while M
                       |    |    | u ← rnd(1)
                       |    |    | phi ← 2.0·pi·u
                       |    |    | ms ← ms + 1
                       |    |    | r1 ← min[(a − x), x]
                       |    |    | r2 ← min[(b − y), y]
                       |    |    | r ← min(r1, r2)
                       |    |    | x ← x + r·cos(phi)
                       |    |    | y ← y + r·sin(phi)
                       |    |    | if y ≥ (b − TOL)
                       |    |    |    | sum ← sum + v3
                       |    |    |    | M ← 0
                       |    |    | if y ≤ (0 + TOL)
                       |    |    |    | sum ← sum + v1
                       |    |    |    | M ← 0
                       |    |    | if x ≥ (a − TOL)
                       |    |    |    | sum ← sum + v2
                       |    |    |    | M ← 0
                       |    |    | if x ≤ (0 + TOL)
                       |    |    |    | sum ← sum + v4
                       |    |    |    | M ← 0
                       |    | if mod(k, 250) = 0
                       |    |    | z_jj ← sum / k
                       |    |    | z_(jj+1) ← ms / k
                       |    |    | jj ← jj + 2
                       | z
```

Appendix 4.c

$V1 := 1984.0$ $V2 := 0.0$ $V3 := 0.0$ $V4 := 0.0$

$delt := 0.05$ $a := \pi$ $b := \pi$

$IM := \frac{a}{delt}$ $JM := \frac{b}{delt}$

```
LapSol(x0, y0, N) := | I0 ← floor(x0/delt)
                     | J0 ← floor(y0/delt)
                     | sum ← 0
                     | ns ← 0
                     | jj ← 0
                     | m1 ← 0
                     | m2 ← 0
                     | m3 ← 0
                     | m4 ← 0
                     | for k ∈ 1.. N
                     |   | I ← I0
                     |   | J ← J0
                     |   | M ← 1
                     |   | while M
                     |   |   | r ← rnd(1)
                     |   |   | ns ← ns + 1
                     |   |   | d ← floor(4·r)
                     |   |   | if d = 0
                     |   |   |   | I ← I + 1
                     |   |   |   | if I = IM
                     |   |   |   |   | m2 ← m2 + 1
                     |   |   |   |   | sum ← sum + V2
                     |   |   |   |   | M ← 0
                     |   |   | if d = 1
                     |   |   |   | I ← I − 1
                     |   |   |   | if I = 0
                     |   |   |   |   | m4 ← m4 + 1
                     |   |   |   |   | sum ← sum + V4
                     |   |   |   |   | M ← 0
                     |   |   | if d = 2
                     |   |   |   | J ← J + 1
                     |   |   |   | if J = JM
                     |   |   |   |   | m3 ← m3 + 1
                     |   |   |   |   | sum ← sum + V3
                     |   |   |   |   | M ← 0
                     |   |   | if d = 3
                     |   |   |   | J ← J − 1
                     |   |   |   | if J = 0
                     |   |   |   |   | m1 ← m1 + 1
                     |   |   |   |   | sum ← sum + V1
                     |   |   |   |   | M ← 0
                     |   | if mod(k, 250) = 0
                     |   |   | z_jj ← sum/k
                     |   |   | z_(jj+1) ← ns/k
                     |   |   | jj ← jj + 2
                     | z
```

Appendix 4.D

$V1 := 20.0 \qquad V2 := 0.0 \qquad V3 := 180.0 \qquad V4 := 80.0$

$delt := 0.05 \qquad a := 4.0 \qquad b := 4.0 \qquad IM := \frac{a}{delt} \qquad JM := \frac{b}{delt}$

```
LapSol(x0, y0, N) :=
  I0 ← floor(x0/delt)
  J0 ← floor(y0/delt)
  sum ← 0
  ns ← 0
  jj ← 0
  m1 ← 0
  m2 ← 0
  m3 ← 0
  m4 ← 0
  for k ∈ 1..N
    I ← I0
    J ← J0
    M ← 1
    while M
      r ← rnd(1)
      ns ← ns + 1
      d ← floor(4·r)
      if d = 0
        I ← I + 1
        if I = IM
          m2 ← m2 + 1
          sum ← sum + V2
          M ← 0
      if d = 1
        I ← I − 1
        if I = 0
          m4 ← m4 + 1
          sum ← sum + V4
          M ← 0
      if d = 2
        J ← J + 1
        if J = JM
          m3 ← m3 + 1
          sum ← sum + V3
          M ← 0
      if d = 3
        J ← J − 1
        if J = 0
          m1 ← m1 + 1
          sum ← sum + V1
          M ← 0
    if mod(k, 250) = 0
      z_jj ← sum/k
      z_jj+1 ← ns/k
```

Appendix 4.E

$n := 200$ $x0 := 0.25$ $y0 := 0.25$ $pi := \pi$ $TOL := 0.005$ $a := 0.5$ $b := 0.5$

```
sds(x0, y0, n) := | m ← 0
                  | sum1 ← 0.0
                  | sum2 ← 0.0
                  | for  k ∈ 1.. n
                  |     | x ← x0
                  |     | y ← y0
                  |     | M ← 1
                  |     | while  M
                  |     |     | u ← rnd(1)
                  |     |     | phi ← 2.0·pi·u
                  |     |     | r1 ← min(x, a − x)
                  |     |     | r2 ← min(y, b − y)
                  |     |     | r ← min(r1, r2)
                  |     |     | x ← x + r·cos(phi)
                  |     |     | y ← y + r·sin(phi)
                  |     |     | m ← m + 1
                  |     |     | if  (x ≤ TOL) ∧ (0 < y < b)
                  |     |     |     | v4 ← 0.0
                  |     |     |     | sum2 ← sum2 + v4
                  |     |     |     | M ← 0
                  |     |     | if  (y ≤ TOL) ∧ (0 < x < a)
                  |     |     |     | v1 ← 0.0
                  |     |     |     | sum2 ← sum2 + v1
                  |     |     |     | M ← 0
                  |     |     | if  [x ≥ (a − TOL)] ∧ (0 < y < b)
                  |     |     |     | v2 ← 200y
                  |     |     |     | sum2 ← sum2 + v2
                  |     |     |     | M ← 0
                  |     |     | if  [0 < x < (a − TOL)] ∧ [y ≥ (b − TOL)]
                  |     |     |     | v3 ← 200x
                  |     |     |     | sum2 ← sum2 + v3
                  |     |     |     | M ← 0
                  |     |     | if  mod(k, 350) = 0
                  |     |     |     | z_jj ← sum1/(4·k) + sum2/k
                  |     |     |     | jj ← jj + 1
```

Appendix 4.F

```
n := 200  pi := π  TOL := 0.005   a := 1   b := 1   v4 := 0.0   v1 := 0   v2 := 0  v3 := 0

sds(x0, y0, n) :=  | m ← 0
                   | sum1 ← 0.0
                   | sum2 ← 0.0
                   | for  k ∈ 1.. n
                   |   | x ← x0
                   |   | y ← y0
                   |   | M ← 1
                   |   | while  M
                   |   |   | u ← rnd(1)
                   |   |   | phi ← 2.0·pi·u
                   |   |   | r1 ← min(x, a − x)
                   |   |   | r2 ← min(y, b − y)
                   |   |   | r ← min(r1, r2)
                   |   |   | x ← x + r·cos(phi)
                   |   |   | y ← y + r·sin(phi)
                   |   |   | m ← m + 1
                   |   |   | sum1 ← sum1 + (−1)·r^2
                   |   |   | if  (x ≤ TOL) ∧ (0 < y < b)
                   |   |   |   | sum2 ← sum2 + v4
                   |   |   |   | M ← 0
                   |   |   | if  (y ≤ TOL) ∧ (0 < x < a)
                   |   |   |   | sum2 ← sum2 + v1
                   |   |   |   | M ← 0
                   |   |   | if  [x ≥ (a − TOL)] ∧ (0 < y < b)
                   |   |   |   | sum2 ← sum2 + v2
                   |   |   |   | M ← 0
                   |   |   | if  [0 < x < (a − TOL)] ∧ [y ≥ (b − TOL)]
                   |   |   |   | sum2 ← sum2 + v3
                   |   |   |   | M ← 0
                   |   |   | if  mod(k, 350) = 0
                   |   |   |   | z_jj ← sum1/(4·k) + sum2/k
                   |   |   |   | jj ← jj + 1
                   | z
```

Appendix 4.G

$TOL := 0.005 \quad a := 1 \quad b := 2 \quad pi := \pi$

```
sds(x0, y0, n) := | m ← 0
                  | sum1 ← 0.0
                  | sum2 ← 0.0
                  | for  k ∈ 1.. n
                  |   | x ← x0
                  |   | y ← y0
                  |   | M ← 1
                  |   | while  M
                  |   |   | u ← rnd(1)
                  |   |   | phi ← 2.0·pi·u
                  |   |   | r1 ← min(x, a − x)
                  |   |   | r2 ← min(y, b − y)
                  |   |   | r ← min(r1, r2)
                  |   |   | x ← x + r·cos(phi)
                  |   |   | y ← y + r·sin(phi)
                  |   |   | m ← m + 1
                  |   |   | sum1 ← sum1 + (−4)·r^2
                  |   |   | if  (x ≤ TOL) ∧ (0 < y < b)
                  |   |   |   | v4 ← y^2
                  |   |   |   | sum2 ← sum2 + v4
                  |   |   |   | M ← 0
                  |   |   | if  (y ≤ TOL) ∧ (0 < x < a)
                  |   |   |   | v1 ← x^2
                  |   |   |   | sum2 ← sum2 + v1
                  |   |   |   | M ← 0
                  |   |   | if  [x ≥ (a − TOL)] ∧ (0 < y < b)
                  |   |   |   | v2 ← (y − 1)^2
                  |   |   |   | sum2 ← sum2 + v2
                  |   |   |   | M ← 0
                  |   |   | if  [0 < x < (a − TOL)] ∧ [y ≥ (b − TOL)]
                  |   |   |   | v3 ← (x − 2)^2
                  |   |   |   | sum2 ← sum2 + v3
                  |   |   |   | M ← 0
                  |   |   | if  mod(k, 350) = 0
                  |   |   |   | z_jj ← sum1/(4·k) + sum2/k
                  |   |   |   | jj ← jj + 1
                  | z
```

Appendix 4.H

$TOL := 0.005 \quad a := 2 \quad b := 1 \quad pi := \pi$

```
sds(x0, y0, n) := | m ← 0
                  | sum1 ← 0.0
                  | sum2 ← 0.0
                  | for  k ∈ 1.. n
                  |   | x ← x0
                  |   | y ← y0
                  |   | M ← 1
                  |   | while  M
                  |   |   | u ← rnd(1)
                  |   |   | phi ← 2.0·pi·u
                  |   |   | r1 ← min(x, a − x)
                  |   |   | r2 ← min(y, b − y)
                  |   |   | r ← min(r1, r2)
                  |   |   | x ← x + r·cos(phi)
                  |   |   | y ← y + r·sin(phi)
                  |   |   | m ← m + 1
                  |   |   | sum1 ← sum1 + [−1(x^2 + y^2)·e^(x·y)]·r^2
                  |   |   | if  (x ≤ TOL) ∧ (0 < y < b)
                  |   |   |   | v4 ← 1
                  |   |   |   | sum2 ← sum2 + v4
                  |   |   |   | M ← 0
                  |   |   | if  (y ≤ TOL) ∧ (0 < x < a)
                  |   |   |   | v1 ← 1
                  |   |   |   | sum2 ← sum2 + v1
                  |   |   |   | M ← 0
                  |   |   | if  [x ≥ (a − TOL)] ∧ (0 < y < b)
                  |   |   |   | v2 ← e^(2·y)
                  |   |   |   | sum2 ← sum2 + v2
                  |   |   |   | M ← 0
                  |   |   | if  [0 < x < (a − TOL)] ∧ [y ≥ (b − TOL)]
                  |   |   |   | v3 ← e^x
                  |   |   |   | sum2 ← sum2 + v3
                  |   |   |   | M ← 0
                  |   |   | if  mod(k, 350) = 0
                  |   |   |   | z_jj ← sum1/(4·k) + sum2/k
                  |   |   |   | jj ← jj + 1
                  | z
```

Appendix 4.I

$a1 := 1 \quad b1 := 1 \quad a := 2 \quad b := 2 \qquad pi := \pi \qquad TOL := 0.05$

```
sds(x0, y0, n) := | m ← 0
                  | sum1 ← 0.0
                  | sum2 ← 0.0
                  | for  k ∈ 1.. n
                  |   | x ← x0
                  |   | y ← y0
                  |   | M ← 1
                  |   | while  M
                  |   |   | u ← rnd(1)
                  |   |   | phi ← 2.0·pi·u
                  |   |   | r1 ← min(x − a1, a − x)
                  |   |   | r2 ← min(y − b1, b − y)
                  |   |   | r ← min(r1, r2)
                  |   |   | x ← x + r·cos(phi)
                  |   |   | y ← y + r·sin(phi)
                  |   |   | m ← m + 1
                  |   |   | sum1 ← sum1 + [−1(x/y + y/x)]·r^2
                  |   |   | if  (x ≤ TOL + a1) ∧ (b1 < y < b)
                  |   |   |   | v4 ← y·ln(y)
                  |   |   |   | sum2 ← sum2 + v4
                  |   |   |   | M ← 0
                  |   |   | if  [(y − b1) ≤ TOL] ∧ (a1 < x < a)
                  |   |   |   | v1 ← x·ln(x)
                  |   |   |   | sum2 ← sum2 + v1
                  |   |   |   | M ← 0
                  |   |   | if  [x ≥ (a − TOL)] ∧ (b1 < y < b)
                  |   |   |   | v2 ← 2·y·ln(2·y)
                  |   |   |   | sum2 ← sum2 + v2
                  |   |   |   | M ← 0
                  |   |   | if  [a1 < x < (a − TOL)] ∧ [y ≥ (b − TOL)]
                  |   |   |   | v3 ← x·ln(4·x^2)
                  |   |   |   | sum2 ← sum2 + v3
                  |   |   |   | M ← 0
                  |   |   | if  mod(k, 350) = 0
                  |   |   |   | z_jj ← sum1/(4·k) + sum2/k
                  |   |   |   | jj ← jj + 1
                  | z
```

Appendix 4.J $nn := 1350 \quad a := \pi \quad b := \pi \quad c := \pi \quad T := 0.007 \quad F(x,y,z) := 0.0 \quad pi := 3.14$

$v1 := 0.0 \quad v2 := 0.0 \quad v3 := 0.0 \quad v4 := 0.0 \quad v5 := 0.0 \quad v6 := 1.0$

```
sds(x, y, z, nn) := | m ← 0
                    | jj ← 0
                    | sum2 ← 0.0
                    | sum0 ← 0.0
                    | for G ∈ 1.. nn
                    |   | x1 ← x
                    |   | y1 ← y
                    |   | z1 ← z
                    |   | M ← 1
                    |   | while M ∧ (m/G < 1000)
                    |   |   | u ← rnd(1)
                    |   |   | u1 ← rnd(1)
                    |   |   | phi ← pi·u
                    |   |   | theta ← acos(1 − 2.u1)
                    |   |   | R ← min(z1, a − z1, y1, b − y1, x1, c − x1)
                    |   |   | x1 ← x1 + R·cos(phi)·sin(theta)
                    |   |   | y1 ← y1 + R·sin(phi)·sin(theta)
                    |   |   | z1 ← z1 + R·cos(theta)
                    |   |   | sum2 ← sum2 + R^2·F(x, y, z)
                    |   |   | m ← m + 1
                    |   |   | if [[T < x1 < (a − T)] ∧ [[T < y1 < (b − T)]] ∧ [z1 > (c − T)]]
                    |   |   |   | sum0 ← sum0 + v6
                    |   |   |   | M ← 0
                    |   |   | if [[T < x1 < (a − T)] ∧ [[T < y1 < (b − T)]] ∧ [z1 < (0 + T)]]
                    |   |   |   | sum0 ← sum0 + v5
                    |   |   |   | M ← 0
                    |   |   | if [[T < x1 < (a − T)] ∧ [T < z1 < (c − T)] ∧ [y1 > (b − T)]]
                    |   |   |   | sum0 ← sum0 + v4
                    |   |   |   | M ← 0
                    |   |   | if [[T < x1 < (a − T)] ∧ [[T < z1 < (c − T)]] ∧ [y1 < (0 + T)]]
                    |   |   |   | sum0 ← sum0 + v3
                    |   |   |   | M ← 0
                    |   |   | if [[T < y1 < (b − T)] ∧ [[T < z1 < (c − T)]] ∧ [x1 < (0 + T)]]
                    |   |   |   | sum0 ← sum0 + v1
                    |   |   |   | M ← 0
                    |   |   | if [[T < y1 < (b − T)] ∧ [[T < z1 < (c − T)]] ∧ [x1 > (a − T)]]
                    |   |   |   | sum0 ← sum0 + v2
                    |   |   |   | M ← 0
                    |   |   | if mod(G, 350) = 0
                    |   |   |   | w_jj ← sum0/G + sum2/(6·G)
                    |   |   |   | jj ← jj + 1
                    | w
```

Appendix 4.K

$nn := 1350 \quad a := \pi \quad b := \pi \quad c := \pi \quad T := 0.007 \quad F(x,y,z) := 0.0 \quad pi := 3.14$

$v1 := 0.0 \quad v2 := 0.0 \quad v3 := 0.0 \quad v4 := 0.0 \quad v5 := 0.0 \quad v6 := 100.0$

```
sds(x,y,z,nn) := | m ← 0
                 | jj ← 0
                 | sum2 ← 0.0
                 | sum0 ← 0.0
                 | for G ∈ 1.. nn
                 |   | x1 ← x
                 |   | y1 ← y
                 |   | z1 ← z
                 |   | M ← 1
                 |   | while M ∧ (m/G < 1000)
                 |   |   | u ← rnd(1)
                 |   |   | u1 ← rnd(1)
                 |   |   | phi ← pi·u
                 |   |   | theta ← acos(1 − 2.u1)
                 |   |   | R ← min(z1, a − z1, y1, b − y1, x1, c − x1)
                 |   |   | x1 ← x1 + R·cos(phi)·sin(theta)
                 |   |   | y1 ← y1 + R·sin(phi)·sin(theta)
                 |   |   | z1 ← z1 + R·cos(theta)
                 |   |   | sum2 ← sum2 + R^2·F(x,y,z)
                 |   |   | m ← m + 1
                 |   |   | if [[T < x1 < (a − T)] ∧ [[T < y1 < (b − T)]] ∧ [z1 > (c − T)]]
                 |   |   |   | sum0 ← sum0 + v6
                 |   |   |   | M ← 0
                 |   |   | if [[T < x1 < (a − T)] ∧ [[T < y1 < (b − T)]] ∧ [z1 < (0 + T)]]
                 |   |   |   | sum0 ← sum0 + v5
                 |   |   |   | M ← 0
                 |   |   | if [[T < x1 < (a − T)] ∧ [T < z1 < (c − T)] ∧ [y1 > (b − T)]]
                 |   |   |   | sum0 ← sum0 + v4
                 |   |   |   | M ← 0
                 |   |   | if [[T < x1 < (a − T)] ∧ [[T < z1 < (c − T)]] ∧ [y1 < (0 + T)]]
                 |   |   |   | sum0 ← sum0 + v3
                 |   |   |   | M ← 0
                 |   |   | if [[T < y1 < (b − T)] ∧ [[T < z1 < (c − T)]] ∧ [x1 < (0 + T)]]
                 |   |   |   | sum0 ← sum0 + v1
                 |   |   |   | M ← 0
                 |   |   | if [[T < y1 < (b − T)] ∧ [[T < z1 < (c − T)]] ∧ [x1 > (a − T)]]
                 |   |   |   | sum0 ← sum0 + v2
                 |   |   |   | M ← 0
                 |   |   | if mod(G,350) = 0
                 |   |   |   | w_jj ← sum0/G + sum2/(6·G)
                 |   |   |   | jj ← jj + 1
                 | w
```

Appendix 4.L

$v6 := 0.0 \quad a := 1 \quad b := 1 \quad c := 1 \quad T := 0.001 \quad v3 := 0.0$

$F(x, y, z) := 0.0 \quad v4 := 0.0 \quad v5 := 0.0 \quad pi := 3.14$

```
sds(x,y,z,nn) := | m ← 0
                 | jj ← 0
                 | sum2 ← 0.0
                 | sum0 ← 0.0
                 | for G ∈ 1.. nn
                 |   | x1 ← x
                 |   | y1 ← y
                 |   | z1 ← z
                 |   | M ← 1
                 |   | v1 ← sin(π·y1)·sin(π·z1)
                 |   | v2 ← 2·sin(π·y1)·sin(π·z1)
                 |   | while M ∧ (m/G < 1000)
                 |   |   | u ← rnd(1)
                 |   |   | u1 ← rnd(1)
                 |   |   | phi ← 2·pi·u
                 |   |   | theta ← acos(1 − 2.u1)
                 |   |   | R ← min(z1, a − z1, y1, b − y1, x1, c − x1)
                 |   |   | x1 ← x1 + R·cos(phi)·sin(theta)
                 |   |   | y1 ← y1 + R·sin(phi)·sin(theta)
                 |   |   | z1 ← z1 + R·cos(theta)
                 |   |   | sum2 ← sum2 + R^2·F(x,y,z)
                 |   |   | m ← m + 1
                 |   |   | if [[T < x1 < (a − T)] ∧ [[T < y1 < (b − T)]] ∧ [z1 > (c − T)]]
                 |   |   |   | sum0 ← sum0 + v6
                 |   |   |   | M ← 0
                 |   |   | if [[T < x1 < (a − T)] ∧ [[T < y1 < (b − T)]] ∧ [z1 < (0 + T)]]
                 |   |   |   | sum0 ← sum0 + v5
                 |   |   |   | M ← 0
                 |   |   | if [[T < x1 < (a − T)] ∧ [T < z1 < (c − T)] ∧ [y1 > (b − T)]]
                 |   |   |   | sum0 ← sum0 + v4
                 |   |   |   | M ← 0
                 |   |   | if [[T < x1 < (a − T)] ∧ [[T < z1 < (c − T)]] ∧ [y1 < (0 + T)]]
                 |   |   |   | sum0 ← sum0 + v3
                 |   |   |   | M ← 0
                 |   |   | if [[T < y1 < (b − T)] ∧ [[T < z1 < (c − T)]] ∧ [x1 < (0 + T)]]
                 |   |   |   | sum0 ← sum0 + v1
                 |   |   |   | M ← 0
                 |   |   | if [[T < y1 < (b − T)] ∧ [[T < z1 < (c − T)]] ∧ [x1 > (a − T)]]
                 |   |   |   | sum0 ← sum0 + 2·v1
                 |   |   |   | M ← 0
                 |   |   | if mod(G,350) = 0
                 |   |   |   | w_jj ← sum0/G + sum2/(6·G)
                 |   |   |   | jj ← jj + 1
                 | w
```

$$u2\left(\frac{3}{6},\frac{3}{6},\frac{3}{6}\right) = 0.322$$