Electromagnetic Simulation by the FDTD method in Java

STEPHEN KIRKUP, IRFAN MULLA, GOODCHILD NDOU and JAVAD YAZDANI
East Lancashire Institute of Higher Education (ELIHE)
Blackburn College
Blackburn, Lancashire
UNITED KINGDOM

Abstract: - The FDTD (Finite Difference Time Domain) is the most popular method for transient electromagnetic simulation. A FDTD method is developed through applying Yee's algorithm and a Mur boundary condition. The method is implemented in Java using object-oriented design principles. The program can simulate and visualize an evolving electromagnetic field in any cuboidal, multiple material domain. The program is made available as open source from www.east-lancashire-research.org.uk (AR-08-17).

Keywords— FDTD, Electromagnetic

1 Introduction

Electromagnetic fields are governed by Maxwell's equations. Through developing a computational model, electromagnetic fields can be simulated on computer and results can be visualised. The design of electrical components can be analysed through the application of computational methods. There are a number of techniques for the numerical solution of Maxwell's equations, the choice of technique depending on the particular circumstances [1]. This paper focuses on the finite difference time domain (FDTD) method and applies it to the design of a capacitor.

The FDTD algorithm is the most popular method for transient electromagnetic simulation. It is easy to understand, easy to implement in software, and since it is a time-domain technique, it can cover a wide frequency range with a single simulation run. [2].

The purpose of this article is to introduce a particular implementation of the FDTD program in the object-oriented programming language Java [3]. The program was originally developed as one in a library of codes for simulating the electromagnetic fields in capacitor structures [4-6] and we use the example of a capacitor structure in this paper to demonstrate the program. The program is made available as *open source*[5, 6].

Manuscript received on Aug 30th 2008. This work was supported in part by ELIHE, Blackburn College[1]. Stephen Kirkup, Javad Yazdani and Irfan Mulla[2] are with the School of Science and Technology, ELIHE[1], Blackburn College UK. Goodchild Ndou[3] is with the Department of Computing, University of Lancaster[4]. The codes associated with this paper can be downloaded from the web pages below[5, 6].

[1] www.elihe.ac.uk
[2] BSc (Hons) Computing Student, ELIHE
[3] BSc (Hons) Computing Graduate, ELIHE
[4] www.lancs.ac.uk
[5] www.east-lancashire-research.org.uk
[6] www.kirkup.info/opensource

2 The FDTD Method

The FDTD method, in its most straightforward and popular form, was introduced in Yee [7]. The domain of interest is divided into a grid of cubes and the electric field (E) at the centres of each of the faces is related to the magnetic field (H) of the pervious half time-step. In the same way, a similar grid is formed, with the cubes being a half pitch away in space and time, but this time updating the magnetic field using the electric field of the previous half time step. Using the two meshes alternatively, and starting from time t=0, we can step forward in time, applying any excitation by setting certain values within the domain at each time step.

Note that the domain of interest, mentioned earlier, is generally a truncation of the true domain. This is often a necessary approximation, but should not adversely affect the model if the main electromagnetic activity is known to occur reasonably within the applied domain. For the FDTD, the applied domain is most likely to be cuboidal, given that Yee's method involves dividing the E and H fields into grids of cubes.

Because of the artificial boundaries introduced in the application of the method, non-physical reflections of the electromagnetic signals tend to occur. There are a number of methods for reducing reflections from the applied boundary of the domain of interest. In this work the simplest of these methods was applied, that of a first order Mur boundary condition [8].

The materials need to be specified throughout the applied domain. Typically, the material will be either free-space (air), metal, or dielectrics, any material can be used, as long as the properties of permeability, permittivity, and conductivity can be specified. [2]

Once the computational domain and the grid material are established, the excitation or source is specified. The source can be an impinging plane wave, a current on a wire, or a potential difference. Since the E and H fields

are determined directly, the output of the simulation is usually the E or H field at a point or a series of points within the computational domain, or the E of H field at certain points, viewed with respect to time. [2]

3 FDTD Software Design

The FDTD Java code has been developed with object-oriented design principles. In general this means that we separate responsibilities for different parts of the method or different data structure to different objects, using the object-oriented techniques of abstraction, composition and inheritance. A UML class diagram of the FDTD code is given in Figure 1. Further information is given through viewing the codes [12, 13 Academic Report AR-08-17].

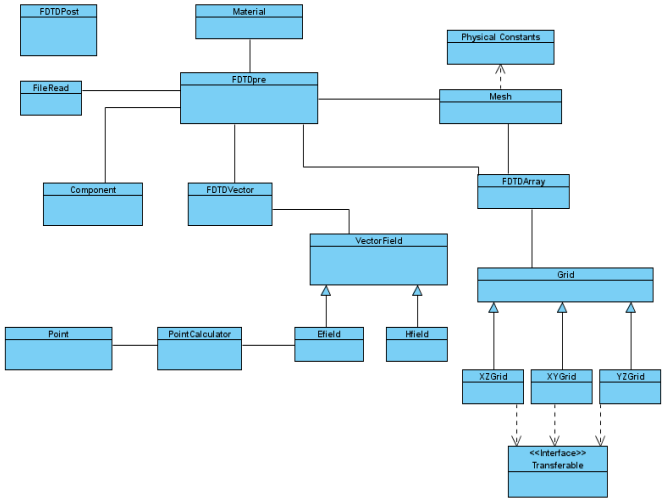


Figure 1. UML diagram of the FDTD program.

The code was originally developed by Stephen Kirkup[4]; it was then re-designed by Goodchild Ndou [9] and has now been revised further by Irfan Mulla [11] to bring it up to be released (Mark 2) as open source.

4 How to use the FDTD software

The FDTD code can simulate the electric field evolution in any cuboidal domain. The total package consists of a pre-processor which inputs a text data file and produces the electric field data files on the three central cross-sections of the domain. The text data file allows the user to define the material components of the electromagnetic domain. The components need to be defined as cuboids and the components can appear (and disappear) at defined times. The data file also allows the user to define voltage excitations within the domain. The package also includes a post-processor which reads in the electric field and outputs it to the screen.

4.1 Test Problem

The electromagnetic problem is set up by the user through completing an input file. A reasonably wide range of structures along with DC voltage sources can be defined. The input file is a .dat file. Open up the file experiment.dat (a text file e.g. with Notepad, WordPad).The experiment.dat file is a data file to describe structure, mesh and excitation used in 3D finite difference time method (FDTD). The file will be used throughout this manual to illustrate the functionality of 3D FDTD code.

For this example, the input file represents a stack of two capacitors, side-by-side, and has six inputs; the file has the name experiment.dat. To illustrate how a particular program may be described and run, we will consider the problem illustrated in Figure 2.

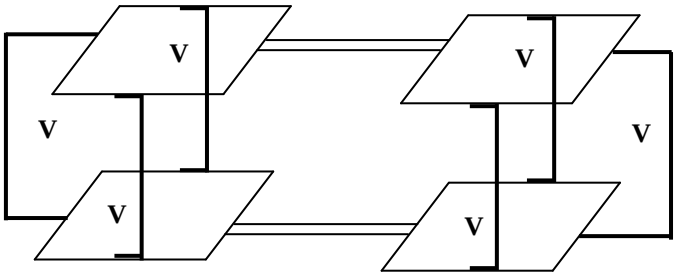


Figure 2. Linked capacitor problem.

The example structure consists of four aluminium plates of dimensions 2mm x 0.1mm x 1mm of equal distance apart (1mm) lying in a polythene medium. An aluminium *fusegate* extends and connects the plates. For more information on capacitor design see Kirkup [6] and the relevant references therein.

A voltage excitation Ey is placed between each pair of wires (in the example the voltage source is modelled by a polythene strip over which the Ey =1000 is set so the voltage across each pair of plates is 1 V). The example is described as experiment.dat

4.2 Input file for test problem

The input file, describing the test problem, is made up as follows.

[1]Name of Structure

In order to set up a particular electromagnetic problem, the user simply needs to insert the relevant numbers in the input file. One line 6 you can place name of the structure in text.

For the example, the name of the structure is called "Capacitor 2 pairs of plates"

[2] Materials

The materials that make up the structure and their properties must be defined. The first line states the name of the material and the second line states the properties of the material i.e. Conductivity, Relative Permittivity and Susceptibility (in that order).

In the example, the domain consists of two materials: polythene and aluminium. The properties for the material polythene are as follows Conductivity=0.0000000001, Relative Permittivity=2.25, Susceptibility=0.0. The properties for the material Aluminium are as follows Conductivity=35400000, Relative Permittivity=1.0, Susceptibility=0.000022

[3]Dimensions of the cuboidal electromagnetic domain

The domain is a cuboid with one corner at the origin. The dimensions here give the length [X] width [Y] and height [Z] in x, y and z co-ordinates. The electromagnetic domain is then [0 → X, 0 → Y, 0 → Z]. The whole electromagnetic domain is illustrated in Figure 3.

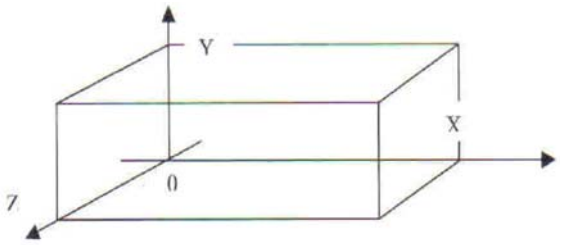


Figure 3. The cuboidal domain.

In the example, the domain is [0 → 0.028, 0 → 0.00050, 0 → 0.0014]

[4]Mesh

The mesh is made of cubic Yee cells which ideally should fit easily into dimensions, so that you have a finite number of cubes in each dimension

In this example, the domain is made of Yee cells of size 0.00002

[5]Material Distribution

The complete domain must be defined as a distribution of materials. Originally it is assumed that the domain is made up of material 1: this is the background material. However the background material can be overwritten by cuboidal components of other material (or the background material (no 1)). You can have as many such components as you want, the number of components must be stated. They must be listed one by one as shown with the information of the material index from which the component is composed and the region [x1 → x2, y1 → y2, z1→ z2] that is made up of the material.

In the example above, the background material is polythene. There are four plates of material (components 1-4) and six thin strips of material (of one or two cells thick) which constitute wires that extend from plates (5-10). The wires are joined up, each pair by a thin line of polythene (11-16).

[6]Duration of Simulation

The time (in seconds) over which the simulation runs is stated. This is the physical time, related to the electromagnetism.

In the example, the duration of simulation is 0.00000000005 seconds.

[7] Excitation

Each component can be assigned a voltage excitation. The components are listed within the voltage source defined. If a zero value is placed it is taken to mean that there is no voltage excitation.

In the example, a voltage excitation is placed on the polythene strips (components 11-16). Ey = 1000.0, so the voltage is 1V.

4.3 Java Programs and executing the software

The software is written in Java. Java is a free programming language. It can be downloaded from the website (http://java.sun.com)[3].

To download the java compiler (for computers running windows) first go to the above website in the browser. Click on the option Products. Then click on Download NetBeans IDE. It will give you numerous NetBeans IDE bundles options for downloading. Click on the Java SE download option. Download the file and follow instructions to install file.

The main codes of the FDTD package come into files

- Pre-processor: FDTDpre.java
- Post-processor: FDTDpost.java

The pre-processor takes the input electromagnetic problem, computes the electric field and saves results to hard disk. The post-processor takes the computed electric fields form the hard disk and displays it on the screen. FDTDPre.java also links to a number of other classes, as shown in the class diagram in section 3. When the .java files are compiled they become .class files, FDTDpre and FDTDpost are then ready to be executed.

The pre-processor takes the input data from the file experiment.dat and computes the electric and magnetic fields at the defined time steps. On completion of the execution of the pre-processor the output file x.out, y.out, z.out is produced. These files correspond to the electric field strengths on the central cross-sections of the domain; in the y-z plane, x-z plane and x-y plane.

The post-processor is to be run after the pre-processor has completed execution. The user is asked whether you want to view the results on the y-z plane, x-y plane. Following on from the answer to that question one of the files x.out, y.out, z.out is input and the results are displayed.

Monitoring File

It is possible that the pre-processor does not work as expected then there may be errors in the input file. Or you may wish to know how many time steps you have to wait before the execution is completed. After the preliminary seconds of the post-processing, the data pertaining to job is stored in the monitor or .mon file.
If the program is executing but the results are not as expected, then the monitor file is the first place to look and find out where the problem may have originated.

Errors

There are necessary limits on the size of the data structures that are included in the program, so that the software will fit in the given memory. Often the input problem will fit within these limits, but the limits may need to be changed by intervention into java codes by the user. Such changes are simple to perform. If such intervention occurs then the codes will need to be re-compiled.

Generally such problems will be flagged when you try to execute the programs. For example, you may get a message saying that "MaxNz is too small" and you will then have to intervene in the Java and increase MaxNx.

5 Results and Visualisation

The pre-processor FDTDpre produces data output, with some monitoring information direct to the screen and to main results to the *.out files. The post-processor FDTDpost reads the *.out files and produces a visualisation of the evolving electromagnetic field.

5.1 Output from FDTDpre

The useful output from FDTDpre – such as the electric field strength in the central cross-sectional x-, y- and z-planes are stored in .dat files. Some direct screen output is produced whilst FDTDpre is running in order to keep the user informed. An example of screen output that is produced when the FDTDpre processor is run is shown below:-

Reading Input File
MaxNtout = 1000
Nt = 1428
Ntout = 714
Sending information to the monitoring file experiment.mon
Initialise E,H data structures to zero
1
Setting Up Material Property Data Structures
Material changes
Computing H
Mur Correction in H
Apply Excitation
5.115907697472721E-12
Computing E
truetruetruetruetruetruetruetruetruetruetruetruetruetruetruetruefalse

[11]

The final line of true/false tells us which material components are in effect on each time step.

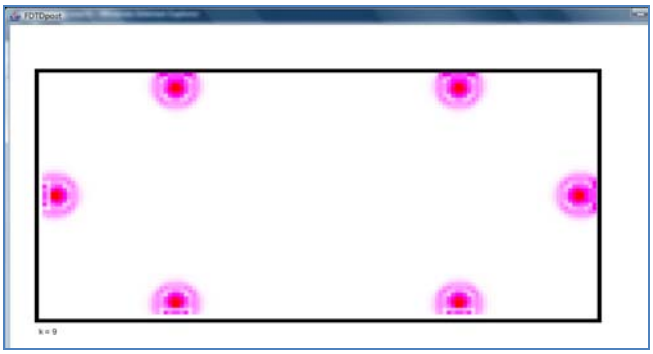
5.2 Output from FDTDpost

The FDTD post-processor reads the data files produced by FDTDpre and plots the evolving electric field to the screen. The following plots show the final outputs from the FDTD postprocessor in the horizontal cross sectional plane of the test problem. The results show the electric field strength. A colour scheme is used to illustrate the electric field strength. The colour ranges from white (low) to magenta (medium) to red (high).

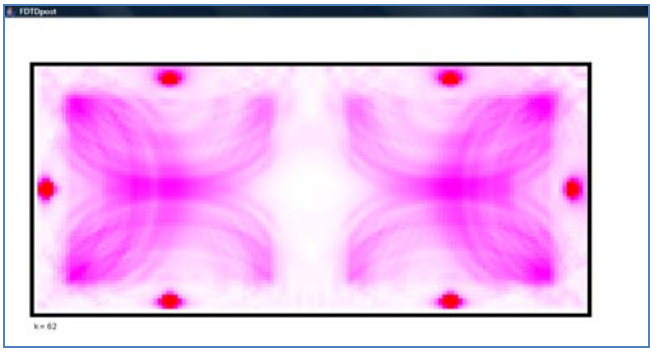
After the FDTDpre processor has finished the computation and stored the results the FDTDpost processor is ran to see a visualisation of the results. When the FDTD processor is run the following is given in the output window:-

Plane x = Plane y-z
Plane y = Plane x-z
Plane z = Plane x-y
Which plane (x, y, or z)?

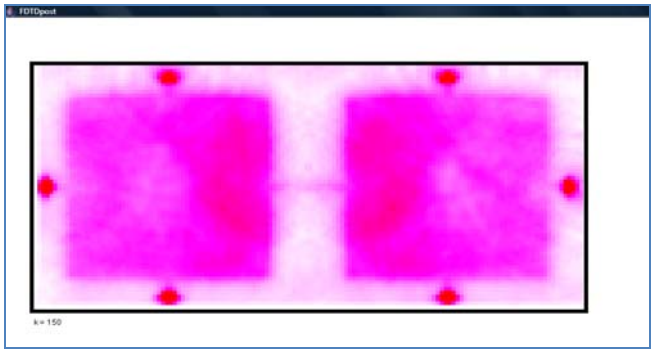
The user chooses the plane that they wish to run. In this example Plane y is chosen. The output given is illustrated in Figure 4 at a set of time steps multiples (K) with the whole simulation taking about 15minutes on a standard modern PC:-



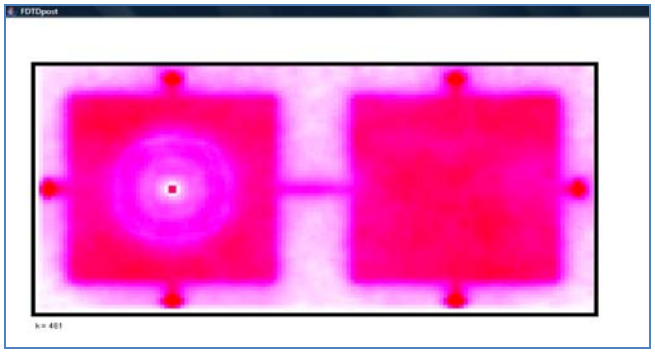
At K=9 the six voltage inputs are shown



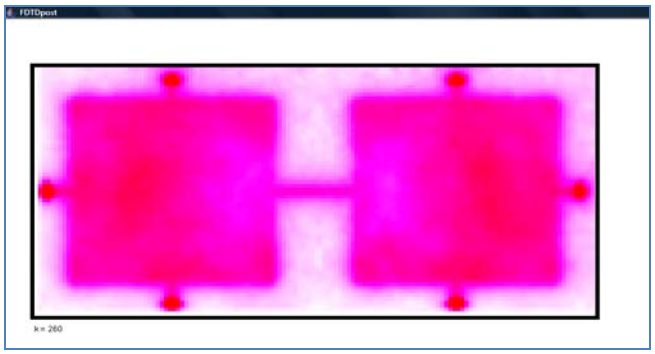
At K=62 the two capacitors are beginning to charge



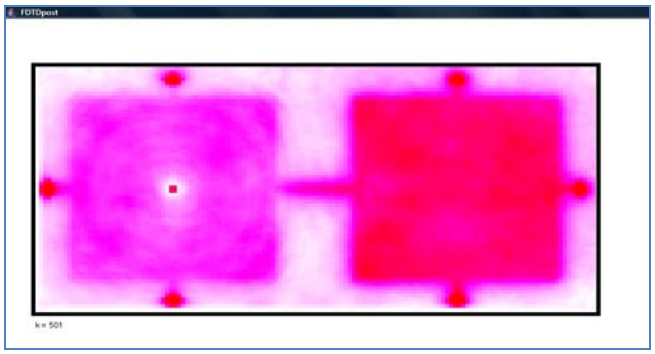
At K=150 the two capacitors are partially charged.



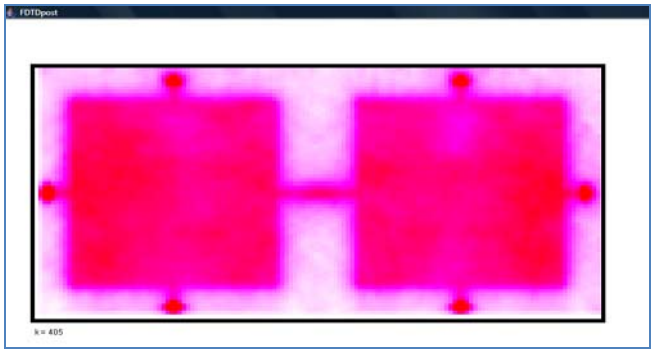
At K=461 the discharge is spreading



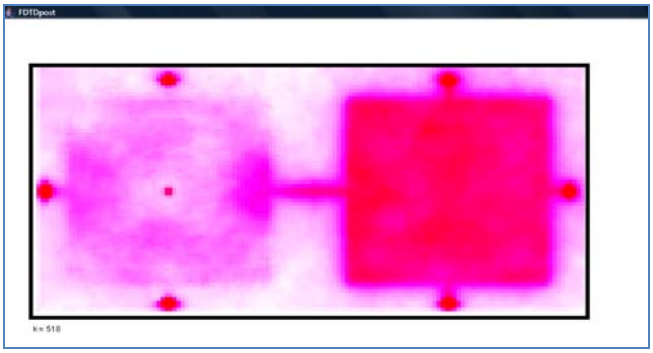
At K=260 the charge on the capacitors is increasing



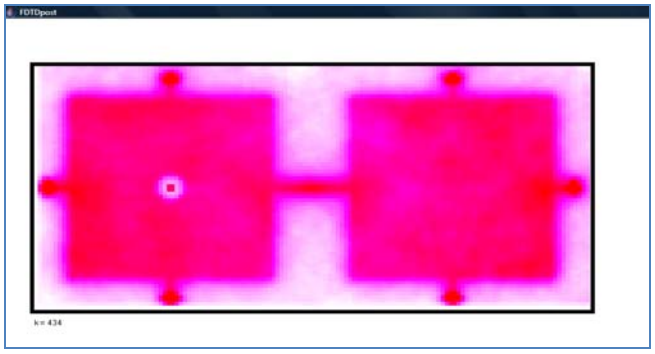
At K=501 the charge on the left capacitor is clearly less than the charge on the right capacitor



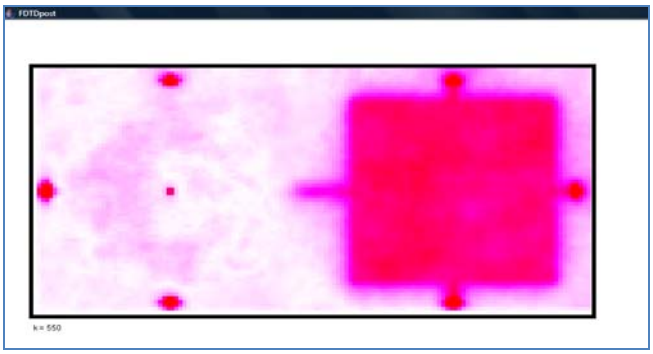
At K=405 the charge on the capacitors is increasing



At K=534 the charge on the left capacitor is weakened



At K=434 the discharge channel in the first capacitor is visible



At K =573 the charge on the left capacitor is very low. There are some oscillations in the charge following this

Figure 4. Evolving electric field strength.

4 Conclusion

In this paper the revised Mark 2 open source object-oriented Java FDTD program has been reported. It has been shown how a material domain and excitation can be defined in the .dat file, how the pre-processor FDTDpre reads the .dat file, carries out the computation of the electric and a magnetic field forward in time and sends the results to a set of .out files. It has been shown how the post-processor FDTDpost reads the .out files and produces a visualisation of the electric field strength. The program has been applied to a test problem of a capacitor structure with a change in materials to simulate a discharge channel (an inherent problem in capacitor design) and screenshots of the evolving field produced by FDTDpost are given.

There is no functional or efficiency improvement from Mark1 to Mark 2; the big change is the design of the code. The improved object-oriented design makes upgrading the software more straightforward. There are a number of areas in which the code can be improved. (i) Firstly, the method of defining the material domain and excitation could be improved; perhaps replacing the .dat file by a graphical user interface. (ii) A wider range of choices of output and a technique for allowing this in the software. (iii) Further consideration of and therefore possible improvements to the visualisation of the output. (iv) There are still aspects on the object-oriented design that could be reviewed.

As a final potential improvement, the code could be used on *larger* problems, that is larger domains or higher resolution of Yee cells or longer time evolution of higher resolution of time steps. However, in order to make progress in this direction, further consideration of the efficiency of the program would be needed. The code could be speeded up through using parallel or cluster machines [14] and through further consideration of the distribution of the objects.

Acknowledgement

The evolving FDTD code has been used by the first author in an Assignment on the BSc Computing (Hons) module *Object-Oriented Development and the Unified Modelling Language* over a number of years. The first author would like to thank all the students for their evaluation of the code; many of the comments have been taken on board. Of particular note were the contributions of Paul Sutcliffe, Chris Seaton, Dan Martin, Paul Threlfall and Katrinna MacFarlane.

References:

[1] S. Wiak, A Krawczyk, M. Trlep Computer Engineering in Applied Electromagnetism, Springer Netherlands (2005).

[2] Inkyu Park, Finite Difference FDTD technique, http://www.pas.rochester.edu/~icpark/Vinos/whatisfdtd.html. Accessed August 2008

[3] www.java.sun.com, it is free to download the Java compiler. Accessed August 2008

[4] S. M. Kirkup, Yi Huang, G. R. Jones and H. M. Looe, Capacitor Modelling by FDTD, Proceedings of the IASTED International Conference on Power and Energy Systems, July 3-6, 2001, Rhodes, Greece. (2001). Available [11].

[5] A. J. Mariani, S. M. Kirkup, Y. Huang and G. R. Jones, On coupling electromagnetic fields and lumped circuits with TLM, *Applied Mathematical Modelling*, **26**(3) , 2002, 377-396. Available [10].

[6] S. M. Kirkup, DC capacitor simulation by the boundary element method, *Comm. in Num. Meth. in Eng.* **23**(9), 855 - 869, 2007. Available [10].
Academic report AR-07-02, East Lancashire Research, ELIHE, Blackburn College. Available [13]

[7] K. S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Trans. Antennas Propagat.* **AP-14**, 302-307, 1966.

[8] G. Mur, Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations, *IEEE Trans. on Electromagnetic Compatibility,* **EMC-23**(4), 1981.

[9] G. Ndou, FDTD, BSc (Hons) Computing Projects 2006, Unpublished.

[10] I.Mulla, Finite Difference Time Domain (FDTD) technique, BSc (Hons) Computer Systems Engineering Advanced Project 2008, Unpublished.

[11] www.kirkup.info/papers

[12] www.kirkup.info/opensource

[13] www.east-lancashire–research.org.uk

[14] V. Holmes and T. McDonough, The ELIHE High-Performance Cluster, presented at the IPSI conference in London 2006. Academic Report AR-08-04, East Lancashire Research, ELIHE, Blackburn College. Available [13].