

GPRMAX2D/3D

USER'S MANUAL

VERSION 2.0

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this manual, and the author was aware of a trademark claim, the designations have been printed in initial caps or small caps.

The procedures and applications presented in this manual have been included for their instructional value. They have been tested with care but are not guaranteed for any particular purpose. The author does not offer any warranties or representations, nor does it accept any liabilities with respect to the programmes or applications.

Typeset in Concrete Roman with L^AT_EX 2_ε
Copyright © 2005 by Antonis Giannopoulos

All rights reserved. Permission is granted to reproduce and use this manual and GPRMAX software for academic or commercial purposes.

LICENCE:

GPRMAX2D V 2.0 (Electromagnetic simulator for Ground Probing Radar)

GPRMAX3D V 2.0 (Electromagnetic simulator for Ground Probing Radar)

AUTHOR: Antonis Giannopoulos

COPYRIGHT: Antonis Giannopoulos, 2005

TERMS AND CONDITIONS

This Licence applies to the GPRMAX2D and GPRMAX3D executable forms, object codes and to every file which is part of the GPRMAX2D and GPRMAX3D source codes. The "Programmes", below, refers to the executable images of GPRMAX2D and GPRMAX3D and to all the files which constitute the source codes for GPRMAX2D and GPRMAX3D. The licensee is addressed as "you". Any modification of the Programmes and their source code including translation to other programming languages is referred to as "modification".

1. You are licenced to copy and distribute only verbatim copies of the Programmes in any medium. You must keep intact all the notices that refer to their Licence and all copyright notices included in the Programmes.

Activities other than executing, copying, distribution and modification are not covered by this Licence; they are outside its scope.

2. (a) You may modify your copy or copies of the Programmes or any portions of them for your own personal use only.
(b) You may not modify your copy or copies of the Programmes or any portions of them, thus forming a work based on the Programmes for any other use different from the one stated in a) above.

NO WARRANTY

THERE IS NO WARRANTY FOR THE PROGRAMMES, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDER PROVIDES THE PROGRAMMES "AS IS" WITHOUT ANY WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAMMES IS WITH YOU. SHOULD THE PROGRAMMES PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL THE COPYRIGHT HOLDER BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAMMES (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAMMES TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Preface

GPRMAX2D and GPRMAX3D are electromagnetic wave simulators for Ground Penetrating Radar modelling. They are based on the Finite-Difference Time-Domain numerical method. This USER'S MANUAL describes the installation of GPRMAX2D/3D and the various commands which are available for the construction of 2D and 3D GPR models. Some simple examples of GPR models are provided to illustrate the procedures that should be followed in order to obtain useful models using GPRMAX2D/3D. Please do report any errors or omissions to the author.

If you require further information about GPRMAX2D and GPRMAX3D or you have any suggestions for improvements to the software or to this USER'S MANUAL please contact the author:

Dr Antonis Giannopoulos
University of Edinburgh
School of Engineering and Electronics
Institute for Infrastructure and Environment
AGB Building, The King's Buildings,
Edinburgh, EH9 3JN, Scotland
E-mail: A.Giannopoulos@ed.ac.uk

Preface to Version 2.0

The GPRMAX2D/3D codes have been around for some time since their initial development in 1996. At that time, 32Mb of RAM and processor speeds of 100 MHz were considered top of the range in computer technology! Clearly things have moved on. As computer power increases we are able to realistically model more complex GPR scenarios and even attempt models of GPR transducers. Version 2.0 of GPRMAX mainly brings the 2D model in a par with the 3D in terms of development of the underlying numerical modelling technique.

New with Version 2.0 are:

- Perfectly Matched Layer (PML) boundaries for both¹ GPRMAX2D and GPRMAX3D
- User specified excitation functions for both GPRMAX2D and GPRMAX3D
- Huygens's Surface in GPRMAX3D
- Plane Wave excitation in GPRMAX3D for free space simulations which will be extended for half space simulations in future.
- Simulation of thin-wires in GPRMAX3D.
- Voltage sources in GPRMAX3D along with 1D transmission lines for feeding models of GPR antennas.
- New excitation structure which makes GPRMAX2D/3D more general in use and allows for multiple sources and receivers at the same time.
- New geometry primitive in GPRMAX3D to create cylinders of any orientation in the model space.
- Both codes have been parallelised using OpenMP ²
- The cryptic error codes of GPRMAX2D have been replaced by error messages as in GPRMAX3D.

I hope to continue developing GPRMAX in the future so it will become more versatile than what it is at the moment. If you need to work with the source code you could request it in writing. You should state that you will not redistribute the code to anyone either verbatim or in any modified version of it. Finally you should put in a few words about the reasons for needing it.

For any ideas or bugs or collaboration please do not hesitate to contact me!

Antonis Giannopoulos, 2005

¹PML boundaries were introduced for GPRMAX3D with Version 1.5

²The executables are serial versions due to licensing of OpenMP compilers. Contact me for more information.

Acknowledgements

I would like to thank the Building Research Establishment (BRE) UK, for providing me with the financial support to develop GPRMAX2D/3D after completing my D.Phil thesis on GPR modelling at the University of York. Without their support which helped to develop the core of the GPRMAX programmes this current release today would not have been possible.

Contents

1	Installation and Getting Started	1
1.1	Contents of the CD-ROM	1
1.2	Downloading from the Website	1
1.3	Running GPRMAX2D/3D	2
1.4	General structure of the input file	2
2	Ground Penetrating Radar modelling with GPRMAX2D/3D	4
2.1	Basic concepts of GPR modelling	4
2.2	Assumptions for the modelling of GPR in two and three dimensions	6
2.3	Capabilities and limitations of GPRMAX2D/3D	7
2.4	GPRMAX2D coordinate system and conventions	7
2.5	GPRMAX3D coordinate system and conventions	8
2.6	Hints and tips for GPR modelling with GPRMAX2D/3D	8
3	GPRMAX2D input file commands	11
3.1	General notes on GPRMAX2D commands	11
3.2	List of all available commands	12
3.3	GPRMAX2D command parameters	13
3.3.1	General commands	13
3.3.2	ABC related commands	17
3.3.3	Media and Object construction commands	19
3.3.4	Excitation and output commands	22
4	GPRMAX3D input file commands	27
4.1	General notes on GPRMAX3D commands	27
4.2	List of all available commands	27
4.3	GPRMAX3D command parameters	29
4.3.1	General commands	29
4.3.2	ABC related commands	30

4.3.3	Media and Object construction commands	31
4.3.4	Excitation and output commands	36
5	GPRMAX2D/3D modelling examples	43
5.1	Single rebar in concrete. – 2D model	43
5.2	Other 2D examples	46
5.2.1	BRE example #2	47
5.2.2	BRE example #4	49
5.3	Snapshots of the electric field of a horizontal dipole over a dielectric half-space. – 3D model	50
5.4	GPR response of a small void. – 3D model	51
6	GPRMAX2D/3D Output File formats	55
6.1	GPRMAX2D/3D Binary format file header	55
6.2	GPRMAX2D BINARY file formats	56
6.3	GPRMAX2D ASCII file formats	59
6.4	GPRMAX3D BINARY file formats	59
6.5	GPRMAX3D ASCII file formats	63
A	GPRMAX2D/3D Media file	64
B	GPRMAX2D/3D Constants and excitation functions	66
	GPRMAX2D/3D Constants	66
	Excitation functions	67
C	MATLAB functions for GPRMAX2D/3D	68
	Bibliography	69

Figures

2.1	The 3D FDTD Yee cell	5
2.2	The 2D GPR forward problem and its GPRMAX2D domain bounded by ABCs . . .	6
2.3	Schematic of the GPRMAX2D coordinate system and conventions (see text). The components included in the oval shaped area are the ones which correspond to space coordinate 3 ($\Delta x = \Delta y = 1$ metre).	8
2.4	Schematic of the GPRMAX3D coordinate system and conventions. The depicted field components are the ones which correspond to space coordinate 1. ($\Delta x = \Delta y = \Delta z = 1$ metre)	9
5.1	Schematic drawing of the rebar in concrete problem.	44
5.2	Normalized power spectrum and time waveform of the ricker excitation function. The centre frequency is 900 MHz.	45
5.3	Simulated GPR scan by GPRMAX2D (left) and image representation of the geometry of the model (right). Note that the vertical scale is exaggerated.	47
5.4	Schematic drawing of the BRE examples 1 and 2	48
5.5	Schematic drawing of the BRE examples 3 and 4	49
5.6	Image produced using the information in the geometry file of the BRE examples 1 and 2 (Note vertical scale is exaggerated).	50
5.7	Image produced using the information in the geometry file of the BRE examples 3 and 4 (Note vertical scale is exaggerated).	51
5.8	Simulated GPR scan of the BRE second example.	52
5.9	Simulated GPR scan of the BRE fourth example.	53
5.10	Three slices through a volume of data. The amplitude of the E_x field component is represented as a gray scale image. The figure was created in MATLAB from the GPRMAX3D data.	53
5.11	Schematic drawing of the 3D model of a small void (after [1])	54
5.12	GPRMAX3D simulated GPR scan over a small void	54
6.1	Representation in pseudo-code of the procedure used in GPRMAX2D in order to store the values of field components in a snapshot	58
6.2	Representation in pseudo-code of the procedure used in GPRMAX3D in order to store the values of field components in a snapshot	62

Chapter 1

Installation and Getting Started

This chapter describes the installation procedure for GPRMAX2D/3D.

1.1 Contents of the CD-ROM

If you have obtained a CD-ROM distribution of GPRMAX then in order to install the right executable for your computer locate the directory which describes your operating system and copy the file(s) into a suitable directory in your machine.

LINUX users should copy the binaries `GprMax2D` and `GprMax3D` found in the `LINUX` directory of the CD-ROM into a suitable directory on their system - preferably one in your path - For example at: `/usr/local/bin`¹. The GPRMAX executable should work on all recent LINUX distributions but it has only been tested on REDHAT ENTERPRISE 4

MS WINDOWS 9X/2000/NT/XP users should copy the three files `GprMax2D.exe`, `GprMax3D.exe` and `cygwin1.dll` found under the `Windows` directory into a suitable directory on the system's hard disk (e.g. `GprMax`). The latter file is required since GPRMAX has been compiled using CYGWIN for the MS WINDOWS platform. Alternatively you can copy the `cygwin1.dll` file into your WINDOWS system directory.

MAC OS X users should copy the files found in the `MacOs` directory to an appropriate location on their system. GPRMAX2D/3D binaries for macs have not been tested as I do not have access to a mac!

In the directory `manual` there is a copy of this manual in ADOBE PDF format. If you do not have the free ADOBE ACROBAT READER you can download it from www.adobe.com

In the directory `examples` some input files used as examples in this manual have been included. The directory `media` contain a sample media file and in the directory `tools` there are some MATLAB functions that can be used to import GPRMAX2D/3D modelled data, stored in binary format, in MATLAB.

1.2 Downloading from the Website

If you have downloaded the compressed .ZIP archive from the GPRMAX website at www.gprmax.org or www.gprforum.org or from my web page at www.see.ed.ac.uk/~agianno follow the same procedures as described above for the CD-ROM distribution after uncompressing it to a directory of your preference.

¹That will require root privileges. If you do not know the root password then you can run the program from a directory in your home space.

1.3 Running GPRMAX2D/3D

To run a 2D or 3D GPR model using GPRMAX2D/3D you have to create an input file in which the parameters of the model are specified. This input file is a plain ASCII text file and should have a valid filename of your preference. The structure of this input file is explained in detail in subsequent chapters.

To run the programs in an MS WINDOWS environment open a system (DOS) command window. In recent MS WINDOWS OS versions you can open a command window by clicking on the Run . . option and typing cmd. At the command window then type at the prompt:

```
GprMax2D your_path\your_input_file
```

The `your_path` can be the absolute path where the input file is located or the relative path where the input file is located with respect to the directory you are currently in. For example, assume your executable image of GPRMAX2D is located in the directory `C:\GprMax\Bin` and assume that your input file is in the directory `C:\GPRModels\Inputs` and is called `Model1.in`. If your current directory is `C:\GprMax` you can issue the commands:

```
Bin\GprMax2D C:\GPRModels\Inputs\Model1.in
```

or

```
Bin\GprMax2D ..\GPRModels\Inputs\Model1.in
```

in case you have included in your PATH the directory `C:\GprMax\Bin` then you can type:

```
GprMax2D C:\GPRModels\Inputs\Model1.in
```

or

```
GprMax2D ..\GPRModels\Inputs\Model1.in
```

To run the programs in a LINUX environment follow the same procedure as for an MS-WINDOWS system².

1.4 General structure of the input file

The input file which has to be supplied to GPRMAX2D/3D contains all the necessary information to run a GPR model. Detailed description of the various commands available that can be used in GPRMAX2D/3D input files are described later in this manual. However, the general structure of an input file is the same in both programs.

The input file is a plain ASCII text file which can be prepared with any editor or word-processing program. In the input file the hash character (#) is reserved and is used to denote the beginning of a command which should be passed to the programs. The general syntax of all available commands is:

```
#command_name: parameter1 parameter2 parameter3 ...
```

For example, in GPRMAX2D the size of the computational domain is specified with a command

²Directory names are separated by a / on a LINUX system and device names like (C:) are not used.

```
#domain: 1.0 1.0
```

A command with its parameters should occupy a single line of the input file and only one command per line is allowed. Hence the first character of a line containing a command **must** be the hash character (#). If the line starts with **any other character** it is ignored by the program. Therefore, user comments or descriptions can be included in the input file. If a line starts with a hash character (#) the program will expect a valid command. In case the command is not spelt correctly the program will abandon execution issuing an error message. When a command requires more than one parameter then these are separated using a white space character.

The order of introducing commands into the input file is not important with the exception of model building commands and execution (i.e. analysis) group commands which will be described in the relevant Chapter (3). Moreover, as will be explained later, introducing particular commands before others will facilitate the creation of any desired GPR model.

Chapter 2

Ground Penetrating Radar modelling with GPRMAX2D/3D

This chapter discusses some basic concepts of GPR modelling using GPRMAX2D/3D. To accomplish such a task GPRMAX2D/3D solve Maxwell's equations using the *finite-difference time-domain* method (FDTD). Detailed information about the use of FDTD for GPR modelling can be found in [1] and general information on the FDTD method in [2] and [3].

2.1 Basic concepts of GPR modelling

All electromagnetic phenomena, on a macroscopic scale, are described by the well known Maxwell's equations. These are first order partial differential equations which express the relations between the fundamental electromagnetic field quantities and their dependence on their sources.

$$(2.1) \quad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$(2.2) \quad \nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J}_c + \mathbf{J}_s$$

$$(2.3) \quad \nabla \cdot \mathbf{B} = 0$$

$$(2.4) \quad \nabla \cdot \mathbf{D} = q_v$$

where t is time (seconds) and q_v is the volume electric charge density (coulombs/ cubic meter). In Maxwell's equations, the field vectors are assumed to be single-valued, bounded, continuous functions of position and time. In order to simulate the GPR response from a particular target or set of targets the above equations have to be solved subject to the geometry of the problem and the initial conditions.

The nature of the GPR forward problem classifies it as an *initial value – open boundary* problem. This means that in order to obtain a solution one has to define an initial condition (i.e. excitation of the GPR transmitting antenna) and allow for the resulting fields to propagate through space reaching a zero value at infinity since, there is no specific boundary which limits the problem's geometry and where the electromagnetic fields can take a predetermined value. Although the first part is easy to accommodate (i.e. specification of the source), the second part can not be easily tackled using a finite computational space.

The FDTD approach to the numerical solution of Maxwell's equations is to discretize both the space and time continua. Thus the discretization spatial Δx , Δy and Δz and temporal Δt steps play a very significant role – since the smaller they are the closer the FDTD model is to a real representation of the problem. However, the values of the discretization steps always have to be

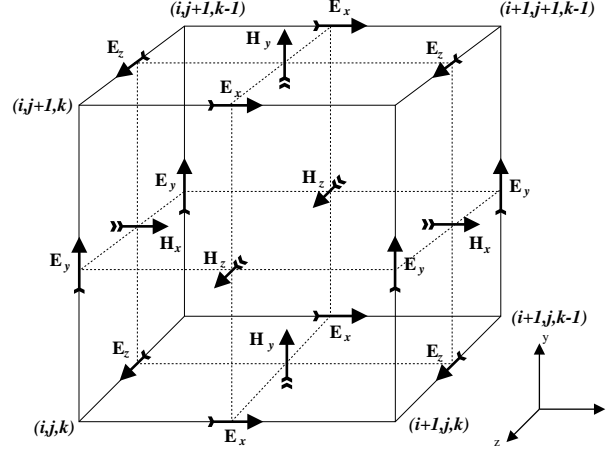


Figure 2.1: The 3D FDTD Yee cell

finite, since computers have a limited amount of storage and finite processing speed. Hence, the FDTD model represents a discretized version of the real problem and of limited size. The building block of this discretized FDTD grid is the Yee cell [4] named after Kane Yee who pioneered the FDTD method. This is illustrated for the 3D case in Figure 2.1. The 2D FDTD cell is a simplification of the 3D one and is depicted in Figure 2.3.

By assigning appropriate constitutive parameters to the locations of the electromagnetic field components complex shaped targets can be included easily in the models. However, objects with curved boundaries are represented using a staircase approximation.

The numerical solution is obtained directly in the time domain by using a discretized version of Maxwell's curl equations which are applied in each FDTD cell. Since these equations are discretized in both space and time the solution is obtained in an iterative fashion. In each iteration the electromagnetic fields advance (propagate) in the FDTD grid and each iteration corresponds to an elapsed simulated time of one Δt . Hence by specifying the number of iterations one can instruct the FDTD solver to simulate the fields for a given time window.

The price one has to pay of obtaining a solution directly in the time domain using the FDTD method is that the values of Δx , Δy , Δz and Δt can not be assigned independently. FDTD is a conditionally stable numerical process. The stability condition is known as the CFL condition after the initials of Courant, Freidrichs and Lewy and is

$$(2.5) \quad \Delta t \leq \frac{1}{c \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}}$$

where c is the speed of light. Hence Δt is bounded by the values of Δx , Δy and Δz . The stability condition for the 2D case is easily obtained by letting $\Delta z \rightarrow \infty$.

One of the most challenging issues in modelling *open boundary* problems as the GPR one is the truncation of the computational domain at a finite distance from sources and targets where the values of the electromagnetic fields can not be calculated directly by the numerical method applied inside the model. Hence, an approximate condition known as *absorbing boundary condition* (ABC) is applied at a sufficient distance from the source to truncate and therefore limit the computational space. The role of this ABC is to absorb any waves impinging on it, hence simulating an unbounded space. The computational space (i.e the model) limited by the ABCs should contain all important features of the model such as sources and output points and targets. Figure 2.2

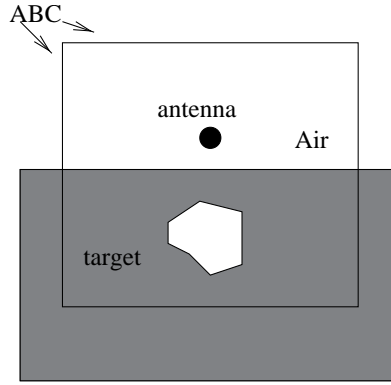


Figure 2.2: The 2D GPR forward problem and its GPRMAX2D domain bounded by ABCs

illustrates this basic difference between the problem to be modelled and the actual FDTD modelled space. In Figure 2.2 it is assumed that the half-space which contains the target(s) is of infinite extent. Therefore, the only reflected waves will be the ones originating from the target. In cases where the host medium is not of infinite extent (e.g. a finite concrete slab) the assumption of infinite extent can be made as far as the actual reflections from the slab termination are not of interest or its actual size is large enough that any reflected waves which will originate at its termination will not affect the solution for the required time window. In general, any objects that span the size of the computational domain (i.e. model) are assumed to extend to infinity. The only reflections which will originate from their termination at the truncation boundaries of the model are due to imperfections of the ABCs and in general are of a very small amplitude compared with the reflections from target(s) inside the model.

All other *boundary conditions* which apply at interfaces between different media in the FDTD model are automatically enforced in GPRMAX2D/3D.

2.2 Assumptions for the modelling of GPR in two and three dimensions

In constructing a GPR model in two and three dimensions, some assumptions are necessary. These mainly result from the need to keep the amount of computational resources, required by the model, to a manageable level and to facilitate the study of the important features of the GPR response to a target, without cluttering the solution with details which will obscure the fundamental response. However, GPRMAX2D/3D can easily handle more “complicated” GPR modelling scenarios if required. The assumptions made for both GPRMAX2D and GPRMAX3D models are:

- all media are considered to be linear and isotropic.
- In GPRMAX3D if the physical structure of the GPR antenna is not included in the model then the antenna is modelled as an ideal Hertz dipole (i.e. a small current source) . In GPRMAX2D the transmitting antenna is modelled as a line source. Using a Hertz dipole in GPRMAX3D results in reducing the computational resources required to run a 3D model in reasonable time, whereas in the GPRMAX2D case the use of a line source is a consequence of the assumption of the invariance of the problem in one direction.
- the constitutive parameters are, in most cases, assumed not to vary with frequency. This assumption simplifies a time domain model. However, a formulation able to handle a Drude (i.e. Debye plus a constant conductivity) relaxation model for the complex permittivity is included in both GPRMAX2D and GPRMAX3D.

Therefore, for the 2D case the governing equations reduce to the ones describing the propagation of *TM* mode electromagnetic waves (relative to the invariance direction (z) of the model).

2.3 Capabilities and limitations of GPRMAX2D/3D

In general, GPRMAX2D/3D solve numerically Maxwell's equations in two (TM_z case) and three dimensions. Any linear, isotropic media with constant constitutive parameters can be included in the model. In addition they can model dielectrics with frequency depended permittivity described by the Debye formula

$$(2.6) \quad \epsilon = \epsilon_{\infty} + \frac{\epsilon_s - \epsilon_{\infty}}{1 + j\omega\tau}$$

where ϵ_{∞} is the permittivity at light frequencies, ϵ_s is the DC permittivity and ω is the angular frequency. The computational domain of both GPRMAX2D/3D could be truncated either using a third order Higdon local absorbing boundary condition (ABC) or by introducing Perfectly Matched Layers (PML) in the model. The default is the Higdon ABC however the performance of the PML is superior especially if more than six (6) layers are used. The parameters of the Higdon ABC can be altered if required using simple commands. In the case of PML only the number of layers of the PML is adjustable by the user.

Excitation of a model in GPRMAX2D is achieved by specifying the current of a line source. In GPRMAX3D - when a Hertzian dipole is used - excitation is achieved by specifying the current and polarisation of the small Hertzian dipole. More than one sources can be active at a given time thus making simulation of GPR arrays simple. There is a choice of excitation waveforms for GPRMAX2D/3D. In addition the user can specify its own excitation function.

The discretization of the GPRMAX2D/3D models can be different in the direction of each coordinate axis but can not vary along this direction. The smallest element in GPRMAX2D which can be allocated with user defined characteristics is an area of $\Delta x \times \Delta y$ and in GPRMAX3D a volume of $\Delta x \times \Delta y \times \Delta z$. For a list of the technical characteristics of GPRMAX2D/3D see Appendix B.

2.4 GPRMAX2D coordinate system and conventions

The GPRMAX2D FDTD algorithm is implemented in the (x - y) plane. The origin of the coordinate system is the *lower left* corner at (0,0). The smallest space that can be allocated to represent a specific medium is a 2D cell ($\Delta x \times \Delta y$). The reference point of the 2D cell is its centre. However, the space coordinates range from the left edge of the first cell to the right edge of the last one. In Figure 2.3 the coordinate system of GPRMAX2D is schematically presented. Only one row of cells in the x direction is depicted. Assuming that $\Delta x = 1$ metre, if one wants to allocate a rectangle with its x dimension equal to 3 metres and 1 as its lower x coordinate the x range is [1..4]. The 2D cells allocated by GPRMAX2D are [1..3]. When source or output positions are specified in space coordinates they correspond directly to cell coordinates. It is important to remember that for a given set of space coordinates (x,y) the actual position of the electromagnetic field components are:

- $(x + \frac{\Delta x}{2}, y + \frac{\Delta y}{2})$ for E_z
- $(x + \frac{\Delta x}{2}, y)$ for H_x
- $(x, y + \frac{\Delta y}{2})$ for H_y

due to the staggered arrangement of field components in the FDTD algorithm. Therefore, the interface between two cells of different constitutive parameters is located on the positions of the magnetic field components. Further, it is important to remember that all sources are actually located at the positions of the E_z field component.

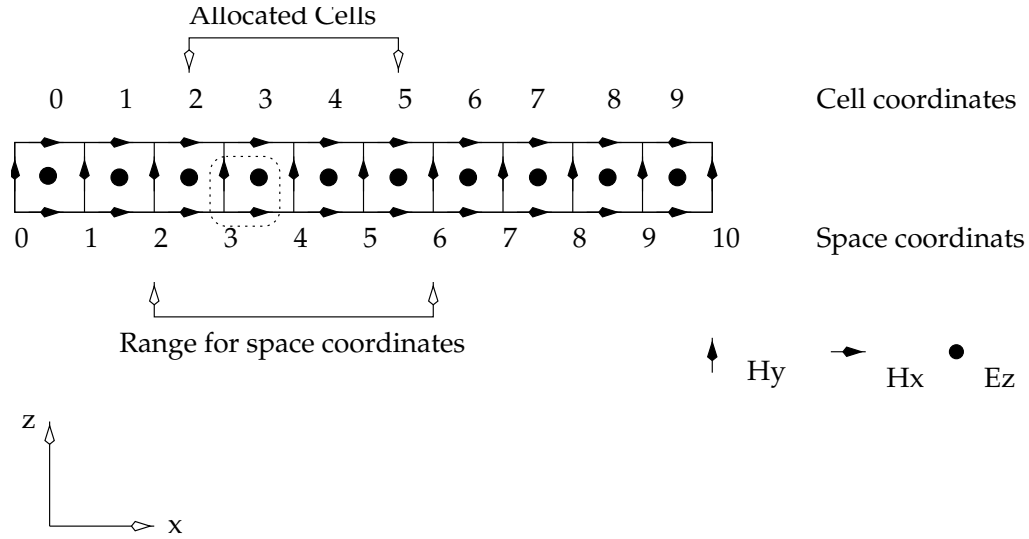


Figure 2.3: Schematic of the GPRMAX2D coordinate system and conventions (see text). The components included in the oval shaped area are the ones which correspond to space coordinate 3 ($\Delta x = \Delta y = 1$ metre).

2.5 GPRMAX3D coordinate system and conventions

The coordinate system used in GPRMAX3D is a right-handed Cartesian. The origin of space coordinates is (0,0,0) and the analogy between space coordinates and 3D FDTD cells is very similar to the one in GPRMAX2D explained above. The difference in GPRMAX3D is that the 3D FDTD cell is rather more complicated than in the 2D case and there are no field components located at the centre of the cell. At interfaces between cells the electric field components are tangential to them and the magnetic field components are normal to them. Source and output points defined in space coordinates are directly converted to cell coordinates and the corresponding field components are depicted in Figure 2.4. The actual position of field components for a given set of space coordinates (x, y, z) are:

- $(x + \frac{\Delta x}{2}, y, z)$ for E_x
- $(x, y + \frac{\Delta y}{2}, z)$ for E_y
- $(x, y, z + \frac{\Delta z}{2})$ for E_z
- $(x, y + \frac{\Delta y}{2}, z + \frac{\Delta z}{2})$ for H_x
- $(x + \frac{\Delta x}{2}, y, z + \frac{\Delta z}{2})$ for H_y
- $(x + \frac{\Delta x}{2}, y + \frac{\Delta y}{2}, z)$ for H_z

It should be noted that Hertzian dipole sources as well as other electric field excitations (i.e. voltage sources, transmission lines) are actually located at the corresponding electric field components.

2.6 Hints and tips for GPR modelling with GPRMAX2D/3D

In order to make the most of GPRMAX2D/3D in modelling GPR responses ultimately one should be familiar with the FDTD method on which these programs are based. There is a very large amount of information available in the relevant literature. Good starting points are [2] and [3] where as the specific application of FDTD to the GPR forward problem is described in [1].

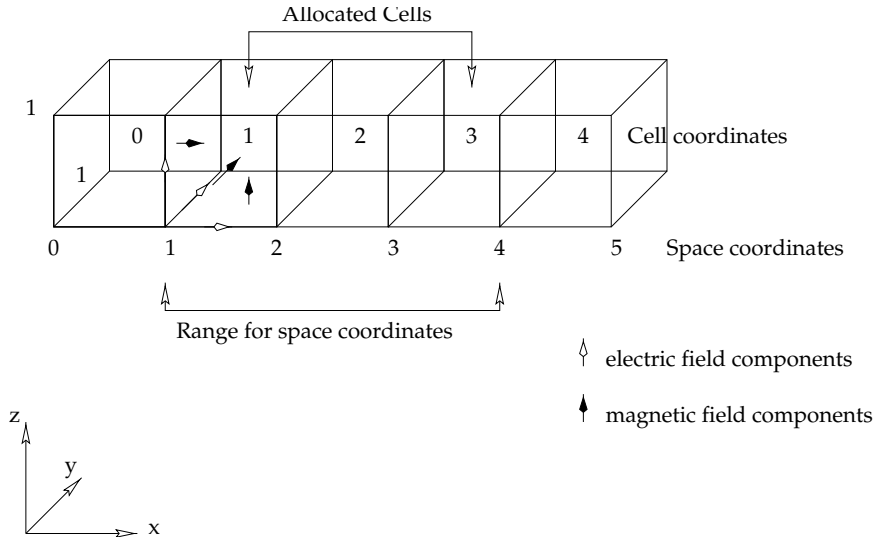


Figure 2.4: Schematic of the GPRMAX3D coordinate system and conventions. The depicted field components are the ones which correspond to space coordinate 1. ($\Delta x = \Delta y = \Delta z = 1$ metre)

This manual, can not serve as an in depth tutorial and a review of the FDTD method. However, some useful hints and tips are given here in order to cover the most fundamental aspects of using an FDTD based program and avoid the most common errors.

Discretization

There is no specific guideline for choosing the right discretization for a given problem. In general, it depends on the required accuracy, the frequency content of the source's pulse and the size of the targets. Obviously, all targets present in a model must be adequately resolved. This means, for example, that a cylinder with radius equal to one or two spatial steps does not really look like a cylinder!

An other important factor which influences the discretization is the errors associated with numerical induced dispersion. This means that contrary to the real world where electromagnetic waves propagate with the same velocity irrespectively of their direction and frequency (assuming no dispersive media and far-field conditions) in the discrete one this is not the case. This error (details can be found in [1] and [2]) can be kept in a minimum if the following *rule of thumb* is satisfied:

the discretization step should be at least ten times smaller than the smallest wavelength of the propagating electromagnetic fields.

Note that in general low-loss media wavelengths are much smaller compared to free space. The above rule is described by the equation:

$$(2.7) \quad \Delta l = \frac{\lambda}{10}$$

Absorbing boundary conditions

The ABCs employed in GPRMAX2D/3D will, in general, perform well (i.e. without introducing significant artificial reflections) if all sources and targets are kept at least 15 cells away from them. The formulation of the type of ABCs employed in GPRMAX2D/3D does not take into account any near-field effects which dominate close to radiation centres (i.e. sources and targets).

Form Version 2.0 Perfectly Matched Layer (PML) absorbing boundaries could be used in GPRMAX2D as well as in GPRMAX3D. These layers which have a user adjustable thickness absorb very efficiently most waves that propagate in them. Although, source and output points can be specified inside these layers **it is wrong to do so** from the point of view of correct modelling. The fields inside these layers are not of interest to GPR modelling. Placing sources inside these layers could have effects that have not been studied and will certainly provide erroneous results from a GPR modeller's point of view.

The above requirements have to be taken into account when the size of the model is to be decided. Further, for the same reason, free space (i.e. air) should be always included above a source for at least 15 to 20 cells in both GPRMAX2D and GPRMAX3D GPR models.

Obviously, the more cells there are between observation points, sources, targets and the absorbing boundaries the better the results will be.

Chapter 3

GPRMAX2D input file commands

3.1 General notes on GPRMAX2D commands

In order to describe the GPRMAX2D commands and their parameters the following conventions are used:

- **f** means a real number which can be entered using either a [.] separating the integral from the decimal part (e.g. 1.5) or in scientific notation as 15e-1 or 0.15e1
 - **i** means an integer number
 - **c** means a single character (e.g. *y*)
 - **str** means a string of characters with *no* white spaces in between (e.g. *sand*)
 - **file** means a filename to be supplied by the user
-
- ☞ All parameters associated with simulated space (i.e. size of model, spatial increments, e.t.c.) should be specified in **metres**.
 - ☞ All parameters associated with time (i.e. total simulation time, time instants, e.t.c) should be specified in **seconds**.
 - ☞ All parameters denoting frequency should be specified in **Hertz**.
 - ☞ All parameters associated with spatial coordinates in the model should be specified in **metres**. The origin of the coordinate system (**0,0**) is at the lower left corner of the model.

It is important to note that GPRMAX2D converts spatial and temporal parameters given in **metres** and **seconds** respectively to integer values corresponding to **FDTD cell coordinates** and **iteration number**. Therefore, rounding to the nearest integer number of the user defined values is performed.

There are **no optional** parameters within a command. Even if some parameters will not be affecting the model's values they have to be supplied.

The order which the commands appear in the input file is not significant with the single exception of the **#analysis:** and **#end_analysis:** commands which since Version 2.0 specify the excitation and outputs of the model.

The fundamental spatial and temporal discretization steps are denoted as Δx , Δy and Δt respectively.

3.2 List of all available commands

In order to construct a 2D GPR model using GPRMAX2D a set of commands are available which can be used to give the necessary instructions to the program.

There are 32 commands available in Version 2.0 of GPRMAX2D :

```
#title:
#domain:
#dx_dy:
#time_step_stability_factor:
#time_window:
#messages:
```

```
#number_of_media:
#nips_number:
#media_file:
#geometry_file:
#medium:
```

```
#abc_type:
#abc_order:
#abc_stability_factors:
#abc_optimization_angles:
#abc_mixing_parameters:
#pml_layers:
```

```
#box:
#cylinder:
#x_segment:
#y_segment:
#triangle:
```

```
#analysis:
#end_analysis:
#tx:
#rx:
#rx_box:
#snapshot:
#tx_steps:
#rx_steps:
```

```
#line_source:
#excitation_file:
```

the parameters that each command requires is explained in the following section.

There are some GPRMAX commands that have been retired from Version 2.0 onwards and will not function anymore in GPRMAX2D. Their functionality has been replaced by new more enhanced structures and commands. The commands that have been “retired” and **are not functioning anymore** are:

```
#scan:
```

```
#csg:
#extra_tx:
```

In addition some of the old commands have changed in terms of the number of parameters they take. These are:

```
#tx:
#snapshot:
```

3.3 GPRMAX2D command parameters

To aid the presentation of GPRMAX2D commands they have been grouped in four categories:

- ① *General commands* which include the ones used to specify the size and discretization of the model.
- ② *ABC related commands* which allow the customisation and optimisation of the absorbing boundary conditions.
- ③ *Media and Object construction commands* which are used to introduce different media in the model and construct simple geometric shapes with different constitutive parameters.
- ④ *Excitation and Output commands* which are used to place source and output points in the model.

Most of the commands are optional with the exception of the ones which are necessary in order to construct the simplest model possible. For example, none of the commands of the *Media and Object construction* section are necessary to run a model. However, without specifying any objects in the model GPRMAX2D simulates a free space (air) region not of particular use for GPR modelling. If you have not specified a command which is essential in order to run a model, for example the size of the model, GPRMAX2D will terminate execution issuing an appropriate error message.

☞ The *essential* commands which represent the minimum set of commands required to run GPRMAX2D are:

- #domain:
- #dx_dy:
- #time_window:
- At least one #analysis: with the corresponding #end_analysis: commands enclosing at least one #tx: and one #rx: and/or #rx_box: commands
- In order for the #tx: command to function properly a new #line_source: command is required as well.

3.3.1 General commands

#title:

With the command #title you can include a title for your model. This title is saved in the output file(s). The syntax of the command is:

```
#title: str
```

The parameter `str` here can contain white space characters to separate individual words. The title has to be contained in a single line.

#domain:

The command `#domain:` should be used to specify the size in metres of the model. Its syntax is:

```
#domain: f1 f2
```

The parameters `f1` and `f2` are the size in metres of your model in the x and y direction respectively. For example:

```
#domain: 1.0 1.5
```

will set the size of the model to be 1.0×1.5 metres.

#dx_dy:

The command `#dx_dy` is used to specify the discretization of space in the x and y directions respectively (i.e. Δx and Δy). The syntax of this command is:

```
#dx_dy: f1 f2
```

where `f1` is the spatial step in the x direction (Δx) and `f2` is the spatial step in the y direction (Δy). Hence a command as

```
#dx_dy: 0.1 0.1
```

will set the discretization steps Δx and Δy to 10 centimetres in each direction. The choice of the discretization steps Δx and Δy is very important in constructing a model. This command combined with the `#domain` command determines the number of cells which are going to be used in the model and consequently the greatest part of the requirements for computer memory for the model. For example using the commands:

```
#domain: 1.0 1.5  
#dx_dy: 0.1 0.1
```

The number of cells in the model will be 10×15 . Changing the above to

```
#domain: 1.0 1.5  
#dx_dy: 0.01 0.01
```

the number of cells in the model will increase to 100×150 . Further, the spatial discretization controls the maximum permissible time step Δt with which the solution advances in time to reach the required simulated time window. The relation between Δt and $\Delta x, \Delta y$ is

$$(3.1) \quad \Delta t \leq \frac{1}{c \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}}$$

where c is the speed of light. In GPRMAX2D the equality is used in 3.1 to determine Δt from Δx and Δy . As is evident from 3.1 small values of Δx and Δy result in small values for Δt which means more iterations in order to reach a given simulated time. However, it is important to note that the smaller the values of $\Delta x, \Delta y$ and Δt are the more accurate your model will be.

#time_step_stability_factor:

With the command `time_step_stability_factor:` you can alter the value of the time step Δt calculated by GPRMAX2D. However, the new value should be within the allowable range for stability determined by equation 3.1. As it was mentioned above, GPRMAX2D uses the equality in equation 3.1 hence the maximum permissible time step. If a smaller time step is required then the command

```
#time_step_stability_factor: f1
```

allows you to accomplish this. The parameter `f1` can take values $0 < f1 \leq 1$. Then the actual time step that GPRMAX2D will use will be $f1 \times \Delta t$ (remember Δt is the one calculated using equality in 3.1).

#time_window:

The command `#time_window:` should be used to specify the total required simulated time. Its syntax is

```
#time_window: f1
```

or

```
#time_window: i1
```

In the first case the `f1` parameter determines the required simulated time in seconds. For example, if you want to simulate a GPR trace of 20 nanoseconds then

```
#time_window: 20e-9
```

can be used. GPRMAX2D will perform the necessary number of iterations in order to reach the required simulated time. Alternatively, if the command is specified with an `i1` GPRMAX2D will interpret this value as a specified total iteration number. Hence the command

```
#time_window: 100
```

means that 100 iterations will be performed. The number of iterations and the total simulated time window are related by

$$(3.2) \quad t_w = \Delta t \times N_{it}$$

where t_w is the time window (seconds), Δt the time step and N_{it} the number of iterations¹.

#number_of_media:

The command `#number_of_media` has to be used when your model requires more than a total of 10 **different** media to be used. Initially GPRMAX2D allocates space to store the constitutive parameters for 10 media (`free_space`, `pec` and 8 user defined). This space is usually adequate for most models but in case more media are required then you should issue the command

```
#number_of_media: i1
```

¹GPRMAX2D converts the specified time window in seconds to a number of iterations internally. That is achieved using equation 3.2. An integer value is obtained by rounding the result of the division to the nearest integer.

where the parameter `i1` should be greater than 10 and at least equal to the number of different media you require. There is no problem in allocating space for even a 1000 different media² however, valuable computer memory can be wasted. If GPRMAX2D requires more space to store media parameters will terminate execution issuing an appropriate error message.

#media_file:

With the command `#media_file:` you can specify the location and filename of a file containing the description of constitutive parameters of frequently used media. Hence, by storing these parameters in such a file you do not have to specify them in every model's input file you want to use them. The syntax of the command is

```
#media_file: file1
```

where the parameter `file1` is the filename of the media file (including the path if necessary). The structure of a media file is described in Appendix A.

#geometry_file:

With the command `#geometry_file:` you can specify a file in which information about the model's geometry is stored in binary format. This information can be used to create an image of the model and check if it is properly constructed. The syntax of the command is

```
#geometry_file: file1
```

The parameter `file1` is the filename of the geometry file. For example the command

```
#geometry_file: model.geo
```

will instruct GPRMAX2D to store information about the geometry of the model in the file `model.geo`. The structure of the geometry file is described in Chapter 6.

#messages:

Using the command `#messages:` in your input file you can partially control the amount of information displayed on the screen at run time. The syntax of the command is:

```
#messages: c1
```

The parameter `c1` can be either `y` (yes) or `n` (no) which turns *on* or *off* the messages on the screen. The default value is `n`. When messages are *on*, GPRMAX2D will display on the screen the translation of space and time values to cell coordinates and iteration number integer values respectively. This information can be useful for error checking.

#nips_number:

The command `#nips_number:` should be included in your input file **only** when GPRMAX2D has requested its use. The syntax of the command is

```
#nips_number: i1
```

where the parameter `i1` is an integer number that GPRMAX2D will advise you to use. This command controls the size of arrays used to store important information about the model. Although

²The actual limit is the highest 2 byte integer value which can be stored in the computer around 32768

the size of these arrays can be calculated internally, the resulting number from such a calculation which takes into consideration all possible required space is usually rather larger than the one which is actually needed in most cases³. Hence, a smaller number is used in order to conserve computer memory. When GPRMAX2D detects that more space is needed it issues an error message and prompts you to use this command in the input file which will set aside more space for allocation.

3.3.2 ABC related commands

The default values for the ABCs used in GPRMAX2D should be adequate for most simulations and are tuned to deliver a good performance without introducing any instabilities. It should be noted however that a characteristic of the formulation of local ABCs such as the one used in GPRMAX2D is prone to instabilities especially when high orders are used. Hence, the default settings in GPRMAX2D should not be altered except if you are familiar with the formulation of the Higdon ABC used in GPRMAX2D. Details about the Higdon ABC can be found in [1].

In addition to local ABCs - from Version 2.0 - GPRMAX2D employs the more powerful Perfectly Matched Layer (PML) ABC. The PML works by surrounding the simulation space with a layer of non physical absorbing material which greatly absorbs all electromagnetic waves and performs very well irrespective of the angle of incidence or the frequency of the impinging pulse. In order to use the PML boundary condition instead of the default Higdon ABC the command

```
#abc_type: pml
```

has to be used. The command #abd_type: higdon sets the ABC back to its default setting. At this release of GPRMAX2D/3D the user can not influence the parameters of the PML which are determined automatically for optimum performance. However, the user can define the thickness (or depth) of the PML using the command

```
#pml_layers: il
```

where `il` is the number of Yee cells that the PML will occupy. The default value is 8 cells. The more cells are used the better the performance of the PML but the computational resources increase substantially when deep PML layers are employed. Further, there are two important issues on using the PML:

- ☞ The PML layers are part of the model's geometry. However, the fields in the PML layers are of **NO INTEREST** and **IT IS WRONG TO USE THEM IN CALCULATIONS**. Therefore, **DO NOT** place sources or receivers in areas of the model occupied by a PML. The depth of PML is defined as number of YEE cells and not as a distance in metres.

The following commands can be used either for experimentation with the Higdon ABCs or for further adjustments if required. None of the following commands are required for any GPR models constructed with GPRMAX2D and should be used with caution.

#abc_order:

The command #abc_order: determines the order of the ABC (see [1]). Its syntax is:

```
#abc_order: il
```

³GPRMAX2D performs averaging of media properties at interfaces between regions of different constitutive parameters. Hence more space is required than the actual number of media in the model.

The parameter `i1` can take the values 1, 2 or 3. The default order is 3. The lower the order of the ABC the worse its performance is. For example the command

```
#abc_order: 1
```

will force GPRMAX2D to use a first order Higdon ABC instead of the default third order one. The only benefit of lower order ABCs is the increased stability that they offer.

#abc_stability_factors:

The command `#abc_stability_factors:` allows you to specify small loss terms which are introduced in the ABC in order to increase its stability (see [1]). The syntax of the command is:

```
#abc_stability_factors: f1 f2 f3
```

The higher the parameters `f1`, `f2`, `f3` are the more stable the ABC although by increasing them, its performance will deteriorate. If you use this command the values for the parameters should not, in general, be greater than 0.5. Further, if you have specified a first order ABC only the first parameter `f1` is effectively used and if you have specified a second order one only the first two are effectively used. However, values for all three parameters have to be given.

#abc_optimization_angles:

The command `#abc_optimization_angles:` can be used to optimize the ABC for specific angles of incidence. If you anticipate that most of the reflected energy will impinge on the ABC at specific angles you can optimize its performance for these particular angles. Its syntax is:

```
#abc_optimization_angles: f1 f2 f3
```

The parameters `f1`, `f2`, `f3` are the angles of optimization in degrees ($\theta \leq 90^\circ$). For example

```
#abc_optimization_angles: 0.0 45.0 65.0
```

will optimize the ABC for normal, 45° and 65° incidence. It is recommended that you do not alter the default GPRMAX2D parameters.

#abc_mixing_parameters:

The command `#abc_mixing_parameters:` can be used to change the discretization of the temporal and partial derivatives in the Higdon ABC used by GPRMAX2D (see [1]). The default discretization is the most commonly used “box” scheme. The syntax of the command is:

```
#abc_mixing_parameters: f1 f2 f3 f4 f5 f6
```

The parameters `f1` and `f2` are the mixing parameters for the first order Higdon, the `f3` and `f4` are used with the ones defined for the first order to determine the ABC coefficients for the second order and similarly the `f5` and `f6` are used with all the previous ones to determine the parameters of the third order ABC. The default values for the “box” scheme are

```
#abc_mixing_parameters: 0.5 0.5 0.5 0.5 0.5 0.5
```

It is strongly recommended that you not change these default values unless you are familiar with the formulation of the Higdon ABC.

3.3.3 Media and Object construction commands

The commands which are available in GPRMAX2D to facilitate the introduction of different media in the model and the construction of specific objects are discussed in this section. GPRMAX2D, after the processing of general commands, sets up a model of the required size initialized to simulate free space (air). By using the commands in this section you can introduce other media and objects in the model.

It is important to note that free space and perfect conductors are already defined internally and you do not have to redefine these two media. Thus the key words `free_space` and `pec` which describe air and perfect conductors, respectively, are reserved and can be used directly without any definition of constitutive parameters.

For any other media their parameters have to be made known to GPRMAX2D by using either or both

- a **media file** defined by the `#media_file:` command which contains the definitions of various frequently used media (see Appendix A).
- a **#medium:** command in the input file.

#medium:

With the `#medium:` command you can introduce into the model a set of constitutive parameters describing a given medium. The syntax of the command is

```
#medium: f1 f2 f3 f4 f5 f6 str1
```

The parameters of the command are

- `f1` the DC (static) relative permittivity of the medium ϵ_{rs}
- `f2` the relative permittivity at theoretically infinite frequency $\epsilon_{r\infty}$
- `f3` the relaxation time of the medium τ (seconds)
- `f4` the DC (static) conductivity of the medium σ (Siemens/metre)
- `f5` the relative permeability of the medium μ_r
- `f6` the magnetic conductivity of the medium σ^*
- `str1` a string characterising the medium (medium identifier)

Since GPRMAX2D has the ability to model a Debye dielectric medium the relative permittivity is described by the Debye formula (see Chapter 2 and [1])

$$\epsilon = \epsilon_{\infty} + \frac{\epsilon_s - \epsilon_{\infty}}{1 + j\omega\tau}$$

If you do not want a debye medium you can set `f3` (i.e. τ) to zero. In such a case, GPRMAX2D will use only the values specified in `f1` and `f4` to describe the dielectric properties. If a medium is non-magnetic you can set `f5` to 1.0 and `f6` to zero.

It is important to remember that the value of τ should always be **higher** than the time step Δt used in the model. GPRMAX2D checks and verifies that this condition holds for all Debye media in the model. If it does not then GPRMAX2D will exit raising an error.

The `str1` parameter should be a meaningful identifier for the medium. For example the command

```
#medium: 3.0 0.0 0.0 0.01 1.0 0.0 my_sand
```

introduces a medium referenced by the identifier `my_sand` which has a relative permittivity (frequency independent) $\epsilon_r = 3$ and conductivity $\sigma = 0.01$ (Siemens/metre); and is non-magnetic. The command

```
#medium: 82.3 5.5 10.9e-12 0.0 1.0 0.0 water
```

introduces a medium with the electric parameters of water at 15° C, which is described by the Debye formula.

An alternative or complimentary way of introducing media parameters in GPRMAX2D is to use a media file . The name and location of this file has to be supplied by the `#media_file:` command in order for GPRMAX2D to be able to use the information stored in it. In this file the user can save the parameters of frequently used media and use them in GPRMAX2D by their identifiers as the internally specified `free_space` and `pec`. If a medium has been already defined in a media file it does not have to be declared in GPRMAX2D with a `#medium:` command. However, all identifiers used in the media file and if any in `#medium:` commands **must** be unique. Otherwise GPRMAX2D will issue an error message and terminate execution.

#box:

With the command `#box:` you can introduce a rectangle of specific properties in the model. The syntax of the command is

```
#box: f1 f2 f3 f4 str1
```

the parameters `f1 f2` are the lower left (x,y) coordinates of the rectangle in metres. Similarly, the `f3 f4` are the upper right (x,y) coordinates of the rectangle. The parameter `str1` is a medium identifier defined either with a `#medium:` command or in a media file currently in use. Further, the identifiers `free_space` or `pec` can be used if the rectangle is to represent a free space region or a perfect conductor. For example the command

```
#box: 0.25 0.25 0.75 0.75 my_sand
```

introduces into the model a rectangle which is 0.5×0.5 metres in size and sets its properties to the ones defined to represent `my_sand`. Changing the above command to

```
#box: 0.25 0.25 0.75 0.75 pec
```

will introduce a perfect conducting rectangle of the same size as above. Note that `pec` does not have to be defined. The size of the rectangle can not exceed the size of the model as defined by the `#domain:` command

#cylinder:

With the command `#cylinder:` you can introduce a circular disk in the 2D model (simulating an infinite cylinder). The syntax of the command is:

```
#cylinder: f1 f2 f3 str1
```

The parameters `f1,f2` are the (x,y) coordinates of the centre of the circular disk in metres. The parameter `f3` is its radius r in metres and `str1` is a medium identifier (see `#box:`). Hence the command

```
#cylinder: 0.5 0.5 0.2 free_space
```

introduces in the 2D model a circular disk at the 0.5,0.5 coordinates, of radius $r = 0.2$ metres having the parameters of free space.

#x_segment:

With the command `#x_segment`: you can introduce a circular segment in the 2D model (simulating an infinite segment of a cylinder). The syntax of the command is:

```
#x_segment: f1 f2 f3 f4 f5 str1
```

The parameters `f1,f2` are the (x,y) coordinates of the centre of the circular disk of which a segment in the x direction is going to be constructed. These values are all in metres. The parameters `f3` and `f4` are the x_{low} and x_{high} coordinates of the extent of the segment along the x axis. The parameter `f5` is the radius r in metres of the circular disk on which the segment calculation is based and `str1` is a medium identifier (see `#box`:). Hence the command

```
#x_segment: 0.5 0.5 0.3 0.5 0.2 pec
```

introduces in the 2D model a segment of a circular disk at the 0.5,0.5 coordinates, with a radius $r = 0.2$ metres having an extent from 0.3 to 0.5 along the x axis and the parameters of a perfect conductor. In other words it creates a semicircular disk.

#y_segment:

With the command `#y_segment`: you can introduce a circular segment in the 2D model (simulating an infinite segment of a cylinder). The syntax of the command is:

```
#y_segment: f1 f2 f3 f4 f5 str1
```

The parameters `f1,f2` are the (x,y) coordinates of the centre of the circular disk of which a segment in the y direction is going to be constructed. These values are all in metres. The parameters `f3` and `f4` are the y_{low} and y_{high} coordinates of the extent of the segment along the y axis. The parameter `f5` is the radius r in metres of the circular disk on which the segment calculation is based and `str1` is a medium identifier (see `#box`:). Hence the command

```
#y_segment: 0.5 0.5 0.3 0.5 0.2 pec
```

introduces in the 2D model a segment of a circular disk at the 0.5,0.5 coordinates, with a radius $r = 0.2$ metres having an extent from 0.3 to 0.5 along the y axis and the parameters of a perfect conductor. In other words it creates a semicircular disk.

#triangle:

With the command `#triangle`: you can introduce a triangular patch with specific parameters in the 2D model. Its syntax is:

```
#triangle: f1 f2 f3 f4 f5 f6 str1
```

The three apexes of the triangular patch are specified in pairs of (x,y) coordinates in metres as:

- `f1 f2`

- f3 f4
- f5 f6

The parameter `str1` is a predefined medium identifier. For example the command

```
#triangle: 0.2 0.2 0.2 0.5 0.5 0.2 my_sand
```

introduces a right angle triangle with the right angle at the lower left side of the model (0.2,0.2).

A note on object construction

It is important to note that GPRMAX2D process all object construction commands in the order they appear in the input file. Thus, model space allocated to a specific medium using for example the `#box:` command can be reallocated to another medium using the same or any other object construction command. Hence, the model's space can be regarded as a canvas in which objects are introduced and one can be overlaid on top of the other overwriting its properties in order to produce the desired geometry. For example the sequence of commands

```
#box: 0.0 0.0 1.0 1.0 concrete
#cylinder: 0.5 0.6 0.1 pec
```

can be used to introduce a perfectly conducting bar of $r = 10$ cm inside a concrete slab. If you reverse the order of the commands, the bar will disappear from the model.

Objects which are introduced by a `#box:` command have to be contained within the model. However, the coordinates and radius which can be used in a `#cylinder:` and `#triangle:` commands can be out of the range of the model. In such a case only the part of the model which intersects with these objects will be allocated with the desired properties.

3.3.4 Excitation and output commands

Starting with GPRMAX2D Version 2.0, before we start an analysis we need to specify the characteristics of at least one source using the command

```
#line_source: f1 f2 str1 str2
```

where `f1` and `f2` are the amplitude in amperes of the line source's current and the frequency in Hertz of the source's excitation waveform respectively. The parameter `str1` controls the type of the excitation waveform. The available choices for `str1` are:

- `cont_sine` which is a continuous sine waveform at the specified frequency. In order to avoid introducing noise in the calculation the waveform's amplitude is modulated for the first cycle of the sine wave (ramp excitation).
- `sine` which is a single cycle of a sine waveform at the specified frequency.
- `gaussian` which is a gaussian waveform
- `ricker` which is the first derivative of the gaussian waveform
- `user` which can be used to introduce into the model User defined excitation functions.

In Appendix B the formulas used to calculate these waveforms are given.

The parameter `str2` is a user supplied ID which will subsequently be used to relate the specification of that source with the point of its application in the model.

For example,

```
#line_source: 1.0 600e6 ricker MySource
```

specifies a line source with the ID `MySource` and a frequency of 600 MHz as having the waveform of a ricker wavelet and amplitude of 1. To use this line source only its ID (`MySource`) would be needed later on.

If the user type has been selected then the command

```
#excitation_file: str1
```

needs to be included in the model. The command `#excitation_file:` allows the user to specify a single ASCII file that contains a list of amplitude values that can be used to excite the model. These values need to be at least equal to the number of iterations that are going to be performed. If they are less than this number then zero values are substituted for the missing values. If more values are specified than needed they are superfluous and are ignored. For example, the command

```
#excitation_file: mysource.dat
```

will read values from the file `mysource.dat` which can be used instead of the values obtained by using one of the GPRMAX's build-in excitation functions to describe the pulse shape of the source in the model.

#analysis: and #end_analysis:

After source types have been introduced then placing sources and output points in GPRMAX2D is accomplished using a pair of new commands `#analysis:` and `#end_analysis:`. The source command `#tx:` and the receiver commands `#rx:` and/or `#rx_box:` are to be placed between these two commands.

The syntax of the command `#analysis:` is

```
#analysis: i1 file1 c1
```

The parameter `i1` is the number of **new runs** of the model. This is similar to what the number of scans was in the old `#scan` command and it means that the model will run again - after resetting all arrays and time to zero - for every single `i1`. The parameter `file1` is the name of the file where **all** the results for this `#analysis:` are going to be stored and the parameter `c1` is a single character either `a` or `b` denoting that the format of the output file (`file1`) will be ASCII or BINARY, respectively.

It is important that an `#analysis:` command is followed - after other source and output controlling commands have been inserted - by the command

```
#end_analysis:
```

which denotes the end of an *analysis section* that was started using an `#analysis:` command. The `#end_analysis:` command has **no** parameters. In your input file there can be any number of `#analysis:` and `#end_analysis:` pair of commands.

#tx:, #rx: and #rx_box:

The commands #tx:, #rx: and #rx_box: are used together in order to introduce a source position (#tx:) and output points (#rx: and #rx_box:).

Any number of #tx: commands could be specified in the model as well as any number of #rx: and #rx_box: commands.

The new syntax of the #tx: command is:

```
#tx: f1 f2 str1 f3 f4
```

The parameters f1 and f2 are the (x,y) coordinates in metres of the source in the model. The parameter str1 is the source ID that has been specified before using a source description command (i.e. #line_source:). The parameter f3 is a delay in the source's initiation. If it is greater than zero then the source will be active after that time delay has passed. The parameter f4 is the time of source removal. If the simulation runs for longer than the source's removal time the source will stop functioning in the model after f4 seconds. If the time of source removal is longer than the simulated time window then the source is active for the complete simulation and the time window value is used instead.

The syntax of the #rx: command is:

```
#rx: f1 f2
```

and the parameters f1 and f2 are the (x,y) coordinates in metres of the output (receiver) point.

The syntax of the #rx_box: command is:

```
#rx_box: f1 f2 f3 f4 f5 f6
```

The parameters f1 and f2 are the lower right (x,y) coordinates of the output region. The f3 and f4 are the higher right (x,y) coordinates of the output region and finally f5 and f6 are the steps which define the number of output points in each direction. The minimum value of f5 is Δx and of f6 is Δy

For example with the commands:

```
#analysis: 1 test1.out a
#tx: 0.5 0.5 MySource 0.0 10e-9
#rx: 0.5 0.7
#rx: 0.5 0.8
#rx: 0.6 0.5
#rx_box: 0.3 0.3 0.5 0.5 0.01 0.01
#end_analysis:
```

GPRMAX2D will run the model once. There is only one source point located at (0.5,0.5) and the excitation with ID MySource - which should have been specified earlier in the file - will be used. The source is active from time 0.0 until simulated time is 10 nanoseconds which could be either the end of the requested time window or earlier than that.

The output will be stored in ASCII format in the file test1.out. The time evolution of electromagnetic fields at the points specified by the three #rx: commands at (0.5,0.7), (0.5,0.8) and (0.6,0.5) will be stored in the output file. In addition to the three points defined with #rx: commands the output will be defined at the points defined by the #rx_box: command (441 points)

Creating scans or running the model more than once!

If `i1` of the `#analysis:` command is greater than 1 then the model will run again to calculate a new step in the current analysis.

In order for that to have any meaning we need to include both or at least either of the commands

```
#tx_steps:
#rx_steps:
```

which will advance the coordinates of any `#tx:` and `#rx:` commands respectively by an increment defined as parameters of the above commands. The definition of `#tx_steps:` is

```
#tx_steps: f1 f2
```

where the parameters `f1` and `f2` are the increments in meters of the x and y coordinates accordingly for all the sources specified using `#tx:` commands. Similarly, the command `#rx_steps:` is defined as:

```
#rx_steps: f1 f2
```

where the parameters `f1` and `f2` are the increments in meters of the x and y coordinates accordingly for all the receivers specified using `#rx:`

For example, the following statements

```
#analysis: 10 test1.out a
#tx: 0.5 0.5 MySource 0.0 10e-9
#rx: 0.6 0.5
#tx_steps: 0.01 0.0
#rx_steps: 0.01 0.0
#end_analysis:
```

will allow the simulation of 10 GPR traces. The `#tx:` for the first trace is at (0.5,0.5) and the `#rx:` is at (0.6,0.5). The remaining traces are collected with a step of 0.01 metres in the x direction for both transmitter and receiver. **IMPORTANT NOTE:** The coordinates of any output points (i.e. receivers) specified using an `#rx_box:` command **do not** get updated using the `#rx_steps:` command and stay the same as defined in the input file throughout out the analysis.

The last command - which is described bellow - that can be inserted between a pair of `#analysis:` and `#end_analysis:` commands is the command `#snapshot:`

#snapshot:

In order to obtain information about the electromagnetic fields within an area of the model at a given time instant you can use the `#snapshot:` command. The syntax of this command is:

```
#snapshot: i1 f1 f2 f3 f4 f5 f6 f7 file1 c1
```

or

```
#snapshot: i1 f1 f2 f3 f4 f5 f6 i2 file1 c1
```

The parameters of the command are:

- `i1` is a counter called the *global source position* which is used to determine the source's position for which the snapshot should be taken. The global source position takes a value between 1 and the number of steps defined as the first parameter in the `#analysis:` command.
- `f1 f2` are the lower left (x,y) coordinates in meters of the rectangular area of the snapshot
- `f3 f4` are the upper right (x,y) coordinates in meters of the rectangular area of the snapshot
- `f5 f6` are the sampling intervals in metres in the x and y direction respectively
- `f7` or `i2` are the snapshot time in seconds (float number) or the iteration number (integer number) respectively which denote the point in time at which the snapshot is to be taken
- `file1` is the filename of the file where the snapshot is to be stored
- `c1` can be `a` or `b` if the format of the snapshot file is to be ASCII or BINARY respectively.

For example the commands:

```
#analysis: 10 test1.out a
#tx: 0.5 0.5 MySource 0.0 10e-9
#rx: 0.6 0.5
#snapshot: 5 0.0 0.0 1.0 1.0 0.1 0.1 3e-9 snap1.out b
#tx_steps: 0.01 0.0
#rx_steps: 0.01 0.0
#end_analysis:
```

will allow us to get a snapshot of the electromagnetic fields in the model at 3 nanoseconds while the model is calculating the **fifth** GPR trace.

Chapter 4

GPRMAX3D input file commands

4.1 General notes on GPRMAX3D commands

Most of the commands that can be used in GPRMAX3D input files are almost the same as the ones described for GPRMAX2D. However, there are some commands which are unique for GPRMAX3D

For the sake of brevity only the syntactical differences of GPRMAX3D commands that perform the same function as in GPRMAX2D will be described here. Therefore, please refer to Chapter 3 for a description of the conventions used to present the parameters of the commands. Further, it should be noted that the same *essential* commands have to be included in your input file in order to run a 3D model as in the case of the 2D one.

The fundamental spatial and temporal discretization steps are denoted as Δx , Δy , Δz and Δt respectively.

4.2 List of all available commands

From Version 2 there are 41 commands available in GPRMAX3D which can be used to give the necessary instructions to the program for the construction of a 3D GPR model. These are:

```
#title:
#domain:
#dx_dy_dz:
#time_step_stability_factor:
#time_window:
#messages:

#number_of_media:
#nips_number:
#media_file:
#geometry_file:
#medium:

#abc_type:
#abc_order:
#abc_stability_factors:
#abc_optimization_angles:
#abc_mixing_parameters:
#pml_layers:
```

```
#box:
#cylinder:
#cylinder_new:
#cylindrical_segment:
#sphere:
#plate:
#edge:
#triangle:
#bowtie:
#thin_wire:
```

```
#analysis:
#end_analysis:
#tx:
#rx:
#rx_box:
#snapshot:
#tx_steps:
#rx_steps:
```

```
#huygens_surface:
#hertzian_dipole:
#voltage_source:
#transmission_line:
#plane_wave:
#excitation_file:
```

There are few commands which were used in previous versions and have been “retired” as they are not functioning anymore. These are:

```
#scan:
#csg:
#extra_tx:
```

There are some new commands that have been added form Version 2 which are:

```
#cylinder_new:
#thin_wire:
```

```
#analysis:
#end_analysis:
#tx_steps:
#rx_steps:
```

```
#huygens_surface:
#hertzian_dipole:
#voltage_source:
#plane_wave:
```

Finally, some older commands have been modified. These are:

```
#tx:
#snapshot:
#transmission_line:
```

4.3 GPRMAX3D command parameters

To aid the presentation of GPRMAX3D commands they have been grouped in four categories:

- ① *General commands* which include the ones used to specify the size and discretization of the model.
- ② *ABC related commands* which allow the customization and optimization of the absorbing boundary conditions.
- ③ *Media and Object construction commands* which are used to introduce different media in the model and construct simple geometric shapes with different constitutive parameters.
- ④ *Excitation and Output commands* which are used to place source and output points in the model.

Most of the commands are optional with the exception of the ones which are necessary in order to construct the simplest model possible. For example, none of the commands of the *Media and Object construction* section are necessary to run a model. However, without specifying any objects in the model GPRMAX3D, simulates a free space (air) region not of particular use for GPR modelling. If you have not specified a command which is essential in order to run a model, for example the size of the model, GPRMAX3D will terminate execution issuing an appropriate error message.

☞ The *essential* commands which represent the minimum set of commands required to run GPRMAX3D are:

- #domain:
- #dx_dy:
- #time_window:
- At least one #analysis: with the corresponding #end_analysis: commands enclosing at least one #tx: and one #rx: and/or #rx_box: commands
- In order for the #tx: command to function properly a source definition command (e.g. Hertzian_dipole) is required as well.

4.3.1 General commands

#title:

The command #title has the same syntax and function as in GPRMAX2D

#domain:

The command #domain: is used to specify the size in metres of the model. Its syntax is:

```
#domain: f1 f2 f3
```

The parameters f1 , f2 and f3 are the size in metres your model in the x , y and z direction respectively.

#dx_dy_dz:

The command `#dx_dy_dz:` is used to specify the discretization of space in the x , y and z directions respectively (i.e. Δx , Δy and Δz). The syntax of the command is:

`#dx_dy_dz: f1 f2 f3`

where $f1$ is the spatial step in the x direction (Δx), $f2$ is the spatial step in the y direction (Δy) and $f3$ is the spatial step in the z direction (Δz). The spatial discretization controls the maximum permissible time step Δt with which the solution advances in time in order to reach the required simulated time window. The relation between Δt and $\Delta x, \Delta y$ and Δz is

$$(4.1) \quad \Delta t \leq \frac{1}{c \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}}$$

where c is the speed of light. In GPRMAX3D the equality is used to determine Δt from Δx , Δy and Δz . As is evident from 4.1 small values of Δx , Δy and Δz result in small values for Δt which means more iterations in order to reach a given simulated time. However, it is important to note that the smaller the values of Δx , Δy , Δz and Δt are the more accurate your model will be.

#time_step_stability_factor:

The use and syntax of this command is the same as the one available in GPRMAX2D. The value of Δt that this command alters in the one calculated by equation 4.1

#time_window:

The purpose and syntax of this command is the same as in GPRMAX2D

#number_of_media:

The use and syntax of this command is as described for GPRMAX2D

#geometry_file:

The use and function of this command is as described for GPRMAX2D

#messages:

The syntax and function of this command is as described for GPRMAX2D

#nips_number:

The command `#nips_number:` should be included in your input file **only** when GPRMAX3D has requested its use. Its syntax is the same as described for GPRMAX2D

4.3.2 ABC related commands

The commands which affect the configuration and performance of the Higdon ABCs in GPRMAX3D have the same function and syntax as described for GPRMAX2D. However, GPRMAX3D employs the recently developed and more powerful PML (perfectly matched layer) ABC. The PML works by surrounding the simulation space with a layer of non physical absorbing material which greatly absorbs all electromagnetic waves and performs very well irrespectively of the angle of incidence or the frequency of the impinging pulse. In order to use the PML boundary condition instead of the default Higdon ABC the command

`#abc_type: pml`

has to be used. The command `#abd_type: higdon` sets the ABC back to its default setting. At this release of GPRMAX3D the user can not influence the parameters of the PML which are determined automatically for optimum performance. However, the user can define the thickness (or depth) of the PML using the command

```
#pml_layers: 11
```

where 11 is the number of Yee cells that the PML will occupy. The default value is 8 cells. The more cells are used the better the performance of the PML but the computational resources increase substantially when deep PML layers are employed. Further, there are two important issues on using the PML:

- ☞ The PML layers are part of the model's geometry. However, the fields in the PML layers are of **NO INTEREST** and **IT IS WRONG TO USE THEM IN CALCULATIONS**. Therefore, **DO NOT** place sources or receivers in areas of the model occupied by a PML. The depth of PML is defined as number of YEE CELLS and not as a distance in metres.
- ☞ **CAUTION:** Currently PML works only for **NON-MAGNETIC media**. Therefore, if your model needs to employ media that are to be terminated by a PML and their magnetic permeability μ must be other the μ_0 then you must use the default Higdon ABC and not the PML. This restriction will be alleviated in a future release of GPRMAX3D

4.3.3 Media and Object construction commands

The commands which are available in GPRMAX3D to facilitate the introduction of different media in the model and the construction of specific objects are discussed in this section. GPRMAX3D, after the processing of general commands, sets up a model of the required size initialized to simulate free space (air). By using the commands in this section you can introduce other media and objects in the model.

It is important to note that as in GPRMAX2D, free space and perfect conductors are already defined in GPRMAX3D and you do not have to redefine these two media. Thus the key words `free_space` and `pec` which describe air and perfect conductors, respectively, are reserved and can be used directly without any definition of constitutive parameters.

For any other media their parameters have to be made known to GPRMAX3D by using either or both

- a **media file** defined by the `#media_file:` command which contains the definitions of various frequently used media (see Appendix A).
- a **#medium:** command in the input file.

#medium:

The `#medium:` command has the same syntax and function as described for GPRMAX2D. Further, the same media files can be used either in GPRMAX2D or GPRMAX3D with no alteration to their structure.

#thin_wire:

With the command `#thin_wire:` you can introduce the properties of a thin wire model in GPRMAX3D. The thin wire should then be used as the medium identifier for an `#edge:` object construction command. The syntax of the command is:


```
#thin_wire: f1 str1
```

The parameter `f1` is the radius r in metres of the wire and must be much smaller than Δl in order to model properly the physics of the thin wire. The parameter `str1` is a medium identifier (ID) not currently in use. A thin wire is considered to be a perfect electric conductor. For example, the command

```
#thin_wire: 0.001 MyWire
```

defines a PEC thin wire of radius $r = 1$ millimetre with the identifier `MyWire`.

#box:

With the command `#box:` you can introduce an orthogonal parallelepiped with specific properties in the model. The syntax of the command is:

```
#box: f1 f2 f3 f4 f5 f6 str1
```

The parameters `f1 f2 f3` are the *lower left* (x,y,z) coordinates of the parallelepiped in metres. Similarly the `f4 f5 f6` are the *upper right* (x,y,z) coordinates of the parallelepiped. The parameter `str1` is a medium identifier defined either with a `#medium:` command or in a media file currently in use. Further, the identifiers `free_space` or `pec` can be used if the parallelepiped is to represent a free space region or a perfect conductor. Apart from the different number of parameters the `#box:` command serves the same purpose in GPRMAX3D as in GPRMAX2D.

#plate:

With the command `#plate:` you can introduce a plate with specific properties in the model. The plate is just a surface of a Yee cell and it can be useful when the modelling of thinner surfaces than a Yee cell are required. The syntax of the command is:

```
#plate: f1 f2 f3 f4 f5 f6 str1
```

The parameters `f1 f2 f3` are the *lower left* (x,y,z) coordinates of the plate in metres. Similarly the `f4 f5 f6` are the *upper right* (x,y,z) coordinates of the plate. The parameter `str1` is a medium identifier defined either with a `#medium:` command or in a media file currently in use. Further, the identifiers `free_space` or `pec` can be used if the plate is to represent a free space region or a perfect conductor. In the `#plate:` command the coordinates should define a surface and not a 3D object as with the `#box:` command. For example, the command

```
#plate: 0.5 0.5 0.5 0.7 0.8 0.5 pec
```

defines a xy oriented plate (i.e. Yee cell surfaces). Note that in the above example the z coordinates are identical. This differs from the `#box` command where the use of such coordinates will be illegal.

#triangle:

With the command `#triangle:` you can introduce a triangular patch with specific properties in the model. The patch is just a triangular surface made as a collection of staircased Yee cells in the same way that is created in GPRMAX2D. The syntax of the command is:

```
#triangle: f1 f2 f3 f4 f5 f6 f7 f8 f9 str1
```

The parameters `f1 f2 f3` are the coordinates of the first apex of the triangle `f4 f5 f6` the coordinates of the second and `f7 f8 f9` the ones of the third. The parameter `str1` is a medium identifier defined either with a `#medium:` command or in a media file currently in use. Further, the identifiers `free_space` or `pec` can be used if the triangular patch is to represent a free space region or a perfect conductor. In the `#triangle:` command the coordinates should define a surface and not a 3D object as with the `#box:` command. For example, the command

```
#triangle: 0.5 0.5 0.5 0.6 0.4 0.5 0.7 0.9 0.5 pec
```

defines a *xy* oriented triangular patch (i.e. Yee cell surfaces). Note that in the above example the *z* coordinates are identical. This differs from the `#box:` command where the use of such coordinates will be illegal.

#bowtie:

With the command `#bowtie:` you can introduce a bowtie antenna with specific properties in the model. The antenna is made up by two triangular patches (the command uses the `#triangle:` command internally). The antenna is defined as follows:

```
#bowtie: c1 c2 f1 f2 f3 f4 f5 str1
```

The parameter `c1` is the direction (i.e. polarization) of the bowtie antenna and can be *x*, *y* or *z*. Similarly, the parameter `c2` denotes the remaining direction so as to define a plane where the antenna lies. The parameters `f1`, `f2` and `f3` are the spatial *x, y, z* coordinates of the antenna's feed point. The parameter `f4` is the length of the antenna's elements (half the total length of the complete bowtie) and the parameter `f5` is the flare angle in degrees. The parameter `str1` is a medium identifier defined either with a `#medium:` command or in a media file currently in use. Further, the identifiers `free_space` or `pec` can be used if the bowtie is to represent a free space region or a perfect conductor. For example, the command

```
#bowtie: x y 0.5 0.5 0.5 0.12 60.0 pec
```

defines a bowtie antenna that lies in the *xy* plane and is polarised in the *x* direction. Further, the antenna has a flare angle of 60 degrees and a total length of 0.24 metres. The antenna's feed point is located at the (0.5, 0.5, 0.5) coordinates. To successfully excite the antenna we need to specify a source at these coordinates which is polarised in the same direction.

#edge:

With the command `#edge:` you can introduce a wire with specific properties in the model. The wire is just an edge of a Yee cell and it can be useful when modelling resistors or thin wires. The syntax of the command is:

```
#plate: f1 f2 f3 f4 f5 f6 str1
```

The parameters `f1 f2 f3` are the starting (*x, y, z*) coordinates of the edge in metres. Similarly the `f4 f5 f6` are the ending (*x, y, z*) coordinates of the edge. The parameter `str1` is a medium identifier defined either with a `#medium:` command or in a media file or using a `#thin_wire:` command currently in use. Further, the identifiers `free_space` or `pec` can be used if the edge is to represent a free space region or a perfect conductor. In the `#edge:` command the coordinates should define a single line and not a 3D object as with the `#box:` command. For example, the command

```
#edge: 0.5 0.5 0.5 0.7 0.5 0.5 pec
```

defines an x directed wire (i.e. Yee cell edges). Observe that the y and z coordinates are identical. This differs from the `#box` command where the use of such coordinates will be illegal.

#cylinder:

With the command `#cylinder:` you can introduce a circular cylinder of finite dimensions in the 3D model. The syntax of the command is:

```
#cylinder: c1 f1 f2 f3 f4 f5 str1
```

The parameter `c1` denotes the direction of the cylinder's axis and it could be either x or y or z . The parameters `f1` and `f2` are the lower and higher coordinates of the cylinder's axis. For example, if the cylinder is oriented in the x direction `f1 f2` are the coordinates $(x1, x2)$. The parameters `f3` and `f4` are the remaining necessary coordinates required to describe the centres of the two circular faces of the cylinder. Hence, for the x oriented cylinder these are y and z respectively. The following list describes all possible combinations:

- for an x oriented cylinder $(f1, f3, f4)$ is $(x1, y, z)$ and $(f2, f3, f4)$ is $(x2, y, z)$
- for an y oriented cylinder $(f3, f1, f4)$ is $(x, y1, z)$ and $(f3, f2, f4)$ is $(x, y2, z)$
- for an z oriented cylinder $(f3, f4, f1)$ is $(x, y, z1)$ and $(f3, f4, f2)$ is $(x, y, z2)$

The parameter `f5` is the radius r in metres of the circular cylinder and `str1` is a medium identifier (see Chapter 3). Hence the command

```
#cylinder: y 0.1 0.8 0.5 0.5 0.1 pec
```

introduces in the 3D model a perfectly conducting circular cylinder with its axis in the y direction having a length of 0.7 metres a radius r equal to 10 cm and with the coordinates of the centres of its two faces being $(0.5, 0.1, 0.5)$ and $(0.5, 0.8, 0.5)$.

#cylinder_new:

With the command `#cylinder_new:` you can introduce a circular cylinder of finite dimensions in the 3D model. The difference of this command with the command `#cylinder:` is that the orientation of the cylinder axis can be arbitrary (i.e. it does not have to be one of the model's cartesian axes). The syntax of the command is:

```
#cylinder_new: f1 f2 f3 f4 f5 f6 f7 str1
```

The parameters `f1, f2` and `f3` are the coordinates of the centre of one of the cylinder's two faces. Similarly, the parameters `f4, f5` and `f6` are the coordinates of the centre of the second face. The parameter `f7` is the radius r in metres of the circular cylinder and `str1` is a medium identifier (see Chapter 3). Hence the command

```
#cylinder_new: 0.5 0.1 0.5 0.5 0.8 0.5 0.1 pec
```

introduces in the 3D model a perfectly conducting circular cylinder - as the above example for the `#cylinder:` command - with its axis in the y direction having a length of 0.7 metres a radius r equal to 10 cm and with the coordinates of the centres of its two faces being $(0.5, 0.1, 0.5)$ and $(0.5, 0.8, 0.5)$.

#cylindrical_segment:

With the command `#cylindrical_segment`: you can introduce a circular cylindrical segment of finite dimensions in the 3D model. The syntax of the command is:

```
#cylindrical_segment: c1 f1 f2 f3 f4 f5 str1 c2 f6 f7
```

The parameter `c1` denotes the direction of the cylinder's axis and it could be either `x` or `y` or `z`. The parameters `f1` and `f2` are the lower and higher coordinates of the cylinder's axis. For example, if the cylinder is oriented in the x direction `f1 f2` are the coordinates $(x1, x2)$. The parameters `f3` and `f4` are the remaining necessary coordinates required to describe the centres of the two circular faces of the cylinder. Hence, for the x oriented cylinder these are y and z respectively. The following list describes all possible combinations:

- for an x oriented segment $(f1, f3, f4)$ is $(x1, y, z)$ and $(f2, f3, f4)$ is $(x2, y, z)$
- for an y oriented segment $(f3, f1, f4)$ is $(x, y1, z)$ and $(f3, f2, f4)$ is $(x, y2, z)$
- for an z oriented segment $(f3, f4, f1)$ is $(x, y, z1)$ and $(f3, f4, f2)$ is $(x, y, z2)$

The parameter `f5` is the radius r in metres of the circular cylinder and `str1` is a medium identifier (see Chapter 3). The parameter `c2` is the direction of the segments variation. The parameters `f6` and `f7` are the coordinates of the starting and ending points of the segment's variation. For example, the command

```
#cylindrical_segment: y 0.1 0.8 0.5 0.5 0.1 pec x 0.45 0.55
```

introduces in the 3D model a perfectly conducting cylindrical segment with its axis in the y direction having a length of 0.7 metres a radius r equal to 10 cm and with the coordinates of the centres of its two faces being $(0.5, 0.1, 0.5)$ and $(0.5, 0.8, 0.5)$. However, only cells which are located between the 0.45 and 0.55 coordinates along the x axis are allocated as a `pec` medium creating the required cylindrical segment.

#sphere:

With the command `#sphere`: you can introduce a spherical object with specific parameters in the 3D model. Its syntax is

```
#sphere: f1 f2 f3 f4 str1
```

The parameters `f1`, `f2` and `f3` are the coordinates of the sphere's centre and `f4` is its radius r in metres. The parameter `str1` is a predefined medium identifier. For example the command

```
#sphere: 0.5 0.5 0.5 0.1 my_sand
```

introduces a sphere of radius $r = 10$ centimetres with its centre at $(0.5, 0.5, 0.5)$ and with the constitutive parameters of `my_sand`.

A note on object construction

It is important to note that GPRMAX3D process all the object construction commands in the order they appear in the input file in the same way GPRMAX2D does. Thus, model's space allocated to a specific medium using for example the `#box`: command can be reallocated to another medium using the same or any other object construction command. Hence, the model's space can be

regarded as a canvas in which objects are introduced and one can be overlaid on top of the other overwriting its properties in order to produce the desired geometry. Note that commands who create surface or line objects like the `#plate:`, `#triangle:`, `#bowtie:` and `#edge:` commands can be similarly used to create complex shapes and configurations.

4.3.4 Excitation and output commands

The commands which can be used to describe the source and output points in a GPRMAX3D model are very similar with the ones used in GPRMAX2D. The only differences are in the `#rx:`, `#tx:` and `#snapshot` commands in GPRMAX3D and the corresponding ones in GPRMAX2D are in the number of parameters that they need.

However, GPRMAX3D offers a much more wider range of excitation procedures than the single `#line_source:` available in GPRMAX2D. The commands that define - or help to define! - sources in GPRMAX3D are:

```
#excitation_file:
#hertzian_dipole:
#voltage_source:
#transmission_line:
#plane_wave:
#huygens_surface:
```

#excitation_file:

The command `#excitation_file:` allows the user to specify a single ASCII file that contains a list of amplitude values that can be used to excite the model. These values need to be at least equal to the number of iterations that are going to be performed. If they are less than this number then zero values are substituted for the missing values. If more values are specified than needed they are superfluous and are ignored. To use this User specified excitation pulse use the `user` option when deciding on the type of excitation in the following source definition commands.

The syntax of the command is very simple:

```
#excitation_file: str1
```

where `str1` is the name of the ASCII file containing the values of the user specified pulse. The format of the file is simply a single column of numbers. For example, the command

```
#excitation_file: mysource.dat
```

will read values from the file `mysource.dat` which can be used instead of the values obtained by using GPRMAX3D build-in excitation functions to describe the pulse shape of the source in the model.

#hertzian_dipole:

The command `#hertzian_dipole:` is used to define the simplest excitation in GPRMAX3D which is to specify a current density term at an electric field location. This will simulate an infinitesimal dipole (it does have a length of Δl). The syntax of the command is:

```
#hertzian_dipole: f1 f2 str1 str2
```

The parameters `f1` and `f2` are the amplitude and frequency of the source's waveform. The parameter `str1` is a waveform type identifier and could be any of the ones available in GPRMAX2D or

the keyword `user` when a User specified waveform is to be employed using an `#excitation_file:` command. The parameter `str2` is a source identifier that will be used in a `#tx:` command to relate this type of source to a location in the model. For example,

```
#hertzian_dipole: 1.0 600e6 ricker MyDipole
```

Introduces in the model the parameters of the current definition of a Hertzian Dipole with unit amplitude and centre frequency of a ricker wavelet of 600 MHz. The command `#hertzian_dipole:` is the complete equivalent of the command `#line_source:` that is used in GPRMAX2D.

#voltage_source:

The command `#voltage_source:` is used to define a voltage source that can be introduced in the position of an electric field component either as a **hard** source (i.e. replacing the electric field component's value) or, as having an internal lumped resistance. The `#voltage_source:` command is mainly useful in exciting GPR antennas when the physical proper of the antenna is included in the model. The syntax of the command is:

```
#voltage_source: f1 f2 str1 f3 str2
```

The parameters `f1` and `f2` are the amplitude and frequency of the source's waveform. The parameter `str1` is a waveform type identifier and could be any of the ones available in GPRMAX2D or the keyword `user` when a User specified waveform is to be employed using an `#excitation_file:` command. The parameter `f3` is the internal resistance R of the voltage source in Ohms (Ω). If `f3` is set to zero then the voltage source is a **hard** source. That means it prescribes exactly what the value of the electric field component would be. If the waveform becomes zero then this source is perfectly reflecting. The parameter `str2` is a source identifier that will be used in a `#tx:` command to relate this type of source to a location in the model. For example,

```
#voltage_source: 1.0 600e6 gaussian 50.0 MyVolt
```

Introduces in the model the parameters of the definition of a voltage source with unit amplitude, centre frequency of 600 MHz and temporal variation of a gaussian pulse. The internal resistance is set to 50 Ω . This source can be referred to in a `#tx:` command using the identifier `MyVolt`

#transmission_line:

The command `#transmission_line:` defines the parameters of a 1D two-wire transmission line that can be used to excite antennas in GPRMAX3D. The syntax of the command **has changed in Version 2** and is

```
#transmission_line: f1 f2 str1 f3 f4 str2
```

The parameters `f1` and `f2` are the amplitude and frequency of the source's waveform. The parameter `str1` is the type of excitation in the line. This can be one of the build-in types like `gaussian`, `sine`, `ricker`, etc. or it could be the type `user` if an excitation file is used.

The parameter `f3` is the length of the transmission line in metres. This can be important if lines of different lengths are employed in order to simulate time delays in excitations. The parameter `f4` is the characteristic impedance Z of the line and finally the parameter `str2` is a source identifier that will be used in a `#tx:` command. For example, the command

```
#transmission_line: 1.0 600e6 gaussian 0.5 200.0 MyLine
```

Defines a transmission line having a length of 0.5 meters and a characteristic impedance $Z = 200$ ohms in which a gaussian pulse will be sourced. The properties of this pulse are: unit amplitude and 600 MHz centre frequency. Finally, it is important to note that a file named `MyLine` (i.e. `str2`) is created where the output of the total and reflected voltages and currents in the line are stored. The use of the `#transmission_line:` command should enable the construction of as faithful reproductions as is possible of real GPR antennas.

#plane_wave:

The command `#plane_wave:` can be used to describe the definition of a plane wave source using the total/scattered field method (see [3]). The plane wave source is to be used - at least on this Version of GPRMAX3D - only in models where the background medium is free space. In order to use a plane wave the model space is partitioned to an inner total field region and an outer scattered field region. In the inner total field region the plane wave is added and exists everywhere while in the outer scattered field region only scattered energy from the objects interrogated by the plane wave propagates. In order to define the plane wave and separate these two regions a `#huygens_surface:` command must be used which defines the surface of the box that contains the total field region.

The syntax of the command is:

```
#plane_wave: f1 f2 str1 f3 f4 f5 str2 str3
```

The parameters `f1` and `f2` are the amplitude and frequency of the source's waveform. The parameter `str1` is the type of excitation in the line - waveform type identifier. This can be one of the build-in types like `gaussian`, `sine`, `ricker`, etc. or it could be the type `user` if an excitation file is used.

The parameters `f3`, `f4` are the angles $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$ in degrees that specify the direction - using a spherical coordinate system with the same origin as the cartesian system used in GPRMAX3D - of the plane wave's propagation vector \mathbf{k} . The parameter `f5` is the angle $0 \leq \psi \leq 2\pi$ that the electric field component of the plane wave makes with the vector $\mathbf{k} \times \hat{z}$. The parameter `str2` is a **huygens surface** identifier (see command `#huygens_surface:`). The plane wave must be associated to a huygens surface in order to be sourced in the model. Finally, the parameter `str3` is a source identifier that will be used in a `#tx:` command.

For example,

```
#plane_wave: 1.0 600e6 gaussian 0.0 90.0 0.0 Huygens1 MyPlaneWave
```

Introduces into the model the description of a plane wave with unit amplitude, centre frequency of 600 MHz and temporal variation of a gaussian pulse. This plane wave will be sourced using the `Huygens1` surface and according to the angles of incidence it will be coming from $-\hat{y}$ and be polarised in the x direction. The identifier to turn this source on using a `#tx:` command is `MyPlaneWave`.

#huygens_surface:

The command `#huygens_surface:` must be used in order to be able to source a plane wave. Using this command the model's space is separated in an inner total field region - bounded by the huygens surface - and an outer scattered field region. In essence the command defines a box inside of which is the total field region and outside the scattered field. Special conditions on the surface allows the sourcing of a plane wave. For details about separating the model into total and scattered field regions see [3].

The syntax of the command is:

```
#huygens_surface: f1 f2 f3 f4 f5 f6 str1
```

The parameters `f1 f2 f3` are the *lower left* (x,y,z) coordinates of the huygens surface parallelepiped (i.e. box) in metres. Similarly, the `f4 f5 f6` are the *upper right* (x,y,z) coordinates of the parallelepiped. The parameter `str1` is an identifier for this surface that should be used in the `#plane_wave:` command to connect this surface to a specific plane wave source. For example,

```
#huygens_surface: 0.2 0.2 0.2 0.8 0.8 0.8 MyHuygens
```

defines a box of total field with lower left (x,y,z) coordinates as (0.2,0.2,0.2) and upper right (0.8,0.8,0.8) bounded by a Huygens surface. The identifier `MyHuygens` will be used in a plane wave command.

#analysis: and #end_analysis:

After source types have been introduced then placing sources and output points in GPRMAX3D is accomplished using a pair of new commands `#analysis:` and `#end_analysis:`. The source command `#tx:` and receiver commands `#rx:` and/or `#rx_box:` are to be placed between these two commands.

The syntax of the command `#analysis:` is

```
#analysis: i1 file1 c1
```

The parameter `i1` is the number of **new runs** of the model. This is similar to what the number of scans was in the old `#scan` command and it means that the model will run again - after resetting all arrays and time to zero - for every single `i1`. The parameter `file1` is the name of the file where **all** the results for this `#analysis:` are going to be stored and the parameter `c1` is a single character either `a` or `b` denoting that the format of the output file (`file1`) will be ASCII or BINARY, respectively.

It is important that an `#analysis:` command is followed - after other source and output controlling commands have been inserted - by the command

```
#end_analysis:
```

which denotes the end of an *analysis section* that was started using an `#analysis:` command. The `#end_analysis:` command has **no** parameters. In your input file there can be any number of `#analysis:` and `#end_analysis:` pair of commands.

#tx:, #rx: and #rx_box:

The commands `#tx:`, `#rx:` and `#rx_box:` are used together in order to introduce a source position (`#tx:`) and output points (`#rx:` and `#rx_box:`).

Any number of `#tx:` commands could be specified in the model as well as any number of `#rx:` and/or `#rx_box:` commands.

The new syntax of the `#tx:` command in GPRMAX3D is very similar to the one in GPRMAX2D and is:

```
#tx: c1 f1 f2 f3 str1 f4 f5
```

The parameter `c1` defines the polarization of the source and could be one of `x`, `y` or `z`. The parameters `f1`, `f2` and `f3` are the (x,y,z) coordinates in metres of the source in the model. The

parameter `str1` is the source ID that has been specified before using a source description command (i.e. `#hertzian_dipole:`, `#voltage_source:`, etc.). The parameter `f4` is a delay in the source's initiation. If it is greater than zero then the source will be active after that time delay has passed. The parameter `f5` is the time of source removal. If the simulation runs for longer than the source's removal time the source will stop functioning in the model after `f5` seconds. If the time of source removal is longer than the simulated time window then the source is active for the complete simulation and the time window value is used instead.

Important note: If a plane wave is to be sourced a huygens surface must have been defined as well and used as an identifier in a `#plane_wave:` definition command. However, in order to activate the source this `#plane_wave:` definition must be used in conjunction with a `#tx:` command. In that case the polarization (`x` or `y` or `z`) that has been included in the `#tx:` command as well as the coordinates specified in the `#tx:` command are ignored as the plane wave is sourced on a huygens surface and not on a single point and its polarization is defined by the angle ψ (see `#plane_wave:`)

The syntax of the `#rx:` command is:

```
#rx: f1 f2 f3
```

and the parameters `f1`, `f2` and `f3` are the (x,y,z) coordinates in metres of the output (receiver) point.

The syntax of the `#rx_box:` command is:

```
#rx_box: f1 f2 f3 f4 f5 f6 f7 f8 f9
```

The parameters `f1`, `f2` and `f3` are the lower left hand corner (x,y,z) coordinates of the output volume. The `f4`, `f5` and `f6` are the high right hand corner (x,y,z) coordinates of the output volume and finally `f7`, `f8` and `f9` are the steps which define the number of output points in each direction. The minimum value of `f7` is Δx , of `f8` is Δy and of `f9` is Δz .

For example with the commands:

```
#analysis: 1 test1.out a
#tx: x 0.5 0.5 0.5 MySource 0.0 10e-9
#rx: 0.5 0.5 0.7
#rx: 0.5 0.5 0.8
#rx: 0.6 0.5 0.5
#end_analysis:
```

GPRMAX3D will run the model once. There is only one source point located at $(0.5,0.5,0.5)$ with polarization along the x direction using an excitation with ID `MySource` - which should have been specified earlier in the file - will be used. The source is active from time 0.0 until simulated time has reached the value of 10 nanoseconds which could be either the end of the requested time window or earlier than that.

The output will be stored in ASCII format in the file `test1.out`. The time evolution of electromagnetic fields at the points specified by the three `#rx:` commands at $(0.5,0.5,0.7)$, $(0.5,0.5,0.8)$ and $(0.6,0.5,0.5)$ will be stored in the output file.

Creating scans or running the model more than once!

If `i1` of the `#analysis:` command is greater than 1 (i.e. $1 < i1$) then the model will run again to calculate a new analysis step.

In order for that to have any meaning we need to include either or both of the following commands

```
#tx_steps: f1 f2 f3
#rx_steps: f1 f2 f3
```

These commands will advance the coordinates of any #tx: and #rx: commands respectively by an increment defined as parameters of the above commands. The definition of #tx_steps: is

```
#tx_steps: f1 f2 f3
```

where the parameters f1, f2 and f3 are the increments in meters of the x , y and z coordinates accordingly for all the sources specified using #tx: commands. Similarly, the command #rx_steps: is defined as:

```
#rx_steps: f1 f2 f3
```

where the parameters f1, f2 and f3 are the increments in meters of the x , y and z coordinates accordingly for all the receivers specified using #rx:

For example, the following statements

```
#analysis: 10 test1.out a
#tx: x 0.5 0.5 0.5 MySource 0.0 10e-9
#rx: 0.6 0.5 0.5
#tx_steps: 0.01 0.0 0.0
#rx_steps: 0.01 0.0 0.0
#end_analysis:
```

will allow the simulation of 10 GPR traces. The #tx: for the first trace is at (0.5,0.5,0.5) and the #rx: is at (0.6,0.5,0.5). The remaining traces are collected with a step of 0.01 metres in the x direction for both transmitter and receiver. **IMPORTANT NOTE:** The coordinates of any output points (i.e. receivers) specified using an #rx_box: command **do not** get updated using the #rx_steps: command and stay the same as defined in the input file throughout the analysis.

The last command - which is described below - that can be inserted between a pair of #analysis: and #end_analysis: commands is the command #snapshot:

#snapshot:

In order to obtain information about the electromagnetic fields within an area of the model at a given time instant you can use the #snapshot: command. The syntax of this command is:

```
#snapshot: i1 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 file1 c1
```

or

```
#snapshot: i1 f1 f2 f3 f4 f5 f6 f7 f8 f9 i2 file1 c1
```

The parameters of the command are:

- i1 is a counter called the *global source position* which is used to determine the source's position for which the snapshot should be taken. The global source position takes a value between 1 and the number of steps defined as the first parameter in the #analysis: command.

- `f1`, `f2` and `f3` are the lower left (x,y,z) coordinates in meters of the orthogonal parallelepiped of the snapshot.
- `f4`, `f5` and `f6` are the upper right (x,y,z) coordinates in meters of the orthogonal parallelepiped of the snapshot.
- `f7`, `f8` and `f9` are the sampling intervals in metres in the x , y and z direction respectively
- `f10` or `i2` are the snapshot time in seconds (float number) or the iteration number (integer number) respectively which denote the point in time at which the snapshot is to be taken
- `file1` is the filename of the file where the snapshot is to be stored
- `c1` can be `a` or `b` if the format of the snapshot file is to be ASCII or BINARY respectively.

For example the commands:

```
#analysis: 10 test1.out a
#tx: 0.5 0.5 0.5 MySource 0.0 10e-9
#rx: 0.6 0.5 0.5
#snapshot: 5 0.0 0.0 0.0 1.0 1.0 1.0 0.1 0.1 0.1 3e-9 snap1.out b
#tx_steps: 0.01 0.0 0.0
#rx_steps: 0.01 0.0 0.0
#end_analysis:
```

will save a snapshot of the electromagnetic fields in the model at a simulated time of 3 nanoseconds while the model is calculating the **fifth** GPR trace.

Chapter 5

GPRMAX2D/3D modelling examples

In this chapter some examples of the use of GPRMAX2D/3D for GPR modelling are presented. The examples are simple but illustrate step by step the modelling process.

5.1 Single rebar in concrete. – 2D model

A schematic for this simple problem is illustrated in Figure 5.1. A GPR model is required for a perfectly conducting rebar of radius $r = 25$ millimetres located in a concrete slab of 600 millimetres thickness at a depth of 75 millimetres.

Step 1

First, the constitutive parameters of the media involved should be determined. Since perfect conductors and free space can be modelled without the need to specify their parameters, only the definition of the constitutive parameters for concrete is required.

Let us assume that concrete can be modelled using $\epsilon_r = 6.0$ and $\sigma = 0.01$ (S/m). Moreover, that can be considered to be non-magnetic. Therefore, $\mu_r = 1.0$ and $\sigma^* = 0.0$. Therefore, by including in the input file:

```
#medium: 6.0 0.0 0.0 0.01 1.0 0.0 concrete
```

the medium identifier `concrete` can be used in the model.

Step 2

Determine the source type and excitation frequency. These should be generally known, included in the problem's specification. Let us assume that a `ricker` source can be used to simulate the GPR antenna and a centre frequency of $f = 900$ MHz is required. In order to control the *numerical dispersion* (see Chapter 2) the spatial steps Δx and Δy should be chosen appropriately. According to the *rule of thumb* of Chapter 2 the wavelength of the *highest frequency of interest* in the model should be resolved by at least 10 cells. This highest frequency of interest is **not** the centre frequency of the pulse used for excitation. This can be easily ascertained by examining the amplitude spectrum of the source waveform which is illustrated in Figure 5.2. Examining this Figure it is apparent that the pulse contains a significant amount of energy at higher frequencies than the center one. In general three to four times the centre frequency of the pulse will give a good estimate for the highest frequency which should be used in the calculations. Therefore, let us assume that the highest frequency which we consider significant in the pulse's spectrum is

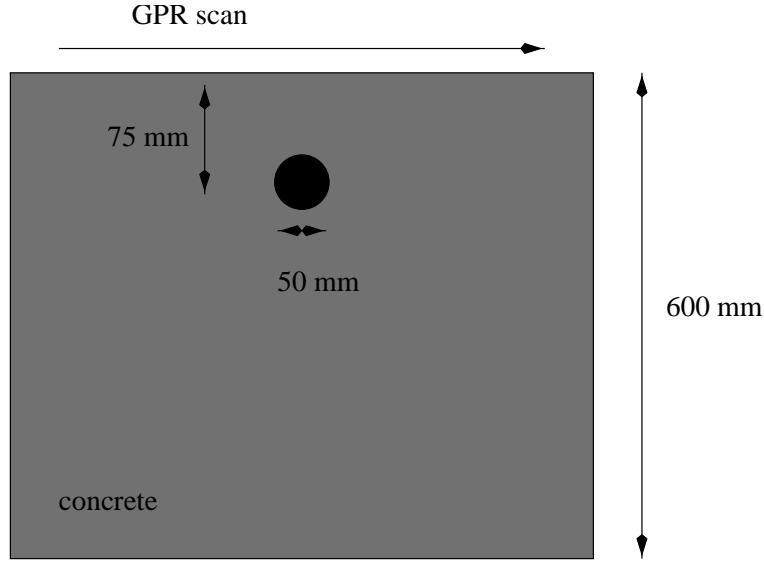


Figure 5.1: Schematic drawing of the rebar in concrete problem.

$f_m = 3 \times f = 2700$ MHz. The wavelength of 2700 MHz in concrete is:

$$(5.1) \quad \lambda = \frac{c}{f_m \sqrt{\epsilon_r}}$$

Hence $\lambda \approx 4.5$ cm. Considering the *rule of thumb* described in Chapter 2 the spatial step can be set to 4.5 millimetres.

Step 3

In order to set the value for the spatial steps first, a decision is made for this model to have $\Delta x = \Delta y = \Delta l$. This is an obvious choice since there is no need to have a smaller step in a particular direction. If we choose as $\Delta l = 4.5$ millimetres then the radius of the rebar will be $r \approx 6$ cells. This will not be considered extremely coarse, but if the value is increased to 10 it will improve the model. With that in mind the spatial steps are set to

$$(5.2) \quad \Delta x = \Delta y = \Delta l = 2.5 \text{ millimetres}$$

The choice of $\Delta l = 2.5$ millimetres is well within the limit described by the *rule of thumb*.

Step 4

Since $\Delta x = \Delta y = 2.5$ millimetres the time step Δt calculated by 3.1 is

$$\Delta t = \frac{\Delta l}{c\sqrt{2}}$$

which gives $\Delta t = 5.8967$ picoseconds (ps). For a time window of 8 nanoseconds the model will require about 1357 iterations.

Step 5

Let us further assume that a 42.5 centimetres long scan is required comprising of 41 GPR traces. In order to determine the size of the model we first decide if any reflections from the termination of the slab will be important. Considering its depth and the time window that is not the case. Hence the concrete slab can be modelled as a half-space. Therefore a 600×300 millimetres model should

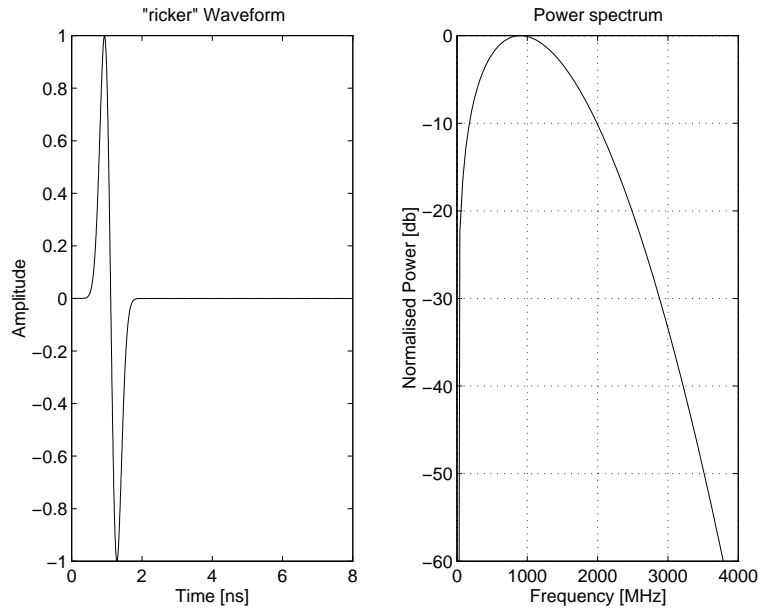


Figure 5.2: Normalized power spectrum and time waveform of the `ricker` excitation function. The centre frequency is 900 MHz.

be adequate. Considering the values of Δx and Δy this translates to 240×120 cells. Therefore in the input file we add:

```
#domain: 0.6 0.3
#dx_dy: 0.0025 0.0025
#time_window: 8.0e-9
```

From the 300 millimetres in the y direction let us use 50 for free space (on top of the slab) hence the slab can be defined by adding to the input file:

```
#box: 0.0 0.0 0.6 0.25 concrete
```

The cylinder is then introduced in order to overwrite with its properties the ones of the slab. Hence we add to the input file:

```
#cylinder: 0.3 0.175 0.025 pec
```

Step 6

A scan of 42.5 cm length consisting of 41 GPR traces needs to be specified. If the scan is centred over the rebar and we assume a separation of 50 millimetres between the transmitter and the receiver the first source should be at (0.075,0.2525) and the first receiver at (0.125,0.2525). The height of both the source and the receiver is set to be at 2.5 millimetres from the interface. The step with which both source and receiver moves in the x direction (scan direction) is 10 millimetres.

From Version 2 a source definition must be included before we introduce the analysis step which will calculate the scan. Therefore, a source definition command is inserted in the file as:

```
#line_source: 1.0 900e6 ricker MyLineSource
```

To compute a scan then the following analysis step is inserted in the input file

```
#analysis: 41 rebar.sca b
#tx: 0.075 0.2525 MyLineSource 0.0 8.0e-9
#rx: 0.125 0.2525
#tx_steps: 0.01 0.0
#rx_steps: 0.01 0.0
#end_analysis:
```

The output file is called `rebar.sca` and will be stored in binary format.

The model is now complete but we can add few more commands to the input file:

```
#title: Model of rebar in concrete
#messages: y
#geometry_file: rebar.geo
```

in order to obtain a geometry file and include a title for the model. The final input file reads like

```
#medium: 6.0 0.0 0.0 0.01 1.0 0.0 concrete
-----
#domain: 0.6 0.3
#dx_dy: 0.0025 0.0025
#time_window: 8.0e-9
-----
#box: 0.0 0.0 0.6 0.25 concrete
#cylinder: 0.3 0.175 0.025 pec
-----
#line_source: 1.0 900e6 ricker MyLineSource
#analysis: 41 rebar.sca b
#tx: 0.075 0.2525 MyLineSource 0.0 8.0e-9
#rx: 0.125 0.2525
#tx_steps: 0.01 0.0
#rx_steps: 0.01 0.0
#end_analysis:
-----
#title: Model of rebar in concrete
#messages: y
#geometry_file: rebar.geo
```

where lines have been drawn to separate the different steps and sections in the input file. Any other comments can be included or if the order of the commands is changed it will not make a difference.

Using GPRMAX2D and the above input file the simulated GPR scan has been obtained. The result is illustrated in Figure 5.3 together with an image representation of the model's space constructed using the information stored in the geometry file.

5.2 Other 2D examples

In the `examples` directory of GPRMAX2D installation disks there four input files which include the modelling instructions for four examples suggested by the Building Research Establishment (BRE), UK. All of the examples involve rebars and hollow tubes located in concrete or sand. Further, most of them include voids located underneath the concrete slab. Schematic drawings of these examples can be found in Figures 5.4 and 5.5. Further, the GPRMAX2D models of these examples are presented in Figures 5.6 and 5.7 as images created in MATLAB using the information stored in the geometry files of the models¹.

¹Before running any of the examples please read the input files.

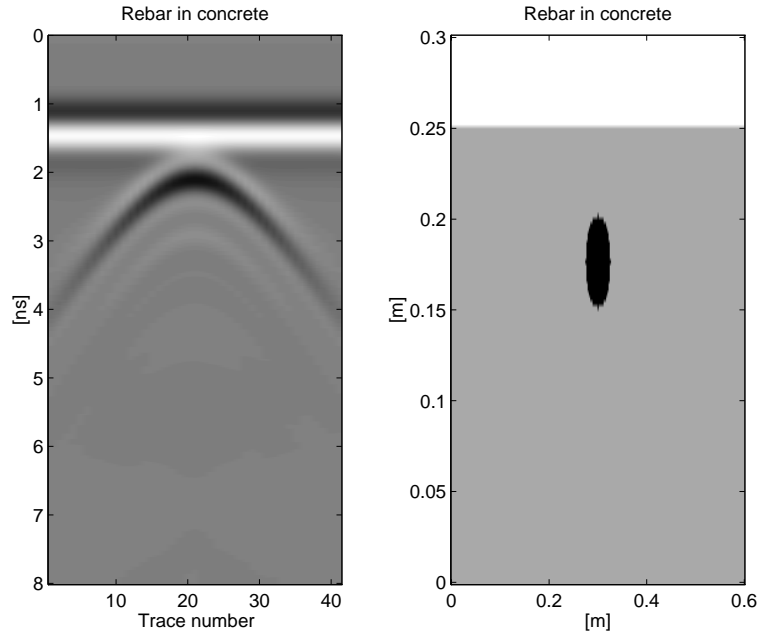


Figure 5.3: Simulated GPR scan by GPRMAX2D (left) and image representation of the geometry of the model (right). Note that the vertical scale is exaggerated.

5.2.1 BRE example #2

The problem that this example represents is a more complicated modelling scenario from the first simple rebar example. A schematic drawing of the problem is presented in Figure 5.4.

Following the same logical steps as in the previous example the fundamental parameters of the model can be determined. The problem at hand includes two different types of material – concrete and wet sand. Particular values for their dielectric properties were not specifically defined. Therefore, it will be assumed that the relative permittivity for the wet sand is $\epsilon_r = 20$ and the parameters of concrete will be the ones used in the previous example. The value of $\epsilon_r = 20$ for wet sand may be rather high. However, is assumed here that it is saturated with water. Using a ricker source at $f = 900$ MHz a simple calculation reveals that a spatial step of $\Delta l = 2.5$ millimetres is close to the limit set by the *rule of thumb*² but it should be small enough for a reasonably accurate model.

The size of the model is set to 2600×800 millimetres which translates to 1040×320 cells. The time window is set to 12 nanoseconds and a 115 GPR traces are to be calculate along a scan line of 2400 millimetres. The input file for this model can thus be created and reads like

```
#medium: 6.0 0.0 0.0 0.01 1.0 0.0 concrete
#medium: 20.0 0.0 0.0 0.1 1.0 0.0 wet_sand
-----
#domain: 2.5 0.5
#dx_dy: 0.0025 0.0025
#time_window: 12e-9
-----
#box: 0.0 0.0 2.5 0.45 wet_sand
-----
```

Use a cylinder of free space and then put
a slab of concrete to cut it in half.

²To calculate the smallest wavelength in wet sand a frequency of $3 \times f$ was used.

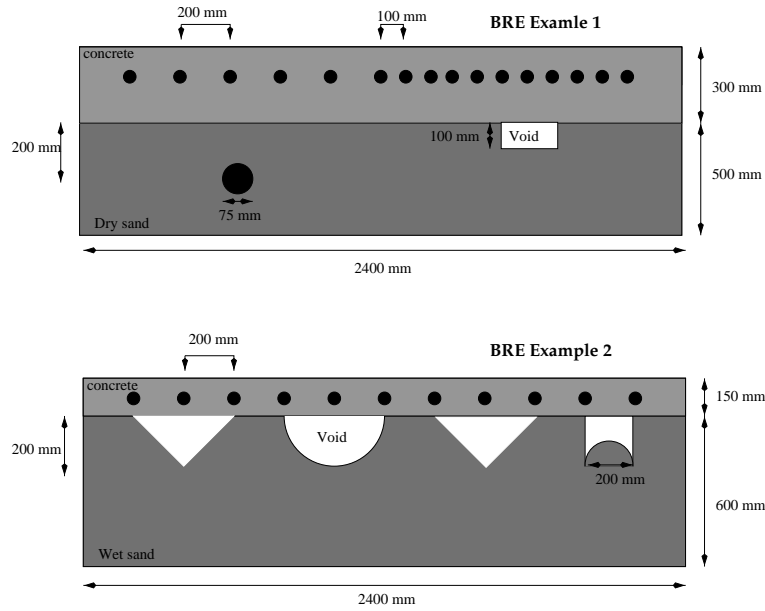


Figure 5.4: Schematic drawing of the BRE examples 1 and 2

```
#cylinder: 1.05 0.3 0.2 free_space
-----
#box: 0.0 0.3 2.5 0.45 concrete
#cylinder: 0.25 0.375 0.0125 pec
#cylinder: 0.45 0.375 0.0125 pec
#cylinder: 0.65 0.375 0.0125 pec
#cylinder: 0.85 0.375 0.0125 pec
#cylinder: 1.05 0.375 0.0125 pec
#cylinder: 1.25 0.375 0.0125 pec
#cylinder: 1.45 0.375 0.0125 pec
#cylinder: 1.65 0.375 0.0125 pec
#cylinder: 1.85 0.375 0.0125 pec
#cylinder: 2.05 0.375 0.0125 pec
#cylinder: 2.25 0.375 0.0125 pec
#triangle: 0.25 0.3 0.65 0.3 0.45 0.1 free_space
#triangle: 1.45 0.3 1.85 0.3 1.65 0.1 free_space
#box: 2.05 0.1 2.25 0.3 free_space
#cylinder: 2.15125 0.10125 0.1 wet_sand
-----
#line_source: 1.0 900e6 ricker MyLineSource
#analysis: 115 bre2.out b
#tx: 0.0875 0.4525 MyLineSource 0.0 12e-9
#rx: 0.1125 0.4525
#tx_steps: 0.02 0.0
#rx_steps: 0.02 0.0
#end_analysis:
-----
#geometry_file: bre2.geo
#title: BRE Model 2
#messages: y
```

It is interesting to examine the order with which the *object construction* commands appear in the input file. For example, in order to construct the semicircular void a `#cylinder:` command is used after the construction of the background rectangle of wet sand. The vertical height of

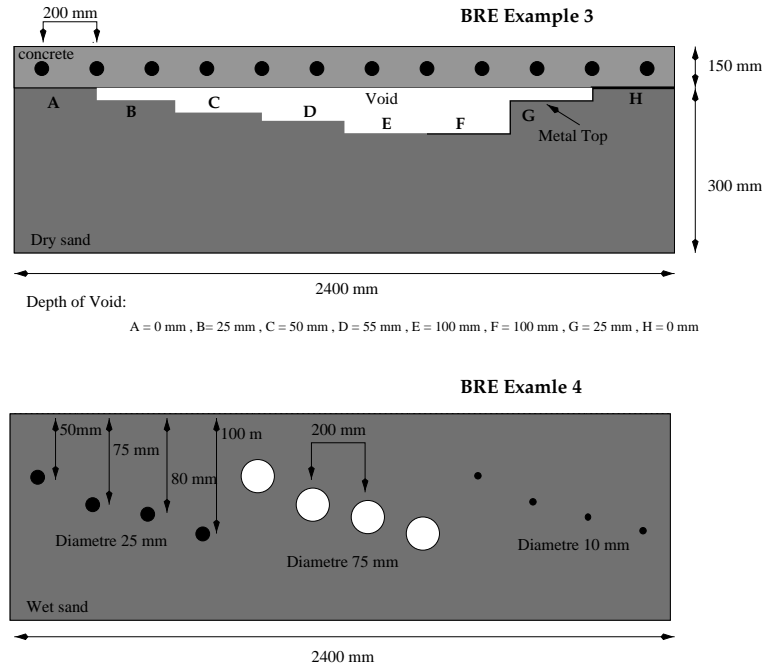


Figure 5.5: Schematic drawing of the BRE examples 3 and 4

the wet sand rectangle can either include the height of the concrete slab or not. In the first case the properties of the wet sand will be replaced by the ones of concrete when the slab will be introduced. In the second case the slab can be placed above the wet sand rectangle overwriting the values of the medium used to initialize the model which is always free space. The first approach is preferable since it ensures that there will not be any nodes left with the properties of free space if an error on the coordinates of the slab is made.

In Figure 5.6, an image representation of the information stored in the geometry file is presented. Observing this image the different size of the model with respect to its schematic drawing is evident. First, the width of the model is 100 millimetres more than in the schematic diagram. This is necessary to ensure that there is enough space between the ABCs and the first source point. Second, there is a part of the model which represents free space and third, the height of the model is less than in the schematic diagram. It is not necessary to include all 600 millimetres of wet sand in the model. Hence, allowing for some space between the targets and the ABCs the rest of the space occupied by wet sand can be ignored. This is done because it is assumed that reflections originate only within the model. If the termination of the wet sand with an other medium is of importance it should be included in the model. The same approach has been used in all four examples and will save a considerable amount of computer memory and run time.

The result of the GPRMAX2D simulation for this example is illustrated in Figure 5.8. The simulated GPR traces are presented using a gray scale image format. To create this image the GPRMAX2D data have been imported into MATLAB.

5.2.2 BRE example #4

The construction of a GPRMAX2D model for the fourth example of the BRE set is more straightforward than the previously described one. A single background medium of wet sand is used and cylinders of different size and type are employed to simulate the rebars and hollow tubes as seen in the schematic drawing of Figure 5.5. It should be noted however, that the size of the 10 millimetres rebars is quite small for a discretization step Δx and Δy of 2.5 millimetres. Better results

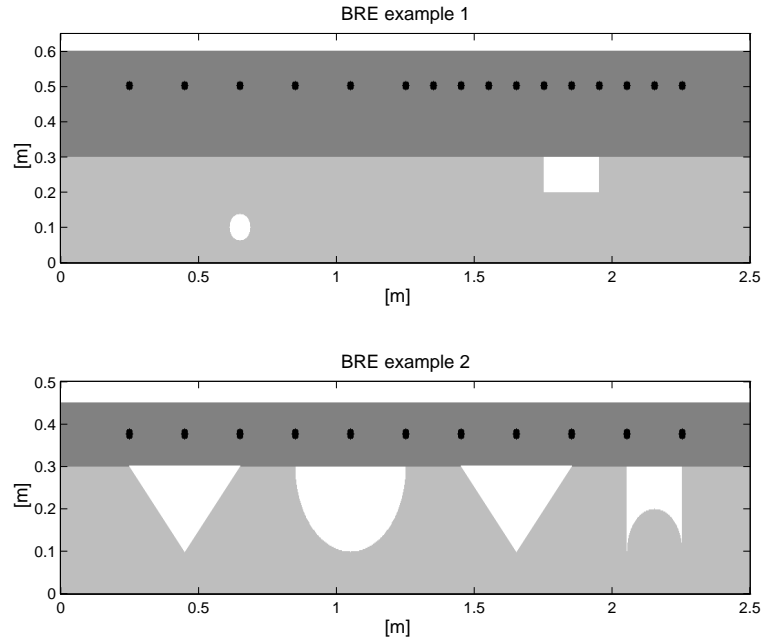


Figure 5.6: Image produced using the information in the geometry file of the BRE examples 1 and 2 (Note vertical scale is exaggerated).

will be obtained using lower values. However, the size of the model will increase substantially. Decreasing Δx and Δy will not result in a different pattern for the response.

The result of the GPRMAX2D simulation for this example is presented in Figure 5.9 using a gray scale image format for the simulated GPR traces.

5.3 Snapshots of the electric field of a horizontal dipole over a dielectric half-space. – 3D model

A simple example of a 3D model is one which can be used to calculate the fields of a hertzian dipole located over a half-space. This problem is of great importance in GPR modelling since the model can provide the necessary information to allow the visualization of the electromagnetic fields as they propagate in the subsurface.

The procedure of creating a 3D model does not differ in principle from the one which should be followed for a 2D model. However, you should always bare in mind that a 3D model requires a rather larger amount of memory than a 2D one. Hence, describing fine details or modelling problems like the previous 2D examples is possible only when significant computational power is available. However, GPRMAX3D does not put any limit on the size of the problem that can be described. The input file for this simple example is:

```
#medium: 3.0 0.0 0.0 0.0 1.0 0.0 sand
-----
#domain: 1.0 1.0 1.0
#dx_dy_dz: 0.02 0.02 0.02
#time_window: 3.35e-9
-----
#box: 0.0 0.0 0.0 1.0 1.0 0.5 sand
-----
#hertzian_dipole: 1.0 600e6 ricker MyDipole
```

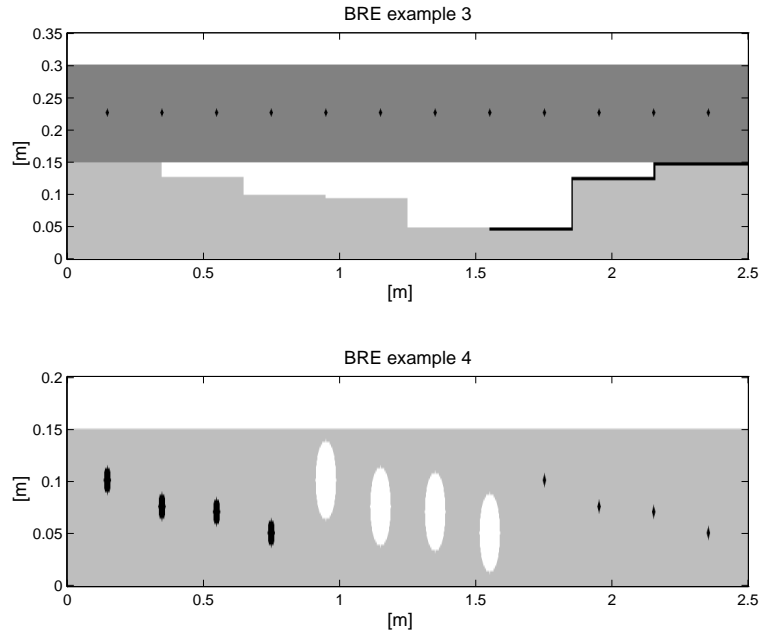


Figure 5.7: Image produced using the information in the geometry file of the BRE examples 3 and 4 (Note vertical scale is exaggerated).

```
#analysis: 1 tx.out a
#tx: x 0.5 0.5 0.5 MyDipole 0.0 3.35e-9
#rx: 0.6 0.6 0.6
#snapshot: 1 0.0 0.0 0.0 1.0 1.0 1.0 0.02 0.02 0.02 3.3e-9 snap3d.out b
#end_analysis:
-----
#messages: y
```

The above input file will produce only a single GPR trace. By changing 1 to a different value in the `analysis:` command more traces can be calculated but then the `tx_sptes:` and `#rx_steps:` must be used to move the source and receiver to different positions in the model.

The `#snapshot:` command as defined above produces a volume of data for each electromagnetic field component. Three slices through this volume of data for the E_x field component (the one parallel to the orientation of the source dipole) are presented in Figure 5.10. Each of these slices corresponds to a constant x , y and z coordinate. The coordinates $(0, 0, 0)$ denote the location of the source. Note that the dipole is located at the interface between air and half-space.

5.4 GPR response of a small void. – 3D model

This final example is taken from [1] and illustrates how a 3D model of a small rectangular void can be designed. The parameters of the model are presented in a schematic drawing in Figure 5.11. A scan comprised of 21 simulated GPR traces is obtained for this target using the following input file in GPRMAX3D

```
#medium: 6.0 0.0 0.0 0.001 1.0 0.0 concrete
-----
#domain: 0.6 1.3 0.65
#dx_dy_dz: 0.01 0.01 0.01
#time_window: 12e-9
-----
```

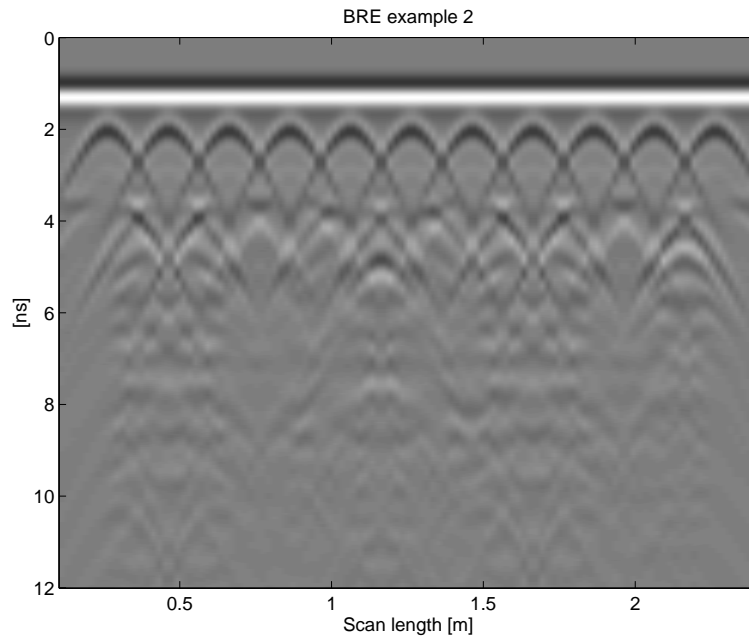


Figure 5.8: Simulated GPR scan of the BRE second example.

```
#box: 0.0 0.0 0.0 0.6 1.3 0.5 concrete
#box: 0.2 0.55 0.1 0.4 0.75 0.3 free_space
-----
#hertzian_dipole: 1.0 600e6 ricker MyDipole
#analysis: 21 void.out b
#tx: x 0.3 0.125 0.55 MyDipole 0.0 12e-9
#rx: 0.3 0.375 0.55
#tx_steps: 0.0 0.04 0.0
#rx_steps: 0.0 0.04 0.0
#end_analysis:
-----
#messages: n
#title: 3D small void
```

The result of the GPRMAX3D simulation is presented in gray scale image format in Figure 5.12 and agrees very well with the result presented in [1].

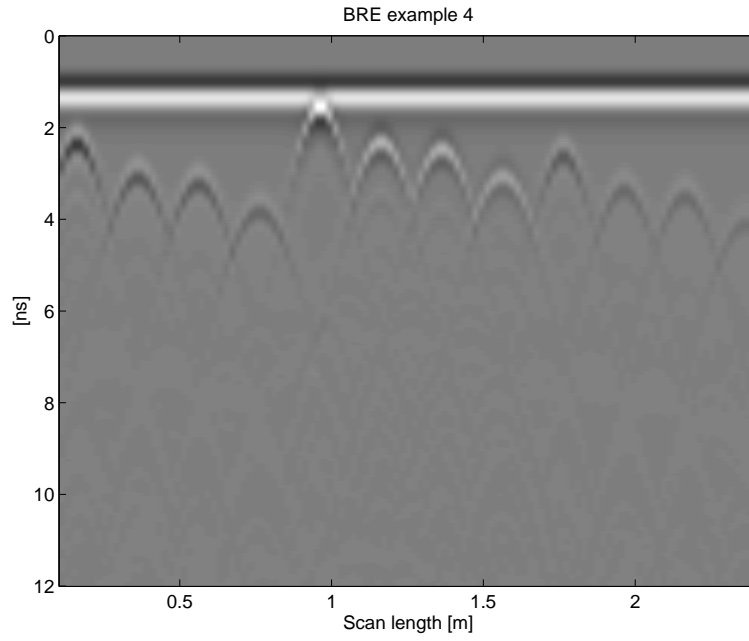


Figure 5.9: Simulated GPR scan of the BRE fourth example.

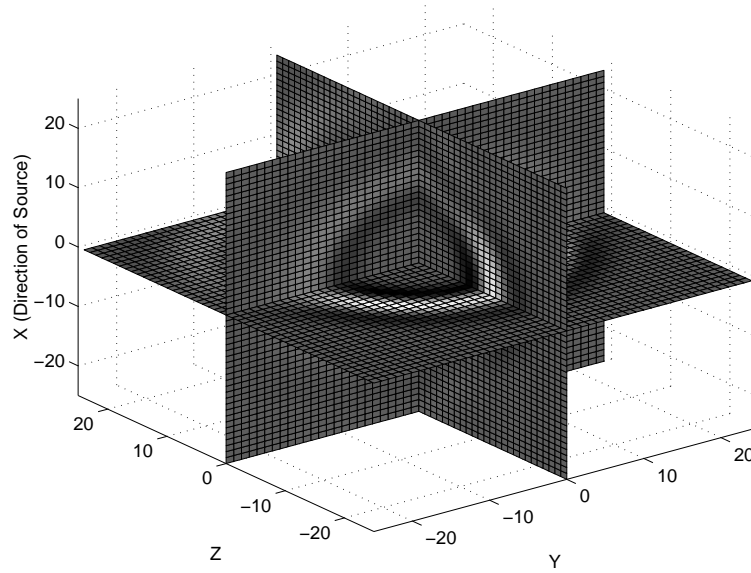


Figure 5.10: Three slices through a volume of data. The amplitude of the E_x field component is represented as a gray scale image. The figure was created in MATLAB from the GPRMAX3D data.

$Tx - Rx = 0.25 \text{ m}$ Yee Cells : [60 130 65] $\epsilon_r = 6$
 $f_s = 600 \text{ MHz}$ $\Delta l = 0.01 \text{ m}$ $\sigma = 0.001 \text{ (S/m)}$
 Num. of traces = 21 $\Delta t = 19.258 \text{ ps}$

$s = 0.8 \text{ m}$
 $h = 0.05 \text{ m}$
 $l = 0.2 \text{ m}$
 $d = 0.2 \text{ m}$

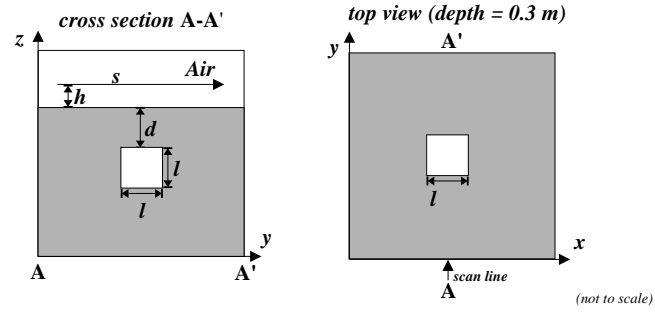


Figure 5.11: Schematic drawing of the 3D model of a small void (after [1])

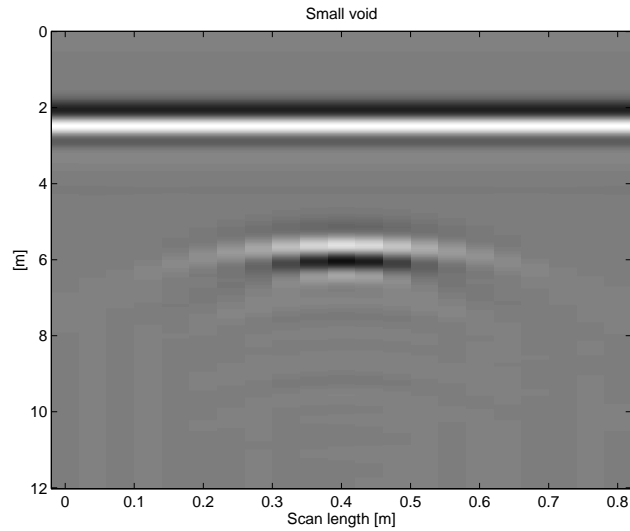


Figure 5.12: GPRMAX3D simulated GPR scan over a small void

Chapter 6

GPRMAX2D/3D Output File formats

This Chapter describes the file formats used by GPRMAX2D and GPRMAX3D to store the results of the simulations. In general, the file formats of GPRMAX2D are similar to the ones used in GPRMAX3D.

6.1 GPRMAX2D/3D Binary format file header

All GPRMAX2D and GPRMAX3D output files stored in binary format have a 16-byte long header part and a data part. The header part is the same in all GPRMAX2D/3D files and is defined as:

- **bytes 0 – 1** contain the integer hexadecimal number 2B67 which identifies the file as a GPRMAX2D/3D one and provides the means to determine whether the low byte (least-significant bits) of multi byte entities is written first or last. If the first byte of the file (**byte 0**) is the hexadecimal number 67 the file is recorded low byte first (LITTLE ENDIAN¹) but if the first byte is the hexadecimal 2B, the file is recorded low byte last (BIG ENDIAN²).
- **bytes 2 – 3** contain a positive integer number which is used to determine the particular type of the file as:
 - **Current Version 2.0**
 - ☞ 20200 GPRMAX2D #analysis: command file
 - ☞ 20204 GPRMAX2D #snapshot: command file
 - ☞ 20205 GPRMAX2D geometry file
 - ☞ 30200 GPRMAX3D #analysis: command file
 - ☞ 30204 GPRMAX3D #snapshot: command file
 - ☞ 30205 GPRMAX3D geometry file
 - **Older Version 1.5**
 - ☞ 20151 GPRMAX2D #tx: command file
 - ☞ 20152 GPRMAX2D #csg: command file
 - ☞ 20153 GPRMAX2D #scan: command file
 - ☞ 20154 GPRMAX2D #snapshot: command file
 - ☞ 20155 GPRMAX2D geometry file
 - ☞ 30151 GPRMAX3D #tx: command file
 - ☞ 30152 GPRMAX3D #csg: command file

¹mainly IBM PCs and some workstations

²mainly UNIX workstations and mainframe computers

- ☞ 30153 GPRMAX3D #scan: command file
- ☞ 30154 GPRMAX3D #snapshot: command file
- ☞ 30155 GPRMAX3D geometry file

- **bytes 4 – 5** contain an integer number which is the value of the number of bytes used for every integer type datum (SWORD) stored in the data part of the file (**NOT IN THE HEADER**).
- **bytes 6 – 7** contain an integer number which is the value of the number of bytes used for every floating point type datum (SREAL) stored in the data part of the file.
- **bytes 8 – 9** contain an integer number which is the number of characters (bytes) used to store the title (TITLELENGTH) in the data part of the file.
- **bytes 10 – 11** contain an integer number which is the number of characters (bytes) used to store any source type specifiers (SOURCELENGTH) in the data part of the file.
- **bytes 12 – 13** contain an integer number which is the number of characters (bytes) used to store any medium identifier strings (MEDIALENGTH) in the data part of the file.
- **bytes 14 – 15** are reserved.

The data part of binary format output files depends on the type of file which in turn is determined by the value of the integer in **bytes 2 – 3**.

It is important to note that in the data part of a GPRMAX2D/3D binary format file

- ☞ All integer numbers denoted by `int` are SWORD bytes long.
- ☞ All floating point numbers denoted by `float` are SREAL bytes long.
- ☞ A character denoted by `char` is stored in one byte.
- ☞ The data part of a binary format file starts at **byte 16**

6.2 GPRMAX2D BINARY file formats

The data part of GPRMAX2D binary format files are described in this section ³.

#analysis: BINARY output file

After the header, information is stored as follows:

- The title of the model in a string of TITLELENGTH bytes
- The number of iterations (NI) which have been performed in an `float`
- The spatial step Δx in a `float`
- The spatial step Δy in a `float`
- The time step Δt in a `float`
- The number of steps in the #analysis: command (NSTEPS) in an `int`
- The step in the x direction with which the source point moves for each trace, normalized by Δx , stored in an `int`

³For data formats of older Versions see their manual for details

- The step in the y direction with which the source point moves for each trace, normalized by Δy , stored in an `int`
- The step in the x direction with which the output point (receiver) moves for each trace, normalized by Δx , stored in an `int`
- The step in the y direction with which the output point (receiver) moves for each trace, normalized by Δy , stored in an `int`
- The number of transmitters specified with `#tx:` commands (NTX) in an `int`
- The number of receivers specified with `#rx:` commands (NRX) in an `int`
- The number of receiver box regions specified with `#rx_box:` commands (NRXBOX) in an `int`
- **For each one of the Transmitters (sources) `#tx:` in the analysis**
 - The x coordinate of the source expressed in cell coordinates in an `int`
 - The y coordinate of the source expressed in cell coordinates in an `int`
 - The source type (i.e. the source ID) in a `string` of SOURCELENGTH bytes
 - The initial source time delay as `float`
 - The time of source removal as `float`
- **For each one of the number of Receivers `#rx` in the analysis**
 - The x coordinate of the receiver expressed in cell coordinates in an `int`
 - The y coordinate of the receiver expressed in cell coordinates in an `int`
- **For each one of the number of Receiver Box regions `#rx_box` in the analysis**
 - The total number of output points for this box output region (NRXBOXOUT) in an `int`
 - The x coordinate of each receiver in this box expressed in cell coordinates in an `int`
 - The y coordinate of each receiver in this box expressed in cell coordinates in an `int`
- The components of the electromagnetic fields E_z , H_x , H_y each stored in a `float` in that order, for every `#rx:` output point (NRX) and then for every output point defined in each `#rx_box:` (NRXBOXOUT), at every time step, for a total of (NI) time steps and a total of (NSTEPS) runs in the analysis.

#snapshot: BINARY output file

After the header, information is stored as follows:

- The title of the model in a `string` of TITLELENGTH bytes
- The number of iterations (NI) which have been performed in an `float`
- The spatial step Δx in a `float`
- The spatial step Δy in a `float`
- The time step Δt in a `float`
- The *global source position* number of the snapshot in an `int`
- The *lower left* x coordinate of the snapshot area expressed in cell coordinates in an `int`

```

For all the X coordinates (Lower -> Higher) {
  For all the Y coordinates (Lower -> Higher) {
    WRITE value
  }
}

```

Figure 6.1: Representation in pseudo-code of the procedure used in GPRMAX2D in order to store the values of field components in a snapshot

- The *lower left* y coordinate of the snapshot area expressed in cell coordinates in an `int`
- The *upper right* x coordinate of the snapshot area expressed in cell coordinates in an `int`
- The *upper right* y coordinate of the snapshot area expressed in cell coordinates in an `int`
- The sampling interval in the x direction, normalized by Δx , stored in an `int`
- The sampling interval in the y direction, normalized by Δy , stored in an `int`
- The time instant when the snapshot has been taken, normalized by Δt (i.e. iteration number), stored in a `float`
- The number of samples in the x direction in an `int`
- The number of samples in the y direction in an `int`
- The values of the E_z electromagnetic field component each stored in a `float` at every x, y sampling point. Figure 6.1 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the H_x electromagnetic field component each stored in a `float` at every x, y sampling point. Figure 6.1 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the H_y electromagnetic field component each stored in a `float` at every x, y sampling point. Figure 6.1 illustrates in pseudo-code the procedure used to store the values of the field component.

Geometry output file

The geometry file is always stored in binary format. After the header, information is stored as follows:

- The title of the model in a `string` of `TITLELENGTH` bytes
- The number of iterations (NI) which have been performed in an `float`
- The spatial step Δx in a `float`
- The spatial step Δy in a `float`
- The time step Δt in a `float`
- The number of cells (NX)⁴ in the x direction in an `int`

⁴Note that the number of cells reported in the geometry file ranges from `[1..NX]`. Internally GPRMAX2D uses the range `[0..NX-1]` according to the coordinate system.

- The number of cells (NY) in the y direction in an `int`
- The number of different media (PMEDIA) present in the model in an `int`
- A list PMEDIA long of medium identifiers. Its medium identifier is stored in a `string` of MEDIALENGTH characters (bytes).
- A cell ID stored in `int` (2 bytes) for every cell in the model. The loop structure shown in Figure 6.1 is used to store these values. Each cell ID corresponds to an entry ([1..PMEDIA]) in the list of different media.

6.3 GPRMAX2D ASCII file formats

The ASCII output files can be edited with any text editor or word processing software and the information stored in them is in general self-explanatory.

6.4 GPRMAX3D BINARY file formats

The file formats of GPRMAX3D differ from GPRMAX2D since information for the polarisation of the source has to be included in addition to the need to store all six components of the electromagnetic fields. The basic structure of the files however, is very similar.

#analysis: BINARY output file

After the header, information is stored as follows:

- The title of the model in a `string` of TITLELENGTH bytes
- The number of iterations (NI) which have been performed in an `float`
- The spatial step Δx in a `float`
- The spatial step Δy in a `float`
- The spatial step Δz in a `float`
- The time step Δt in a `float`
- The number of steps in the `#analysis:` command (NSTEPS) in an `int`
- The step in the x direction with which the source point moves for each trace, normalized by Δx , stored in an `int`
- The step in the y direction with which the source point moves for each trace, normalized by Δy , stored in an `int`
- The step in the z direction with which the source point moves for each trace, normalized by Δz , stored in an `int`
- The step in the x direction with which the output point (receiver) moves for each trace, normalized by Δx , stored in an `int`
- The step in the y direction with which the output point (receiver) moves for each trace, normalized by Δy , stored in an `int`
- The step in the z direction with which the output point (receiver) moves for each trace, normalized by Δz , stored in an `int`
- The number of transmitters specified with `#tx:` commands (NTX) in an `int`

- The number of receivers specified with `#rx:` commands (NRX) in an `int`
- The number of receiver box volume regions specified with `#rx_box:` commands (NRXBOX) in an `int`
- **For each one of the Transmitters (sources) `#tx:` in the analysis**
 - The polarization of the source x , y or z in a `char`
 - The x coordinate of the source expressed in cell coordinates in an `int`
 - The y coordinate of the source expressed in cell coordinates in an `int`
 - The z coordinate of the source expressed in cell coordinates in an `int`
 - The source type (i.e. the source ID) in a `string` of SOURCELENGTH bytes
 - The initial source time delay as `float`
 - The time of source removal as `float`
- **For each one of the number of Receivers `#rx` in the analysis**
 - The x coordinate of the receiver expressed in cell coordinates in an `int`
 - The y coordinate of the receiver expressed in cell coordinates in an `int`
 - The z coordinate of the receiver expressed in cell coordinates in an `int`
- **For each one of the number of Receiver Box regions `#rx_box` in the analysis**
 - The total number of output points for this box output region (NRXBOXOUT) in an `int`
 - The x coordinate of each receiver in this box expressed in cell coordinates in an `int`
 - The y coordinate of each receiver in this box expressed in cell coordinates in an `int`
 - The z coordinate of each receiver in this box expressed in cell coordinates in an `int`
- All six components of the electromagnetic fields E_x , E_y , E_z , H_x , H_y and H_z as well as the currents I_x , I_y and I_z calculated as $(\nabla \times \mathbf{H})_x$, $(\nabla \times \mathbf{H})_y$, $(\nabla \times \mathbf{H})_z$ respectively at electric field component locations and each stored in a `float` in that order, for every output point (NRX) and then for every output point defined in each `#rx_box:` (NRXBOXOUT), at every time step, for a total of (NI) time steps and a total of (NSTEPS) runs in the analysis.

#snapshot: BINARY output file

After the header, information is stored as follows:

- The title of the model in a `string` of TITLELENGTH bytes
- The number of iterations (NI) which have been performed in an `float`
- The spatial step Δx in a `float`
- The spatial step Δy in a `float`
- The spatial step Δz in a `float`
- The time step Δt in a `float`
- The *global source position* number of the snapshot in an `int`
- The *lower left* x coordinate of the snapshot area expressed in cell coordinates in an `int`
- The *lower left* y coordinate of the snapshot area expressed in cell coordinates in an `int`

- The *lower left* z coordinate of the snapshot area expressed in cell coordinates in an `int`
- The *upper right* x coordinate of the snapshot area expressed in cell coordinates in an `int`
- The *upper right* y coordinate of the snapshot area expressed in cell coordinates in an `int`
- The *upper right* z coordinate of the snapshot area expressed in cell coordinates in an `int`
- The sampling interval in the x direction, normalized by Δx , stored in an `int`
- The sampling interval in the y direction, normalized by Δy , stored in an `int`
- The sampling interval in the z direction, normalized by Δz , stored in an `int`
- The time instant when the snapshot has been taken, normalized by Δt (i.e. iteration number), stored in a `float`
- The number of samples in the x direction in an `int`
- The number of samples in the y direction in an `int`
- The number of samples in the z direction in an `int`
- The values of the E_x electromagnetic field component each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the E_y electromagnetic field component each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the E_z electromagnetic field component each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the H_x electromagnetic field component each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the H_y electromagnetic field component each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the H_z electromagnetic field component each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the field component.
- The values of the I_x current calculated at an electric field component location as $(\nabla \times \mathbf{H})_x$ each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the current component.
- The values of the I_y current calculated at an electric field component location as $(\nabla \times \mathbf{H})_y$ each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the current component.
- The values of the I_z current calculated at an electric field component location as $(\nabla \times \mathbf{H})_z$ each stored in a `float` at every x, y, z sampling point. Figure 6.2 illustrates in pseudo-code the procedure used to store the values of the current component.

```

For all the X coordinates (Lower -> Higher) {
  For all the Y coordinates (Lower -> Higher) {
    For all the Z coordinates (Lower -> Higher) {
      WRITE value
    }
  }
}

```

Figure 6.2: Representation in pseudo-code of the procedure used in GPRMAX3D in order to store the values of field components in a snapshot

Geometry output file

The geometry file is always stored in binary format. After the header, information is stored as follows:

- The title of the model in a string of TITLELENGTH bytes
- The number of iterations (NI) which have been performed in an float
- The spatial step Δx in a float
- The spatial step Δy in a float
- The spatial step Δz in a float
- The time step Δt in a float
- The number of cells (NX)⁵ in the x direction in an int
- The number of cells (NY) in the y direction in an int
- The number of cells (NZ) in the z direction in an int
- The number of different media (PMEDIA) present in the model in an int
- A list PMEDIA long of medium identifiers. Its medium identifier is stored in a string of MEDIALENGTH characters (bytes).
- A cell ID stored in int (2 bytes) for every cell in the model. The loop structure shown in Figure 6.2 is used to store these values. Each cell ID corresponds to an entry ([1..PMEDIA]) in the list of different media.
- A component ID stored in int (2 bytes) for the E_x field component. This ID can be used to determine the extend of surfaces objects in GPRMAX3D. The ID corresponds to an entry ([1..PMEDIA]) in the list of different media. The range of this ID is [0..NX-1,0..NY,0..NZ]
- A component ID stored in int (2 bytes) for the E_y field component. This ID can be used to determine the extend of surfaces objects in GPRMAX3D. The ID corresponds to an entry ([1..PMEDIA]) in the list of different media. The range of this ID is [0..NX,0..NY-1,0..NZ]
- A component ID stored in int (2 bytes) for the E_z field component. This ID can be used to determine the extend of surfaces objects in GPRMAX3D. The ID corresponds to an entry ([1..PMEDIA]) in the list of different media. The range of this ID is [0..NX,0..NY,0..NZ-1]

⁵Note that the number of cells reported in the geometry file ranges from [1..NX]. Internally GPRMAX3D uses the range [0..NX-1] according to the coordinate system.

6.5 GPRMAX3D ASCII file formats

The ASCII output files can be edited with any text editor or word processing software and the information stored in them is in general self-explanatory.

Appendix A

GPRMAX2D/3D Media file

The media file has the role of a simple database of frequently used materials. Media files can be used in the same way in both GPRMAX2D and GPRMAX3D. The structure of media files is very simple. Similarly to an input file, the hash character # is reserved.

Each definition of a medium parameters should occupy a single line in the media file. This line **must start** with the hash # character. The parameters of the medium follow in the same order which is required by a #medium: command (see Chapter 3). Hence, a general entry in a media file looks like

```
# f1 f2 f3 f4 f5 f6 str1
```

the parameters are:

- f1 the DC (static) relative permittivity of the medium ϵ_{rs}
- f2 the relative permittivity at theoretically infinite frequency $\epsilon_{r\infty}$
- f3 the relaxation time of the medium τ (seconds)
- f4 the DC (static) conductivity of the medium σ (Siemens/metre)
- f5 the relative permeability of the medium μ_r
- f6 the magnetic conductivity of the medium σ^*
- str1 a string characterizing the medium

The maximum length of the identifier str1 is determined by a constant (MEDIALENGTH) and has been set to 30 characters. No white spaces are allowed in the identifier. Further, every identifier must be unique within a media file.

As an example, a media file constructed for some common materials according to tabulated values [1] of their constitutive parameters at a frequency of 100 MHz is

```
# 80.0 0.0 0.0 0.01 1.0 0.0 distilled_water
# 80.0 0.0 0.0 0.5 1.0 0.0 fresh_water
# 80.0 0.0 0.0 3e4 1.0 0.0 sea_water
# 3.0 0.0 0.0 0.01 1.0 0.0 dry_sand_min
# 5.0 0.0 0.0 0.01 1.0 0.0 dry_sand_max
# 20.0 0.0 0.0 0.1 1.0 0.0 saturated_sand
# 7.0 0.0 0.0 0.5 1.0 0.0 limestone
```

Since GPRMAX2D/3D can model Debye media you can use the definition

```
# 82.3 5.5 10.9e-12 0.0 1.0 0.0 water
```

for water in 15° C. This will take into account the polarization relaxation phenomena associated with water which are described by the Debye equation 2.6 (see Chapter 2).

Appendix B

GPRMAX2D/3D Constants and excitation functions

GPRMAX2D/3D Constants

In GPRMAX2D/3D there are some fundamental constants defined which can not be altered unless the source code is edited. This is however not wise if you are not familiar with both C programming and the FDTD method.

These constants are:

- **Physical**

- Speed of light c defined as 2.9979245×10^8 (metre/second)
- Permittivity of free space ϵ_0 defined as 8.854187×10^{-12} (Farads/metre)
- Permeability of free space μ_0 defined as 1.256637×10^{-6} (Henrys/metre)
- Characteristic impedance of free space Z_0 defined as 376.7303134 Ohms

- **Program related**

- The maximum length of media identifiers (MEDIALENGTH) is 30 characters.
- The maximum length of filenames (FILELENGTH) is 255 characters.
- The maximum length of source types (SOURCELENGTH) is 30 characters.
- The maximum length of title (TITLELENGTH) is 250 characters.
- The maximum number of different media in a model is 32768.
- The maximum number of cells which can be allocated depends solely on the amount of available computer memory.
- The precision of integer numbers is 2 bytes.
- The precision of floating point numbers is 4 bytes.
- The minimum number of cells which can be allocated in any direction is 9
- The maximum number of iterations is 2147483648 which is by far more than will be required. Take note that the number of iterations in binary output files and the iteration counter for binary stored snapshots is converted and stored as a `float`

Excitation functions

The definitions of the source functions which can be used in GPRMAX2D are:

- The `cont_sine` excitation function is defined as:

$$I = R \sin(2\pi ft)$$

and

$$R = \begin{cases} tR_c f & \text{if } R \leq 1, \\ 1 & \text{if } R > 1. \end{cases}$$

where I is the current, R_c is set to 0.25, t is time and f is the frequency.

- The `sine` excitation function is defined as:

$$I = R \sin(2\pi ft)$$

and

$$R = \begin{cases} 1 & \text{if } tf \leq 1, \\ 0 & \text{if } tf > 1. \end{cases}$$

- The `gaussian` excitation function is defined as:

$$I = e^{-\zeta(t-\chi)^2}$$

where $\zeta = 2\pi^2 f^2$ and $\chi = \frac{1}{f}$

- The `ricker` excitation function is defined as:

$$I = -2\zeta \sqrt{e^{1/(2\zeta)}} e^{-\zeta(t-\chi)^2} (t - \chi)$$

which is the first derivative of the gaussian function normalized to unity. The parameters ζ and χ are as defined for the `gaussian` function

Appendix C

MATLAB functions for GPRMAX2D/3D

In the `tools` subdirectory there are simple MATLAB functions which can be used to import modeled data created by GPRMAX2D/3D using the binary format option into MATLAB for further processing and visualization.

NOTICE: These functions are not guaranteed for any particular purpose. The author does not offer any warranties or representations, nor does it accept any liabilities with respect to their function or application. You are free to redistribute these functions but the copyright remains with the author.

In each function there is a small header which explains its purpose, arguments and return values.

Bibliography

- [1] Giannopoulos A., (1997) "The investigation of Transmission-Line Matrix and Finite-Difference Time-Domain Methods for the Forward Problem of Ground Probing Radar", D.Phil thesis, Department of Electronics, University of York, UK.
- [2] Kunz K. S. and Luebbers R. J., (1993) "The Finite Difference Time Domain Method for Electromagnetics", *CRC Press*
- [3] Taflov A., (1995) "Computational Electrodynamics: The Finite-Difference Time-Domain Method", *Artech House*
- [4] Yee K. S., (1966), "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media", *IEEE Transactions on Antennas and Propagation*, Vol. 14, pp. 302-307