gprMax: Open source software to simulate electromagnetic wave propagation for Ground Penetrating Radar

Craig Warren *, Antonios Giannopoulos, Iraklis Giannakis

Institute for Infrastructure and Environment, School of Engineering, The University of Edinburgh, Edinburgh, Scotland, United Kingdom

Abstract

gprMax is open source software that simulates electromagnetic wave propagation, using the Finite-Difference Time-Domain (FDTD) method, for the numerical modelling of Ground Penetrating Radar (GPR). gprMax was originally developed in 1996 when numerical modelling using the FDTD method and, in general, the numerical modelling of GPR were in their infancy. Current computing resources offer the opportunity to build detailed and complex FDTD models of GPR to an extent that was not previously possible. To enable these types of simulations to be more easily realised, and also to facilitate the addition of more advanced features, gprMax has been redeveloped and significantly modernised. The original C-based code has been completely rewritten using a combination of Python and Cython programming languages. Standard and robust file formats have been chosen for geometry and field output files. New advanced modelling features have been added including: an unsplit implementation of higher order Perfectly Matched Layers (PMLs) using a recursive integration approach; diagonally anisotropic materials; dispersive media using multi-pole Debye, Drude or Lorenz expressions; soil modelling using a semi-empirical formulation for dielectric properties and fractals for geometric characteristics; rough surface generation; and the ability to embed complex transducers and targets.

Program summary

Program title: gprMax
Catalogue identifier: AFBG_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AFBG_v1_0.html
Program obtainable from: CPC Program Library, Queen’s University, Belfast, N. Ireland
Licensing provisions: GNU GPL v3
No. of lines in distributed program, including test data, etc.: 627180
No. of bytes in distributed program, including test data, etc.: 26762280
Distribution format: tar.gz
Programming language: Python.
Computer: Any computer with a Python interpreter and a C compiler.
Operating system: Microsoft Windows, Mac OS X, and Linux.
RAM: Problem dependent
Classification: 10.
External routines: Cython[1], h5py[2], matplotlib[3], NumPy[4], mpi4py[5]
Nature of problem: Classical electrodynamics
Solution method: Finite-Difference Time-Domain (FDTD)
Running time: Problem dependent

* This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail addresses: Craig.Warren@ed.ac.uk (C. Warren), A.Giannopoulos@ed.ac.uk (A. Giannopoulos), I.Giannakis@ed.ac.uk (I. Giannakis).

http://dx.doi.org/10.1016/j.cpc.2016.08.020
0010-4655/© 2016 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
Computing power has increased dramatically since gprMax was initially developed—multi-core CPUs and gigabytes of RAM were available from 2005 [14]. Significant modernisations to the code and also added of new advanced features to the software.

The paper is organised as follows: Section 2 provides an overview of the design of the software, the tools that were used, and the principles behind some of the design choices; Section 3 describes the key advanced features that have been developed for modelling GPR; and finally Section 4 gives examples of GPR simulations that take advantage of many of these new features.

2. Software overview

2.1. Design principles and general features

gprMax was developed as cross-platform software for Linux, Microsoft Windows, and later Mac OS X. It was originally written using the C programming language, with the computationally intensive parts—the FDTD solver loops—parallelised using OpenMP [15]. The original design principal was to create a general computational electromagnetic solver, and then build features specifically for modelling GPR onto that core. We continued to use this philosophy for the redesign of gprMax whilst also considering how to facilitate the implementation of new advanced features, and how to lay better foundations for future developments.

We decided that the code should be rewritten in Python—a modern, interpreted language that is intended to be highly readable and extensible. There are advantageous features of Python such as dynamic typing, automatic memory management, and object orientation. However some of these attributes come at a performance cost compared with statically typed languages such as C. For a typical FDTD solver, most of the computational time is spent solving the electromagnetic field update equations. Therefore we focused object orientation and abstraction on the parts of the code that construct the model (prior to the solving), and then used Cython—a superset of Python that generates efficient C source code that can be compiled into extension modules—to write simple methods with minimal decision-making for the FDTD solver. Additionally, Cython supports OpenMP which allowed the FDTD solver to be multi-threaded on machines with multiple CPUs/cores. As an example of this design philosophy, materials have their own class and methods but prior to the solving phase, the update coefficients for the electric and magnetic field equations for each material are stored in simple floating-point NumPy arrays. A NumPy array of integers is used to represent materials and their locations in the computational domain, i.e. the geometry of the model. The integers provide a lookup (index) into the array of the actual material properties/coefficients. Therefore a significant memory saving is made by not having to store material properties/coefficients at every location in the computational domain.

References:


© 2016 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
Listing 1: Input file for a simple 2D GPR simulation of a metal cylinder buried in a lossless dielectric half-space.

```
#domain: 0.240 0.210 0.002
#dx_dy_dz: 0.002 0.002 0.002
#time_window: 3e-9
#material:6010 half_space
#hertzian_dipole: z 0.100 0.170 0
#waveform: ricker 1 1.5e9 my_ricker
#rx: 0.140 0.170 0
```

2.2. User interface, scripting and file formats

gprMax uses a text-based input file in which users specify all of the parameters for a simulation, e.g., model size, discretisation, time window, geometry, materials, and excitation, via pre-defined
commands. We considered developing a CAD-based graphical user interface (GUI) or creating a pure programming interface for

programming; and thirdly we decided the limited resources we had were best concentrated on developing advanced modelling features for GPR within software that could easily interface with other tools. Although a CAD-based GUI is useful for creating single
simulations it becomes increasingly cumbersome for a series of simulations or where simulations contain heterogeneities, e.g. a
model of a soil with stochastically varying electrical properties.

Listing 1 provides an example of an input file for a simple 2D GPR simulation of a metal cylinder buried in a lossless dielectric half-space. Fig. 2 shows the geometry of the model.

1 A B-scan is a GPR image composed of multiple A-scans recorded at different locations.

2 An A-scan is a single time-domain trace/signal from a GPR.

All commands begin with a hash symbol followed by the name of the command, and then a list of associated parameters. In lines 1–2 the size of the computational domain and discretisation of the model are given in x, y, z directions. The model is 2D as the z dimension of the domain is only a single cell. Line 3 specifies the duration of time to simulate, with the time step being calculated automatically at the CFL limit. In line 4 a material is defined which is used to build the half-space. The material has the identifier name half_space, a relative permittivity of six, electric conductivity of zero (S/m), relative permeability of one, and zero magnetic loss (Ω/m). A Hertzian dipole fed with a Ricker waveform with a centre frequency of 1.5 GHz is used as a source (lines 5–6). A receiver is used to record the time histories of the electric and magnetic fields at a specific location for the duration of the simulation. Finally, a box object (used to represent the half-space) and a cylinder object are created. The identifiers half_space and pec\(^4\) refer to the materials that the objects are built from. The order of the objects is important as a layered canvas approach is used, i.e. subsequent

1 All units are in the International System of Units (SI).

2 pec is a built-in material.
objects overwrite the properties of previous objects if they specify the same location. The full syntax of every command can be found in the User Guide (http://docs.gprmax.com).

We have made it easier to create more complex simulations in gprMax by allowing scripting in the input file. This is achieved because blocks of Python code can be written in the input file and are then executed when the file is read. Listing 2 shows a simple example of how a repetitive geometry command can be scripted directly in the input file using a for loop in Python. A PEC cylinder extending from 0 to 100 mm in the z-direction, with y-coordinate 50 mm, and radius 5 mm, is repeated every 20 mm in the x-direction from 20 mm to 160 mm.

Listing 2: Python scripting in an input file

```python
#python:
for x in range(8):
    print('#cylinder: z 0 0.1 {}
0.05 0.005
pec'.format(0.02 + x*0.02))
#end_python:
```

Alongside improvements to the input file we have introduced new file formats for field outputs and geometry information. We wanted to design gprMax to be as flexible as possible and based around robust and standardised formats which would allow users a choice of tools for creating input, and viewing and processing output. We have used HDF5 [19] as the output file format to handle the larger and more complex data sets that are being generated. HDF5 is a robust, portable and extensible format with a number of free readers available. The Visualization Toolkit (VTK) [20] is used for improved handling and viewing of the FDTD geometry meshes. The VTK is an open source system for 3D computer graphics, image processing and visualisation. It also has a number of free readers available such as Paraview (http://www.paraview.org). gprMax allows the user to view geometry information for the entire model domain or any specified sub-volume within the model domain. The geometry information can be requested on a per-cell basis, useful for viewing volumetric objects, or a per-cell-edge basis, which is useful for viewing fine or more complex geometrical features.

### 3. Advanced features for modelling GPR

gprMax contains many powerful and customisable features for modelling GPR. This section focuses on a selection of the new and advanced capabilities that have been developed.

#### 3.1. Library of antenna models

Models of antennas have been included in numerical simulations of GPR intermittently over the past 20 years with varying degrees of realism. Those that have included models of the actual antenna have been mainly of antennas used in academia or for research purposes [21–30]. There has been very limited published work of GPR simulations with models of commercial antennas [31–34]. In fact, many simulations have used a theoretical Hertzian dipole source to represent a real GPR antenna where only far-field behaviour or travel-time information was of interest, or where computational resources were limited. However, advances in computational power, coupled with the desire to investigate quantitative amplitude information from GPR, means there is a need to develop and use detailed 3D FDTD models of realistic GPR antennas in simulations.

gprMax now includes a library with pre-defined models of antennas that behave similarly to commercial antennas. Currently, models of antennas similar to Geophysical Survey Systems, Inc. (GSSI) (http://www.geophysical.com) 1.5 GHz (Model 5100) antenna, and MALA Geoscience (http://www.malags.com/) 1.2 GHz antenna are included. This simplifies the process of adding such intricate structures into a model. Listing 3 demonstrates how a model of a high-frequency GPR antenna, shown in Fig. 3, can be inserted into a simulation without having to be built step-by-step by the user. The antenna model is imported from a library and inserted at a specific location in the computational domain.

Listing 3: Inserting a complex antenna model into an input file

```python
#python:
import from user_libs.antennas import antenna_like_MALA_1200
antenna_like_MALA_1200(0.05, 0.05, 0.05)
#end_python:
```

#### 3.2. Absorbing boundary conditions

With increased research into quantitative amplitude information from GPR, it has become necessary for simulations to have more efficient and better-performing Perfectly Matched Layer (PML) absorbing boundary conditions (ABC). Since 2005 gprMax has featured PML ABCs based on the uniaxial PML (UPML) [35]...
formulation. A PML based on a recursive integration (RI) approach to the complex frequency shifted (CFS) PML [36] has now been developed for grprMax. The implementation is such that a standard UPML, first order CFS-PML, or second order mixed RIPML can now be configured. Additionally, for advanced usage, the parameters of the PML can be customised, which allows the performance of the PML to be better optimised for specific applications. One of the attractions of the RIPML is that it is easily applied as a correction to the electric and magnetic field values after the complete FDTD grid has been updated using the standard FDTD update equations. Moreover, the RIPML is media agnostic so it can be used, without change, to problems involving dispersive and anisotropic materials.

3.3. Materials

Many of the environments where GPR is used are complex, heterogeneous, and contain materials with dispersive properties. Therefore we have focused on developing new features and making improvements to how materials are created and simulated in the software.

3.3.1. Anisotropic materials

gprMax allows anisotropic objects to be modelled in a simulation. Materials such as wood and fibre-reinforced composites, which are often imaged with GPR, can now be more accurately described. This has been achieved by enabling every volumetric geometry object to specify up to three material identifiers. It is therefore possible for every object to have diagonal anisotropy. Listing 4 demonstrates the uniaxial anisotropy of a carbon-fibre-reinforced polymer (CFRP) composite material.

```
1 #material: 40 5.41 1 0 cfrpX
2 #material: 7.5 0.016 1 0 cfrpYZ
3 #box: 0 0 0.1 0.05 cfrpX cfrpYZ
```

Listing 4: Uniaxial anisotropy of a carbon-fibre-reinforced polymer (CFRP) composite material

The material cfrpX is used to define the material properties of the CFRP in the x direction, and the material cfrpYZ for the y and z directions. A box of CFRP is created on line 3, with the object using three identifiers to associate it with its materials properties in the x, y, z directions.

3.3.2. Dispersive materials

gprMax has always included the ability to represent dispersive materials using a single-pole Debye model. Many materials can be adequately represented using this approach for the typical frequency ranges associated with GPR. However, multi-pole Debye, Drude and Lorentz functions are often used to simulate the electric susceptibility of materials such as: water [37], human tissue [38], cold plasma [39], gold [40], and soils [41,42,29]. Electric susceptibility relates the polarisation density to the electric field, and includes both the real and imaginary parts of the complex electric permittivity variation. gprMax now uses a recursive convolution based method to express dispersive properties as apparent current density sources [43]. A major advantage of this implementation is that it creates an inclusive susceptibility function that holds, as special cases, Debye, Drude and Lorentz materials. Listing 5 gives an example of the command to add a 2-pole Debye material that simulates human fatty tissue [38].

```
1 #material: 3 0.026 1 0 fat_tissue
2 #add_dispersion_debye: 2 1.42 13e-12
             ← 1.87 0.65e-9 fatty_tissue
```

Listing 5: A 2-pole Debye material that simulates human fatty tissue

Line 1 defines the basic material properties and in line 2 the #add_dispersion_debye command adds dispersive behaviour to the material based on the Debye formulation. The parameters for the #add_dispersion_debye command define the number of poles, the difference between the DC (static) relative permittivity and the relative permittivity at infinite frequency for the first Debye pole, the relaxation time (seconds) for the first Debye pole, the difference between the DC (static) relative permittivity and the relative permittivity at infinite frequency for the second Debye pole, and the relaxation time (seconds) for the second Debye pole.

3.3.3. Soil models and topography

The inclusion of improved models of soils is important for many GPR simulations. gprMax can now be used to create soils with more realistic dielectric and geometrical properties [44]. A semi-empirical model, initially suggested by [45], is used to describe the dielectric properties of the soil. The model relates the permittivity of the soil to its bulk density, sand particle density, sand fraction, clay fraction and volumetric water fraction. Using this approach, a more realistic soil with a stochastic distribution of the aforementioned parameters can be modelled. The real and imaginary parts of this semi-empirical model can be approximated using a multi-pole Debye function plus a conductive term. This can now be achieved in grprMax using the new dispersive material functionality described in Section 3.3.2. For example, to create a soil with bulk density, \( \rho_b = 2 \) g/cm\(^3\), sand particle density, \( \rho_s = 2.66 \) g/cm\(^3\), sand fraction, \( S = 0.5 \), clay fraction, \( C = 0.5 \), and a volumetric water fraction in the range 0.001–0.25, the command #soil_peplinski: 0.5 0.5 2 2.66 0.001 0.25 soil_properties would be used.

Fractals are scale invariant functions and can be used to express the topography of soils for a wide range of scales with sufficient detail [46]. Fractals can be generated by the convolution of Gaussian noise with the inverse Fourier transform of 1/\( k^2 \), where \( k \) is the wavenumber and \( b \) is a constant related to the fractal dimension [47].

The combination of the Peplinski soil models and the fractal functions can be used to generate a soil model in gprMax with more realistic dielectric and geometrical properties. Listing 6 gives an example of the commands required to generate the soil model shown in Fig. 4. The soil is composed of ten different dispersive materials and features a rough surface.

```
1 #soil_peplinski: 0.5 0.5 2 2.66 0.001
           ← 0.25 soil_properties
2 #fractal_box: 0 0 0.1 0.1 0.07 1.5 1
             ← 1 1 10 soil_properties soil
3 #add_surface_roughness: 0 0 0.07 0.1
             ← 0.1 0.07 1.5 1 0.065 0.075 soil
```

Listing 6: Simulated soil using a Peplinski model, with a rough surface

5 When a material has a dispersive modifier, the relative permittivity should be specified as the relative permittivity at infinite frequency.
Fig. 4. Stochastic distribution of an arbitrarily chosen property of the soil and a rough surface created using fractal correlated noise.

Fig. 5. B-scan of a metal cylinder buried in a homogeneous dielectric half-space.

4. Example GPR simulations

The following three examples demonstrate how simple and more advanced simulations of GPR that can be carried out using gprMax.

4.1. B-scan of a buried cylindrical object

This is an example of a B-scan from a simple 2D GPR simulation of a metal cylinder buried in a lossless dielectric half-space. Listing 7 is the input file required to generate this model.

Listing 7: Input file to generate a B-scan of a buried cylindrical object

Listing 7 is identical to Listing 1 except that to create the B-scan the source and receiver are moved in steps to a new position every time the simulation is run, i.e. for each A-scan. The resulting B-scan is shown in Fig. 5 and is composed of 60 A-scans, i.e. 60 model runs.

4.2. Antenna patterns in a heterogeneous soil

This example shows how to simulate the field patterns of a GPR antenna over a heterogeneous soil.

Listing 8: Input file to generate field patterns of a GPR antenna over a heterogeneous soil

Listing 8 demonstrates using Python scripting within an input file to generate the model. Fig. 6 shows a series of the resulting H-plane field patterns at different observation distances from the antenna. Further research into the characteristics of GPR antennas in lossless and lossy environments can be found in [31,34,48].
4.3. Complex environment

The geometry of the final example model is shown in Fig. 7. The simulation is of a complex environment that can be often be encountered in GPR surveys. It includes a heterogeneous soil with a rough surface and pools of surface water. Grass and roots are simulated, and a model of GPR antenna is included. Listing 9 shows that all of this complexity is achieved using relatively few commands which demonstrates the power and ease of use of the software.

```
#domain: 1 0.208 0.7
#dx_dy_dz: 0.001 0.001 0.001
#time_window: 10e-9
#soil_peplinski: 0.5 0.5 2.0 2.66 0.001
  0.25 soil_properties
#fractal_box: 0 0 0 1 0.208 0.5 1.5 1 1
  1 10 soil_properties soil
#add_surface_roughness: 0 0 0.5 1 0.208
  0.5 1 1 1 0.45 0.55 soil
#add_surface_water: 0 0 0.5 1 0.208
  0.52 soil
#add_grass: 0 0 0.5 1 0.208
  0.5 1 0.4 0.6 300 soil
#python:
  from user_libs.antennas import antenna_like_GSSI_1500
  antenna_like_GSSI_1500(0.5, 0.104, 0.6)
@end_python:
#geometry_view: 0 0 0 1 0.208 0.7 0.001
  0.001 0.001 complex_environment n
```

Listing 9: Input file for a complex environment for GPR

5. Conclusion

Current computing resources offer the possibility to build ever larger and more complex simulations of GPR that have not been possible before. A new version of gprMax has been developed that is open source and written using Python and Cython programming languages. Improvements have been made to existing features of gprMax as well as the addition of new advanced modelling features including: an unsplitted implementation of higher order perfectly matched layers (PMLs) using a recursive integration approach; diagonally anisotropic materials; dispersive media using multipole Debye, Drude or Lorenz expressions; soil modelling using a semi-empirical formulation for dielectric properties and fractals for geometric characteristics; rough surface generation; and the ability to embed complex transducers and targets. A series of example simulations demonstrate some of these features and the ease with which they can be used. The open source principle of the software provides a platform for developers to contribute new ideas and algorithms which will be of future benefit to the GPR research community.

Acknowledgements

This work was supported by The Defence Science and Technology Laboratory (Dstl), UK, and the Engineering and Physical Sciences Research Council (EPSRC), UK (grant no. EP/J501943/1), and benefited from networking activities carried out within the EU funded COST Action TU1208 “Civil Engineering Applications of Ground Penetrating Radar”.

References

[14] Elsevier. Scopus, the largest abstract and citation database of peer-reviewed literature [online, cited 2015].