Microwave Techniques for
Detection of Buried Objects

A thesis submitted to the University of Manchester for the degree of Master Philosophy in the Faculty of Engineering and Physical Sciences

2016

Yuanqing Geng

School of Electrical and Electronic Engineering
List of figures

Figure 1 stockpiles of antipersonnel mines (http://www.the-monitor.org/) ......................... 8
Figure 2 GPR system measurement with reflected waves(Huisman, Hubbard, Redman, & Annan, 2003) ............................................................................................................................... 13
Figure 3 GPR system diagram ................................................................................................ 15
Figure 4 GPR system detailed diagram .................................................................................. 16
Figure 5 a prototype of GPR system ...................................................................................... 17
Figure 6 FPGA working logic (FSM: finite state machine; DSP: digital signal processing; MEM: memory unit; TL: timing logic module; PG pulse generator; ADC: analog to digital convertor.) .............................................................................................................................. 18
Figure 7 the initial interface designed for detecting buried objects ........................................ 30
Figure 8 six parts of designed GUI ........................................................................................ 30
Figure 9 the initial buried object 's location ............................................................................ 31
Figure 10 the geometry algorithm of X and Z ...................................................................... 32
Figure 11 signals transmitted to detect the buried object ....................................................... 34
Figure 12 image reconstructed in Part 2 ............................................................................... 35
Figure 13 selected examples stored in advance ...................................................................... 36
Figure 14 Equivalent circuit of the SRD in both forward and reverse bias ......................... 41
Figure 15 Simulated results of a series configuration. (a) Normal p-n diode; GC2510 SRD ... 43
Figure 16 simulated circuit and result of a SRD in shunt configuration ................................. 44
Figure 17 simulation circuit and result for a double-shunt SRDs ........................................... 44
Figure 18 the prototype of pulse generation ........................................................................... 46
Figure 19 Modified pulse generation circuit and results ......................................................... 47
Figure 20 Pulse generation with variable resistance value .................................................... 48
Figure 21 simplified circuit of the pulse generator ................................................................. 49
Figure 22 tunable pico-second pulse generator .................................................................... 50
Figure 23 measured input signal ............................................................................................ 51
Figure 24 Single serial SRD .................................................................................................. 51
Figure 25 Double serial SRD with a 51 Ohm load ................................................................. 52
Figure 26 tunable pico-second pulse generator with different shunt resistance .................. 53
Abstract

Detection and identification of the buried objects beneath the surface have been gained a lot of attention from researchers, since it has been utilized in many regions. Such as landmine detection, dam monitoring, and pavement monitoring, etc. Due to its special advantage, ground penetrating radar (GPR) system has been considered as one of the major methods to image the subsurface.

This thesis provides a ground penetrating radar (GPR) system that is used to detect and identify buried objects beneath the surface. A method for numerical modelling of GPR system is presented in this thesis and an optimized algorithm is developed. Simulation of the GPR system under the environment of MATLAB is applied to observe and analyze the change of waveform propagation.

A new tunable pico-second pulse generator is revealed in this thesis. Step recovery diode (SRD) has been introduced here due to its particular advantages. Simulation and experiment operate for examination of the properties of the new tunable pico-second pulse generator.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright Statement

I. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

II. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

III. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

IV. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on Presentation of Theses.
Chapter 1

Introduction

1-1 Background

Detection and identification of the buried objects beneath the surface have been gained a lot of attention from researchers, since it has been utilized in many regions, such as landmine detection, dam monitoring, and pavement monitoring, etc. Due to its special advantage, ground penetrating radar (GPR) system has been considered as one of the major methods to image the subsurface.

Landmines are usually placed on or beneath the surface, and its action causes damage of machines and casualties of human beings. Landmines are the remains of human wars and there are many undetected landmines all over the world, which is a potential danger for local residents. In terms of the reports from the United Nations, there are approximately more than 100 million landmines buried in 62 countries throughout the world, and the number keeps increasing as more landmine fields has been located (Boutros-Ghali, 1994). Furthermore, since the development of material techniques, the metal contained in landmines has been replaced by plastic whose electric properties are much similar with soil, which brings much bigger difficulty for detection and identification.

Dam monitoring and pavement monitoring requires a long-term and continuous method to detect voids and cracks. The safety management should be very cautious to make sure every suspicious point identified. Hence, compared with landmine detection, Dam monitoring and pavement monitoring need more effort on object identification than
detection.

Figure 1 stockpiles of antipersonnel mines (http://www.the-monitor.org/)

Imaging is the measurement of the spatial distribution of some physical property of an object by use of an instrument such as a camera, an optical or ultrasonic scanner, a microscope, a telescope, a radar system, or an X-ray machine (Saleh, 2011). For imaging an object, which is buried below the surface and surrounded by a medium, such as soil, water or others, a nondestructive method will be helpful.

1-2 GPR: Basic Principles

Introduction

Ground penetrating radar (GPR) system is a nondestructive method that observe the world of subsurface by using radar pulse. In 1910, Gotthelf Leimbach and Heninrich Lowy had designed a system by using continuous-wave radar to detect buried objects. Sixteen years
later an advanced system came out by using radar pulse rather than a continuous wave (obonic, 2016). GPR has been well established and extensively utilized in our modern society, which includes the detection of landmine or other buried objects, the detection of voids or cavities, cancer investigation, as well as archaeological, environmental and hydro-geological surveys (Benedetto & Pajewski, 2015).

GPR’s have several advantages. They can be designed as portable equipment which operate object detection rapidly in most environment. They does not cost much relatively and provides good depth penetration and resolution (Witten, 1998). Induction metal detectors can only operate well against objects with high metal materials. FLIR systems rely highly on the stability of the environment. X-ray detectors’ high frequency property limits the penetration of the signal, and it works slow and costs high. Acoustic sensors have problems with the transmission of energy through the air-ground interface (Witten, 1998). Passive microwave sensors offer bad resolution.

<table>
<thead>
<tr>
<th></th>
<th>Scan speed</th>
<th>Depth penetration</th>
<th>low cost</th>
<th>Diverse environment</th>
<th>Good resolution</th>
<th>dielectric property</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Induction metal detector</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLIR system</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>X-ray backscatter detector</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Acoustic sensor</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Passive microwave sensor</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1 comparison between GPR and other sensors
Antenna

Antennas are usually used for coupling energy between different matters. As an important component of a radar system, antenna does not gain as much attention as the imaging techniques and the improvement of modelling/processing techniques. Therefore, more research activities on antennas should be implemented in the GPR area, in order to develop an innovative GPR antenna that matches all the requirements of object detection.

Considering GPR antennas usually operate in a specific condition, a number of characteristics must be included in GPR antennas, which have the GPR antennas different from those of conventional radar antennas. Considering the GPR antennas have to transmit and receive signal within a short-time, normally in the order of nanoseconds, the antennas demands an ultra-wide broadband property (Vescovo, Pajewski, & Tosti, 2016). GPR antennas should transmit the waveform in a condition of low distortion or ring-down, which requires a short impulse response. The optimum polarization should be capable of reducing the clutter of the received signal as much as possible, depending on the detection area feature (Witten, 1998). Circular polarization solves the problem of coupling between transmit and receive antennas, but the ground reflection is high which weakens the penetration. Vertical polarization provides a better penetration than horizontal polarization, while the clutter return is much lower for horizontal polarization. Therefore, in order to reduce the coupling between transmit and receive antennas and the clutter as much as possible, the information of the area without the object can be used as a reference.

\[
S_{\text{receive}} = S_{\text{coupling}} + S_{\text{clutter}} + S_{\text{object}}
\]
\[
S_{\text{ref}} = S_{\text{coupling}} + S_{\text{clutter}}
\]
\[
S_{\text{object}} = S_{\text{receive}} - S_{\text{ref}}
\]

Many previous research has used different kinds of antennas, among which resistively-loaded bow-tie and dipole antennas are still the most widely used in impulse GPR (Karim,
Malek, Jamlos, Seng, & Saudin, 2013; Lestari et al., 2010; Montoya & Smith, 1996, 1999). Array of antennas has been presented in GPR systems as they can collect data faster by extending the area at each scan (Vescovo et al., 2016).

**Frequency range and depth of penetration**

The frequency range of GPR normally stands within the region between 0.2 and 8GHz(Witten, 1998). The time-duration of the pulses are a tradeoff between the desired radar resolution and the depth of penetration (Vescovo et al., 2016). Low frequency signals allow relatively high penetration depth, while a relatively high resolution is given by high frequency signals. The conductivity property of material also acts an important role in scattering buried objects. High-conductive material absorbs signal energy heavily and there may be heterogeneous conditions in earth, such as rocks, useless metal or plastic objects. Therefore, we need to find an optimal solution on the frequency, which means GPR system could only be used to detect objects buried in a region of depth, normally 2 meters or less. A rough correlation between the horizontal resolution $\Delta x$, ground attenuation coefficient $\alpha$, and depth of penetration $d$ can be expressed by(Daniels, Gunton, & Scott, 1988):

$$D_x = 4d \sqrt{\frac{\ln 2}{2 + \alpha d}}$$

Assuming a dependence of the received signal power on $d$ in terms of $1/d^4$, the horizontal resolution improves as the attenuation increases, which means as the frequency increases(Soliman, 2008). Table 2 shows the approximate relationship of pulse durations, center frequency and depth resolution for a range of target depths given certain assumptions(Daniels, 1995).
<table>
<thead>
<tr>
<th>Target depth in metres</th>
<th>Pulse duration in nanoseconds</th>
<th>Centre Frequency</th>
<th>Depth resolution in meters</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0.25</td>
<td>0.5</td>
<td>2 GHz</td>
<td>0.03</td>
</tr>
<tr>
<td>&lt;0.5</td>
<td>1.0</td>
<td>1 GHz</td>
<td>0.05</td>
</tr>
<tr>
<td>&lt;1.0</td>
<td>2.0</td>
<td>500 MHz</td>
<td>0.1</td>
</tr>
<tr>
<td>&lt;2.0</td>
<td>4.0</td>
<td>250 MHz</td>
<td>0.2</td>
</tr>
<tr>
<td>&lt;4.0</td>
<td>8.0</td>
<td>125 MHz</td>
<td>0.4</td>
</tr>
<tr>
<td>&lt;8.0</td>
<td>16.0</td>
<td>63 MHz</td>
<td>0.8</td>
</tr>
<tr>
<td>&lt;16.0</td>
<td>32.0</td>
<td>31 MHz</td>
<td>1.6</td>
</tr>
</tbody>
</table>

1.…. Depth in a medium loss (<20 dB m⁻¹ attenuation) material.
2.…. Pulse duration to the half power width of the main peak.
3.…. Assumes a transmitted pulse in the general form of a Rayleigh wavelet.
4.…. Assumes a material of relative permittivity = 9.

Table 2 Approximate relationship of pulse duration, centre frequency, and depth resolution (Daniels, 1995).

Data acquisition

The electromagnetic waves emitted by GPR system propagate in each direction, and those which propagate into the earth will have an interaction with the buried objects. These interaction results in a scattered wave, which is measured by the receiving antenna of the radar (Soliman, 2008). By changing the location of the radar, a group of scattered data will be saved in the system which will be processed with a function of time and location. Then image reconstruction will show us the location of the buried objects finally. Figure 2 shows us two common measurements with reflected waves: common-midpoint (CMP, top) and wide angle reflection and refraction (WARR, bottom) acquisition (Huisman et al., 2003).
Modeling

Since the detection and identification of the buried objects have been in high favor with many researchers for years, ground penetrating radar system has been well developed. A number of methods have been studied the numerical modeling of GPR system, which include ray-based methods (Cai & Mcmechan, 1995; Goodman, 1994), frequency-domain methods (Zeng, Mcmechan, Cai, & Chen, 1995), integral methods (Ellefsen, 1999), and pseudospectral methods (Carcione, 1996; Casper & Kung, 1996). Among these methods, the finite-difference time-domain (FDTD) technique has been exploited mostly and proved to be the most common method for GPR modeling (Bergmann, Robertsson, & Holliger, 1996; Bourgeois & Smith, 1996; Holliger & Bergmann, 2002; Teixeira, Chew, Straka, Oristaglio, & Wang, 1998; Wang & Tripp, 1996). The FDTD
approach is easy to understand, accurate for model design and accommodates the specific demands of a certain GPR environment perfectly (Taflove, 1995).

Two-dimensional modelling methods has been highly exploited to provide us a better visual image (Irving & Knight, 2006), even a comprehensive 3-D imaging method has been use in GPR system (Catapano, Affinito, Crocco, Gennarelli, & Soldovieri, 2013). On the other hands, lots of kinds of signal generators are specifically designed with different components, such as step recovery diode (SRD), comb circuits, etc (Loranger, Iezzi, & Kashyap, 2012; Moll & Hamilton, 1969; Zhou, Yang, Lu, & Liu, 2015)

Conclusion

GPR could be roughly separated into three segments: the control unit, signal-transmitting unit (antennas), and power supply unit. Since the buried objects are beneath the surface, unlike free-space, the environment among them absorbs the signal strongly, which limits the frequency of signal in a range of 100 MHz to 2 GHz (Benedetto & Pajewski, 2015). The frequency of signal should be precisely chosen by considering the depth of the buried objects and the property of the media surrounding them.

Ground Penetration Radar system will bring a significant change to our normal life, and it will replace the traditional solutions in many fields, both military and civil applications. However, GPR system has its limitations which means we have to make sure each component included will cooperate in harmony, in order to take the most advantage of GPR system. Low frequency signals allow relatively high penetration depth, while a relatively high resolution is given by high frequency signals. The conductivity property of material also acts an important role in scattering buried objects. High-conductive material absorbs signal energy heavily and there may be heterogeneous conditions in earth, such as rocks, useless metal or plastic objects. Therefore, we need to find an optimal solution on the frequency, which means GPR system could only be used to detect objects
buried in a region of depth, normally 2 meters or less.

1-3 System Diagram

As mentioned above, Ground Penetrating Radar system normally includes three parts: control unit, antenna unit and power supply unit. Control unit is used for pulse generation and data processing, and antenna is responsible for signal transmission, and power supply unit is in charge of the power for driving each components included in the system. Control unit generates a pulse at the first step, and then the antenna emits the pulse by the transmitting part and receives the scattered signal by the receiving part, and then the collected data is processed by control unit and an image reconstruction completes.

We have developed a GPR system for detection and identification in time-domain, and the data acquired from the system will be processed by optical algorithms before being used to reconstruct the image. This is a prototype of GPR system, which means there are still several parts need to be modified and improved.
Figure 4 and 5 show the detailed diagram of the GPR system. An initial clock signal with 10 MHz pulse repetition frequency (PRF) is generated by field programmable gate array (FPGA), which is used to drive a <1 ns pulse generator. Then the pulse signal is modulated and amplified before it is transmitted via an ultra-wideband (UWB) antenna. At the signal receiving side, the signal is acquired by the other UWB antenna, and then amplified through a low-noise amplifier. Next, the signal is demodulated and down-converted in to tow parts of data, namely I and Q, which has a 90 degree difference. After that I and Q will be sampled by selected ADC. Then, all the sampling data are sent to a FPGA circuitry for storage and communication with PC for data processing and image construction (Wang., Fathy., & Mahfouz., 2011).
Field programmable gate array (FPGA) is a kind of semiconductor device that allow user to meet some application or functionality requires. Mainly FPGA contains a matrix of configurable logic blocks (CLP), and the combination of some particular CLPs would generate a logical action for users. There are two ways for FPGA design, namely hardware design language (HDL)-based design and schematic-based design. Schematic-based design provides gates and wires that creates functional blocks, and wires connect functional blocks to assemble as a program. However, a program in which functional blocks have relatively complex relationship with each other, the schematic-based design become a negative factor. So a HDL-based design is welcome here, while we also utilize the advantage of schematic-based design to make the whole model and each functional block intuitionally.

A FPGA with a style of Virtex-7 invented by Xilinx Inc. will fulfill our demands on FPGA. Virtex-7 FPGAs are optimized for system performance and integration at 28 nm and bring best-in-class performance/watt fabric, DSP performance, and I/O bandwidth to researchers’ designs(Xilinx, 2016).
As shown in figure 6, four modules are created inside FPGA by using HDL, which are finite state machine (FSM), digital signal processing module (DSP), memory unit (MEM), and timing logic module (TL). FPGA is in charge of sending a square wave with no false to pulse generator and collect data sent from ADC. Inside FPGA, finite state machine controls the other three modules to make sure each data has its specific address for reading and writing and operate a phase-shift function for each signal. Timing logic module generates all timing signal to make sure each module operates orderly. Digital signal processing module reads the data stored in memory unit and integrates all useful data for image reconstruction on computer.

1-4 Overview of chapters

The thesis consists of four chapters. Chapter 1 introduces ground penetrating radar and provides a system diagram including specification and architecture, which could be properly used for the detection of buried objects. The work flow of the system should be explained as well.

Chapter 2 mainly focuses on numerical modelling of ground penetrating radar (GPR)
system, and an optimized algorithm has been developed on the basis of Maxwell’s Equations. In this chapter, simulations of the detection and identification of buried objects are included.

A new tunable pico-second pulse generator is introduced in chapter 3. The step recovery diode (SRD) is introduced in this chapter, and the design, simulation and fabrication of the new pulse generator are included in the following parts of this chapter.

The final chapter gives a conclusion of the results of the entire work and the potential improvement of this project.
Chapter 2

Algorithm and simulation with MATLAB

2-1 Overview

Maxwell’s Equations are a group of four complicated equations, and we use them as laws to observe the behavior of electric and magnetic fields

\[ \nabla \cdot B = 0 \quad \text{Gauss’ law for the magnetic field} \]
\[ \nabla \cdot D = \rho_v \quad \text{Gauss’ law for the electric field} \]
\[ \frac{\partial B}{\partial t} = - \nabla \times E \quad \text{Faraday’s law} \]
\[ \frac{\partial D}{\partial t} = \nabla \times H - J \quad \text{Ampere’s law} \]

Where \( B \): magnetic flux density (weber/meter\(^2\)), \( D \): electric flux density (coulomb/meter\(^2\)), \( \rho_v \): electric charge density (coulomb/meter\(^3\)), \( E \): electric field (volt/meter), \( H \): magnetic field (ampere/meter), and \( J \): electric current density (ampere/meter\(^2\)). In order to exploit Maxwell’s equations, many methods come out, especially those with computer’s implement, and Finite Difference Time Domain (FDTD) is one of the most popular methods among them. Since FDTD method is a time-domain method, it allows us to analysis a set of results for different frequency with a single simulation run. According to Yee’s algorithm (Yee, 1966), each electric filed vector (E) component is surrounded by
four magnetic field vector components (H) and each magnetic field vector component is surrounded by four electric filed vector components in the grid. Furthermore, Yee shows a leapfrog manner which explains how resulting equations are solved. The first E component is computed in a particular time instant, then the relative H component will be computed in the next time instant. The process keeps running until all E and H components in a particular time zone are solved and stored.

Normally, when we start a simulation, an ideal circumstance is assumed in which materials have independence of position, direction, and their electric and magnetic properties are not changed by frequency. Therefore, we can use E and H to describe D and B by a linear relationship:

\[
D = \varepsilon E = \varepsilon_r \varepsilon_0 E
\]
(2.5)

\[
B = \mu H = \mu_r \mu_0 H
\]
(2.6)

Where \(\varepsilon\) is the electrical permittivity, \(\varepsilon_r\) is the relative permittivity, \(\varepsilon_0\) free-space permittivity \((8.854 \times 10^{-12}\ \text{farads/meter})\), \(\mu\) is the magnetic permeability, \(\mu_r\) is the relative permeability, and \(\mu_0\) is the free-space permeability \((4\pi \times 10^{-7}\ \text{henrys/meter})\). Also

\[
J = J_{\text{source}} + \sigma E
\]
(2.7)

By substituting equation (2.5) – (2.7) in (2.3), and (2.4), we obtain three 1-D case wave propagation for x, y, z direction.

X-direction:
\[
\frac{\partial E_Z}{\partial t} = \frac{1}{\varepsilon} \left( \frac{\partial H_y}{\partial x} - \sigma E_z \right)
\]

(2.8a)

\[
\frac{\partial H_y}{\partial t} = \frac{1}{\mu} \left( \frac{\partial E_z}{\partial x} \right)
\]

(2.9a)

Y-direction:

\[
\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon} \left( \frac{\partial H_z}{\partial y} - \sigma E_x \right)
\]

(2.8b)

\[
\frac{\partial H_z}{\partial t} = \frac{1}{\mu} \left( \frac{\partial E_x}{\partial y} \right)
\]

(2.9b)

Z-direction:

\[
\frac{\partial E_y}{\partial t} = \frac{1}{\varepsilon} \left( \frac{\partial H_x}{\partial z} - \sigma E_y \right)
\]

(2.8c)

\[
\frac{\partial H_x}{\partial t} = \frac{1}{\mu} \left( \frac{\partial E_y}{\partial z} \right)
\]

(2.9c)

In 1996, Tim Bergmann provided a 1-D FDTD modelling method, which is a 2\textsuperscript{nd}-order time and 4\textsuperscript{th}-order space accurate [O(2,4)] finite-difference approximations of equations above (Bergmann, Robertsson, & Holliger, 1996). In order to get the finite-difference approximations from equations 2.8-2.9, a few operators are used here (Taflove, Hagness, & Piket-May, 2005):

\[
E \varphi_n = \varphi_{n+1}
\]

(2.10a)
\[ \mu \varphi_n = \frac{\varphi_\frac{n+1}{2} + \varphi_\frac{n-1}{2}}{2} \]  
(2.10b)

\[ \delta \varphi_n = \varphi_{n+1/2} - \varphi_{n-1/2} \]  
(2.10c)

\[ \delta^+ \varphi_n = \varphi_{n+1} - \varphi_n \]  
(2.10d)

\[ \delta^- \varphi_n = \varphi_n - \varphi_{n-1} \]  
(2.10e)

\[ D \varphi = \frac{\partial \varphi}{\partial x} \]  
(2.10f)

Here \( E \) is the shift operator, \( \mu \) is the averaging operator, \( \delta \) is the central difference operator, \( \delta^+ \) is the forward difference operator, \( \delta^- \) is the backward difference operator, and \( D \) is the difference operator.

\[
\begin{align*}
\varphi(x + \Delta x) &= \varphi(x) + \Delta x \varphi_x + \frac{(\Delta x)^2}{2!} \varphi_{xx} + \cdots \\
\varphi(x + \Delta x) &= \varphi(x) + \Delta x D \varphi(x) + \frac{(\Delta x)^2}{2!} D^2 \varphi(x) + \cdots \\
\varphi(x + \Delta x) &= (1 + \Delta x D + \frac{(\Delta x)^2}{2!} D^2 + \cdots) \varphi(x) \\
\varphi(x + \Delta x) &= e^{\Delta x D} \varphi(x)
\end{align*}
\]
(2.11)

Therefore

\[ E \varphi(x) = e^{\Delta x D} \varphi(x) \]  
(2.12)

Or
\[ E = e^{\Delta xD} \]  
(2.13)

For the central difference formula

\[ \delta \varphi_i = \left( E^\frac{1}{2} - E^{-\frac{1}{2}} \right) \varphi_i \]  
(2.14)

In this case

\[ \delta = e^{\Delta xD/2} - e^{-\Delta xD/2} = 2\sinh \frac{\Delta xD}{2} \]  
(2.15)

So

\[ \Delta xD = 2\sinh^{-1}\frac{\delta}{2} = 2\left[ \frac{\delta}{2} - \frac{1}{2 \cdot 3} \left( \frac{\delta}{2} \right)^3 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5} \left( \frac{\delta}{2} \right)^5 + \cdots \right] \]

\[ = \delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7168} + \cdots \]  
(2.16)

Keeping only the first term gives the second-order formula

\[ \frac{\partial \varphi}{\partial x} \bigg|_i = \frac{\varphi_{i+\frac{1}{2}} - \varphi_{i-\frac{1}{2}}}{\Delta x} - \frac{(\Delta x)^2}{24} \frac{\partial^3 \varphi}{\partial x^3} + O\left( \frac{\partial \varphi}{\partial x}^4 \right) \]

(2.17)

Therefore,

\[ \frac{\partial \varphi}{\partial x} \bigg|_i = \frac{-\varphi_{i+\frac{3}{2}} + 27 \varphi_{i+\frac{1}{2}} - 27 \varphi_{i-\frac{1}{2}} + \varphi_{i-\frac{3}{2}}}{24\Delta x} + O\left( \frac{\partial \varphi}{\partial x}^4 \right) \]

(2.18)
The $\frac{\partial \varphi}{\partial x_i}$ is approximated using forth-order-accurate finite different expression

Normally we choose a 2-D or 3-D modelling since 1-D modelling does not carry enough information we have. In 2006, James Irving and Rosemary Knight (Irving & Knight, 2006) present a 2-D numerical modelling and this model is developed.

$$H_{x,i,j+y/2}^{n+1/2} = H_{x,i,j+y/2}^n - D_b \left|_{i,j+y/2} \right. \left[-E_y|_{i,j+1}^n + 27E_y|_{i,j+1}^{n} - 27E_y|_{i,j-1}^n + E_y|_{i,j-1}^n \right]$$

$$- D_c \left|_{i,j+y/2} \right. \left[\Psi_{H_x}|_{i,j+y/2}^n \right]$$

(2.19a)

$$H_{z,i+y/2,j}^{n+1/2} = H_{z,i+y/2,j}^n - D_b \left|_{i+y/2,j} \right. \left[-E_y|_{i+2,j}^n + 27E_y|_{i+2,j}^{n} - 27E_y|_{i-1,j}^n + E_y|_{i-1,j}^n \right]$$

$$+ D_c \left|_{i+y/2,j} \right. \left[\Psi_{H_x}|_{i+y/2,j}^n \right]$$

(2.19b)

$$E_y|i,j+1 = C_a |_{i,j} \left[ E_y|i,j \right] + C_b |_{i,j} \left[-H_x|i+3/2,j|^{n+1/2} + 27H_x|i+3/2,j|^{n+1/2} - 27H_x|i-1,j|^{n+1/2} + H_x|i-1,j|^{n+1/2} \right]$$

$$- C_{bc} |_{i,j} \left[-H_x|i+3/2,j|^{n+1/2} + 27H_x|i+3/2,j|^{n+1/2} - 27H_x|i-3/2,j|^{n+1/2} + H_x|i-3/2,j|^{n+1/2} \right]$$

$$+ C_{c} |_{i,j} \left[\Psi_{E_y}|_{i,j}^{n+1/2} - \Psi_{E_y}|_{i,j}^{n+1/2} \right]$$

(2.19c)

The convolution terms in equations (2.19a)-(2.19c) are modeled using recursive convolution technique (Luebbers & Hunsberger, 1992).

$$\zeta_z(t) * \frac{\partial E_y}{\partial z} = \Psi(t) = \int_0^t \zeta(\tau)E(t - \tau)\,d\tau$$

(2.20)
Where \(\zeta(t)\) represents \(\zeta_z(t)\), and \(E(t)\) represents \(\frac{\partial E_y}{\partial z}\). Using Yee notation with \(t=n\Delta t\), we obtain from equation (2.20)

\[
\Psi(t) \approx \Psi(n\Delta t) = D^n = \int_0^{n\Delta t} \zeta(\tau)E(n\Delta t - \tau)d\tau
\]

(2.21)

If we make the approximation that all field quantities are constant over each time interval \(\Delta t\), and assume that all fields are zero for \(t < 0\), then the integration becomes in part a summation so that equation (2.21) is equivalent to

\[
\Psi(t) \approx \Psi(n\Delta t) = D^n = \sum_{m=0}^{n-1} E^{n-m} \int_{m\Delta t}^{(m+1)\Delta t} \zeta(\tau)d\tau = \sum_{m=0}^{n-1} E^{n-m} \zeta^m
\]

(2.22)

Since

\[
\sum_{m=0}^{n-1} E^{n-m} \zeta^m = \sum_{m=1}^{n-1} E^{n-m} \zeta^m + E^n \zeta^0 = e^{-\frac{t}{\varepsilon_0 k^2}} \sum_{m=0}^{n-2} E^{n-m} \zeta^m + E^n \zeta^0
\]

\[
= e^{-\frac{t}{\varepsilon_0 k^2}} \Psi(t-1) + E^n \zeta^0
\]

(2.23)

And

\[
\zeta^0 = \int_0^{\Delta t} \zeta(\tau)d\tau = -\frac{\sigma}{\varepsilon_0 k^2} \int_0^{\Delta t} e^{-\frac{\tau}{\varepsilon_0 k^2}}\left(\frac{\sigma}{\varepsilon_0 k^2}\right)^2 d\tau = \frac{\sigma}{\sigma k + ak^2}\left(e^{-\frac{\Delta t}{\varepsilon_0 k^2}}\left(\frac{\Delta t}{\varepsilon_0 k^2}\right)\right) - 1)
\]

(2.24)

Hence
Finally, we obtain

\[
\Psi(t) = e^{\left(-\frac{\Delta t}{\varepsilon_0 \sqrt{k^2 + \alpha^2}}\right)} \Psi(t-1) + \frac{\sigma}{\sigma k + \alpha k^2} (e^{\left(-\frac{\Delta t}{\varepsilon_0 \sqrt{k^2 + \alpha^2}}\right)} - 1) \cdot \frac{\partial E_y}{\partial z}
\]  
(2.25)

\[
\Psi_{H_z|_{i,j+1/2}}^n = B_x|_{i,j+1/2} \left[ \Psi_{H_z|_{i,j+1/2}}^{n-1} \right] + A_x|_{i,j+1/2} \left[ -E_y|_{i,j+1} + 27 E_y|_{i,j+1} \right] 
\]
(2.26a)

\[
\Psi_{E_y|_{i,j}}^{n+1/2} = B_x|_{i,j} \left[ \Psi_{E_y|_{i,j}}^{n-1/2} \right] + A_x|_{i,j} \left[ H_z|_{i+1/2,j} + 27 H_z|_{i+1/2,j} \right] 
\]
(2.26b)

\[
\Psi_{H_x|_{i-1/2,j}}^n = B_x|_{i+1/2,j} \left[ \Psi_{H_x|_{i-1/2,j}}^{n-1} \right] + A_x|_{i-1/2,j} \left[ -E_y|_{i-1,j} + 27 E_y|_{i-1,j} \right] 
\]
(2.26c)

\[
\Psi_{E_x|_{i,j}}^{n+1/2} = B_x|_{i,j} \left[ \Psi_{E_x|_{i,j}}^{n-1/2} \right] + A_x|_{i,j} \left[ H_z|_{i,j+1/2} + 27 H_z|_{i,j+1/2} \right] 
\]
(2.26d)

Where \( A_k = \frac{\sigma_k}{\sigma_k k^2 + \alpha_k k^2} (B_k - 1) \), and \( B_k = \exp\left[-\frac{\Delta t}{\varepsilon_0 (\sigma_k k^2 + \alpha_k k^2)}\right] \) are PML update coefficients that vary with location in the modeling grid. As can be seen from equations...
(2.26a) to (2.26d), the values of the convolution terms at the current time step are computed from those at the previous time step. Therefore, $\Psi_{H_x}$, $\Psi_{H_z}$, $Y_{E_y}$, and $\Psi_{E_z}$ must be stored, in addition to the $H_x$, $H_z$, and $E_y$ field components, during the FDTD simulation (Irving & Knight, 2006).

2-2 Absorbing Boundary Condition

When we try to simulate the behavior of electromagnetic wave propagation, the fact, part of the wave will be reflected by the simulation boundary, should not be ignored. Considering the real circumstance of where GPR system used is normally outdoor, in other words, an infinite region, an absorbing boundary condition (ABC) must be applied to eliminate the effect of limitation of simulation environment. An approach to realize an ABC is to choose a proper absorbing material as the boundary. Ideally, the thickness of the boundary can be controlled within a few units’ length (Taflove et al., 2005).

Compared with other absorbing boundary conditions, the Perfectly Matched Layer (PML) method has earned a lot of reputation due to its practicability and convenience. In order to ideally reduce the reflections from the boundary, the width of PML only needs a few units’ length. And the change of PML’s electromagnetic properties does not affect the behavior of waveform inside the grid. The Perfectly Matched Layer (PML) method is selected in this thesis.

The split-field PML can also be applied directly within an FDTD discretization using a very effective scheme referred to as the Convolutional PML (CPML), introduced by Roden and Gendy (Roden & Gedney, 2000). The CPML accommodates more general metric tensor coefficients that can lead to improve absorption of slowly varying evanescent waves (Irving & Knight, 2006). Therefore, the finite-difference expressions can be derived by applying the PML as following. Assuming the stretched-coordinate tensor coefficient $S_w$ is:
\[ S_w = k_w + \frac{\sigma_w}{a_w + j\omega e_0}, \quad w = x, y, z \]

(2.27)

2-3 Simulation using MATLAB Graphical User Interface (GUI)

MATLAB Graphical User Interface is a product that allows us to operate in front of the PC, instead of modifying the programming code at background. For a series of simulations among which only a few factors changed, GUI is more convenient for us. Furthermore, GUI provides us a face-to-face simulation environment that gives us an advantage to observe the result of simulation by image. Considering we will run a simulation targeted to the propagation of waveform and buried objects, GUI should be a good tool to realize that.

Figure 7 shows the initial interface designed for detecting buried objects. Through GUI, we can handle every step of simulation more visually and briefly, compared with MATLAB m-file editor. We can easily change the properties of buried objects to see the simulation results quickly and intuitively. The total running time for simulation is less than 5 minutes, therefore, it is very convenient for users to identify the location and material of the buries objects, without doing nothing but waiting a long time.

Next, we will get a deep understanding about each part of the designed interface, in order to know the principles and methods in it.

In Figure 8, we divide the interface into 6 parts, according to their application purpose and UI Control types. The UI Control elements used in this simulation include Editable Text, Frames, Pop-Up Menus, Push Buttons, Sliders, Static Text, Axes. The most amazing advantage of GUI is that it allows us to change the size and properties of UI Controls, which save that much time on writing code for the object design. On the other hand, with
this advantage, various kinds of GUI come out. Each part of the GUI has a fundamental use which will perform an entire simulation system when combined together. The most important thing to remember is that make sure each use can work together harmonically and no interruption occurs during simulation.

Figure 7 the initial interface designed for detecting buried objects

Figure 8 six parts of designed GUI
Part 1: This plot is exploited to display the figure created by data from Part 4, and when we push the push-button “SET” in Part 5, the call-back function will get the command and plot the figure in Part 1. All Part 1 need to do is set the current axes as the one in Part 1, so that the figure will be plotted in Part 1, rather than the main figure, namely test3.m in this condition. Shown in Figure 9.

![Figure 9 the initial buried object’s location](image)

Part 2: The UI Control element used in this part is axes, which gives us a continuous image of EM wave propagation. It uses the former GPR modeling using Finite Difference Time Domain (FDTD) Method. We use common offset method to transmit and receive signals, and 20 sources are chosen, considering the accuracy of transmission result and computing period. The distance between two sources is 5cm, and the transmitter and receiver has a 0.5cm distance.

The time domain transmission response without the presence of the object is denoted as $S_{WO}(t)$, and that with the presence of the object is denoted as $S_{WO}(t)$. The direct coupling between the transmitting and receiving antennas can be removed by subtracting these two responses as in equation (2.28),
To gain more information about the object, B-scan data acquisition scheme needs be applied. Also, to emphasis the presence of an object, equation (2.29) given below is proposed for the first time to measure the percentage change of the reflected signal power at each A-scan position where \( t \) is the number of time points.

\[
\Delta S_2 = \frac{\sum_t \left( S_{\text{MOO}} - S_{\text{MOO}}^2 \right)^2}{\sum_t S_{\text{MOO}}^2} \times 100\%
\]

(2.29)

For convenience, both X- and Z-directions are defined to simulate the experiment setup. X-direction is horizontal to the ground surface, along which both transmitting and receiving antennas are located and B-scan takes place. Z-direction is normal to the ground surface, representing the height of the object. The XoZ plane represents a cross-sectional view of the ground surface. Both X- and Z- directions are shown in figure 10.

![Figure 10 the geometry algorithm of X and Z](image-url)
For image reconstruction, the XoZ cross-section is divided into N x N pixels. The distance from the transmitter in an A-scan to pixel \((i,j)\) is \(R_1\) and that from the receiver is \(R_2\) giving rise to the total distance of \(d_{ij}\). These quantities can be expressed as,

\[
\begin{align*}
R_{1\downarrow i,j} &= \sqrt{\left(x_i - x_0 + \frac{\text{offset}}{2}\right)^2 + z_j^2} \\
R_{2\downarrow i,j} &= \sqrt{\left(x_i - x_0 - \frac{\text{offset}}{2}\right)^2 + z_j^2}
\end{align*}
\]

(2.30)

\[d_{i,j} = R_{1\downarrow i,j} + R_{2\downarrow i,j}\]

(2.31)

where \text{offset} is the distance between the transmitting and receiving antennas and \(x_o\) is the midpoint X-value, and \(x_i\) and \(z_j\) are the corresponding X- and Z- values at pixel \((i,j)\). The values of \(R_{1\downarrow i,j}, R_{2\downarrow i,j}\) and \(d_{ij}\) are unique for each A-scan and pixel position. However, in the inverse case, a constant value of \(d\) with,

\[d = R_1 + R_2\]

(2.32)

The time domain result \(\Delta S_i(t)\) in equation (2.28) for each A-scan is converted into a 2D response on the XoZ plane with

\[d = vt\]

(2.33)

where \(v\) is the speed of propagation, and the corresponding \(\Delta S_i(t)\) value is assigned to the pixels with distance \(d\). The process is then repeated for all other A-scans in a B-scan of \(M\) measurements along the X-direction. The values of \(\Delta S_i(t)\) for all A-scans in a pixel are accumulated to give a resultant value \(G(i,j)\) for the pixel, i.e.
\[ G(i,j) = \sum_{m=1}^{M} \Delta S_{1,m}(t) \mid_{\text{pixel } (i,j) \in d=vt} \]

The plot of the resultant values on the XoZ plan produces an image, from which the position of the object can be located in terms of both X and Z coordinates. Also, the object dimensions can be clarified from the image obtained in terms of the image intensity levels.

When the wave propagation is finished, the detected location will show up on this axes, hence, we can compared the real one and the detected one to see if they match well. The axes in Part 1 and the one in Part 2 have the same size, so that we can compare the results intuitively.

While though this GUI we cannot identify its material, and the main index is dielectric permittivity. We can analyze the intensity of the reflected signals to determine the amount of permittivity. This will be studied in the further research.

Figure 11 signals transmitted to detect the buried object
Part 3: The UI Control element used in this part is axes, after the wave propagation showed in Part 2, the initial detected image come out in this part, we can see from Figure 5 that the image reconstructed in part 2 is much more clear and accurate than the first one in Part 3.

Part 4: The UI Control elements used in this part include frame, sliders, editable text, and static text. This Part is designed for users to decide the location and size of the buried objects they want. You can either move the slider bar or type the value in the editable text to change the property values. This part guarantees users remember these important values through the whole simulation process and make it visible for us.

Part 5: The UI Control element used in this part includes push buttons. One is named “SET”, and the other is “RUN”, it is not difficult to understand their duty from their titles. “SET” is used to get the data from Part 4, and give the command to Part 1 to plot the location image of the buried object. While “RUN” controls the rest parts of the entire GUI. First, it orders Part 2 to indicate the continuous image of wave propagation, and then has
Part 3 plot the initial image of the detected result, finally, it reconstructs the image of buried object with an innovative image reconstruct technique, and plot it on the axes in Part 2.

Part 6: The UI Control element used in this part includes pop-up menu. This part is designed for demonstration to potential users. We can choose any of these examples to see a brief, but comprehensive display of this simulation.

![Figure 13 selected examples stored in advance](image)

2-4 Summary

This chapter is an update and improvement for the previous study, with the application of Graphic User Interface, we can put all of the information we need in a single interface. It brings convenience and high efficiency in the way we can manage each step visually and intuitively.
The designed GUI still has some flaws, as we do not know the electric properties of the detected object, so we cannot identify whether it is the buried object we want to find out. But we can utilize the information hidden inside the reflected signals, and extract the information we need to figure out the value of dielectric permittivity and other parameters we need. That should be the next step we take.
Chapter 3

A Novel Circuit for Pico-second Pulse Generation

3-1 Introduction

Ultra-Wideband (UWB) systems have been applied in many areas due to its advanced properties. For instance, the detection and identification of landmines, safety management of dam and pavement, and communications within a short range. However, most of the pulse generators used in this systems have constant pulse durations.

Pulse generators, which are capable of tuning the pulse duration electronically, offers us more convenience and allow us to obtain more reliable data in UWB systems. For instance, the difference among fabrications of each pulse generators. A pulse with deep penetration and fine resolution does not exist. A pulse with a long wave-length can penetrate deeply due to its large energy, while the bandwidth goes narrower as the energy increase. A pulse that has a wide band-width would ensure a fine range resolution, but its energy could not allow for a deep penetration. Therefore, a tunable pulse generator would change its pulse during application brings a lot of advantages for GPR systems.

3-2 Step recovery diode (SRD)

Step recovery diode (SRD) is a device which can bring remarkable help in many applications, such as pulse shaping and waveform generation, or harmonic frequency multiplication and frequency comb generation and so on.

In general, the SRD is applied as a switch controlled by charge. For instance, when the
SRD is forward biased, it is in the charging mode, which means the charge is stored in the SRD which is low impedance; when reverse biased, it changes to discharging mode, and SRD keeps a low impedance before all charge removed, then it switches from a low impedance to a high impedance in a very short time. Therefore, the SRD has a very unique advantage that it can change its impedance level very rapidly, and this allows us for generating pico-second level pulses and general waveform shaping. Considering the SRDs are inexpensive, simple to design, and have low power consumption with relatively high output voltage swings, they are the most ideal devices for UWB pulse generation.

Compared with a usual p-n junction diode, the step recovery diode (SRD) has a quite different dynamic characteristics. The most distinguish feature of the SRD is the very abrupt dependence of its junction impedance upon its internal charge storage (J.L.Moll, S.Krakauer, & R.Shen, 1962). This storage of charge occurs as a result of the non-zero recombination time of minority carriers that have been injected across the junction under forward bias conditions (Packard, 1984).

According to the charge continuity equation, we can obtain the charge in the junction.

\[
i = \frac{q}{\tau_{mc}} + \frac{dq}{dt}
\]

(3.1)

Where \(i\) is the instantaneous diode current, \(q\) is the charge stored in junction and \(\tau_{mc}\) is the minority carrier lifetime of a diode. By using Laplace transforms, we can obtain

\[
Q(s) = \frac{1}{s(s + \frac{1}{\tau_{mc}})} + \frac{Q_0}{s + \frac{1}{\tau_{mc}}}
\]

(3.2)

The first step is charging. Assuming the SRD has no charge stored at the beginning and the forward current is constant, then
\[
Q(s) = \frac{I_F}{s(s + \frac{1}{\tau_{mc}})}
\]

(3.3)

Converting equation () from S domain to time domain, then

\[
q_F = i_F \tau_{mc} \left[ 1 - e^{-\frac{t_F}{\tau_{mc}}} \right]
\]

(3.4)

Where \(q_F\) is the charge store from the forward current \(i_F\) and \(t_F\) is the length of time for which the forward current \(i_F\) is applied (Packard, 1984).

The second step is discharging, for a particular time \(t_R\), all the charge stored in SRD will be removed, and the initial amount of stored charge is \(q_0\), hence

\[
q(t) = -i_R \tau_{mc} \left[ 1 - e^{-\frac{t_R}{\tau_{mc}}} \right] + q_0 e^{-\frac{t_R}{\tau_{mc}}}
\]

(3.5)

When all the charge is removed, \(q(t)=0\), and \(q_0\) is the charge stored during the charging step. Usually \(t_F\) is much longer than \(\tau_{mc}\) and \(i_F \ll i_R\) Therefore, this gives us

\[
t_R = t_s \approx \frac{\tau_{mc} i_F}{i_R}
\]

(3.6)

\(t_s\) is often recognized as the snap time of SRD.

The SRD is a kind of an ideal nonlinear capacitor, and we must take care to recognize the fundamental difference between the ideal nonlinear reactive device and the ideal nonlinear resistive device. The ideal rectifier does not store charge when the circuit is forward-biased, and it will turn immediately as the terminal voltage turns. However, the
ideal step recovery diode (SRD) does not turn as soon as the terminal voltage reverse. Since it is charged during forward-biased time, the stored charge need to be removed by negative voltage, the SRD’s voltage will remain negative for a while. Then SRD changes to the same voltage of terminal voltage at a speed depending on:

(1) The RC time constant: where R is the combination of the generator and load resistance and C is the reverse bias capacitance of the diode.

(2) The transition speed of the diode (in the ideal case, assumed to be zero).

![Equivalent circuit of the SRD in both forward and reverse bias](image)

3-2-1 SRD Modelling

An ADS basic p-n junction diode model is used and modified to obtain the SRD model. By exploiting the p-n junction model, we can modify four parameters which are the primary difference between a normal P-I-N junction diode and a SRD.

The four parameters include the Transit time (Tt), emission coefficient (N), zero-bias
junction capacitance ($C_{j0}$), and grading coefficient ($M$). The transit time parameter is actually the minority carrier lifetime of the SRD. It describe the time for a carrier to recombine. The emission coefficient is also called the diode ideality factor. It indicates the electromagnetic power output per unit time and is affected by the fabrication process and materials used. This value ranges from 1 to 2 where 1 represents an ideal p-n junction diode. The zero bias junction capacitance is used to describe the linear capacitance that exists between the p-type and n-type material, before the application of a bias. This capacitance affects the charging and discharging characteristics of the diode. The grading coefficient describes the slope associate with the impurities introduced at the diode junction.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Unit</th>
<th>P-N Junction Diode Model parameters</th>
<th>GC2510 Device Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_s$</td>
<td>Saturation current</td>
<td>$A^0$</td>
<td>$10^{-14}$</td>
<td>$82 \times 10^{-13}$</td>
</tr>
<tr>
<td>$R_s$</td>
<td>Ohmic resistance</td>
<td>$\Omega$</td>
<td>0.0</td>
<td>1.2</td>
</tr>
<tr>
<td>$N_e$</td>
<td>Emission coefficient</td>
<td>$-\psi$</td>
<td>1.0</td>
<td>1.8</td>
</tr>
<tr>
<td>$T_t$</td>
<td>Transit time</td>
<td>nsec</td>
<td>0.0</td>
<td>10</td>
</tr>
<tr>
<td>$C_{j0}$</td>
<td>Zero-bias junction capacitance</td>
<td>$\mu F$</td>
<td>0.0</td>
<td>0.545</td>
</tr>
<tr>
<td>$C_{j0}$</td>
<td>Sidewall zero-biased capacitance</td>
<td>$\mu F$</td>
<td>$0.9 \times 10^{-12}$</td>
<td>0.3</td>
</tr>
<tr>
<td>$V_{j}$</td>
<td>Junction potential</td>
<td>$V$</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>$M$</td>
<td>Grading coefficient</td>
<td>$\psi$</td>
<td>0.5</td>
<td>0.235</td>
</tr>
<tr>
<td>$XTI$</td>
<td>Saturation-current temperature exponent</td>
<td>$-\omega$</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>$E_G$</td>
<td>Energy gap</td>
<td>$eV$</td>
<td>1.11</td>
<td>1.12</td>
</tr>
<tr>
<td>$B_V$</td>
<td>Reverse break down voltage</td>
<td>$V$</td>
<td>$\infty$</td>
<td>15</td>
</tr>
<tr>
<td>$IB_V$</td>
<td>Current at reverse break down voltage</td>
<td>$A$</td>
<td>0.001</td>
<td>$10^{-5}$</td>
</tr>
</tbody>
</table>

Table 3 a p-n junction diode model and GC2510 parameters
From Figure 15, the dynamic characteristics of a SRD are shown intuitively. Compared with the result of normal p-n junction diode, the output of SRD has a voltage drop off, and a reverse recovery period which has been indicated in Figure 15 (b).

When a single SRD is shunt connected to the load, a sharp edge occurs at the rising edge or the falling edge of the sinusoidal input, depending on the type of bias:
When a double-shunt SRDs connected to the circuit, both rising and falling edge of the sinusoidal waveform will be cut into a sharp edge, and the positive part and negative part of the waveform would be relatively smooth. The circuit and simulated result are shown below.

(a) Simulation circuit for a double-shunt SRDs

(b) Simulation result of the double-shunt SRDs

Figure 16 simulated circuit and result of a SRD in shunt configuration

Figure 17 simulation circuit and result for a double-shunt SRDs
3-3 A new circuit for pico-second Generation

The pulse generation has become a very important part in the UWB system. To some extent, the performance of UWB system would count much on the pulse shape. Therefore, a good pulse generation method would guarantee a good operation.

Compared with other techniques of generating a UWB pulse with a sharp transition, such as FET, transmission line, etc., SRDs allow low power consumption and simple circuit, which will provide a much promising circuit for pulse generation.

3-3-1 Circuit design and modification

Considering the dynamic characteristics of the SRD, the prototype of pulse generation circuit is shown below. This circuit is simply composed of a forward-bias SRD and a load of 51 Ohm. The simulation results are shown below.

The input signal is a square waveform, which generated from the FPGA board. Otherwise we can modify the sinusoidal waveform to an ideal square waveform by applying the double-shunt SRDs circuit. The rising edge and falling edge of the square waveform is set at 500 ps, which is reasonable assumption. Amplitude is 2volts, and no negative part included in the waveform, for the nature of the input signals a clock signal generated by FPGA.

(a) Circuit schematic
In Figure 18(c), the circled higher voltage indicates the forward recovery, which happens when a diode switches rapidly from its off-state to its on-state. When a transient happens, the initial conductivity of the diode is very low, so the forward voltage would be higher than normal. After a time the voltage will decrease to a normal value since the diode’s conductivity rises. The difference value between the peak voltage and the normal voltage is related to the initial resistance, and the time spent on the recovery is decided by the transition time.

Also we can observe the voltage has a drop about 0.8 volts, because of the steady-state voltage in SRD. And the reverse period shown in the figure comprehensively explained
the equation

\[ t_r = t_s \approx \frac{\tau_{mf}}{I_R} = \frac{8 \times 1.1}{0.9} = 9.8 \text{ ns} \]  

(3.7)

Next step, we will serially connect another SRD to the load, because the output of above signal has a great advantage for the pico-second pulse generation

In Figure 19, the output signal has a negative pulse which has a less than 1 ns width and the steady positive part is around 0.3 volts. This happens is because the forward current decreased and the reverse current increased, due to the steady-state voltage drop of SRD.
thereby, the period for carrier to recombine is much shorter, which promises a pico-second pulse.

3-3-2 Tunable pulse generator

To a UWB application system, tunable pulse generator provides much more help and convenient. A tunable pulse generator would give a series of different width pulses and this ability allows simpler operation and less cost.

In the new pulse generation circuit introduced above, the resistance load, which is between the first and second SRDs, has an important position where determines the ability of tuning pulse width.

Figure 20 Pulse generation with variable resistance value
As shown in Figure 20, with different values of R2, the widths of the pulse is direct proportional related. During the reverse recovery, a ring effect occurs since the SRD switch from its turn-on state to its turn-off state. According to the equivalent circuit of the SRD, the SRD can be briefly replaced by a serial combination of steady resistance and reactance. The simplified circuit is shown below.

From Figure 21, we can figure out that

\[
\begin{align*}
R' &= \frac{R}{jX_2 + R_2 + R_1} \\
V_1 &= \frac{R'}{R' + jX_1 + R_1} V_{in} \\
V_{out} &= \frac{R_{load}}{R_{load} + R_2 + jX_2} V_1
\end{align*}
\]

(3.7)

then we can find out

\[
\begin{align*}
R' &= \frac{R_{load} + R_2 + jX_2}{1 + \frac{R_{load} + R_2 + jX_2}{R}} \\
V_1 &= \frac{1}{1 + \frac{R_1 + jX_1}{R'}} V_{in} \\
V_{out} &= AV_1
\end{align*}
\]

(3.8)
Therefore, \( V_{\text{out}} \propto V_1 \propto R' \propto R \). When the SRD forward bias, the voltage on the load would decrease as \( R \) decreases. However, when reverse bias, the voltage would remain the same since the carrier’s recombination are affected by the input signal rather than the circuit construction. So a lower forward voltage results in a shorter period for recombination, in other words, the width of the pulse would be narrower.

3-3-3 Fabrication and measurement

The tunable pico-second pulse generator was fabricated on a FR-4 glass epoxy substrate with a dielectric constant of 4.5 and a board thickness of 0.017 mm. The setup used to test the generator included the use of a Xilinx Virtex6 ML605 FPGA Board to produce the 10 MHz 1.78 V\(_{pp}\) square stimulus that was required at the input of the circuit. The setup also included an Angilent 54854A 4GHz oscilloscope to capture the output. The measured data that was obtained from this setup was compared to data simulated using Advanced Design System (ADC) 2009A.

![Figure 22 tunable pico-second pulse generator](image)

The input square signal has a peak voltage of 1.784 V, and the minimum of it is -16 mV. The rising time is 741.05 ps and the falling time is 356.79 ps, which have a slight
difference with the simulation parameter.

Figure 23 measured input signal

Figure 24 Single serial SRD
Figure 25 Double serial SRD with a 51 Ohm load

(a) Output with a 51 ohm R2

(b) Output with a 25 Ohm R2
With a single serial SRD forward-biased in the circuit, the output has a 10 ns negative pulse (Figure 25), and when the second SRD added in the circuit, the pulse has been narrowed into only a few ns. From figure 26, we can see that the width of the pulse depends on the resistance.

3-4 Summary

In this chapter, a new circuit of tunable pico-second pulse generator has been introduced. Different from the previous methods for pulse generation, the pulse generator created in this project mainly depends on the properties of the chosen SRD and the amplitude of input signal. The SRDs are inexpensive, simple to design, and have low power consumption with relatively high output voltage swings.

First, the minority carrier life time of the SRD should be considered, which determines the speed of carrier recombination. Then, the amplitude of the input signal should be a little more than twice of the static voltage of the specific SRD.

The new circuit would be very important in the GPR system, given the pulse generated by it carries the information of the buried objects. Therefore, the new tunable pico-second pulse generator would be significantly helpful.
Chapter 4

Conclusions

In this thesis, the author intends to develop a ground penetrating radar (GPR) system that can be applied for the detection and identification of buried objects. In order to realize this, the project is divided into two categories: software programming and hardware design.

The software programming provides a tool that can process large amount of data and reconstruct the image of subsurface in real-time. Therefore, the author has improved an algorithm which is used for numerical modelling of GPR system, and operate a simulation which reveals the algorithm can detect and identify buried objects whose electric properties are different with those of soil. While it is only a simulation and experiment should be taken in the future to verify it.

In this project, the author also focused on creating a novel pulse generator with Nano-seconds pulse. The new tunable pico-second pulse generator has a very simple circuit design, while it is capable of generating a series of pulse signals with a stable time-range difference.

Besides, a further work is necessary. FPGA acts a vital role in GPR system designed in this thesis, and s few modules should be designed by FPGA, such as finite state machine, digital signal processing module, timing logic module, memory unit module and I/O module. A full understanding of FPGA should be noticed in this research.

The image reconstruction algorithm should be improved in the future, since it stays in the simulation step, and a lot of problem would occur when operating measurements. Researchers need to keep in touch with those cutting-edge ideas and equipment in the research field of ground penetrating system.
Reference


Bow-Tie Antenna for Improved Pulse Radiation. *Ieee Transactions on Antennas and Propagation*, 58(7), 2184-2192.


**Appendix MATLAB Code**

It is modified from the code by *(Irving and Knight, 2006)*

function varargout = test3(varargin)

% TEST4 M-file for test4.fig
% TEST4, by itself, creates a new TEST4 or raises the existing singleton*.
%
% H = TEST4 returns the handle to a new TEST4 or the handle to the existing singleton*.
%
% TEST4('CALLBACK', hObject, eventData, handles, ..) calls the local function named CALLBACK in TEST4.M with the given input arguments.
%
% TEST4('Property', 'Value', ...) creates a new TEST4 or raises the existing singleton*. Starting from the left, property value pairs are applied to the GUI before test3_OpeningFcn gets called. An unrecognized property name or invalid value makes property application stop. All inputs are passed to test3_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE’s Tools menu. Choose “GUI allows only one instance to run (singleton)”.
%
% See also: GUIDE, GUIDATA, GUIDATAS

% Edit the above text to modify the response to help test4

% Last Modified by GUIDE v2.5 30-Mar-2011 15:58:18

% Begin initialization code - DO NOT EDIT

% gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'string', 'gui_Singleton', gui_Singleton, ...)
'gui_OpeningFcn', @test3_OpeningFcn, ...
'gui_OutputFcn', @test3_OutputFcn, ...
'gui_LayouFcn', [], ...
'gui_Callback', []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    {[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});}
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before test4 is made visible.
function test3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to test4 (see VARARGIN)

% Choose default command line output for test4
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes test4 wait for user response (see UIRESUME)
% uiwait(handles.test4);

% --- Outputs from this function are returned to the command line.
function varargin = test3_OutputFcn(hObject, eventdata, handles)
% varargin cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
--- Executes on slider movement.

function slider_per_Callback(hObject, eventdata, handles)
  % hObject    handle to slider_per (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)

  % Hints: get(hObject,'Value') returns position of slider
  %        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

  value=get(handles.slider_per, 'value');
  set(handles.edit_per,'string',num2str(value));
  guidata(hObject,handles);

  % --- Executes during object creation, after setting all properties.
  function slider_per_CreateFcn(hObject, eventdata, handles)
  % hObject    handle to slider_per (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    empty - handles not created until after all CreateFcns called

  % Hint: slider controls usually have a light gray background.
  if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
  end

  function edit_per_Callback(hObject, eventdata, handles)
  % hObject    handle to edit_per (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)

  % Hints: get(hObject,'String') returns contents of edit_per as text
  %        str2double(get(hObject,'String')) returns contents of edit_per as a double
  edit_value=get(handles.edit_per,'string');
  value1=str2num(edit_value);
  if (isempty(value1))||value1>100||value1<0
    set(handles.slider_per,'value',0);
    set(handles.edit_per, 'string',0);
  else
    set(handles.slider_per,'value',value1);
    set(handles.edit_per, 'string',value1);
  end
% --- Executes during object creation, after setting all properties.
function edit_per_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to edit_per (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

    % Hint: edit controls usually have a white background on Windows.
    %       See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
            get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

% --- Executes on button press in set.
function set_Callback(hObject, eventdata, handles)
    % hObject    handle to set (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    handles.fig=findobj('Type','figure','tag','test4');
    set(0,'CurrentFigure',handles.fig);
    handles.axes1=findobj('Type','axes','Tag','axes1');
    set(handles.fig,'CurrentAxes',handles.axes1);
    set(gca,'NextPlot','replacechildren');
    x=0:0.01:1;
    z=-0.03:0.01:0.47;
    mu0 = 1.2566370614e-6;
    ep0 = 8.8541878176e-12;
    mu = ones(size(ep));
    sig = 0.0892*ones(size(ep));
    ep(:,z<=0) = 1;
    sig(:,z<=0) = 0;

    value_per=get(handles.slider_per,'value');
    value_x=get(handles.slider_x,'value');
    value_z=get(handles.slider_z,'value');

    ep = 4*ones(length(x),length(z));
    mu = ones(size(ep));
    sig = 0.0892*ones(size(ep));
value_length=get(handles.slider_length,'value');
value_width=get(handles.slider_width,'value');

ep(x>=value_x & x<=(value_x+value_length),z>=value_z & z<=(value_z+value_width)) = value_per; % PLASTIC MINE
sig(x>=value_x & x<=(value_x+value_length),z>=value_z & z<=(value_z+value_width)) = 0;
imagesc(x,z,ep'); colormap('default'); axis image; axis on;
%X1=imagesc(x,z,ep');
%fig2 = figure(3);
%newax = copyobj(X1,fig2);
%savePlotWithinGUI(handles.axes1);
%saveas(handles.axes1,'example1.bmp','bmp');
guidata(hObject,handles);
% --- Executes on slider movement.
function slider_x_Callback(hObject, eventdata, handles)
  % hObject    handle to slider_x (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)

  % Hints: get(hObject,'Value') returns position of slider
  %        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
  value=get(handles.slider_x, 'value');
  set(handles.edit_x,'string',num2str(value));
guidata(hObject,handles);

% -- Executes during object creation, after setting all properties.
function slider_x_CreateFcn(hObject, eventdata, handles)
  % hObject    handle to slider_x (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    empty - handles not created until after all CreateFcsns called

  % Hint: slider controls usually have a light gray background.
  if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
  end

function edit_x_Callback(hObject, eventdata, handles)
  % hObject    handle to edit_x (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit_x as text
% str2double(get(hObject,'String')) returns contents of edit_x as a double

edit_value=get(handles.edit_x,'string');
value1=str2num(edit_value);
if (isempty(value1))||value1>1||value1<0
    set(handles.slider_x,'value',0);
    set(handles.edit_x, 'string',0);
else
    set(handles.slider_x,'value',value1);
end

% --- Executes during object creation, after setting all properties.
function edit_x_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_x (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_z_Callback(hObject, eventdata, handles)
% hObject    handle to edit_z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_z as text
% str2double(get(hObject,'String')) returns contents of edit_z as a double
edit_value=get(handles.edit_z,'string');
value1=str2num(edit_value);
if (isempty(value1))||value1>0.47||value1<-0.03
    set(handles.slider_z,'value',0);
    set(handles.edit_z, 'string',0);
else
    set(handles.slider_z,'value',value1);
end
% --- Executes during object creation, after setting all properties.
function edit_z_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to edit_z (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

    % Hint: edit controls usually have a white background on Windows.
    % See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
                get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function edit_length_Callback(hObject, eventdata, handles)
    % hObject    handle to edit_length (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'String') returns contents of edit_length as text
    % str2double(get(hObject,'String')) returns contents of edit_length as a double

    edit_value=get(handles.edit_length,'string');
    value1=str2num(edit_value);
    if (isempty(value1))||value1>1||value1<0
        set(handles.slider_length,'value',0);
        set(handles.edit_length, 'string',0);
    else
        set(handles.slider_length,'value',value1);
    end

    % --- Executes during object creation, after setting all properties.
function edit_length_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to edit_length (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

    % Hint: edit controls usually have a white background on Windows.
    % See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
                get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
function edit_width_Callback(hObject, eventdata, handles)
    % hObject    handle to edit_width (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    edit_value=get(handles.edit_width,'string');
    value1=str2num(edit_value);
    if (isempty(value1))||value1>0.5||value1<0
        set(handles.slider_width,'value',0);
        set(handles.edit_width, 'string',0);
    else
        set(handles.slider_width,'value',value1);
    end

    % --- Executes during object creation, after setting all properties.
    function edit_width_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to edit_width (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

    % Hint: edit controls usually have a white background on Windows.
    % See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

    % --- Executes on slider movement.
    function slider_z_Callback(hObject, eventdata, handles)
    % hObject    handle to slider_z (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'Value') returns position of slider
    %        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
    value=get(handles.slider_z, 'value');
    set(handles.edit_z,'string',num2str(value));
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function slider_z_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor', [0.9 0.9 0.9]);
end

% --- Executes on slider movement.
function slider_length_Callback(hObject, eventdata, handles)
% hObject    handle to slider_length (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
value=get(handles.slider_length, 'value');
set(handles.edit_length,'string',num2str(value));
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function slider_length_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_length (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor', [0.9 0.9 0.9]);
end

% --- Executes on slider movement.
function slider_width_Callback(hObject, eventdata, handles)
% hObject    handle to slider_width (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
value = get(handles.slider_width, 'value');
set(handles.edit_width, 'string', num2str(value));
guidata(hObject, handles);

function slider_width_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to slider_width (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

    % Hint: slider controls usually have a light gray background.
    if isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject, 'BackgroundColor', [.9 .9 .9]);
    end

function edit_con_Callback(hObject, eventdata, handles)
    % hObject    handle to edit_con (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: get(hObject, 'String') returns contents of edit_con as text
    % str2double(get(hObject, 'String')) returns contents of edit_con as a double
    edit_value = get(handles.edit_con, 'string');
    value1 = str2num(edit_value);
    if (isempty(value1)) || (value1 > 1) || (value1 < 0)
        set(handles.slider_con, 'value', 0);
        set(handles.edit_con, 'string', 0);
    else
        set(handles.slider_con, 'value', value1);
        set(handles.edit_con, 'string', value1);
    end

    % --- Executes during object creation, after setting all properties.
    function edit_con_CreateFcn(hObject, eventdata, handles)
        % hObject    handle to edit_con (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    empty - handles not created until after all CreateFcns called

        % Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function slider_con_Callback(hObject, eventdata, handles)
% hObject    handle to slider_con (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
value=get(handles.slider_con, 'value');
set(handles.edit_con,'string',num2str(value));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function slider_con_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_con (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on button press in RUN.

function RUN_Callback(hObject, eventdata, handles)
% hObject    handle to RUN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.fig=findobj('Type','figure','tag','test4');
set(0,'CurrentFigure',handles.fig);
handles.axes2=findobj('Type','axes','Tag','axes2');
set(handles.fig,'CurrentAxes',handles.axes2);
set(gca,'NextPlot','replacechildren');
set(handles.axes2,'Xlim',[0 100]);
set(handles.axes2,'Ylim',[0 50]);

x=0:0.01:1;
z=-0.03:0.01:0.47;
mu0 = 1.2566370614e-6;
ep0 = 8.8541878176e-12;

ep = 4*ones(length(x),length(z));%permittivity of dry soil
mu = ones(size(ep));
sig = 0.0892*ones(size(ep));%conductivity of the dry soil

ep(:,z<=0) = 1;
sig(:,z<=0) = 0;

value_per=get(handles.slider_per,'value');
value_x=get(handles.slider_x,'value');
value_z=get(handles.slider_z,'value');
value_length=get(handles.slider_length,'value');
value_width=get(handles.slider_width,'value');
disp(value_per);
disp(value_x);
disp(value_z);
disp(value_length);
disp(value_width);

ep(x>=value_x & x<=(value_x+value_length),z>=value_z & z<=(value_z+value_width)) = value_per;%PLASTIC MINE
sig(x>=value_x & x<=(value_x+value_length),z>=value_z & z<=(value_z+value_width)) = 0;

epmax=max(max(ep));
epmax=epmax*ep0;
epmin=min(min(ep));
epmin=epmin*ep0;
mumax=max(max(mu));
mumax=mumax*mu0;
mumin=min(min(mu));
mumin=mumin*mu0;
% compute the Blackman-Harris window as specified in Chen et al. (1997)
a = [0.35322222 -0.488 0.145 -0.01022222];
t=0:0.1e-10:100e-9;
fr = 2e9;
T = 1.14/fr;
window = zeros(size(t));
for n=0:3
    window = window + a(n+1)*cos(2*n*pi*t./T);
end
window(t>=T) = 0;

% for the pulse, approximate the window's derivative and normalize
p = window(:)';
p = [window(2:end) 0] - window(1:end);
p = p./max(abs(p));

%axes(handles.axes3);
%subplot(2,1,1);
%plot(t,p);
%axis([0 8e-10 -1 1]);
%xlabel('Time (sec)');
%ylabel('Norm. Amplitude');
%grid on;

srcpulse = p;
n1 = 2^nextpow2(length(srcpulse));
W = abs(fftshift(fft(srcpulse,n1))));
W = W./max(W);
fn = 0.5/(t(2)-t(1));
df = 2.*fn/n1;
f = -fn:df:fn-df;
W = W(n1/2+1:end);
f = f(n1/2+1:end);

%subplot(2,1,2);
%plot(f,W);
%axis([0 10e9 0 1]);
%xlabel('Frequency (Hertz)');
%ylabel('Norm. Amplitude');
%grid on;

% determine the maximum allowable spatial discretization
% (5 grid points per minimum wavelength are needed to avoid dispersion)
thres=0.02;
fmax = f(max(find(W>=thres)))% 5.6885e+009
wlmin = 1/(fmax*sqrt(epmax*mumax));
dxmax = wlmin/5;
dzmax = dxmax;

% set dx and dz here (m) using the above results as a guide
dx = 0.005
dz = 0.005

% determine maximum allowable time step for numerical stability
dtmax = 6/7*sqrt(epmin*mumin/(1/dx^2 + 1/dz^2));

% set proper dt here (s) using the above results as a guide
dtmax = 1e-11

t1=0:dtmax:10e-9;
%t1=0:dtmax:12e-9;
%t1=100e-9;
a1 = [0.35322222 -0.488 0.145 -0.010222222];
fr1 = 2e9;
T1 = 1.14/fr1;
window1 = zeros(size(t1));
for n2=0:3
    window1 = window1 + a1(n2+1)*cos(2*n2*pi*t1./T1);
end
window1(t1>=T1) = 0;

% for the pulse, approximate the window's derivative and normalize
p1 = window1(:)';
p1 = window1(2:end) - window1(1:end);
p1 = p1./max(abs(p1));
srcpulse1 = p1;

disp('Interpolating electrical property matrices...');
disp(' ');
x2 = min(x):dx/2:max(x);
z2 = min(z):dx/2:max(z);
ep2=gridinterp(ep,x,z,x2,z2,'nearest');
mu2=gridinterp(mu,x,z,x2,z2,'nearest');
sig2=gridinterp(sig,x,z,x2,z2,'nearest');
% pad electrical property matrices for PML absorbing boundaries
npml = 10; % number of PML boundary layers
[ep3,x3,z3] = padgrid(ep2,x2,z2,2*npml+1);
[mu3,x3,z3] = padgrid(mu2,x2,z2,2*npml+1);
[sig3,x3,z3] = padgrid(sig2,x2,z2,2*npml+1);

% clear unnecessary matrices taking up memory
clear x x2 z z2 ep ep2 mu mu2 sig sig2

% create source and receiver location matrices
% (rows are [x location (m), z location (m)])
srcx = (0:0.05:0.95)';
%srcx = (0:0.03:0.97)';
%srcx = 10;
srcz = -0.03*ones(size(srcx));
recx = srcx + 0.05;
recz = srcz;
srcloc = [srcx srcz];
recloc = [recx recz];

% set some output and plotting parameters
outstep = 2;
plotopt = [1 50 0.002];

% pause
%disp('Press any key to begin simulation...');
%disp(' ');
%pause;

% run the simulation

tic;
%x=x3;
%z=z3;
ep=ep3;
mu=mu3;
sig=sig3;
xprop = x3;
zprop = z3;
srcpulse = srcpulse1;

if nargin == 10; outstep = 1; plotopt = [1 50 0.05]; end
if nargin == 11; plotopt = [1 50 0.05]; end

if size(mu) ~= size(ep) | size(sig) ~= size(ep); disp('ep, mu, and sig matrices must be the same size'); return; end
if [length(xprop), length(zprop)] ~= size(ep); disp('xprop and zprop are inconsistent with ep, mu, and sig'); return; end
if mod(size(ep,1),2)~=1 | mod(size(ep,2),2)~=1; disp('ep, mu, and sig must have an odd # of rows and columns'); return; end
if size(srcloc,2)~=2 | size(recloc,2)~=2; disp('srcloc and recloc matrices must have 2 columns'); return; end
if max(srcloc(:,1))>max(xprop) | min(srcloc(:,1))<min(xprop) | max(srcloc(:,2))>max(zprop) | min(srcloc(:,2))<min(zprop); disp('source vector out of range of modeling grid'); return; end
if max(recloc(:,1))>max(xprop) | min(recloc(:,1))<min(xprop) | max(recloc(:,2))>max(zprop) | min(recloc(:,2))<min(zprop); disp('receiver vector out of range of modeling grid'); return; end
if length(srcpulse)~=length(t1); disp('srcpulse and t vectors must have same # of points'); return; end
if npml>=length(xprop)/2 | npml>=length(zprop)/2; disp('too many PML boundary layers for grid'); return; end
if length(plotopt)~=3; disp('plotopt must be a 3 component vector'); return; end

% determine number of field nodes and discretization interval
% mu0 = 1.2566370614e-6;
% ep0 = 8.8541878176e-12;
% ep = ep*ep0;
% mu = mu*mu0;

nx = (length(xprop)+1)/2;    % maximum number of field nodes in the x-direction
nz = (length(zprop)+1)/2;    % maximum number of field nodes in the z-direction
dX = 2*(xprop(2)-xprop(1));  % electric and magnetic field spatial discretization in x (m)
dZ = 2*(zprop(2)-zprop(1));  % electric and magnetic field spatial discretization in z (m)
% x and z position vectors corresponding to Hx, Hz, and Ey field matrices
% (these field matrices are staggered in both space and time, and thus have different
coordinates)
xHx = xprop(2):dX:xprop(end-1);
zHx = zprop(1):dZ:zprop(end);
xHz = xprop(1):dX:xprop(end);
zHz = zprop(2):dZ:zprop(end-1);
xEy = xHx;
zEy = zHz;

% determine source and receiver (i,j) indices in Ey field matrix,
% and true coordinates of sources and receivers in numerical model (after discretization)
nsrc = size(srcloc,1);                          % number of sources
nrec = size(recloc,1);                          % number of receivers
for s=1:nsrc;
    % source x index in Ey field matrix
    temp = min(abs(xEy - srcloc(s,1))); % source x index in Ey field matrix
    srci(s) = find(temp == min(temp)); % true source x location
    srcx(s) = xEy(srci(s));
    % source z index in Ey field matrix
    temp = min(abs(zEy - srcloc(s,2))); % source z index in Ey field matrix
    srcj(s) = find(temp == min(temp)); % true source z location
    srcz(s) = zEy(srcj(s));
end
for r=1:nrec;
    % receiver x index in Ey field matrix
    temp = min(abs(xEy - recloc(r,1))); % receiver x index in Ey field matrix
    reci(r) = find(temp == min(temp)); % true receiver x location
    recx(r) = xEy(reci(r));
    % receiver z index in Ey field matrix
    temp = min(abs(zEy - recloc(r,2))); % receiver z index in Ey field matrix
    recj(r) = find(temp == min(temp)); % true receiver z location
    recz(r) = zEy(recj(r));
end

% determine time stepping parameters from supplied time vector
dt1 = t1(2)-t1(1);                                % temporal discretization
%dt1 = dtmax;
umit = length(t1);                          % number of iterations

% ---------------------------------------------------------------
% COMPUTE FDTD UPDATE COEFFICIENTS FOR ENTIRE SIMULATION GRID
% note:  these matrices are twice as large as the field component matrices
% (i.e., the same size as the electrical property matrices)
% ---------------------------------------------------------------
disp('Determining update coefficients for simulation region...')

% set the basic PML parameters, keeping the following in mind...
% - maximum sigma_x and sigma_z vary in heterogeneous media to damp waves most
efficiently
% - Kmax = 1 is the original PML of Berenger (1994); Kmax > 1 can be used to damp
evanescent waves
% - keep alpha = 0 except for highly elongated domains (see Roden and Gedney (2000))
m = 4; % PML Exponent (should be
between 3 and 4)
Kxmax = 5; % maximum value for PML
K_z parameter (must be >=1)
Kzmax = 5; % maximum value for PML
K_z parameter (must be >=1)
sigxmax = (m+1)./(150*pi*sqrt(ep./ep0)*dX); % maximum value for PML
sigma_x parameter
sigzmax = (m+1)./(150*pi*sqrt(ep./ep0)*dZ); % maximum value for PML
sigma_z parameter
alpha = 0; % alpha parameter for PML
(CFS)

% indices corresponding to edges of PML regions in electrical property grids
kpmlLout = 1; % x index for outside of PML region on
left-hand side
kpmlLin = 2*npml+2; % x index for inside of PML region on
left-hand side
kpmlRin = length(xprop)-(2*npml)+1; % x index for inside of PML region on
right-hand side
kpmlRout = length(xprop); % x index for outside of PML region on
right-hand side
lpmlTout = 1; % z index for outside of PML region at
the top
lpmlTin = 2*npml+2; % z index for inside of PML region at the
top
lpmlBin = length(zprop)-(2*npml)+1; % z index for inside of PML region at the
bottom
lpmlBout = length(zprop); % z index for outside of PML region at the
bottom

% determine the ratio between the distance into the PML and the PML thickness in x and
z directions
% done for each point in electrical property grids; non-PML regions are set to zero
xdel = zeros(length(xprop),length(zprop)); % initialize x
direction matrix
k = kpmlLout:kpm LIN; k = k(:); % left-hand
PML layer
xdel(k,:) = repmat(((kpmlLin-k)./(2*npml)),1,length(zprop));
k = kpmlRin:kpmlRout; k = k(:); % right-hand PML layer
xdel(k,:) = repmat(((k-kpmlRin)./(2*npml)),1,length(zprop));
zdel = zeros(length(xprop),length(zprop)); % initialize z direction matrix
l = lpmlTout:lpmlTin; % top PML layer
zdel(:,l) = repmat(((lpmlTin-l)./(2*npml)),length(xprop),1);

l = lpmlBin:lpmlBout; % bottom PML layer
zdel(:,l) = repmat(((l-lpmlBin)./(2*npml)),length(xprop),1);

% determine PML parameters at each point in the simulation grid
% (scaled to increase from the inside to the outside of the PML region)
% (interior non-PML nodes have sigx=sigz=0, Kx=Kz=1)
sigx = sigxmax.*xdel.^m;
sigz = sigzmax.*zdel.^m;
Kx = 1 + (Kxmax-1)*xdel.^m;
Kz = 1 + (Kzmax-1)*zdel.^m;

% determine FDTD update coefficients
Ca = (1-dt1*sig./(2*ep))./(1+dt1*sig./(2*ep));
Cbx = (dt1./ep)./((1+dt1*sig./(2*ep))*24*dX.*Kx);
Cbz = (dt1./ep)./((1+dt1*sig./(2*ep))*24*dZ.*Kz);
Cc = (dt1./ep).*((1+dt1*sig./(2*ep)));
Dbx = (dt1./(mu.*Kx*24*dX));
Dbz = (dt1./(mu.*Kz*24*dZ));
Dc = dt1./mu;
Bx = exp(-((sigx./Kx + alpha).*(dt1/ep0)));
Bz = exp(-((sigz./Kz + alpha).*(dt1/ep0)));
Ax = ((sigx./Kx + Kx.^2*alpha + 1e-20).*(Bx-1))./(24*dX);
Az = ((sigz./Kz + Kz.^2*alpha + 1e-20).*(Bz-1))./(24*dZ);

% clear unnecessary PML variables as they take up lots of memory
clear sigmax xdel zdel Kx Kz sigx sigz

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %
% RUN THE FDTD SIMULATION
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %

disp('Beginning FDTD simulation...')
% initialize gather matrix where data will be stored
gather_wo = zeros(fix((numit-1)/outstep)+1,nrec,nsrc);

% loop over number of sources
for s=1:nsrc

% zero all field matrices
Ey = zeros(nx-1,nz-1);          % Ey component of electric field
Hx = zeros(nx-1,nz);            % Hx component of magnetic field
Hz = zeros(nx,nz-1);            % Hz component of magnetic field
Eydiffx = zeros(nx,nz-1);       % difference for dEy/dx
Eydiffz = zeros(nx-1,nz);       % difference for dEy/dz
Hxdiffz = zeros(nx-1,nz-1);     % difference for dHx/dz
Hzdiffx = zeros(nx-1,nz-1);     % difference for dHz/dx
PEyx = zeros(nx-1,nz-1);        % psi_Eyx (for PML)
PEyz = zeros(nx-1,nz-1);        % psi_Eyz (for PML)
PHx = zeros(nx-1,nz);           % psi_Hx (for PML)
PHz = zeros(nx,nz-1);           % psi_Hz (for PML)

% time stepping loop
for it=1:numit

% update Hx component...

% determine indices for entire, PML, and interior regions in Hx and property grids
i = 2:nx-2; j = 3:nz-2;          % indices for all components in Hx matrix to update
k = 2*i; l = 2*j-1;             % corresponding indices in property grids
kp = k((k<=kpmlLin | k>=kpmlRin)); % corresponding property indices in PML region
lp = l((l<=lpmlTin | l>=lpmlBin));
ki = k((k>kpmlLin & k<kpmlRin)); % corresponding property indices in interior (non-PML) region
li = l((l>lpmlTin & l<lpmlBin));

% update to be applied to the whole Hx grid
Eydiffz(i,j) = -Ey(i,j+1) + 27*Ey(i,j) - 27*Ey(i,j-1) + Ey(i,j-2);
Hx(i,j) = Hx(i,j) - Dbz(k,l).*Eydiffz(i,j);
% update to be applied only to the PML region

\[ \text{PHx}(ip,j) = \text{Bz}(kp,l) .* \text{PHx}(ip,j) + \text{Az}(kp,l) .* \text{Eydiffz}(ip,j); \]
\[ \text{PHx}(ii,jp) = \text{Bz}(ki,lp) .* \text{PHx}(ii,jp) + \text{Az}(ki,lp) .* \text{Eydiffz}(ii,jp); \]
\[ \text{Hx}(ip,j) = \text{Hx}(ip,j) - \text{Dc}(kp,l) .* \text{PHx}(ip,j); \]
\[ \text{Hx}(ii,jp) = \text{Hx}(ii,jp) - \text{Dc}(ki,lp) .* \text{PHx}(ii,jp); \]

% update Hz component...

% determine indices for entire, PML, and interior regions in Hz and property grids

\[
i = 3: \text{nx}-2; \quad j = 2: \text{nz}-2; \quad \text{indices for all components in Hz grids}
\]
\[
k = 2*i-1; \quad l = 2*j; \quad \text{corresponding indices in property grids}
\]
\[
k_p = k((k<=kpmlLin \mid k>=kpmlRin)); \quad \text{corresponding property indices in PML region}
\]
\[
l_p = l((l<=lpmlTin \mid l>=lpmlBin)); \quad \text{corresponding property indices in interior (non-PML) region}
\]
\[
k_i = k((k>kpmlLin \wedge k<kpmlRin)); \quad \text{corresponding property indices in interior (non-PML) region}
\]
\[
l_i = l((l>lpmlTin \wedge l<lpmlBin)); \quad \text{corresponding property indices in interior (non-PML) region}
\]

% determine indices for entire, PML, and interior regions in Hz and property grids

\[
i = 3: \text{nx}-2; \quad j = 2: \text{nz}-2; \quad \text{indices for all components in Hz}
\]
\[
k = 2*i-1; \quad l = 2*j; \quad \text{corresponding indices in property grids}
\]
\[
k_p = k((k<=kpmlLin \mid k>=kpmlRin)); \quad \text{corresponding property indices in PML region}
\]
\[
l_p = l((l<=lpmlTin \mid l>=lpmlBin)); \quad \text{corresponding property indices in interior (non-PML) region}
\]
\[
k_i = k((k>kpmlLin \wedge k<kpmlRin)); \quad \text{corresponding property indices in interior (non-PML) region}
\]
\[
l_i = l((l>lpmlTin \wedge l<lpmlBin)); \quad \text{corresponding property indices in interior (non-PML) region}
\]

% update to be applied to the whole Hz grid

\[ \text{Eydiffx}(i,j) = -\text{Ey}(i+1,j) + 27*\text{Ey}(i,j) - 27*\text{Ey}(i-1,j) + \text{Ey}(i-2,j); \]
\[ \text{Hx}(i,j) = \text{Hz}(i,j) + \text{Dbx}(k,l) .* \text{Eydiffx}(i,j); \]

% update to be applied only to the PML region

\[ \text{PHz}(ip,j) = \text{Bx}(kp,l) .* \text{PHz}(ip,j) + \text{Ax}(kp,l) .* \text{Eydiffx}(ip,j); \]
\[ \text{PHz}(ii,jp) = \text{Bx}(ki,lp) .* \text{PHz}(ii,jp) + \text{Ax}(ki,lp) .* \text{Eydiffx(ii,jp);} \]
\[ \text{Hz}(ip,j) = \text{Hz}(ip,j) - \text{Dc}(kp,l) .* \text{PHz}(ip,j); \]
\[ \text{Hz}(ii,jp) = \text{Hz}(ii,jp) - \text{Dc}(ki,lp) .* \text{PHz}(ii,jp); \]

% update Ey component...

% determine indices for entire, PML, and interior regions in Ey and property grids

\[
i = 2: \text{nx}-2; \quad j = 2: \text{nz}-2; \quad \text{indices for all components in Ey}
\]
\[
i = 2: \text{nx}-2; \quad j = 2: \text{nz}-2; \quad \text{indices for all components in Ey}
\]
\( k = 2^{*}i; \quad l = 2^{*}j; \) \hspace{1cm} \% \text{corresponding indices in property grids}

\[ k_{p} = k(\{k \leq k_{pmlLin} \mid k \geq k_{pmlRin}\}); \] \hspace{1cm} \% \text{corresponding property indices in PML region}

\[ l_{p} = l(\{l \leq l_{pmlTin} \mid l \geq l_{pmlBin}\}); \] \hspace{1cm} \% \text{corresponding property indices in interior (non-PML) region}

\[ k_{i} = k(\{k > k_{pmlLin} \& k < k_{pmlRin}\}); \] \hspace{1cm} \% \text{corresponding property indices in interior (non-PML) region}

\[ l_{i} = l(\{l > l_{pmlTin} \& l < l_{pmlBin}\}); \] \hspace{1cm} \% \text{Ey indices in PML region}

\[ i_{i} = k_{i}/2; \quad j_{i} = l_{i}/2; \] \hspace{1cm} \% \text{Ey indices in interior (non-PML) region}

\[ \text{update to be applied to the whole Ey grid} \]

\[ H_{xdiffz}(i,j) = -H_{x}(i,j+2) + 27^{*}H_{x}(i,j+1) - 27^{*}H_{x}(i,j) + H_{x}(i,j-1); \]

\[ H_{zdiffx}(i,j) = -H_{z}(i+2,j) + 27^{*}H_{z}(i+1,j) - 27^{*}H_{z}(i,j) + H_{z}(i-1,j); \]

\[ E_{y}(i,j) = Ca(k,l)^{*}E_{y}(i,j) + C_{bx}(k,l)^{*}H_{zdiffx}(i,j); \]

\[ E_{y}(i,j) = E_{y}(i,j) + C_{c}(k_{p},l)^{*}(P_{E_{y}x}(i,j) - P_{E_{y}z}(i,j)); \]

\[ (\text{emulates infinitesimal Ey directed line source with current} = \text{srcpulse}) \]

\[ i_{s} = \text{srci}(s); \quad j_{s} = \text{srcj}(s); \]

\[ E_{y}(i,j) = E_{y}(i,j) + \text{srcpulse1}(it); \]

\% plot the Ey wavefield if necessary

\% if plotopt(1) == 1; \% if mod(it-1,plotopt(2)) == 0

\% add source pulse to Ey at source location

\% plot the Ey wavefield if necessary

\% if plotopt(1) == 1; \% if mod(it-1,plotopt(2)) == 0

\% add source pulse to Ey at source location
pause(0.01);
end
end

% record the results in gather matrix if necessary
if mod(it-1,outstep)==0
    tout((it-1)/outstep+1) = t1(it);
    for r=1:nrec
        gather_wo((it-1)/outstep+1,r,s) = Ey(reci(r),recj(r));
    end
end
end
end
end

disp(' ');
disp(['Total running time = ',num2str(toc/3600),', hours']);

% extract common offset reflection GPR data from multi-offset data cube and plot the results
for i=1:length(srcx);
    codata_wo(:,i) = gather_wo(:,i,i);
end
pos = (srcx+recx)/2;
%figure;
handles.fig=findobj('Type','figure','tag','test4');
set(0,'CurrentFigure',handles.fig);
handles.axes3=findobj('Type','axes','Tag','axes3');
set(handles.fig,'CurrentAxes',handles.axes3);
set(gca,'NextPlot','replacechildren');
imagesc(pos,tout*1e9,codata_wo);
axis([0 1 0 10]);
set(gca,'plotboxaspectratio',[1 1 1]);
caxis([-5e-4 5e-4]);
ax=gca;
colormap(ax,'gray');
xlabel('Position (m)');
ylabel('Time (ns)');
save codata_wo(:,i);
save gather_wo(:,i,i);

S1R30;
S6R35;  
S11R40;  
S16R45;  
S21R50;  
S26R55;  
S31R60;  
S36R65;  
S41R70;  
S46R75;  
S51R80;  
S56R85;  
S61R90;  
S66R95;  
S71R100;  

load SR61TX1_M14;  
load SR111TX1_M14;  
load SR161TX1_M14;  
load SR211TX1_M14;  
load SR251TX1_M14;  
load SR301TX1_M14;  
load SR351TX1_M14;  
load SR401TX1_M14;  
load SR451TX1_M14;  
load SR501TX1_M14;  
load SR551TX1_M14;  
load SR601TX1_M14;  
load SR651TX1_M14;  
load SR701TX1_M14;  
load SR751TX1_M14;  

SWARR1R_M14=SR61TX1_M14+SR111TX1_M14+SR161TX1_M14+SR211TX1_M14+SR251TX1_M14+SR301TX1_M14+SR351TX1_M14+SR401TX1_M14+SR451TX1_M14+SR501TX1_M14+SR551TX1_M14+SR601TX1_M14+SR651TX1_M14+SR701TX1_M14+SR751TX1_M14;  

%for e=1:length(d)  
%for f=1:length(d)  
%if d(e,f)<=0.0003  
%d(e,f)=0;  
%end  
%end  
%end
save SWARR1R_M14;

R1 = SWARR1R_M14;
for e=1:length(R1)
  for f=1:length(R1)
    if R1(e,f)<=0.65*(max(max(R1)))
      R1(e,f)=0;
    end
  end
end
save R1;

handles.fig=findobj('Type','figure','tag','test4');
set(0,'CurrentFigure',handles.fig);
handles.axes2=findobj('Type','axes','Tag','axes2');
set(handles.fig,'CurrentAxes',handles.axes2);
set(gca,'NextPlot','replacechildren');

%figure(3);
X2=imagesc(X1,Y1,R1);
axis image;
title('LOCATION OF MINE M14 USING COMMON OFFSET Simulated DATA AQUISITION');
axis([0 100 -4 47]);
xlabel('Position (cm)');
ylabel('Depth (cm)');
axis on;
guidata(hObject,handles);

% --- Executes on selection change in example.
function example_Callback(hObject, eventdata, handles)
% hObject    handle to example (see GCBO)
% eventdata  reserved to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns example contents as cell array
%        contents{get(hObject,'Value')} returns selected item from example

switch get(handles.example,'Value')
  case 2
    example0;
case 3
  example1;

case 4
  example2;

case 5
  example3;

case 6
  example4;

otherwise
end

---

Executes during object creation, after setting all properties.

function example_CreateFcn(hObject, eventdata, handles)
  hObject    handle to example (see GCBO)
  eventdata   reserved - to be defined in a future version of MATLAB
  handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
  get(0,'defaultUicontrolBackgroundColor'))
  set(hObject,'BackgroundColor','white');
end

---

Executes on button press in pushbutton10.

function pushbutton10_Callback(hObject, eventdata, handles)
  hObject    handle to pushbutton10 (see GCBO)
  eventdata   reserved - to be defined in a future version of MATLAB
  handles    structure with handles and user data (see GUIDATA)
  handles.fig=findobj('Type','figure','tag','test4');
  set(hObject,'CurrentFigure',handles.fig);
  handles.axes2=findobj('Type','axes','Tag','axes2');
  set(hObject,'CurrentAxes',handles.axes2);
  set(gca,'NextPlot','replacechildren');

set(handles.axes2,'Xlim',[0 100]);
set(handles.axes2,'Ylim',[0 50]);

x=0:0.01:1;
z=-0.03:0.01:0.47;
mu0 = 1.2566370614e-6;
ep0 = 8.8541878176e-12;
ep = 4*ones(length(x),length(z));% permitivity of dry soil
mu = ones(size(ep));
sig = 0.0892*ones(size(ep));% conductivity of the dry soil%
sig = ones(size(ep));
ep(:,z<=0) = 1;
sig(:,z<=0) = 0;

value_per=get(handles.slider_per,'value');
value_x=get(handles.slider_x,'value');
value_z=get(handles.slider_z,'value');
value_length=get(handles.slider_length,'value');
value_width=get(handles.slider_width,'value');
disp(value_per);
disp(value_x);
disp(value_z);
disp(value_length);
disp(value_width);

ep(x>=value_x & x<=(value_x+value_length),z>=value_z & z<=(value_z+value_width)) = value_per; % PLASTIC MINE
sig(x>=value_x & x<=(value_x+value_length),z>=value_z & z<=(value_z+value_width)) = 0;

epmax=max(max(ep));
epmax=epmax*ep0;
epmin=min(min(ep));
epmin=epmin*ep0;
mumax=max(max(mu));
mumax=mumax*mu0;
mumin=min(min(mu));
mumin=mumin*mu0;

% compute the Blackman-Harris window as specified in Chen et al. (1997)
a = [0.35322222 -0.488 0.145 -0.010222222];
t=0:0.1e-10:100e-9;
fr = 2e9;
T = 1.14/fr;
window = zeros(size(t));
for n=0:3
    window = window + a(n+1)*cos(2*n*pi*t./T);
end
window(t>=T) = 0;

% for the pulse, approximate the window's derivative and normalize
p = window(:)';
p = [window(2:end) 0] - window(1:end);
p = p./max(abs(p));

%axes(handles.axes3);
%subplot(2,1,1);
%plot(t,p);
%axis([0 8e-10 -1 1]);
%xlabel('Time (sec)');
%ylabel('Norm. Amplitue');
%grid on;

srcpulse = p;
n1 = 2^nextpow2(length(srcpulse));
W = abs(fftshift(fft(srcpulse,n1)));
W = W./max(W);
fn = 0.5/(t(2)-t(1));
df = 2.*fn/n1;
f = -fn:df:fn-df;
W = W(n1/2+1:end);
f = f(n1/2+1:end);

%subplot(2,1,2);
%plot(f,W);
%axis([0 10e9 0 1]);
%xlabel('Frequency (Hertz)');
%ylabel('Norm. Amplitue');
%grid on;

% determine the maximum allowable spatial disretization
% (5 grid points per minimum wavelength are needed to avoid dispersion)
thres=0.02;
fmax = f(max(find(W>=thres)));
wmin = 1/(fmax*sqrt(epmax*mumax));
dxmax = wmin/5;
dzmax = dxmax;

% set dx and dz here (m) using the above results as a guide
dx = 0.005
dz = 0.005
% determine maximum allowable time step for numerical stability
\[ dt_{\text{max}} = \frac{6}{7} \sqrt{\varepsilon_{\text{min}} \mu_{\text{min}} / (1/dx^2 + 1/dz^2)}; \]

% set proper dt here (s) using the above results as a guide
\[ dt_{\text{max}} = 1 \times 10^{-11}; \]

t1=0:dtmax:10e-9;
% t1=0:dtmax:12e-9;
% t1=100e-9;
\[ a1 = [0.35322222 -0.488 0.145 -0.010222222]; \]
\[ fr1 = 2e9; \]
\[ T1 = 1.14/fr1; \]
\[ \text{window1} = \text{zeros(size(t1))}; \]
for n2=0:3
    \[ \text{window1} = \text{window1} + a1(n2+1) \times \cos(2 \times n2 \times \pi \times t1./T1); \]
end
\[ \text{window1}(t1>=T1) = 0; \]

% for the pulse, approximate the window's derivative and normalize
\[ p1 = \text{window1}(:)'; \]
\[ p1 = [\text{window1}(2:end) 0] - \text{window1}(1:end); \]
\[ p1 = p1./\text{max(abs(p1))}; \]
\[ \text{srcpulse1} = p1; \]

disp('Interpolating electrical property matrices...');
disp('');
\[ x2 = \text{min}(x):dx/2:\text{max}(x); \]
\[ z2 = \text{min}(z):dx/2:\text{max}(z); \]
\[ \text{ep2} = \text{gridinterp}(\text{ep},x,z,x2,z2,'\text{nearest}'); \]
\[ \text{mu2} = \text{gridinterp}(\text{mu},x,z,x2,z2,'\text{nearest}'); \]
\[ \text{sig2} = \text{gridinterp}(\text{sig},x,z,x2,z2,'\text{nearest}'); \]

% pad electrical property matrices for PML absorbing boundaries
\[ \text{npml} = 10; \]
\[ \text{[ep3,x3,z3]} = \text{padgrid(ep2,x2,z2,2*\text{npml}+1)}; \]
\[ \text{[mu3,x3,z3]} = \text{padgrid(mu2,x2,z2,2*\text{npml}+1)}; \]
\[ \text{[sig3,x3,z3]} = \text{padgrid(sig2,x2,z2,2*\text{npml}+1)}; \]
% clear unnecessary matrices taking up memory
clear x x2 z z2 ep ep2 mu mu2 sig sig2

% create source and receiver location matrices
% (rows are [x location (m), z location (m)])
srcx = (0:0.05:0.95)';
%srcx = (0:0.03:0.97)';
%srcx = 10;
srcz = -0.03*ones(size(srcx));
recx = srcx + 0.05;
recz = srcz;
srcloc = [srcx srcz];
recloc = [recx recz];

% set some output and plotting parameters
outstep = 2;
plotopt = [1 50 0.002];

% pause
%disp('Press any key to begin simulat...');
%disp('');
%pause;

% run the simulation

tic;
%x=x3;
%z=z3;
ep=ep3;
mu=mu3;
sig=sig3;
xprop=x3;
zprop=z3;
srclpuls=srclpuls1;

%if nargin==10; outstep=1; plotopt=[1 50 0.05]; end
%if nargin==11; plotopt=[1 50 0.05]; end
if size(mu)==size(ep) | size(sig)==size(ep); disp('ep, mu, and sig matrices must be the same size'); return; end
if [length(xprop),length(zprop)]~=size(ep); disp('xprop and zprop are inconsistent with ep, mu, and sig'); return; end
if mod(size(ep,1),2)~=1 | mod(size(ep,2),2)~=1; disp('ep, mu, and sig must have an odd # of rows and columns'); return; end
if size(srcloc,2)~=2 | size(recloc,2)~=2; disp('srcloc and recloc matrices must have 2 columns'); return; end
if max(srcloc(:,1))>max(xprop) | min(srcloc(:,1))<min(xprop) |
  max(srcloc(:,2))>max(zprop)... |
  min(srcloc(:,2))<min(zprop); disp('source vector out of range of modeling grid'); return; end
if max(recloc(:,1))>max(xprop) | min(recloc(:,1))<min(xprop) |
  max(recloc(:,2))>max(zprop)... |
  min(recloc(:,2))<min(zprop); disp('receiver vector out of range of modeling grid'); return; end
if length(srcpulse)~=length(t1); disp('srcpulse and t vectors must have same # of points'); return; end
if npml>=length(xprop)/2 | npml>=length(zprop)/2; disp('too many PML boundary layers for grid'); return; end
if length(plotopt)~=3; disp('plotopt must be a 3 component vector'); return; end

% determine number of field nodes and discretization interval
%mu0 = 1.2566370614e-6;
%ep0 = 8.8541878176e-12;
ep=ep*ep0;
mu=mu*mu0;

nx = (length(xprop)+1)/2; % maximum number of field nodes in the x-direction
nz = (length(zprop)+1)/2; % maximum number of field nodes in the z-direction
dX = 2*(xprop(2)-xprop(1)); % electric and magnetic field spatial discretization in x (m)
dZ = 2*(zprop(2)-zprop(1)); % electric and magnetic field spatial discretization in z (m)

% x and z position vectors corresponding to Hx, Hz, and Ey field matrices
% (these field matrices are staggered in both space and time, and thus have different coordinates)
xHx = xprop(2):dX:xprop(end-1);
zHx = zprop(1):dZ:zprop(end);
xHz = xprop(1):dX:xprop(end);
zHz = zprop(2):dZ:zprop(end-1);
xEy = xHx;
\[ \text{zEy = zHz}; \]

% determine source and receiver (i,j) indices in Ey field matrix, 
% and true coordinates of sources and receivers in numerical model (after discretization)
nsrc = size(srcloc,1);                          % number of sources
nrec = size(recloc,1);                          % number of receivers
for s=1:nsrc;
    [temp,srci(s)] = min(abs(xEy - srcloc(s,1))); % source x index in Ey field matrix
    [temp,srcj(s)] = min(abs(zEy - srcloc(s,2))); % source z index in Ey field matrix
    srcx(s) = xEy(srci(s));                     % true source x location
    srcz(s) = zEy(srcj(s));                     % true source z location
end
for r=1:nrec;
    [temp,reci(r)] = min(abs(xEy - recloc(r,1))); % receiver x index in Ey field matrix
    [temp,recj(r)] = min(abs(zEy - recloc(r,2))); % receiver z index in Ey field matrix
    recx(r) = xEy(reci(r));                     % true receiver x location
    recz(r) = zEy(recj(r));                     % true receiver z location
end

% determine time stepping parameters from supplied time vector
dt1 = t1(2)-t1(1);                             % temporal discretization
%dt1 = dtmax;
numit = length(t1);                              % number of iterations

% COMPUTE FDTD UPDATE COEFFICIENTS FOR ENTIRE SIMULATION GRID
% note:  these matrices are twice as large as the field component matrices
% (i.e., the same size as the electrical property matrices)

% set the basic PML parameters, keeping the following in mind...
% - maximum sigma_x and sigma_z vary in heterogeneous media to damp waves most efficiently
% - Kmax = 1 is the original PML of Berenger (1994); Kmax > 1 can be used to damp evanescent waves
% - keep alpha = 0 except for highly elongated domains (see Roden and Gedney (2000))
m = 4;                                          % PML Exponent (should be between 3 and 4)
Kxmax = 5;                                      % maximum value for PML
Kzmax = 5;                                      % maximum value for PML
\( K_z \) parameter (must be \( \geq 1 \))

\[
\text{sigxmax} = \frac{(m+1)}{(150 \pi \sqrt{\frac{\varepsilon_p}{\varepsilon_0}}) dX}; \quad \% \text{maximum value for PML}\n\]

\( \sigma_x \) parameter

\[
\text{sigzmax} = \frac{(m+1)}{(150 \pi \sqrt{\frac{\varepsilon_p}{\varepsilon_0}}) dZ}; \quad \% \text{maximum value for PML} \sigma_z \text{ parameter}\n\]

\( \alpha = 0; \quad \% \text{alpha parameter for PML} \)

\( \text{(CFS)} \)

\% indices corresponding to edges of PML regions in electrical property grids
\n\[
k_{pmlLout} = 1; \quad \% x \text{ index for outside of PML region on left-hand side}
k_{pmlLin} = 2n_{pml} + 2; \quad \% x \text{ index for inside of PML region on left-hand side}
k_{pmlRin} = \text{length}(xprop)-2n_{pml} + 2; \quad \% x \text{ index for inside of PML region on right-hand side}
k_{pmlRout} = \text{length}(xprop); \quad \% x \text{ index for outside of PML region on right-hand side}
l_{pmlTout} = 1; \quad \% z \text{ index for outside of PML region at the top}
l_{pmlTin} = 2n_{pml} + 2; \quad \% z \text{ index for inside of PML region at the top}
l_{pmlBin} = \text{length}(zprop)-2n_{pml} + 2; \quad \% z \text{ index for inside of PML region at the bottom}
l_{pmlBout} = \text{length}(zprop); \quad \% z \text{ index for outside of PML region at the bottom}\n\]

\% determine the ratio between the distance into the PML and the PML thickness in \( x \) and \( z \) directions
\% done for each point in electrical property grids; \( \) non-PML regions are set to zero
\[
x_{del} = \text{zeros}(\text{length}(xprop), \text{length}(zprop)); \quad \% \text{initialize } x \text{ direction matrix}
k = k_{pmlLout}:k_{pmlLin}; \quad k = k(:,); \quad \% \text{left-hand PML layer}
x_{del}(k,:) = \text{repmat}(((k_{pmlLin}-k)./(2n_{pml})),1,\text{length}(zprop));
k = k_{pmlRin}:k_{pmlRout}; \quad k = k(:,); \quad \% \text{right-hand PML layer}
x_{del}(k,:) = \text{repmat}(((k-k_{pmlRin})./(2n_{pml})),1,\text{length}(zprop));
z_{del} = \text{zeros}(\text{length}(xprop), \text{length}(zprop)); \quad \% \text{initialize } z \text{ direction matrix}
l = l_{pmlTout}:l_{pmlTin}; \quad \% \text{top PML layer}
z_{del}(l,:) = \text{repmat}(((l_{pmlTin}-l)./(2n_{pml})),\text{length}(xprop),1);
l = l_{pmlBin}:l_{pmlBout}; \quad \% \text{bottom}
PML layer
zdel(:,1) = repmat(((l-lpmlBin)./(2*npml)),length(xprop),1);

% determine PML parameters at each point in the simulation grid
% (scaled to increase from the inside to the outside of the PML region)
% (interior non-PML nodes have sigx=sigz=0, Kx=Kz=1)
sigx = sigxmax.*xdel.^m;
sigz = sigzmax.*zdel.^m;
Kx = 1 + (Kxmax-1)*xdel.^m;
Kz = 1 + (Kzmax-1)*zdel.^m;

% determine FDTD update coefficients
Ca  = (1-dt1*sig./(2*ep))./(1+dt1*sig./(2*ep));
Cbx  = (dt1./ep)./((1+dt1*sig./(2*ep))*24*dX.*Kx);
Cbz  = (dt1./ep)./((1+dt1*sig./(2*ep))*24*dZ.*Kz);
Cc = (dt1./ep)./(1+dt1*sig./(2*ep));
Dbx = (dt1./(mu.*Kx*24*dX));
Dbz = (dt1./(mu.*Kz*24*dZ));
Dc = dt1./mu;
Bx = exp(-(sigx./Kx + alpha)*(dt1/ep0));
Bz = exp(-(sigz./Kz + alpha)*(dt1/ep0));
Ax = (sigx./sigx.*Kx + Kx.^2*alpha + 1e-20).*(Bx-1)./(24*dX);
Az = (sigz./sigz.*Kz + Kz.^2*alpha + 1e-20).*(Bz-1)./(24*dZ);

% clear unnecessary PML variables as they take up lots of memory
clear sigmax xdel zdel Kx Kz sigx sigz

% -----------------------------------------------------------------
% RUN THE FDTD SIMULATION
% -----------------------------------------------------------------

disp('Beginning FDTD simulation...')

% initialize gather matrix where data will be stored
gather_wo = zeros(fix((numit-1)/outstep)+1,nrec,nsrc);

% loop over number of sources
for s=1:nsrc

% zero all field matrices
Ey = zeros(nx-1,nz-1);  % Ey component of electric field
Hx = zeros(nx-1,nz);    % Hx component of magnetic field

90
\text{Hz} = \text{zeros(nx,nz-1)}; \quad \% \text{Hz component of magnetic field}
\text{Eydiffx} = \text{zeros(nx,nz-1)}; \quad \% \text{difference for dEy/dx}
\text{Eydiffz} = \text{zeros(nx-1,nz);} \quad \% \text{difference for dEy/dz}
\text{Hxdiffz} = \text{zeros(nx-1,nz-1);} \quad \% \text{difference for dHx/dz}
\text{Hzdiffx} = \text{zeros(nx-1,nz-1);} \quad \% \text{difference for dHz/dx}
\text{PEyx} = \text{zeros(nx-1,nz-1);} \quad \% \text{psi}_{\text{Eyx}} \text{ (for PML)}
\text{PEyz} = \text{zeros(nx-1,nz-1);} \quad \% \text{psi}_{\text{Eyz}} \text{ (for PML)}
\text{PHx} = \text{zeros(nx-1,nz);} \quad \% \text{psi}_{\text{Hx}} \text{ (for PML)}
\text{PHz} = \text{zeros(nx,nz-1);} \quad \% \text{psi}_{\text{Hz}} \text{ (for PML)}

\% \text{time stepping loop}
\text{for it=1:numit}
\% \text{for it=1:1}
\% \text{update Hx component...}
\% \text{determine indices for entire, PML, and interior regions in Hx and property grids}
i = 2:nx-2; \quad j = 3:nz-2; \quad \% \text{indices for all components in Hx matrix to update}
k = 2*i; \quad l = 2*j-1; \quad \% \text{corresponding indices in property grids}
\text{kp} = k((k<=kpmlLin \text{ | } k>=kpmlRin)); \quad \% \text{corresponding property indices in PML region}
\text{lp} = l((l<=lpmlTin \text{ | } l>=lpmlBin));
\text{ki} = k((k>kpmlLin \& k<kpmlRin)); \quad \% \text{corresponding property indices in interior (non-PML) region}
\text{li} = l((l>lpmlTin \& l<lpmlBin));
\text{ip} = kp./2; \quad \text{jp} = (lp+1)./2; \quad \% \text{Hx indices in PML region}
\text{ii} = ki./2; \quad \text{ji} = (li+1)./2; \quad \% \text{Hx indices in interior (non-PML) region}
\% \text{update to be applied to the whole Hx grid}
\text{Eydiffz(i,j)} = -\text{Ey(i,j+1)} + 27*\text{Ey(i,j)} - 27*\text{Ey(i,j-1)} + \text{Ey(i,j-2)};
\text{Hx(i,j)} = \text{Hx(i,j)} - Dbz(k,l).*\text{Eydiffz(i,j)};
\% \text{update to be applied only to the PML region}
\text{PHx(ip,j)} = \text{Bz(kp,l)}.*\text{PHx(ip,j)} + \text{Az(kp,l)}.*\text{Eydiffz(ip,j)};
\text{PHx(ii,jp)} = \text{Bz(ki,lp)}.*\text{PHx(ii,jp)} + \text{Az(ki,lp)}.*\text{Eydiffz(ii,jp)};
\text{Hx(ip,j)} = \text{Hx(ip,j)} - Dc(kp,l).*\text{PHx(ip,j)};
\text{Hx(ii,jp)} = \text{Hx(ii,jp)} - Dc(ki,lp).*\text{PHx(ii,jp)};
\% \text{update Hz component...}
% determine indices for entire, PML, and interior regions in Hz and property
% grids
i = 3:nx-2; j = 2:nz-2; % indices for all components in Hz
matrix to update
k = 2*i-1; l = 2*j; % corresponding indices in
property grids
kp = k((k<=kpmlLin | k>=kpmlRin)); % corresponding property indices
in PML region
lp = l((l<=lpmlTin | l>=lpmlBin));
ki = k((k>kpmlLin & k<kpmlRin)); % corresponding property indices
in interior (non-PML) region
li = l((l>lpmlTin & l<lpmlBin));

% Hz indices in PML region
ip = kp./2; jp = lp./2; % Hz indices in interior (non-PML)
region

% update to be applied to the whole Hz grid
Eydiffx(i,j) = -Ey(i+1,j) + 27*Ey(i,j) - 27*Ey(i-1,j) + Ey(i-2,j);
Hz(i,j) = Hz(i,j) + Dbx(k,l).*Eydiffx(i,j);

% update to be applied only to the PML region
PHz(ip,j) = Bx(kp,l).*PHz(ip,j) + Ax(kp,l).*Eydiffx(ip,j);
PHz(ii,jp) = Bx(ki,lp).*PHz(ii,jp) + Ax(ki,lp).*Eydiffx(ii,jp);
Hz(ip,j) = Hz(ip,j) + Dc(kp,l).*PHz(ip,j);
Hz(ii,jp) = Hz(ii,jp) + Dc(ki,lp).*PHz(ii,jp);

% update Ey component...

% determine indices for entire, PML, and interior regions in Ey and property
% grids
i = 2:nx-2; j = 2:nz-2; % indices for all components in Ey
matrix to update
k = 2*i; l = 2*j; % corresponding indices in
property grids
kp = k((k<=kpmlLin | k>=kpmlRin)); % corresponding property indices
in PML region
lp = l((l<=lpmlTin | l>=lpmlBin));
ki = k((k>kpmlLin & k<kpmlRin)); % corresponding property indices
in interior (non-PML) region
li = l((l>lpmlTin & l<lpmlBin));

% Ey indices in PML region
ip = kp./2; jp = lp./2; % Ey indices in PML region

\[ ii = ki./2; \quad ji = li./2; \quad \% \text{ Ey indices in interior (non-PML) region} \]

\% update to be applied to the whole Ey grid
\[
\text{Hxdiffz}(i,j) = -\text{Hx}(i,j+2) + 27\times\text{Hx}(i,j+1) - 27\times\text{Hx}(i,j) + \text{Hx}(i,j-1); \\
\text{Hzdiffx}(i,j) = -\text{Hz}(i+2,j) + 27\times\text{Hz}(i+1,j) - 27\times\text{Hz}(i,j) + \text{Hz}(i-1,j); \\
\text{Ey}(i,j) = \text{Ca}(k,l)\times\text{Ey}(i,j) + \text{Cbx}(k,l)\times\text{Hzdiffx}(i,j) - \text{Cbz}(k,l)\times\text{Hxdiffz}(i,j); \\
\]

\% update to be applied only to the PML region
\[
\text{PEyx}(ip,j) = \text{Bx}(kp,l)\times\text{PEyx}(ip,j) + \text{Ax}(kp,l)\times\text{Hzdiffx}(ip,j); \\
\text{PEyx}(ii,jp) = \text{Bx}(ki,lp)\times\text{PEyx}(ii,jp) + \text{Ax}(ki,lp)\times\text{Hzdiffx}(ii,jp); \\
\text{PEyz}(ip,j) = \text{Bz}(kp,l)\times\text{PEyz}(ip,j) + \text{Az}(kp,l)\times\text{Hxdiffz}(ip,j); \\
\text{PEyz}(ii,jp) = \text{Bz}(ki,lp)\times\text{PEyz}(ii,jp) + \text{Az}(ki,lp)\times\text{Hxdiffz}(ii,jp); \\
\text{Ey}(ip,j) = \text{Ey}(ip,j) + \text{Cc}(kp,l)\times(\text{PEyx}(ip,j) - \text{PEyz}(ip,j)); \\
\text{Ey}(ii,jp) = \text{Ey}(ii,jp) + \text{Cc}(ki,lp)\times(\text{PEyx}(ii,jp) - \text{PEyz}(ii,jp)); \\
\]

\% add source pulse to Ey at source location
\% (emulates infinitesimal Ey directed line source with current = srcpulse)
i = srci(s); j = srcj(s); 
\text{Ey}(i,j) = \text{Ey}(i,j) + \text{srcpulse}(1); \\
\]

\% plot the Ey wavefield if necessary
if plotopt(1)==1; 
\quad if mod(it-1,plotopt(2))==0
\quad \quad disp([''Source '',num2str(s),'',''/'',num2str(nsrc),''], Iteration '','num2str(it),''/'',num2str(numit),''],
\quad \quad \quad ' t = '',num2str(t1(it)*1e9),'' ns'])
\quad \quad imagesc(xEy,zEy,Ey'); axis image
\quad \quad title([''Source '',num2str(s),'',''/'',num2str(nsrc),''], Iteration '','num2str(it),''/'',num2str(numit),''],
\quad \quad \quad ' Ey wavefield at t = '',num2str(t1(it)*1e9),'' ns']);
\quad \quad xlabel('Position (m)'); ylabel('Depth (m)');
\quad \quad caxis([-plotopt(3) plotopt(3)]);
\quad \quad pause(0.01); 
\quad end 
end 

\% record the results in gather matrix if necessary
if mod(it-1,outstep)==0 
\quad tout((it-1)/outstep+1) = t1(it);
\quad for r=1:nrec 
\quad \quad gather_wo((it-1)/outstep+1,r,s) = Ey(reci(r),recj(r)); 
\quad end 
end
end
end
end

disp('');
disp(['Total running time = ',num2str(toc/3600),' hours']);

% extract common offset reflection GPR data from multi-offset data cube and plot the results
for i=1:length(srcx);
    codata_example2axes3(:,i) = gather_wo(:,i,i);
end
pos = (srcx+recx)/2;
%figure;
handles.fig=findobj('Type','figure','tag','test4');
set(0,'CurrentFigure',handles.fig);
handles.axes3=findobj('Type','axes','Tag','axes3');
set(handles.fig,'CurrentAxes',handles.axes3);
set(gca,'NextPlot','replacechildren');
imagesc(pos,tout*1e9,codata_example2axes3);
axis([0 1 0 10]);
set(gca,'plotboxaspectratio',[1 1 1]);
caxis([-5e-4 5e-4]);
ax=gca;
colormap(ax,'gray');
xlabel('Position (m)');
ylabel('Time (ns)');
save codata_example2axes3(:,i);
save gather_wo(:,i,i);

S1R30;
S6R35;
S11R40;
S16R45;
S21R50;
S26R55;
S31R60;
S36R65;
S41R70;
S46R75;
S51R80;
S56R85;
S61R90;
S66R95;
S71R100;

load SR61TX1_M14;
load SR111TX1_M14;
load SR161TX1_M14;
load SR211TX1_M14;
load SR251TX1_M14;
load SR301TX1_M14;
load SR351TX1_M14;
load SR401TX1_M14;
load SR451TX1_M14;
load SR501TX1_M14;
load SR551TX1_M14;
load SR601TX1_M14;
load SR651TX1_M14;
load SR701TX1_M14;
load SR751TX1_M14;

SWARR1R_M14=SR61TX1_M14+SR111TX1_M14+SR161TX1_M14+SR211TX1_M14+SR251TX1_M14+SR301TX1_M14+SR351TX1_M14+SR401TX1_M14+SR451TX1_M14+SR501TX1_M14+SR551TX1_M14+SR601TX1_M14+SR651TX1_M14+SR701TX1_M14+SR751TX1_M14;

%for e=1:length(d)
%for f=1:length(d)
%if d(e,f)<=0.0003
%d(e,f)=0;
%end
%end

save SWARR1R_M14;

example2axes2 = SWARR1R_M14;
for e=1:length(example2axes2)
for f=1:length(example2axes2)
if example2axes2(e,f)<=0.65*(max(max(example2axes2)))
example2axes2(e,f)=0;
end
end
end
end
save example2axes2;

handles.fig=findobj('Type','figure','tag','test4');
set(0,'CurrentFigure',handles.fig);
handles.axes2=findobj('Type','axes','Tag','axes2');
set(handles.fig,'CurrentAxes',handles.axes2);
set(gca,'NextPlot','replacechildren');

%figure(3);
imagesc(X1,Y1,example2axes2);
axis image;
title('LOCATION OF MINE M14 USING COMMON OFFSET Simulated DATA AQUISITION');
axis([0 100 -4 47]);
xlabel('Position (cm)');
ylabel('Depth (cm)');
axis on;
guida(hObject,handles);