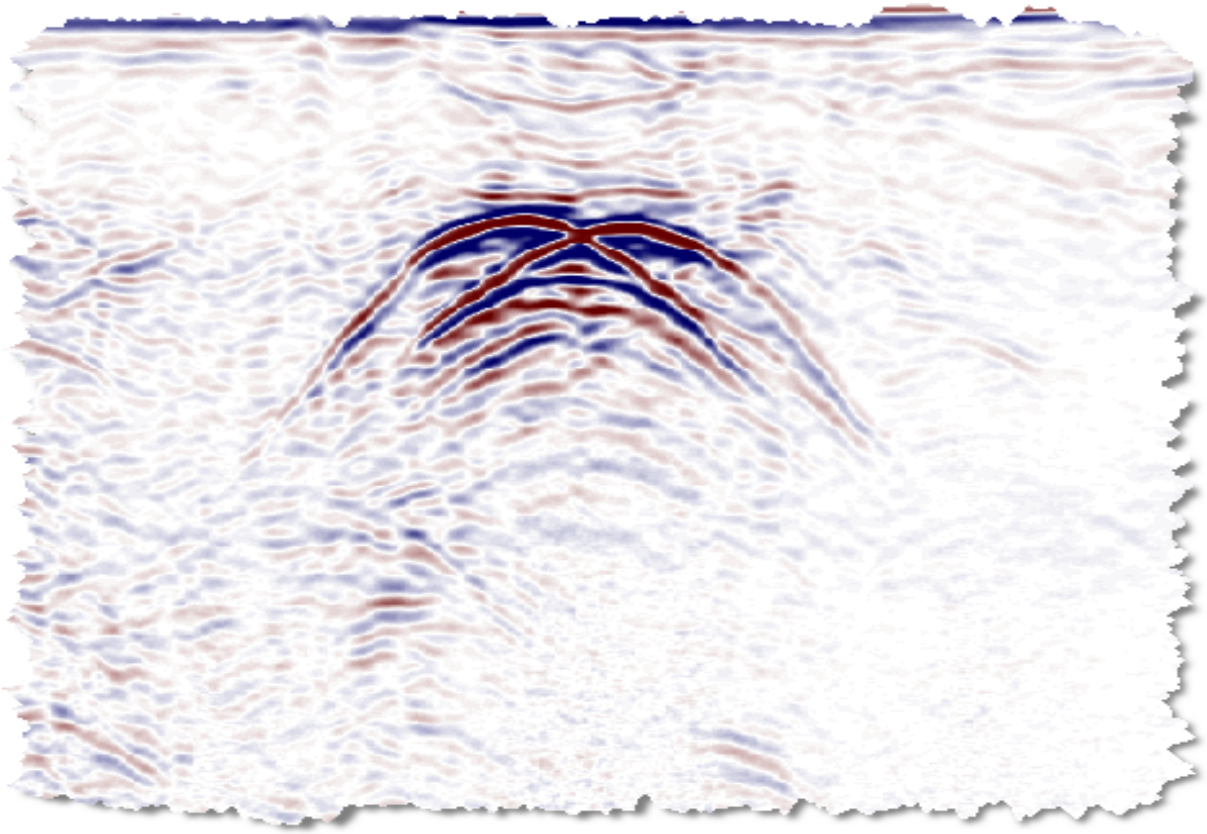
MATGPR

Release 3.5
Manual and Technical Reference

by

Andreas Tzanis, PhD

Department of Geophysics
University of Athens



Athens, Greece, March 2016

LICENSE

Dr Andreas Tzanis
Department of Geophysics,
University of Athens
Panepistimiopoli,
Zografou 15784
Greece
Tel: +30-210-727-4785
E-mail: atzanis@geol.uoa.gr

CONTENTS

CHAPTER 1. PLEAMBLE

1.1. Why matGPR?

The Ground Probing Radar (GPR) has become an invaluable and almost indispensable means of exploring shallow structures for geoscientific, engineering environmental and archaeological work. At the same time, GPR analysis software is mostly proprietary and usually available from GPR manufacturers or a handful of other vendors. There are few exceptions. A good but quite limited freeware package provided by the USGS (Lucius and Powers, 2002); this only works in non-Microsoft platforms and due to the particularities of its graphics drivers is not fully functional in Windows XP. A second freeware package is the **Radar Unix** by Grandjean and Durand (1999), which is limited to Unix and Linux platforms; it does not work in Windows, OS(2) or Mackintosh systems unless they're augmented with the CygWin Linux emulator, and then under severe restrictions. Furthermore, RU draws processing power from the Seismic Unix (SU) analysis system, but as it is based on an outdated version of SU, it requires extensive overhaul. Both freeware packages are written in C, while RU's graphical interface (Xforms) is written in C++. This renders both of them rather unwieldy to modification or augmentation by the average practitioner. Finally, a rather impressive piece of work, the openGPR project of Matthias Schuh, Tuebingen, Germany, appeared briefly on the scene; it was designed for Linux OS, was also dependent on SU and, unfortunately, it is lo longer available.

With the references above aside, the academic community has been slow to react on this issue. Limited collections of freely distributed software are usually focused on very particular problems (mainly data input / output), generally unorganized and so diversely programmed, that cannot form a consistent basis for the reliable manipulation of GPR data.

An effort to create a free GPR analysis and interpretation package that is cross-platform and expandable/ customizable to the needs of a particular user began in 2005 with the MATLAB-based matGPR Release 1 and continued with matGPR Release 2 (2010) and Release 3 (2013). This was an ambiguous project, albeit feasible because MATLAB provides an all-inclusive high level programming environment facilitating the development of advanced software. Release 3.5 of matGPR carries on but with changes in the distribution policy, as specified in Section 1.4 of this document.

At the present stage of development matGPR provides a rather broad and functional range of tools for the analysis of zero and single-offset GPR data. Although one can think several analysis tools to be included in future releases, it offers a decent and in several aspects advanced means of treating GPR data.

1.2. Why MATLAB and Which MATLAB?

In Section 1.1 above, I have made a brief statement about why MATLAB was chosen for the development of this GPR analysis software. However, there's more to that.

MATLAB was not available until the latter part of the 1980's, and prior to that, FORTRAN was the language of choice for serious numerical computation. C was making its first steps towards establishing a foothold in the world of scientific computation, but it didn't offer built-in facilities for doing complex arithmetic and this was a serious drawback. Conversely, FORTRAN lacked some of C's advantages such as structures, pointers, and dynamic memory allocation.

The appearance of MATLAB and its imitators made a big impact in the scientific community. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, familiar to FORTRAN programmers for being robust collections of tools for linear algebra. To do this, MATLAB also introduced a new vector-oriented programming language, an interactive environment, and built-in graphics. These features offered many advantages and boosted productivity in comparison to more traditional application development environments. Since then, MATLAB has evolved to embed a large collection of state-of-the-art numerical tools, high quality graphics, object-oriented extensions, strict runtime error checking, built-in interactive debugger, web services and a host of other facilities: it is a rapidly evolving beast, going through many changes and continuously acquiring new features.

Of course, C and FORTRAN have also evolved, with some editions also acquiring high-level (visual) application development environments. However, neither one offers as many advantages as MATLAB does, graphics, for instance, being a major issue. When it comes to effectiveness, what really matters is the efficiency of the entire process of realizing a new scientific idea, or creating an analysis and interpretation procedure. In this respect MATLAB is overall much more efficient and facilitates very rapid prototyping of new algorithms or applications.

matGPR tries to exploit all the facilities and utilities offered by MATLAB, which is a vast system, with many add-on application-specific *toolboxes* that are priced and licensed separately. Some of those, (e.g. Signal Analysis, Image Processing etc.), include algorithms and analysis techniques that are very useful to matGPR. It is conceivable that many people do not enjoy access to the full range of MATLAB products. Accordingly, special provision was made for matGPR to offer in-house solutions for those algorithms and analysis techniques that are necessary, but *not* supplied with the basic MATLAB system. Thus, matGPR can be used by all those in possession of a MATLAB license, even if they do not have access to any toolbox.

In a last comment, it is worth noting that M-code (MATLAB's language), on account of being interpreted and not compiled, is much slower than C and FORTRAN when it comes to computationally intensive tasks. This is quite true, as would be appreciated by anyone who, tired of waiting went for a cup of coffee and found that the computations were not even halfway through when he returned. MATLAB offers a solution in that it can integrate fast, compiled C or FORTRAN code with M-code by means of the MEX-file system. matGPR *did* make use of this facility but as of Release 3, it has phased it out. Another solution is to use M-code to drive customized, stand-alone C or FORTRAN programs and import their results for further processing. When truly heavy number crunching is needed, matGPR can make use of this approach by offering M-code alternatives that drive compiled software and instructions as to how to implement them. At any rate, many problems can be solved with more efficient programming: for many applications, vectorized and well-written M-code is as fast as compiled FORTRAN code, to the point of being competitive.

Up to Release 2.2.4.3, matGPR was programmed to be as backward compatible with earlier MATLAB editions, as possible. For instance, provisions were made for matGPR to detect the MATLAB version and, when necessary, not to invoke from an earlier version functions and procedures added in later versions, offering alternative, in-house solutions. As of Release 3, however, the code of will be *opaque* (*p-code*); transparent *m-code* will not be released but only under the specific conditions explained in Section 1.4 below. In consequence,

➔ matGPR will *not* work with any version of MATLAB earlier than 7.5 (Release 2007b).

It will be fully functional with MATLAB v7.5 and above, less the few bugs I may have overlooked, or unforeseen changes in the programming language (which may be quite more frequent than appreciated).

1.3. Downloading and Installation

matGPR is distributed as single compressed file (*.zip or *.rar). The archive can be downloaded from URL http://users.uoa.gr/~atzanis/matgpr/matgpr.html. Copy it to your work directory and uncompress. The matGPR home and subdirectory structure will be unfolded and the program will be ready to use. Start matGPR by executing the function **MATGPR_R3/matgpr.m** from the MATLAB command window. Use this manual and the on-line help to navigate through the program and the test data provided in the directory **MATGPR_R3/work/** to familiarize with it.

1.4. Licensing and Disclaimer

There are *two* versions of MATGPR R3: the ***Classic*** and the compiled, ***Stand-Alone*** version.

- The **Classic matGPR** requires a MATLAB installation and comprises two editions:
  - **matGPR Release 3.1** is an upgraded version of Release 3 (includes some coding improvements and enhanced data input capabilities). Until a decision is made as to the opposite, this version will continue to be freely distributed and licensed for education and research.
  - **matGPR Release 3.5** includes all the improvements of Release 3.1, as well as a number of new processing and analysis applications (curvature attributes robust spline smoothing, topographic compensation of depth-migrated data etc.). This edition, as well as future upgrades may be available *only* on request, (e.g. by e-mail). A *voluntary donation* toward the continued development of matGPR is *strongly encouraged*!
- In order to use the **Classic matGPR**, prospective users (Licensees) must agree to be bound by the terms and conditions specified in the *End User License Agreements* applicable to Release 3.1 and Release 3.5 respectively, which are attached to the corresponding distribution bundles. The terms of these Agreements can be summarized as follows:

1) Licensees are granted permission to install Release R3.1 in one or more computer systems.
2) Licensees are granted permission to install Release R3.5 in a *single* computer system identifiable to matGPR by its Media Access Control address (MAC address).
3) Licensees are granted permission to expand matGPR by *strictly adhering* to the instructions found Section 2.6 of this document, titled "*Expanding matGPR: How to write customized modules*".
4) Licenses are *revocable*, non-exclusive and non-transferable. With the sole exception specified in Clause 3 above, Licensees will not, and will not permit others to decrypt, reverse-engineer, disassemble, otherwise modify or reduce, re-package, license, sell, rent, lease, distribute, transmit, host, outsource or disclose matGPR under any circumstances.
5) The Classic matGPR is provided in the hope that it will be useful, but *without any warranty*; without even the implied warranty of merchantability or fitness for a particular purpose and without guarantees or warranties regarding its accuracy, safety, or any other quality or characteristic.
6) Under no circumstances shall the Author of matGPR (Licensor) be liable for any indirect, incidental, consequential, special or exemplary damages arising out of, or in connection to the Licensees' use of, or inability to use matGPR.
7) The Author of matGPR (Licensor) reserves the right to revoke a license and terminate any License Agreement in his sole discretion, at any time and for any or no reason.
8) The License Agreement will terminate immediately, without prior notice, in the event that Licensees fail to comply with any of its provisions. Licensees may also terminate the Agreement by deleting/uninstalling matGPR. Upon termination of the Agreement, Licensees shall cease all use, un-

install matGPR and delete all copies of matGPR in his possession.

9) The Classic matGPR is provided *as is* and *without* technical support or tutoring, except for Licensees who have made a voluntary monetary donation toward its development (see below).
10) Reports of bugs or errors *will* be examined and corrections, if any, will be communicated to the Licensee who reported the error.
11) In publications, reference to matGPR must be given, using one or all of the following:
    a. Tzanis, A. and Kafetsis, G., 2004. A freeware package for the analysis and interpretation of common-offset Ground Probing Radar data, based on general purpose computing engines, Bulletin Geol. Soc. Greece, vol. 36, No 3, 1347-1354.
    b. Tzanis, A., 2010. matGPR Release 2: A freeware MATLAB® package for the analysis & interpretation of common and single offset GPR data, FastTimes, 15 (1), 17 – 43.

- Accessibility to **Classic matGPR** Source Code:
  1) All editions of the Classic matGPR Release 3 and later are distributed in *opaque* (encrypted) *p-code* form, except for a limited number of third-party public domain functions and sub-programs that will continue to be distributed in *transparent* (readable) *m-code* form according to the specifications of their Authors and Copyright Holders.
  2) Transparent *m*-code might be released for research, after consultation with the Author. This would generally be done on a *reciprocal* basis (e.g. exchange of code, collaborative research etc.).
  3) Postgraduate students wishing to negotiate the release of *m*-code are required to provide, (a) evidence of their status, e.g. in the form of a letter by their supervisor, and, (b) a plan for the *required* reciprocal action. Requests by prospective users with e-mail accounts not belonging to accredited academic institutions will not be considered!

- The **stand-alone matGPR** *does not* require MATLAB and will *always* comprise the *latest* and *most updated* edition of the program (currently it is Release 3.5). At present it is available for use in a *single* computer system with Windows OS. The Stand-Alone matGPR would primarily be used by *non-academic* users. Accordingly, a *voluntary donation* toward the continued development of the software is ***expected*** and, ideally, would hover around a reasonable middle to high 3-digit Euro range. For details contact the Author by e-mail. Licensed Stand-Alone users will be eligible for technical support.

- Last but not least, I am sure that most of you appreciate the amount of time, effort and resources invested in producing a program like matGPR. If you use the classic matGPR (R3.1 and earlier) and find it useful, you might consider making a *voluntary donation* toward the continued development of matGPR. If you would like to use *matGPR R3.5* and later, a *voluntary donation* toward its continued development is *strongly encouraged*. This is particularly relevant in case you are going to use it for profit! I am sure that you will understand how in these times of economic turmoil, with research grants decimated, donations can be of great assistance in keeping research alive and ensuring the continued availability of matGPR.
  - To make a donation, please follow this link to the matGPR home page and then follow the link to the PayPal services provided therein.
  - Donations render users eligible for technical support, *commensurable in material and temporal extent* to the generosity of the donation. Generous donations, (e.g. in the middle 3-digit Euro range) are eligible for *long-term* technical support.
  - If you are not prepared to make a donation, you are kindly requested to refrain from asking for technical support.
  - Support *will definitely* be provided when you report bugs or errors. In this case the response will be prompt but will be *limited* to the specific bug or error!

CHAPTER 2. INTRODUCTION TO matGPR

matGPR is a two-layered software system, in which the bottom layer comprises a suite of functions to handle, display and process the data, while the top layer organizes these functions, automating data management and streamlining the flow of work by means of a GU Interface.

2.1. Program structure and the matGPR GUI.

The matGPR GUI (top layer) is produced by a long script (*matgpr.m*), which organizes the bottom layer functions and automates data management according to the flowchart of Figure 2.1. Figure 2.2 shows the physical appearance of the GU Interface; this is the start up matGPR window, which appears when the *matgpr.m* script is executed and *prior* to importing any data.

The design philosophy is quite simple: work flows in a continuous cycle between the ***Current Input Data (IPD)***, i.e. that data before some processing operation (step) and the ***Output Data (OPD)***, i.e. the data resulting from this operation. You import, display and inspect the Input Data with the appropriate choices under the **Data** and **View** menus. Then you decide and apply a processing step and inspect the result, i.e. the Output Data. If satisfied, you may keep (or 'hold') the result replacing the current Input Data with the Output Data and repeat the cycle with a new processing step. If not, you may ignore or discard the result and cycle with another processing step. A *multi-level* undo/ restore utility is available at any time at which control is returned to the matGPR GUI. You may save or export the current Input Data via the matGPR GUI. Soft and hard copies of the current Input and Output data (and any figure produced during a session), can be made at any time using the **FigureTools** menu of the figure windows (see below).

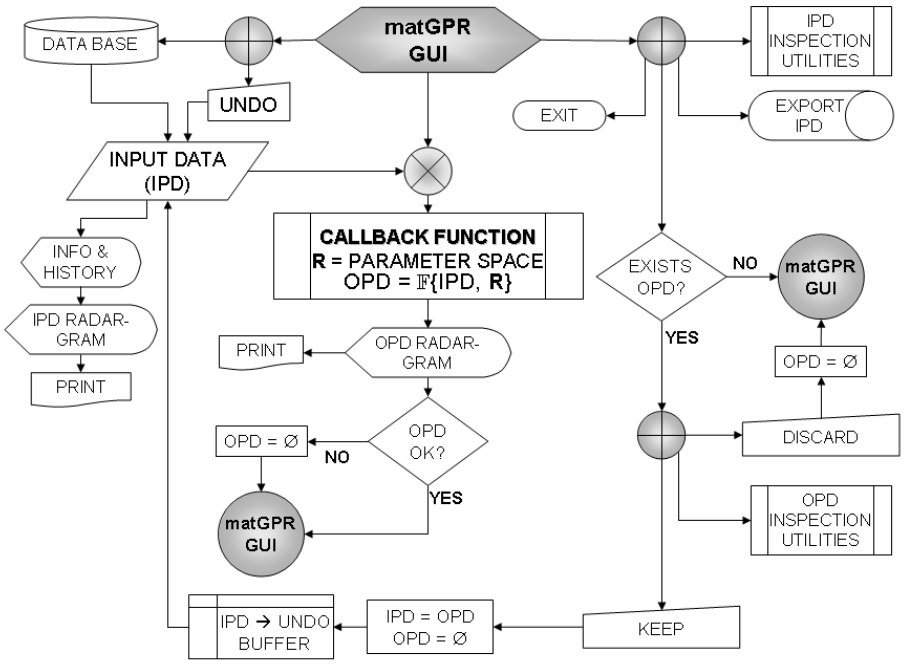


Figure 2.1. The flowchart of a data analysis session with matGPR

The current Input Data is held in a data structure named IPD and the Output Data in an identical structure named OPD. Henceforth, the Input Data will be referred to at the "IPD structure" or simply the "IPD". Likewise, the Output Data will be referred to as the "OPD structure" or simply the "OPD". The IPD and OPD structures rotate during a typical data processing cycle as follows:

- The IPD is copied onto the OPD.
- OPD is modified by application of some processing step (the "Callback Function" of Fig. 2.1).
- If the result is acceptable, the OPD is copied onto the IPD.
- If the result is not acceptable the OPD may be discarded, otherwise it will be automatically erased on application of the next processing step.

The IPD and OPD data structures comprise of the following fields (DATA stands for either the IPD or the OPD):

`DATA.origin`	Header, specifying the type of system that collected the original data set (manufacturer and monostatic of bistatic).
`DATA.pname`	String, path of the data file
`DATA.fname`	String, name of the data file
`DATA.d`	2-D array of size [`DATA.ns, DATA.ntr`], data matrix
`DATA.ns`	Scalar, Number of samples per trace (scan) in `DATA.d`
`DATA.dt`	Scalar, temporal sampling rate of the traces in `DATA.d`
`DATA.tt2w`	Vector of size [1 x `DATA.ns`], 2-way traveltime
`DATA.sigpos`	Scalar, signal position (time-zero determined during data acquisition)
`DATA.dz`	Scalar, spatial (depth) sampling rate of the columns (traces) of `DATA.d` – applies to depth migrated data
`DATA.z`	Vector, depth
`DATA.zlab`	String, label of the vertical axis, used for display
`DATA.ntr`	Scalar, Number of traces (samples per row) in `DATA.d`
`DATA.dx`	Scalar, trace spacing
`DATA.x`	Vector of size `[1 x DATA.ntr]`, horizontal coordinates of the rows (traces) of `DATA.d` (scan axis). Can be either number of traces for unequally spaced data, or distances from the start of the scan line.
`DATA.xlab`	String, label of the horizontal axis, used for display.
`DATA.markertr`	Vector, ID numbers and coordinates of marker traces. The coordinates are assumed fixed with respect to a local system of reference.
`DATA.xyz.Tx`	Array of size [`DATA.ntr x 3`], with the X, Y and Z coordinates of the *source* antenna in a local frame of reference. For monostatic GPR systems (zero-offset data) `DATA.xyz.Tx = DATA.xyz.Rx` (see below)
`DATA.xyz.Rx`	Array of size [`DATA.ntr x 3`], with the X, Y and Z coordinates of the *receiver* antenna in a local frame of reference. For monostatic GPR systems (zero-offset data) `DATA.xyz.Rx = DATA.xyz.Tx` (see above)
`DATA.TxRx`	Scalar, antenna offset (Tx - Rx separation)
`DATA.Antenna`	String, antenna name / designation
`DATA.DZThdgain`	Vector, variable gain settings used for data acquisition with GSSI systems (DZT)
`DATA.TimesSaved`	Scalar, the number of times that a "save data"operation has been done during the current session.
`DATA.comments`	On input, contains the file header, or header file comments of original data set. May be updated by the user.
`DATA.history`	Cell array of strings, containing the which processing history of the current data set
`DATA.matgprversion`	The current version of matGPR.

- **NOTE:** *The IPD and OPD structures are accessible through MATLAB's base workspace and can used or manipulated independently of the matGPR GUI. For example, their fields can be input / output to yours, or a third party's processing functions, while the results will still be available to matGPR for further manipulation*

When a data set is loaded, the matGPR GUI displays essential information about the *current input data* (IPD) in the **Data Info** area (Figure 2.3). This information is updated after each and every processing step. If a field has not yet been assigned a value, the Not Available (NA) message is projected.

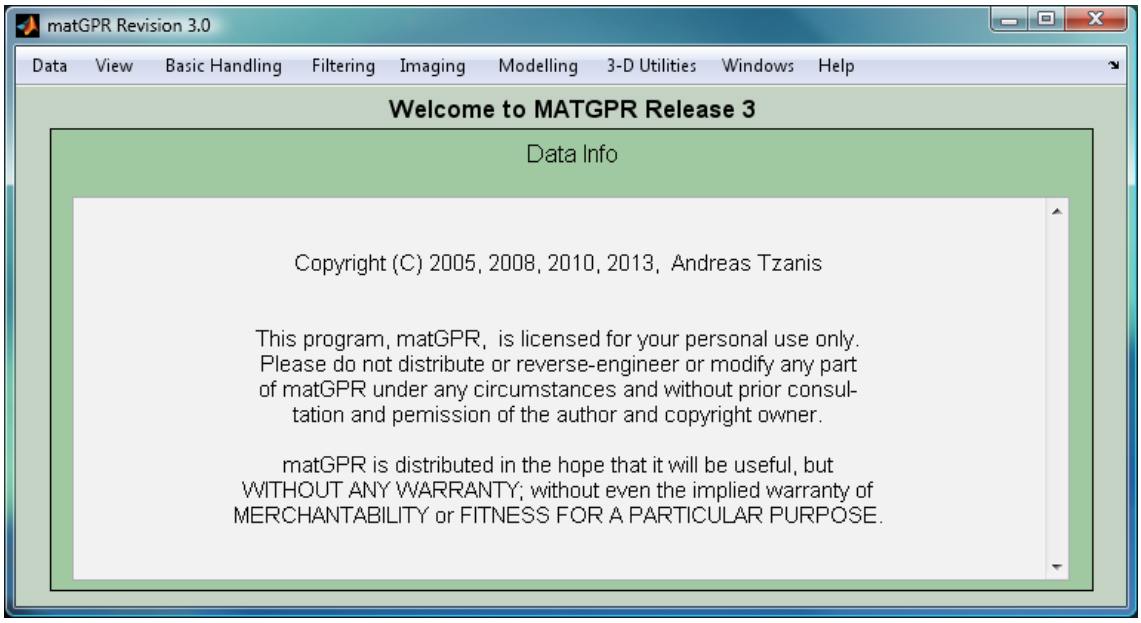


Figure 2.2. The start up window of the matGPR GUI.

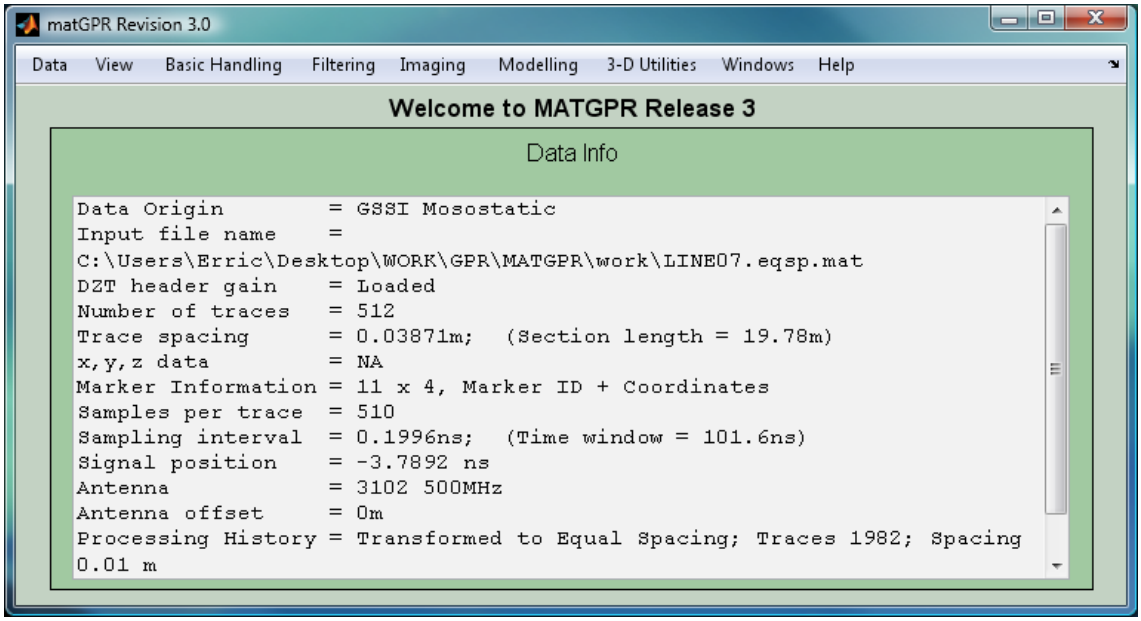


Figure 2.3. The matGPR GUI displays essential information about the IPD

2.2. Directory Structure

matGPR is a modular program. Each module performs a self-contained operation which is driven by the Callback Function associated with the corresponding menu choice (uimenu item). The modules are organized and linked into an effective set by *matgpr.m* and a small complement of other functions.

- The matGPR GUI is programmed in *matgpr.m,* which is the backbone of the top level and resides in the highest level directory **MATGPR_R3/**. When executed, *matgpr.m* adds the following directories to the MATLAB search path:
- The directory **toplevel/**, with the complement of data management and flow control, basic display and system utility functions, which together with *matgpr.m*, comprise the backbone of the top level.
    - The directory **toplevel/io/** contains data input/ output functions.
    - The subdirectory **toplevel/callbacks/** contains the executive branch of the top level, namely functions that are executed as Callbacks to the menu items (the matGPR modules). The names of the Callback functions have the form *do_xxxxxx.m* where *xxxxxx* is by default the name of an analysis function or a specific task. The Callback functions, in turn execute data handling and analysis functions residing in the **toplevel/io /** and **analysis/** directories.
- The directory **analysis/** contains data processing functions. Furthermore, **analysis/** is parent to the folders **fdtd/** and **alternative/**.
    - **analysis/fdtd/** contains Irving and Knight's (2006) 2-D, TM mode, finite difference time-domain modelling code, specifically adapted to the requirements of matGPR. For more information see item FDTD 2-D Modelling of the Modelling menu.
    - **analysis/alternative/** contains alternative implementations of some analysis functions that rely of external programs for faster execution. For example, phase-shift migration requires some serious number crunching and M-code can be slow in such cases. If you need the inherent speed to compiled FORTRAN code you might wish to consider using the *alternative* routines in this subfolder, together with the associated FORTRAN 90 programs. The existence of alternative implementations of analysis programs is pointed out in the detailed descriptions of the relevant analysis modules in Chapter 6 to 8.
- The directory **work/** is a workspace. In the matGPR distribution bundle it includes example data sets and model files. It is important to note that matGPR does not create any particular (project) workspace for a new data set. Rather, it uses the parent directory of the *raw* data as the *home directory* of the data analysis session. In the end it is really up to you to decide how to organize your project. In this respect, the directory **MATGPR/work/** may be a convenient temporary workspace. See also the items **Import Raw Data** and **Save Data to Mat-File** in Section 3.1, for additional information.
- The subdirectory **doc/** contains the documentation of matGPR; the *"Manual and Technical Reference"* (this book) in folder **doc/PDF/** and the on-line help in folder **doc/www/** (HTML). For all MATLAB releases featuring an in-house web browser, the on-line help is displayed automatically. Otherwise, you will have to bookmark the html help files (*matgpr_help_R3.html and Terms_of_Use_R3.html*) in your preferred web browser.

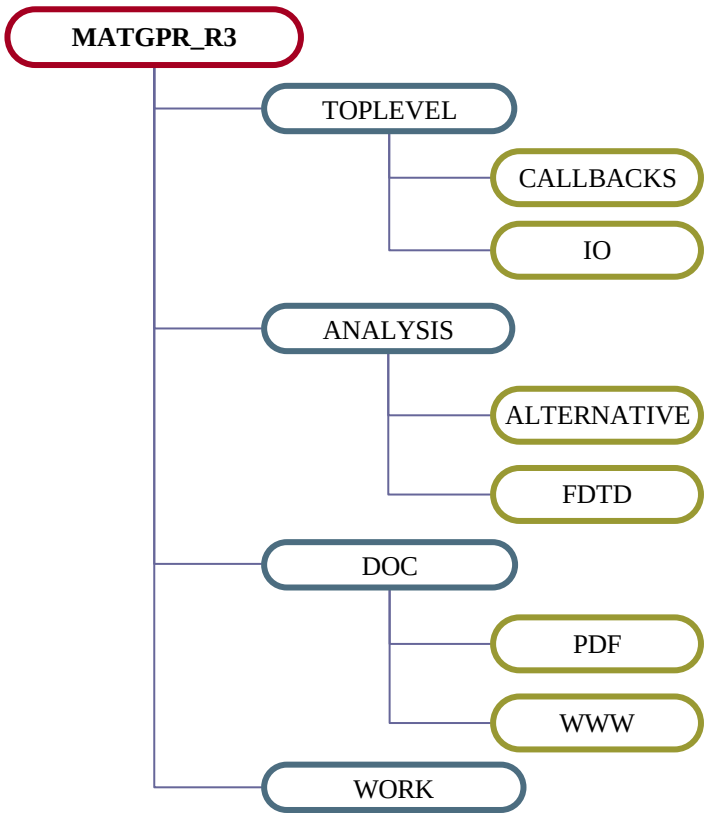


Figure 2.4. The directory structure of matGPR

2.3. Native matGPR file formats

Release 2 of matGPR introduced two new, specific binary files formats for I/O operations, in addition to MATLAB's standard .MAT file format. These are identified by the extensions *.MGP* and *.M3D* and supplemented the MAT-file format which was the hitherto standard. At the time of their introduction the new formats were *considerably* more compact (less than half of the space required by MAT-files). This is no longer true, but the new formats are here to stay…

The *MGP-format* is used for the temporary or permanent storage of processed data (i.e. the IPD structure). The MGP-file structure is quite simple and is given in the following table:

Record	**Type**	**Description**
Scalar integer	int16	Length of IPD.origin
IPD.origin	Uchar	Character, Data origin
Scalar integer	int16	Length of IPD.pname
IPD.pname	Uchar	Character, path of data file
Scalar integer	int16	Length of IPD.fname
IPD.fname	Uchar	Character, Data file name
IPD.ns	int16	Number of samples per trace

IPD.ntr	int16	Number of traces
IPD.d	Float	[IPD.ns x IPD.ntr] data matrix
IPD.dt	Float	Sampling rate. If the data has been depth-migrated, then IPD.dt = 0 and the traveltime vector will not be reconstructed.
IPD.sigpos	Float	Signal position
Real scalar	Float	First depth estimate if depth-migrated, else 0
IPD.dz	Float	Depth spacing. If the data has *not* been depth-migrated, then IPD.dz = 0 and the depth-axis will not be reconstructed.
Scalar integer	int16	Length of IPD.zlab
IPD.zlab	Uchar	Character, Z-axis label
Real scalar	Float	Location of first trace in scan axis
IPD.dx	Float	Trace spacing
Scalar integer	int16	Length of IPD.xlab
IPD.xlab	Uchar	Character, label of scan axis
Scalar integer	int16	Rows of IPD.markertr
Scalar integer	int16	Columns of IPD.markertr
IPD.markertr	Float	Marker traces and coordinates
Scalar integer	int16	Rows of IPD.xyz.Tx
Scalar integer	int16	Columns of IPD.xyz.Tx
IPD.xyz.Tx	Float	Tx coordinates in survey reference frame
Scalar integer	int16	Rows of IPD.xyz.Rx
Scalar integer	int16	Columns in IPD.xyz.Rx
IPD.xyz.Rx	Float	Rx coordinates in survey reference frame
IPD.TxRx	Float	Transmitter - Receiver separation
Scalar integer	int16	Length of IPD.Antenna
IPD.Antenna	Uchar	Character, Antenna name if in header files
Scalar integer	int16	Rows of IPD.DZThdgain
Scalar integer	int16	Columns of IPD.DZThdgain
IPD.DZThdgain	Float	Header gain for DZT data
IPD.TimesSaved	int16	Number of times saved
Scalar integer	int16	Rows of IPD.comments
Scalar integer	int16	Columns of IPD.comments
IPD.comments	Uchar	Header comments
Scalar integer	int16	Rows of IPD.history
for 1 : Rows of IPD.history,		
Scalar integer	int16	Length of i'th row in IPD.history
IPD.history{i}	Uchar	Character, i'th item in IPD.history -

- For additional information please refer to item Save to MGP or MAT file of the **Data** menu.
- The MAT-file format will continue to be supported and will be *simultaneously* available with the MGP format – it is useful and can be very handy, as all MATLAB users can attest to. You may override the default from the **Settings** item of the **Data** menu.

The M3D-file format is intended for storage of 3-D data volumes. The compact M3D-format is mandatory in matGPR, although it may be overridden by the user through the command window. An M3D-file is structured as follows:

Record	**Type**	**Description**
nx	int16	Size of X-axis (X-dimension)

x1	float	The minimum value of the (X-axis), i.e. the location of first trace(s) on the scan axis.
dx	float	Trace spacing in metres.
ny	int16	Size of Y-axis (Y-dimension)
Y	float	The Y-axis (vector).
nz	int16	Size of Z-axis (Z-dimension)
z1	float	Min(Z) - Minimum value of vertical axis. Can be Time (ns) or Depth (m)
dz	float	Z-spacing (can be Time or Depth)
d3d	float	The [*ny* × *nx* × *nz*] data volume is exported by writing a consecutively, *ny* slices of size [*nx* × *nz*].
lst	int16	Size of the Z-axis label
zlab	uchar	Z-axis label (time or depth).

The X- and Z- axis vectors are reconstructed from (x1, dx, *nx*), and (z1,dz, *nz*) respectively. Thence, the x-, y- and z- coordinates of the data voxels can be reconstructed using *meshgrid*. See also Load 3D Data and Save 3D Data for additional information.

2.4. The SEG-Y and Seismic Unix Standards.

matGPR may use the SEG-Y Revision 1 Data Exchange Format to import/ export data. This has achieved widespread usage within the geophysical community and has become a standard. Inasmuch as it can store all the information generated during GPR data processing (and much more), the SEG-Y standard can be adopted as the preferred method of exporting data for long-term storage or for exchange between different computer systems. The SEGY i/o utilities have been implemented using *SEG Y standard* (http://seg.org/publications/tech-stand/) as defined by the *Society of Exploration Geophysicists* (SEG). matGPR also supports the format used by CWP's *Seismic Unix* (http://www.cwp.mines.edu/cwpcodes/) package (*the SU format*), which is a simplified version the SEG-Y Rev.0 format (the first 3600 byte header is not used). A short description of the formats is given below.

2.4.1. Structure of a SEG-Y and a SU file

The SEG-Y file format is the standardized data storage for seismic magnetic reel tapes and has the following general structure:

<Reel Identification Header>

<Trace Data Block 1>

<Trace Data Block 2>

... ...

<Trace Data Block N>

The 3600 byte reel identification header comprises three parts:

- A 3200 (0xC80) byte Textual File Header, representing 40 × 80-byte "card images", which can be either EBCDIC formatted (Revision 0), or, either EBCDIC or ASCII formatted (Revision 1).
- A 400 (0x190) byte Binary File Header of fixed-point integers, of which only 60 (Revision 0) or 66 (Revision 1) are assigned. The remaining 340 / 334 bytes are unassigned for optional use (matGPR exploits this facility).
- In Revision 1 of the SEG-Y standard, an optional number of 'Extended Textual File Headers', each 3200 bytes long, also ASCII or EBCDIC formatted.

- This is followed by the data block consisting of trace ID headers and trace data as follows:
- Trace Header: 240 (0xF0) bytes of binary fixed-point and floating-point numbers, and,
- Trace Data: samples (number, length, and type defined in the reel ID header) that can be formatted in a number of ways, including IEEE floating point, IBM Floating Point and 1, 2 or 4 fixed point (integer) numbers.

A SU file is similar to SEG-Y with the following differences:

- • There is no 3600 byte reel header.
- • There are no optional Textual File Headers.
- • The trace data are IEEE floats.
- • Data can be both little and big endian formatted.

2.4.2. Implementation

matGPR R3 supports the following parts of the SEG - Revision 0 and 1 formats.

- **matGPR can import data written in both Revision 0 and Revision 1 styles.**
- **matGPR will export data in Revision 1 style only.**

2.4.2.1. Reel Identification Header

The 3200 byte textual file header can be ***only*** ASCII formatted. If the ***imported*** textual header is EBCDIC formatted, (as is usually the case with Revision 0 files), it will be read but will not be decoded. The ***exported*** textual header (created by matGPR) comprises three parts:

a The leading line is automatically assigned and has the form:
'C Created with matGPR on dd-mmm-yyyy hh:mm:ss. Format: SEG-Y Revision 1'.

b The next N × 80 bytes (next N lines) are user comments, inserted on file creation by means of a dialog box.

c The next (N+1):40 × 80 bytes (next N+1:40 lines,) are occupied by the original file header, or header file comments and the processing history of the exported data set (IPD.history). If the history is longer than (N+1):40 lines, then only the first (N+1):40 lines are exported.

2.4.2.2. Extended Textual Headers

In SEG-Y revision 1 a number of 3200-byte extended textual file headers are allowed. This feature ***is not supported*** in matGPR. Extended textual headers may be implemented in future releases.

2.4.2.3. Data Sample Format

The following data formats are supported:

SEG-Y revision 0 (1975)

Type	DataSampleFormat	Supported
1	4 Byte IBM Floating Point (as defined in IBM Form GA 22-6821)	**Yes**
2	4 Byte Fixed Point	**Yes**
3	2 Byte Fixed Point	**Yes**
4	4 Byte Fixed Point with Gain	No

SEG-Y revision 1 (2002)

Type	DataSampleFormat	Supported
1	4-Byte IBM Floating Point	**Yes**
2	4-Byte two's complement Integer	**Yes**
3	2-Byte two's complement Integer	**Yes**
4	4-Byte Fixed Point with Gain	No
5	4-Byte IEEE Floating Point	**Yes**
6	Not Specified	No
7	Not Specified	No
8	1-Byte two's complement Integer	**Yes**

The Type number is the number that should be used as the *Data Sample Format*, in functions *readsegy.m* and *writesegy.m* (field SEGYreelhdr.format = Type). The most used format has been Revision 0, Type 1 (IBM Floating Point). Now that Revision 1 has been published, the preferred way to read and write SEG-Y data may soon be Type 5 (IEEE Floating Point).

- In matGPR, **the default** data sample format is **Type 5 (IEEE Floating Point)**.

The SEG-Y reel ID header and the SEG-Y / SU trace headers live in the form of MATLAB data structures that can have fields with static or variable content. The static fields contain such invariant parameters as standard flags and constants, the sampling rate, number of traces etc. Such fields are the same in all trace headers. The variable fields contain parameters that change as a function of location, (e.g. coordinates), and are trace specific. The following two tables detail how matGPR assigns parameter values to the reel ID and trace header fields (and also where it expects to find and read these parameters).

2.4.2.4. SEG-Y Reel Header Value Assignments (structure SEGYreelhdr)

Header Field	Byte	Type	Description	Value
headertext	1	uchar	3200-byte text area ASCII-coded block	User text
Jobid	3201	int32	Job identification number	User
lino	3205	int32	Line number (only one scan line per reel)	User
Reno	3209	int32	Reel number	User
Ntrpr	3213	int16	Number of traces in line	IPD.ntr
nart	3215	int16	Number of auxiliary traces	0
Hdt	3217	uint16	Sample interval in microseconds	1000 × IPD.dt
Dto	3219	uint16	Same for original field recording	0
Hns	3221	uint16	Number of samples per trace	IPD.ns
Nso	3223	uint16	Number of samples per trace for original field recording	0
format	3225	int16	Data sample format code: 1 = IBM Floating Point (4 bytes) 2 = Fixed-point (4 bytes) 3 = Fixed-point (2 bytes) 5 = IEEE Floating Ppoint 8 = Fixed-point, 1-byte	User
fold	3227	int16	CDP fold expected per CDP ensemble	0

tsort	3229	int16	Trace sorting code: 1 = as recorded (no sorting) 2 = CDP ensemble 3 = single fold continuous profile 4 = horizontally stacked …	3
vscode	3231	int16	Vertical sum code: 1 = no sum 2 = two sum ... N = N sum (N = 32,767)	1
hsfs	3233	int16	Sweep frequency at start	0
hsfe	3235	int16	Sweep frequency at end	0
hslen	3237	int16	Sweep length (ms)	0
hstyp	3239	int16	Sweep type code: 1 = linear; 2 = parabolic; 3 = exponential; 4 = other	0
schn	3241	int16	trace number of sweep channel	0
hstas	3243	int16	Sweep trace taper length (msec) at start if tapered (the taper starts at zero time and is effective for this length)	0
hstae	3245	int16	Sweep trace taper length (msec) at end (the ending taper starts at sweep length minus the taper length at end)	0
htatyp	3247	int16	Sweep trace taper type code: 1 = linear; 2 = $\cos^2$; 3 = other	0
hcorr	3249	int16	Correlated data traces code: 1 = no , 2 = yes	0
bgrcv	3251	int16	Binary gain recovered code: 1 = yes, 2 = no	0
rcvm	3253	int16	Amplitude recovery method code: 1 = none, 2 = spherical divergence, 3 = AGC, 4 = other	1
mfeet	3255	int16	Measurement system code: 1 = meters; 2 = feet	1
polyt	3257	int16	Impulse signal polarity code: 1 = negative; 2 = positive	0
vpol	3259	int16	Vibratory polarity code:	0
dt	3261	float32	Sample spacing (sampling rate) in ns	IPD.dt
t1	3265	float32	First sample location (in ns)	IPD.tt2w(1)
dx	3269	float32	Sample spacing between traces in m	IPD.dx
x1	3273	float32	First trace location in m	IPD.x(1)
hunass1	3277	int16	Unassigned, 224 bytes	112 × 0
revision	3501	uint16	Revision number	1
fixedlenr	3503	int16	Fixed length trace flag	1
ntexthdrs	3505	int16	Number of extended textual headers	0
hunass2	3507	int16	Unassigned, 94 bytes	47 × 0

- The *yellow highlighted* fields above are *matGPR-specific* and occupy the first 16 bytes of the first 240-byte unassigned block. These fields contain the sampling rate, the first sample location (in time), the trace spacing and the first trace location. Some of this information has been assigned in other fields of the reel ID structure in the form of 2-byte integers (and will be reassigned in appropriate fields of the trace header structures). Here they are entered as IEEE floats in order to avoid rounding errors (from the back-conversion of the integers) and loss of precision. Moreover, their presence in certain locations of the reel ID and trace headers serves as a test, by which matGPR verifies whether an imported SEG-Y file was produced by itself, or by a third party. On importing a data file, matGPR

will cross-check the corresponding (floating point) dt and dx fields in the reel ID and trace headers and if it finds them identical (file is own), it will adopt the values read from the SEG-Y reel ID header.

- Attention should be paid to the *orange highlighted* field above. It is a mandatory SEG-Y Rev.1 field and contains the "fixed trace length" flag: fixedlen = 1 implies that all traces in the reel have the same length (number of samples). This is *mandatory* in matGPR, which *cannot handle variable length traces*. If matGPR finds any value other than one in bytes 3503 – 3504 of the reel header, it will abort!

2.4.2.5. SEG-Y Trace Header assignments (structure SEGYtracehdr)

Header Field	Byte	Type	Description	Value
tracl	1	int32	Trace sequence number within line	Trace number (i)
trac	5	int32	Trace sequence number within reel	Trace number (i)
fldr	9	int32	Field record number	1
tracf	13	int32	Trace number within field record	Trace number (i)
ep	17	int32	Energy source point number	0
cdp	21	int32	CDP ensemble number	0
cdpt	25	int32	Trace number within CDP ensemble	0
trid	29	int16	Trace identification code: 1= seismic data; 2= dead; 3= dummy; 4= time break; 5= uphole; 6= sweep; 7= timing; 8= water break; 9 - N= optional use N= 32,767) 200 = GPR data (matGPR flag)	7182 (ASCII [71 82] = 'GR' for GeoRadar)
nvs	31	int16	Number of vertically summed traces (see vscode in reel header structure)	1
nhs	33	int16	Number of horizontally summed traces (see vscode in reel header structure)	1
duse	35	int16	Data use: 1 = production; 2 = test	1
offset	37	int32	Distance from source point to receiver group(negative if opposite to direction in which the line was shot)	$10^3 \times$ IPD.TxRx
gelev	41	int32	Receiver group elevation from sea level (above sea level is positive)	$10^3 \times$ IPD.xyz.Rx(i,3) or 0
selev	45	int32	Source elevation from sea level above sea level is positive)	$10^3 \times$ IPD.xyz.Tx(i,3) or 0
sdepth	49	int32	Source depth below surface (positive)	0
gdel	53	int32	Datum elevation at receiver group	0
sdel	57	int32	Datum elevation at source	0
swdep	61	int32	Water depth at source	0
gwdep	65	int32	Water depth at receiver group	0
scalel	69	int16	Scale factor for previous 7 entries with value +/- 10 to the power power 0, 1, 2, 3, or 4 (if positie multiply, if negative divide)	-3
scalco	71	int16	Scale factor for next 4 entries with value +/- 10 to the power 0, 1, 2, 3, or 4 (if positive, multiply, if negative divide)	-3
sx	73	int32	X source coordinate	$10^3 \times$ IPD.xyz.Tx(i,1) or 0
sy	77	int32	Y source coordinate	$10^3 \times$ IPD.xyz.Tx(i,2) or 0

gx	81	int32	X group coordinate	$10^3 \times$ IPD.xyz.Rx(i,1) or 0
gy	85	int32	Y group coordinate	$10^3 \times$ IPD.xyz.Rx(i,2) or 0
counit	89	int16	Coordinate units code for previous four entries 1= length (meters or feet) 2= seconds of arc X -longitude Y -latitude, positive to the east of Greenwich or north of the equator.	1
wevel	91	int16	Weathering velocity	0
swevel	93	int16	Subweathering velocity	0
sut	95	int16	Uphole time at source	0
gut	97	int16	Uphole time at receiver group	0
sstat	99	int16	Source static correction	0
gstat	101	int16	Group static correction	0
tstat	103	int16	Total static applied	0
laga	105	int16	Lag time A, time in ms between end of 240-byte trace identification header and time break, positive if time break occurs after end of header, time break is defined as the initiation pulse which maybe recorded n an auxiliary trace or as otherwise specified by the recording system.	0
lagb	107	int16	Lag time B, time in ms between the time break and the initiation time of the energy source, may be positive or negative.	0
delrt	109	int16	Delay recording time, time in ms between initiation time of energy source and time when recording of data samples begins (if recording does not start at zero time).	$10^3 \times$ IPD.sigpos
muts	111	int16	Mute time--start	0
mute	113	int16	Mute time--end	0
ns	115	uint16	Number of samples in this trace	IPD.ns
dt	117	uint16	Sampling interval; in micro-secs	$10^3 \times$ IPD.dt
gain	119	int16	Gain type of field instruments code: 1 = fixed; 2 = binary, 3 = floating point; 4 - N=optional use.	0
igc	121	int16	Instrument gain constant	0
igi	123	int16	Instrument early or initial gain	0
corrsu	125	int16	Correlated: 1 = no; 2 = yes	0
sfs	127	int16	Sweep frequency at start	0
sfe	129	int16	Sweep frequency at end	0
slen	131	int16	Sweep length in ms	0
styp	133	int16	Sweep type code: 1 = linear; 2 = parabolic; 3 = exponential; 4 = other	0
stas	135	int16	Sweep trace taper length at start in ms	0
stae	137	int16	Sweep trace taper length at end in ms	0
tatyp	139	int16	Taper type: 1 =linear, 2 =$\cos^2$, 3 =other	0
afilf	141	int16	Alias filter frequency if used	0
afils	143	int16	Alias filter slope	0

nofilf	145	int16	Notch filter frequency if used	0
nofils	147	int16	Notch filter slope	0
lcf	149	int16	Low cut frequency if used	0
hcf	151	int16	High cut frequency if used	0
lcs	153	int16	Low cut slope	0
hcs	155	int16	High cut slope	0
year	157	int16	Year data recorded	Year of file creation
day	159	int16	Day of year	Day of file creation
hour	161	int16	Hour of day (24 hour clock)	Hour of file creation
minute	163	int16	Minute of hour	Minute of file creation
sec	165	int16	Second of minute	Second of file creation
timbas	167	int16	Time basis code: 1 = local; 2 = GMT; 3 = other.	1
trwf	169	int16	Trace weighting factor, defined as 1/2^N volts for the least significant bit.	0
grnors	171	int16	Geophone group number of roll switch position one	0
grnofr	173	int16	Geophone group number of trace one within original field record	0
grnlof	175	int16	Geophone group number of last trace within original field record	0
gaps	177	int16	Gap size (total number of groups dropped)	0
otrav	179	int16	Overtravel taper code: 1 = down (or behind); 2 = up (or ahead)	0
cdpx	181	int32	X coordinate of CDP position of this trace	0
cdpy	185	int32	Y coordinate of CDP position of this trace	0
inline3d	189	int32	In-line number for 3-D poststack data	0
crossline3d	193	int32	Cross-line number for 3-D poststack data	0
shotpt	197	int32	Shot Point	0
shotptscalar	201	int16	Shot Point Scalar	0
tvmu	203	int16	Trace Value Measurement Unit	0
tcm	205	int32	Transduction Constant Mantissa	0
tcp	209	int16	Transduction Constant Power	0
tdu	211	int16	Transduction Units	0
traceid	213	int16	Trace Identifier	0
sthdr	215	int16	Scalar Trace Header	0
srctype	217	int16	Source Type	0
sedm	219	int32	Source Energy Direction Mantissa	0
sede	223	int16	Source Energy Direction Exponent	0
smm	225	int32	Source Measurement Mantissa	
sme	229	int16	Source Measurement Exponent	
smu	231	int16	Source Measurement Unit	0
dx	233	float32	Sample spacing between traces	IPD.dx or 0
ntr	237	int16	Number or traces in scan line	IPD.ntr
Mark	239	int16	Mark selected traces	1 = Marker Trace 0 = Normal Trace

- The *yellow highlighted* fields above are *matGPR-specific* and occupy the last 8 bytes of the unassigned block.

2.4.2.6. SU Trace Header assignments (structure SUHDR)

Header Field	Byte	Type	Description	Value
• The SU and SEG-Y *Revision 1* trace headers are *identical* from *byte 1 to byte 180* • The SU trace header is designed on the basis of the SEG-Y *Revision 0* trace header. • **The SU standard trace header uses the *unassigned* last 140 bytes of the SEG-Y Revision 0 header as follows:**				
d1	181	float32	Sample spacing for non-seismic data	IPD.dt*1000;
f1	185	float32	First sample location for non-seismic data	IPD.tt2w(1)
d2	189	float32	Sample spacing between traces	IPD.dx or 0
f2	193	float32	First trace location	IPD.x(1) or 1
ungpow	197	float32	Negative of power used for dynamic range compression	0
unscale	201	float32	Reciprocal of scaling factor to normalize range	0
ntr	205	int32	Number of traces	IPD.ntr
mark	209	int16	Mark selected traces	1 = Marker Trace 0 = Normal Trace
shortpad	211	int16	Alignment padding	0
unass	213	int16	Unassigned	14 × 0

2.5. The FigureTools and ImageColours features

matGPR provides a set of utilities for managing figures and colour schemes (colormaps). A brief description is given below:

FigureTools: The MATLAB "Figure Menu" contains a large collection of utilities, some of which may be useful in rare circumstances, and some of which may actually not apply in the context of matGPR (for instance curve-fitting while displaying images, lighting a 2-D image or saving the workspace from a GUI dialog). The **FigureTools** menu provides a concise collection of the tools and utilities that everybody would like to have handy, while doing away with the full (and sometimes cumbersome) complement of the MATLAB Figure Menu. However, if one must access some feature of the "Figure Menu" (for instance advanced annotation utilities), it is always possible to switch it on and off. **FigureTools** provides for:

1. Colour scheme editing, specific to the current figure as opposed to the global colour scheme manipulation utilities provided by the **ImageColors** menu.
2. Zooming, panning and data inspection.
3. Exporting the current figure as a MATLAB *.fig file or as an image in various graphics formats supported my MATLAB.
4. Setting up the printer and printing the current figure.
5. Editing the current figure, the current axes and their children graphics objects.
6. Toggling the MATLAB Figure Menu on and off.

ImageColours: Menu to change and manipulate the colors of data displays (see Show Data/ Show Processed Data items). The default colormap is **gray** (gray-scale). The menu choices allow switching the colormap to:

MATLAB colormaps:

- **gray** : Linear greyscale.
- **bone** : Greyscale with a higher value for the blue component.
- **jet** : Rainbow, ranges from blue to red and passes through cyan, yellow and orange. It is a variation of the hsv colormap.
- **hot** : Reel varying smoothly from black through shades of red, orange, and yellow, to white.

matGPR additional colormaps

- **Blue – Red** : Reel varying smoothly from navy blue to white and from white to brick red. Useful for emphasizing the higher amplitude reflections while masking low amplitude noise.
- **Brown – Black** : Reel varying smoothly brown to white and from white to black.
- **Red – Black** : Reel varying smoothly from brick red to white and from white to black.

Colormap manipulation features include:

- **Colour Saturation** : Increases/decreases *contrast* (colour saturation) via a slider GUI. This option is available *only* in 'image' display mode. The change applies only to the current figure.
- **brighten** : Increases the brightness.
- **darken** : Decreases the brightness.
- **flipud** : Reverses colour order.
- **permute** : Interchanges colour indices

These changes apply to the current global colormap and persist until they are reset or the colormap changes, with the exception of "Colour Saturation" which only applies to the current figure.

2.6. Expanding matGPR: How to write customized modules

matGPR has modular architecture and although you cannot generally see the code, **you can always expand matGPR with your own analysis modules easily a**nd without having to know details about the program's infrastructure. This section will provide a simple example on how to expand matGPR. Some basic familiarity with the MATLAB language and the use of uimenu's in particular, is necessary.

There are three steps towards integrating your work into matGPR: **1)** Prepare your bottom level I/O or analysis function, **2)** set up a menu (uimenu) to access the function, and, **3)** set up the callback routine to execute the function and handle results and exceptions. Assuming that your I/O or analysis function has already been prepared, the following sections will deal only with Steps 2 and 3 above.

2.6.1 Setting up the menu.

The recommended way to expand matGPR is by creating your very own add-on menus in a separate M-file. To do this, you should first create an M-file for it. Let us call this file *mymenu.m*. The first line to this file is, of course, a function declaration:

```
function mymenu();
```

The first thing that *mymenu.m* should do, is to focus on the matGPR GUI. Because the tag of the matGPR GUI is **`fig0`**, you can use the commands:

```
%%% Focus on matGPR home window
    if ~isempty(findobj('Tag','fi0')),     % Ensure that matGPR is running

    else
        erh = errordlg('matGPR is not running!', 'matGPR : ERROR');
        uiwait(erh);
        return;
    end
```

This code includes checking of whether the matGPR GUI is active, so as to preclude the annoying red error messages. The next step is simply to define your custom superset menu, by giving it a label and a tag. Let us label it "MY MENU":

```
%%% ========================================================================
%%%  THIS IS MY VERY OWN MENU
%%% ========================================================================
uimenu('Label',' MY MENU ', 'Tag', 'mynewmenu');
```

Now, define the menu choices as children of "MY MENU" and assign them their label and their **callback** property. For example, suppose that you want to add a spiking deconvolution utility:

```
uimenu(findobj('Tag','mynewmenu'),'Label','Spiking Deconvolution',...
    'Callback', 'OPD = do_spikedc(IPD); ' );
```

In a final step, save *mymenu.m* to the **MATGPR_R3/toplevel/** directory. *Mymenu.m* should be executed right after invoking the matGPR GUI (i.e. after running *matgpr.m*). Note that you can automate this procedure by preparing a dummy script with only two lines:

```
matgpr
mymenu
```

When done, the augmented matGPR GUI should look as in Figure 2.5 and ready to recall the analysis function.

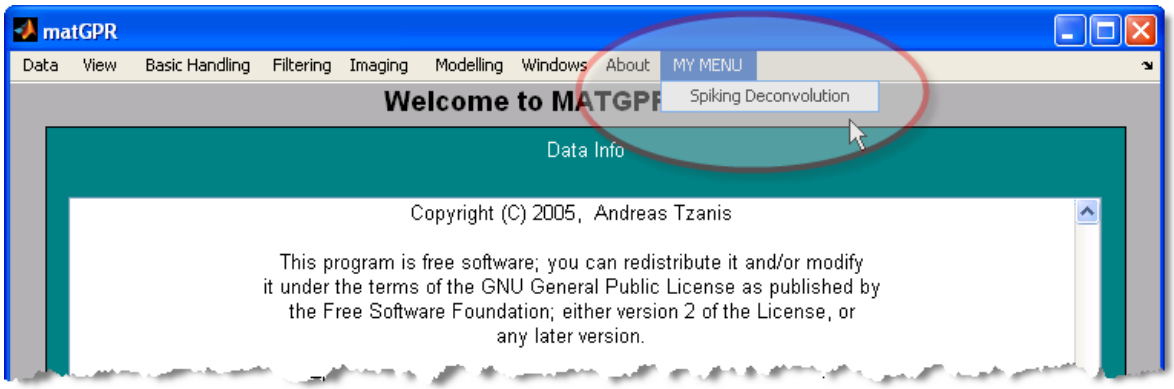


Figure 2.5. Augmenting matGPR with your own menus

2.6.2. Setting up a Callback function

Augmenting the matGPR GUI is only a part of the job. We now need to specify the Callback function, which in turn will execute the analysis function.

- Note that for simple tasks, a Callback function is not necessary.

There's no rule or restriction in naming the Callback function and you call it whatever you like. However, and in order to associate Callbacks with I/O and analysis functions, the official matGPR release(s) by convention uses a name of the form *do_xxxxxx.m*, where the prefix *do_* indicates a driver (callback)

function and *'xxxxxx'* is the name of the driven *analysis* function. For example, if the driven function name is *gainagc.m*, the driver (Callback) function is named *do_gainagc.m*.

Callback functions would normally reside in the **toplevel/callbacks/** directory. This is also not obligatory, so long as the function's parent directory is in the MATLAB search path. It is however neat and helps with bookkeeping. For these reasons alone the convention is faithfully observed in the official MATLAB release(s). A general purpose Callback function would be:

```
function OPD = do_MY_FUNCTION(IPD);
%
% Interface to drive "MY_FUNCTION.m" … and do something …
%
OPD = discardprocdata;                              % discard the current OPD
%
% Trap common errors - for example
if isempty(IPD.d),
   erh = errordlg('No data to process!', 'matGPR: ERROR');
   uiwait(erh);
   return
end
% Proceed ...
OPD = IPD;                                                  % Copy IPD to OPD
% Apply Standard AGC
[OPD.d, …, Exit_Condition] = MY_FUNCTION(IPD.d, MY_ARGUMENTS);
% Check if analysis was aborted for some reason
if ~Exit_Condition,
    disp('MY FUNCTION  > Operation aborted - No data returned!');
    return
end
% Display the results
viewdata(OPD.x,OPD.tt2w,OPD.d,'outdata',OPD.xlab,OPD.zlab);
% Update processing history
iss = size(OPD.history,1);
text = 'MY_FUNCTION DID THIS AND THAT …';
OPD.history(iss+1,1) = cellstr(text);
return
```

The size and complexity of the Callback function is unrestricted. However, it should rather have some standard features. Some of them are obligatory (☞), while others are merely recommended as being helpful (•):

- Discard of the OPD structure. This is not absolutely necessary, but it tidies up memory for the next processing step, and because it re-initializes the OPD structure, on occasion, it might prove helpful if something goes wrong during the execution of the analysis (driven) function.
- Obvious or Common Error trapping. In our example, it is not inconceivable that one will accidentally try to execute the driven function before loading any data, or after accidentally destroying the IPD. Common error trapping is also not absolutely necessary, but helps in avoiding unpleasant crashes and error messages.

☞ Copy the IPD onto the OPD.

☞ Processing (this modifies the OPD).

☞ Display the results.

- Update the processing history. This is also not necessary, but it is very useful for bookkeeping.

In order to help with creating Callback functions for new modules, matGPR furnishes a (sort of) template, which is found in the **Callbacks/** directory and is called *do_SOMETHING.m*, where *SOMETHING* stands for the new analysis function that will be driven by the new Callback function.

CHAPTER 3. I/O and Flow Control: The 'Data' Menu

Import Raw Data. In matGPR, the term *"Raw Data"* defines data sets that exist in the GPR manufacturer's native storage format and *prior* to their importation /conversion to the matGPR data structure format.

- matGPR can import *raw* data in the native formats of the GPR manufacturers: 1) **GSSI** (*.DZT files) including 8/16-bit data collected with models up to SIR-3000 and 16/32-bit data collected with model SIR-4000. 2) **Måla Geophysics** (RAMAC) including 16-bit data stored in *.RD3 files and 32-bit data stored in *.RD7 files. 3) **Sensors and Software** (PULSE EKKO / DT1) and, 4) **ZOND** (SEG-Y). *Only single-channel* data files are acceptable, inasmuch as matGPR is currently being offered for zero- or single-offset surveys.
- With reference to RAMAC RD3 or RD7 data, it is possible to import and assimilate GPS trace positioning, if available, by decoding the associated **.cor** file where GPS information is stored. The positions of the GPS-located traces are projected in the UTM coordinate system and are treated as *marker traces*: they are stored in the **IPD.markertr** field whence they can be manipulated in various ways.
- With reference to bistatic Pulse Ekko (DT1) radars, note that only data recorded in *reflection mode* (single-offset) are fully supported. Multi-offset data, (e.g. CMP gathers) can be imported and displayed, but at this stage of development, matGPR does not offer any processing utilities beyond rudimentary editing and filtering.
- Data stored in **Seismic Unix (SU)** and **SEG-Y Revision 0** and **Revision 1** formats can also be imported. Please refer to Section 2.4 for details on the implementation of the SEG-Y and SU standards.

Once read, the data may be automatically saved in the matGPR-specific MGP-file binary format (default) or in MATLAB's native MAT-file binary format for faster and easier subsequent access. The default is ***not*** to save and can be changed through the Settings menu. When saving after reading, the destination directory of the MGP- or MAT-file is the parent directory of the raw data file and the saved file name is simply the raw data file name with the extension '*.mgp*' or '*.mat*' appended. For example, if the raw data file *testdata.dzt* is imported, the IPD will be saved in the MGP-file *testdata.dzt.mgp*. Likewise, *trench.su* will beget *trench.su.mat*.

After a file is read, *all* subsequent input and output operations will be directed to the parent directory of the raw data file, which thus becomes the *home directory* of the data analysis session. The home directory can change only by importing data from a different directory. For additional information refer to the item Save to MGP or MAT file.

View Header Information. Prior to importing any raw data and at any time during a matGPR session, you can inspect the complete information held in the file header of any one of the supported data formats. Figure 3.1 shows the "Header Information" window generated by matGPR for a single channel RADAN / DZT data file.

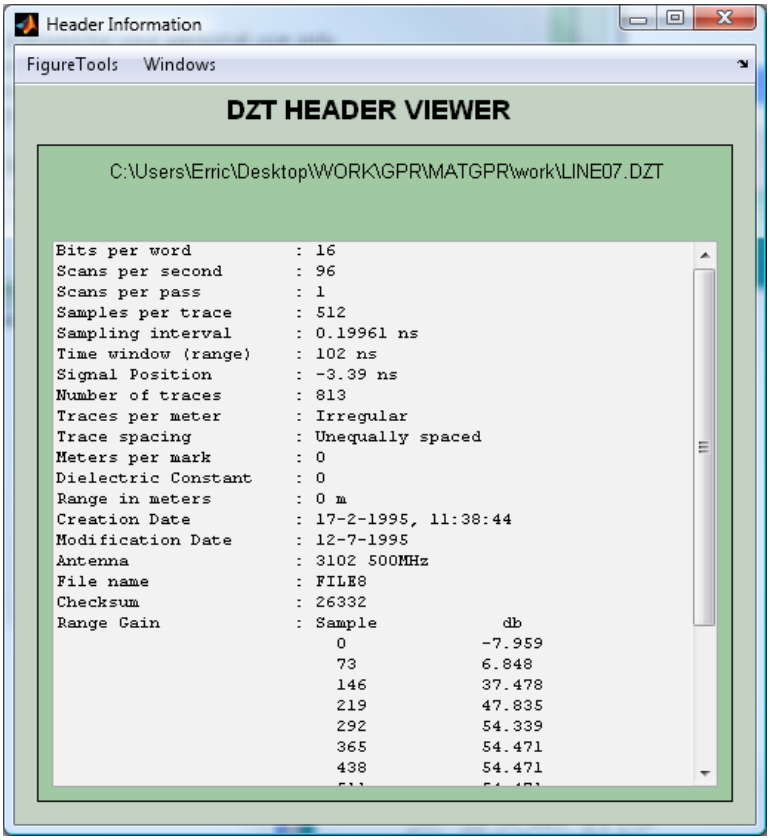


Figure 3.1. The **Header Information Viewer** window for RADAN / DZT data.

Import from MGP or MAT file. matGPR saves the Input Data structure (IPD) either in the native MGP-file format (default), or in MATLAB binary format (native MAT-file), for fast and easy access during subsequent analysis sessions. A save operation is done automatically, right after importation of a raw data file. In addition, the IPD can be saved to an MGP- or MAT-file at any time during a processing session. All these data sets are imported using the **Data → Import from MGP or MAT file** choice. For additional information see the items Import Raw Data and Save to MGP or MAT file.

Concatenate Sections. Consecutive or broken GPR sections can be joined into a single data set using this menu option. All data sets must be in one of the matGPR native formats, i.e. MGP, MAT or both, and have the same number of samples per trace (IPD.ns), the same sampling rate (IPD.dt) and the same trace spacing (IPD.dx). The first two conditions also apply to unequally spaced data, i.e. those taken with instruments without survey wheels, or prior to their transformation to equally-spaced data (IPD.dx is empty). Marker trace information is preserved during concatenation. When selected, matGPR generates the window of Figure 3.2, to assemble files for concatenation. This can be done either by typing full path names, or by using the file browse (**Browse** button). When using the browser, *multiple files* can be selected using the **Cntrl** and **Shift** keys. Files can be moved out of the list by selecting and clicking of the **Remove from list** button. The **GO** button starts the concatenation procedure and returns a new, *contiguous* input data set.

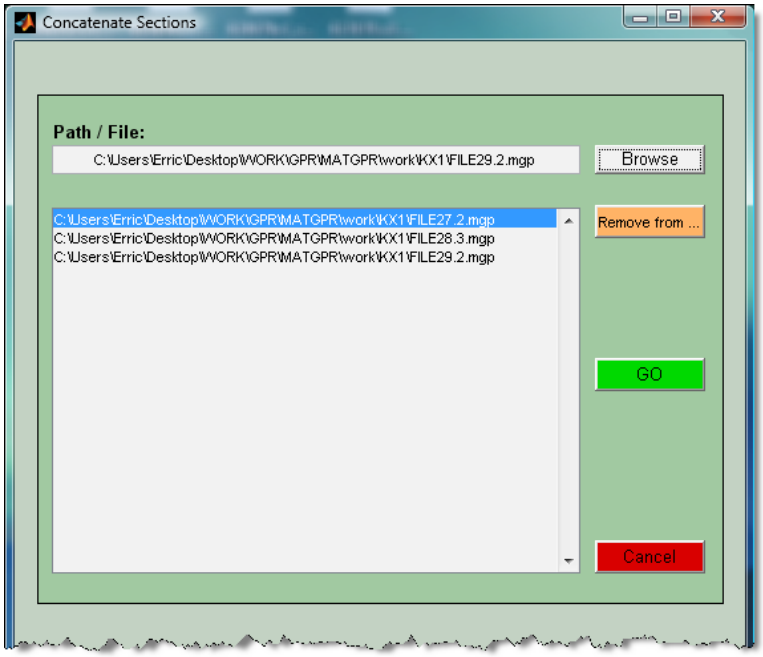


Figure 3.2. The "Concatenate Files" GUI

Hold Processed Data / Discard Processed Data. As stated above, the IPD and OPD structures rotate during a processing session as follows:

1) The IPD structure is copied onto the OPD structure.
2) OPD is modified by application of some processing step.
3) If the result is acceptable, OPD is copied onto IPD. This is done by selecting **Data → Hold Processed Data**.
4) If the result is not acceptable you may discard the OPD using the **Data → Discard Processed Data** choice, otherwise it will be automatically erased, upon application of another processing step.

Undo / Restore. It is possible to undo a number of processing steps in order to restore the data to a previous state. Selecting this menu item invokes a GUI as in Figure 3.1.3, which displays the processing history of the current data set *during the current session*. You may specify any stage during the displayed processing history, to which the data will be restored. In the example of Figure 3.3, the data will be restored to the state it was just after removing the global background. The number of *undo levels*, i.e. the number of processing steps you may undo, can be changed during runtime – see Settings for details. Theoretically, there's no limit to the number of undo levels, but because the previous processing steps are stored in core memory, your computer system will set a practical limit, which you have to determine on by yourself. The number 5 or 6 appears to be a good compromise. Note that processing steps applied and saved prior to the current analysis session cannot be undone!

Clear Undo Buffer. Remove previous processing steps stored in the Undo Buffer to reclaim memory. Once the buffer is emptied, the counter is reset and the buffer begins to fill from naught.

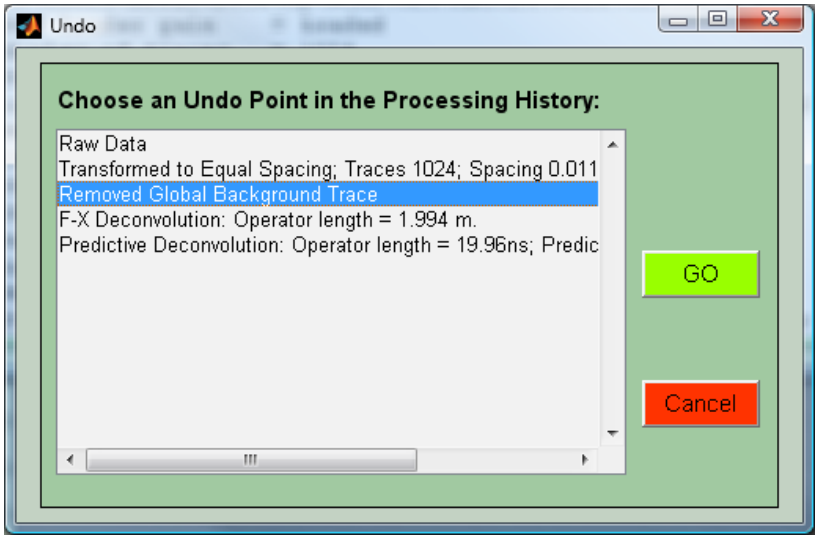


Figure 3.3. The "Undo" GUI. When the **Go** button is pressed, the IPD will be restored to the state specified by the highlighted step in the data processing history.

Save to MGP or MAT file. At any time, you may save the IPD structure in the matGPR-specific MGP-file format (default), or in MATLAB's native MAT-file binary format, by means of the **Data → Save to MGP or MAT file** request.

By default, the destination (home) directory of the MGP- and MAT-files is the home directory of the imported data file. The home directory can change only by importing data from a different directory.

The file names generated during a save operation comprise three parts, e.g. *testdata.2.mgp*. The 1st part is the principal file name; by default it is the same as the file name of the raw data begetting the Current IPD. In the above example, *testdata* derives from the raw file name *testdata.dzt*. The 2nd part is an incremental *save identifier*, indicating the sequential number of the save operation. In the above example, a save identifier of 2 indicates that this was the second time the Current IPD was saved after the data was originally imported from *testdata.dzt*. This helps to keep track of the processing history of a given data set. Third is the default extensions *".mgp"* or *".mat"*. You can modify any, or both of the first and second parts without consequences. The saved data sets can be re-imported to matGPR for further manipulation using the **Data → Import from MGP or MAT file** option.

- **The MGP-file format** is the matGPR default. This was introduced with matGPR Release 2 because at the time, it was *considerably* more compact than the MAT-file format. This is no longer true but the format will continue to be the default for as long as there is matGPR.
- The MAT-file format will also continue to be supported and will be *simultaneously* available with the MGP format. You can change the default binary file format from MGP to MAT through the submenu Data → Settings → Output Binary File → MAT file**.**
  - **Note on the MAT-file format:** On start up, matGPR specifies that the MAT-file format will be the default of the MATLAB version under which it is running. However, while the MAT-files saved by MATLAB versions 5.x and 6.x are readable by versions 7.x, the converse is *not* true because it compresses the data and uses Unicode character encoding. If for some odd reason you still want your MAT-files to be readable by earlier versions, you must make the **Data → Settings → MAT-file Save Format → V.6** choice *before* any saving operation. You can switch back by selecting the **Data → Settings → MAT-file Save Format → V.7 (Default)** option. Saving to MATLAB Version 4 format *altogether* impossible!

Save Depth Migrated Data. Depth migration marks the end-point of a data processing line. At present, there's practically nothing more that can be done with matGPR that would be theoretically correct and would extract meaningful information. Accordingly, depth migrated data can only be saved to a binary file, for future use (e.g. concatenation or generation of 3-D volumes), in the form of an IPD structure. Note, however, that on saving depth migrated data, time parameters, namely the sampling rate (IPD.dt) and the 2-way traveltime vector (IPD.tt2w) are *not* preserved. This is important to have in mind when accessing depth-migrated data sets. Also note that the "Processed GPR Data" figure can be saved as a *'.fig'* file, using the FigureTools utilities. The '.fig' format fully preserves the information displayed in the figure, as well as the data!

Export. For distribution or exchange, the data can be exported to **SEG-Y Revision 1, SU** and **DZT (RADAN)** file formats. The exportation procedure is straightforward. The first two formats have achieved widespread usage within the geophysical community and have become standards readable by practically all existing GPR analysis and interpretation software. For this important reason, these formats, SEG-Y in particular, have been adopted as the preferred method of exporting data for exchange between different computer systems. For details about the implementation of the SEG-Y and SU standards refer to Section 2.4 and for changing between the different supported SEG-Y binary codings to Group 6 of the Settings menu. It is for the same reasons that earlier versions of matGPR did not export data to proprietary file formats (RADAN, RAMAC, etc.). matGPR will gradually incorporate such utilities

- **Important Note:** The DZT exportation utility does not include the SIR-3000 and SIR-4000 extensions of the header block. It also *does not* calculate the header checksum. In consequence, you should *expect* errors or warnings when trying to import DZT data created with matGPR into another program. For instance, the USGS package (Lucius and Powers, 2002), will declare a corrupt header and abort. Conversely, REFLEXW will effortlessly import a DZT file created with matGPR.

Settings. matGPR features a collection of runtime variables that control certain parameters of the program and its modules and can be changed according to your requirements. These are:

Group	Submenu	Choices
1	**Appearance**	Toplevel Background Color Toplevel Foreground Color Analysis Window Background Color Analysis Window Foreground Color Restore Default Colors
2	**Display Mode**	Image Display Wiggle Display Variable Area Display Wiggle + Image Var.Area + Image
3a	**Output Binary File**	'mgp' 'mat'
3b	**MAT-file Save Format**	v7.3 (Default) v6
4	**Save after Read**	'Always" 'Never'

5	**Number of Undo levels**	Any reasonable number
6	**Byte Ordering**	Set Little Endian Set Big Endian
7	**SEG-Y Data Sample Format**	4-byte IBM Floating Point 4-byte Integer, two's complement 2-byte Integer, two's complement 4-byte IEEE Floating Point 1-byte Integer, two's complement
8	**Design F or K Filters**	On a test trace or scanline On the mean trace or scanline Just type the cutoff frequencies

1. Appearance: As of Release 3, matGPR includes a basic colour customization utility. Your options are:

- 'Toplevel Background Color' Change the background of figures related to top level functions
- 'Toplevel Foreground Color' Change the colour of panels, frames, uicontrols etc. of figures related to top-level functions.
- 'Analysis Window Background Colour Change the background of figures related to bottom-level (processing and analysis) functions.
- 'Analysis Window Foreground Colour' Change the colour of panels, frames, uicontrols etc., of figures related to bottom-level (processing and analysis) functions
- 'Restore Default Colors' Reset all colors to the original scheme set by the Author.

The display method can be changed through the **Data → Settings → Appearance** group of options.

2. Display Mode. The variables in this group determine the method of data display. This may be:

- 'Image Display' (the default) ,
- 'Wiggle Display' for simple wiggle-trace presentation,
- 'Variable Area Display' for wiggle-trace variable-area presentation.
- 'Wiggle + Image Display' for combined wiggle-trace / image presentation (image under wiggles).
- 'Variable Area + Image' for combined variable-area / image presentation (image under wiggles).
- The display method can be changed through the **Data → Settings → Display Mode** group of options.

→ The colour scheme for image displays is controlled through the ImageColours menu. The default colour scheme is **'gray'.**

→ When in wiggle-trace or wiggle-trace/ variable are display modes, the amplitude scale and number of wiggles to be shown or printed is controlled by means of the slider-rule interfaces attached to the data display figures (see View Data).

3a. Output Binary File Type: Change the default I/O binary file format. There are two options: The *default* **'mgp'** will cause data to be written in matGPR's own MGP-file format. The alternative is **'mat',** which will select MATLAB's MAT-file format. The MGP-file format was introduced with matGPR Release 2 because, at the time, it was *considerably* more compact (packed the same information in less than half of the space required by MAT-files). This is no longer true, but the MGP format will continue to be supported as the matGPR default. The MAT-file format will *continue* to be supported and used

simultaneously with MGP; it is useful and can be very handy, as all MATLAB users can attest to. Thus, you may still save to, or import from MAT-files. You can change the format using the submenu **Data → Settings → Output Binary File → …** Also see Save to MGP or MAT-file**.**

3b. '**MAT-file Save Format'** controls the format of the MAT-file for I/O operations. On start up, matGPR specifies that the MAT-file format will be the default for the version under which it is running, i.e. it always sets ENVAR.matfileformat = '-mat'**.** This implies MATLAB version 7.3 and later. However, while the MAT-file formats of MATLAB versions 5.x and 6.x are perfectly readable by version 7.x, the converse is not true! If you use MATLAB v7.x and want your MAT-files to be readable by earlier versions choose **Data → Settings → MAT-file Save Format → V.6.** You can switch back to the default by **Data → Settings → MAT-file Save Format → V.7 (Default)** . Also see Save to MGP or MAT-file**.**

4. Save after Read: After reading in a *raw data* file, matGPR may automatically save the same data in the native matGPR binary formats (MGP or MAT) for faster and easier subsequent access. This may be handy when subsequent operations require native data formats (e.g. concatenation, 3-D volume generation etc.), but quite often it is not necessary. You can choose the appropriate course of action using the **Data → Settings → Save after Read → …** menu with options:

- 'Always' : Save after reading to the format specified by '**Output Binary File Type**'.
- 'Never' **:** **DO NOT** save after reading **(default)**.

5. 'Number of Undo levels' determines the number of processing steps you may undo, in order to restore the data to a previous state. The default value is 4. Theoretically, there's no limit to the number of undo levels, but because the previous processing steps are stored in core memory, your computer system will set a practical limit that you have to determine on your own. The number 4 appears to be a good compromise for a Windows XP OS with 2GB of core memory. For additional information please refer to the items Undo/Restore and Clear Undo Buffer .

6. 'Byte Ordering' can is used for exchanging data between little and big endian machines (particularly data in SU and SEG-Y formats). On start up, both variables are automatically adjusted to the parameters of the OS in use. You can toggle between the big and little endian options through the **Data → Settings → Byte Ordering** menu.

7. SEG-Y Data Sample Format: matGPR can use the SEG-Y Revision 0 and 1 standards to import data and *only* Revision 1 standard to export data. In fact, this is the preferred method of exporting data for exchange between different computer systems. The SEG-Y standard offers a choice of different storage formats; those supported by matGPR are listed in Group 7 of the Settings Table and can be selected through the **Data → Settings → SEGY Data Sample Format → …** set of choices. The **default** is **4-Byte IEEE Floating Point**. Full details are given in Section 2.4.

8. Design F or K Filters determines the method of designing frequency or wavenumber filters (see FIR Frequency Filter and FIR Wavenumber Filter for additional information). There are three options:

- **On a test trace or scanline** **:** The filter will be designed on the basis of the spectrum of a *particular test trace* selected by the user. This is the default.
- **On the mean trace or scanline** **:** The filter will be designed on the basis of the spectrum of the *average* (mean) trace and
- **Just type the cutoff frequencies** **:** The filter parameters will be given directly through an appropriate dialog box.

→ The method can be changed through the **Data → Settings → Design F or K Filters** group of options.

CHAPTER 4. Data Visualization and Properties: The "View" menu

Data visualization options include image (colour-coded) displays with several colour maps for better discrimination of the details, wiggle-trace displays and variable-area displays. It is also possible to plot and scrutinize individual traces, trace spectra and their time-frequency characteristics. matGPR also provides utilities to study the attenuation characteristics of the input and processed data, as well as their instantaneous attributes and the centroid frequency of the input data.

Show Data/ Show Processed Data. Display the *Current Input Data* and the *Output Data* respectively, according to the choice made with the **Data → Settings → Display Mode → …** menu. The 'GPR Data' figure is created automatically whenever a new data set is imported or loaded and is updated whenever a processed data set is held (i.e. when the OPD replaces the IPD). It can also be refreshed through the **View → View Data** menu choice at any time. The 'Processed GPR Data' figure is created whenever a processing step is taken, in order to display the OPD, and is destroyed when the OPD is held, discarded, or another processing step is taken.

- The colour scheme can be changed by means of the ImageColours menu.
- The FigureTools menu provides for colour scheme editing, zooming, panning and data inspection, exporting the figure in various graphics formats, printing, and editing the figure, axes and their children graphics objects.

Figure 4.1 shows the matGPR GUI together with the Data and Processed Data figures in "image" display mode. Figure 4.2 shows the 'GPR Data' in wiggle-trace display mode. The vertical slider-rule GUIs facilitate changing the scale of the wiggles (**Scale**) and the number of traces shown (**# Traces**). Figure 4.3 shows the IPD data in variable-area display mode and Figure 4.4 in combined wiggle-trace / image display mode. The slider rules again serve the purpose of adjusting the scale and number of traces shown.

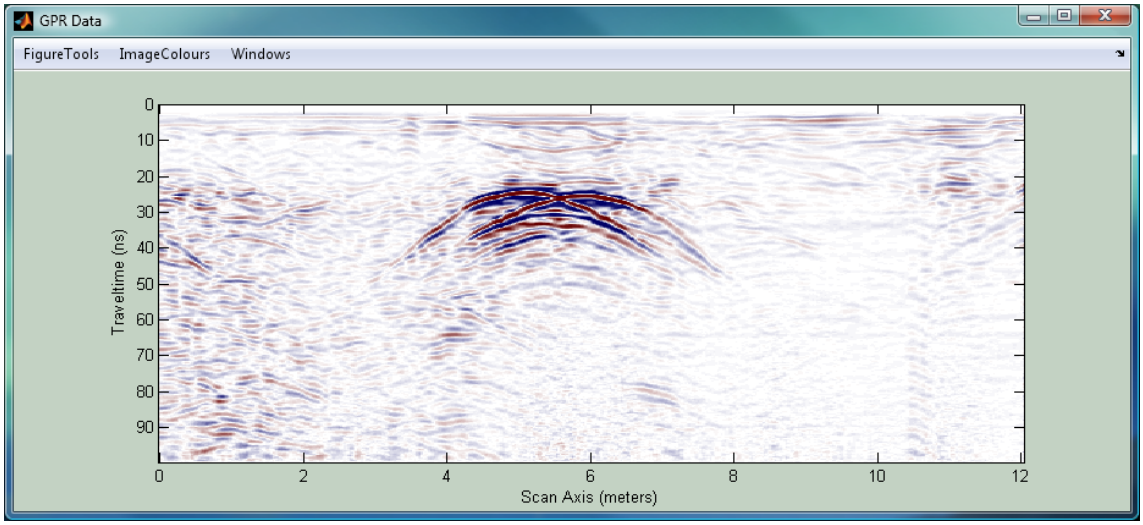


Figure 4.1. The "GPR Data" figure presenting the radargram contained in the IPD structure and in "image" display mode. The OPD is displayed by an identical figure.

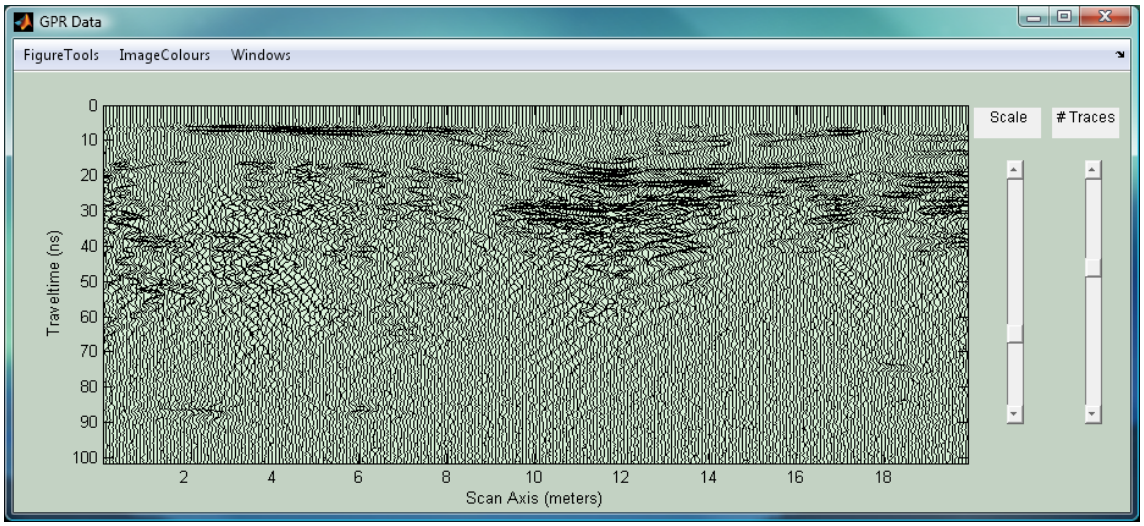


Figure 4.2. The "GPR Data" figure in wiggle-trace display mode.

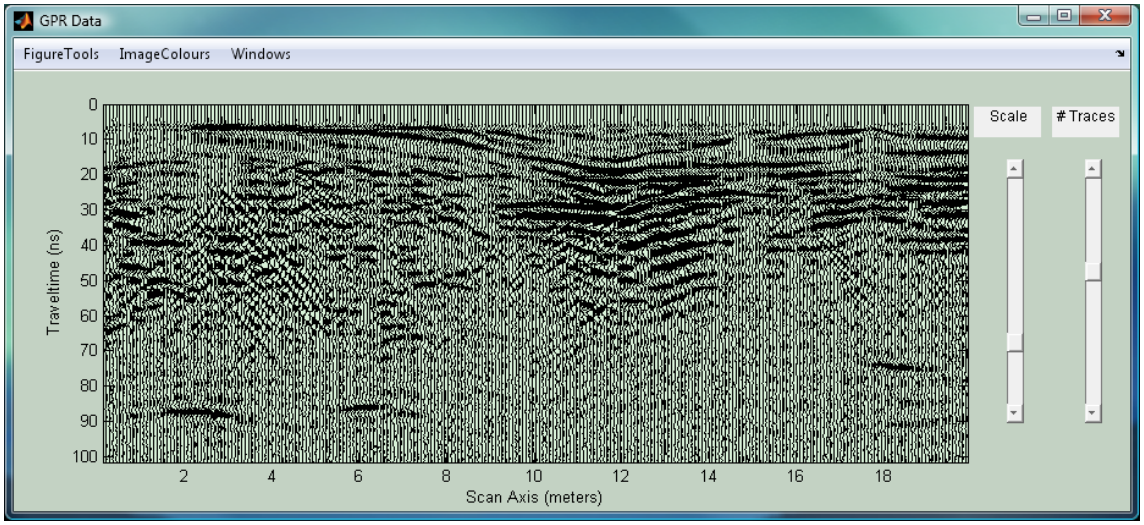


Figure 4.3. The "GPR Data" figure in variable-area display mode.

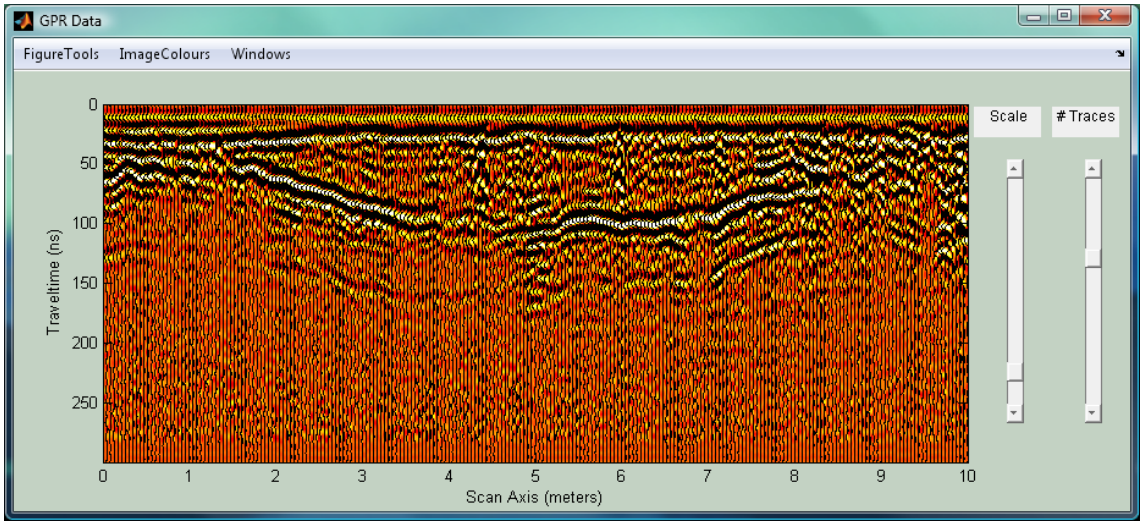


Figure 4.4. The "GPR Data" figure in combined wiggle-trace / image display mode. The image is displayed in a *brightened* ***hot*** colormap.

The "GPR Data" and the "Processed GPR Data" figures feature *context menus* intended to facilitate the handing of the current input and output data and the flow of work. These are accessible by *right clicking* on the axes, or on the images etc. of the Input and Output Data displays. The former ("GPR Data") context menu is shown in the left panel of Figure 4.5 and the latter ("Processed GPR Data") in the right panel of Figure 4.5. Either menu contains decision making items from the "Data menu", while the "Processed GPR Data" context menu includes a utility to facilitate the inclusion of topography to depth-migrated data (Fig. 4.5 right).

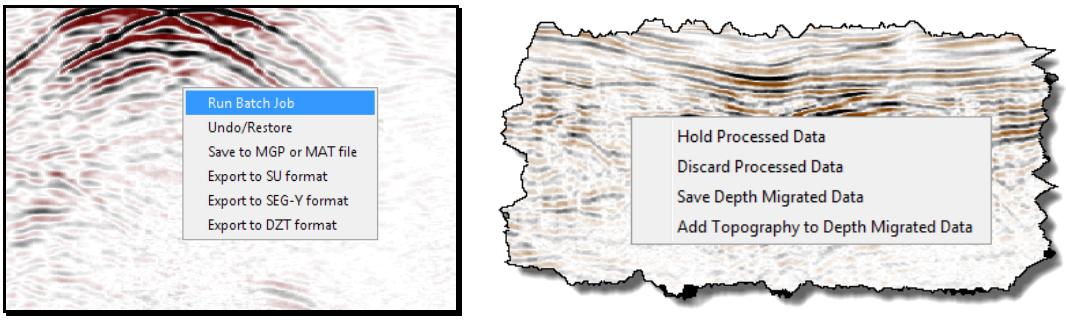


Figure 4.5. The context menus available in the "GPR Data" (left) and the "Processed GPR Data" figures (right).

Inspect Traces. This is the first of three utilities intended to facilitate the detailed study of the input or output data. It comprises a "**Trace Viewer**" to visualize and inspect the traces of the input or output data (Figure 4.6). The program uses a *horizontal* slider to help you *scan* through the traces and a *vertical* slider to zoom into the earlier times of the traces. Further scrutiny into the time series of any individual trace is possible with appropriate graphical tools (e.g. zooming, panning and inspection/ peeking) available from **FigureTools** menu.

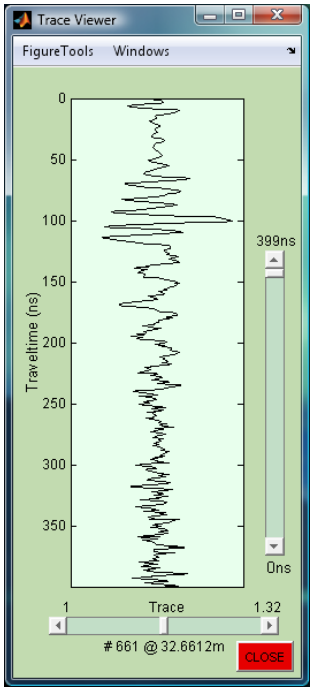


Figure 4.6. The "Trace Viewer" utility.

Inspect Trace Spectra. The second of three utilities intended to facilitate the detailed study of the input or output data. It comprises a "**Spectra Viewer**" to visualize and inspect the Fourier amplitude spectra of individual traces (Figure 4.7). The program also uses a *horizontal* slider to *scan* through the traces and a *vertical* slider to focus at the lower frequencies of the trace spectra. Further scrutiny into the spectrum is possible with appropriate graphical tools (e.g. zooming, panning and inspection/ peeking) available from **FigureTools** menu.

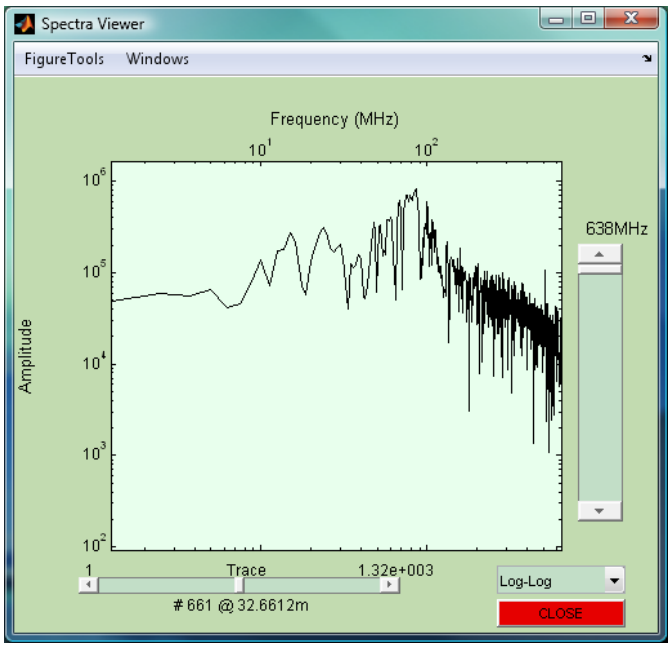


Figure 4.7. The "Spectra Viewer" utility.

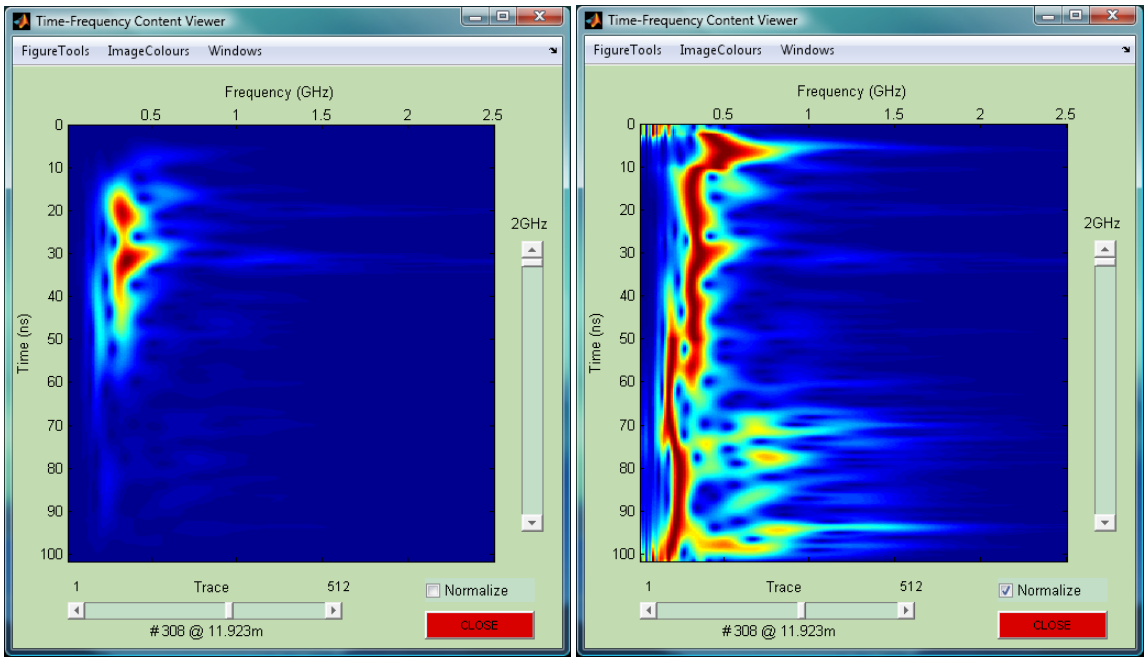


Figure 4.8. The "Time-Frequency Spectra Viewer" utility.

Inspect Time-Frequency Spectra. This is the third of the utilities intended for the detailed study of the input or output data. It computes a *time-frequency representation* of individual traces using the ultra-high resolution S-transform (Stockwell et al. 1996) and displays its amplitude it in the "**Time-Frequency Content Viewer**" (Figure 4.8-left). This is very helpful in assessing the characteristics of propagation in complex media (e.g. existence of dispersion etc.; see also "Centroid Frequency" for additional information). As before, the program also uses a *horizontal* slider to *scan* through the traces and a *vertical* slider to focus at the lower frequencies of the time-frequency content. It is also possible to display normalized spectra by checking the "Normalize" box (Figure 4.8-right). Further scrutiny is possible with the tools available from the **FigureTools** menu (e.g. zooming, panning and inspection/ peeking).

Attenuation Characteristics. This option allows visualization and scrutiny into the propagation and attenuation properties of the GPR signal. The program computes the analytic signal for all traces in the IPD or OPD data, hence their instantaneous power. Then it computes the median and a mean attenuation function, that is, the respective median and mean instantaneous power of all traces in the section. It also determines best fitting models for power-law and exponential attenuation based on the median attenuation function. Finally, it displays the attenuation functions and the best fitting power-law and exponential decay models as per Figure 4.9. The result of this operation may allow insight into the properties of the propagation medium and facilitate gain manipulations.

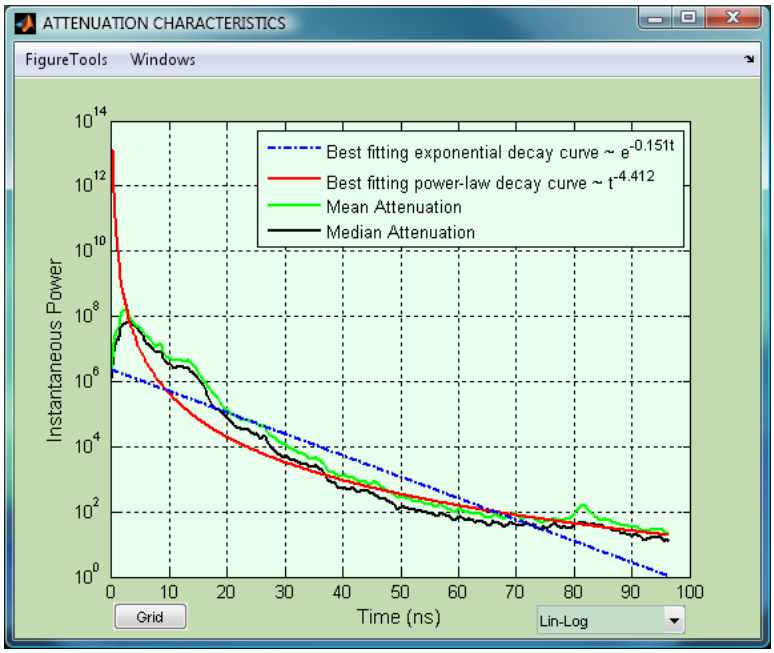


Figure 4.9. The "Attenuation Characteristics" figure. It features the median and mean power attenuation functions and the best fitting power-law and exponential decay models.

Instantaneous Attributes. This will calculate the analytic signal and hence the instantaneous attributes of GPR traces. The instantaneous amplitude of the input sequence is the amplitude of the analytic signal. For GPR data it measures the reflectivity strength, reducing the appearance of random signal in the data. The instantaneous phase angle of the input sequence is the (unwrapped) angle of the analytic signal; the instantaneous frequency is the time rate of change of the instantaneous phase angle.

Centroid Frequency. Computes and displays the frequency of the *Spectral Centroid*, i.e. the location of the "centre of mass of the signal spectrum, as a function of time and distance. This is calculated as a weighted mean of the frequencies present in the signal:

$$f_c = \frac{\int_0^\infty fS(f)df}{\int_0^\infty S(f)df},$$

Mapping the centroid frequency offers a synopsis of the location of the spectral content of the data, hence a measure of changes in propagation conditions. For instance, the centroid frequency is expected to decrease (increase) as a function of time, as the signal enters high (low) attenuation domains and is also expected to exhibit consistent gradual downshift in cases of dispersive propagation (e.g. Irving and Knight, 2003). The calculation of the centroid frequency is based on an *ultra-high resolution* time-frequency representation of a time series, i.e. of the amplitude spectrum at each instant along a trace, computed with the S-transform (Stockwell et al., 1996).

The centroid frequency is computed and displayed in 'Processed Data' mode (Figure 4.10). This means that the centroid frequency matrix (or image) can be held and further manipulated (e.g. smoothed) with other filtering methods.

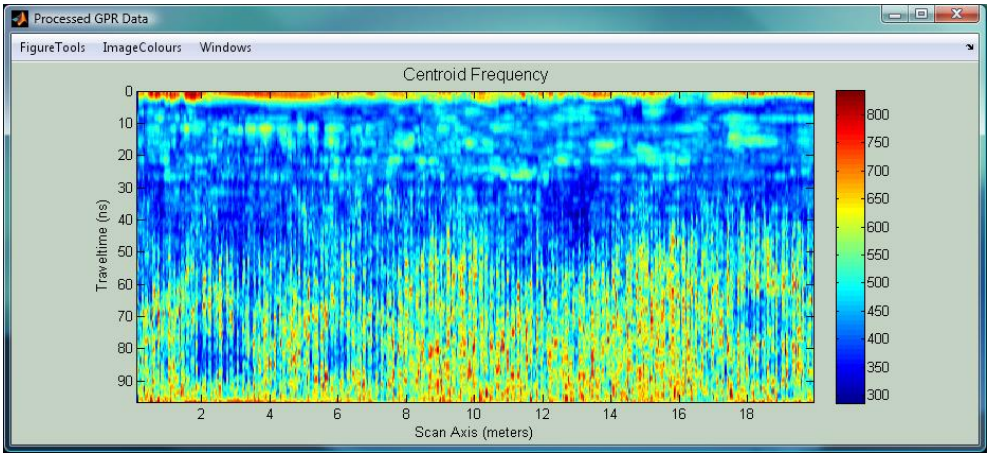


Figure 4.10. The centroid frequency map of the data shown in Figures 3.2.2 and 3.2.3. Note the blue pockets of low centroid frequency inside red areas of high centroid frequency at traveltimes longer than 70 ns: the blue pockets correlate with dull reflections in the observed data and indicate areas where high frequencies are severely attenuated. This, for instance, may indicate the presence of pockets of moisture.

Curvature Attributes. Computes and displays Curvature Attributes applicable to 2-D GPR data. The interpretation of 2-D GPR data can be supported by imaging and analysis of the most positive, most negative, maximum and minimum curvatures, which are attributes successfully used for fault/edge detection in 3-D seismics (for a comprehensive introduction see Roberts, 2001; Chopra and Marfurt, 2007; references therein). The extrinsic curvature of a surface can be defined as the rate of change of the angle through which the tangent to a surface turns while moving along a curve on the surface, or even more naïvely, how fast the surface bends. By convention, the curvature at a point on a surface is positive when vectors normal to the surface diverge in the vicinity of that point, i.e. when the shape of the surface is convex as in hills, ridges etc. Conversely, the curvature is negative when vectors normal to the surface

diverge in the vicinity of a point, i.e. when the surface is concave as in valleys. Out of the infinity of possible curves and corresponding curvatures on a three-dimensional surface, the most useful subset is the one defined by curves in the directions of the normals to the surface, i.e. by curves defined by the intersection of the surface with planes normal to the surface. The curvatures of such curves are called *normal*. The *maximum curvature* is the absolutely largest of all possible normal curvatures determined at a given point. Conversely, the *minimum curvature* is the absolutely smallest of all normal curvatures. The *most positive curvature* K_+ at a point on a surface is the largest of all positive values determined from all possible normal curvatures at that point. Conversely, the *most negative curvature* K_- is the smallest of all negative values determined from all negative normal curvatures. By mapping the maximum, minimum, most positive and most negative attributes one obtains edge-type images. Lineaments of maximum/minimum curvature attributes are effective in delimiting discontinuities and discontinuity geometries. Lineaments of high K_+ values trace ridges and lineaments of low K_- values trace valleys and troughs in the texture of the data. What is important to clarify in this necessarily parsimonious introduction, is that curvature attributes can be used to identify faulting and fragmentation down to the level of individual rock blocks, because these would be delimited by the ridges and troughs of reflections from their interfaces with neighbouring blocks, provided of course that said interfaces are sufficiently reflective.

On selecting the "**Curvature Attributes**" option, matGPR will respond with a GUI asking the user to specify which attribute to compute (see below). By checking/unchecking the "**Equalize**" box, the user may also specify whether to equalize the data before attribute computations commence, or not. The default is not to equalize for Most Positive and Most Negative curvature computations and to equalize for Maximum and Minimum curvature computations. The curvature attributes are computed by clicking "**GO**".

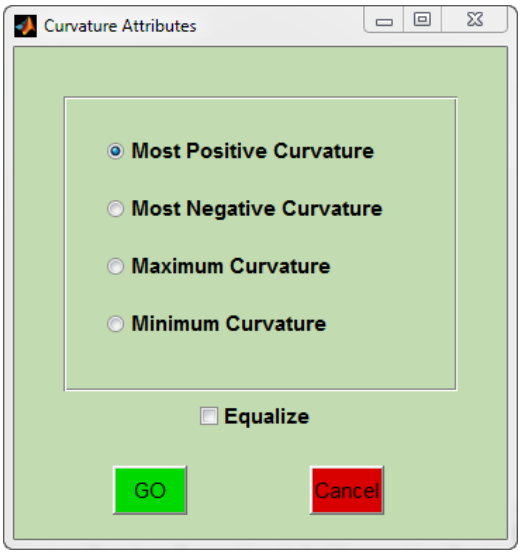


Figure 4.11 shows a radargram measured on a levelled surface above massive limestone fragmented by conjugate normal faulting and jointing; karstification has nucleated along the fault walls and many of the resulting voids were subsequently filled with ferriferous argillaceous material of lateritic composition. The radargram is shown after global background removal, amplification with the Inverse Amplitude Decay technique and F-K migration with a uniform velocity of 0.085 m/ns. Features observable in the migrated section are down-dipping reflections attributed to faults and fractures, *very* faint up-dipping quasi-linear reflections also attributed to fractures and clusters of strong reflections along the larger fractures and fracture zones, (e.g. between 5–10m and 40–80ns), sometimes associated with the intersection of down- and up-dipping fractures. Figure 4.12 illustrates the corresponding most negative curvature. It exhibits alignments corresponding to quasi-linear reflections observable in Fig. 4.11. At

distances 5–11 m and traveltimes 20–80 ns, the rock appears to be heavily fragmented and brecciated; it comprises a mosaic of blocks whose outlines and orientations are clearly associated with up- and down-dipping fractures. The heavy apparent fragmentation indicates that this area may correspond to a down-dipping fault zone. At distances beyond 15m "fragmentation" is less intense but still present, within and immediately around fractures. The quality of estimation deteriorates sharply with increasing noise, so that curvature attributes cannot yield reliable information at traveltimes longer than 80ns.

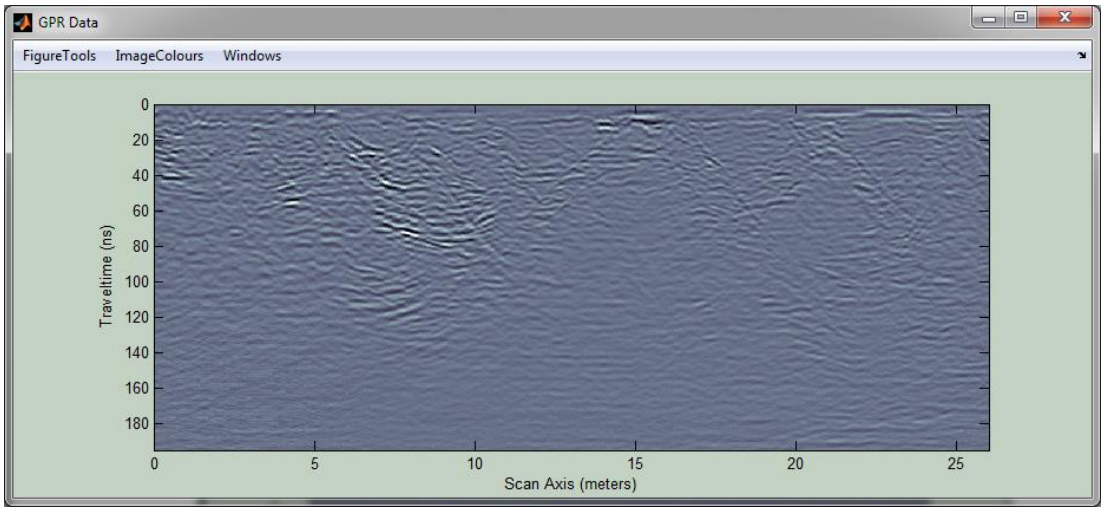


Figure 4.11. Radargram measured above massive fragmented limestone. The data has been migrated with a uniform velocity of 0.085 m/ns.

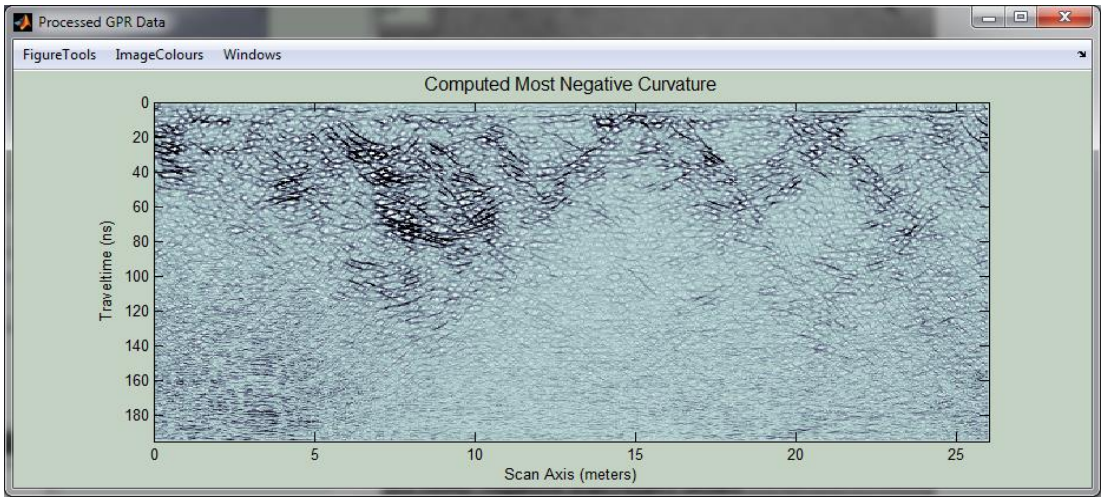


Figure 4.12. The Most Negative curvature attribute computed from the data of Fig. 4.11.

Show Markers. Overlays **Marker Trace** locations on the "GPR Data" figure.

CHAPTER 5. Basic Processing: The "Basic Handling" menu

Basic data handling (conditioning) includes the planning and execution of batch jobs (sequential processing operations) in single or multiple data sets, determination and adjustment of time-zero, basic trace editing (trimming and extraction utilities), dewow and gain manipulation. The latter includes Automatic Gain Control (AGC) functions, both in the standard form and with Gaussian tapering, as well as compensation for the amplitude or power attenuation of the signal. Other utilities include resampling (increase or decrease of the sampling-rate) in time and in space (along the scan line), using the versatile band limited sinc interpolation algorithm of Smith and Gosset (1984). For GPR instruments not equipped with survey wheels or other automatic distance measuring and triggering devices, a suite of marker interpolation routines is also provided, so as to transform data collected at equal-time spacing mode, to data at equal-distance spacing. This suite of routines also facilitates the three-dimensional determination of trace locations, with respect to some local co-ordinate system. Such information is necessary for the application of static corrections prior to imaging, as well as for 3D data visualization among other things.

Set up Batch Job. matGPR allows for many processing operations to be applied in batch mode (batch jobs), thus saving valuable time. The operations available for execution in batch mode and the relevant required parameters/ options are shown in Figure 5.1 and include:

A. Items for the "**Basic Handling**" menu (Chapter 5):

(1) Time-zero adjustment (Adjust Signal Position).
(2) Radargram size reduction by discarding late-time arrivals (Trim Time Window).
(3) Per-trace subtraction (removal) of the DC component (Remove DC).
(4) Wow elimination (Dewow).
(5) Deamplification of raw SIR-200/3000 DZT data (removal of DZT Header Gain).
(6) Amplification by Standard Automatic Gain Control (Standard AGC);
(7) Amplification by Gaussian-tapered Automatic Gain Control (Gaussian-tapered AGC).
(8) Amplification by the Inverse Power Decay method.
(9) Amplification by the Inverse Amplitude Decay method.
(10) Modification of the sampling rate (Resample Time-Axis).
(11) Modification of trace spacing – spatial resampling (Resample Scan Axis).
(12) Trace equalization (Equalize).

B. Items from the "**Filtering**" and "**Imaging**" menus (Chapters 6 and 7):

(13) Global Background or mean trace subtraction.
(14) Karhunen – Loeve Filtering (lower-dimensional subspace approximation)
(15) FIR Frequency Filtering
(16) B-Spline Wavelet Filtering, that is Directional Wavelet Filtering applied *only* along the temporal axis (zero azimuth).
(17) F-X Deconvolution.
(18) Predictive Deconvolution
(19) **1-D Migration** with choice between F-K Migration (Stolt) or Phase-shifting Migration (Gazdag). The velocity model must be imported from disk file; for details see"1-D velocity model".
(20) 1-D Time-to-Depth Conversion. The velocity model must be imported from disk file; for details see "1-D velocity model".

Jobs in the batch are executed sequentially and the GUI of Figure 5.1 will assist you in setting up the execution sequence (*batch queue*). To do so, you have to set the **priority** (**STEP #**) of each operation you need to include in the batch, as well as the relevant execution parameters. In the example of Figure 5.1,

the programmed sequence of operations (queue) is: 1) Trimming of the Time W; 2) Global Background Removal; 3) DC Removal; 4)Resampling of Time Axis to 512 samples; 5) Resampling of Scan Axis to 512 samples; 6) F-X Deconvolution; 7) Time-to-Depth Conversion with velocity model imported from file "*veldata_test.dat*". Also note that:

a. Jobs without set priority (STEP #) are not executed. If you want to exclude or remove a job from the queue, simply *eliminate* the STEP # in the pertinent panel (as in Figure 5.1).

b. The routine does not check the queue for repetitions of the same priority (STEP) number. At present, this responsibility is left to the user who must exercise due caution.

c. The parameters required for the execution of the operations included in the queue are supplied through the relevant controls of Figure 5.1. The nature, significance and function of the parameters can be found in the relevant entries of Chapters 5 and 6, pointed to by the hyperlinks above.

d. The queue and execution parameters are stored internally and can be used time and again with different data sets, until they are changed by a new execution of "**Set up Batch Job**".

Figure 5.1. GUI to set up and store a batch job.

Run Single Batch Job. Execute a batch queue with priority and parameters retrieved from memory and prepared using **Basic Handling → Set up Batch Job**. The queue and parameters can be used repeatedly

with different data sets. Moreover, because they are saved to/ loaded from the matGPR preferences file, they are available for use at later analysis sessions.

Run Multiple Batch Jobs. Execute the same batch queue for multiple GPR data files with queue and parameters retrieved from memory. The GUI of Figure 5.2, (which is similar in layout and function to the Concatenate Sections window of the 'Data" menu), facilitates the assemblage of the list. The files in the list may be in *any* of the allowed data formats (DZT, RD3, DT1, SEGY, ZOND SEGY, SU, MAT and MGP), as specified in the "Import Raw Data" item. The list is assembled by left-clicking the **Browse/Add** button and invoking the file browser. Multiple files can be selected using the Cntrl and Shift keys. Files can be removed from the list by selecting them and left-clicking the **Remove from list** button. At present, the *GPR data files cannot span multiple directories* and the program does not issue a warning.

The **Proceed** button begins the execution of the batch job on each file in the list. matGPR displays each of the input and processed data files but does not store the resulting IPD internally: instead, *it exports* the each of the processed data files in the preferred matGPR format (MAT or MGP) for later analysis.

As with single batch jobs, the queue can be used repeatedly with different data sets and because it is saved to/ loaded from the matGPR preferences file, it can also be used at a later time: the queue remains unchanged and available until it is modified by a new execution of Set Up Batch Job.

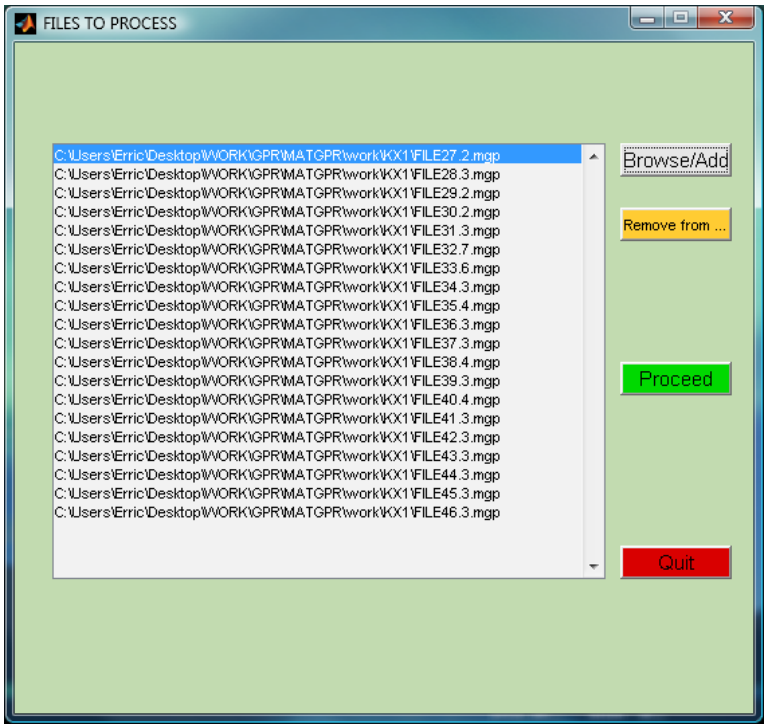


Figure 5.2. GUI to assemble a list of files on which to execute batch jobs.

Adjust Signal Position. This option allows control of the vertical position of the surface reflection, i.e. the place in time where the radar pulse leaves the antenna, and enters the subsurface. It can therefore be considered to be "time zero", and its position should be at the top of the scan. Modern GPR systems are usually supplied with built-in utilities, manual or automatic, to identify the surface reflection and place it correctly at the top of the time window. Quite often, the system overshoots, leaving a zone of flat-line data at the top of the trace (no signal). The large reflection just below the flat data zone will be the surface reflection. Sometimes system undershoots or is incorrectly set up. At any rate, proper positioning of the

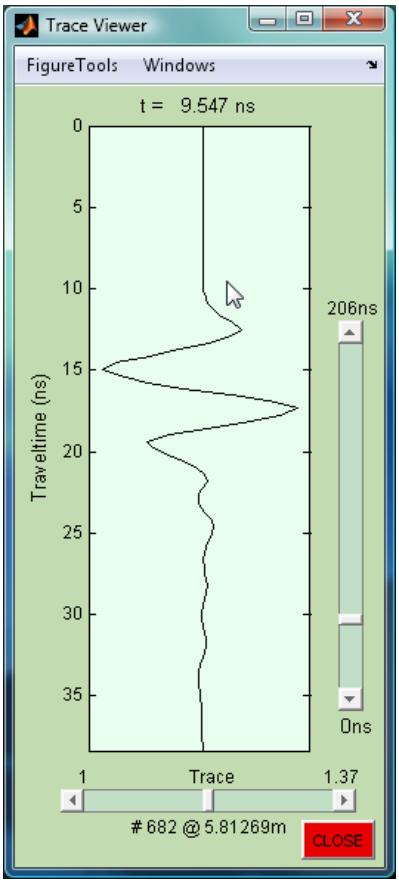
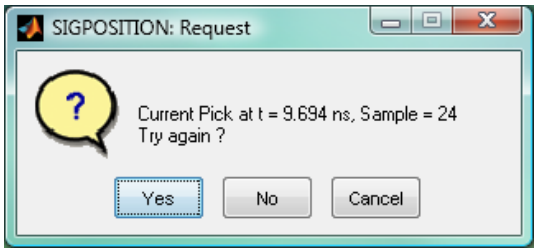
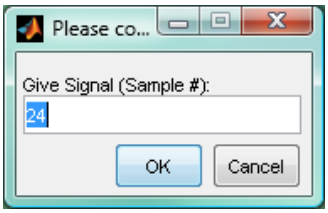


Figure 5.3. A snapshot in the course of determining the true time-zero of GPR data

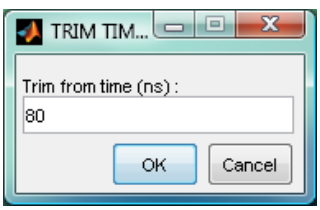
time-zero is necessary, otherwise the location of sub-surface reflectors may be misidentified. The post-acquisition determination of time-zero amounts to a static correction. matGPR provides for the graphical determination and adjustment of time-zero using the "Trace Viewer" utility.

Details of this operation are shown in Figure 5.3. You select a representative trace and zoom in at the beginning of the time window using the vertical slider rule. The time-zero is determined by pointing with the cursor and clicking the left mouse button. The location of the cursor is shown above the axes. The system's estimate of pulse duration, if any, is displayed at the matGPR information window and may assist in making a better decision. matGPR displays the current pick through the "Request" interface and waits for your decision (see below). By pressing **Yes** you loop to improve your selection. By pressing **No** the program asks for confirmation by means of the 'Please confirm' dialog box, in which you can fine-tune your decision if necessary. Pressing **OK** completes the operation.





Trim Time Window. This option reduces the size of the GPR data matrix by discarding the late arrivals. Very often, the later parts of the traces contain only useless noise occupying valuable memory. This option is also useful when one wishes to extract a certain part of the data for some specific purpose. You need to provide the *cut-off time* at which the time window will be trimmed: the data *after* this time will be discarded). matGPR displays a menu asking how you wish to enter the required information.



If you choose to '**Use Cursor'** the program focuses on the 'GPR Data' figure and a hairline cursor appears. You enter the cut-off time by pointing and left clicking at the desired location along the time axis. If you choose to '**Use Fingertips'**, the program displays a dialog (above right) in which you enter the exact cut-off time in nanoseconds.

Edit Scan Axis. This option reduces the size of the GPR data array by discarding groups of traces at the beginning or at the end of the section's scan axis (scanline). It is also used to extract groups of traces or even large parts of the section to a separate data set. The routine initializes a GUI, in which you specify the desired operation (e.g. trim, or extract, input mode, see Figure 3.3.4). The table below explains how to use the GUI. By pressing **Go** matGPR performs the desired operation.

- If the data (IPD.d) is trimmed, then the origin of the scan axis (IPD.x) is reset, i.e. the first trace of the trimmed data will be at 0 m. Note also that if Marker Traces exist and IPD.markertr is assigned, the ID numbers of the Marker Traces are reset but their coordinates (assumed fixed with respect to a local reference frame) do not change. Likewise, if you have created XYZ coordinates of the traces, then IPD.xyz is trimmed but the remaining coordinates remain fixed w.r.t. the reference frame and do not change.

Input mode	Desired action.
Use Pointer selects location by pointing the hair-line cursor and clicking on the "GPR Data" figure.	**Trim Front** removes groups of traces at the front of the scan axis (beginning of the profile).
Use Fingers allows direct input of location in the edit boxes below.	**Trim Back** removes groups of traces at the rear of the scan axis (toward the end of the profile).
✓ The edit boxes are disabled when 'Use Pointer' is checked (Figure 5.4.left).	**Extract** removes a group of traces to a separate data set.
✓ Edit boxes are enabled when 'Use Fingers' is checked, but which edit box is enabled depends on the desired action, i.e. which check-box is activated (Figure 5.4.right).	

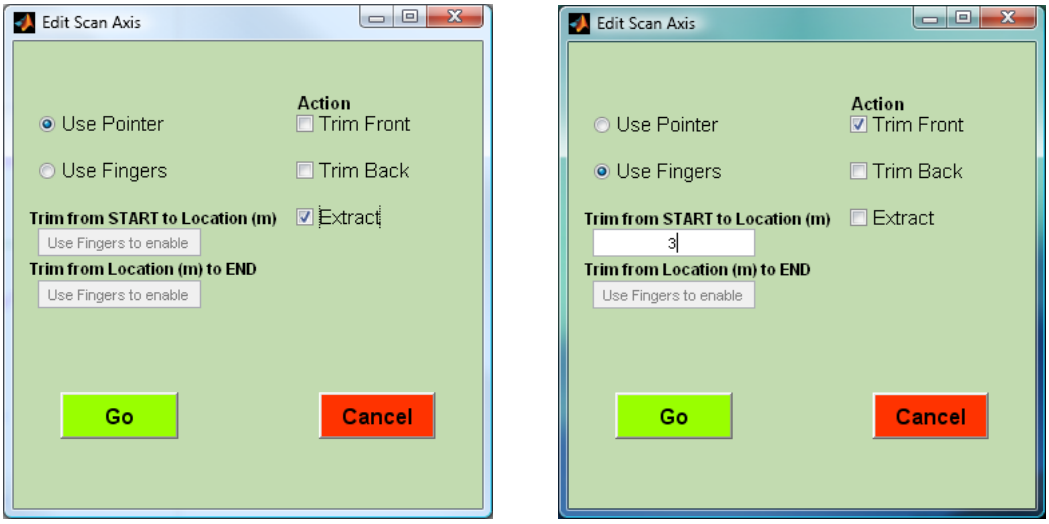


Figure 5.4. The **Edit Scan Axis** GUI: **Left**, when in graphical (**Use Pointer**) input mode and **right,** in direct (**Use Fingers**) input mode.

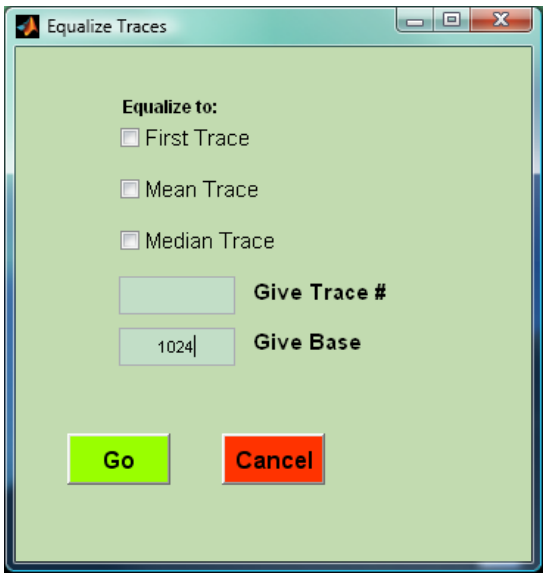
Remove Bad Traces. Pick bad traces by pointing and clicking and replace them with interpolants computed from their near neighbours. The routine displays the input GPR section in the display mode determined by ENVAR. Then, it issues a short help message at the lower left corner of the screen and with a delay of 0.5s it displays a cross-hair cursor. Pick the bad traces by pointing with the cursor and clicking the *left* mouse button. To facilitate positioning, the cursor co-ordinates are displayed at the bottom of the figure. Zooming and panning can also be used. A selected bad trace is marked with a vertical red line. If you make a mistake, (pick a wrong trace), you can undo it by clicking the *middle* mouse button. To finish, click the *right* button. The routine then uses the near neighbours of the bad traces to compute their interpolant substitutes and replaces the bad traces in the output data section.

- Note, because this function uses interpolation, it works best for *isolated* bad traces, or small clusters of adjacent bad traces. In the latter case the clusters should better be separated by relatively extended groups of consecutive good traces. For the same reason, the function is *not recommended* for removing extended groups of bad traces, as it will probably fail to produce reliable interpolants.

Remove DC. This will remove the DC component (arithmetic mean) from each trace of the GPR section.

Dewow. This operation will eliminate wow by applying a zero-phase high pass FIR filter with a cutoff frequency at exactly 2% of the Nyquist.

Equalize traces. Make the sum of the absolute values of all samples in a trace the same for all traces. For instance, if the trace that is selected to be used for equalization has a sum of all absolute values of 1000, then the sum of absolute values of every trace in the data set will be set to 1000 using a multiplying factor. When called, the routine displays a GUI, in which the user specifies the trace which will be used as the *base trace*.

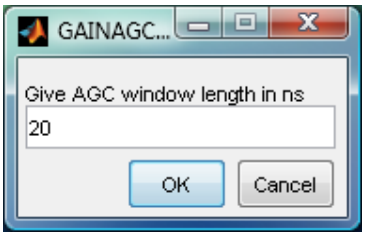


This can be a) the **First Trace**; b) the **Mean trace**; c) the **Median Trace**; d) any trace specified by the user in the '**Give Trace #**' box, and, e) any positive number (*base value*) provided in the **'Give Base'** box.

Remove DZT header gain. GSSI DZT data usually have time-varying (range) gain applied and the gain breakpoints and values are stored in the file header. On importing a DZT data set, the range gain data is stored in the IPD.DZThdgain field. The header gain information is preserved during manipulations that change the size of the data array (e.g. signal position adjustments, trimmings, marker interpolation, resampling, etc.) and is possible to remove at any time, recovering the raw data without amplification. This operation is useful if you wish to compare data files acquired with different gain functions or use true amplitude information.

- This option applies **ONLY** to GSSI, SIR-2000 and SIR-3000 DZT data. The other formats data usually do not have gain applied.

Standard AGC. *Automatic Gain Control* (AGC) is a process by which gain is automatically adjusted in a specified manner, as a function of a specified parameter, such as signal level. In matGPR, the time-varying signal level is measured by the RMS amplitude computed over a sliding time window of a given length. matGPR requests the length of the window in nanoseconds, by means of a dialog box:

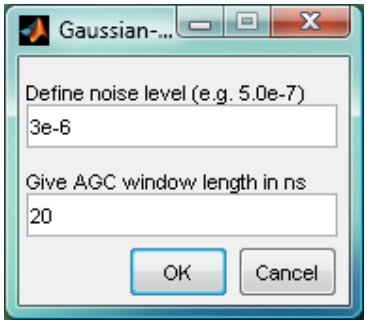


By scaling the amplitude of the data at the centre of the sliding window with respect to the RMS amplitude of the window, the process ensures that ranges of low amplitudes are emphasized with respect to ranges with high amplitudes. Information of the true (measured) amplitude is lost. The result depends essentially on the length of the AGC sliding window. Short lengths (e.g. a very few ns) will introduce a more uniform distribution in the amplitudes of the output traces and will tend to smear the stronger reflections. On the other hand, very long windows may not provide adequate amplification of low amplitude ranges. Inasmuch as each data set has its own peculiarities, experimentation is recommended prior to holding the output data.

Gaussian-tapered AGC. *Automatic Gain Control* (AGC) is a process by which gain is automatically adjusted in a specified manner, as a function of a specified parameter, such as signal level. In matGPR, the time-varying signal level is measured by the RMS amplitude computed over a sliding time window of a given length. However, in contrast to the simple boxcar window of the standard AGC operation described above, this variant a uses a boxcar window weighted (tapered) with a Gaussian bell function, whose breadth is a function of the noise level. matGPR will request an estimate of the noise level (if unknown, zero is an acceptable figure) and the length of the AGC window in ns.

By scaling the amplitude of the data at the centre of the sliding window with respect to the RMS amplitude of the window, the process ensures that ranges of low amplitudes are amplified with respect to ranges with high amplitudes. However, the Gaussian taper emphasizes the contribution of the data around the centre of the window, thus producing a more focused result than standard AGC. Information of the true (measured) amplitude is lost. Here as well, the result depends strongly on the length of the AGC

sliding window. Inasmuch as each data set has its own peculiarities, experimentation is recommended prior to holding the output data.



Inverse Amplitude Decay. Applies an empirical gain function which exactly compensates the mean or median attenuation observed in a 2-D GPR section. The procedure:

(i) Computes the analytic signal for all traces in the GPR section, hence their instantaneous amplitude.

(ii) Computes a median and a mean amplitude attenuation function, i.e. respectively the median and mean instantaneous amplitude of all traces in the section.

(iii) Computes empirical best fitting attenuation models with a function of the form

$$A(t) = c_1 \, \mathrm{e}^{-a_1 t} + c_2 \, \mathrm{e}^{-a_2 t} + \cdots + c_N \, \mathrm{e}^{-a_N t}$$

with N linear parameters and N non-linear parameters.

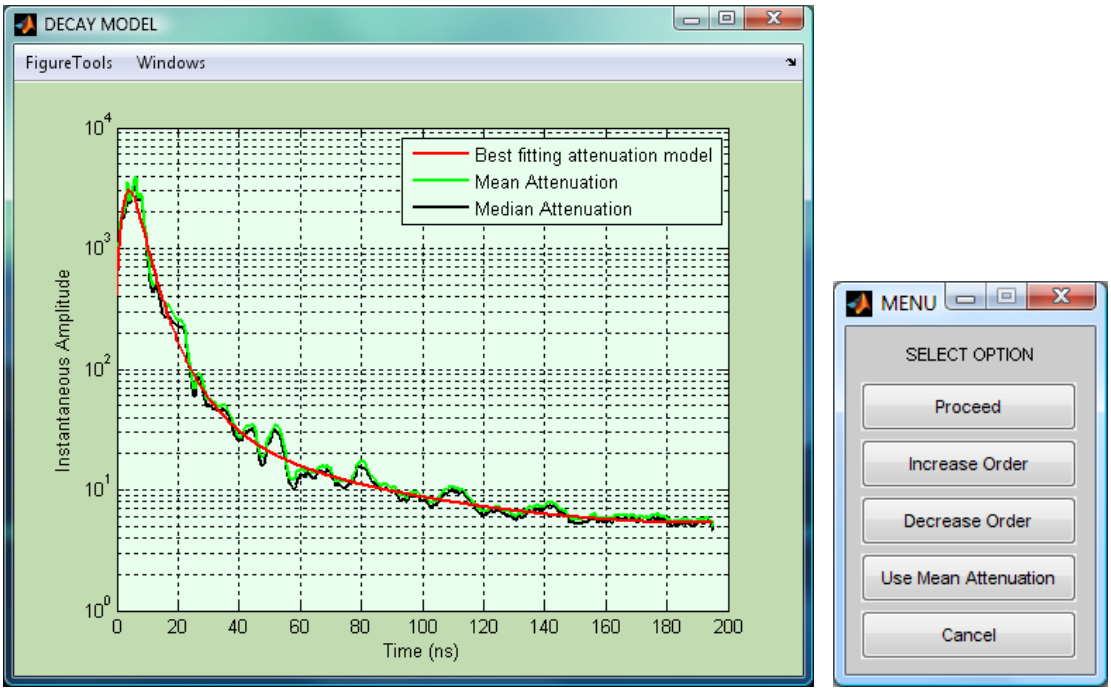


Figure 5.5. The **left** panel shows the result of the Inverse Amplitude Decay analysis. The **right** panel shows the floating menu to be used for obtaining an optimal decay (hence gain) function.

(iv) Displays the mean and median attenuation and best fitting model (e.g. Figure 5.5 left) and by means of a floating menu (Figure 5.5 right), allows you to experiment with the *order N* of the fitting function (**Increase Order** or **Decrease Order** buttons) and explore the merits of using the

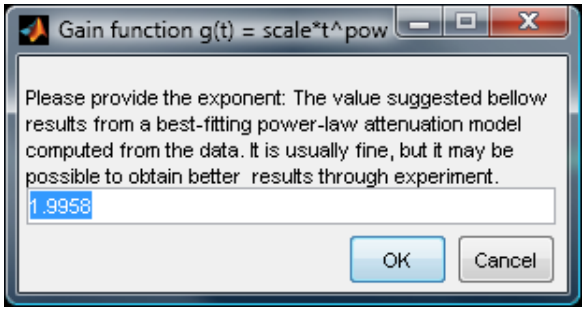
median or mean amplitude decay curve (**Use Mean Attenuation** or **Use Median Attenuation** buttons) to obtain an optimal gain function. By pressing **Proceed**, the fitting session is terminated and the gain function is computed.

(v) The gain function is the normalized inverse of the amplitude decay model, i.e. has the form

$$g(t) = \left(\frac{A(t)}{\max A(t)} \right)^{-1}$$

and produces a practically true white noise series, while preserving relative signal amplitudes across the scan axis (i.e. at similar 2-way traveltimes).

Inverse Power Decay (apply g(t)=scale * t^power). This option applies a gain function of the form $g(t) = scale \times t^{power}$. You will be asked to supply the *power*. The function computes the attenuation characteristics of the input data and obtains a best fitting power-law attenuation model. The suggested *power* is the exponent of the best-fitting model less 1. This is usually a very good solution, but experimentation may provide a result more suitable for the data. You can inspect the attenuation characteristics and the best fitting power-law model by selecting Attenuation Characteristics under the **View** menu. The *scale* is computed automatically, so as to preserve the amplitude range values of the input data (compensate for the effects of t^{power}).



Experience shows that for zero-mean and time-zero adjusted data, the suggested *power* is around 2 and the observed attenuation is fractional and around 3. From a practical point of view, *power* ~ 3 applies little gain, or even attenuates the early (near surface) arrivals, while applying too much gain and emphasizing the late arrivals. On the other hand, *power* ~ 2 apportions higher gain at the early parts of the time window and lower gain at the late parts, providing for a more balanced eventual distribution of amplitudes. Claerbout (1996, pp. 222-223) gives a theoretical justification for using *power = 2* in seismic data. He shows that the basic geometry of energy spreading predicts a single power of time for the spherical divergence correction and argues that an additional power arises from a simple absorption calculation: This justification has not been demonstrated and may, or may not hold water in the case of GPR data. This is a theoretical exercise for the future, but is nevertheless worth noting!

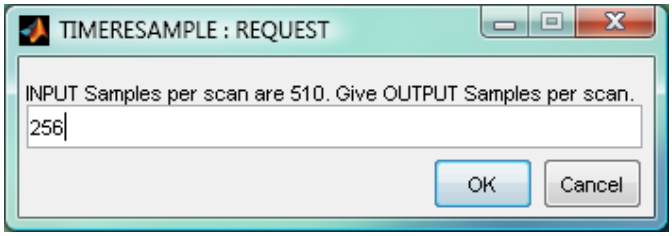
Resample Time Axis/ Resample Scan Axis. This option will increase or decrease the sampling rate, either in time (IPD.dt) or is space (IPD.dx), using band-limited sinc interpolation.

Band-limited interpolation of discrete-time signals is a basic tool with extensive application in digital signal processing. The problem is to correctly compute signal values at arbitrary continuous times from a set of discrete-time samples of the signal amplitude (interpolate the signal between samples). Since the original signal is always assumed to be band-limited to half the sampling rate, (otherwise aliasing distortion would occur upon sampling), Shannon's sampling theorem tells that the signal can be exactly and uniquely reconstructed for all time from its samples by band-limited interpolation. Crochiere and

Rabiner (1983) provide a comprehensive summary of classical signal processing techniques for sampling-rate conversion. In these techniques, the signal is first interpolated by an integer factor L and then decimated by an integer factor M. This provides sampling-rate conversion by any rational factor L/M. The conversion requires a digital low pass filter whose cut-off frequency depends on max{L, M}. While sufficiently general, this formulation is less convenient when it is desired to resample the signal at arbitrary times / locations or change the sampling-rate conversion factor smoothly over time.

matGPR employs a flexible public-domain resampling algorithm, which will evaluate a signal at any time or location specifiable by a fixed-point number (Smith and Gosset, 1984). In addition, one low pass filter is used regardless of the sampling-rate conversion factor. The algorithm effectively implements the "analogue interpretation" of rate conversion, as discussed in Crochiere and Rabiner (1983), in which a certain low pass filter impulse response must be available as a continuous function. Continuity of the impulse response is simulated by linearly interpolating between samples of the impulse response stored in a table.

The resampling routines have only one request: The output number of samples in the time dimension (Samples per scan / OPD.ns), or the output number of traces in the horizontal (spatial) dimension (OPD.ntr).



Edit Markers. matGPR provides a suite of transformation and positioning utilities. For GPR instruments not equipped with survey wheels or other automatic distance measuring and triggering device, this allows to use information about the *known* positions of certain *marker traces* in order to transform data collected in equal-time spacing mode to data with equal-distance spacing (see Transform to Equal Spacing below). In all other cases, the known positions of the marker traces help in assigning coordinates to all traces with reference to the coordinate system of the survey. For PULSE EKKO (DT1) data, these utilities are available ONLY when the data is collected in REFLECTION MODE. Key to this operation is the marker information matrix (i.e. the IPD.markertr field), which must be a 4 column matrix as follows:

1) First Column : The ID (sequential) number of the Marker Traces;
2) Second Column : The x-coordinates of the Marker Traces in a local reference system;
3) Third Column : The y-coordinate of the Marker Traces in the local reference system;
4) Fourth Column : The z-coordinates of the Marker Traces in the local reference system.

On importing raw, unequally spaced data with Marker Trace information *only*, as for instance in the DZT format, IPD.markertr will comprise a single column vector of the ID (sequential) numbers of the marker traces. In order to be useful for transformation and positioning, the field must be augmented to the four column format specified above.

The co-ordinates of the marker traces can be assigned or **edited and changed** in two ways:

1) Using an in-house screen editing utility (**Basic Handling → Edit Markers → Use Editor Mode**). In this case matGPR generates the "Marker Information" window featuring an edit area, where the Marker Trace information is displayed. You may insert or change the Marker Trace co-ordinates in a simple screen-editing process, as shown in Figure 5.6 and click **OK** to accept and save them, or **Abort** to discard them.

2) Entering the information through dialog boxes (**Basic Handling → Edit Markers → Use Interactive Mode**). In this case matGPR will first inquire whether the marker traces are regularly or irregularly spaced.
   a. In the former case it will respond by asking for the location of the first marker trace and the (regular) spacing of the markers along each one of the x, y and z coordinate axes. Once it has accepted this data, it will generate the IPD.markertr field.
   b. In the latter case it will loop asking for the x, y and z coordinates of *each* marker trace, one by one.

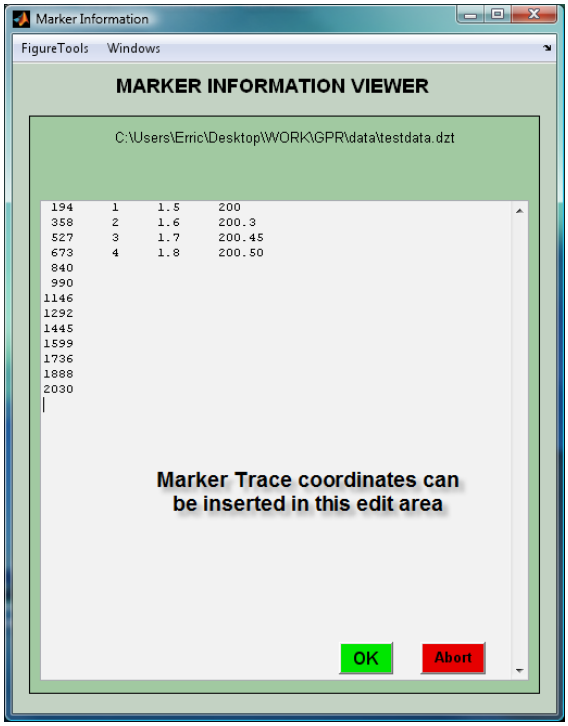


Figure 5.6. The "**Marker Information**" window generated when the user decides to process Marker Trace information in "**Editor Mode**".

Import/Replace Positioning Data. This utility allows a user to import and/or replace Marker Trace (positioning) information from a disk file. This allows one to prepare trace positioning (marker) data independently of the built-in matGPR marker editing utilities (see above). More importantly, it allows one to introduce *new* or *updated/more accurate* positioning information (e.g. obtained by DGPS) and *reposition* the traces with reference to the coordinate system of the survey using the Make XYZ utility. The format of the "Marker Trace" file is specified in the "Edit Markers" item above.

Transform to Equal Spacing. This option uses the spatial data of Marker traces (prepared by **Edit Markers** or imported from an ASCII disk file), to transform data collected at equal-time spacing mode, to data at equal-distance spacing. The transformation can be done in two ways:

1. By **interpolation** using the (shape preserving) piecewise cubic Hermite interpolating polynomial method (PCHIP, e.g. Kahaner et al, 1989). The algorithm is as follows: The locations of the (unequally) spaced traces of the input radargram are determined by interpolation, using spatial

information from the marker traces and assuming an approximately constant recording rate between successive markers. The number of traces in the transformed radargram is automatically set to $N = 2^{\lceil \log_2(M) \rceil}$, where *M* is the number of traces of the input (untransformed) radargram. The equal spacing between the traces of the transformed radargram is automatically determined to conform to the number of traces and the start and stop locations specified by the first and last markers. The locations of the traces of the transformed radargram are then determined. The output traces are then calculated by interpolation between the input traces. Owing to the choice of *N*, the transformed radargram is *always* an *exact* map of the measured radargram.

2. By **stacking** as follows: The locations of the (unequally) spaced traces of the input radargram are determined by interpolation, using spatial information from the marker traces and assuming an approximately constant recording rate between successive markers. The number of traces in the transformed radargram is automatically set to $N = 2^{\lfloor \log_2(M) \rfloor}$, where *M* is again the number of traces of the input radargram. The equal spacing between the traces of the transformed radargram is automatically determined to conform to the number of traces and the start and stop locations specified by the first and last markers. The locations of the traces in the transformed radargram are then determined and a bin size equal to this spacing is specified; the bin is centred at a new trace location. All traces of the input radargram located within a bin are stacked and averaged to create the new trace for each location of the output (transformed) radargram. Owing to the choice of *N*, the bin size is *always* larger than the approximate spacing between the traces of the input data, so as to ensure that there will always be more than one trace per bin to be stacked.

For normally measured data the two methods perform comparably: use whichever you think is more appropriate for your data. The interpolation method, however, is *guaranteed* to work in the case of *sparsely measured* input data and will never return empty traces.

N.B. 1: The new transformation routines are improved with respect to previous releases of matGPR. For instance, unlike Releases 1 and 2, the input data does not have to be measured along quasi-straight profiles with monotonically increasing or decreasing distance. The geometry of the measurements can be *anything you like*!

N.B. 2. The new transformation routines automatically assign the x-, y- and z- coordinates of the traces of the transformed radargram.

N.B. 3: Inasmuch as the x, y and z coordinates of the marker traces are *fixed* with respect to the reference frame, if the radargram was recorded in a profile running opposite to the positive direction of the coordinate axes, as for instance when the survey is conducted in a meandering forward - backward sense, the transformed section and marker trace data are flipped and streamlined with the axes of the reference frame.

N.B. 4: This function is presently ***not*** available for PULSE EKKO (DT1) data.

Make X Y Z. This utility generates x, y and z coordinates of the IPD traces with respect to a local (survey) reference frame. This is done by interpolation between the known coordinates of control (marker) traces. The coordinates of the control traces are passed as arguments (by the field IPD.markertr) or read from a disk (Marker Trace) file. The IPD must have equally-spaced traces. The output x, y and z coordinates are stored in IPD.xyz.

N.B. For PULSE EKKO (DT1) data, this function is available ONLY when the data has been collected in REFLECTION MODE.

CHAPTER 6. Advanced Processing: The "Filtering" menu

matGPR offers an extensive suite of processing utilities. These include: Mean and median spatial filtering in one and two dimensions, removal the background (mean) trace to reduce source effects, removal of a sliding window mean (background) trace to expose reflections that dip at high angles and removal of the sliding-window's "foreground" traces to eliminate high-angle reflections and enhance horizontal (e.g. hydrogeologic features). Karhunen - Loeve filtering (Karhunen, 1946; Loeve, 1955; Fukunaga, 1990), is useful in enhancing the lateral coherence of the GPR data. matGPR also provides for zero-phase FIR filtering of the frequency content (applied 'vertically' to traces), and of the wavenumber content (applied 'horizontally' to - equally spaced - scan lines). All types of filters are available (low / high / band pass, band stop and notch). All filters can be designed interactively (graphically) and tested before they're applied to the data. An interactive F-K filter design and implementation routine is also available, facilitating the application of zone pass/stop filters, fan filters and up-dip/ down-dip mapping in the frequency - wavenumber domain. Advanced filtering is available in the form *directional and multidirectional two-dimensional wavelet filters* (Tzanis, 2013) and the multi-scale/ multidirectional *curvelet transform* (Candes et al, 2006); these methods provide for S/N enhancement and *scale-and-orientation-sensitive* data manipulation. An interactive Tau-P domain modelling and filtering routine is also offered, as well as an *F-X deconvolution* (Canales, 1984; Gulunay, 1986) utility for the suppression of random noise, *predictive deconvolution* to suppress multiple reflections and *sparse deconvolution* to extract the more significant information from the data.

Mean Filter / Median Filter. These options will respectively perform mean and median spatial filtering in one and two dimensions. The filter is applied by sliding a user-specified filter window over a 2-D data wall. The data value corresponding to the central element of the window is substituted by the mean or median of the window data. Zero padding equal to 1/2 the window length is applied; therefore, the edges of the output data are expected to be distorted. For this reason, the size (dimensions) of the smoothing window should be kept reasonably small. For meaningful results, you should nor exceed 20% of the data matrix (radargram) size.

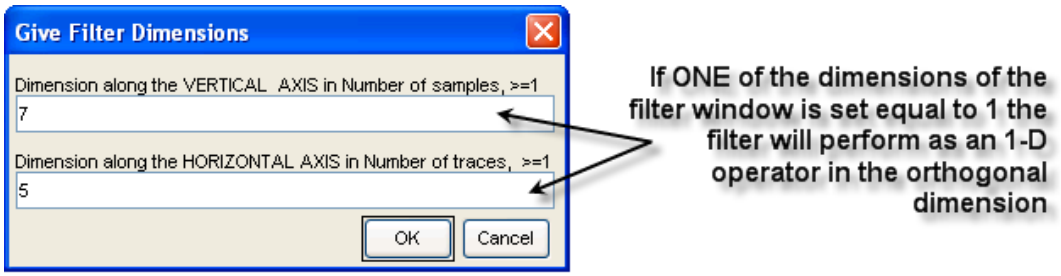
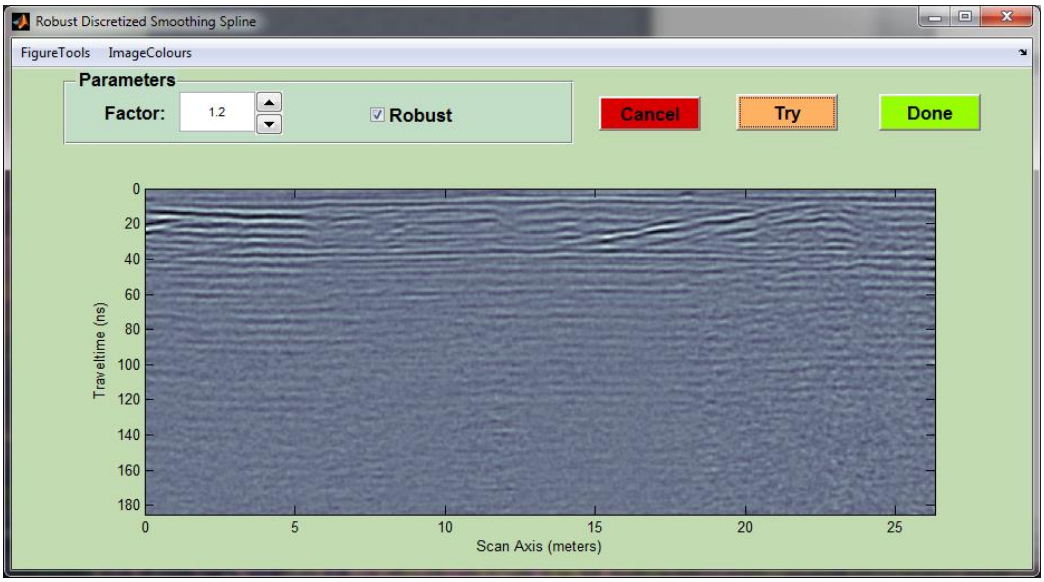


Figure 6.1. Dialog box to define the size of the mean or median filter window.

Smoothing Spline. This option invokes a *powerful* smoothing procedure based on the robust smoothing spline approach of Garcia (2010). The method can handle missing data, tolerate NaNs or Infs and only requires a real positive smoothing parameter (**'Factor'**) which controls the rigour of smoothing. The "Factor" can be very small but the larger it is the smoother the output The default "Factor" is unity; it can be changed manually by typing new values in the appropriate editable box and can be fine-tuned by clicking on the *increase* (▲) or *decrease* (▼) push buttons. For data with outliers (or localized large-

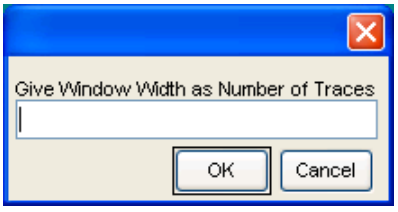
amplitude signal components) a *robust* computational procedure can be applied; this is activated/deactivated by checking/unchecking the **'Robust'** box. The default is non-robust smoothing.



The user can perform smoothing experiments by entering different "Factors" and pressing the **'Try'** button to compute and visualize the output. If, after some trials the result is satisfactory, the user can keep by pressing the **'Done'** button.

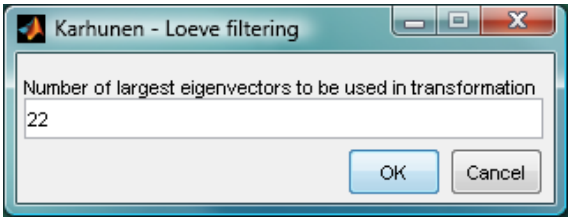
Remove Global Background. This will remove a "global" background trace from the data. The background trace is the average trace determined by adding all traces together and dividing by the number of traces. This is also called stacking. The stacking process enhances coherent signal and reduces randomly varying signal (or noise). In this case, the coherent signal is the horizontal banding often seen in GPR data (what we call system noise) and the randomly varying signal is the received radar signal from the subsurface. The appearance of the data is often improved by removing the horizontal banding. Caution must be exercised, however, with small data sets (less than about 1000 traces) or data that has strong natural horizontal reflectors. Frequency filtering may be an alternative for some data sets.

Suppress Horizontal Features. Background trace removal calculations: The mean (background) trace of a sliding window is subtracted from the data in the window. This operation will eliminate small horizontal features (coherent signal) from the data and may be used to expose reflections that dip at high angles, (removing for instance most reflections due to hydrogeologic sources). matGPR has only one request (see below): the size (width) of the sliding window. The appropriate size of the window is unique for each data set and should be decided after due experimentation.



Suppress Dipping Features. Background trace calculation and retention. As opposed to removing the sliding-window's average, trace as above, this operation removes the foreground: that is the background trace is assigned to the centre trace in the sliding window (rather than being subtracted from the data). This operation will remove high-angle reflections (a dip filter), for instance to expose the sub-horizontal hydrogeologic features more clearly. matGPR has only one request (see above): the size (width) of the sliding window. The appropriate size of the window is unique for each data set and should be decided after due experimentation.

Karhunen - Loeve Filter. The Karhunen–Loeve (KL) transform is a preferred method for approximating a set of vectors or images by a low-dimensional subspace. In matGPR the approximating subspace is defined in terms of the first (largest) *N* singular values and singular vectors (*eigenimages*) of the data. The MATLAB routine *svds* is used for the transformation. The low-dimensional approximation is re-synthesized from the *N largest eigenimages* and is a smooth version of the input data, exhibiting enhanced lateral coherence. matGPR requests the number *N* of the largest singular values:



The appropriate value of *N* is unique for each data set and should be determined by experiment. After computing the approximation, matGPR displays the reconstructed data (Figure 6.2, top), and the residuals after subtracting the reconstruction from the data (Figure 6.2, bottom). If *R* is the rank of the data matrix, the residuals comprise a low-dimensional approximation based on the *R-N smallest eigenimages*. Figure 6.2a, *N* = 22 and the reconstruction effectively a smooth version of the data without small scale variations.

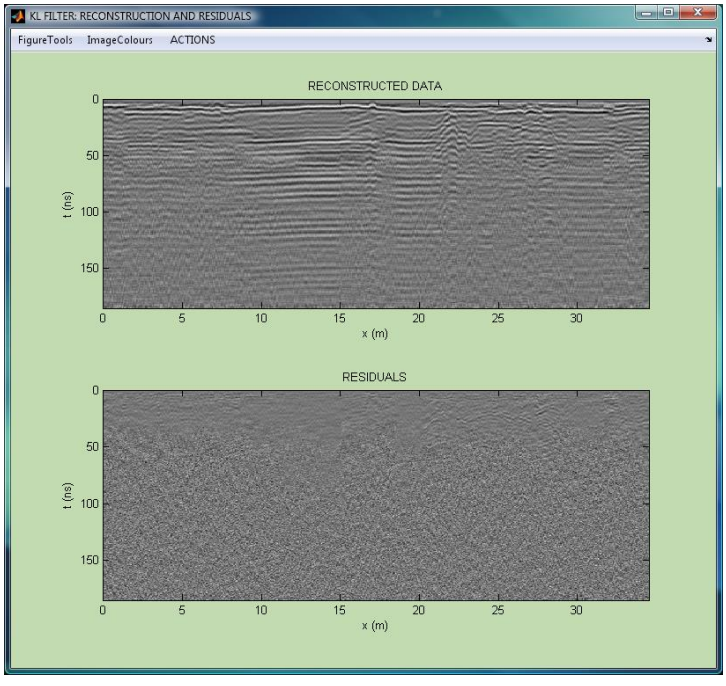


Figure 6.2a. Top: A low-dimensional subspace approximation of a data set using the first 22 eigenimages (principal components). **Bottom:** The complementary approximation (residuals) using the 462 smallest eigenimages.

In Figure 6.2b, $N = 2$ and the reconstruction comprises only the most powerful (principal) components of the input data, namely strong horizontal reflections and ringing. In the latter case, the residuals offer an opportunity to study details possibly masked by the powerful principal components. By means of the **ACTIONS** menu, (see Figure 6.2b), you may decide whether to keep the reconstruction or the residuals as the output of the K-L filtering operation.

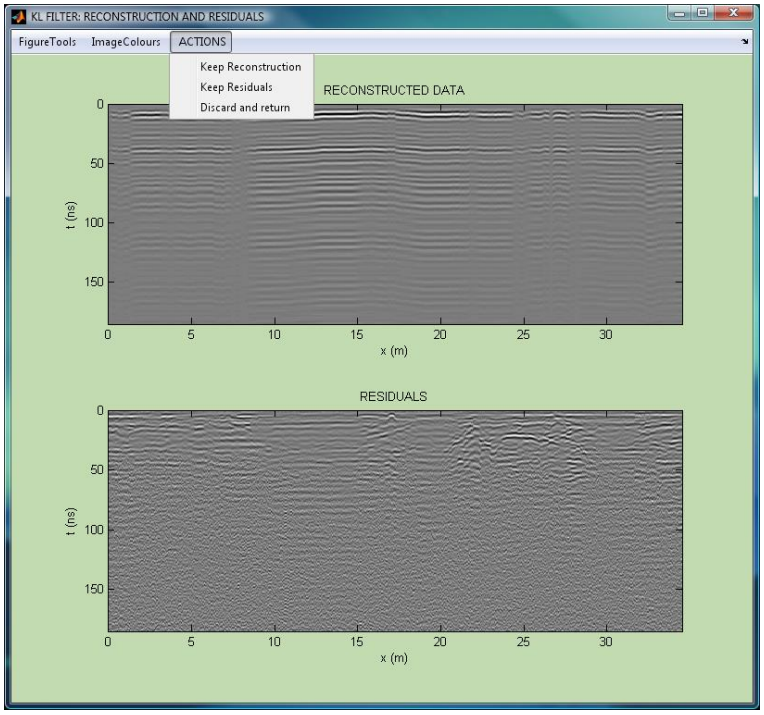


Figure 6.2b. Top: A low-dimensional subspace approximation of the same data set using the first 2 (largest) eigenimages. **Bottom:** The complementary approximation (residuals) based on the 482 smallest eigenimages.

FIR Frequency Filter. Design and apply a FIR frequency filter on the traces of the GPR data. Low / high pass, band pass and band stop filters can be designed. The simple strategy employed in matGPR is to construct a very long FIR wavelet (75% of the data length) in order to achieve very narrow transition bands, while eliminating Gibbs effects and ripple structure. The filter is applied in the frequency domain and in a forward and a backward sense, so to preserve phase information. The process is fully vectorized, so that very large data matrices can be processed in less than 1 second with modern machines.

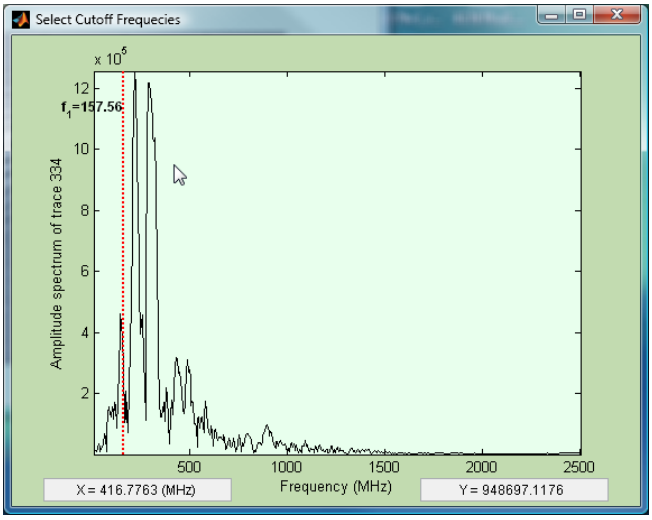


Figure 6.3. An instance from the design of a FIR band-pass filter. The lower cutoff frequency has already been determined and the upper cutoff is about to be decided and picked with the crosshair cursor

There are three ways to design the filter, which you can control the filter design mode through the **Data → Settings → Design F or K Filters** menu and one of the choices:

- **On a test trace or scanline:** The filter will be designed on the basis of the spectrum of a *particular test trace*. This is the default. When you request a filtering operation, matGPR will prompt you to pick a test trace, whose *spectrum* will be used as a *template* for determining the characteristics of the filter. You will then point the hairline cursor on the desired test trace and click the left mouse button. The spectrum of the trace will be displayed and you will again be prompted to pick the cut-off frequency(ies) using the crosshair pointer. Figure 6.3 shows an instance from the design of a band-pass filter, where the lower cut-off frequency has already been entered and the upper cut-off is about to be decided. matGPR will then compute the filter and will display the spectrum of the test trace (blue), the spectrum of the filter (red) and the spectrum of the filtered test trace (green), as per Figure 6.4. Then, it will ask you to accept or reject the design. A negative reply will discard the filter and repeat the procedure. A positive reply will apply the filter.

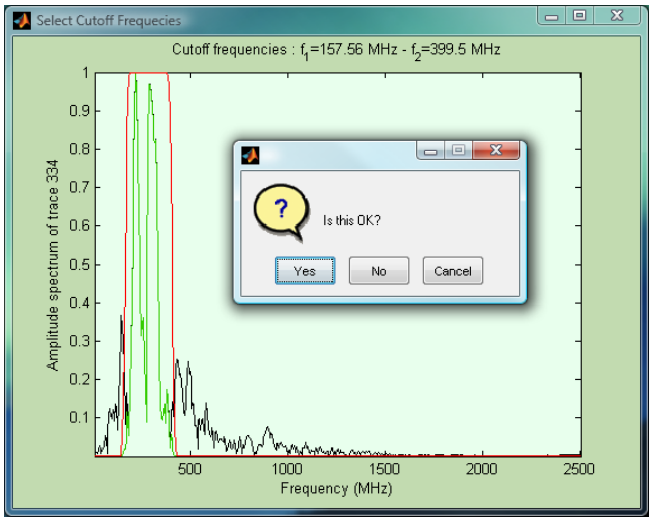


Figure 6.4. The filter is designed and applied to the test trace. The program displays the result and waits for the user to decide the next step.

- **On the mean trace or scanline:** The filter will be designed on the basis of the spectrum of the *average (mean) trace* computed from the input data, following exactly the same procedure as above.
- **Just type the cut-off frequencies:** The predefined cut-off frequencies will be given directly, through an appropriate dialog box.

FIR Wavenumber Filter. Design and apply a FIR frequency filter along the scan axis of the GPR data. A necessary condition for the application of the wavenumber filter is that traces are equally spaced. This operation is exactly the same with **FIR Frequency Filter** described above, but works line-wise instead of column-wise, thus filtering wavenumbers instead of frequencies. Identical procedures are also used in designing the wavenumber filters. Thus, selecting **Data → Settings → Design F or K Filters** menu and:

- **On a test trace or scanline** determines that the filter will be designed on the spectrum of a test particular test line,
- **On the mean trace or scanline,** determines that the filter will be designed on the spectrum of the average (mean) line,
- **Just type the cut-off frequencies** determines that cut-off frequencies will be given through an appropriate dialog box.

F - K Filter. Design and apply an F - K filter. A necessary condition for the application of the F-K filter is that traces are equally spaced – if not, the procedure aborts automatically. The available F-K filtering options include zone-pass / stop filtering, fan-filtering (velocity fan pass/ stop) and up-dipping / down-dipping event separation. The desired option is selected through sub-menus appearing as soon as the **Advanced Filtering → F-K Filter** choice is made (Figure 6.5).

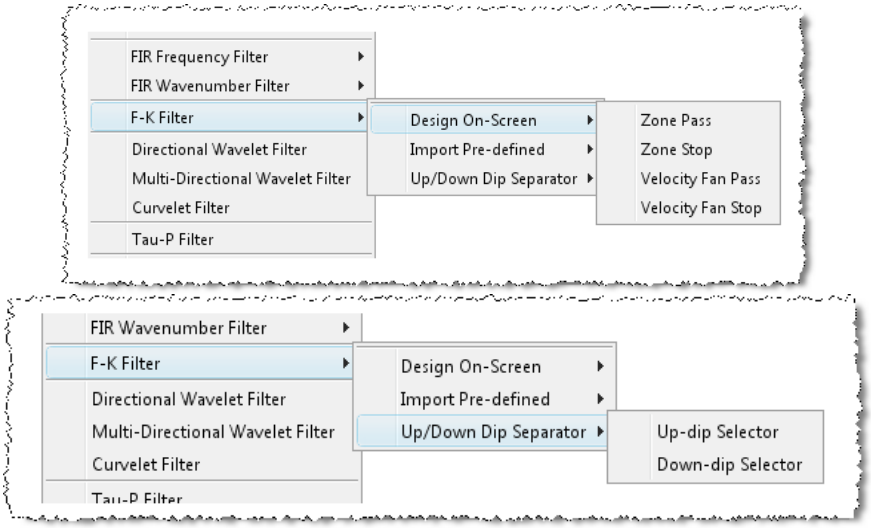


Figure 6.5. The menus of available F - K filtering operations (**left**) and methods of defining the polygonal pass / stop zone (**right**).

Zone pass/ Zone stop filters work like band pass/ band stop filters respectively. You are asked to provide the coordinates of a polygonal area (*zone*): if you select **Zone pass** the F-K spectral contributions *inside* and *on* the sides of the zone will be retained. Those outside the zone will be rejected. If you select **Zone stop** the spectral contributions *outside* the zone will be retained. Those inside and on the sides of the zone will be rejected. There are two methods of defining the pass / stop zone (Figure 6.5):

1) **Import Pre-defined coordinates** (vertices) of the polygonal zone from an ASCII file by selecting **Advanced Filtering → F-K Filter → Import Pre-defined → …**. This allows for repeated applications of the same F-K filter on different data. The structure of the file is very simple: No header; two columns of floats, the *first* column being the *wavenumber co-ordinates* and the *second* column being the *frequency co-ordinates* of the F-K spectrum. Define the polygonal area using only the positive frequency coordinates and both the positive and negative wavenumber co-ordinates.
2) **Define the polygonal zone interactively on-screen** by choosing **Advanced Filtering → F-K Filter → Design On-Screen → …**matGPR generates the F-K spectrum and displays the logarithm of its amplitude and displays a cross-hair cursor. You can set the vertices of the pass or stop zone by clicking the *left* mouse button at the desired location of the F-K spectrum image. The cursor co-ordinates are displayed at the bottom of the figure. The zone vertices are marked with a star and the enclosed area is outlined with a transparent rubberband to enhance visualization (Figure 6.6). If you set a vertex incorrectly, you can *delete it* by clicking the *middle* mouse button. There's no backward limit to this type of such undo operations other than the size of the zone. To finish, click the *right* button. matGPR will ask whether the polygon was defined to your satisfaction. If not, the process starts over. If yes, matGPR proceeds to apply the filter.

Velocity Fan Pass/ Velocity Fan Stop. This option allows for fan filtering in the F-K domain. This application only works in interactive mode by selecting **Advanced Filtering → F-K Filter → Design On-Screen → Velocity Fan…**. You need to define the [upper, lower] negative, and [lower, upper] positive apparent velocity limits in an appropriate dialog box (Figure 6.7). If you have chosen **Velocity Fan Pass**, the spectral contributions *within* the [lower, upper] apparent velocity range will be retained.

Those outside the range will be rejected. If you have chosen **Velocity Fan Stop**, the spectral contributions *outside* the [lower, upper] apparent velocity range will be retained. Those within the range will be rejected. The positive and negative velocity fans do not need to be symmetric, but in this case caution is necessary because some dipping events may be differentially amplified or attenuated. matGPR paints the velocity fan on the F-K spectrum image and awaits for your response (Figure 6.8). If satisfied, you will press **Yes** and matGPR will apply the filter. If not, the procedure starts over.

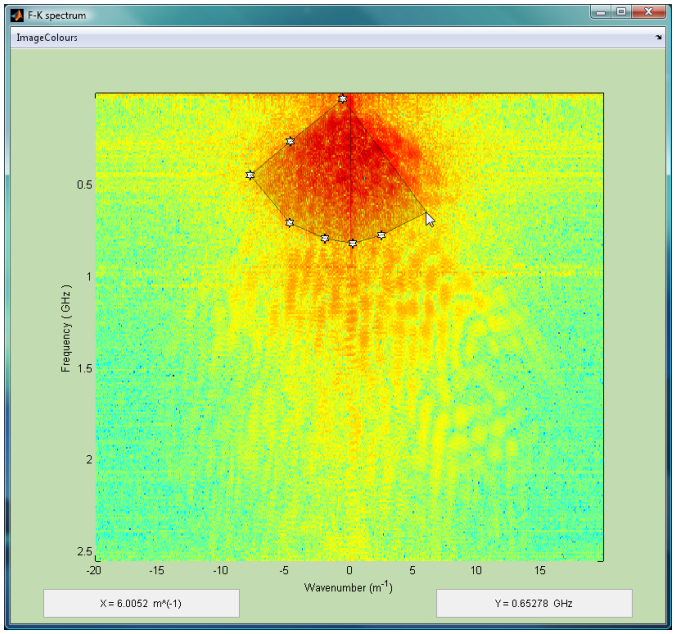


Figure 6.6. F - K filtering: Graphical introduction of a polygonal pass / reject zone.

Up-dip/ Down-dip Separator. The F-K transform has the property that reflections with positive slope (up-dipping) map into the positive-K quadrant while reflections with negative slope (down-dipping) map into the negative-K quadrant. Therefore, by specifying a filter that stops an entire quadrant while passing the other, it is possible to reject reflections dipping in one direction or the other. The result is a section with features having a unique dip direction. In this way one can separate intersecting or overlaid reflections and study tem independently.

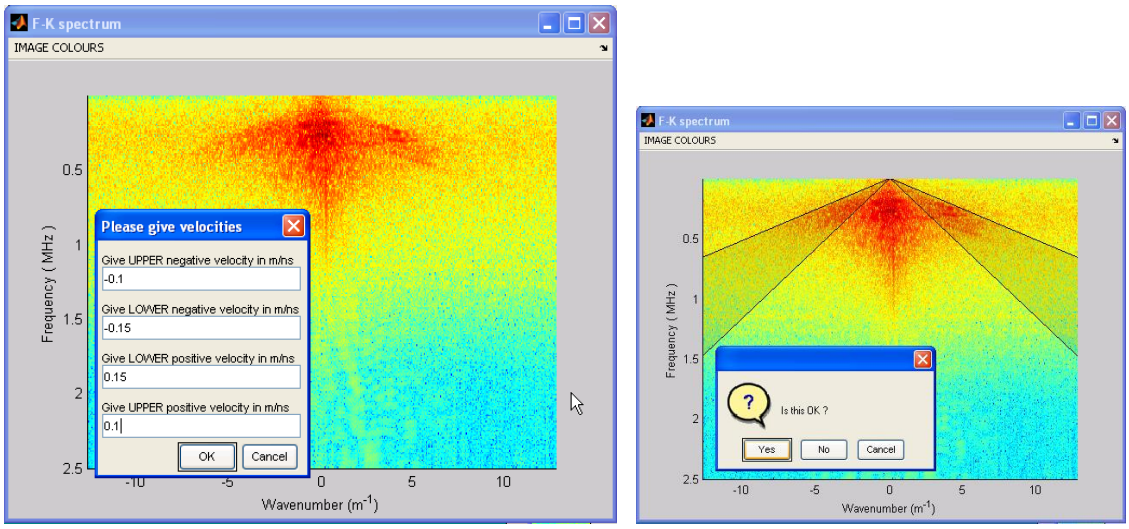


Figure 6.7. F-K fan-filtering: **Left:** Definition of the pass or reject velocity fan. In this case the fan is focused on rejecting clutter. **Right:** The velocity fan designed in Figure 6.7–left is displayed and the user is prompted to accept or reject it

Directional Wavelet Filter. Signal enhancement and information retrieval based on scale selective 1-D Wavelet Filters and *orientation-and-scale selective* 2-D Wavelet and Gabor Filters. A comprehensive presentation of the theory and *several examples* can be found in Tzanis, 2013; the same is also provided in the document "*Directional Wavelet Filters.pdf*".

- The user is strongly advised to familiarize with the concepts presented in the above literature before using the software. The rest of this section assumes that the reader is familiar both with the concepts and with the terminology used.

1-D Wavelet Filters

Wavelets effectively are linear, narrow-band filters which can be rescaled to longer or shorter lengths, providing a suite of different size convolutional filters suitable for picking out features with scales matching the filters' bandwidth. 1-D Wavelet Filters may operate *only* as frequency filters with different localization properties (scale). In order of *coarser* to *finer* localization, they may comprise Cubic B-spline Derivative wavelets, Linear B-Spline wavelets, Quadratic B-Spline wavelets Cubic B-spline wavelets and Morlet wavelets. *User-defined* wavelets are also foreseen (Figure 6.8).

The wavelet is selected via the pop-up menu labelled **Wavelet Type** (Figure 6.8) and the parameter **Filter Width (span)** *must* be unity. The **Smoothing Window** and **Orientation** properties are *irrelevant*. When the **Wavelet Type** is set to **USER DEFINED**, then, the name of the m-file implementing the wavelet should be typed in the editable box labelled **User Wavelet Function** and should always be a callable in the form wavelet = Wavelet_Function_Name(length_of_wavelet).

The spectral properties (frequency localization) are exclusively determined by the **Filter Length**, which can be set via the relevant editable box and adjusted by clicking on the associated up and down arrows (push buttons). The GUI displays the wavelet (left) and the normalized amplitude of its Fourier transform (right). The filter can be tuned to the desired peak frequency by adjusting the length; the tuning process is displayed in real time.

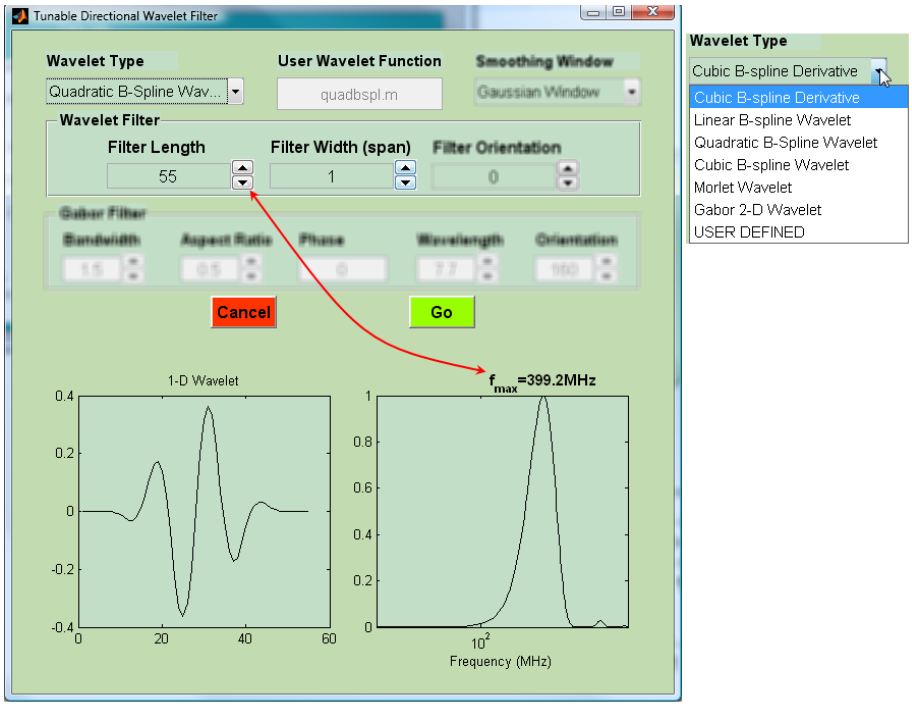


Figure 6.8. GUI to control the parameters of 1-D wavelet filters

2-D Wavelet Filters

A 2-D wavelet filter is built by sidewise arranging a number of identical 1-D wavelets (longitudinal direction), tapering the transverse direction (span) with an orthogonal window and rotating the resulting matrix to the desired orientation. The length of the wavelet (longitudinal dimension) determines the scale of the data features to be selected (isolated). The number of parallel wavelets (transverse direction/ span) determines the width over which to smooth. For data dependent on two dimensions, the application of the 2-D filter will produce an output that contains information at a specific location, of features with scales matching the filter's bandwidth. The existence of two independent variables allows each scale to be coupled with a particular orientation. Thus, scale and orientation can be varied so as to construct a matrix filter tuneable at any trait in the data.

2-D Wavelet Filters can be designed by using:

- *Longitudinal dimension* in order of *coarser* to *finer* localization: Cubic B-spline Derivative wavelets, Linear B-spline wavelets, Quadratic B-spline wavelets, Cubic B-spline wavelets and Morlet wavelets. *User-defined* wavelets are also possible.
- *Transverse dimension*: Gaussian or Hanning windows (Figure 6.9).

The properties of the 2-D Wavelet Filters are controlled by the **Wavelet Type** and the **Smoothing Window** functions, as well as the parameters **Filter Length**, **Filter Width** and **Filter Orientation** (*azimuth*) – see Figure 6.9. All the parameters can be set via the relevant editable boxes of the 'Wavelet Filter' panel and adjusted by clicking on the associated up and down arrows. The length and width are given in number of samples; the orientation is given in degrees *clockwise* with respect to the time axis. When the **Wavelet Type** is set to **USER DEFINED**, then, the name of the m-file implementing the wavelet should be typed in the editable box labelled **User Wavelet Function** and should always be a function callable in the form wavelet = Wavelet_Function_Name(length_of_wavelet).

Once the necessary definitions and parameters have been set, the GUI displays the 2-D Wavelet Filter (in Figure 6.9 left) and the normalized amplitude of its f-k transform (Figure 6.9 right). At any given orientation, the f-k spectrum consists of two lobes symmetric with respect to the origin.

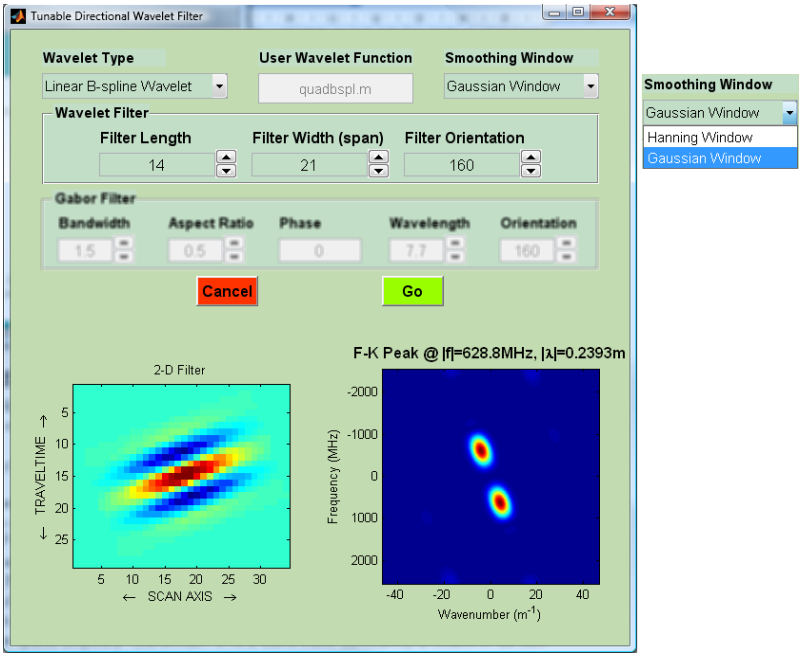


Figure 6.9. GUI to control the parameters of 2-D tuneable directional Wavelet Filters

The location of the peak of the spectral lobes is determined by the **Filter Length** (scale), which thus defines the temporal and spatial scale(s) to be isolated. The shape and roll-off rate of the spectral lobes is

determined by the type/order of the wavelet, the **Filter Width (span)** and the type of the smoothing window. Increasing (decreasing) the span contracts (expands) the lobes in the azimuthal direction and therefore influences spatio-temporal localization, but does not change the location of the peak, hence the resolvable data scale. The shape of the smoothing window influences the slope (roll-off rate) of the spectral lobes in the azimuthal direction. The frequency and wavenumber coordinates of the spectral peak are clearly displayed above the f-k spectral plot. The 2-D filter can be tuned at the desired frequency and/or wavenumber by adjusting (and micro-adjusting) the length of the wavelet and the orientation of the filter. The tuning process is displayed in real time.

2-D Gabor Filters

The 2-D Gabor filter comprises of Gaussian kernel function modulated by a sinusoidal plane wave. In the parameterization adopted in matGPR, the filter depends on five parameters: the **Wavelength** of the sinusoidal factor, the **Orientation**, i.e. *azimuth*' of the filter in degrees, the spatial **Aspect Ratio** which specifies the ellipticity of the filter and the **Bandwidth** which comprises the half-response spatial frequency of a Gabor filter (in octaves). The **Phase** offset defaults to zero. All the necessary parameters can be set in the editable boxes of the 'Gabor Filter' panel of the GUI (Figure 6.9) and micro-adjusted by clicking on the up and down push buttons; the panel is activated only when the **Gabor 2-D Wavelet** is selected in the **Wavelet Type** popup menu. Once the necessary definitions and parameters have been set, the GUI displays the 2-D Gabor Filter (Figure 6.10 bottom-left) and the normalized amplitude of its f-k transform (Figure 6.10 bottom-right).

The f-k spectrum consists of two lobes symmetric with respect to the origin. For a given bandwidth and orientation, the coordinates of the peaks of the spectral lobes are determined by the wavelength which thus defines the temporal or spatial scale to be isolated. The width of the pass-band and the roll-off rate of the spectral lobes are determined by the bandwidth while the span of the filter (size of the smoothing window) is determined by the aspect ratio. The smaller the bandwidth is, the larger the width of the pass-band. For a unity aspect ratio the spectral lobes are circular. For aspect ratios less than unity they are oblong in the azimuthal direction. For ratios greater than unity they are oblong in the radial direction (perpendicularly to the orientation). In consequence, the combination bandwidth/aspect ratio controls the dilation or contraction of the filter, hence the amount of information allowed through at the given scale.

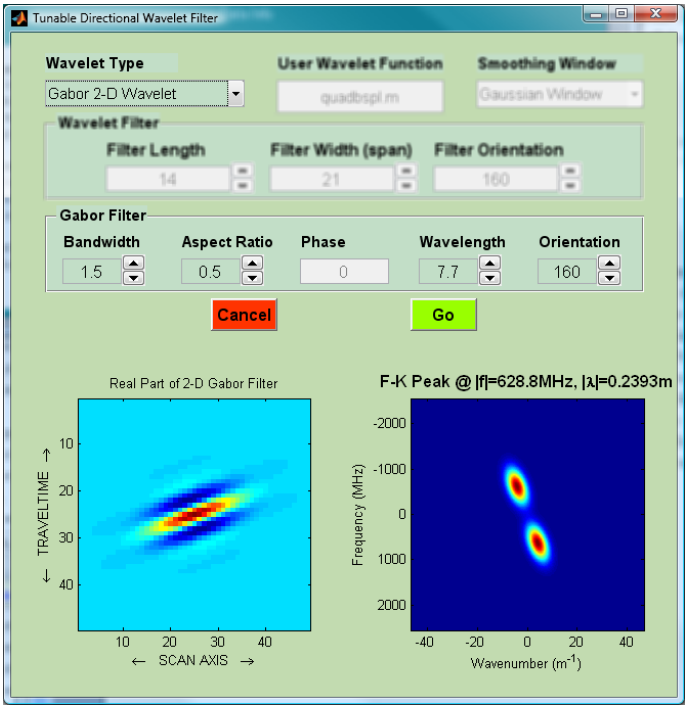


Figure 6.10. GUI to control the parameters of 2-D Gabor Filter

With respect to the 2-D Wavelet Filters discussed above, the length of the B-Spline wavelet corresponds to the wavelength of the Gabor Filter and the combination span/smoothing window to the combination bandwidth/aspect ratio. The frequency and wavenumber coordinates of the spectral peaks are clearly displayed above the f-k spectral plot. The Gabor Filter can, then, be tuned at the desired frequency and/or wavenumber by adjusting (and micro-adjusting) the wavelength and the orientation. The tuning process is displayed in real time.

Multi-Directional Wavelet Filter. Signal enhancement and wide-angle-and-scale-dependent information retrieval from 2-D GPR data with *tuneable* Directional Wavelet Filters. A comprehensive presentation of the theory and *several examples* can be found in Tzanis, 2013 and is also provided in the accompanying document "*Directional Wavelet Filters.pdf*.

The user is strongly advised to familiarize with the concepts presented in the above literature before using the software. The rest of this section assumes that the reader is familiar both with the concepts and with the terminology used.

A radargram may contain reflections from variable-dip reflectors, multiple reflectors with different dips etc. A single-dip Directional Wavelet Filter will extract only part of the available dip-dependent information because it is highly selective. This is also restrictive when one wishes to extract scale-dependent information over a range of dips (multiple- or variable-dip geometric features). The solution proposed herein borrows insight from the techniques used in edge detection and entails: **i)** Application of a Directional Wavelet Filter rotated to different angles under adaptive control so that it will remain tuned at a given frequency or wavenumber. This will yield a series $j = 1,\ldots N$ of single-dip dependent outputs. **ii)** Stacking of the single-dip dependent outputs in the weighted least-squares sense, with the stacking weight generally being a function of a measure of the energy contained in the output data normalized by the same measure of the energy contained in the input data. This weighting scheme guarantees that the final output will not be disproportionally dominated by powerful spectral components and that it will be a faithful representation of the scale-dependent information contained in the input data. Moreover, stacking will tend to smear dip-dependent noise features eluding the filter at a given temporal or spatial scale, further enhancing the S/N ratio. This filtering scheme facilitates the combination of several "partial" (same-scale-and-dip-dependent) data subsets into an image that is scale-dependent but dip-independent over a given arc $\theta_1\theta_N$. The output image will account for any variation in the angle of dip, including the case of curved interfaces. The application of multi-directional wavelet filters is facilitated with the GUI of Figure 6.11.

Wavelet Filters: the relevant parameters are selected via **Wavelet Type**, **User Defined Wavelet Smoothing Window**, **Length** and **Width** controls, as defined in the1-D Wavelet Filters and 2-D Wavelet Filters subsections of the Directional Wavelet Filter item above. The definition of filter orientations (azimuths) will be given below. Moreover:

- Filter length is computed automatically when one of the **F-Tune** or **K-Tune** radio buttons is selected in the 'Rotation Mode' panel (see below). In this case the **Length** box in the 'Wavelet Filters' panel is *deactivated* (as in Figure 6.11) and its content is irrelevant.
- Filter length has to be set manually when the **Free Rotate** radio buttons is pressed in the 'Rotation Mode' panel; this activates the **Length** box in the 'Wavelet Filters' panel.

Gabor Filters: the relevant parameters are selected via the **Wavelet Type**, **Wavelength**, **Bandwidth** and **Aspect ratio** controls, as defined in the 2-D Gabor Filters subsection of the Directional Wavelet Filter item above. The definition of the filter orientations will be given below. Moreover:

- The wavelength is computed automatically when one of the **F-Tune** or **K-Tune** radio buttons is selected in the 'Rotation Mode' panel (see below). In this case the **Wavelength** box in the 'Gabor Filter' panel is deactivated (as in Figure 6.11) and its content is irrelevant.
- The wavelength has to be set manually when the **Free Rotate** radio buttons is selected in the 'Rotation Mode' panel; this activates the **Wavelength** box in the 'Gabor Filter' panel.

Figure 6.11. GUI to control the parameters of multi-directional wavelet filtering

The **Rotation Mode** panel determines how to perform the tuning.

- If the **F-tune** radio button is selected (default), the tuning will focus on a *target (tuning) frequency* set via the **Target Freq.** box in the 'Target Frequency/Wavenumber' panel.
- If the **K-tune** radio button is selected, the tuning will focus on the *target (tuning) wavenumber* set via the **Target Waven.** box in the 'Target Frequency/Wavenumber' panel.
- If the **Free Rotate** radio button is selected, there will be *no tuning*. Depending on the choice of **Wavelet Type** this action activates the **Length** or **Wavelength** boxes and the pertinent parameters have to be set manually. Filters thus defined will be applied with fixed parameters along the range set in the 'Azimuth Range' panel.

The **Stacking Weight** drop menu determines the type of the weight. The weights implemented herein are generally functions of a measure of the energy contained in the output data normalized by the same measure of the energy contained in the input data in the sense $w(\theta_j) = \left\|\hat{\mathbf{D}}(\theta_j)\right\|_n \cdot \|\mathbf{D}\|_n^{-1}$, where **D** is the f-k transform of the data**,** $\mathbf{D}(\theta_J)$ is the output f-k transform at the angle θ_j. It is possible to set n = 1 (**L_1 norm** or Manhattan norm), n = 2 (**L_2 norm** or Euclidean norm), $n = \infty$ (**infinity norm**). It is also possible to set w=1, i.e. obtain a straight arithmetic average of the outputs at different angles. The default is n=2 (**L_2 norm**).

The **Azimuth Range** panel is used to define the range of angles over which to apply directional wavelet filters. The starting angle (θ_1) is set in the '**From Angle:**' box, the ending angle (θ_N) is set in the '**To Angle:**' box and the increment between angles is set in the '**Step size:**' box. Zero azimuth (θ = 0) corresponds to the *vertical direction*, i.e. is normal to the surface of the Earth (perpendicular to the dip direction). All angles are measured clockwise and the condition $\theta_1 < \theta_N$ must be upheld. By checking the **Visualize** box, the user may track the tuning process by means of real time graphical displays.

The **Target Frequency/Wavenumber** panel is used to define the tuning (target) frequency of wavenumber. The former is set via the **Target Freq.** box and the latter via the **Target Waven.** box. If f_0 is the target frequency, then the **F-Tolerance** box is used to define a tolerance value f_T such that any

length or wavelength that tunes the peak to within the tolerance interval [f_0 - f_T, f_0 + f_T] is deemed acceptable. Likewise, if k_0 is the target wavenumber, then the **K-Tolerance** box is used to set a tolerance value k_T such that any length or wavelength that tunes the peak o within the tolerance interval [k_0 - k_T, k_0 + k_T] is deemed acceptable.

Curvelet Filter. Signal enhancement and geometrical information retrieval with an interactive application of the 2nd Generation Curvelet Transform (CT). The CT is a *multiscale* and *multidirectional* expansion that formulates an *optimally sparse* representation of the input signal with *optimal reconstruction* characteristics. (Candès and Donoho, 2003a; 2003b; 2004; Candès et al., 2006). A comprehensive review and *several examples* can be found in Tzanis (2015) and in the document "GPR and Curvelets.pdf". Curvelets are ideally adapted to represent functions with *curve-punctuated smoothness* (or *intermittent regularity*) which are piecewise smooth with discontinuities (singularities) along a curve of bounded curvature. 2-D GPR data contain wavefronts that correspond to reflections from structural inhomogeneities; these are generally curved, relatively smooth in their longitudinal direction and oscillatory in their transverse direction. Wavefronts are functions with intermittent regularity and their singularities correspond to geological inhomogeneities at which waves reflect. Curvelets are ideally adapted to detect wavefronts at different angles and scales because aligned curvelets of a given scale, locally correlate with wavefronts of the same scale.

N.B. matGPR uses the MATLAB implementation of the Fast Discrete Curvelet Transform (FDCT), developed by E. Candés, L. Demanet and L. Ying and available as part of the "**Curvelab**" software package at http://www.curvelet.org. More precisely, matGPR uses the routines *fdct_wrapping_matlab.m* (forward FDCT), *ifdct_wrapping_matlab.m* (inverse FDCT) and *fdct_wrapping_window.m*, all by Laurent Demanet. You *need* a *license* to use Curvelab; for this reason, the matGPR_R3 distribution bundle is supplied with limited functionality and will refuse any data matrix with size larger than 2^{18} elements. Please comply by visiting Curvelab's home page and following the instructions. Once you obtain the license, download and *untar* Curvelab, then go to the directory *.../CurveLab-2.1.1/fdct_wrapping_matlab* and copy the three files to the directory *.../MATGPR_R3/analysis*. Then delete the files *fdct_wrapping_matlab.**p***, *ifdct_wrapping_matlab.**p*** and *fdct_wrapping_window.**p***. You should now be set to enjoy unlimited use matGPR's Curvelet filtering utility.

- Before using the software, the user is strongly advised to familiarize with curvelets, as for example explained in Candès et al., 2006, in Tzanis (2015) and in "GPR and Curvelets.pdf". This section will assume that the reader is familiar with the concepts and terminology used in these references.

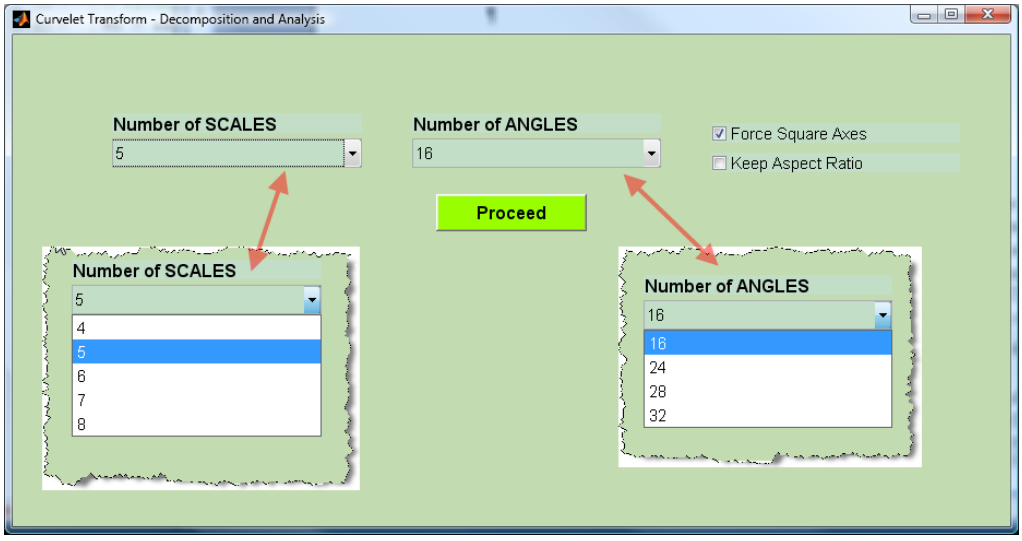


Figure 6.12a. GUI to set up the FDCT.

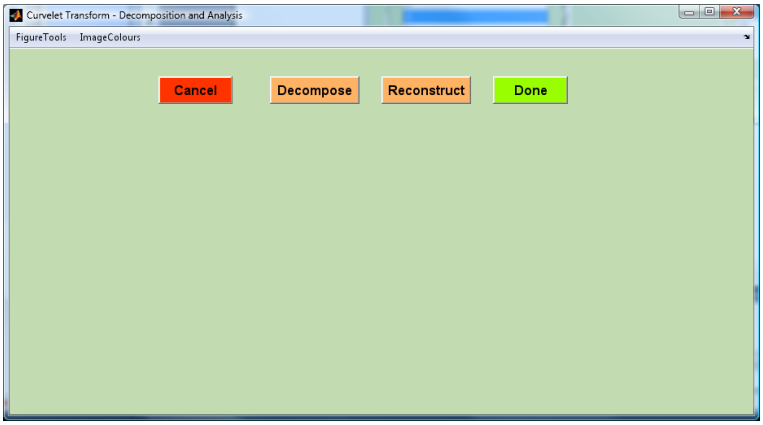


Figure 6.12b. GUI to compute the forward and inverse FDCT.

On executing a "curvelet filtering" request, matGPR inquires for *the number of scales* for which to perform the pyramidal decomposition, with $j = 1$ corresponding to the coarsest-scale inner partition of the Fourier plane. It also inquires for the *number of angles* at the second coarser scale ($j = 2$), i.e. the first scale at which an angular decomposition is computed, so as to set up the angular decomposition. The number of angles doubles in every second scale. The permissible combinations of number of scales and angles are preset and can be selected via drop menus (Figure 6.12). Because the complete curvelet decomposition of the input data is displayed and interactively manipulated, as in Figure 6.14, you may specify how you wish to have the axes of the Fourier plane drawn. By checking the **Force Square Axes** box (default), the horizontal axis (corresponding wavenumber) and the vertical axis (corresponding to frequency) will be drawn with equal lengths. By checking the **Keep Aspect Ratio** box, the lengths of the horizontal and vertical axis are adjusted according to the horizontal and vertical dimensions of the data matrix. (**N.B.** that the lengths of the axes are drawn according to the dimensions of the data matrix and not according to units of frequency and wavenumber). By left-clicking **Proceed**, the parameters are registered, the display changes into that of Figure 6.12b and the program is ready to compute the curvelet decomposition. This is done by pressing **Decompose**.

After the FDCT is computed, the upper-right diagonal of the pseudo-polar partition of the Fourier plane is displayed as shown in Figure 6.13 for a 512-by-512 data matrix decomposed into six scales and 24 angles at the second coarser scale. Each *trapezoidal wedge* drawn in Figure 6.14 represents the support of a curvelet.

The coarsest scale of the decomposition is isotropic and corresponds to very large scale features that do not admit a proper angular decomposition (i.e. description with proper set of curvelet basis functions. Instead, it comprises an isotropic low-pass window that is displayed as a rectangle at the centre of the tile. The last (finest level) scale extracts the highest frequency/ longest wavenumber content.

The design of appropriate curvelet basis functions at this scale (outermost corona) is not straightforward due to difficulties relating to issues of over- or under- sampling (Candés et al. 2006). One apparent solution is to assign wavelets instead of curvelets to the finest scale and treat it with isotropic high-pass windows, in a manner analogous to the treatment of the coarsest level scale. This is simple and effective, albeit not consistent with the idea of directional basis elements at the finest scale. Proper curvelet-based remedies exist, but because GPR data is usually oversampled, there is little, if any, useful information at the high-end of the Fourier spectrum which usually comprises noise. In consequence, the wavelet-based remedy was adopted in matGPR. In Figure 6.13, the finest level scale (in the form of wavelet coefficients) is displayed separately at the space that would have been occupied by the lower-left diagonal and illustrates the associated set of wavelet coefficients

- Each trapezoidal wedge is associated with one set of curvelet coefficients. More precisely, the graphical object associated with each trapezoidal wedge is assigned with the indices of the

corresponding curvelet coefficients. Wedges and coefficients are indexed in a clockwise sense starting at the top-left corner of the north quadrant.

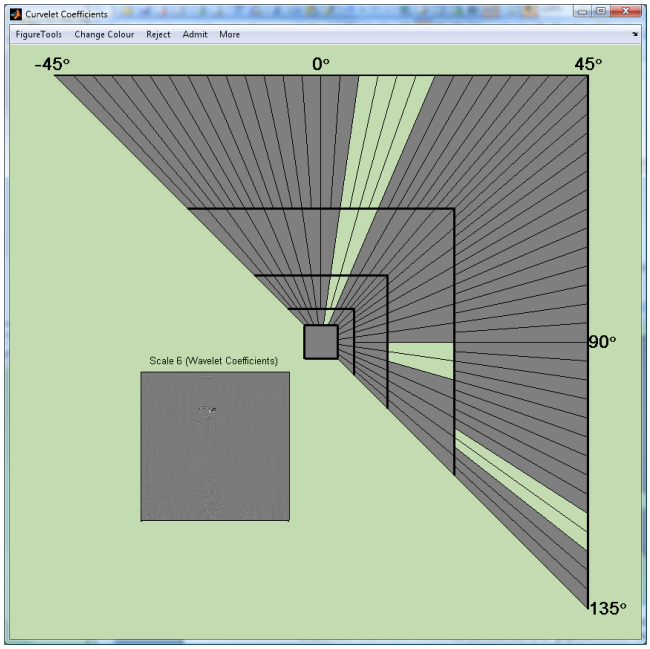


Figure 6.13. Display of the upper-right diagonal (north and east quadrants) of the pseudo-polar tiling of the Fourier plane for a 512-by-512 data matrix decomposed into six scales and 24 angles at the second coarser scale (j=2). Each trapezoidal wedge, as well as the rectangles corresponding to j=1 (central) and j=6 (lower left), is associated with a set of coefficients and functions as a "graphical switch" whose state can be toggled by pointing and clicking. The "On" state is shaded and the corresponding coefficients are included in a reconstruction. The "Off" state is blank and the corresponding coefficients are excluded from a reconstruction.

- The wedges are set up to function as *virtual switches* whose On/Off state is controlled by the screen pointing device (e.g. the mouse). All the wedges are initially displayed in their "On" state, which is indicated by shading. The "Off" state is indicated by a blank wedge. Both the coarsest and finest scale graphical objects are also set to function as virtual switches just like the trapezoidal wedges.

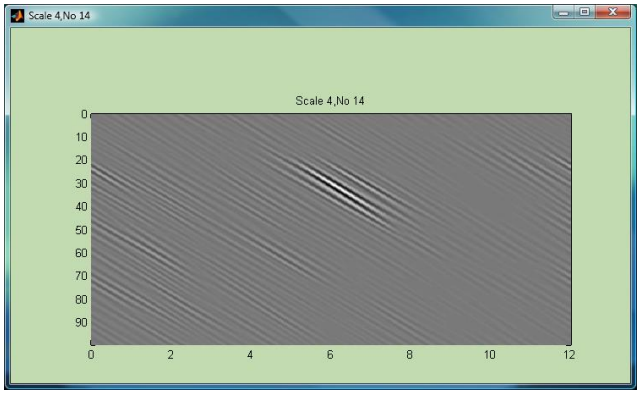
➢ It is possible to decide which curvelet coefficients to include (exclude) from a processed (reconstructed) version of the input data by pointing and clicking inside a wedge or rectangle. This will negate (reject) the associated coefficients and will toggle its state to 'Off', blanking it out (Figure 6.13). The negated coefficients may be restored (accepted) by clicking again inside a blank wedge or rectangle, in which case the state is reset to "On" (shaded).

Important Tip: Sometimes, when the decomposition is very detailed (several scales and many angles) it is difficult to click insidesmall sized wedges, especially in those located toward the centre of the Fourier plane. In this case, you can use the zooming and panning utilities. In order to zoom in, always use the sequence **FigureTools → Zoom** to activate the utility, select the zoom area and finish by **FigureTools → Zoom** to deactivate the utility. If you *do not* deactivate the utility control *does not* return to the switches! The display can be restored by selecting **FigureTools → Zoom out**. In order to pan, use the analogous sequence.

Additional useful information to guide the analysis can be obtained as follows:

- Clicking the ***middle*** mouse button inside a wedge causes the program to display the index and *central slope* of the wedge in a dedicated message box.

- Clicking the ***right*** mouse button inside a wedge causes the program to display the effect of the curvelet supported on this wedge on the data, i.e. the data filtered by this particular curvelet. The same happens you click inside the isotropic inner and outermost rectangles (coarsest and finest scale respectively). For example, if the data of Figure 3.2.1 comprise a 512-by-512 matrix decomposed into 6 scales and 24 angles at the second coarsest scale, (see Figure 6.13), by right-clicking on wedge No 15 of the fourth scale the program displays the filtered data:



After inspection of the output, the figure can be deleted by clicking *any* button while the pointer is inside the display window.

In addition to pointing and clicking on individual wedges, **it is possible to toggle the state of entire scales of angular subsets of scales** with appropriate GUI controls. These are shown in Figure 6.14 and include:

- Rejection or Admission of **the entire decomposition,** i.e. all scales and angles, using the **Reject → All** or **Admit → All** choice respectively.

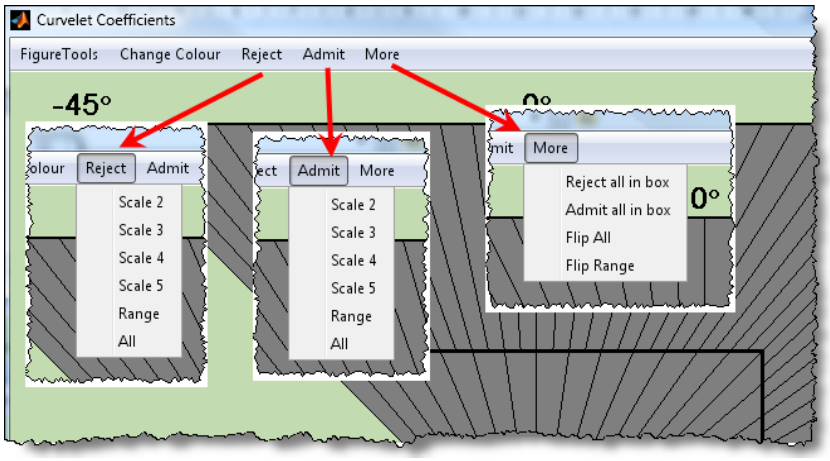
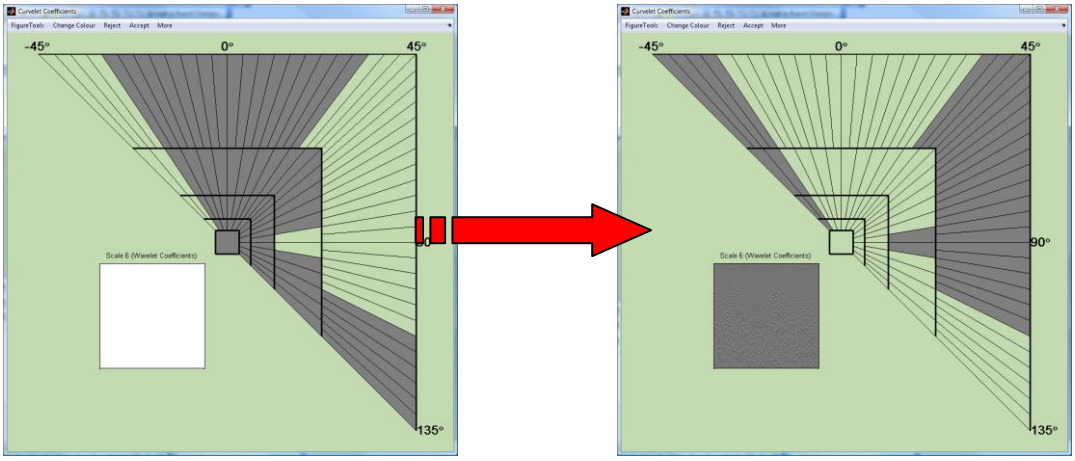


Figure 6.14. GUIs to manipulate groups of wedges.

- Rejection or Admission of **entire scales** using the **Reject → Scale #** or **Admit → Scale #** choice respectively.
- Rejection or Admission of **angular subsets of a scale** using the **Reject → Range** or **Admit → Range** choice respectively. This will produce the dialog window in which you need to specify the **SCALE**, as well as the starting angle (in the '**FROM angle:**' box) and the ending angle in the ('**TO angle:**' box) that delimit the arc you want to reject or admit.
- Rejection or Admission of curvelets inside an area defined by a *rubberband box* using the **More → Reject all in box** or **More → Admit all in box** choice.

- **Flipping the state of the entire decomposition** using the **More → Flip All** choice, as in the example below.



- **Flipping the state of an angular subset of a scale** using the **More → Flip Range** choice and a dialogue box as above to specify the 'range'.

➡ A partial or whole reconstruction of the data can be computed at any time during the editing of the decomposition by pressing **Reconstruct** in the 'Curvelet Transform – Decomposition and Analysis' window (e.g. Figure 6.15).

➡ Pressing '**Done'** in the 'Curvelet Transform – Decomposition and Analysis' window finalizes the analysis and returns control to the matGPR GUI.

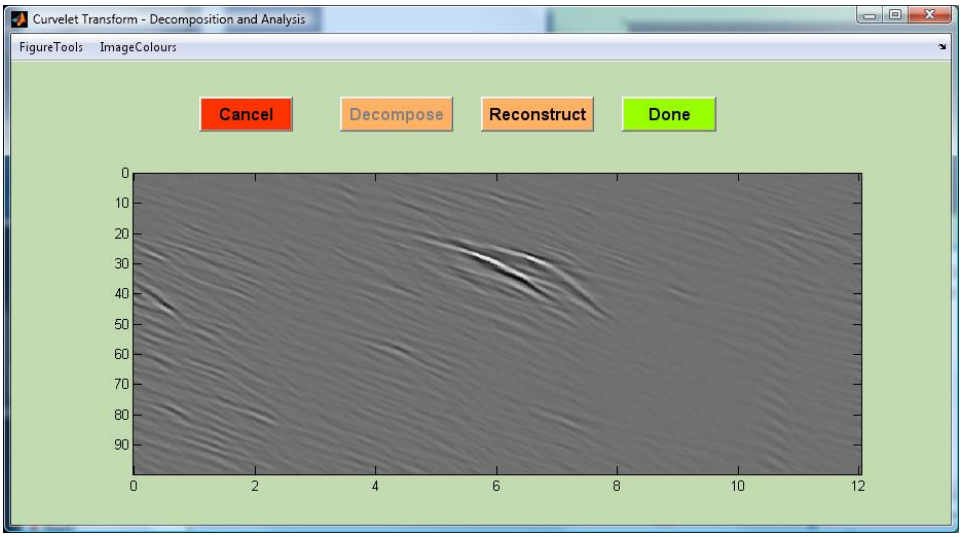


Figure 6.15. Partial reconstruction of the data of Figure 3.2.1 assuming a 512-by-512 image size decomposed into 6 scales with 24 angles at the 2nd coarsest scale. The reconstruction focuses on the down-dipping components and was based on the subset of curvelets belonging to scales 3, 4 and 5 and angles between 30° and 70°.

Tau-P Filtering. Filter data by modelling and muting noise in the τ-p domain. The GPR section data is transformed to τ-p domain by direct integration in the t-x domain. A model of the data (noise) is extracted by muting τ-p domain contributions without (within) a pass (stop) polygonal zone and inverse transforming to the t-x domain also by direct integration. The resulting model (residuals) is taken to comprise the filtered data section. A necessary condition for τ-p filtering is that traces are equally spaced.

There are only two options, *zone-pass* or *zone-stop* filtering and choice is made by means of a menu (Figure 6.16 left). If you select **Zone pass**, the τ-p domain contributions *in* and *on* the sides of the zone will be retained and outside the zone will be muted. If you select **Zone stop**, the contributions *outside* the zone will be retained and those inside and on the sides of the zone will be muted.

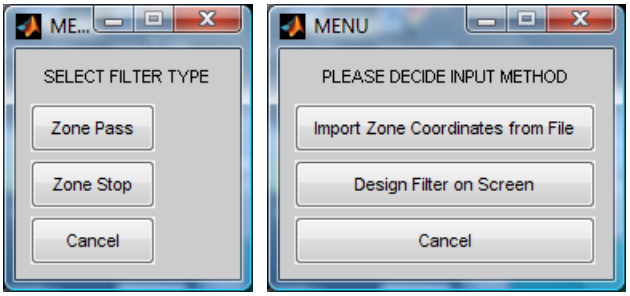


Figure 6.16. The menus of available τ-p filtering operations (**left**) and methods of defining the polygonal pass / stop zone (**right**).

As with F-K filtering, there are two methods of defining the pass / stop zone (Figure 6.16 right):

1. The coordinates (vertices) of the zone are imported from an ASCII disk file. This means that you have accurately *pre-defined* the pass or stop region and allows for repeated applications of the τ-p filter on multiple data sets. The structure of the ASCII file is very simple: No header; two columns of floats, the *first* column being the *p-coordinates* and the *second* column being the τ*--coordinates* of the τ-p domain.
2. The coordinates of the polygonal area are set interactively on-screen, exactly as with F-K filtering. matGPR generates and displays the τ-p panel and displays a cross-hair cursor. Set the vertices of the polygon defining the pass / stop zone by clicking the *left* mouse button at the desired position of the τ-p panel. To facilitate positioning, the cursor co-ordinates are displayed at the bottom of the figure. The zone vertices are marked with a star and the enclosed area is outlined with a transparent rubber band to facilitate the design process (as in Figure 6.6 and 6.18 Left). If you set a vertex incorrectly, you can undo it by clicking the *middle* mouse button. There's no backward limit to this type of undo operations, other than the size of the zone. To finish, the *right* button must be clicked *after* the last vertex has been set. matGPR will then ask whether the polygon was defined to your satisfaction. If not, the zone is erased and the process starts over.

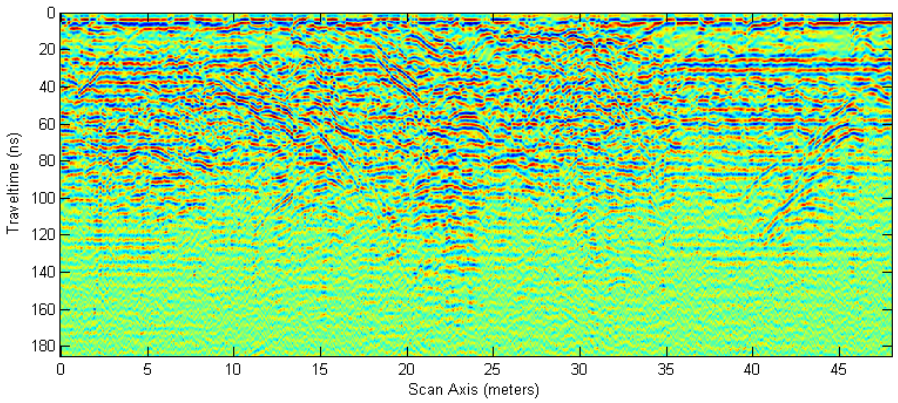


Figure 6.17. GPR section above karstified lime-stone. The data are infested with intense ringing effects.

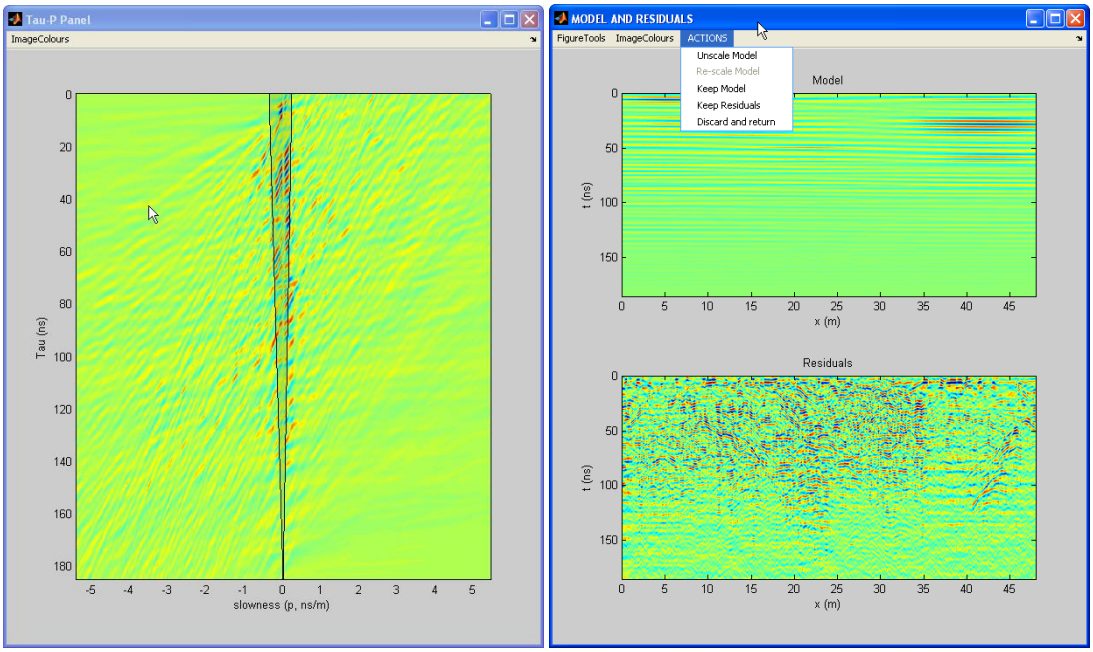


Figure 6.18. Left: The τ-p panel of the data of Figure 6.17. The shaded area represents the pass-zone containing the main contributions of the ringing effects. **Right:** A model of the ringing computed by inverse transforming the τ-p elements within the pass zone (top) and the residuals after subtracting the model from the data (bottom).

Figure 6.17 shows a GPR section measured above fragmented and carstified limestone, with intercalations of argillaceous materials. The data suffer from a considerable degree of ringing, presumably due to antenna coupling with a quite moist weathering layer. After τ-p transformation (Figure 6.18-left), the ringing is modelled in the τ-p domain by muting all contributions *outside* the shaded pass-zone shown in Figure 6.18-left. Upon inverse τ-p transformation, matGPR displays the model (in this case the ringing effects) and the residuals after scaling the model with respect to the input data (Figure 6.18-right). By means of the **ACTIONS** menu, (Figure 6.18-right), the user may experiment with scaled and unscaled versions of the model and the residuals, and, decide whether to keep the model, or the residuals as the output of the τ-p filtering process.

Figure 6.19 displays the *residuals*, which in this case were chosen to be the filtered output data (OPD). It is apparent that in this case, the output data would be virtually free of ringing.

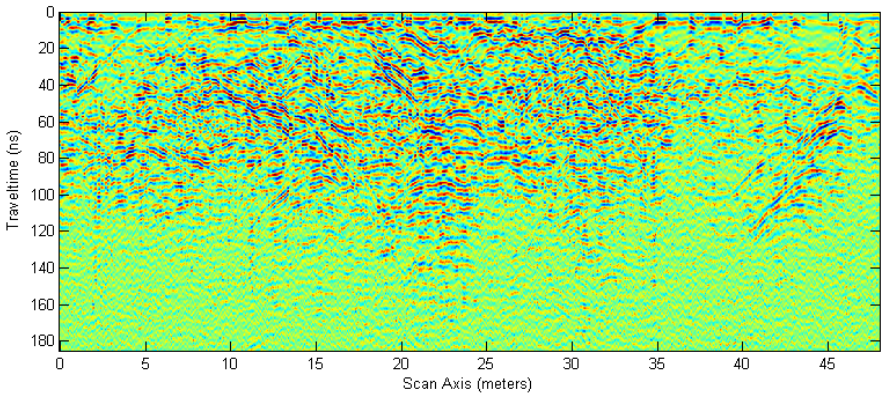
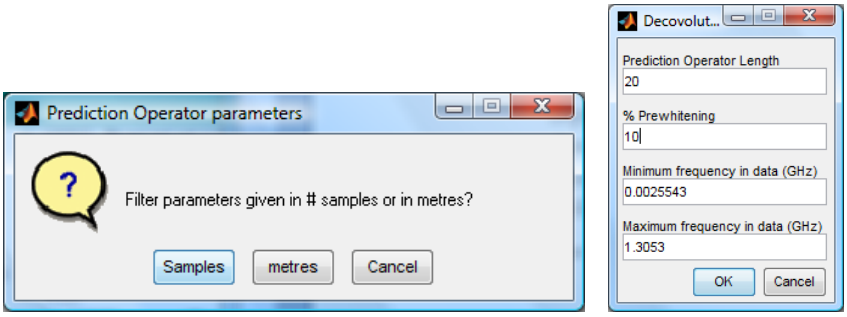


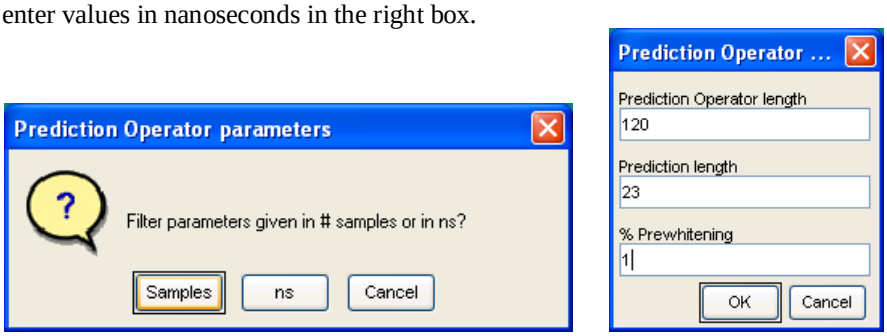
Figure 6.19: The residuals of the τ-p filtering process (Figure 6.18-right) are free of ringing

F-X Deconvolution. Noise attenuation after Canales (1984) and Gulunay (1986). For each frequency, a Wiener auto-regressive filter of unity prediction distance in space is used to predict the next sample. At the end of the process the data is mapped back to T-X domain. The deconvolution altogether needs four parameters: prediction operator length (prediction filter length), percent prewhitening and minimum and maximum frequencies to process. The length of the filter wavelet (filter length) normally can be kept relatively small and cannot exceed 1/2 of the scan axis length. The default minimum and maximum frequencies are the fundamental and Nyquist respectively, but they can be changed to correspond to the useful range of frequencies in the data set. Filter length is given either in space units (m) or in number of samples, depending on the user's preference (see below). If you choose **Samples** in the box to the left, then you should enter number of samples in the box to the right (which appears immediately afterwards). Conversely, if you choose **metres** in the box to the left, you should enter values in metres in the other box.

- Credits for the F-X deconvolution routine go to M.D. Sacchi, University of Alberta. Only minor changes have been effected by A. Tzanis, for compliance with matGPR. For more information about the F-X routine and many other seismic processing programs, please look at http://www-geo.phys.ualberta.ca/saig/SeismicLab



Predictive Deconvolution. The objective of predictive deconvolution is the suppression of multiples or reverberations. The desired output is a time advanced (lagged) version of the input signal. The deconvolution altogether needs three parameters: The prediction distance (lag), the length of the filter wavelet and percent prewhitening). The length of the filter wavelet (filter length) normally can be kept relatively small and cannot exceed 1/2 of the trace length. The lag must be smaller than filter length. To suppress multiples choose a lag corresponding to the two-way-traveltime of the multiple, to suppress reverberations choose a small lag corresponding to the duration of the primary reverberation. Filter length and lag can be given either in time units (ns) or in number of samples, depending on the user's preference (see below). If you choose **Samples** in the left box, then you should enter number of samples in the box to the right (appearing immediately afterwards). Conversely, if you choose **ns** (nanoseconds) in the left box, you should enter values in nanoseconds in the right box.



Sparse Deconvolution. Estimate a maximally parsimonious reflectivity structure of the subsurface.

- Credits for the sparse deconvolution routine go to M.D. Sacchi, University of Alberta. Minor changes were made by A. Tzanis, for compliance with matGPR. For more information about the sparse deconvolution routine and many other seismic processing programs, please look at http://www-geo.phys.ualberta.ca/saig/SeismicLab

Given a source wavelet *f*, *sparse-spike deconvolution* attempts to find an Earth reflectivity series *r* consistent with the observed data series *y* in the sense $y = f \cdot r$, but one which exhibits the *minimum possible* number of reflectors. By design, sparse deconvolution emphasizes the larger amplitudes and more important events in the observed data while suppressing small amplitude or spurious events and noise. The requirement of minimum structure (minimum number of reflectors) leads to an ill-posed least-squares problem, which is solved with regularization. matGPR implements sparse deconvolution as a method of extracting the more significant information from the data, utilizing the procedure described by Sacchi (1997). This author approaches the problem from a robust statistical/ influence function point of view, regularizing the solution with an iterative reweighted least-squares technique. Accordingly, matGPR requires the following information (Figure 6.20):

1. The **Regularization factor (μ)**. The default value is 0.1, which would normally return a moderately sparse reflectivity series. The larger μ is, the sparser the reflectivity and the higher the emphasis put on large amplitude events. The optimal value of μ cannot be known beforehand and should be determined empirically.
2. The **maximum number of iterations**. The default value is 20, which appears to more than suffice in most cases.
3. The **source wavelet** *f*. You may choose between the derivative of the Blackman – Harris window (value = 1) and the Ricker wavelet (value = 2). The default is 1 (Blackman – Harris window), which comprises a sharp pulse and returns a somewhat sharper reflectivity series than the Ricker wavelet.
4. The **central frequency** in MHz, of the antenna with which the data was measured. This is used in the computation of the source wavelet.

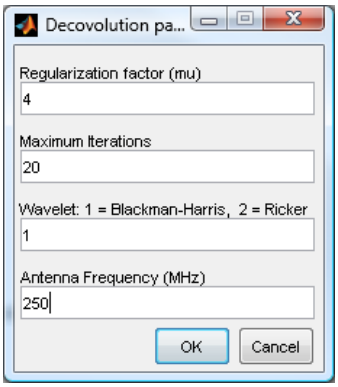


Figure 6.20. The information required for sparse-spike deconvolution.

matGPR displays the *model* $\hat{y} = f * r$ of the data, (Figure 6.21-top), in which the more significant events have been retained, and the recovered reflectivity series *r* (Figure 6.21-bottom). By means of the ACTIONS pull-down menu, you may decide whether to use the model, or the reflectivity as the output of the sparse deconvolution operation.

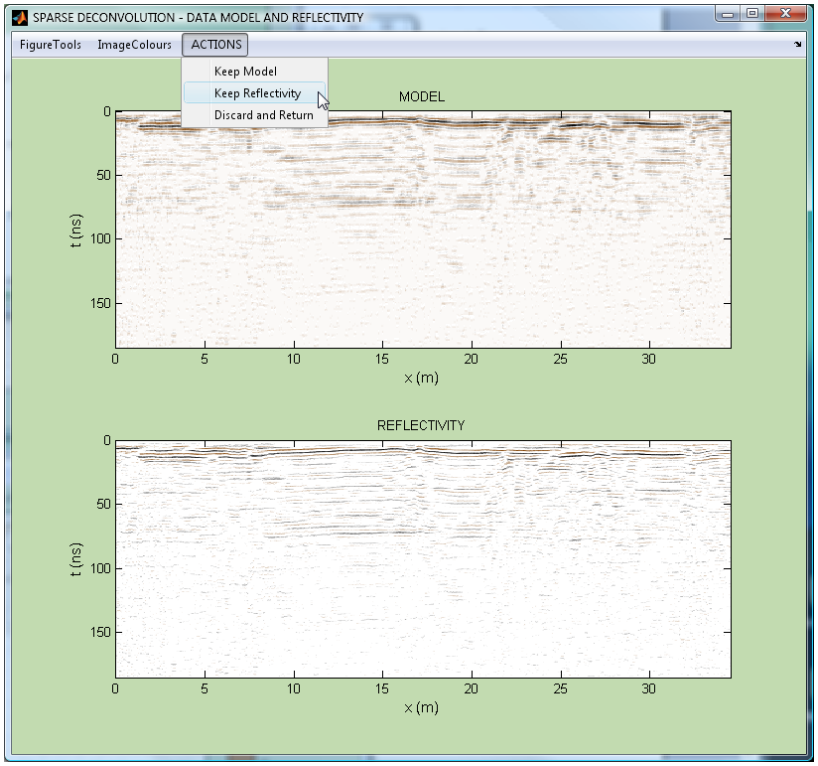


Figure 6.21. The data model (top) and reflectivity (bottom) recovered by the sparse deconvolution method.

CHAPTER 7. Interpretation I: The "Imaging" menu

matGPR offers a number of imaging and modelling tools to assist interpretation. These include velocity analysis by interactive fitting of diffraction front hyperbolae (assumes non- dispersive propagation) and static (topographic) corrections. Advanced interpretation tools include F-K migration (Stolt, 1978), Phase-shift migration (Gazdag, 1978) and time-to-depth conversion for uniform or layered velocity structures. They also include Split-step F-K migration (Stoffa et al., 1990; Sena et al., 2003) and Phase-shift plus Interpolation migration for 2-D velocity structures (Gazdag and Sguazzero, 1984).

Fit Diffraction Hyperbola. A simple approximation to the problem of fitting a diffraction front hyperbola, as it is based on the premise of non-dispersive propagation in a uniform halfspace and deals only with point diffractors, or finite sized targets of circular cross section (quasi-cylinders and quasi-spheres). matGPR generates the 'Fit Diffraction Hyperbola' figure, which displays the IPD structure in '*image display*' mode (Figure 7.1). Parameters involved in the fitting process are the *halfspace velocity*, the *radius* of the target and the coordinates of its centre (*depth* and *location* along the scan line). The parameters can be changed interactively,

1. By means of slider UI controls, as shown at the bottom of Figure 7.1. The upper and lower limits of the sliders and the minimum and maximum slider steps are determined automatically. The calculated hyperbola is plotted on the data and the quality of fitting is determined by the eye, (depends on the experience of the user).
2. By pointing the cursor at the *apex* of a diffraction hyperbola and clicking the ***middle*** mouse button to begin the fitting process, then using the scroll-wheel to automatically change velocity and target depth until the hyperbola is fitted. This is a fast, easy and accurate method, but *only* for MATLAB releases featuring the scroll-wheel callback function (WindowScrollWheelFcn).
3. Using both methods (1) and (2) above.

When finished, by clicking DONE, you cause the current estimate of the halfspace velocity to become a global variable and be accessible by other imaging and interpretation functions (it is also displayed on the information window of the matGPR GUI). The figure can be fully manipulated using the **FigureTools** and **ImageColours** menus

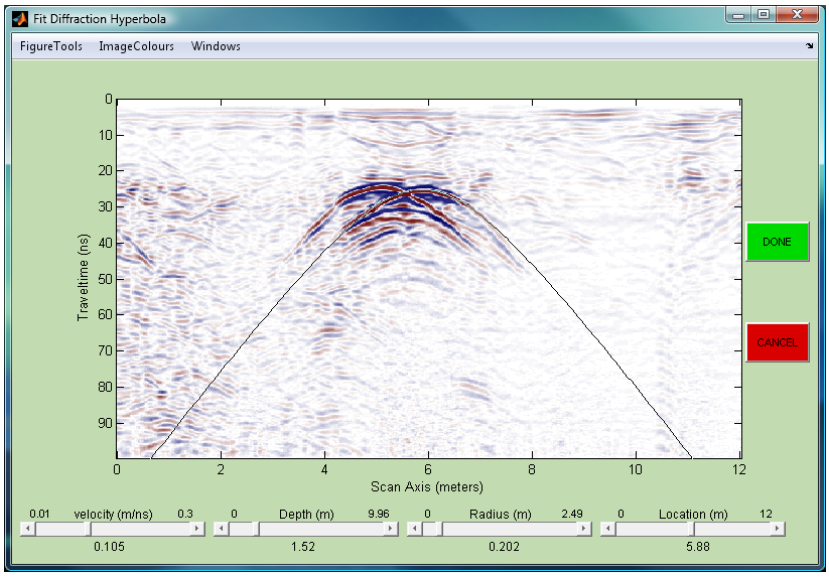


Figure 7.1. Interactive fitting of a diffraction front hyperbola. A uniform half-space is assumed and the model parameters are changed with the slider rules.

Static Correction. Apply topographic corrections to zero-offset GPR data. The routine initializes a GUI, in which you specify the necessary parameters: Velocity of the "*subweathering*" layer to be used for correction, the sense of the correction (up or down) and the elevation datum in which to refer the reduced GPR section (see Figure 7.2 for details). The elevation data necessary for the corrections are passed with the array IPD.xyz, prepared with the Make X Y Z procedure. By pressing **Go** the routine performs the topographic reduction.

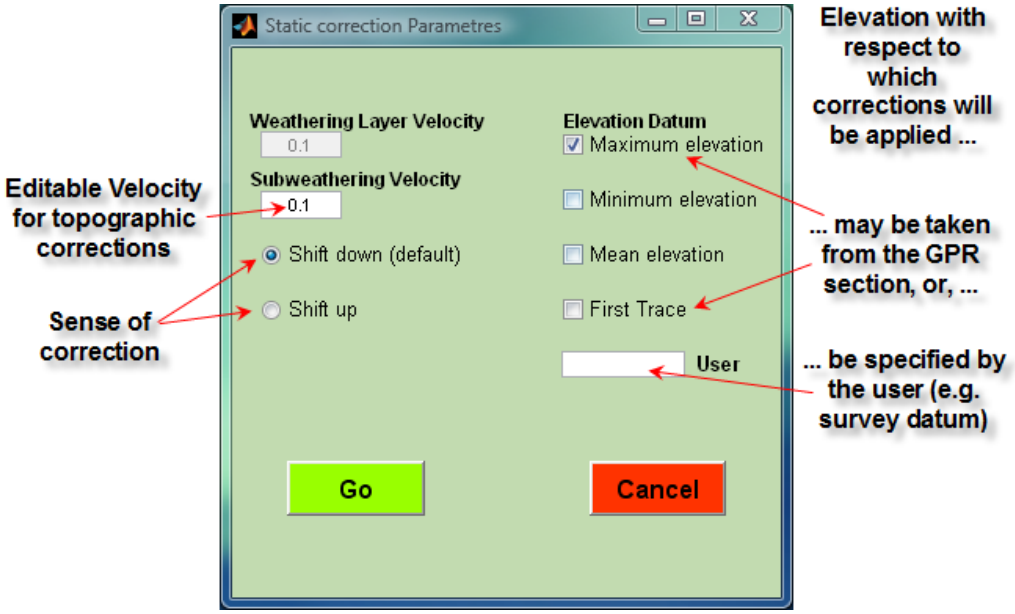


Figure 7.2. The GUI to define parameters necessary for topographic corrections.

1-D velocity model. This is a utility to import, or introduce and export 1-D (layered) velocity models for use with 1-D migration routines. At this stage development, the migration routines will only implement the real (non-dispersive) term of the phase velocity. The models can be imported from a disk file or may be given interactively in a dialog box. The format of the disk file is very simple. The first line is the number of layers NLAY, including the basal half space. This is followed by NLAY pairs of velocities (m/ns) and thicknesses (in m), one per line as in the following table:

Nlay	
Velocity of layer 1	Thickness of layer 1
Velocity of layer 2	Thickness of layer 2
...	...
Velocity of layer NLAY-1	Thickness of layer NLAY-1;
Velocity of basal halfspace	0

- The thickness of the basal halfspace is always equal to zero!
- A uniform halfspace is introduced as a 1-LAYER velocity structure with a thickness equal to zero.
- Basic error checking (e.g. for velocities exceeding the speed of light in free space) is done before the routine returns.
- The velocity structure is stored as a field of the VS structure.

-

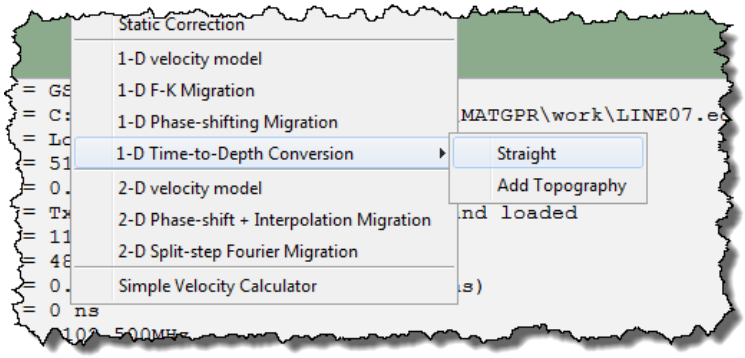
1-D F-K Migration. This is a compact implementation of Stolt's (1978) F-K migration for uniform or layered velocity structures. The program uses Stolt stretching to accommodate non-uniform (layered) velocity structures, while the actual migration code was built with tips from Jon Claerbout (1996, pp 53-57). However, this is a much more compact, robust and fully vectorized code, fast enough to outperform some compiled Fortran or C implementations of Stolt migration. The required input velocity model must have been prepared with "1-D velocity model". User intervention is not required during execution of the program.

1-D Phase-shifting migration. This is a compact implementation of Gazdag's phase-shift migration for uniform or layered velocity structures. The required input velocity model must have been prepared with "1-D velocity model". There actually two implementations of the image reconstruction code, one for layered structures and one for constant velocity structures. The former is by the author, Andreas Tzanis. The latter was tipped off by Jon Claerbout's book (1996, pp 50-53) and is quite fast and accurate, enough to warrant its separate place in matGPR. User intervention is not required during execution of the program.

N.B. 1: Inasmuch as Phase-shifting migration requires a good deal of number crunching, you might consider using the *alternative* phase-shift migration routine *gazdagmig.m* in the folder **analysis/alternative/gazdagmig+migrate.f90/**, together with the program *migrate.f90*, which only takes a FORTRAN 90 compiler to make work. In this way, you may exploit the advantage of speed afforded by compiled programs.

N.B. 2: The Fortran-90 MEX-file implementation of the migration routines is no longer supported.

Time-to-Depth Conversion. This utility migrates (remaps) a GPR section from travel time vs. distance to depth vs. distance, assuming a uniform or layered velocity structure. The required 1-D velocity model must have been prepared with "1-D velocity model".



The user may choose to make a "**Straight**" conversion, in which case the effect topography – if any – is omitted, or to "**Add Topography**", in which case the effect of topography is included. In "straight" conversions the program:

1. Computes a time vs depth function $t(z)$ from the velocity vs. depth function $v(t)$.
2. Inverts $t(z)$ to $z(t)$ by inverse linear interpolation.
3. Remaps $z(t)$ to depth z and, based on this mapping,
4. Remaps $E(t) \rightarrow E(z)$, where E stands for a GPR trace (A-scan).

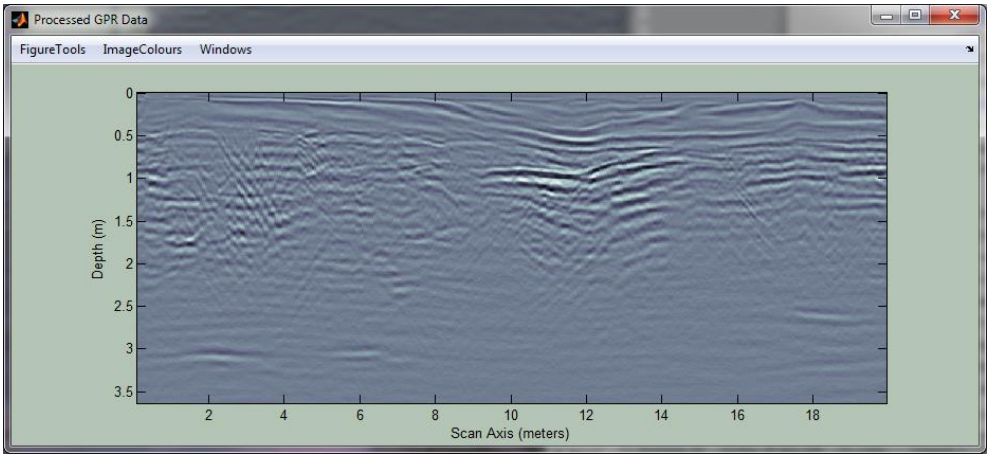


Figure 7.3. Example of "straight" time-to-depth conversion

Time-to-depth conversions *with* topography comprise two stages. The first stage is a "straight" conversion. In the second stage, the elevation (of each trace) is added to the depth vector (of each trace) and the resulting data set is resampled so as to generate a topography-adjusted radargram, which *always* has a larger number of rows than the original (unconverted) radargram. The elevation of each trace *must have been prepared* with the "Make XYZ" utility.

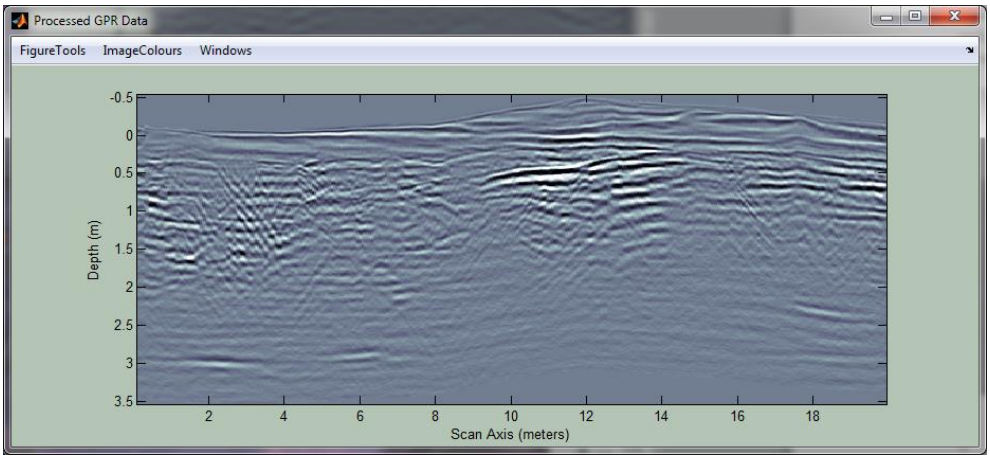


Figure 7.4. Time-to-depth conversion of Fig. 7.3 adjusted for topography ("Add Topography" option).

2-D velocity model. Import or build a 2-D velocity structure for Split-step and PSPI migrations. matGPR will ask whether you wish to import, or create the velocity model. In the former case, the velocity model will be read from a binary disk file, where it was stored during some previous session. In the latter case, the program will ask you to import (from a disk file) a synthetic structural model prepared by "Build 2-D Model" and to confirm the *antenna central frequency* and the *depth step size* to be used for integration. Then, it will generate the corresponding velocity model, optionally saving it to a binary disk file (recommended). The velocity structure is returned in a three-element cell array containing:

1) The non-dispersive term of the phase velocity model, with values expressed in m/ns. This is computed after Bano (1996).
2) The quality factor *Q* of the model.
3) The central frequency of the antenna (a scalar).

- The cell array v2d is introduced to matGPR as a field of the VS structure.

The antenna frequency and the quality factor are useful in incorporating the frequency dependence of the phase velocity (dispersion) in 2-D Split-step Fourier depth migration (and future implementations of other 2-D spectral domain migration / modelling methods), also in the sense of Bano (1996). The size of the velocity matrix and the *Q* matrix are the same as the size of the GPR section in the IPD. Along with the velocity model, the program generates and stores the array of z-coordinate axis of the velocity model and the depth stepping size to be used for migration.

The structure of the velocity model is [iz][ix], i.e. the x-direction is the fastest (MATLAB default). Such a structure is more convenient for the downward continuation type migration algorithm than using z as fastest dimension.

Overview:

In the frequency domain, the total current density flowing in the Earth is related to the EM field as

$\mathbf{J}_T(\omega) = \sigma\mathbf{E}(\omega) + i\omega\varepsilon(\omega)\mathbf{E}(\omega) = \sigma^*(\omega)\mathbf{E}(\omega) = i\omega\varepsilon^*(\omega)\mathbf{E}(\omega)$,

where the quantities

$$\sigma^*(\omega) = \sigma + i\omega\varepsilon(\omega) \qquad \text{and} \qquad \varepsilon^*(\omega) = -i\frac{\sigma^*(\omega)}{\omega},$$

respectively are the *composite conductivity* and the *composite permittivity*. They express the electric properties of the medium in terms of a unique complex quantity. Hence, we can write:

$$\varepsilon^*(\omega) = \varepsilon'(\omega) - i\frac{\sigma + \omega\varepsilon''(\omega)}{\omega} = \varepsilon'(\omega) - i\varepsilon''(\omega)\cdot$$

The total (ohmic + dielectric) losses are expressed in terms of the imaginary part ε″ of ε*. The characteristics of energy dissipation are frequently expressed in terms of the *loss tangent*, i.e. the ratio of the imaginary to the real part of ε*:

$$\tan\delta = \frac{\sigma + \omega\varepsilon''}{\omega\varepsilon'}\cdot$$

The *quality factor* is defined to be the inverse of the loss tangent:

$$Q = \frac{1}{\tan\delta}$$

In a medium where ε′(ω) and ε″(ω) have the same frequency dependence, attenuation is linearly dependent on frequency, so that the medium can be approximated by a constant *Q* factor. The reference relative permittivity *K* of such a medium, is defined to be the relative permittivity when $Q \rightarrow \infty$. The constant-*Q* approximation is valid over narrow bandwidths (e.g. Turner and Siggins, 1994). In the case of GPR, because we use narrow frequency bandwidths Δ*f* around the antenna central frequency f_c such, that $\Delta f = f_c$, (Davis and Annan, 1989), the constant *Q* approximation is still valid. Bano (1996) has shown that the phase velocity *V*(ω) are given by the product of a non-dispersive term V_0 and a power-law term *P*(ω), as follows:

$$V(\omega) = V_0 P(\omega), \qquad V_0 = \frac{1}{\sqrt{\mu K\varepsilon_0}\cos\left(\frac{\pi}{4}(1-n)\right)}, \qquad P(\omega) = \left(\frac{\omega}{\omega_c}\right)^{\frac{1-n}{2}}, \qquad n = \frac{2}{\pi}\tan^{-1}Q$$

2-D Phase-shift + Interpolation Migration. This is a fully *vectorized* MATLAB implementation of Gazdag and Sguazzerro's (1984) phase-shift plus interpolation migration for zero-offset data, with lateral velocity variation.

The velocity model used for migration must have been prepared by "2-D velocity model" and comprises the {1 x 3} cell array VS.v2d = {V_0 Q f_c}. However, the PSPI migration routine only uses the non-dispersive term V_0 = VS.v2d {1} because at present, there's no accurate method to interpolate *Q* in the sense required by the Gazdag and Sguazzero algorithm. Accordingly, this implementation of PSPI migration should rather be used when you are confident that there's no significant frequency dependence in the data. Otherwise, the 2-D Split-step Fourier Migration scheme should be preferred. At any rate, the

velocity model V_0 must have the *same number of columns* as in the GPR section. The PSPI migration algorithm used in matGPR implements large-scale vectorization and is rather fast.

2-D Split-step Fourier Migration. A *vectorized* implementation of the algorithm by Stoffa et al. (1990). In addition, the algorithm has been augmented to account for the frequency dependence of the phase velocity (dispersion), depending on the form of the variable conveying the velocity model used for migration.

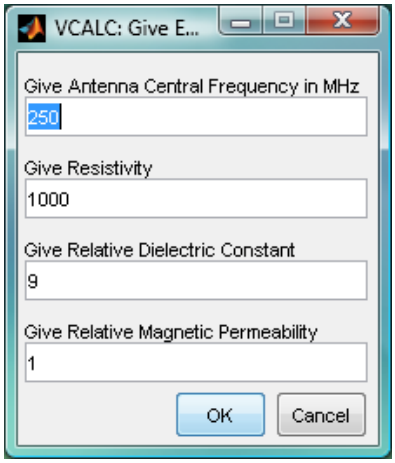
The default for matGPR is to include dispersion. The velocity model must have been prepared by "2-D velocity model" and comprises a {1 x 3} cell array v2d = {V_0 Q f_c }, with the three sub-array elements respectively being the non-dispersive term of the phase velocity, the quality factor and the antenna central frequency. V_0 and Q must have the same number of columns and rows as in the GPR section. Inclusion of the dispersion effects follows the work of Bano (1996), who has shown that

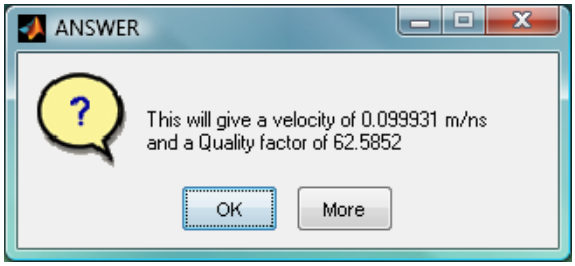
$$V(\omega) = V_0 \left(\frac{\omega}{\omega_c} \right)^{\frac{1-n}{2}}, \qquad V_0 = \frac{1}{\sqrt{\mu K \varepsilon_0} \cos\left(\frac{\pi}{4}(1-n) \right)}, \qquad n = \frac{2}{\pi} \tan^{-1} Q$$

with K being the reference relative permittivity and μ the magnetic permeability. In the augmented Spilt-step algorithm realized in matGPR, both the layer reference velocities (vertically) and the layer perturbation velocities (horizontally) are reduced for their frequency dependence.

Credits: Some programming tips were taken from a split-step migration routine by G.F. Margrave, CREWES Project, U. of Calgary, 1996.

Simple Velocity Calculator. Utility to calculate EM field velocities in finite media. The function enquires the nominal frequency of the GPR antenna, the resistivity, relative dielectric constant and relative permeability of the medium (below left) and calculates the non-dispersive term of the phase velocity, with values expressed in m/ns, and the quality factor, both after Bano (1996). For additional details see item "2D velocity model".





The VS construct: Storage of velocity information. matGPR stores 1-D or 2-D velocity data in a data structure with fields:

Field	Variable
1	VS.v1d
2	VS.v2d
3	VS.zv2d
4	VS.dzmig

1. The field "VS.v1d" contains a one-dimensional velocity distribution model to be used with one-dimensional time and depth migration functions. It is an NLAYER × 2 array, in which the first column comprises layer velocities and the second column comprises layer thicknesses. The thickness of the basal halfspace is always equal to zero! A *uniform halfspace* is introduced as a 1-LAYER velocity structure with thickness equal to zero. For more details refer to the items Fit Diffraction Hyperbola and "1-D Velocity Model".

2. The field "VS.v2d" contains a two dimensional velocity distribution model to be used with one-dimensional time and depth migration functions. The field "v2d" is a three-element cell array containing, a) the non-dispersive term of the phase velocity model, with values expressed in m/ns. This is computed after Bano (1996); b) the quality factor *Q* of the model, and c) The central frequency of the antenna (a scalar). For more information refer to items "2-D Velocity Model" and "Build 2-D Model".

3. The field "VS.zv2d" contains the z-coordinate axis of the 2-D velocity model conveyed by "VS.v2d".

4. The field "VS.dzmig" contains the depth step to be used for 2-D depth migrations (see items 2-D Split-step Fourier Migration and 2-D Phase-shift + Interpolation Migration).

CHAPTER 8. Interpretation II: The "Modelling" menu

Build 2-D Model. matGPR offers a GUI utility (the *Model Builder*) to construct 2-D models for processing with the Split-step and Phase-shift plus Interpolation migration methods, as well as for forward modelling with Bitri and Grandjean's (1998) Split-step and Irving and Knight's (2006) time-domain finite difference algorithms. At this stage of development, the model comprises an ensemble of objects with polygonal or circular cross-sections. The objects can be overlapping and the order in which they are introduced in the model determines the way in which they will appear in the velocity model (or whether they will appear at all). The program employs a front to back hierarchy: foreground objects will appear whole in the model; background objects will not appear at all if they are totally obscured or will appear partially if they are partially obscured (their visible portion). Therefore, the objects added later to the model take precedence over the objects introduced earlier! The (Model) Builder also provides for the graphical editing of object shape and properties (inserting, relocating or removing vertices and changing EM properties, with unlimited levels of undo). The following few paragraphs will provide a basic description of the utility.

Figure 8.1 shows the window and menus of the Model Builder. The **Figuretools** and **Windows** menus have already been described in Section 2.1.3. Particular to the Builder are the **Data** and **Actions** menus. The **Data** menu helps you initialize the program and import or export models. Its function and options are straightforward and there will be no time wasted for detailed descriptions. The **Actions** menu provides access to the main collection of tools by which you can build and edit a model and will be given due consideration.

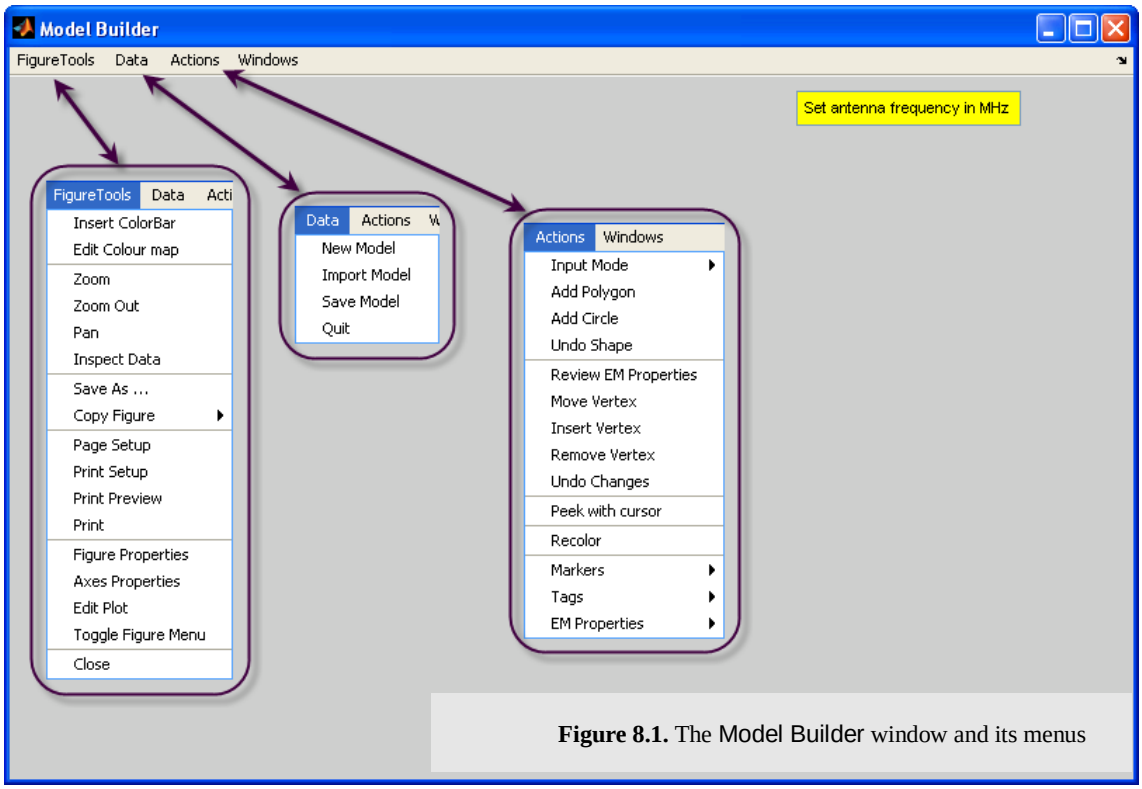
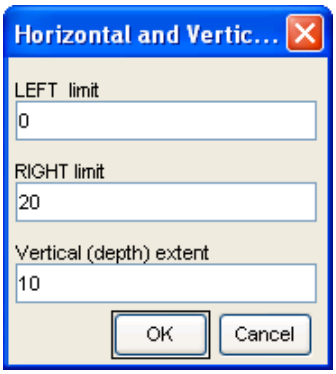


Figure 8.1. The Model Builder window and its menus

1. Creating a new model

On selecting **Data → New Model**, the Builder displays a dialog box, in which you will define the size and extent of the model. Required parameters are the left and right limits in the horizontal direction and the vertical (depth) extent of the model. The model is always assumed to begin at the surface (z=0). In order to create a velocity model for 2-D migrations, i.e. one that will be processed by 2-D velocity model, make *absolutely sure* that the horizontal extent (*x*-axis) of the model *matches* the horizontal extent (scan axis) of the GPR section.



Once the size of the model is defined, the Builder automatically creates and paints a background structure with dimensions equal to the size of the model and enquires for the antenna central frequency, the background resistivity, relative dielectric constant and relative magnetic permeability (Figure 8.2 left). Given these figures, it calculates the phase velocity for the central frequency and asks for your approval (Figure 8.2 right). If you press **No**, the Builder asks for new electric and magnetic properties and so on... **Cancel** discards the current background and the procedure starts over. If you press **Yes** the background is accepted as the first object in the list. The builder is now ready to accept new structural objects (bodies). As mentioned previously, the model comprises an ensemble of bodies with polygonal or circular cross-sections.

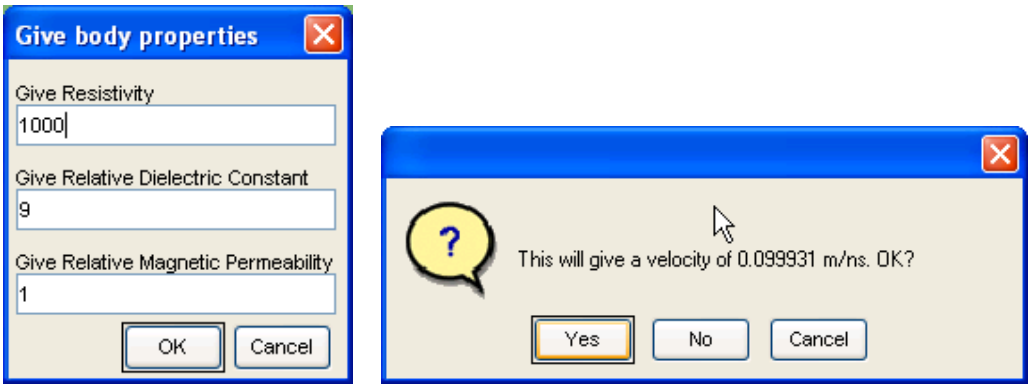
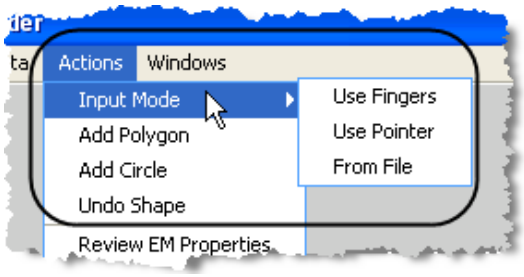
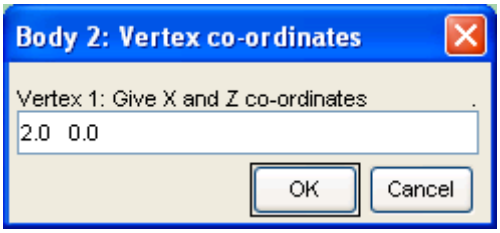


Figure 8.2. Left: The dialog box enquiring the EM properties of the new object. **Right:** The Builder always asks for confirmation of the body's properties

1.1. Inserting polygonal objects (bodies): There are three ways to enter a polygonal object, selectable from the **Actions → Input Mode → …** menu:



Use Fingers to insert polygonal object via dialogue boxes: The Builder will first ask for the number of vertices in the body. Then, it will loop through the number of vertices, asking for the vertex coordinates, one at a time. You should enter (x, z) pairs of coordinates as per Figure 8.6. When the coordinates have been entered, the Builder will paint the body and will enquire its properties, also by means of dialog boxes as per Figures 8.2 and 8.4. Accepting the body will cause the Builder to ask for an ID tag by which it will identify the body (Figure 8.5). The tag is then painted on the centre of gravity of the polygonal body.



Use Pointer to insert polygonal graphically: The Builder displays a crosshair cursor; set the vertices of the polygonal body by clicking the *left* mouse button on the desired coordinates. To facilitate positioning, the cursor coordinates are displayed at the bottom of the figure.

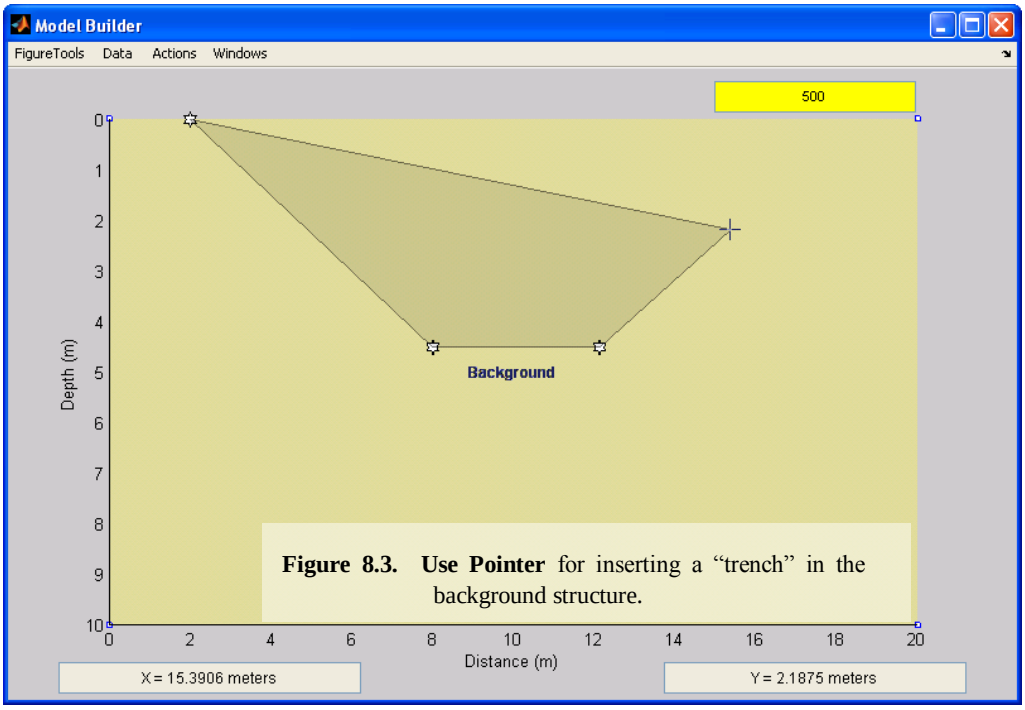


Figure 8.3. Use Pointer for inserting a "trench" in the background structure.

The vertices are marked with a star and the area they enclose is outlined with a transparent rubber band to enhance visualization and facilitate the design process (Figure 8.3). This is particularly helpful when inserting complex shapes. If you set a vertex incorrectly, you can undo it by clicking the *middle* mouse button, whereupon the vertex, the line to the vertex and the rubber band to the vertex are erased. There's no backward limit to this type of undo operations, other than the size of the zone. To finish, click the *right* mouse button *after* the last vertex has been set. The Builder will now paint the body and will ask its properties, as per Figure 8.4. Accepting the body (by pressing **Yes** in the quest dialog) will cause the Builder to ask for an ID tag by which it will identify the body (as per Figure 8.5). The tag is then painted on the centre of gravity of the polygonal shape.

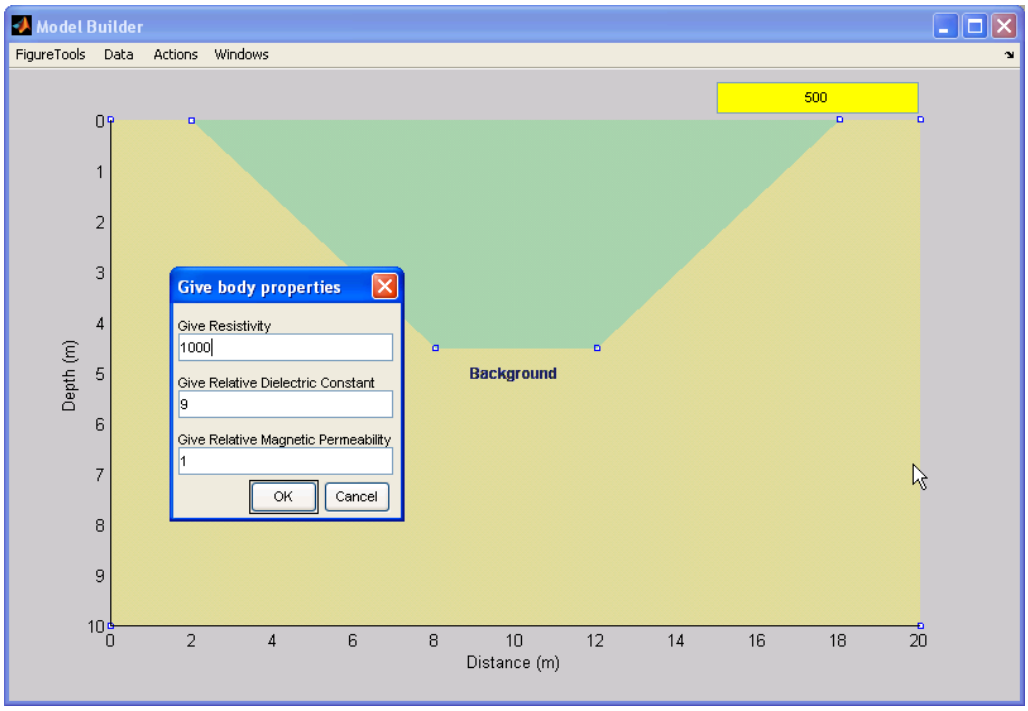


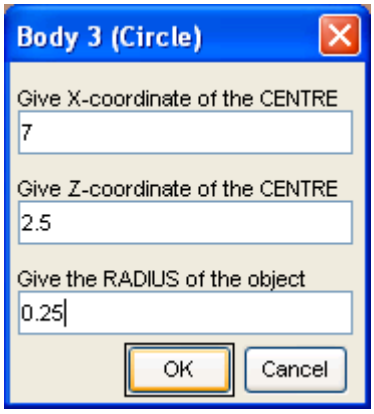
Figure 8.4. The trench is painted on the model and the properties of the filling material are now enquired.

Figure 8.5. The Trench is now given a name (Tag).

From File to read the coordinates (vertices) from an ASCII file. This file has very simple structure: No header; two columns of floats, the *first* column being the *horizontal* (*x*) *coordinate* and the *second* column being the *vertical* (*z*) *coordinate.* After importing the polygon, the Builder paints the body and asks for its properties with the procedure described above.

1.2. Inserting circular objects (bodies): There are two ways to enter a circular object, selectable from the **Actions → Input Mode → …** menu:

Use Fingers: The Builder will ask you to type the coordinates of the centre of the circular body and the length of its radius in a dialog box as shown below. In this here example, a water pipe of diameter 0.25m is specified and put in the trench of Figure 8.4. The result is shown in Figure 8.6.



Use Pointer: The Builder displays a crosshair cursor. You are required to point and left-click, first at the coordinates of the *centre* and then at a distance equal to the *radius* from the centre. To facilitate positioning, the cursor coordinates are displayed at the bottom of the figure and an animated line connects the centre with the current position of the cursor. Note that this input method is not recommended when precise placement of small objects is required.

When the location and size of the circular object have been set, the Builder calculates a polygonal approximation to the circumference, paints it on the model enquires its properties, also by means of dialog boxes as per Figures 8.2 and 8.4. Accepting the body and defining its ID tag will complete the process. Figure 8.6 shows a model comprising a "trench" dug in the background, with a water pipe located at (7, 2.5) metres, and a metal pipe at (13, 2) metres, both parallel to the strike of the trench.

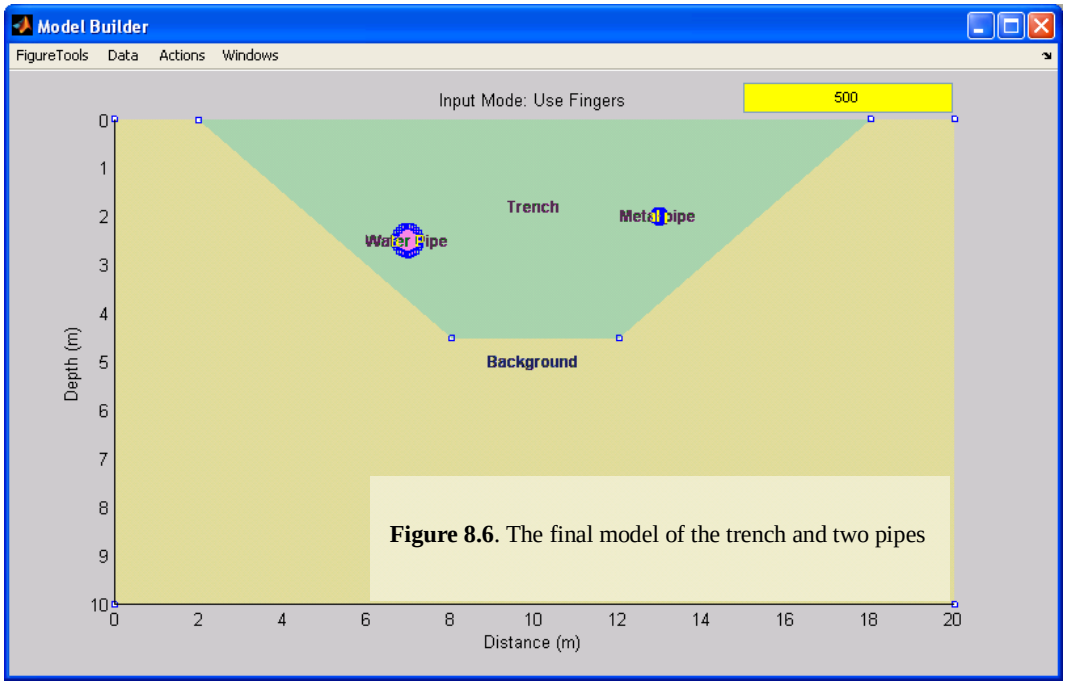


Figure 8.6. The final model of the trench and two pipes

1.3. Deleting an object: A body can be removed from the model through the **Actions → Undo Shape → …** choice. The Builder will then display a drop menu above the top left corner of the model axes, in which you can identify a body by its tag and remove it.

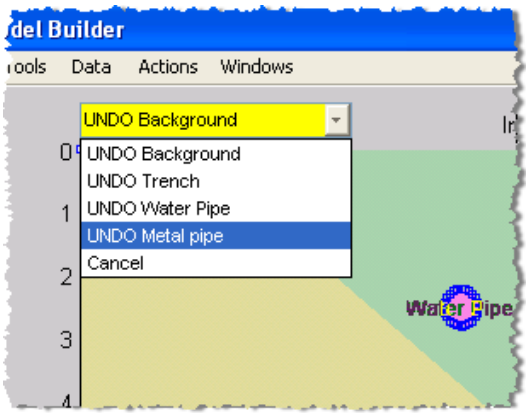


Figure 8.7. Remove (Undo) a body by selecting it in the drop menu.

2. Editing a model.

The Model Builder allows extensive editing and modification of the model. This includes the geometry and the properties of the constituent bodies.

2.1 Changing EM properties. The EM properties of any body can be changed through the **Actions → Review EM Properties → …** menu item. The Builder will then display a drop menu above the top left corner of the model axes, in which you can identify a body by its tag and select it. The Builder projects the current body properties in a dialog box, as per Figure 8.2.left, and asks for the new properties, calculates the phase velocity for the central frequency and asks for your approval (as in Figure 8.2 right). If you press **Yes** the new properties are accepted for the body. If you press **No**, the Builder asks for new electric and magnetic properties and so on... **Cancel** discards the procedure and returns the program to the idle state.

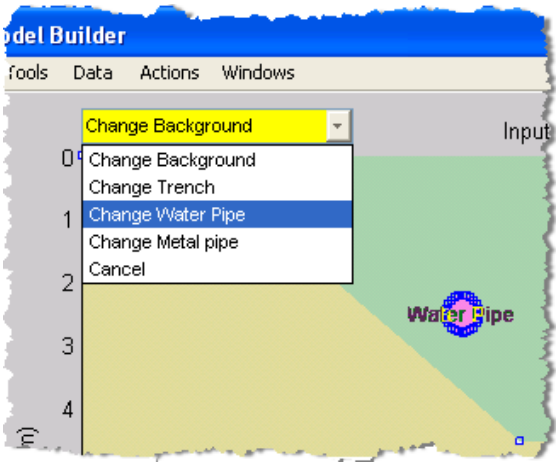


Figure 8.8. Select a body in the drop menu to change its EM properties.

2.2. Changing model geometry. The geometry of the constituent bodies can be changed through the menu choices **Actions → Move Vertex / Remove Vertex / Insert Vertex**.

Relocate a vertex: First select the vertex by pointing on the vertex and clicking any button. If you're in **Use Fingers** or **From File** input mode, the Builder will ask for its new location by displaying the following dialog box

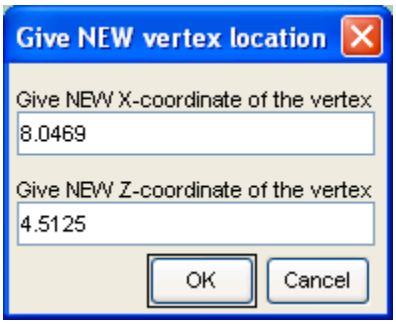


Figure 8.9

If you're in **Use Pointer** input mode you must relocate the vertex by pointing and clicking on its new location. The cursor coordinates displayed at the bottom of the model figure assist you to point as accurately as possible, however, this relocating method is not recommended when precise positioning is required.

Remove a vertex simply by pointing on it and clicking any mouse button.

Insert a new vertex: First you must *define*, by pointing and clicking, the *two* vertices *bracketing* the new vertex, (equivalently the side of the polygon to be split by the new vertex). The builder will remind you this by issuing a help message. Then, if you're in **Use Fingers** or **From File** input mode, the Builder will display the dialog box of Figure 8.9 asking for the location of the new vertex. If you're in **Use Pointer** input mode, you should insert the new vertex by pointing and clicking on its desired location. The cursor coordinates displayed at the bottom of the model figure assist you to point as accurately as possible, but this method of inserting vertices is not recommended when precise positioning is required.

2.3. Undoing changes to the model geometry. Use the **Actions → Undo Changes** menu choice to reverse any modifications you have made in the geometry of the bodies. The number of undo operations is not limited but is also not indefinite: the backup register is *reset* every time there's a major change in the model, i.e. when a new object (body) is inserted or removed. You must finish undoing geometry changes *before* such an action, or they will become permanent. It is a good idea to consider doing geometry changes *after* laying out the basic structural elements of the model.

2.4. Recolouring the model. The Builder assigns colours to bodies *randomly*. Sometimes the resulting coloration is difficult to read, or even aesthetically offensive. Use the menu choice **Actions → Recolor** to change the scheme to something more acceptable.

2.5. Utilities. Some utilities useful in editing and illustrating the model are provided with menu choices **Actions → Marks / Tags / EM Properties**.

- You can show or hide the markers demarcating the vertices of the bodies.
- You can show / hide the tags or toggle them with the corresponding EM properties
- You can show / hide the EM properties or toggle them with the corresponding tags

For example, Figure 8.10 illustrates the trench and pipes model of Figure 8.6 without markers, and Figure 8.11 the same model without markers but with the EM properties shown in the place of tags.

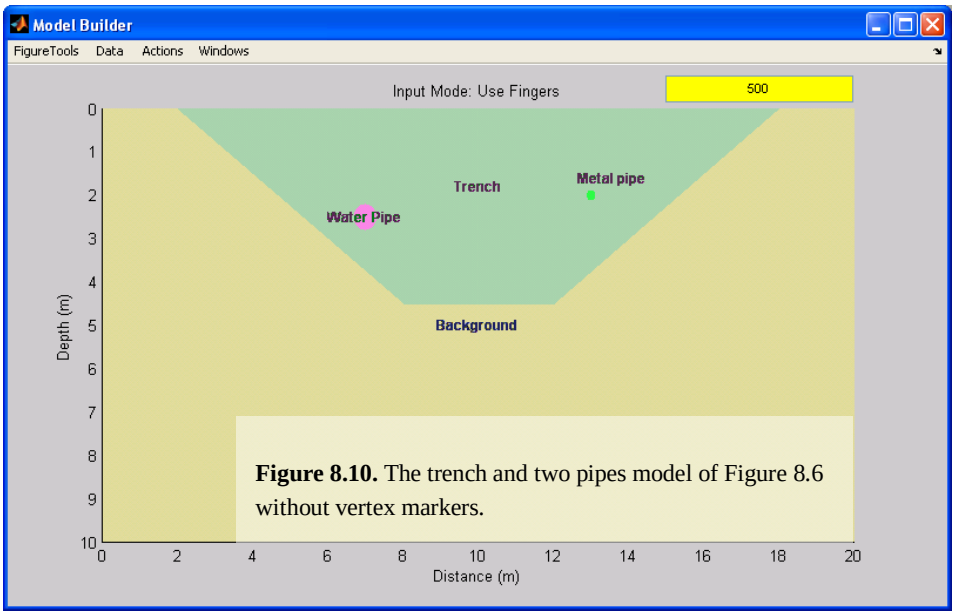


Figure 8.10. The trench and two pipes model of Figure 8.6 without vertex markers.

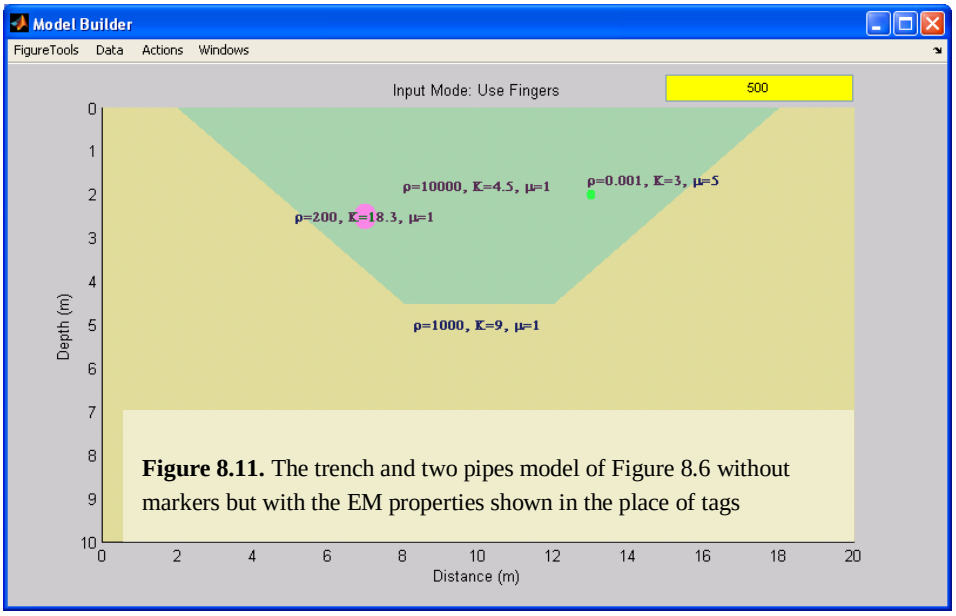


Figure 8.11. The trench and two pipes model of Figure 8.6 without markers but with the EM properties shown in the place of tags

3. The model file.

The Builder saves the model for later access by itself, *get2dvelocity.m* and *splitstep2dmodel.m*, as a free format ASCII file (the *model* file), with the following structure:

Line	**Model Parameter**	**Size and description**
1	antenna	float, antenna frequency.
2	Xmin Xmax Zmin Zmax	float, horizontal and vertical extent of the model.
3	nbody	int, number of bodies in the model.
4	tag	string, Tag of 1st body.
5	nv, K, Q, mu, rho	int + 4 x float. Respectively are: a) The # vertices in 1st body, b) its relative dielectric constant, c) its quality factor, d) its relative magnetic permeability, e) its resistivity
6	x(1) z(1)	float, float: coordinates of the first vertex
7	x(2) z(2)	float, float: coordinates of the second vertex
…	... ...	...
nv	x(nv) z(nv)	float, float: coordinates of the last vertex
nv+1	Tag	string, Tag of the 2nd body.
nv+2	nv, K, Q, mu, rho	int + 4 x float: Respectively # of vertices, relative dielectric constant, quality factor, relative permeability and resistivity of the 2nd body
nv+3	x(1) z(1)	float, float: coordinates of the first vertex of the 2nd body
…	.... and so on	

For example, the "background" and the "trench" of the model in Figure 8.7 would be written as:

```
500.000                                  antenna
0.000 20.000 0.000 10.000                    Xmin Xmax Zmin Zmax
2                                nbody
Background                               tag
4 9.000 125.170476 1.000 1000.000            nv, K, Q, mu, rho
0.000  0.000                             x(1)       z(1)
20.000  0.000                            x(2)       z(2)
20.000  10.000                           x(3)       z(3)
0.000  10.000                            x(4)       z(4)
Trench                                   tag
4 4.500000 625.852380 1.000 10000.000        nv, K, Q, mu, rho
2.015625  0.012500                           x(1)       z(1)
8.046875  4.512500                           x(2)       z(2)
12.015625  4.512500                          x(3)       z(3)
18.015625  0.000                         x(4)       z(4)
```

Split-step 2-D Modelling. Generate synthetic GPR sections with the method of Bitri and Grandjean (1998). This program imports (from disk file) a synthetic structural model prepared as detailed above and generates matrices of the relative dielectric constant, magnetic permeability and quality factor structures. These are passed to a routine that computes the corresponding velocity structure after Bano (1996), and hence the forward model.

The program will first ask you to specify a model file. It will import the model and based on the range of velocities it will calculate from the EM properties of its elements, it will compute and suggest a low (Small), Medium and high (Large) resolution grid for the synthetic radargram (Figure 8.13.left). It will also offer you the choice of customizing the synthetic radargram for particular requirements (e.g. simulation of observed data).

Experience shows that the high resolution synthetic is not necessarily the best choice, as it always comes at the expense of computing time. It does, however, exhibit the least amount of artefacts and aliasing. The low resolution synthetic is computed quickly and is often acceptable, but depending on the model, it occasionally exhibits artefacts and aliasing. It is quite alright for the fast assessment of your data and/or interpretation. The medium resolution synthetic is often a good compromise. However, it should be noted that the dimension of the traveltime axis often tends to be overestimated and may be safely reduced (through customization), saving a lot of storage space and computing time. At any rate, you shouldn't rely exclusively on the program's suggestions and it usually pays to make a few experiments to determine the optimal parameters before setting out to work.

If you choose to customize the grid, the program responds with a dialog box as per Figure 8.12.right. The program always suggests the use of the Medium Resolution dimensions but this is only the dummy default. You should specify the desired dimension of the scan axis (nx), depth axis (nz) and time axis (nt), as well as the sampling rate (dt). For data simulation exercises, when dt, nx and nt are fixed, it may take a few experiments to determine the optimal dimension of the depth axis (nz), i.e. the optimal depth stepping size to be used in the calculations.

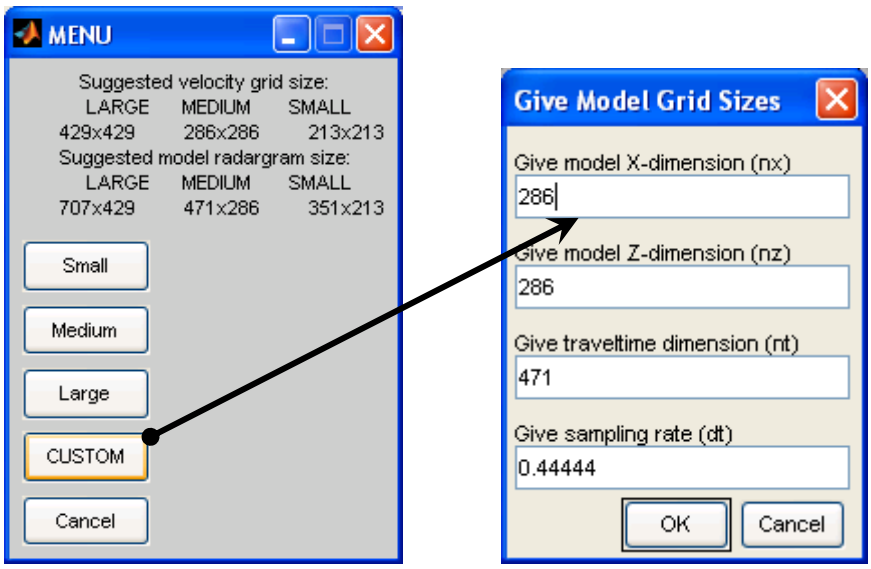
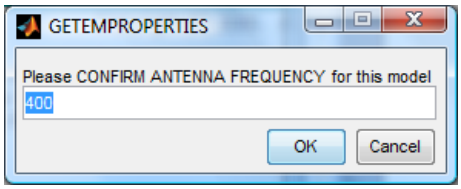


Figure 8.12. The dialog boxes produced by the matGPR Split-step forward modelling routine

The computations are carried out with fully vectorized code that executes rather fast, almost comparably to FORTRAN serial code. Accordingly, the support of FORTRAN MEX-files has been discontinued as of Release 3 of matGPR. However, if you still want to use Fortran, , e.g. for very large jobs, you may onsider using the *alternative* modelling routine *splitstep2dmodel.m* in the folder **analysis/alternative/ Splitstep_model+radar2d4.f90/**, together with the program *radar2d4.f90,* which only needs a FORTRAN 90 compiler to make it work.

FDTD 2-D Modelling. Generate synthetic GPR sections with the TM-mode, Finite Difference, time-domain method of Irving and Knight (2006). This program imports (from disk file) a synthetic structural model prepared by *build2dmodel.m* and requests that you confirm or change the central frequency of the "probing antenna".



Then, it computes minimum non-dispersive velocity in model (Bano, 1996), and uses it to construct an initial discretization of the model with the approximations

$$dz = \frac{v_{\min}}{8F_c}, \qquad dx = 2dz, \qquad dt = \frac{1}{8F_c}$$

where F_c is the central frequency, $v_{\min}$ is the minimum velocity, dz is the depth spacing, dx is the horizontal spacing and dt is the sampling rate.

- Use of $v_{\min}$ ensures adequate discretization of the high velocity areas of the model, where the wave is propagating fast and higher resolution is required in order to model it.

Following discretization, the model properties are displayed as images of the relative dielectric constant, conductivity, relative magnetic permeability and velocity grids (Figure 8.13).

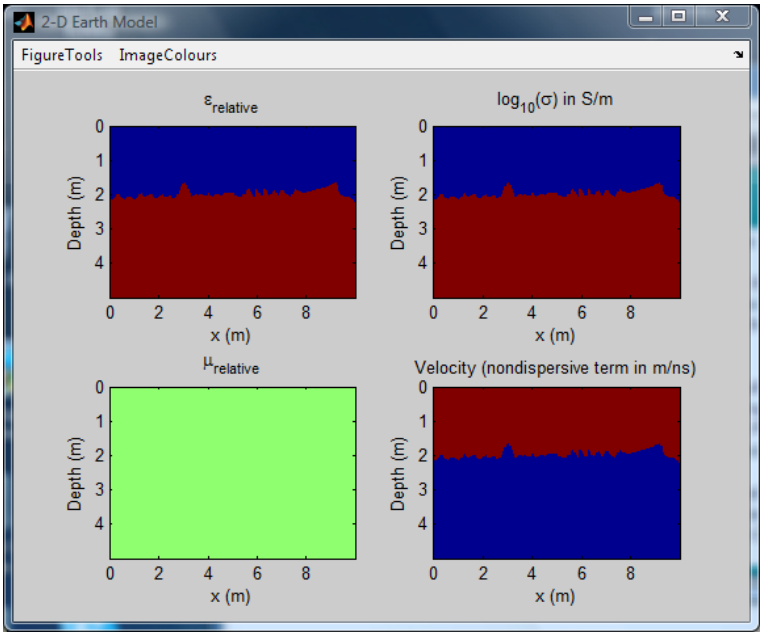


Figure 8.13.

Based on the initial discretization, matGPR computes maximum permissible values for dx and dz and requests that you specify these parameters. By default, it suggests values close to the maximum permissible. It also requests the initial location of the source and trace (source) spacing along the scan axis (Figure 8.14-left).

- The term *"maximum permissible"* refers to the values below which, aliasing effects should not appear. The program will work with grid and time spacing > maximum but aliases are to be expected!

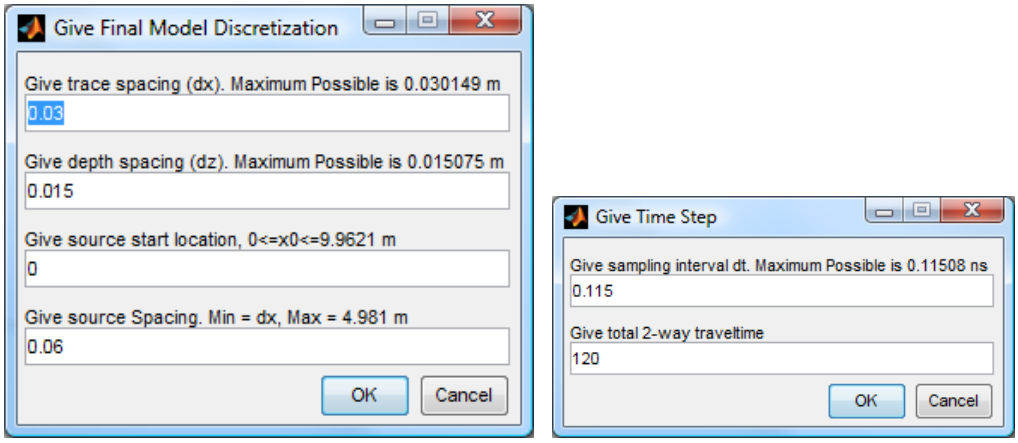
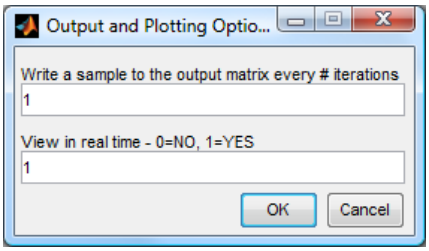
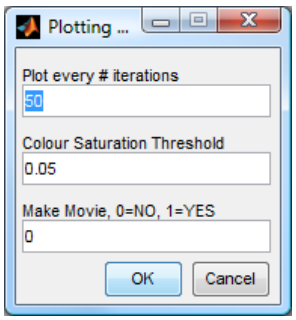


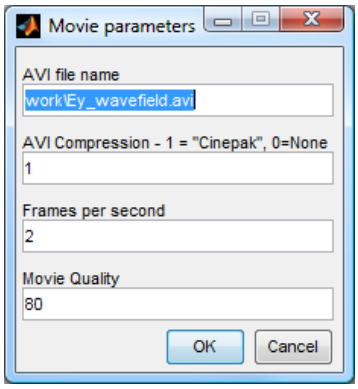
Figure 8.14. Left: dialogue box to specify the discretization parameters. **Right:** Dialogue box to specify the sampling interval and total 2-way traveltime

Also based on the initial discretization, the program computes maximum permissible values for *dt* and the total length of the traveltime vector and requests that you specify these parameters (Figure 8.14-right). Here as well, the program suggests values close to the maximum permissible.

- The final model grids for the FDTD simulation are constructed with the final values of *dx* and *dz*.







The program offers the possibility to:

- Control the size of the output (*Write sample to the output every # iterations*). Default = 1.
- View the simulation process in real time, i.e. view the simulated E_y wavefield as it propagates through the model (default = 1, i.e. view in real time).

In the latter case, you must specify:

- How frequently to update the simulation (*Plot every # iterations*). Default is *Nplot* = 50.
- The contrast (colour saturation) by which to accentuate the wavefield characteristics. Default = 0.05.
- It is also possible to make AVI movies of the simulation. To do this, you should enter 1 in the *Make Movie* dialogue (default = 0, i.e. no movie)

In order to make a movie, you need to specify:

- The AVI file name (the default is *work/Ey_wavefiled.avi*)
- the compression method (default is 1 ≡ 'Cinepak', otherwise 0, i.e. no compression)
- The number of frames per second (default = 2) and,
- The quality of the compression (default = 80).

If real time viewing of the simulation is selected, the progress is shown displayed as per Figure 8.15, which is updated every *Nplot* iterations (real time graphic display).

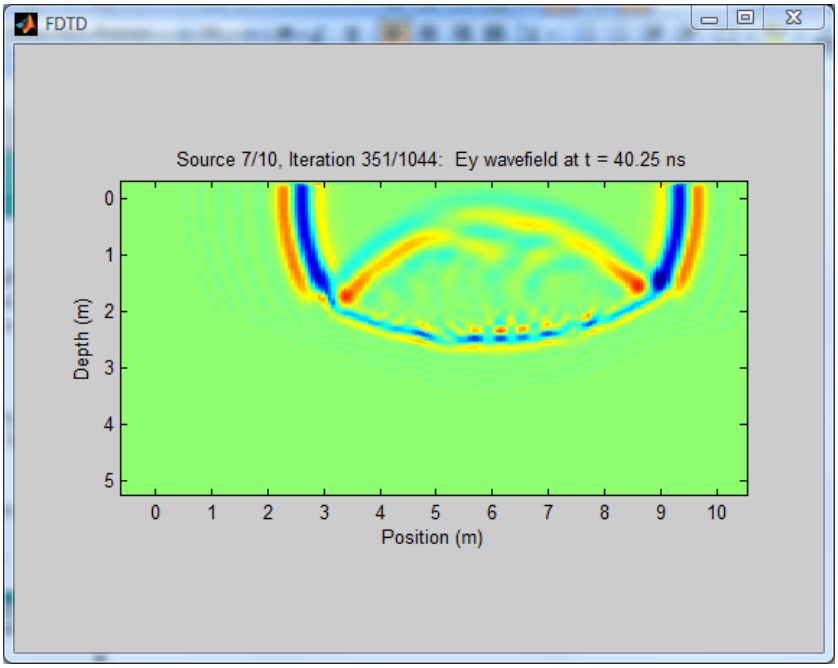
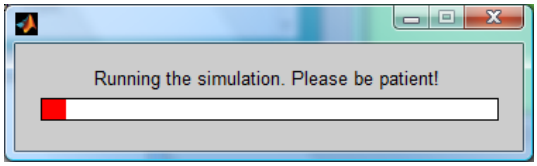
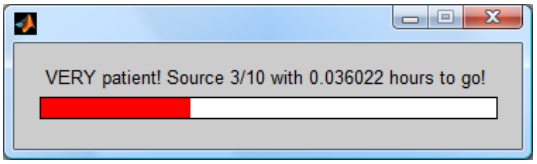


Figure 8.15. The progress of the simulation, i.e. the propagation of the E_y wavefield in the model can be graphically displayed in real time.

If real time plots are not desired, matGPR displays the progress in a wait bar which is continuously updated with time-to-completion information.





Additional Notes:

The FDTD solution of the forward problem can be *quite slow*, especially when large models and high frequencies are involved. In the latter case, the size of the problem may overflow the memory allocatable by MATLAB. Normal sized problems may take several hours, even days to complete and it is advisable to run a trial simulations (e.g. with a very small number of source locations) before committing your system to heavy number crunching. The size of the model is very important in this respect and it is always recommended to scale it down with increasing frequency.

CHAPTER 9. Interpretation III: The "3-D Utilities" menu

matGPR offers a suite of functions to assemble, manipulate and visualize three-dimensional GPR data volumes. The first group of these functions enables the generation of data volumes (albeit with limitations) and their presentation in the form of isometric surfaces (equal intensity of reflection) and 3-D slices with, or without translucence. These utilities are only a first step toward a more complete and comprehensive set of 3-D visualization tools, designed to facilitate interpretation of complex data sets.

Load 3-D data. Import a 3-D data volume previously created by selecting **Create 3-D Data** or **GRR-Slice** and saved to disk file (M3D-file format).

Create 3-D Data. Generate a data volume from 2-D GPR profiles for 3-D vizualization. The program assumes that the 3-D GPR survey has been conducted in a local co-ordinate system with the X-axis pointing to the right and the Z-axis pointing down. In this system, the radargrams (GPR profiles) can have *any* shape, length or orientation with respect to the axes. Moreover:

- The input profiles need not have the same trace spacings or be equally spaced in any direction. The vertical axis can be either time or depth. The data need not have the same traveltime vectors, or sampling rates, or depth vectors, or depth spacings.
- Input data is acceptable *only* in the matGPR-specific MGP and MAT formats.
- X-, Y- and Z- coordinates of *all traces* in *all profiles* must have been assigned prior to the generation of the volume, otherwise the process will abort (see Make XYZ item of the 'Basic Handling' menu).

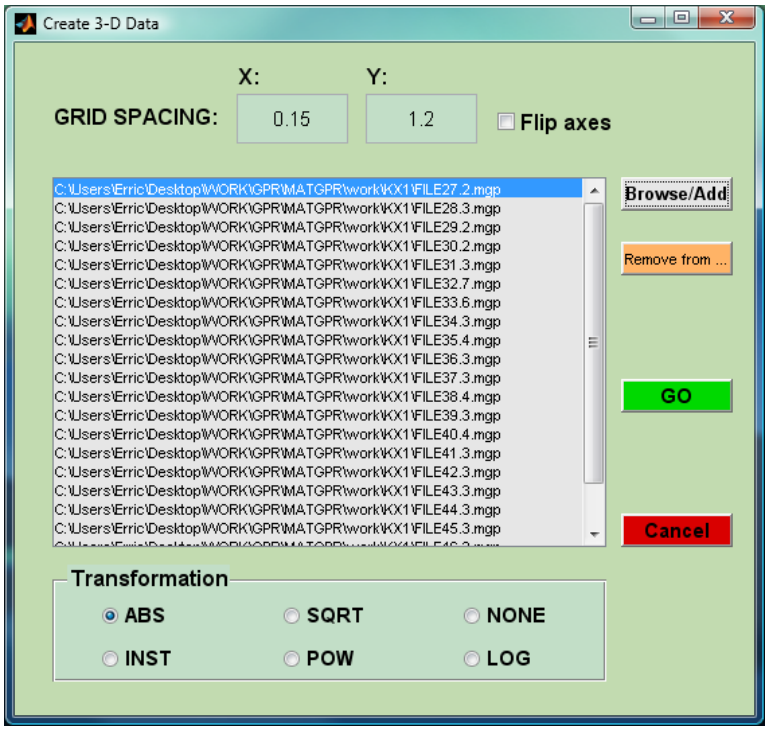


Figure 9.1. The GUI to assemble files for 3-D data volume generation

The general procedure is as follows. GPR data files (GPR profiles) are read in. For each file, traces that fall within a two-dimensional horizontal search window around a volume cell are transformed (e.g. converted to absolute values or instantaneous amplitudes) and stacked together. After all the data files have been processed, the final volume cell value is calculated by averaging all values added to it.

On selecting **3-D Utilities → Create 3-D Data**, matGPR generates the GUI of Figure 9.1 – which is similar in layout and function to the Concatenate Sections window of the 'Data' menu – to assemble the GPR data files and take the necessary parameters. This can be done either by typing full path names, or by using the file browser. To assemble the files press the **Browse/Add** button to invoke the file browser and select *multiple files* using the Cntrl and Shift keys as per Figure 9.2. Files can be taken out of the list by selecting them individually and pressing the **Remove from list** button.

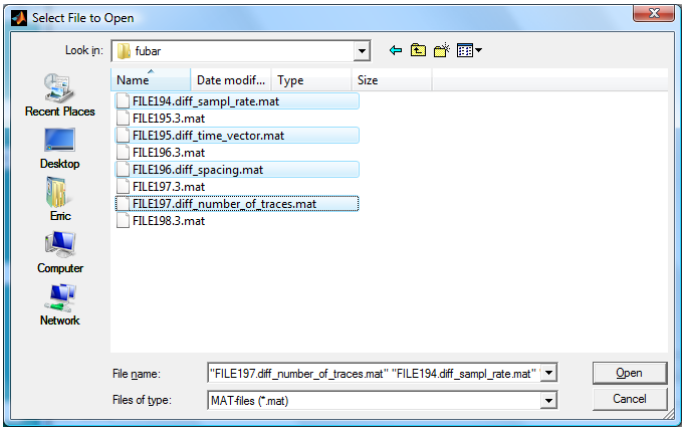


Figure 9.2. Multiple file selection.

- It is ***necessary*** to specify the horizontal spacings of the 3-D grid (volume) in the *X*- and *Y*- directions by typing the spacings (in data units) in the editable boxes located above the file list.
- Occasionally, it is convenient to measure in a system in a right-handed coordinate system in which the *Y-axis* points to the right (for instance when measuring in a rectangular field with profiles parallel to the smaller dimension). You can change the handedness of the coordinate system, i.e. make it compatible with left-handed geographical systems in which the X-axis point to the right – by checking the '**Flip axes:**' box – this interchanges (flips) the X and Y axes.
- As mentioned above, the volume is generated by stacking traces. It is possible to generate different types of data volumes by applying different transformations to the input traces. This is controlled by the options of the **'Transformation'** panel. Thus, it is possible to create slices of:
  - Absolute trace values, by pressing the **ABS** button (default).
  - Squared trace values, by pressing **SQRT**.
  - Log-absolute trace values by pressing **LOG**.
  - Instantaneous trace amplitude, by pressing **INST**.
  - Instantaneous trace power, by pressing **POW**.
  - Untransformed stacked traces by pressing **NONE**.

The **GO** button starts the volume generation process. The procedure is as follows:

Import GPR profiles and store in temporary work arrays
For all profiles
    If necessary, homogenize sampling rates or depth spacings by interpolation.
    Equate the lengths of traveltime or depth vectors by trimming or padding with zeros.
Initialize the XYZ grid (volume) with the size of Z being equal to the size of the homogenized traces.
For every GPR profile,
  For every X in X-Y grid,
    For every Y in X-Y grid,

For every trace in GPR profile,
If the X and Y locations of a trace are within an X-Y grid cell,
Transform the trace if necessary
Stack the trace along the Z dimension of the XYZ grid
Compute averages of the stacked traces.

If one or more 2-D radargrams have different sampling rates or depth spacings, then the medians of all the sampling rates, or trace spacings, or depth spacings are set to be the respective unique common parameter of the data ensemble and volume, and the radargrams are automatically resampled. Finally, if one or more radargrams have different traveltime vectors, or depth vectors, or scan-axis lengths, the user is asked to decide the respective dimension of the volume, with dialogues as per Figure 9.3. Thus, either the longer dimensions are trimmed to the shortest dimension in the radargram ensemble, or the shorter dimensions are zero-padded to the longest dimension in the ensemble.

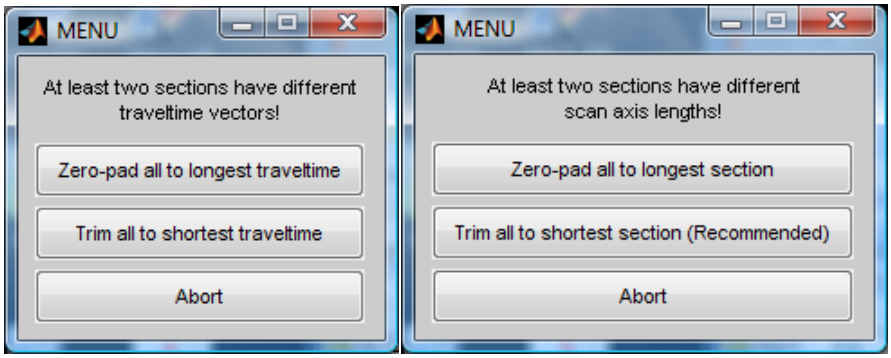


Figure 9.3. If one or more radargrams have different traveltime vectors, or depth vectors, or scan-axis lengths, the user is asked to decide the respective dimension of the final data volume.

GPR-Slice. Generates a 3-D volume of GPR data and enables its visualization in the form of opaque or translucent slices (sections). The volume may be exported to the MATLAB workspace whence it can be saved to a disk file in M3D or ASCII column format, or be visualized with the 'Isosurface Display' and '3-D Slices' techniques to be described below.

The method of 3-D data generation is presented in the 'Create 3-D Data' item above. However, the volume and the slices are created with the assistance of the GPR-Slice GUI (Figure 9.4). The left hand side of the GUI is used to assemble the files necessary to generate the 3-D volume and is identical in function to the Create 3D Data GUI described above. To assemble the files press the **Browse/Add** button to invoke the file browser and select *multiple files* using the Cntrl and Shift keys. Files can be taken out of the list by selecting them individually and pressing the **Remove from list** button. As before, it necessary to specify the horizontal spacings of the 3-D grid in the *X*- and *Y*- directions by typing the spacings (data units) in the editable boxes located above the file list. Likewise, is possible to toggle the handedness of the coordinate system by checking the '**Flip axes:**' box. Finally, it is possible to apply different transformations to generate different types of data volumes through the choices of the **'Transformation'** panel. The **VOLUME** button starts the volume generation process.

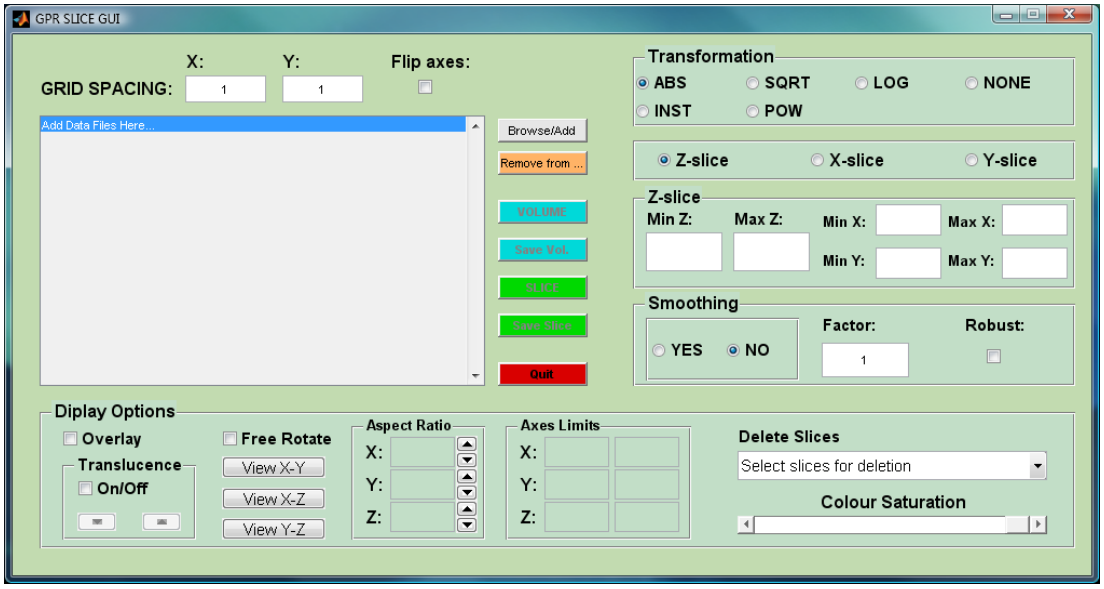


Figure 9.4. The GPR-Slice GUI with GPR files loaded and ready to create the 3-D data volume.

Once the volume is created:

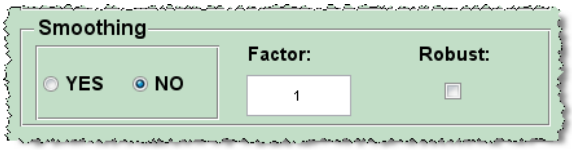
- It can be exported to the MATLAB workspace by pressing the **Save Vol.** button and hence be saved to disk file (in M3D-file or ASCII column format), or visualized with the 'Isosurface Display' and '3-D Slices' techniques to be described below.
- It can be sliced perpendicular to any one of the three coordinate axes. Slices are specified by the panels located at the mid-right of the GUI.

  The most common type of slice would be *horizontal layers* perpendicular to the *vertical* grid direction (Z-axis). A horizontal slice is requested by pressing the **Z-slice** button; it comprises the average values of *vertical columns* located within the layer specified by **Min Z** and **Max Z** and extending between the grid intervals [**Min X, Max X**] in the *x-direction* and [**Min Y, Max Y**] in the *y-direction*. All values are given in data units. The GUI initializes the editable boxes with the minimum and maximum values of the three grid axes and you can determine the actual extent of the slice by changing them to the desired values. The slice will be plotted at the midpoint of the layer Min Z – Max Z.

  Likewise, *vertical* slices perpendicular to *x*-direction can be requested by pressing the **X-slice** button. The bottom panel now changes to accept the *X-slice* specifications, which analogously comprise the average values of *rows parallel to the y-direction*, located in the *x*-interval [**Min X, Max X**] and extending in the vertical interval (layer) [**Min Z, Max Z**] and the *y*-interval [**Min Y, Max Y**]. Again, all

values are given in data units. The GUI initializes the editable boxes with the minimum and maximum values of the three grid axes and you can determine the actual extent of the slice by changing them to the desired values. The slice will be plotted at the midpoint of the interval Min X – Max X.

Finally, *vertical* slices perpendicular to the *y*-direction can be requested by pressing the **Y-slice** button. The bottom panel now changes to accept the *Y-slice* specifications, which similarly comprise the average values of *rows parallel to the x-direction*, located within the *y*-interval [**Min Y, Max Y**] and extending in the layer [**Min Z, Max Z**] and the *x*-interval [**Min X, Max X**]. Here also, all values are given in data units. The GUI initializes the editable boxes with the minimum and maximum values of the three grid axes and you can determine the actual extent of the slice by changing them to the desired values. The slice will be plotted at the midpoint of the interval Min Y – Max Y.



Prior to displaying, it is possible to smooth the slice with a robust discretized smoothing spline technique due to Garcia, (2010), which can handle missing data and tolerate NaNs or Infs. Press **YES** to activate smoothing. The method requires a real positive smoothing parameter (**Factor**); this can be very small, e.g. one thousandth, and larger it is, the smoother the result. For data with outliers you can use a robust computational procedure; to activate this option check the '**Robust**' box. The default is not-robust smoothing.

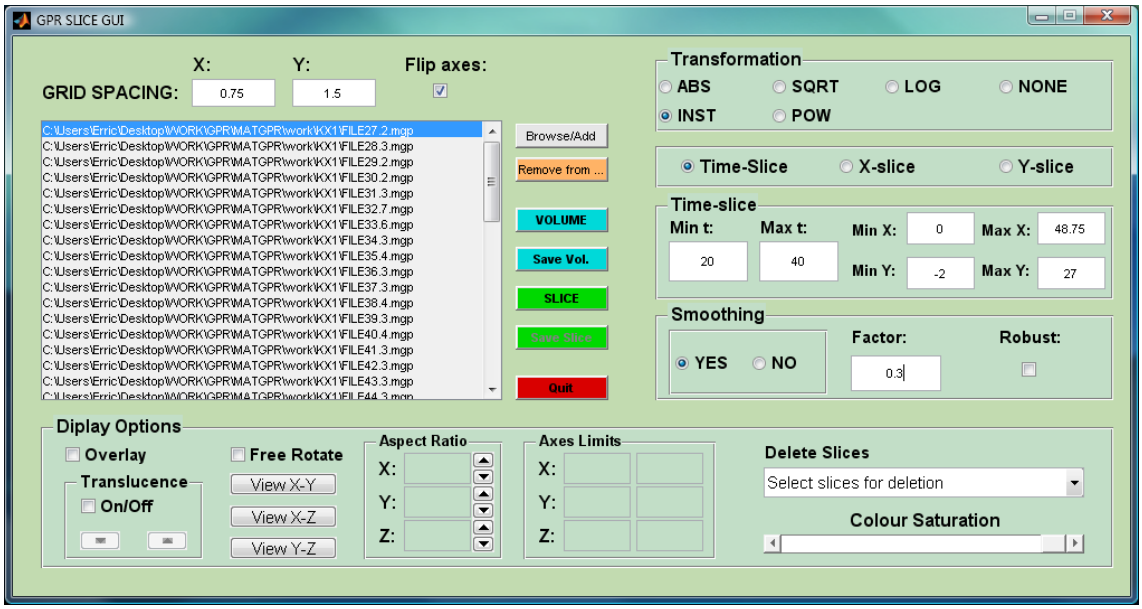


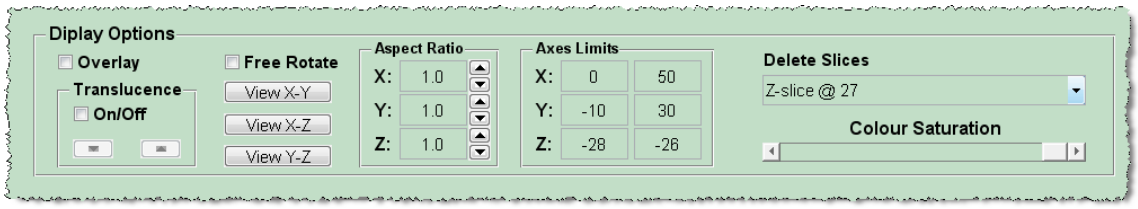
Figure 9.5. The GPR-Slice GUI after the 3-D volume has been created and the parameters for at least one Z-slice have been specified. The Z-axis has units of time.

- Once the parameters have been specified, e.g. Fig. 9.5, a slice can be created and displayed by pressing the **SLICE** button (e.g. Fig. 9.6). The *last* slice created can be exported to the MATLAB workspace by pressing **Save Slice**. The slice is saved as a structure 'CurrentSlice' with elements: *x*, a 2-D the array of X-coordinates values; *y*, a 2-D the array of Y-coordinates; *.z*, a 2-D the array of Z-coordinates; *sl*, a 2-D array containing the slice and *zlab*, the label of the vertical axis (time or depth).

Basic manipulation of the display is possible with the controls of the '**Display Options**' panel. You can:

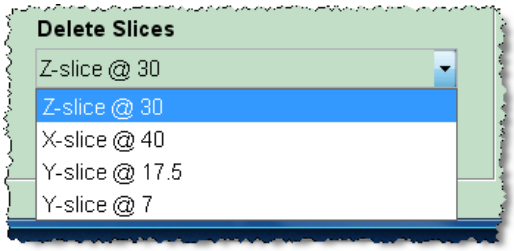
- Adjust the axes aspect ratio using the controls of the 'Aspect Ratio' panel. The default aspect ratio is 1:1:1 and may be changed either by means of the respective X, Y and Z editable boxes, or by means of the *increase* (▲) or *decrease* (▼) push buttons.

- Adjust the range of the display using the controls of the 'Axes Limits' panel. The limits are initially determined automatically by MATLAB and can be modified using the corresponding X, Y and Z editable boxes.



- Control the Viewer Position. Options available are:
  - Free interactive rotation using the mouse. The option is activated/ deactivated with the **Free Rotate** checkbox. Details about interactive rotation may be found in MATLAB documentation.
  - Plan view of Z-slices pressing the **View X-Y** button.
  - Profile view of X-slices pressing the **View X-Z** button
  - Profile view of Y-slices pressing the **View Y-Z** button

- Display several slices together. The default state of the '**Overlay**' checkbox is not checked. Every new slice will replace the precious one. By changing the state to checked, every new slice will be stacked on the display together with previous slices and they all be shown together (e.g. Fig 9.7). Unchecking the box reverts to the default state – new slices replace previous slices.
  - When a slice is created it is assigned with a tag composed of its orientation (L-, T- or Z-slice) and the coordinate at which it is plotted. These tags are listed in the **Deleted Slices** menu, as in the example shown below. By *selecting* the appropriate *tag* you can *delete* the corresponding *slice*.



- Display slices with translucence, whether created in single or overlay mode. To enable translucence, check the box labeled **On/Off** in the panel labeled **'Translucence'**. To disable, uncheck the box. Once enabled, the degree of transparency may be changed via the *increase* (▲) or *decrease* (▼) push buttons.
- Change Colour Saturation (contrast) whether in single or overlay mode with an appropriate slider. In combination with translucence, increasing the contrast allows the delineation of laterally extended targets.

At any time during a GPR-Slice session:

- New GPR files can be added to or removed from the list.
- The grid or/and transformation parameters can be changed and the volume recalculated and exported to the MATLAB workspace.
- Slices can be added to or removed from the 'GPR Slice' figure and the last slice can be exported to the MATLAB workspace.

☞ These operations can be done in any combination. Of course, some combinations are detrimental, as for instance overlaying slices with different transformations. This, however, is up to the user to mind about.

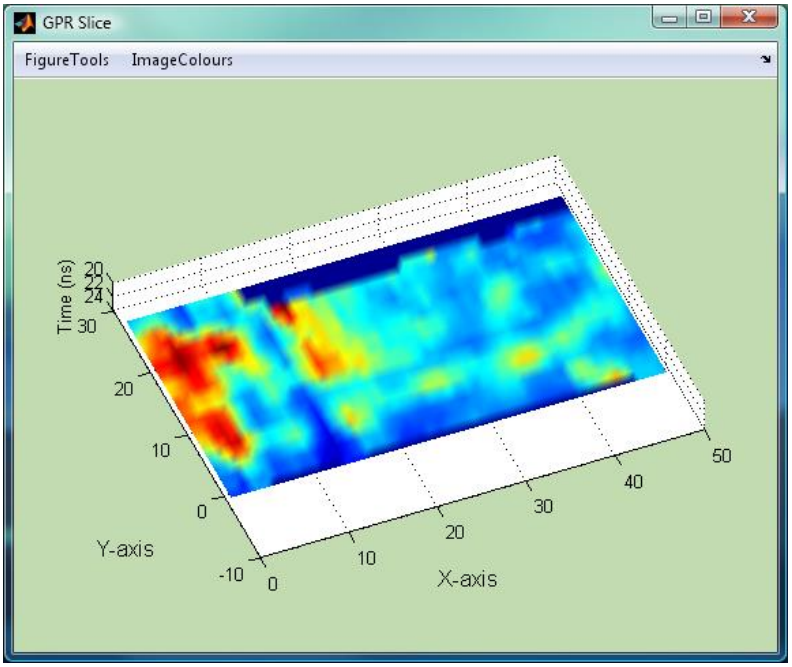


Figure 9.6. The Z-slice created with the specifications shown in Fig 9.5. It illustrates the outline of buried ruins.

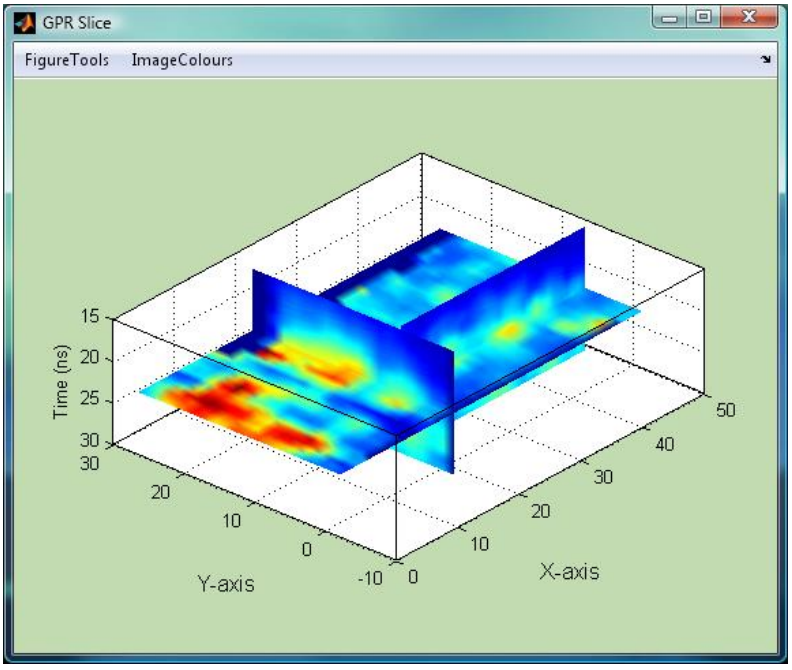


Figure 9.7. The slice of Figure 9.6 augmented with X- and Y-slices and viewed from a different perspective.

Isosurface Display. Enable the presentation of 3-D data in the form of (isometric) surfaces with equal signal amplitudes, or less rigorously stated surfaces of equal reflectivity.

When invoked, this option will generate and display an orthographic projection of the isosurface of amplitudes equal to 55% of the maximum amplitude contained in the 3-D data volume, as per Figures 9.8 and 9.9.

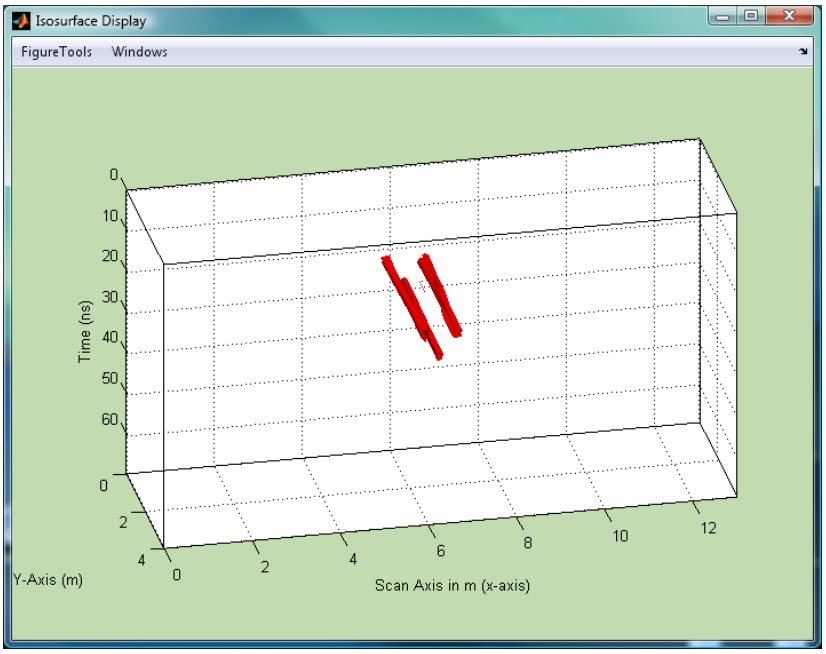


Figure 9.8. Equal signal amplitude display: Pipes buried at depths of 1.4–1.8 metres.

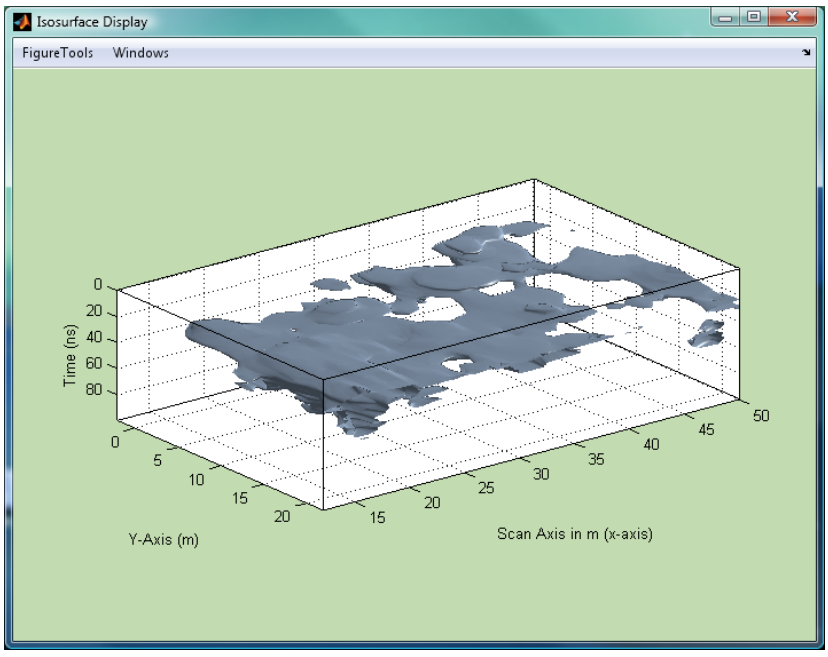


Figure 9.9. Equal instantaneous power display of a geological interface.

It will also create the "*Iso-Surface Display Controls*" shown in Figure 9.10, which allow you to manipulate and study the isosurface.

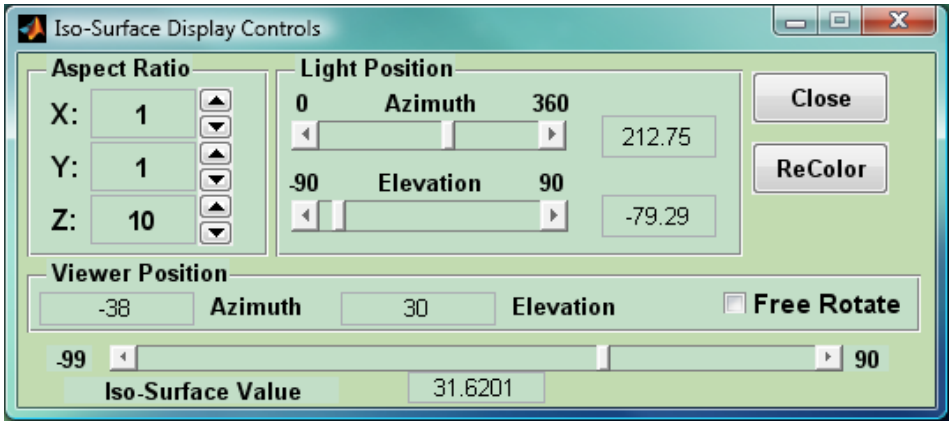


Figure 9.10. GUI to control isometric surface displays of 3-D GPR data volumes.

The controls include:

1. Specification of the isosurface value (signal amplitude) using the **Iso-Surface Value** slider *and editable* box.
2. Specification/ change of Viewer Position, either using the **Azimuth** and **Elevation** boxes, or by free interactive rotation using the mouse. The latter option is activated/ deactivated with the **Free Rotate** checkbox; details about interactive rotation may be found in MATLAB documentation. Notably, either option is simultaneously available.
3. Specification/ change of the axes **Aspect Ratio**. The default aspect ratio is X:Y:Z = 1:1:10 and changes can be effected either by means of the respective X, Y and Z editable boxes, or gradually by means of the respective *increase* (▲) or *decrease* (▼) push buttons (fine tuning).
4. Specification/ change of the direction of lighting (**Light Position**). The light source is located at infinity (parallel rays) and you may only specify the azimuth and elevation of the light source using the respective sliders and editable boxes. This function may be useful in studying the details of the iso-intensity surface.
5. Change the colour of the iso-surface using the **ReColor** button.

3-D Slices. Enable the presentation of 3-D data in the form of opaque or translucent slices (sections). When invoked, this option will generate and display an orthographic projection of the radargrams in the form of slices (sections) parallel to the scan-axis (X-axis), as in Figure 9.11. The slices are initially opaque.

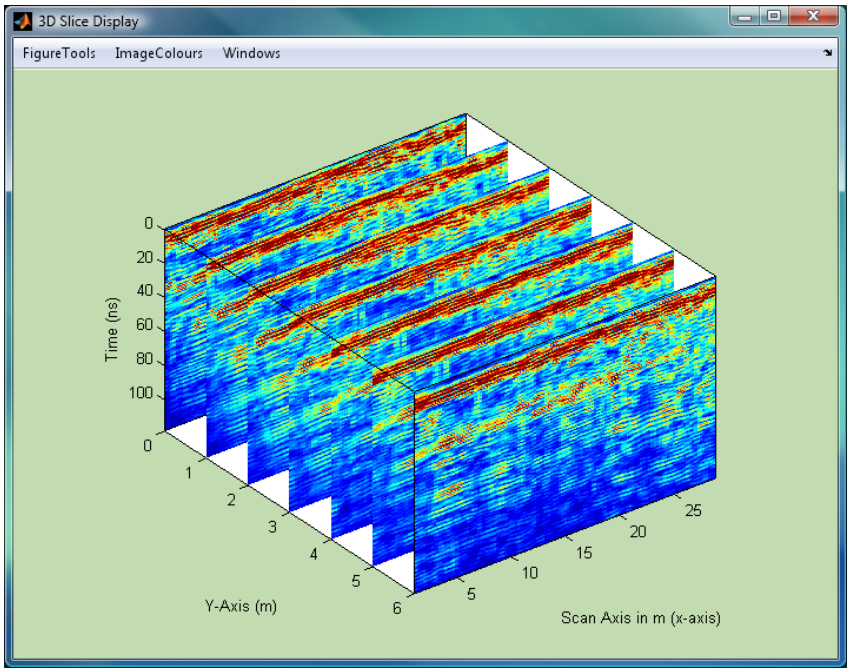


Figure 9.11. Opaque 3-D slices of instantaneous amplitudes parallel to the scan axis.

It will also create the "*3-D Display Controls*" shown in Figure 9.12, which allow you to manipulate and study the slices.

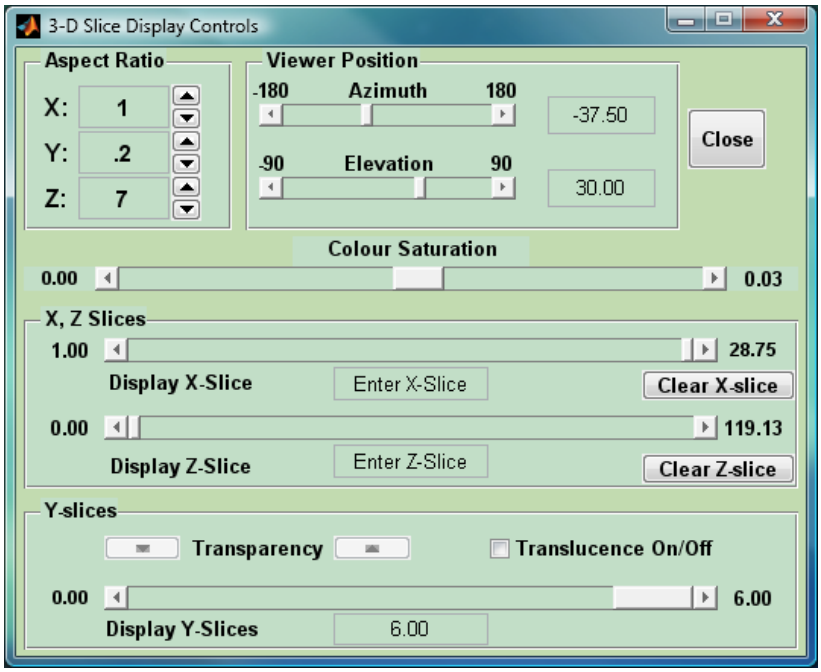


Figure 9.12. GUI to control 3-D slice presentation of GPR data volumes.

The GUI is organized in panels of homologous control groups as follows:

1. Y-slices panel: This is the most important group of controls because it determines the number of slices displayed and their transparency. The number of slices shown is controlled via a slider and an editable box, both labeled **Display Y-slices**, according to their location in metres, along the Y-axis. All of the Y-slices, i.e. all the profiles comprising the data volume are shown by default at program startup and the maximum Y-value is displayed in the slider and editable box. The slices are arranged from last to first along the Y-axis. You may decrease the number of slices by moving the slider to the left (from last to first) and increase by moving it to the right (from first to last); in this way, you may scan for laterally extended targets through the data volume. You may also type an exact Y-location to jump to, via the editable box.

It is also possible to display the slices with translucence. To enable translucence, check the box labeled **Translucence On/Off**. To disable, uncheck the box. Once enabled, the degree of transparency may be changed via the *increase* (▲) or *decrease* (▼) push buttons. Figure 9.13 shows the same data as per Figure 9.11, but with translucence. The method by which to introduce translucence in the display is credited to Conroy and Radzevicius (2003).

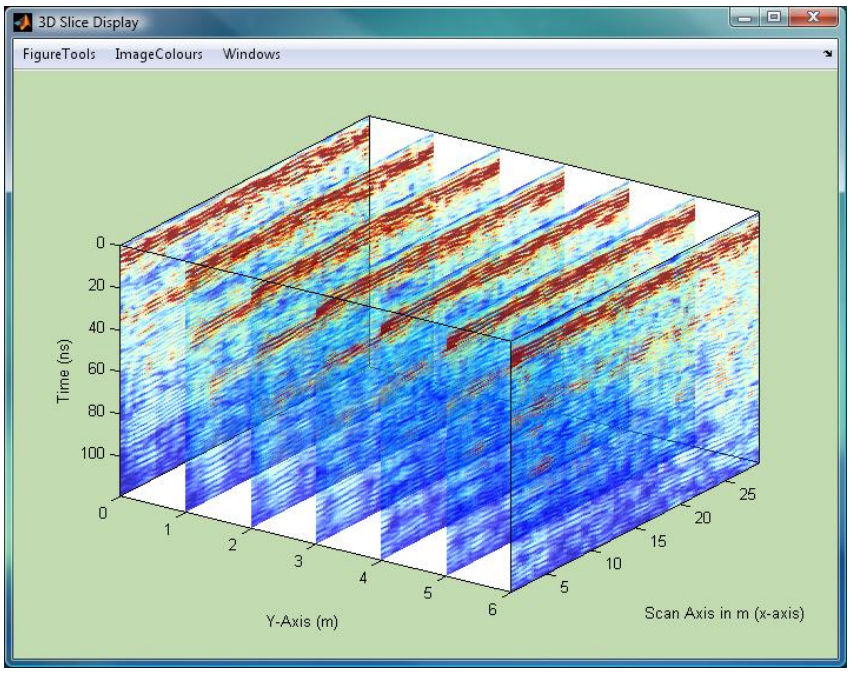


Figure 9.13 The data of Fig. 9.11 displayed with translucence.

2. X-Slices and Z-Slices panels: These controls allow you to insert slices parallel to the Y-axis (X-slices) or/and (horizontal) slices parallel to the X-Y (Z-slices). Only a single, opaque X-slice and/or a single, opaque Z-slice are allowed at a time. The slices are opaque because their main purpose is to provide a backdrop to facilitate the interpretation of the (translucent) Y-slices.

The location of X- and Z- slices can be controlled with sliders and editable boxes. X- and Z- slices are not displayed at start-up and the respective boxes are labelled **Enter X-slice** and **Enter Y-slice**. You may insert them, either by moving the respective sliders to the appropriate position, or by directly typing their location in metres (X-slices) and metres or ns (Z-slices) in the respective editable boxes. Finally, X- and Z- slices can be removed altogether, with the **Clear X-slice** and **Clear Z-slice** buttons respectively.

Figure 9.14 shows the same data as per Figure 9.13, but with increased transparency, an X-slice at 23m along the scan axis (where a buried wall was detected), and a Z-slice at 38ns along the traveltime axis, where there are intense reflections from a buried interface.

3. Colour Saturation (contrast) can be varied with an appropriate slider. In combination with translucence, increasing the contrast allows the delineation of interfaces and laterally extended targets.

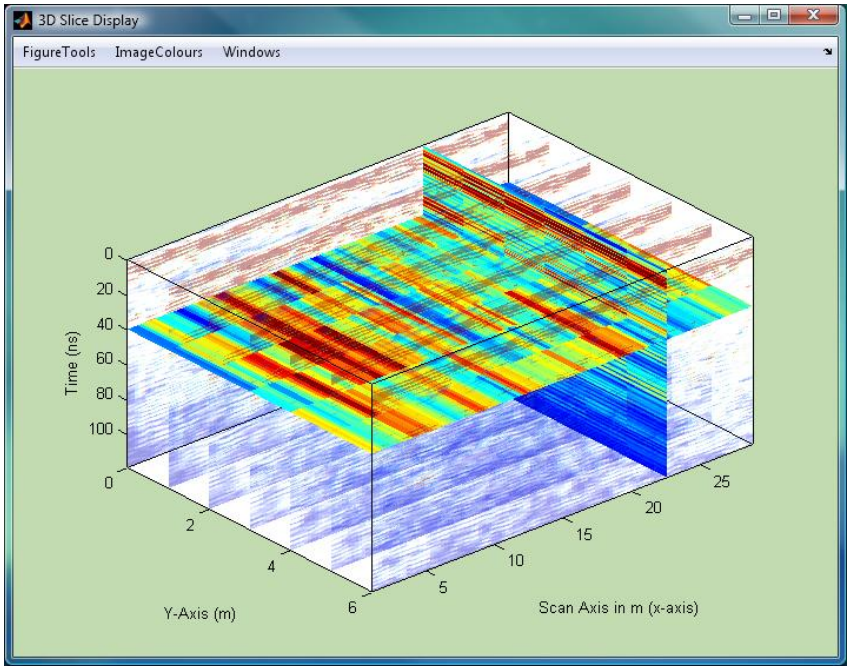


Figure 9.14. The data of Figures 9.11 and 9.13, with increased transparency and an X-slice and a Z-slice overlaid.

4. Aspect Ratio: The default aspect ratio is X:Y:Z = 1:1:10 and may be changed either by means of the respective X, Y and Z boxes, or gradually by means of the *increase* (▲) or *decrease* (▼) push buttons (fine tuning).

5. Viewer Position: The azimuth and elevation of the observer with respect to the experimental coordinate system can be specified with the respective **Azimuth** and **Elevation** sliders and boxes. All values must be given in degrees.

Save 3-D Data. Export to an M3D-file, a 3-D data volume previously generated by the Create 3-D Data command.

CHAPTER 10. REFERENCES

Bano, M., 1996. Constant dielectric losses of ground-penetrating radar waves, *Geophysical Journal Int.*, 124, 279-288.

Bitri, A. and Grandjean, G., 1998. Frequency - wavenumber modelling and migration of 2D GPR data in moderately heterogeneous dispersive media, *Geophysical Prospecting*, 46, 287-301.

Canales, L. L., 1984, Random noise reduction: 54th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, Session:S10.1.

Candès, E. and Donoho, D., 2003a. Continuous curvelet transform: I. Resolution of the wavefront set. Appl. Comput. Harmon. Anal., 19, 162-197.

Candès, E. and Donoho, D., 2003b. Continuous curvelet transform: II. Discretization and frames. Appl. Comput. Harmon. Anal., 19, 198-222.

Candès, E. and Donoho, D., 2004. New tight frames of curvelets and optimal representations of objects with piecewise C2 singularities. Comm. Pure Appl. Math., 57, 219-266.

Candès, E. J., L. Demanet, D. L. Donoho, and L. Ying, 2006. Fast discrete curvelet transforms (FDCT). Multiscale Modeling and Simulation, 5, 861–899.

Chopra, S. and Marfurt, K.J., 2007. Volumetric curvature attributes for fault/fracture characterization, First Break, 25 (7), 35-46; doi: 10.3997/1365-2397.2007019.

Claerbout, J., 1996, *Imaging the Earth's Interior*, Network Distribution Version, pp. 50-57 and 222-223; http://sepwww.stanford.edu/sep/prof/index.html.

Conroy, J. P. and Radzevicius, S. J., 2003. Compact MATLAB code for displaying 3D GPR data with translucence, *Computers and Geosciences*, 29/5, 679-681.

Crochiere, R and Rabiner, L. R., 1983. *Multirate Digital Signal Processing, Englewood Cliffs*, NJ: Prentice-Hall, Inc.

Davis, J.L. and Annan A.P., 1989. Ground-penetrating radar for high resolution mapping of soil and rock stratigraphy, *Geophysical Prospecting*, 37, 531-551.

Fukunaga, K., 1990, "Introduction to Statistical Recognition", Academic Press.

Garcia, D., 2010. Robust smoothing of gridded data in one and higher dimensions with missing values, Computational Statistics and Data Analysis, 54, 1167 – 1178; DOI:10.1016/j.csda.2009.09.020.

Gazdag, J., 1978. Wave equation migration with the phase-shift method, *Geophysics*, 43, 1342-1351.

Gazdag, J. and Sguazzero, P., 1984. Migration of seismic data by phase-shift plus interpolation, *Geophysics*, 49, 124-131.

Grandjean, G. and Durand, H., 1999. Radar Unix: a complete package for GPR data processing, *Computers & Geosciences*, 25 141-149.---

Gulunay, N., 1986, FX deconvolution and complex Wiener prediction filter: 56th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, Session:POS2.10.

Irving, J.D. and Knight, R.J., 2003. Removal of wavelet dispersion from ground-penetrating radar data. Geophysics, 68 (3), 960-970.

Irving, J. and Knight, R., 2006. Numerical modeling of ground-penetrating radar in 2-D using MATLAB, *Computers & Geosciences*, 32, 1247–1258.

Kahaner, D., Moler, C. and Nash, S., 1989. Numerical Methods and Software; Prentice Hall.

Karhunen, K., 1946, "Zur Spektraltheorie Stochastischer Prozesse", *Ann. Acad. Sci. Fennicae*, 372.

Loeve, M.M., 1955, "Probability Theory", Princeton, N.J.: VanNostrand.

Lucius, J.E. and Powers, M.H., 2002. GPR Data Processing Computer Software for the PC, USGS Open-File Report 02-166.

Roberts, A., 2001. Curvature attributes and their application to 3D interpreted horizons. First Break, 19 (2), 85–100; doi: 10.1046/j.0263-5046.2001.00142.x.

Sacchi, M.D., 1997. Re-weighting strategies in seismic deconvolution, Geophysical Journal International, 129, 651-656.

Sena, A.R., Stoffa, P.L. and Sen, M. K., 2003. Split Step Fourier Migration of Ground Penetrating Radar Data: 73th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1023-1026.

Smith, J. O. and Gossett, P., 1984. A flexible sampling-rate conversion method in *"Proceedings of the International Conference on Acoustics, Speech, and Signal Processing"* ICASSP-84, Volume II, pp. 19.4.1-19.4.2, New York: IEEE Press. Also, see expanded tutorial and associated free software at the *Digital Audio Resampling Home Page*: http://www-ccrma.stanford.edu/~jos/resample/

Stockwell, R.G., Manshinha, L. and Lowe, R.P., 1996. Localization of the complex spectrum: The S-transform. IEEE Trans. Signal Proces., 44, 998-1001.

Stoffa, P.L., Fokkema, J.T., de Luna Freire, R.M. and Kessinger, W.P., 1990. Split-step Fourier migration, *Geophysics*, 55, 410-421.

Stolt, R.H., 1978. Migration by Fourier Transform, *Geophysics*, 43, 23-48.

Tzanis, A., 2013. Detection and extraction of orientation-and-scale-dependent information from two-dimensional GPR data with tuneable directional wavelet filters, Journal of Applied Geophysics, 89, 48-67. DOI: 10.1016/j.jappgeo.2012.11.007.

Tzanis, A., 2015. The Curvelet Transform in the analysis of 2-D GPR data: Signal enhancement and extraction of orientation-and-scale-dependent information , Journal of Applied Geophysics, 115, 145-170; doi: http://dx.doi.org/10.1016/j.jappgeo.2015.02.015.

Turner, G. and Siggins, A.F., 1994. Constant *Q* attenuation of subsurface radar pulses, *Geophysics*, 59, 410-421.