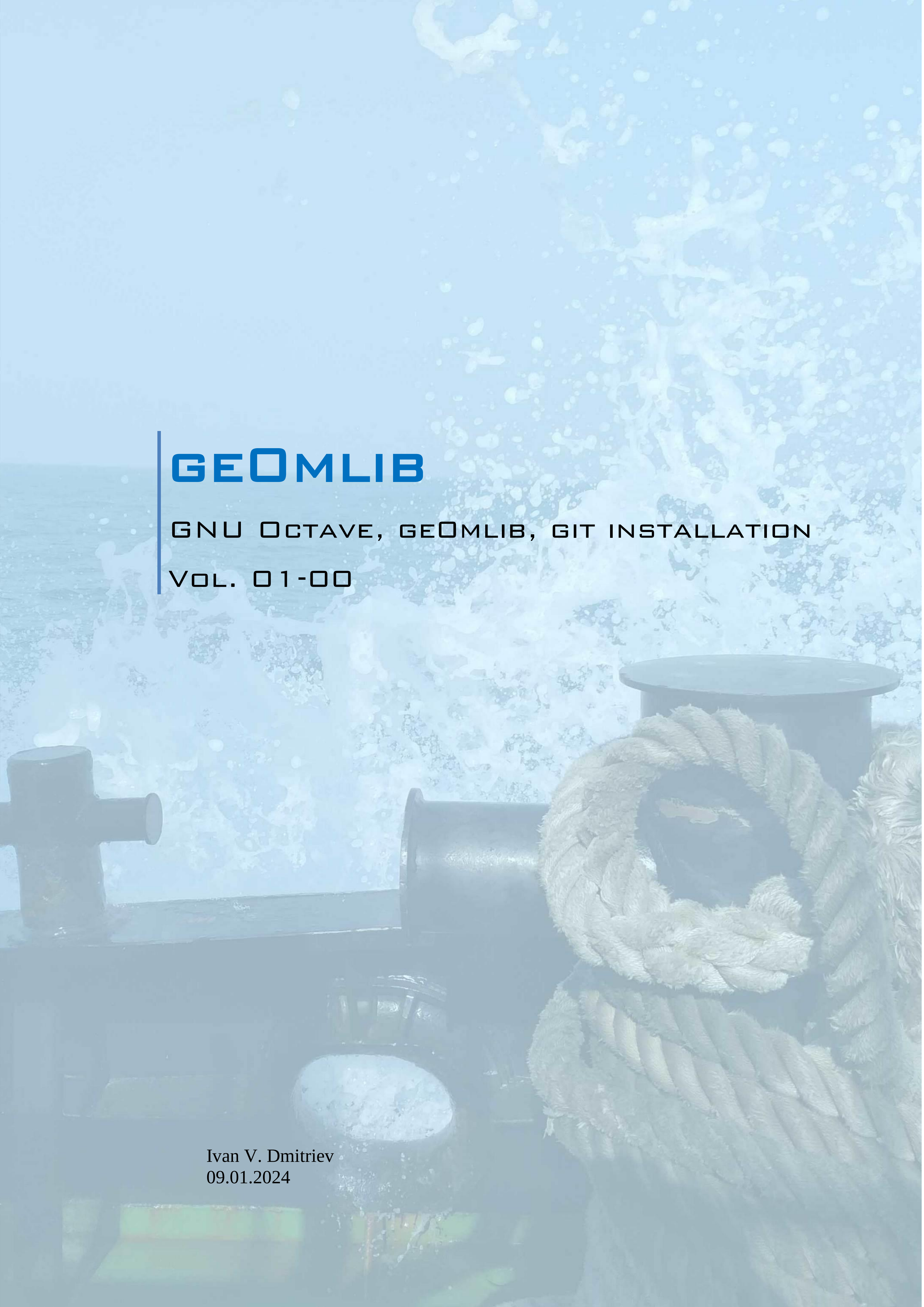


geOmlib

GNU Octave, geOmlib, git installation

Vol. 01-00

Ivan V. Dmitriev
09.01.2024

Contents

Introduction

The ge0mlib library is designed for marine engineering geophysics data processing. The processing functions are developed in MatLab 2018b (https://mathworks.com/) and adapted for the free software GNU Octave 8.4.0 (https://www.octave.org). Project website: https://ge0mlib.com/. The library is available on Git Hub: https://github.com/ge0mlib/ge0mlib.

This document is written for a non-programmer Windows user who needs to "make data processing" using a ready-made script written for MatLab or Octave. To complete the task, there are:

1) m-file with a script for "data processing", previously written for the MatLab and/or GNU Octave programming language (the development date can be found at the end of the script text);
2) a list of additional MatLab and/or GNU Octave libraries that must be installed to run the script (written in the comments at the beginning of the script text);
3) a description of the script commands or verbal instructions of the form "processed with this script" (a description of each individual command should be given in the text of the script, and examples of "working sequences" of commands should also be given there).

The purpose of this document is to provide a quick installation of the software to begin processing data. The document describes the following steps:

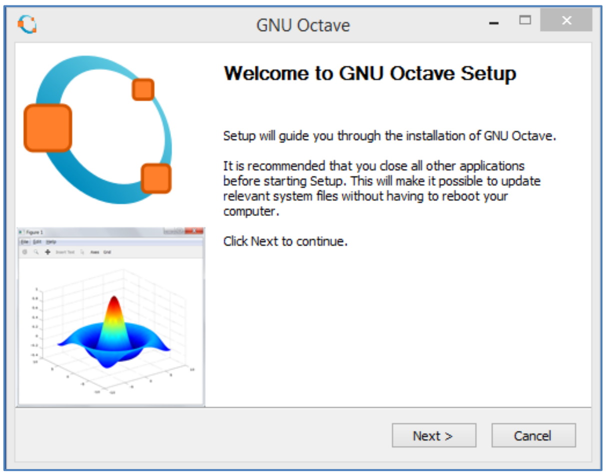
-- download and install GNU Octave (Windows version);
-- download and install additional packages (libraries) for GNU Octave (Windows);
-- download and install the ge0mlib library;
-- download and install Git (Windows version);
-- restoring a snapshot of the ge0mlib library to the date the script was developed using Git;
-- work with script commands (description of the principles of constructing script commands and information contained in the script body-text).

The text of the document includes a little more information than necessary. However, this information makes you feel more comfortable working with GNU Octave, ge0mlib and Git. Such "optional reading" text is highlighted in blue.

1 GNU Octave

1.1 GNU Octave installation

Octave distribution kit is downloaded from https://octave.org/download. At the moment of writing this text, it is the file octave-8.4.0-w64-installer.exe (for Win64).

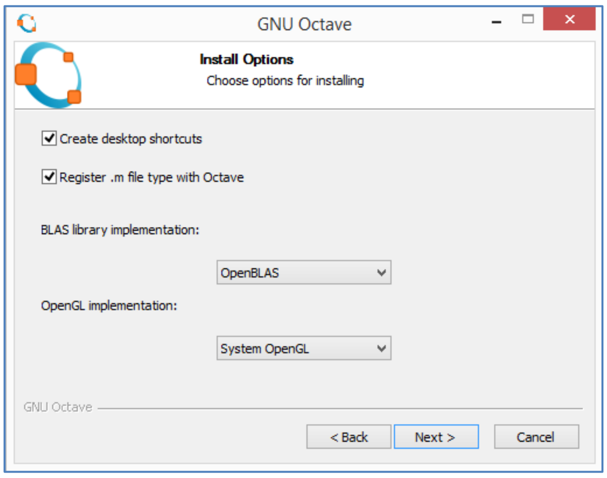


Welcome to GNU Octave Setup

Setup will guide you through the installation of GNU Octave.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.



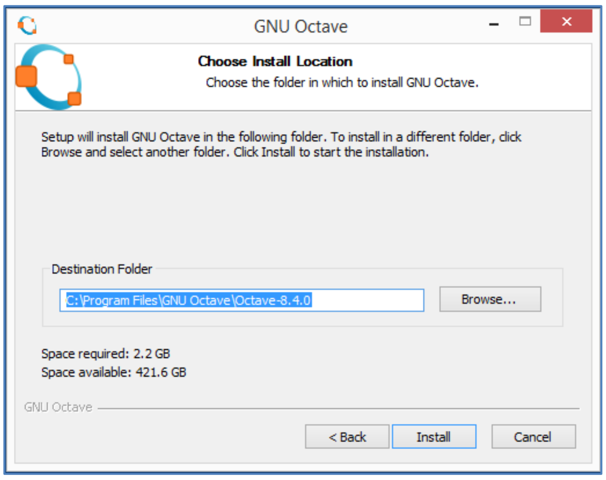
Install Options
Choose options for installing

☑ Create desktop shortcuts

☑ Register .m file type with Octave

BLAS library implementation: OpenBLAS

OpenGL implementation: System OpenGL



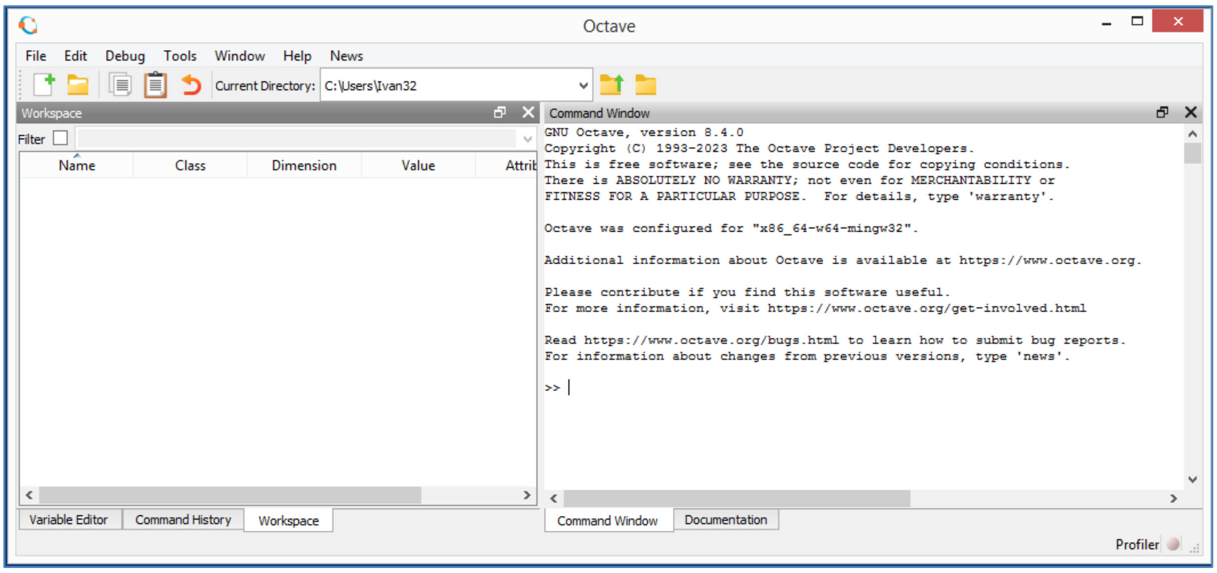
Choose Install Location
Choose the folder in which to install GNU Octave.

Setup will install GNU Octave in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder
C:\Program Files\GNU Octave\Octave-8.4.0

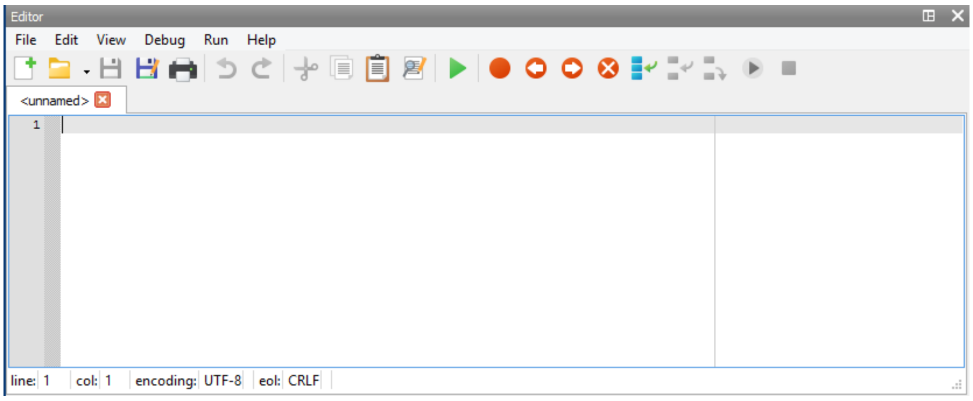
Space required: 2.2 GB
Space available: 421.6 GB

The Octave command window after installation, with the tabs moved for convenience, is shown below. Pay attention to the Variable Editor, Command History and Workspace tabs.



```
GNU Octave, version 8.4.0
Copyright (C) 1993-2023 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

>> |
```

The m-file debugger window is shown below. It may be convenient to enter a sequence of script commands in the debugger window, then save this file in the Script folder (a description of the folder is given below in section 2, when describing the installation of ge0mlib). After this, instead of entering a "sequence of script commands" into the command window, you can enter only the name of the file generated in the debugger (this file itself, in fact, will be a simple script).


Editor
File Edit View Debug Run Help
<unnamed>
line: 1 col: 1 encoding: UTF-8 eol: CRLF

1.2 Additional packages (libraries) installation

Additional libraries in Octave are usually called "packages", and in MatLab – "toolbox". The link to the page with the new version of the official packages for Octave: https://gnu-octave.github.io/packages/.

The link to the page with old versions of packages: https://octave.sourceforge.io/packages.php

There is a list of packages below look useful for solving marine geophysics problems. Of course, the selection is very subjective and is intended primarily to give a general idea of the packages included in GNU Octave.

control – Computer-Aided Control System Design (CACSD) Tools for GNU Octave, based on the SLICOT Library.

data-smoothing – Algorithms for smoothing noisy data.

divand – Performs an n-dimensional variational analysis (interpolation) of arbitrarily located observations.

fda – Functional Data Analysis.

fileio – I/O function for files holding structured data, such as JSON and XML files.

fuzzy-logic-toolkit – A mostly MATLAB-compatible fuzzy logic toolkit for Octave.

geographiclib – Native Octave/MATLAB implementations of a subset of the C++ library, GeographicLib. Key components of this toolbox are: (a) Geodesics, direct, inverse, area calculations; (b) Projections, transverse Mercator, polar stereographic, etc; (c) Grid systems, UTM, UPS, MGRS; (d) Geoid lookup, egm84, egm96, egm2008 geoids supported; (e) Geometric transformations, geocentric, local cartesian; (f) Great ellipse, direct, inverse, area calculations.

geometry – Library for extending MatGeom functionality.

image – Functions for image processing, feature extraction, image statistics, spatial and geometric transformations, morphological operations, linear filtering, and much more.

io – Input/Output in external formats.

linear-algebra – Additional linear algebra code, including matrix functions.

mapping – Simple mapping and GIS .shp .dxf and raster file functions.

matgeom – Geometry toolbox for 2D/3D geometric computing.

miscellaneous – Miscellaneous tools that don't fit somewhere else.

mvn – Multivariate normal distribution clustering and utility functions.

nan – A statistics and machine learning toolbox for data with and w/o missing values.

octproj – This package allows to call functions of PROJ library for cartographic projections and CRS transformations.

optim – Non-linear optimization toolkit.

optiminterp – An optimal interpolation toolbox providing functions to perform a n-dimensional optimal interpolations of arbitrarily distributed data points.

packajoozle – Enhanced package manager for GNU Octave.

pkg-octave-doc – This package provides functions for generating HTML pages that contain the help texts of the functions of an octave package. The package is designed to work with installed packages and use their INDEX file for creating the respective functions' HTML pages. The default layout is based on booootstrap 5 and it follows the design of the Octave Packages GitHub page.

quaternion – Quaternion package for GNU Octave, includes a quaternion class with overloaded operators.

signal – Signal processing tools, including filtering, windowing and display functions.

sparsersb – Interface to the librsb package implementing the RSB sparse matrix format for fast shared-memory sparse matrix computations.

splines – Additional spline functions

sqlite – Basic Octave implementation of the sqlite toolkit

statistics – The Statistics package for GNU Octave.

statistics-resampling – Estimate bias, uncertainty (standard errors and confidence intervals) and test hypotheses (p-values) using resampling methods. (Note that versions of this package <= 5.4.3 are named the statistics-bootstrap package).

stk – The STK is a (not so) Small Toolbox for Kriging. Its primary focus is on the interpolation/regression technique known as kriging, which is very closely related to Splines and Radial Basis Functions, and can be interpreted as a non-parametric Bayesian method using a Gaussian Process (GP) prior. The STK also provides tools for the sequential and non-sequential design of experiments. Even though it is, currently, mostly geared towards the Design and Analysis of Computer Experiments (DACE), the STK can be useful for other applications areas (such as Geostatistics, Machine Learning, Non-parametric Regression, etc.).

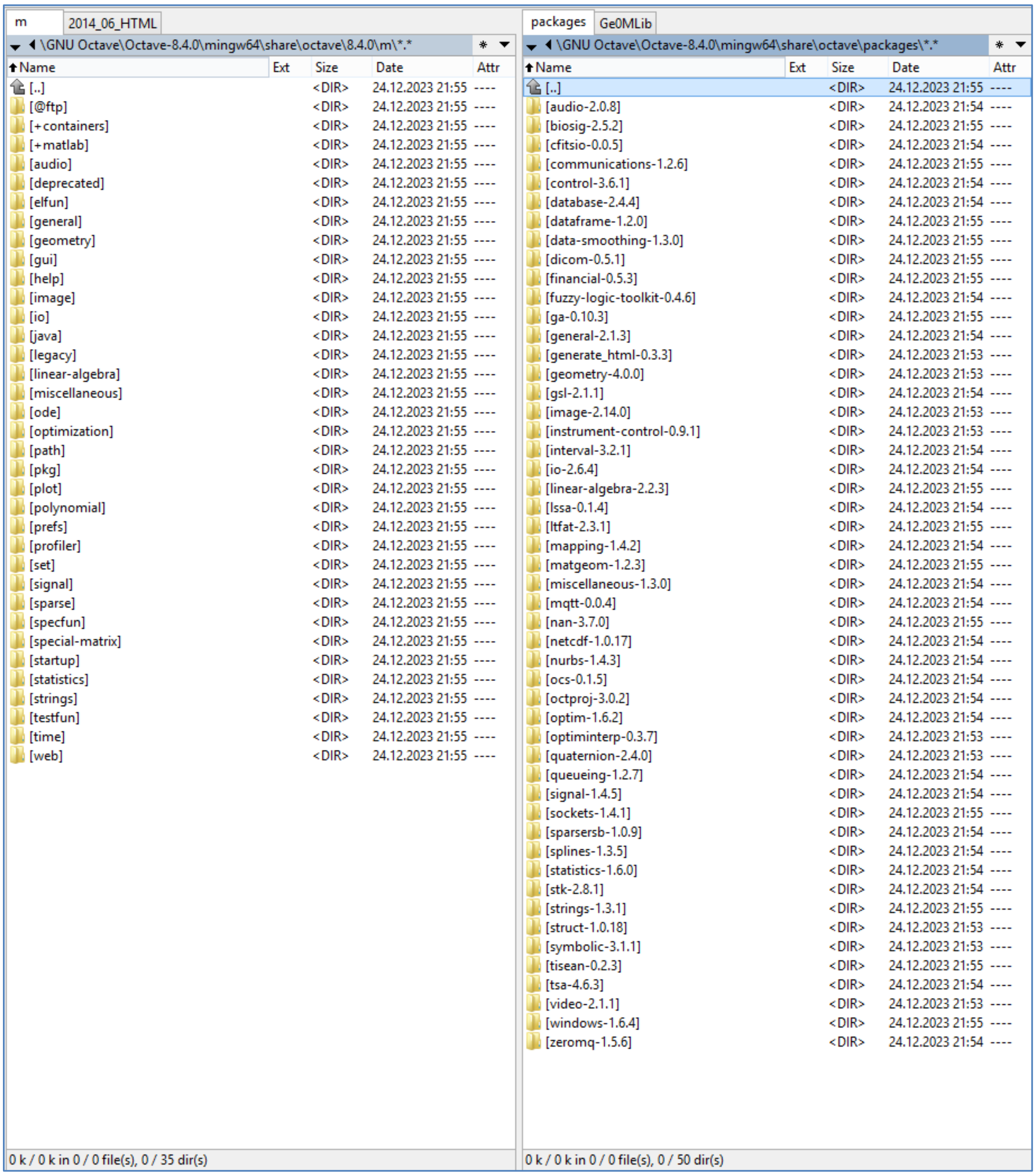
strings – Additional functions for manipulation and analysis of strings.

struct – Additional structure manipulation functions.

In order for Octave to “see” the functions of a package, the path to it must be “*registered in the system*”. In the Octave root folder, there are two folders with packages:

1) c:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\ – the folder contains "system-integrated" packages such as signal and statistics;
2) c:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\packages\ – the folder contains non-integrated packages, the paths for which were not initially *registered* in the system.

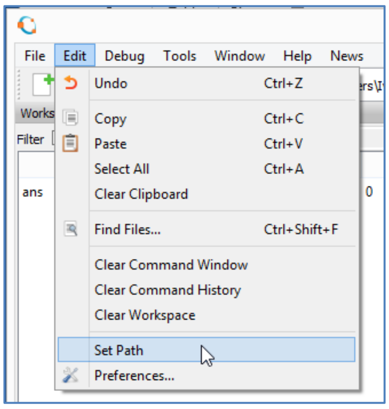
It is highly likely that the required package is located in these folders. Then you won't need to download it from the Internet.

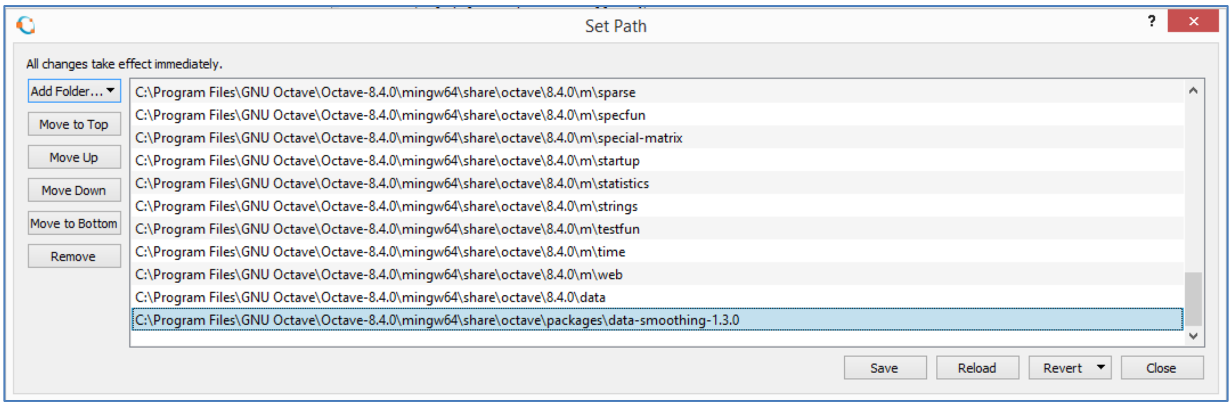


```
m   2014_06_HTML
\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\*.*
Name                 Ext  Size   Date              Attr
[..]                      <DIR>  24.12.2023 21:55  ----
[@ftp]                    <DIR>  24.12.2023 21:55  ----
[+containers]             <DIR>  24.12.2023 21:55  ----
[+matlab]                 <DIR>  24.12.2023 21:55  ----
[audio]                   <DIR>  24.12.2023 21:55  ----
[deprecated]              <DIR>  24.12.2023 21:55  ----
[elfun]                   <DIR>  24.12.2023 21:55  ----
[general]                 <DIR>  24.12.2023 21:55  ----
[geometry]                <DIR>  24.12.2023 21:55  ----
[gui]                     <DIR>  24.12.2023 21:55  ----
[help]                    <DIR>  24.12.2023 21:55  ----
[image]                   <DIR>  24.12.2023 21:55  ----
[io]                      <DIR>  24.12.2023 21:55  ----
[java]                    <DIR>  24.12.2023 21:55  ----
[legacy]                  <DIR>  24.12.2023 21:55  ----
[linear-algebra]          <DIR>  24.12.2023 21:55  ----
[miscellaneous]           <DIR>  24.12.2023 21:55  ----
[ode]                     <DIR>  24.12.2023 21:55  ----
[optimization]            <DIR>  24.12.2023 21:55  ----
[path]                    <DIR>  24.12.2023 21:55  ----
[pkg]                     <DIR>  24.12.2023 21:55  ----
[plot]                    <DIR>  24.12.2023 21:55  ----
[polynomial]              <DIR>  24.12.2023 21:55  ----
[prefs]                   <DIR>  24.12.2023 21:55  ----
[profiler]                <DIR>  24.12.2023 21:55  ----
[set]                     <DIR>  24.12.2023 21:55  ----
[signal]                  <DIR>  24.12.2023 21:55  ----
[sparse]                  <DIR>  24.12.2023 21:55  ----
[specfun]                 <DIR>  24.12.2023 21:55  ----
[special-matrix]          <DIR>  24.12.2023 21:55  ----
[startup]                 <DIR>  24.12.2023 21:55  ----
[statistics]              <DIR>  24.12.2023 21:55  ----
[strings]                 <DIR>  24.12.2023 21:55  ----
[testfun]                 <DIR>  24.12.2023 21:55  ----
[time]                    <DIR>  24.12.2023 21:55  ----
[web]                     <DIR>  24.12.2023 21:55  ----
0 k / 0 k in 0 / 0 file(s), 0 / 35 dir(s)
```

```
packages   Ge0MLib
\GNU Octave\Octave-8.4.0\mingw64\share\octave\packages\*.*
Name                          Ext  Size   Date              Attr
[..]                               <DIR>  24.12.2023 21:55  ----
[audio-2.0.8]                      <DIR>  24.12.2023 21:54  ----
[biosig-2.5.2]                     <DIR>  24.12.2023 21:55  ----
[cfitsio-0.0.5]                    <DIR>  24.12.2023 21:54  ----
[communications-1.2.6]             <DIR>  24.12.2023 21:55  ----
[control-3.6.1]                    <DIR>  24.12.2023 21:54  ----
[database-2.4.4]                   <DIR>  24.12.2023 21:54  ----
[dataframe-1.2.0]                  <DIR>  24.12.2023 21:55  ----
[data-smoothing-1.3.0]             <DIR>  24.12.2023 21:55  ----
[dicom-0.5.1]                      <DIR>  24.12.2023 21:55  ----
[financial-0.5.3]                  <DIR>  24.12.2023 21:55  ----
[fuzzy-logic-toolkit-0.4.6]        <DIR>  24.12.2023 21:54  ----
[ga-0.10.3]                        <DIR>  24.12.2023 21:55  ----
[general-2.1.3]                    <DIR>  24.12.2023 21:54  ----
[generate_html-0.3.3]              <DIR>  24.12.2023 21:53  ----
[geometry-4.0.0]                   <DIR>  24.12.2023 21:53  ----
[gsl-2.1.1]                        <DIR>  24.12.2023 21:54  ----
[image-2.14.0]                     <DIR>  24.12.2023 21:53  ----
[instrument-control-0.9.1]         <DIR>  24.12.2023 21:53  ----
[interval-3.2.1]                   <DIR>  24.12.2023 21:54  ----
[io-2.6.4]                         <DIR>  24.12.2023 21:54  ----
[linear-algebra-2.2.3]             <DIR>  24.12.2023 21:55  ----
[lssa-0.1.4]                       <DIR>  24.12.2023 21:54  ----
[ltfat-2.3.1]                      <DIR>  24.12.2023 21:55  ----
[mapping-1.4.2]                    <DIR>  24.12.2023 21:54  ----
[matgeom-1.2.3]                    <DIR>  24.12.2023 21:55  ----
[miscellaneous-1.3.0]              <DIR>  24.12.2023 21:54  ----
[mqtt-0.0.4]                       <DIR>  24.12.2023 21:54  ----
[nan-3.7.0]                        <DIR>  24.12.2023 21:55  ----
[netcdf-1.0.17]                    <DIR>  24.12.2023 21:54  ----
[nurbs-1.4.3]                      <DIR>  24.12.2023 21:54  ----
[ocs-0.1.5]                        <DIR>  24.12.2023 21:54  ----
[octproj-3.0.2]                    <DIR>  24.12.2023 21:54  ----
[optim-1.6.2]                      <DIR>  24.12.2023 21:54  ----
[optiminterp-0.3.7]                <DIR>  24.12.2023 21:53  ----
[quaternion-2.4.0]                 <DIR>  24.12.2023 21:53  ----
[queueing-1.2.7]                   <DIR>  24.12.2023 21:54  ----
[signal-1.4.5]                     <DIR>  24.12.2023 21:54  ----
[sockets-1.4.1]                    <DIR>  24.12.2023 21:55  ----
[sparsersb-1.0.9]                  <DIR>  24.12.2023 21:54  ----
[splines-1.3.5]                    <DIR>  24.12.2023 21:54  ----
[statistics-1.6.0]                 <DIR>  24.12.2023 21:54  ----
[stk-2.8.1]                        <DIR>  24.12.2023 21:54  ----
[strings-1.3.1]                    <DIR>  24.12.2023 21:55  ----
[struct-1.0.18]                    <DIR>  24.12.2023 21:53  ----
[symbolic-3.1.1]                   <DIR>  24.12.2023 21:53  ----
[tisean-0.2.3]                     <DIR>  24.12.2023 21:55  ----
[tsa-4.6.3]                        <DIR>  24.12.2023 21:54  ----
[video-2.1.1]                      <DIR>  24.12.2023 21:53  ----
[windows-1.6.4]                    <DIR>  24.12.2023 21:55  ----
[zeromq-1.5.6]                     <DIR>  24.12.2023 21:54  ----
0 k / 0 k in 0 / 0 file(s), 0 / 50 dir(s)
```

You can see which paths are already *registered* in Octave using the Path Browser (the paths to all packages, in the “m\” folder, are *registered*). Using the “Add Folder” button you can add paths to additional packages, after which you need to click on the “Save” button to save the list of paths, and it will be automatically loaded on subsequent Octave starts.

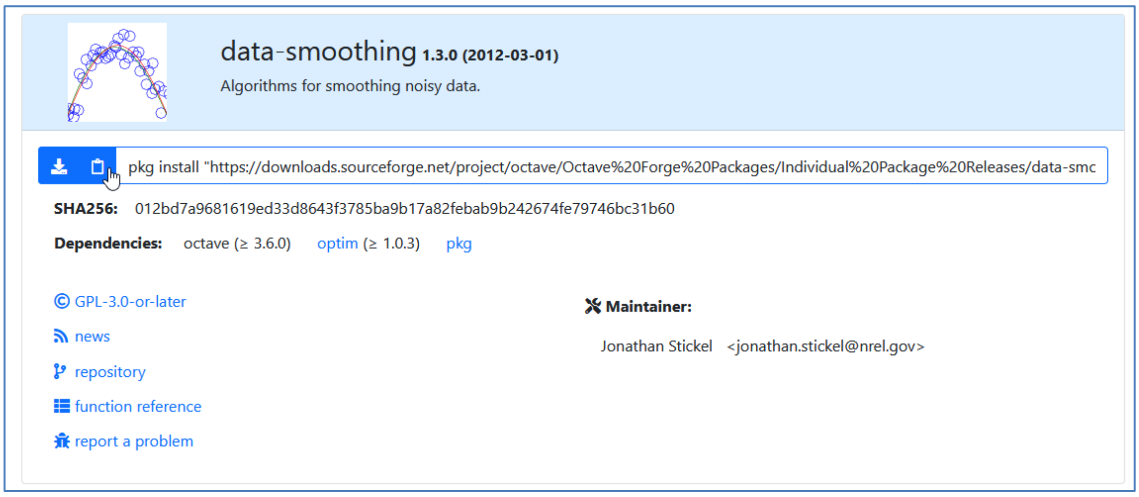




All changes take effect immediately.

```
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\sparse
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\specfun
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\special-matrix
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\startup
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\statistics
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\strings
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\testfun
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\time
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\web
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\data
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\packages\data-smoothing-1.3.0
```

If the required package is not in the above folders, you need to find it on the Internet. For example, when we go to the "data-smooth package" page, we see the window shown below. Copying the “pkg install” command and path into the Octave command window, and then running it, should install the data-smooth package directly from the Internet. You can also download the package and use the “pkg install” command (with the path specified) to install it offline. You may need to additionally *register* the path to the package after installation.



data-smoothing 1.3.0 (2012-03-01)

Algorithms for smoothing noisy data.

```
pkg install "https://downloads.sourceforge.net/project/octave/Octave%20Forge%20Packages/Individual%20Package%20Releases/data-smc
```

SHA256: 012bd7a9681619ed33d8643f3785ba9b17a82febab9b242674fe79746bc31b60

Dependencies: octave (≥ 3.6.0) optim (≥ 1.0.3) pkg

GPL-3.0-or-later

news

repository

function reference

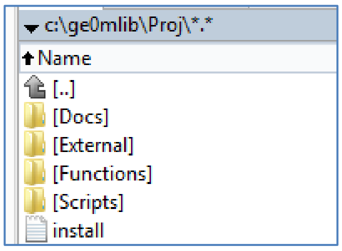
report a problem

Maintainer:

Jonathan Stickel <jonathan.stickel@nrel.gov>

2 ge0mlib installation (MatLab, GNU Octave)

You can download the ge0mlib library from the link https://ge0mlib.com/g/ge0mlib.zip, and then unpack the archive to a convenient place. The archive contains the folders shown in the figure below.



c:\ge0mlib\Proj\*.*

Name

[..]

[Docs]

[External]

[Functions]

[Scripts]

install

The purpose of the folders is as follows:

-- Docs – contains documentation on working with the library (pdf format), as well as examples of scripts (m-files and descriptions in pdf format; video materials are present on the site, but are not included in the archive with the library);

-- External – folder for additional libraries that are used or planned to be used with ge0mlib. Currently it is m_map, mksqlite-2.11-win64, n_vector;

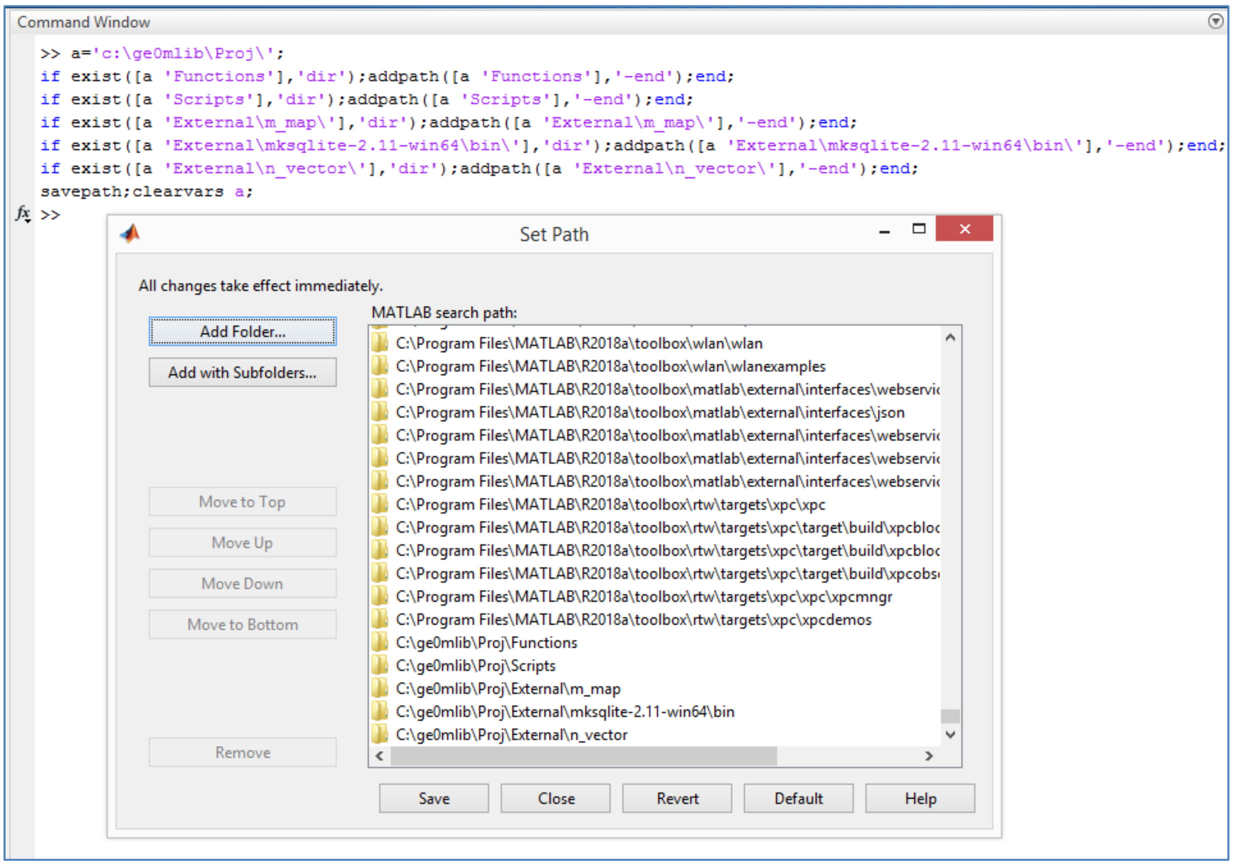
-- Functions – functions of the ge0mlib library (the contents of the folder are available on Git Hub: https://github.com/ge0mlib/ge0mlib);

-- Scripts – an empty folder for scripts, for local use. Here you need to put the m-file with a script that needs to be executed to “process the data”. If you later write your own script or function, you can also place them in this folder;

-- install.txt – a file with commands for *registering* paths (instead of using Path Browser).

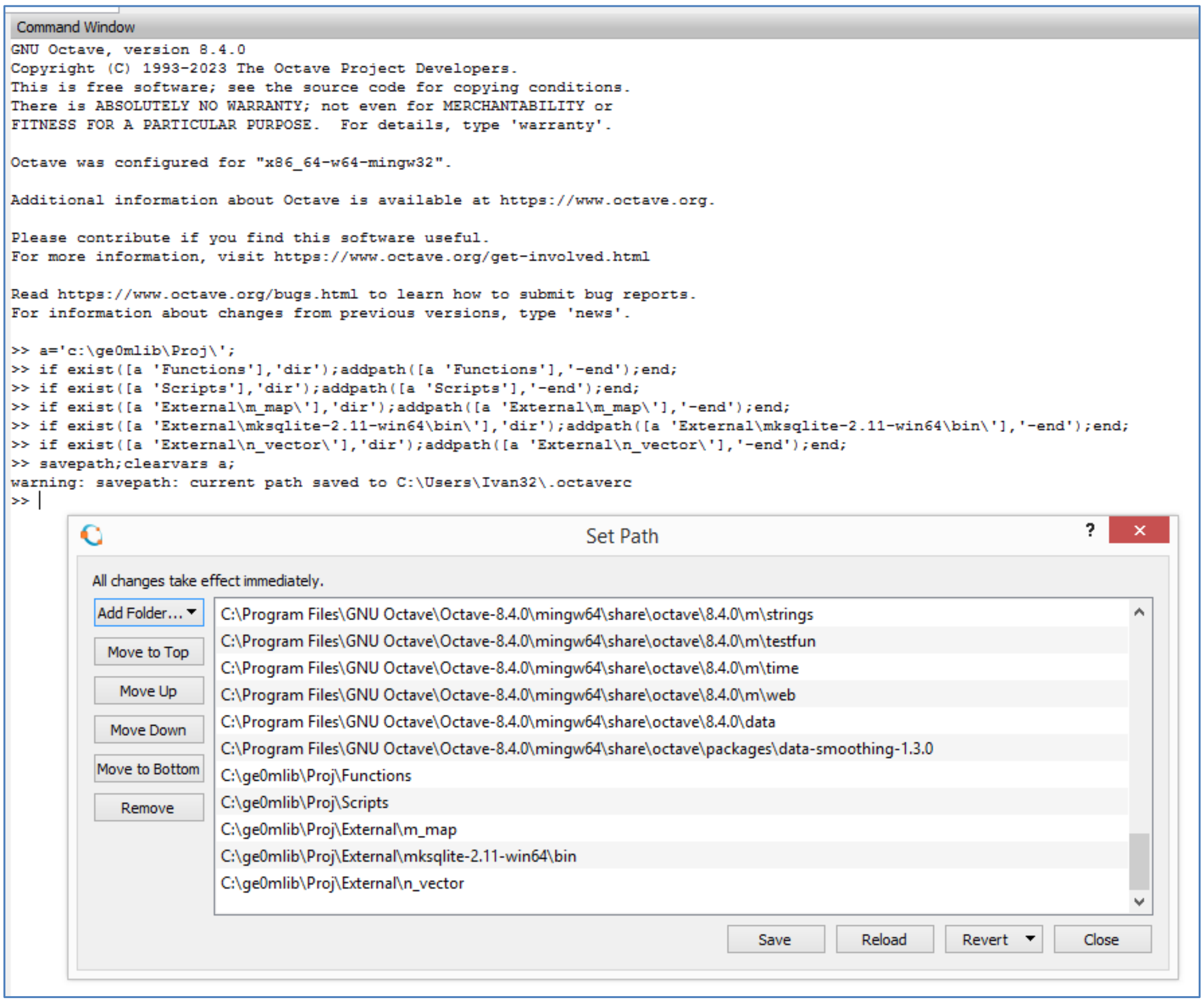
To install ge0mlib, you can *register* the path to the m-files manually using the Path Browser or write the path to the ge0mlib root folder in the first line of install.txt, then copy the text from the file into the command window and press Enter (to execute the entered commands). An example of such setting paths for the root folder 'c:\ge0mlib\Proj\' is shown in the figures below. When executing commands, the presence of the corresponding subfolders in External, as well as the Functions and Scripts folders, will be checked. If there are corresponding folders on the disk, they will be registered in Path and saved. If there are no folders, then the paths will not be written (for example, if you delete the External folder in advance, then the paths to the deleted additional libraries that were contained in External will not be written to the Path file).

It should be noted that the ge0mlib library was originally written for MatLab, and after that the functions were “adapted” for Octave. The purpose of this adaptation is to use free software that can be installed on any computer without purchasing or manipulating a license (QGIS and Python are good examples of such programs). At the same time, a small part of ge0mlib functions (using MatLab toolboxes) may not work in Octave (at the moment, when developing functions, compatibility with MatLab is primarily taken into account).



```
>> a='c:\ge0mlib\Proj\';
if exist([a 'Functions'],'dir');addpath([a 'Functions'],'-end');end;
if exist([a 'Scripts'],'dir');addpath([a 'Scripts'],'-end');end;
if exist([a 'External\m_map\'],'dir');addpath([a 'External\m_map\'],'-end');end;
if exist([a 'External\mksqlite-2.11-win64\bin\'],'dir');addpath([a 'External\mksqlite-2.11-win64\bin\'],'-end');end;
if exist([a 'External\n_vector\'],'dir');addpath([a 'External\n_vector\'],'-end');end;
savepath;clearvars a;
>>
```

```
C:\Program Files\MATLAB\R2018a\toolbox\wlan\wlan
C:\Program Files\MATLAB\R2018a\toolbox\wlan\wlanexamples
C:\Program Files\MATLAB\R2018a\toolbox\matlab\external\interfaces\webservic
C:\Program Files\MATLAB\R2018a\toolbox\matlab\external\interfaces\json
C:\Program Files\MATLAB\R2018a\toolbox\matlab\external\interfaces\webservic
C:\Program Files\MATLAB\R2018a\toolbox\matlab\external\interfaces\webservic
C:\Program Files\MATLAB\R2018a\toolbox\matlab\external\interfaces\webservic
C:\Program Files\MATLAB\R2018a\toolbox\rtw\targets\xpc\xpc
C:\Program Files\MATLAB\R2018a\toolbox\rtw\targets\xpc\target\build\xpcbloc
C:\Program Files\MATLAB\R2018a\toolbox\rtw\targets\xpc\target\build\xpcbloc
C:\Program Files\MATLAB\R2018a\toolbox\rtw\targets\xpc\target\build\xpcobs
C:\Program Files\MATLAB\R2018a\toolbox\rtw\targets\xpc\xpc\xpcmngr
C:\Program Files\MATLAB\R2018a\toolbox\rtw\targets\xpc\xpcdemos
C:\ge0mlib\Proj\Functions
C:\ge0mlib\Proj\Scripts
C:\ge0mlib\Proj\External\m_map
C:\ge0mlib\Proj\External\mksqlite-2.11-win64\bin
C:\ge0mlib\Proj\External\n_vector
```



```
GNU Octave, version 8.4.0
Copyright (C) 1993-2023 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

>> a='c:\ge0mlib\Proj\';
>> if exist([a 'Functions'],'dir');addpath([a 'Functions'],'-end');end;
>> if exist([a 'Scripts'],'dir');addpath([a 'Scripts'],'-end');end;
>> if exist([a 'External\m_map\'],'dir');addpath([a 'External\m_map\'],'-end');end;
>> if exist([a 'External\mksqlite-2.11-win64\bin\'],'dir');addpath([a 'External\mksqlite-2.11-win64\bin\'],'-end');end;
>> if exist([a 'External\n_vector\'],'dir');addpath([a 'External\n_vector\'],'-end');end;
>> savepath;clearvars a;
warning: savepath: current path saved to C:\Users\Ivan32\.octaverc
>> |
```

```
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\strings
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\testfun
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\time
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\m\web
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\8.4.0\data
C:\Program Files\GNU Octave\Octave-8.4.0\mingw64\share\octave\packages\data-smoothing-1.3.0
C:\ge0mlib\Proj\Functions
C:\ge0mlib\Proj\Scripts
C:\ge0mlib\Proj\External\m_map
C:\ge0mlib\Proj\External\mksqlite-2.11-win64\bin
C:\ge0mlib\Proj\External\n_vector
```

3 Git

3.1 Purposes of using Git and Github

The contents of the Functions folder (ge0mlib library functions) use Git version control. The ge0mlib library is available on Git Hub: https://github.com/ge0mlib/ge0mlib. The purpose of using Git, discussed below, is to restore "snapshots" of a library made in the past and with a 90% chance you will not need this information.

Since the functions in the library are gradually changing, the "old script" may not run with the "new version of the library". To run such scripts, it is convenient to be able to select a "snapshot" of the library while writing the script. To do this, in the last line of the script code, the date of its develop must be indicated. Using Git, you can select a "snapshot" (commit) prior to the date the script was created and run it without having to make corrections to the code. The same applies to example scripts in the Docs folder written after 12/26/2023.

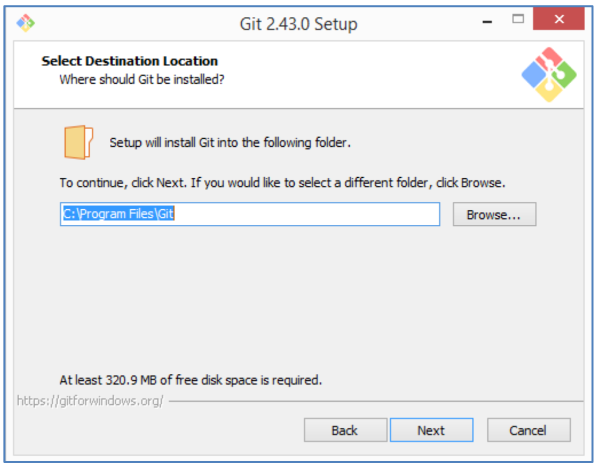
The described solution has its drawbacks - of course, the old version of the library will contain all the old bugs (which is a definite disadvantage), but there will be no new bugs (which is some advantage). We assume that the script, written and tested *at one time*, produced adequate calculation results that were not subject to bugs, and therefore can be used in the future. An alternative solution is to edit the script, making changes to the command syntax (it is assumed that it has changed) for the new version of the library. This is positively the best solution, but it requires time, knowledge, and testing how the modified script works.

The main purpose of using a tool like Git is collaborative development. The library is free and open to any changes. If you want to rewrite or add some functions, you can do this in a separate git branch, which you can later upload to Git Hub and merge with the main git branch, including the functions in the library. However, if you don't use Git, you can simply send the m-files and their description to mail@ge0mlib.com, I will add them to Git Hub, documentation and library.

3.2 Git installation

The Git distribution is downloaded from https://git-scm.com/downloads. At the time of writing the file is Git-2.43.0-64-bit.exe (for Win64). The git installation runs with default settings except
(1) changing the editor to Nano or another convenient one (third screenshot),
(2) checking the box to launch Git Bash in the last screenshot.

Both these actions are not critical, since after installing git you can independently replace the editor in the git settings and run git itself.



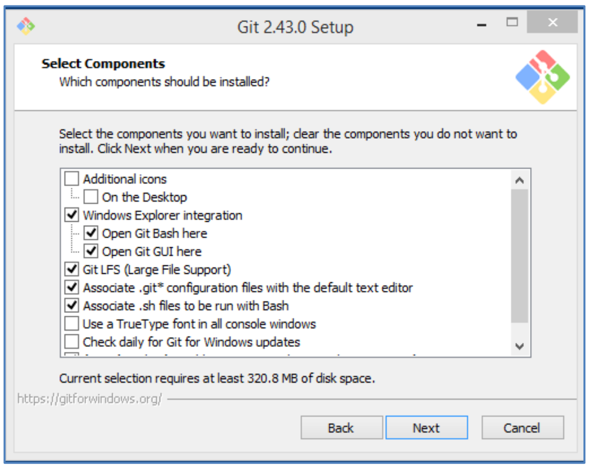
Select Destination Location
Where should Git be installed?

Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 320.9 MB of free disk space is required.

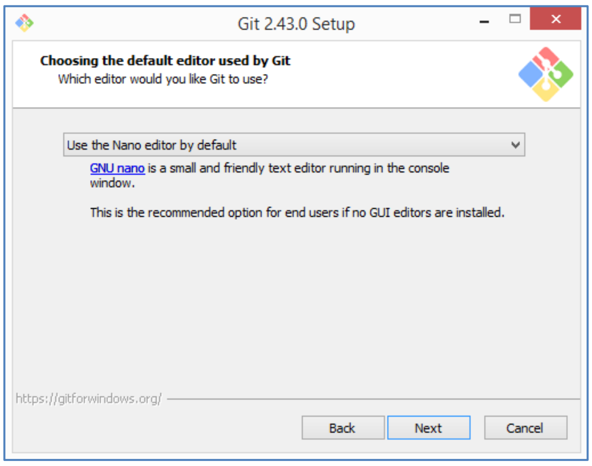


Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] Additional icons
  - [ ] On the Desktop
- [✓] Windows Explorer integration
  - [✓] Open Git Bash here
  - [✓] Open Git GUI here
- [✓] Git LFS (Large File Support)
- [✓] Associate .git* configuration files with the default text editor
- [✓] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows
- [ ] Check daily for Git for Windows updates

Current selection requires at least 320.8 MB of disk space.

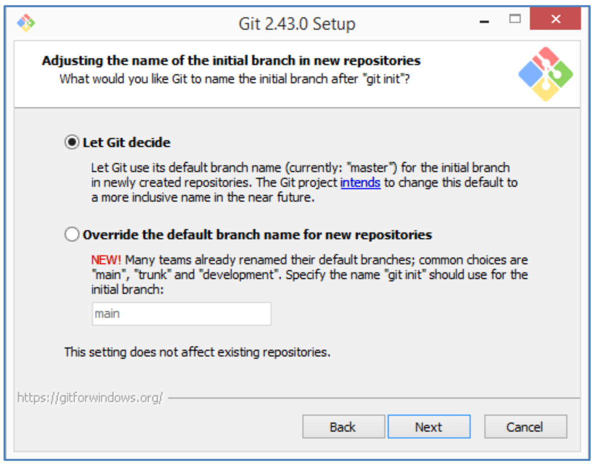


Choosing the default editor used by Git
Which editor would you like Git to use?

Use the Nano editor by default

GNU nano is a small and friendly text editor running in the console window.

This is the recommended option for end users if no GUI editors are installed.



Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?

● **Let Git decide**

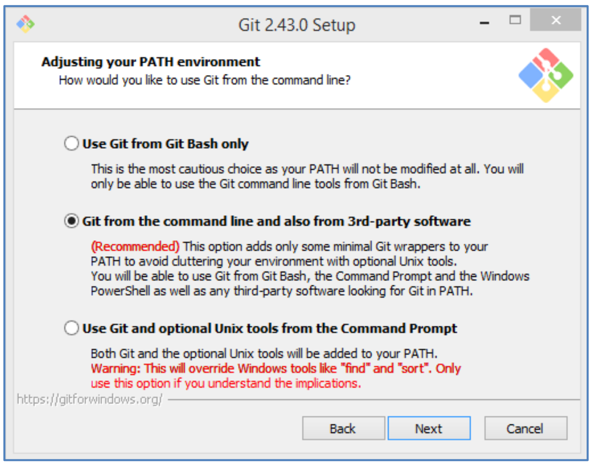
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project intends to change this default to a more inclusive name in the near future.

○ **Override the default branch name for new repositories**

NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

main

This setting does not affect existing repositories.



Adjusting your PATH environment
How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

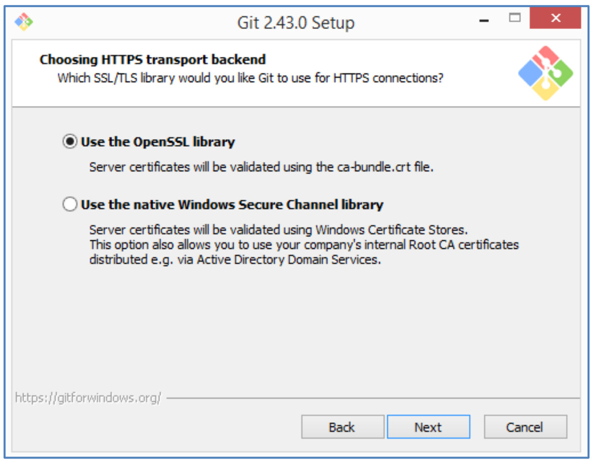
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

● **Git from the command line and also from 3rd-party software**

(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

○ **Use Git and optional Unix tools from the Command Prompt**

Both Git and the optional Unix tools will be added to your PATH. Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.



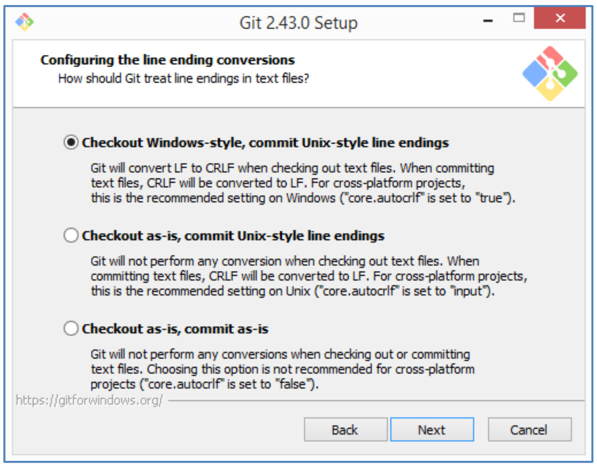
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?

● **Use the OpenSSL library**

Server certificates will be validated using the ca-bundle.crt file.

○ **Use the native Windows Secure Channel library**

Server certificates will be validated using Windows Certificate Stores. This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.



Configuring the line ending conversions
How should Git treat line endings in text files?

● **Checkout Windows-style, commit Unix-style line endings**

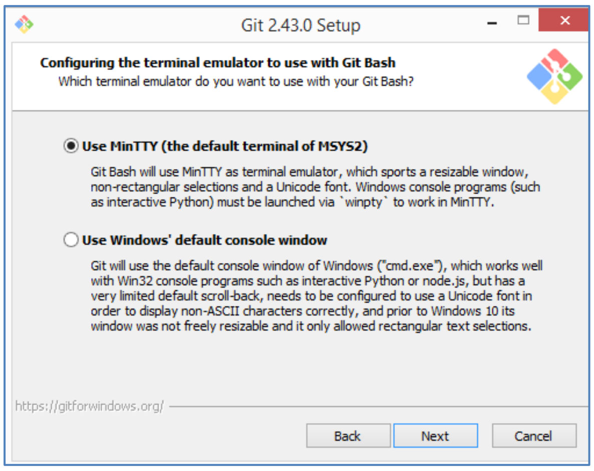
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

○ **Checkout as-is, commit Unix-style line endings**

Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

○ **Checkout as-is, commit as-is**

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").



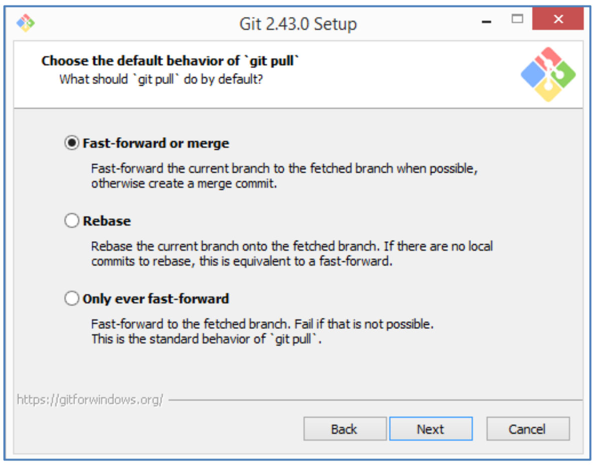
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

● **Use MinTTY (the default terminal of MSYS2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

○ **Use Windows' default console window**

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

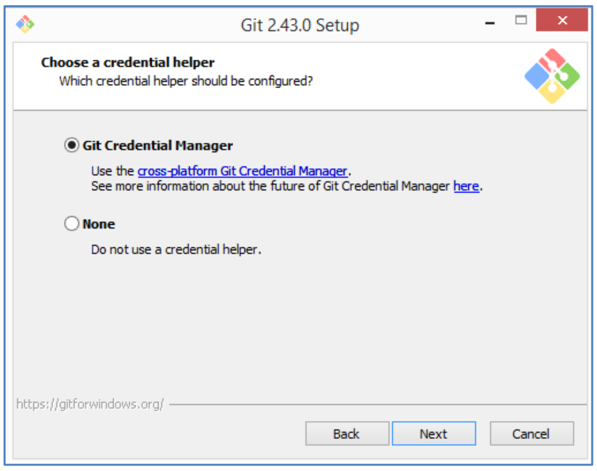


Choose the default behavior of `git pull`
What should `git pull` do by default?

[✓ **Fast-forward or merge**]
Fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

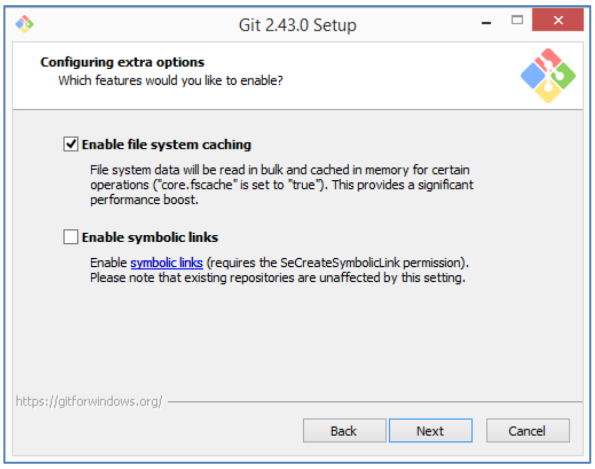
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible. This is the standard behavior of `git pull`.



Choose a credential helper
Which credential helper should be configured?

[✓ **Git Credential Manager**]
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.

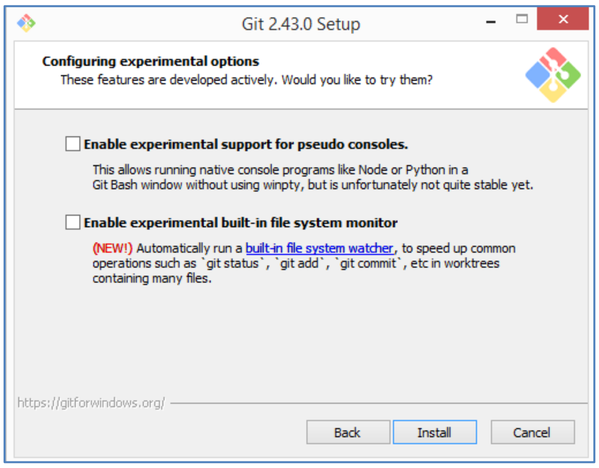
None
Do not use a credential helper.



Configuring extra options
Which features would you like to enable?

[✓] **Enable file system caching**
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

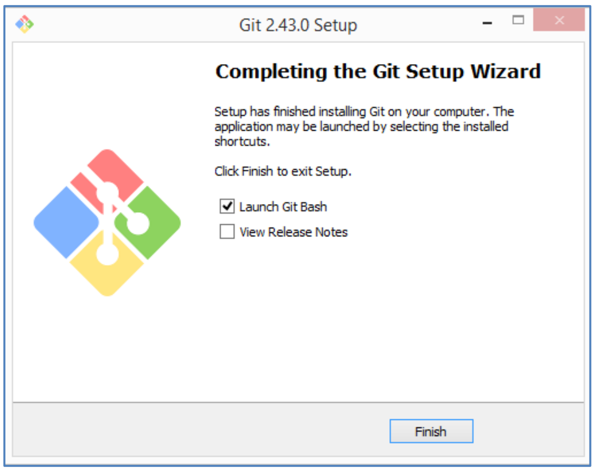
[ ] **Enable symbolic links**
Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.



Configuring experimental options
These features are developed actively. Would you like to try them?

[ ] **Enable experimental support for pseudo consoles.**
This allows running native console programs like Node or Python in a Git Bash window without using winpty, but is unfortunately not quite stable yet.

[ ] **Enable experimental built-in file system monitor**
(NEW!) Automatically run a built-in file system watcher, to speed up common operations such as `git status`, `git add`, `git commit`, etc in worktrees containing many files.

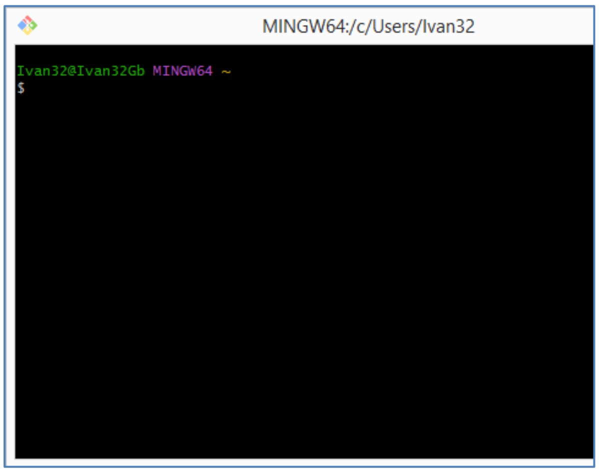


Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

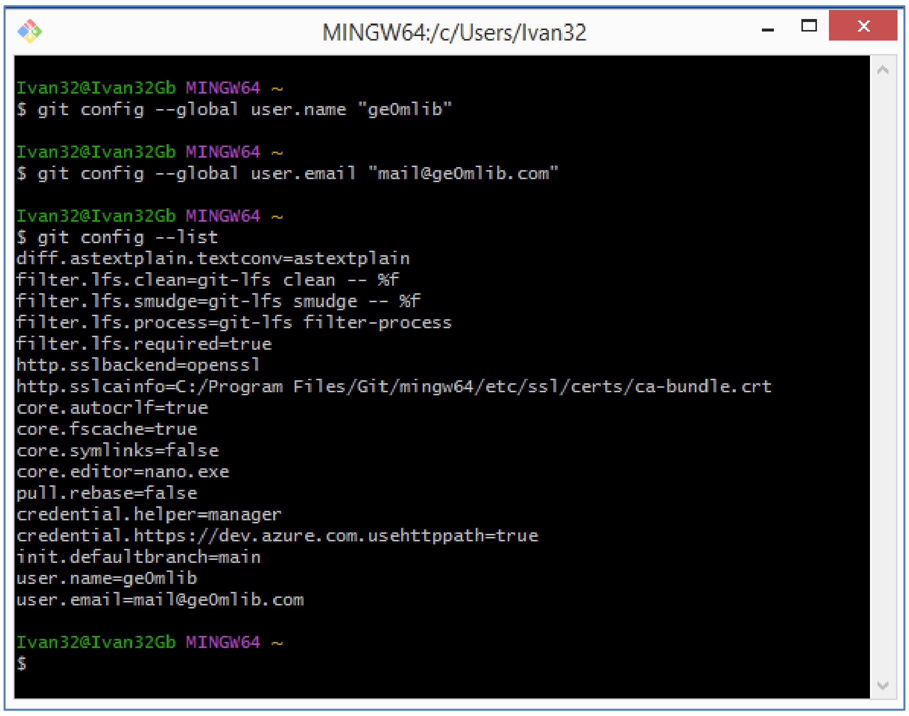
[✓] Launch Git Bash
[ ] View Release Notes



```
Ivan32@Ivan32Gb MINGW64 ~
$
```

After installation, you need to run the commands below in the git window, by adding your user.name and user.email. Commands can be copied and pasted using the right mouse button.

```
git config --global user.name "xxx"
git config --global user.email "xxxx@xxx.xxx"
git config --list
exit
```



```
MINGW64:/c/Users/Ivan32

Ivan32@Ivan32Gb MINGW64 ~
$ git config --global user.name "ge0mlib"

Ivan32@Ivan32Gb MINGW64 ~
$ git config --global user.email "mail@ge0mlib.com"

Ivan32@Ivan32Gb MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
core.editor=nano.exe
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
user.name=ge0mlib
user.email=mail@ge0mlib.com

Ivan32@Ivan32Gb MINGW64 ~
$
```

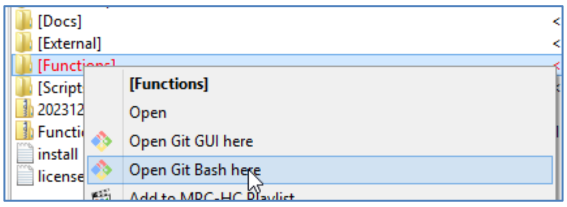
After executing the exit command, the git window will close.

3.3 Creating a git project and downloading ge0mlib from the repository

Before working with git, it is better to delete all files from the Functions folder to avoid conflicts between file versions. In the first step, you can run git and go to the Functions folder using the cd command, for example:

cd c:/ge0mlib/Proj/Functions

As an alternative, you can open git directly in the required folder by right-clicking.



After running git, you need to enter the following sequence of commands:

-- creating an invisible .git folder containing git information

git init

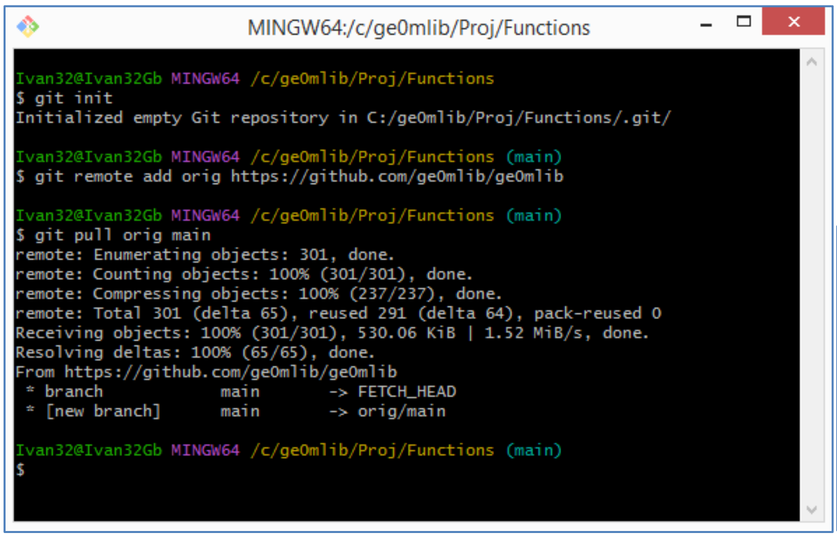
-- adding the path to the repository on Git Hub

git remote add orig https://github.com/ge0mlib/ge0mlib

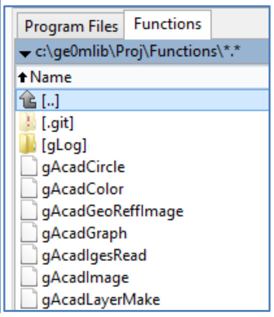
-- copying repository files to the .git folder

git pull orig main

An example of the git window and the contents of the Functions folder after executing the commands is shown below.



```
Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions
$ git init
Initialized empty Git repository in C:/ge0mlib/Proj/Functions/.git/

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (main)
$ git remote add orig https://github.com/ge0mlib/ge0mlib

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (main)
$ git pull orig main
remote: Enumerating objects: 301, done.
remote: Counting objects: 100% (301/301), done.
remote: Compressing objects: 100% (237/237), done.
remote: Total 301 (delta 65), reused 291 (delta 64), pack-reused 0
Receiving objects: 100% (301/301), 530.06 KiB | 1.52 MiB/s, done.
Resolving deltas: 100% (65/65), done.
From https://github.com/ge0mlib/ge0mlib
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> orig/main

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (main)
$
```

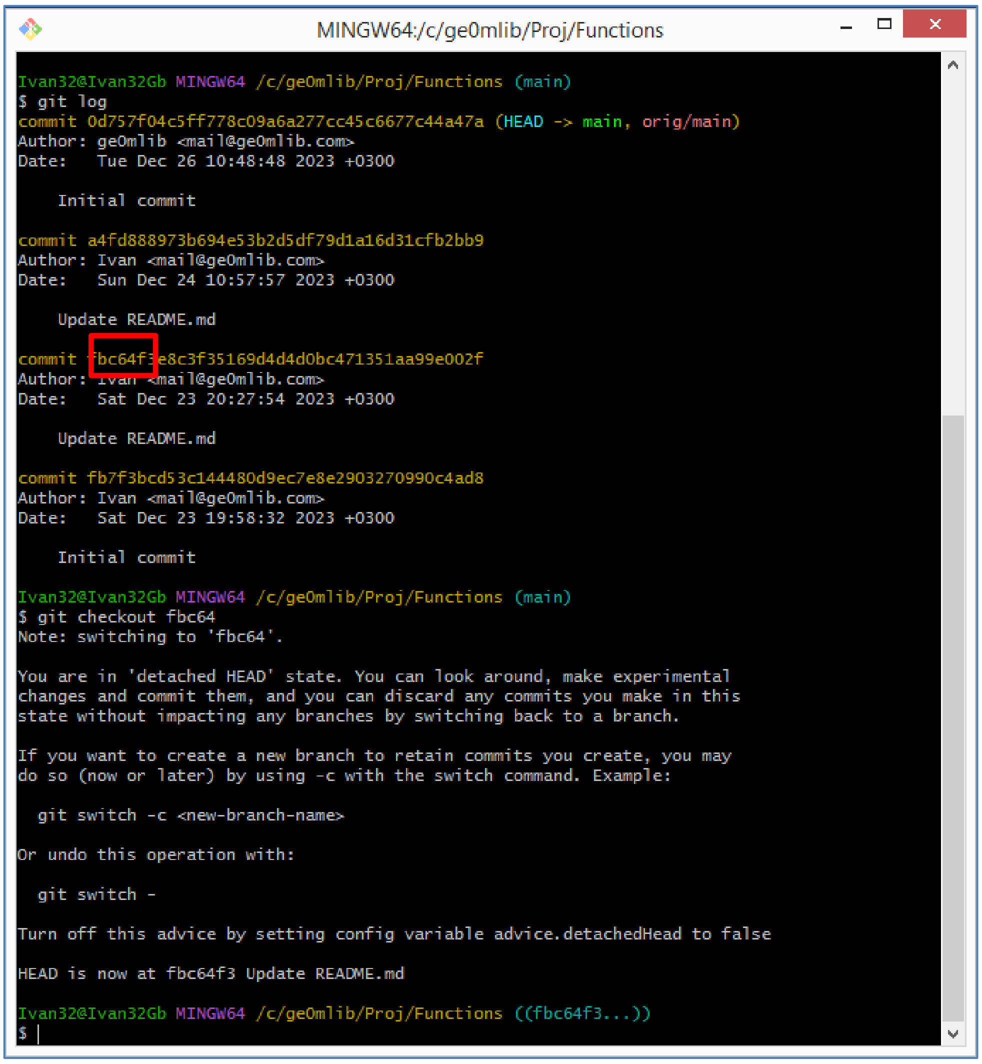


3.4 Navigating history in Git

To list the “library snapshots” (commit), you need to use the command

git log

use the “q” key to exit. When the command is executed, git displays a list of “snapshots” (commits) with information about the date and a unique hash code (name) for each commit.

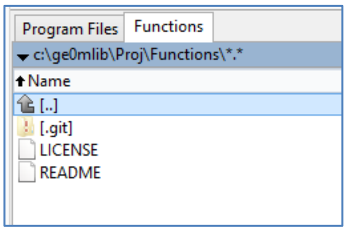


```
Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (main)
$ git log
commit 0d757f04c5ff778c09a6a277cc45c6677c44a47a (HEAD -> main, orig/main)
Author: ge0mlib <mail@ge0mlib.com>
Date:   Tue Dec 26 10:48:48 2023 +0300

    Initial commit

commit a4fd888973b694e53b2d5df79d1a16d31cfb2bb9
Author: Ivan <mail@ge0mlib.com>
Date:   Sun Dec 24 10:57:57 2023 +0300

    Update README.md

commit fbc64f3e8c3f35169d4d4d0bc471351aa99e002f
Author: Ivan <mail@ge0mlib.com>
Date:   Sat Dec 23 20:27:54 2023 +0300

    Update README.md

commit fb7f3bcd53c144480d9ec7e8e2903270990c4ad8
Author: Ivan <mail@ge0mlib.com>
Date:   Sat Dec 23 19:58:32 2023 +0300

    Initial commit

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (main)
$ git checkout fbc64
Note: switching to 'fbc64'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at fbc64f3 Update README.md

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions ((fbc64f3...))
$ |
```

You need to select commit by date (so that the date is the closest date before the script date) and use the hash code (in our example it is fbc64) with the "checkout" command:

git checkout fbc64

After the command is executed, the special git pointer HEAD will move to the commit with hash code fbc64 and git will change the contents of the Functions folder to match the contents on the commit date. The result of changing the contents of the Functions folder, for our example, is shown below.



After such a shift in the HEAD pointer, it is very preferably to create a new branch for this commit (for example, with the name my001) using the command

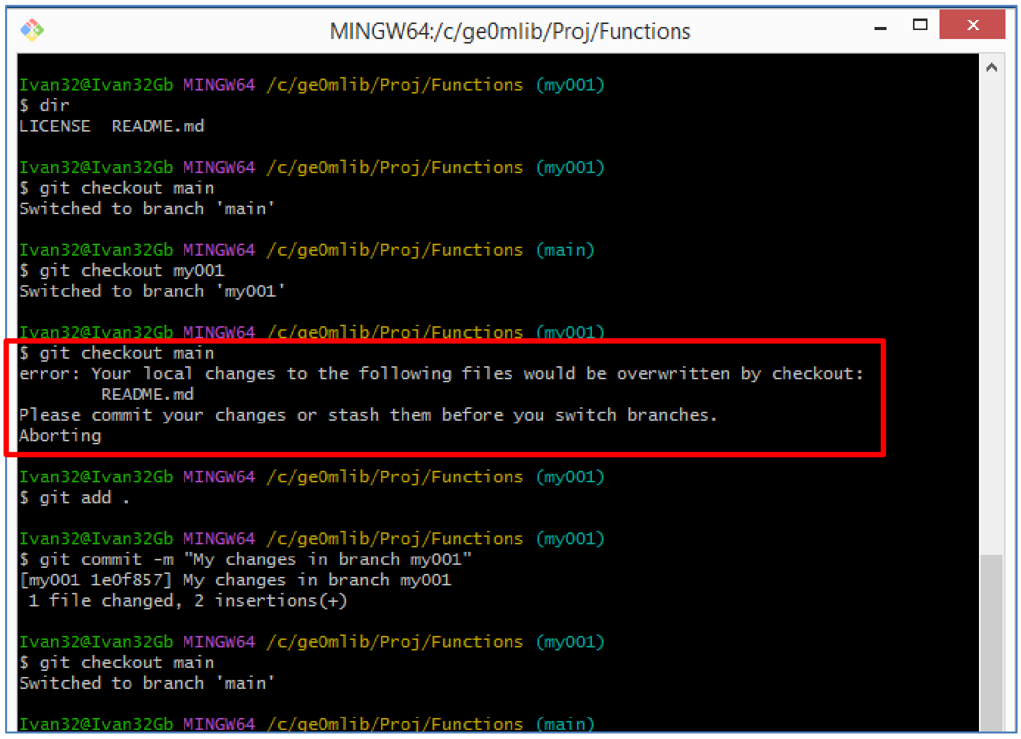
git branch my001

Now we can switch between the "latest version" of the library and the "my001 version" using the commands

git checkout main

git checkout my001

in this case, the contents of the Functions folder will be replaced each time. Strictly speaking, not all files will be replaced, but only the *tracked files* contained in the commit (with the appropriate names and hash code). If we make changes to such a *tracked file*, then when we try to switch between versions of the library, an error message will be displayed, marked with a rectangle in the screenshot below.



```
Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (my001)
$ dir
LICENSE  README.md

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (my001)
$ git checkout main
Switched to branch 'main'

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (main)
$ git checkout my001
Switched to branch 'my001'

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (my001)
$ git checkout main
error: Your local changes to the following files would be overwritten by checkout:
        README.md
Please commit your changes or stash them before you switch branches.
Aborting

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (my001)
$ git add .

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (my001)
$ git commit -m "My changes in branch my001"
[my001 1e0f857] My changes in branch my001
 1 file changed, 2 insertions(+)

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (my001)
$ git checkout main
Switched to branch 'main'

Ivan32@Ivan32Gb MINGW64 /c/ge0mlib/Proj/Functions (main)
```

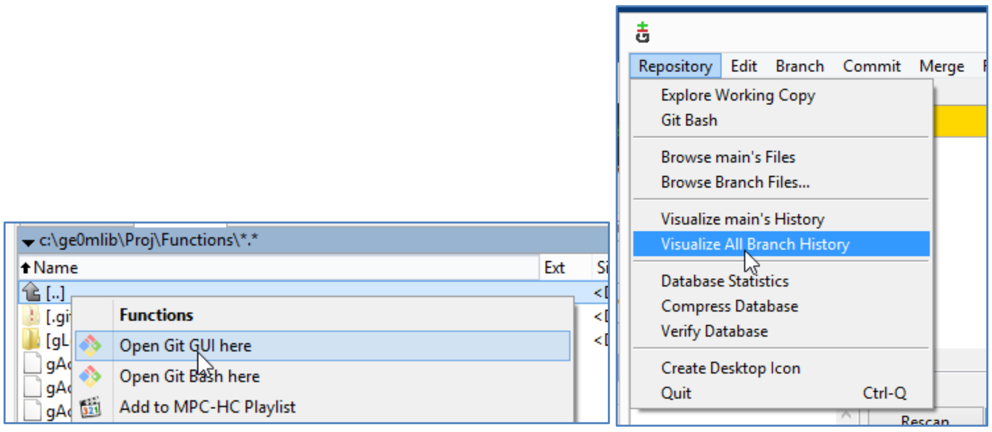
This is logical, because in the event of a switch, the changes made would be “lost” (erased). Now, in order to perform the switch, we need to create a new commit with our changes. To do this, for instance, you can run the following sequence of commands:

```
git add .
git commit -m "My changes in branch my001"
git checkout main
```

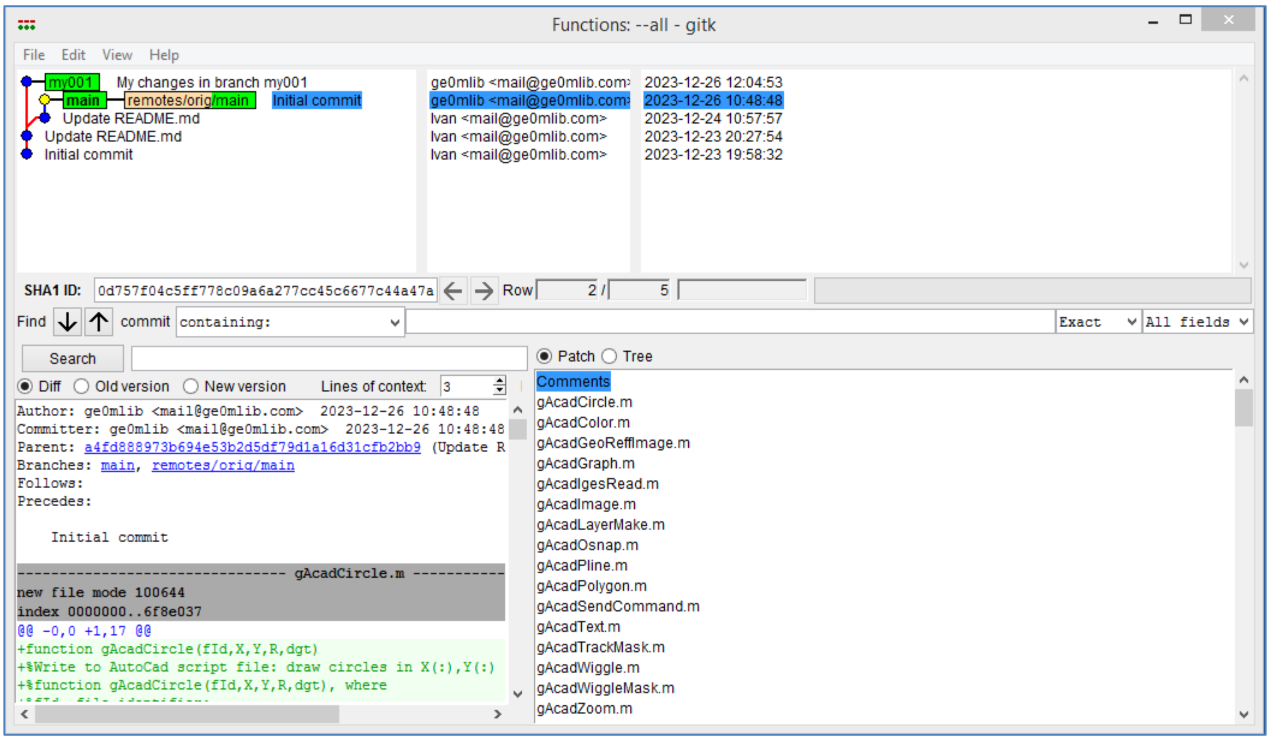
Further “deepening into the topic” requires a full study of working with git, which is beyond the scope of this document.

3.5 Launching the GUI

You can select commit by date not only in the terminal, but also using the graphical interface. To do this, by right-clicking you need to select not “Open Git Bash here”, but “Open Git GUI here” and then – Visualize All Branch History.



In the window that opens, you can see the rendering of branches, commits, and hash codes.



```
My changes in branch my001        ge0mlib <mail@ge0mlib.com>   2023-12-26 12:04:53
Initial commit                    ge0mlib <mail@ge0mlib.com>   2023-12-26 10:48:48
Update README.md                  Ivan <mail@ge0mlib.com>      2023-12-24 10:57:57
Update README.md                  Ivan <mail@ge0mlib.com>      2023-12-23 20:27:54
Initial commit                    Ivan <mail@ge0mlib.com>      2023-12-23 19:58:32

SHA1 ID: 0d757f04c5ff778c09a6a277cc45c6677c44a47a   Row 2 / 5
Lines of context: 3
```

```
Author: ge0mlib <mail@ge0mlib.com>  2023-12-26 10:48:48
Committer: ge0mlib <mail@ge0mlib.com>  2023-12-26 10:48:48
Parent: a4fd888973b694e53b2d5df79d1a16d31cfb2bb9 (Update R
Branches: main, remotes/orig/main
Follows:
Precedes:

    Initial commit

-------------------------------- gAcadCircle.m ----------
new file mode 100644
index 0000000..6f8e037
@@ -0,0 +1,17 @@
+function gAcadCircle(fId,X,Y,R,dgt)
+%Write to AutoCad script file: draw circles in X(:),Y(:)
+%function gAcadCircle(fId,X,Y,R,dgt), where
```

```
gAcadCircle.m
gAcadColor.m
gAcadGeoRefImage.m
gAcadGraph.m
gAcadIgesRead.m
gAcadImage.m
gAcadLayerMake.m
gAcadOsnap.m
gAcadPline.m
gAcadPolygon.m
gAcadSendCommand.m
gAcadText.m
gAcadTrackMask.m
gAcadWiggle.m
gAcadWiggleMask.m
gAcadZoom.m
```

4 Structure of the script and script commands

For writing scripts for the ge0mlib library, there is a certain set of rules that are best adhered to so that the use of scripts by other people is as convenient as possible for them.

Each script can execute multiple "script commands". For each script command, multiple "script command parameters" can be passed. To execute a script command, you need to enter *a command-to-execute* of the form in the MatLab or Octave window:

{'LoadMag1x8_r01','d:\002\1mag\0052_MAG_'};MagToolR01;

here:

MagToolR01 – name of the script (name of the file with the script that was copied to the Scripts folder);

{'LoadMag1x8_r01','d:\002\1mag\0052_MAG_'} – parameters of the script command, and the first parameter is always the name of the script command. In curly braces, separated by commas, many parameters of the script command that are required for its execution can be entered;

'LoadMag1x8_r01' – the name of the script command that executes a specific section of the script code;

'd:\002\1mag\0052_MAG_' – a script command parameter which, in our case, contains the path to a specific file that will be used when running the script.

Let's consider the structure of the MagToolR01 script (for recalculating magnetic survey data), which is shown in the figure below (only part of the code is shown). When creating scripts, it is advisable to adhere to this structure at all times (line numbers, of course, are indicated for a specific example).

-- Script name (line 1);

-- Brief purpose of the script (line 2)

-- List of additional libraries required to execute the script (line 2);

-- Description of script commands and, if necessary, command parameters (lines 3-7);

-- Several examples with “working sequences” of commands, written after “Example” (in our case there is only one such sequence, and it is given in line 8);

-- Blocks executed according to the conditions for each of the commands (conditions with command names are shown in lines 11, 17 and 31);

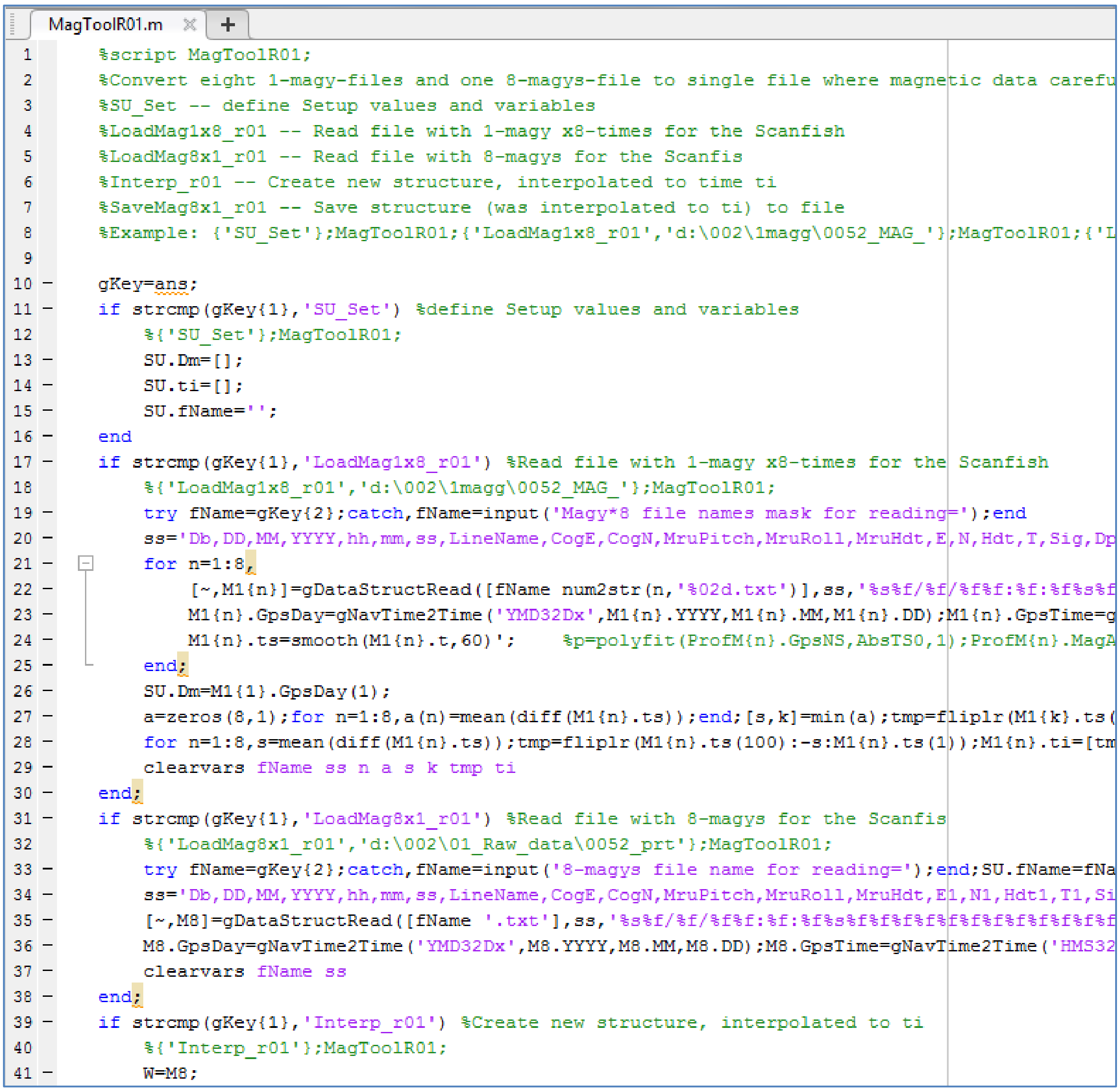
-- A brief description of the command being executed at the beginning of each block (the description is given as a comment on lines 11, 17 and 31);

-- An example of a command and script parameters for each block (examples are given as comments in lines 12, 18, 32);

-- A section of code that requests input of script parameters if they were skipped when entering the execution command. This code also contains descriptive text for each script parameter (lines 19, 33);

-- The clearvars function, executed at the end of each block (corresponding to the script command) to remove “extra/temporary” variables (lines 29, 37);

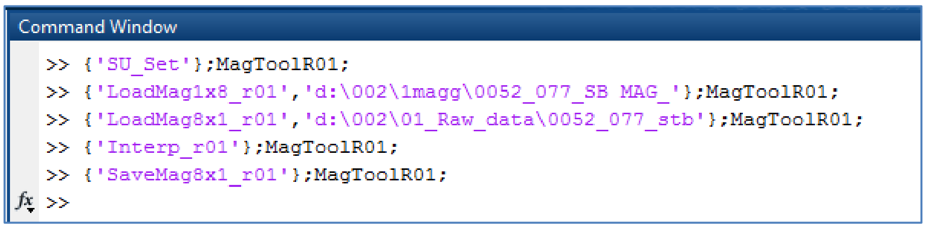
-- Date of creation of the script, email for communication, MatLab and/or Octave environment indicating the version for which the script was tested (comment in the last line 69: mail@ge0mlib.com 12/10/2023 MatLab2018b&Octave8.4.0).


MagToolR01.m

```
    %script MagToolR01;
    %Convert eight 1-magy-files and one 8-magys-file to single file where magnetic data carefu
    %SU_Set -- define Setup values and variables
    %LoadMag1x8_r01 -- Read file with 1-magy x8-times for the Scanfish
    %LoadMag8x1_r01 -- Read file with 8-magys for the Scanfis
    %Interp_r01 -- Create new structure, interpolated to time ti
    %SaveMag8x1_r01 -- Save structure (was interpolated to ti) to file
    %Example: {'SU_Set'};MagToolR01;{'LoadMag1x8_r01','d:\002\1magg\0052_MAG_'};MagToolR01;{'L

    gKey=ans;
    if strcmp(gKey{1},'SU_Set') %define Setup values and variables
        %{'SU_Set'};MagToolR01;
        SU.Dm=[];
        SU.ti=[];
        SU.fName='';
    end
    if strcmp(gKey{1},'LoadMag1x8_r01') %Read file with 1-magy x8-times for the Scanfish
        %{'LoadMag1x8_r01','d:\002\1magg\0052_MAG_'};MagToolR01;
        try fName=gKey{2};catch,fName=input('Magy*8 file names mask for reading=');end
        ss='Db,DD,MM,YYYY,hh,mm,ss,LineName,CogE,CogN,MruPitch,MruRoll,MruHdt,E,N,Hdt,T,Sig,Dp
        for n=1:8,
            [~,M1{n}]=gDataStructRead([fName num2str(n,'%02d.txt')],ss,'%s%f/%f/%f%f:%f:%f%s%f
            M1{n}.GpsDay=gNavTime2Time('YMD32Dx',M1{n}.YYYY,M1{n}.MM,M1{n}.DD);M1{n}.GpsTime=g
            M1{n}.ts=smooth(M1{n}.t,60)';     %p=polyfit(ProfM{n}.GpsNS,AbsTS0,1);ProfM{n}.MagA
        end;
        SU.Dm=M1{1}.GpsDay(1);
        a=zeros(8,1);for n=1:8,a(n)=mean(diff(M1{n}.ts));end;[s,k]=min(a);tmp=fliplr(M1{k}.ts(
        for n=1:8,s=mean(diff(M1{n}.ts));tmp=fliplr(M1{n}.ts(100):-s:M1{n}.ts(1));M1{n}.ti=[tm
        clearvars fName ss n a s k tmp ti
    end;
    if strcmp(gKey{1},'LoadMag8x1_r01') %Read file with 8-magys for the Scanfis
        %{'LoadMag8x1_r01','d:\002\01_Raw_data\0052_prt'};MagToolR01;
        try fName=gKey{2};catch,fName=input('8-magys file name for reading=');end;SU.fName=fNa
        ss='Db,DD,MM,YYYY,hh,mm,ss,LineName,CogE,CogN,MruPitch,MruRoll,MruHdt,E1,N1,Hdt1,T1,Si
        [~,M8]=gDataStructRead([fName '.txt'],ss,'%s%f/%f/%f%f:%f:%f%s%f%f%f%f%f%f%f%f%f%f%f
        M8.GpsDay=gNavTime2Time('YMD32Dx',M8.YYYY,M8.MM,M8.DD);M8.GpsTime=gNavTime2Time('HMS32
        clearvars fName ss
    end;
    if strcmp(gKey{1},'Interp_r01') %Create new structure, interpolated to ti
        %{'Interp_r01'};MagToolR01;
        W=M8;
```

In addition, there are certain variable names (in the example given it is SU) and certain field names for structures (in the example given these are DD, MM, YYYY, hh, mm, ss and others). Which are specialized and common. Using these names will make the script text more readable (and sometimes even necessary for the execution of functions in the code of which such "specialized" fields are written).

With the approach described above, "data processing" will be implemented as the sequential execution of several script commands with the appropriate parameters. In our example, this will be reading several files with magnetometer data (LoadMag1x8_r01, LoadMag8x1_r01 commands), some mathematical calculations based on the loaded data (Interp_r01), outputting the calculation results to a file (SaveMag8x1_r01 command). An example of a window with a completed working sequence of MagToolR01 script commands is shown in the figure below.



```
>> {'SU_Set'};MagToolR01;
>> {'LoadMag1x8_r01','d:\002\1magg\0052_077_SB MAG_'};MagToolR01;
>> {'LoadMag8x1_r01','d:\002\01_Raw_data\0052_077_stb'};MagToolR01;
>> {'Interp_r01'};MagToolR01;
>> {'SaveMag8x1_r01'};MagToolR01;
>>
```

Conclusion

The document covers:

-- installing the freely GNU Octave programming environment and its additional libraries;

-- installing the ge0mlib library;

-- installing git and using commands to get a commit (snapshot) of the ge0mlib library at some point in the past.

An example of a script for recalculating magnetic survey data (UXO) is given. Considered where the following information can be found in the script text:

-- List of additional libraries required to execute the script;

-- Description of script commands and command parameters;

-- Examples with “working sequences” of commands;

-- Date of creation of the script;

-- The environment used when testing the script (MatLab, Octave);

-- Email the script developer, to write to him all what you think about him.

The information provided allows you to install the software quickly and proceed to data processing using a script (m-file).