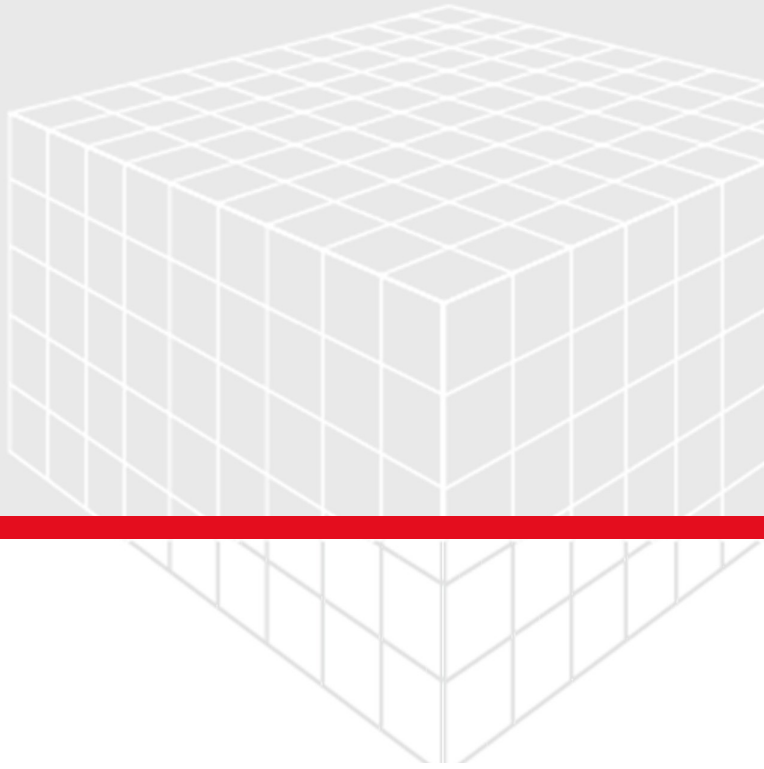
GeoLab

User Guide



GEODICT Release 2026

Published: June 2026

GeoDict

https://doi.org/10.30423/userguide.geodict

GeoLab

1 GeoLab

GeoPython offers the most convenient way to automate GeoDict workflows. However, for users who prefer MATLAB, the GeoLab interface provides control over GeoDict input and output files. This allows you to easily wrap custom MATLAB scripts and functionalities around GeoDict.

GeoDict macro files can be used to steer GeoDict with MATLAB. It is possible to run GeoDict simulations for different parameter settings and afterwards analyze the results directly in MATLAB. This automation possibility is applicable, e.g., for running parameter studies or performing material optimization.

To use GeoLab, a valid installation of MATLAB is required. Licenses for the installation of MATLAB are not provided by Math2Market GmbH.

Since GeoDict 2024, the minimal MATLAB version requirement is MATLAB R2019. Also the examples used in this GeoLab chapter are based on MATLAB R2019.

All GeoDict related file formats are supported in GeoLab, such as:

- GeoDict analytic data files (.gad)
- GeoDict structure files (.gdt)
- GeoDict result files (.gdr)
- GeoDict macro files (.py, .gmc)
- GeoDict universal files (.vap, .gpp, etc.)

That means that each GeoDict result file format (for flow fields, particle trajectories, particle positions, stress and strain fields...) can be loaded in MATLAB and the contained data can be accessed.

GeoLab offers various options for user specific analysis of the data contained in the GeoDict input and output files, modifications of input structures, post-processing for results data as well as applying the visualization properties of MATLAB to the data.

Examples provided with GeoLab show modelling capabilities for combining GeoDict, GeoLab and MATLAB functionality. They also show how results of the different GeoDict modules can be used and further analyzed with GeoLab.

Learn how to use GeoLab

- GeoLab Installation 5
  Install GeoLab on both Windows and Linux system.
- GeoLab Documentation 8
  Access GeoLab documentation from MATLAB.
- GeoLab Classes 11
  The available GeoLab classes and their functionalities.
- GeoLab Examples 21
  The examples shipped along GeoLab.

Join the conversation

Visit the Community on GeoDict Forum to be inspired and get answers to top questions.
Find illustrative application examples on the website.
Our tutorials and seminar videos give you a first hands-on experience. They are collected in the Learning Center.
Send your Support Request to our technical Support.

1.1 GeoLab Installation

GeoLab can be found inside the GeoDict installation folder in the subfolder **`GeoLab`**. The default location in Windows is:

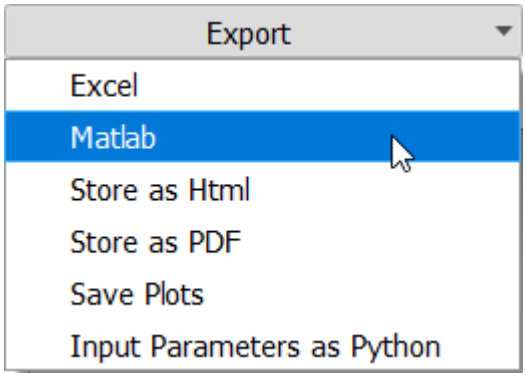
GeoDict may be installed in another folder, especially in case of Linux systems. For Linux, use the installation path of GeoDict on your system instead of the path above.

GeoLab consists of two parts. The first is the folder **`GeoLab\Lib`** that contains all functionality provided for working with GeoDict files. In the folder **`GeoLab\Examples`** you find several examples about working with GeoLab.

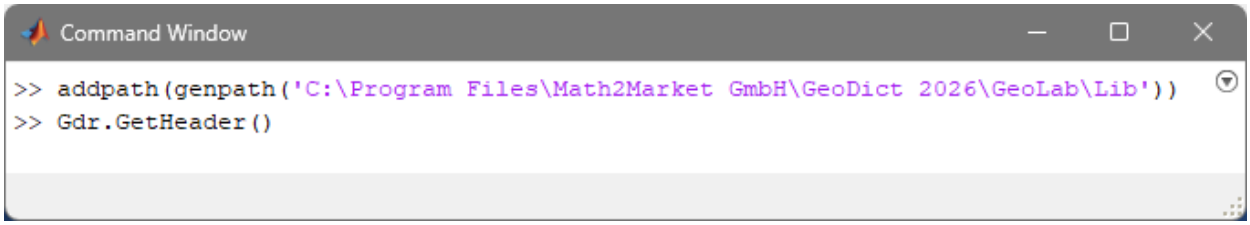
To use GeoLab, the path of the folder **`GeoLab\Lib`** needs to be added to the MATLAB search path. This can be done in different ways.

For Windows installations, a GeoLab desktop icon is created during the GeoDict installation. This allows to directly start GeoLab on computers where MATLAB is installed. It starts MATLAB and adds the path of the Lib folder of your GeoLab installation to the MATLAB search path.

Alternatively, GeoLab can be started by selecting **Export → Matlab** at the bottom of the GeoDict Result Viewer. If a MATLAB installation is available, MATLAB is started, and the path of the GeoLab Lib folder is added to the search path. The GeoDict result file (GDR file), selected in the GeoDict Result Viewer, is imported to MATLAB additionally.

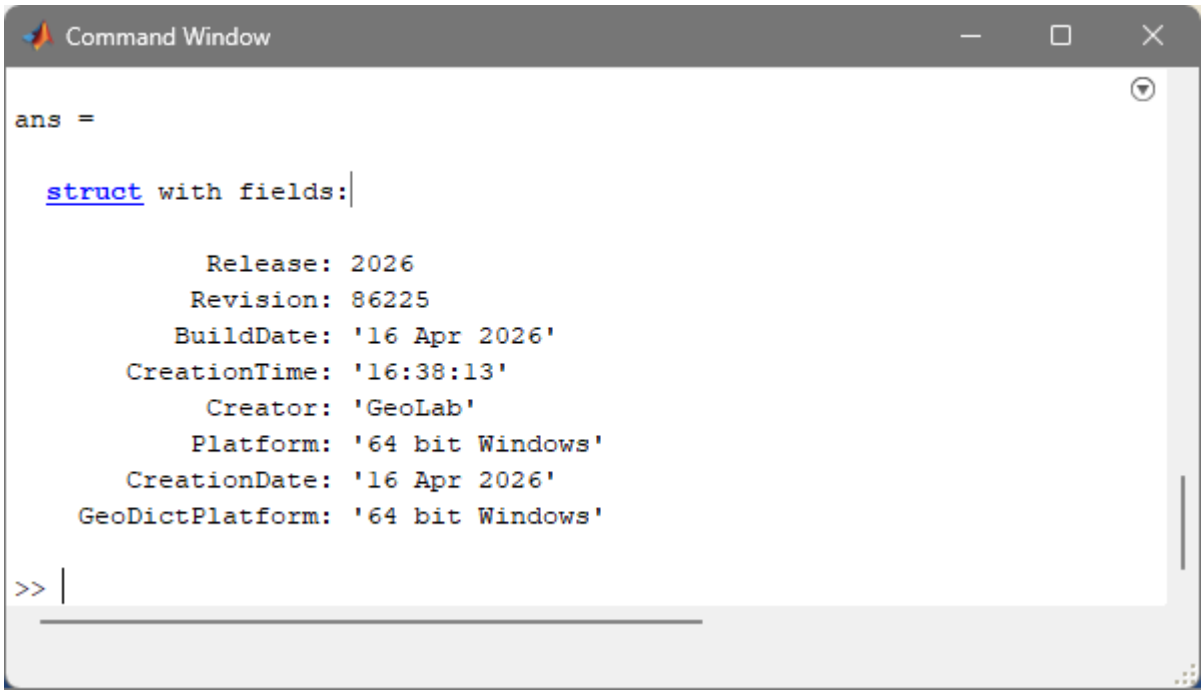


For Linux installations or if you prefer to start MATLAB directly in Windows, this can be done by adding the path in the MATLAB command window. Writing the following in the command window of MATLAB, adds the path and checks that this worked correctly.



The second command returns a MATLAB struct containing information about your GeoLab Program Files version. If **`Gdr.GetHeader()`** does not return an error, GeoLab is installed correctly. When support needs to be contacted about any issues with the installation, it is important that this information is enclosed.

The following should be displayed:

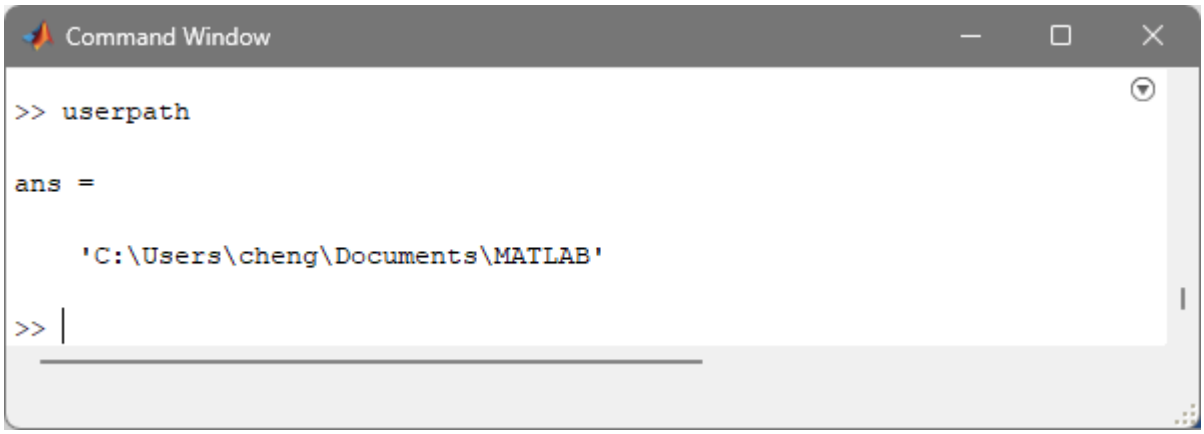


Your MATLAB version may differ from the one used in this chapter, but most information should hold regardless of the version.

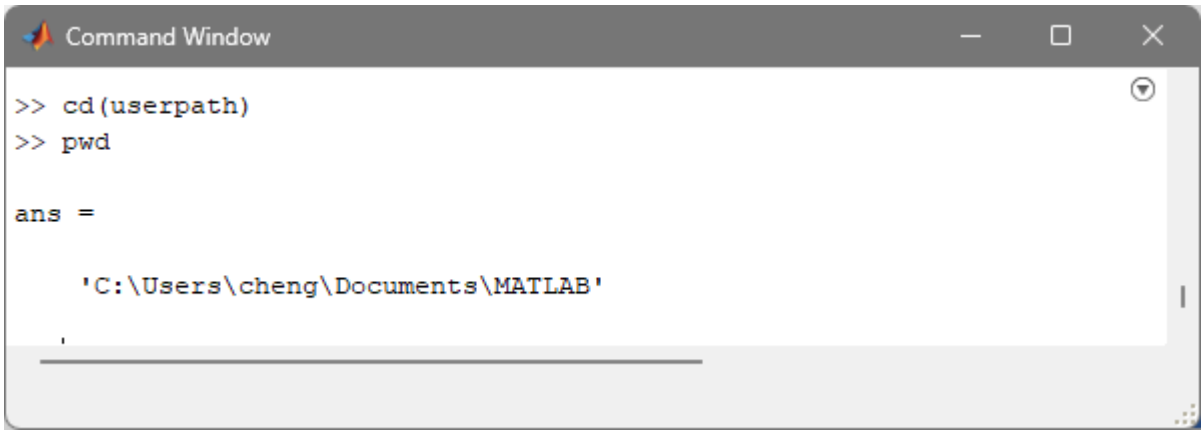
To use GeoLab functionality, it is sufficient to add the folder GeoLab\Lib to the MATLAB search path. If you are running an example of GeoLab, required additional paths are added automatically.

The process of installing GeoLab needs to be repeated every time MATLAB is started, unless the user sets this to be done automatically at startup, as follows:

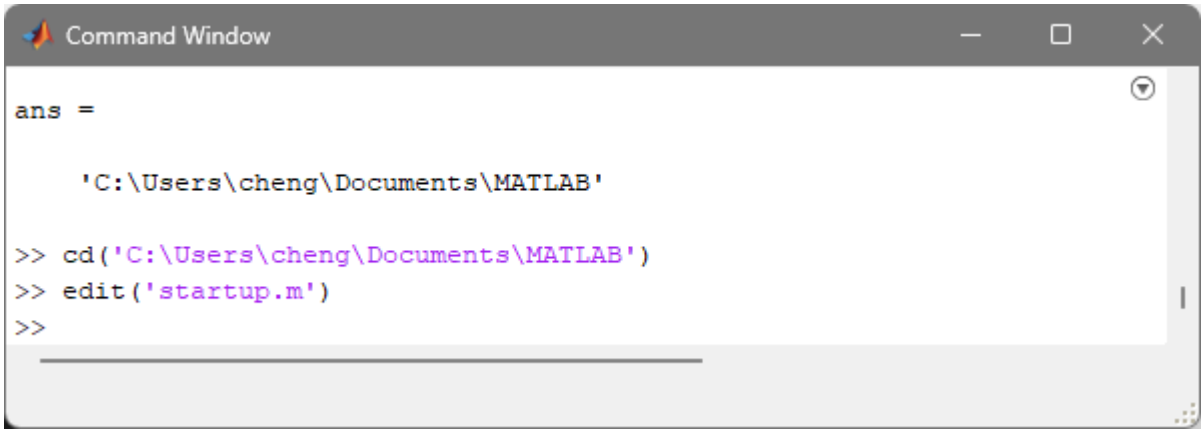
1. Write **`userpath`** in the Command Window. The path is the startup directory for MATLAB. Here it is **`C:\Users\cheng\Documents\MATLAB`** as shown in the command window. If needed, the path could be changed to another location with **`userpath('C:\...\NewPath')`**.



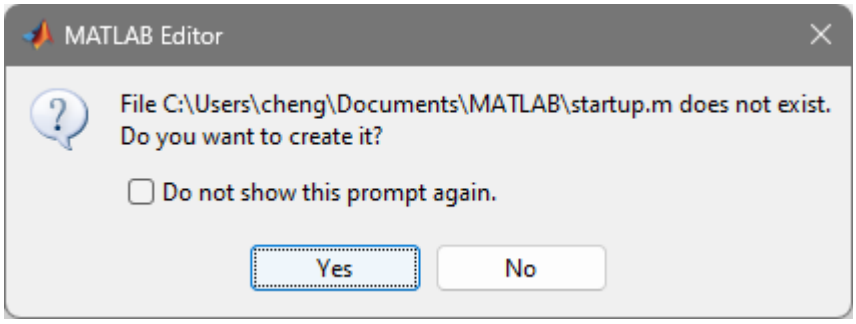
2. Navigate to the startup directory with **`cd('userpath')`**:



3. With `edit('startup.m')`, create a file that is executed at every MATLAB startup, if this doesn't already exist.

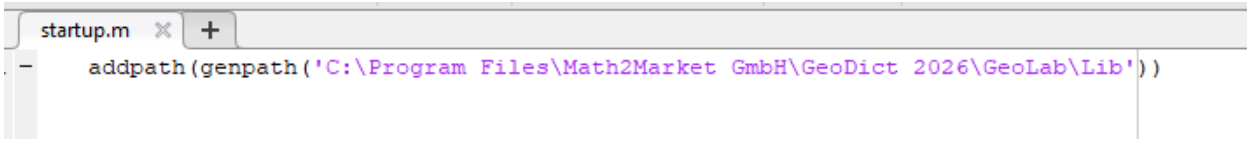


The MATLAB Editor window appears on top of the Command Window.



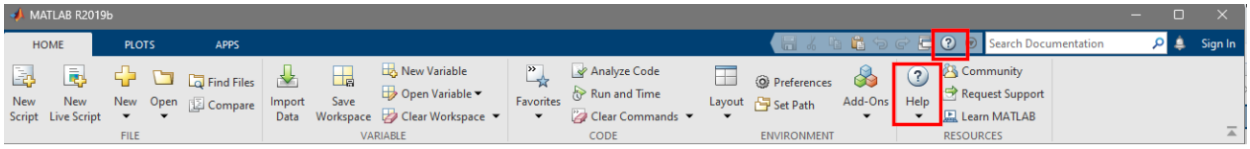
If the file `startup.m` already exists, it is opened in the editor.

4. Write `addpath(genpath(...))` in the `startup.m` file, pointing to the GeoLab subfolder in the GeoDict 2026 installation folder. If the file already exists, it is recommended to add the command at the end of the file. Save the `startup.m` file by selecting **Edit → Save** in the GeoDict GUI toolbar.

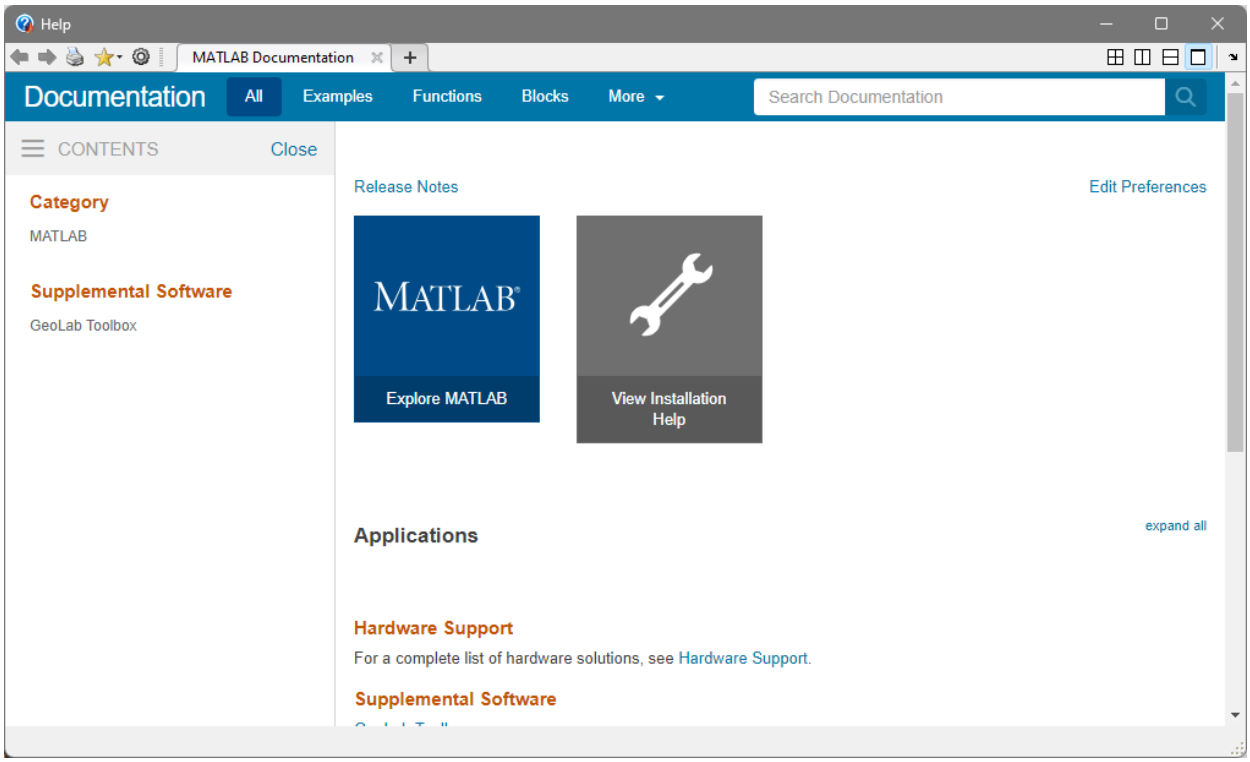


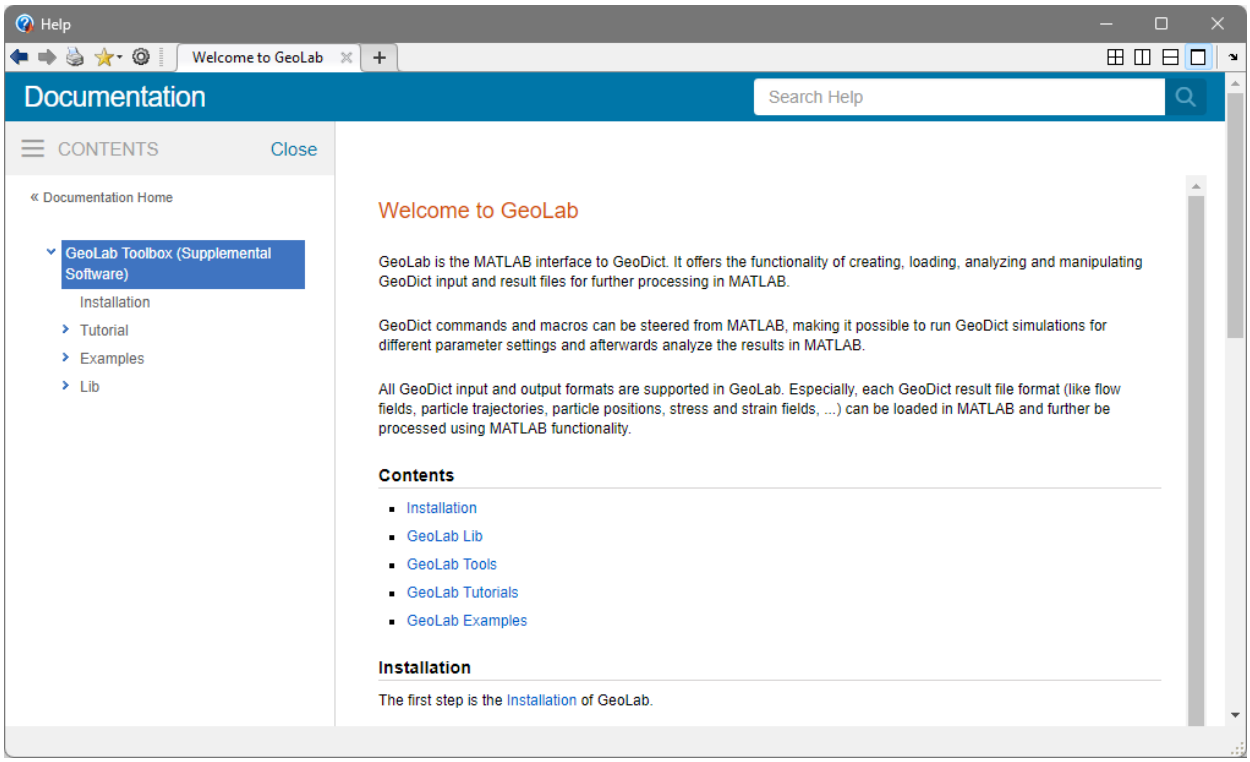
1.2 GeoLab Documentation

After installation is complete, you can access the GeoLab documentation in addition to this GeoLab User Guide by clicking **Help** in the MATLAB toolbar.

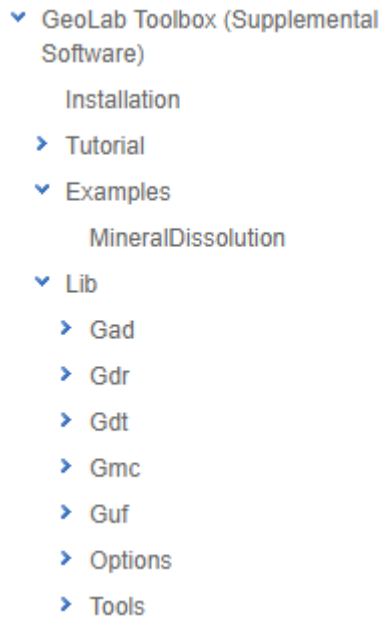


In the opened **Help** window, click **GeoLab Toolbox**.





The GeoLab documentation in MATLAB contains a detailed description of the functionality available for each GeoDict input and output format, as well as a documentation of the available examples and tutorials.



Installation

Installation explains how to access GeoLab functionality in MATLAB, see also GeoLab Installation 5.

Tutorial

Tutorial shows the use of the GeoLab functionality for a specific GeoDict input or output format. In the documentation is explained step by step how to perform a task for a specific GeoDict file format.

Examples

The **Examples** available show how GeoDict results of different modules are further analyzed in MATLAB, or an input for visualization or processing in GeoDict is created, see also GeoLab Examples 21.

The case study **Mineral Dissolution** illustrates the combination of the modelling capabilities of GeoDict with GeoLab and MATLAB functionalities to perform complex user defined tasks.

For the **Examples**, all files necessary to run them are provided in the subfolder `Examples` of your GeoLab installation.

Lib

Lib contains a list of all available GeoLab classes according to the GeoDict input and output formats. The functions of the classes are explained in GeoLab Classes 11.

Under **Lib** section, also **Options** and **Tools** are listed.

- **Options** contain some functions to change the default settings of GeoLab.
- Functions in the **Tools** folder provide an easy access to frequently used user defined tasks.

1.3 GeoLab Classes

The GeoLab functionality is centered on the ability to analyze GeoDict related files. GeoLab allows to work with

- GeoDict analytic data files (.gad)
- GeoDict structure files (.gdt)
- GeoDict result files (.gdr)
- GeoDict universal files (.vap, .gpp, etc.)
- GeoDict macro files (.gmc, .py)

For each of these file types, a class (data type) is available in GeoLab. This class allows to load and edit the content of the files and provides additional functionality to work with the data in MATLAB (depending on the file type).

Topics of this section

1.3.1 GAD Class

The classes for the different GeoDict file formats are named like the file ending for the file type, but with capital first letter. The class for working with *.gad files (GeoDict Analytic Data) is named Gad class. The same holds for the other GeoDict input and output formats.

Creating an object of a class by loading an already existing file is done by typing

```
>> gad = Gad('ExampleGADFile.gad');
```

Like this, an object of the Gad class is created in MATLAB with the name **gad**. The object name can be chosen arbitrarily.

To access any function available for the class, call the function on the existing MATLAB object. For example, for accessing the first analytic object of the **.gad** file (a sphere in the example), call

```
>> s = gad.GetObject(1)

s =

  struct with fields:

    MaterialIDModel: 'Constant'
         MaterialID: 1
               Type: 'Sphere'
           Position: [2.0000e-05 2.0000e-05 2.0000e-05]
           Diameter: 2.0000e-05
              Axis1: [0.1526 -0.7978 0.5833]
              Axis2: [0.8832 -0.1547 -0.4428]
              meta_: [1×1 struct]
```

Note, that the name of the created object (**gad**) is used here, in contrast to the creation of the object above where the class name (**Gad**) is used.

Note also, that the word **object** has two different meanings in the example shown above.:

- gad is an object of the Gad class, with all information of the file **ExampleGADFile.gad**.
- The variable s contains the parameters of the sphere, an analytic geometric object, created in GeoDict, that is contained in the file ExampleGADFile.gad.

For further information about classes and object-oriented programming, see object oriented programming in MATLAB.

To display a list of all available functions for objects of a class, type, e.g.,

```
>> gad.what()
```

in MATLAB, this function is available for all GeoLab classes.

```
>> gad.what()
Gad.what:
gad    = Gad (filename)

gad    = CreateNew (filename)

gad    = CreateNew ()

count  = gad.NumberOfObjects

domain = gad.Domain

header = gad.Header

types  = gad.Types

obj    = gad.GetObject (number);

obj    = gad.GetObjects (numbers);

         gad.SetObject (number, object);

         gad.SetObjects (numbers, objects);

         gad.DeleteObject (number);

number = gad.AddObject (object);

         gad.SetDomain (newDomain);

         gad.SetHeader (newHeader);

         gad.Save(filename);

         gad.Save();
>>
```

A detailed description of the functionalities available can be accessed in the **Lib** section of the MATLAB GeoLab documentation.

GAD files are text files which contain structure information in the form of analytic object data. GAD files can be modified with GeoLab.

The analytic objects can be edited, and new objects can be added to the structure. Various applications are possible. For example, it is possible to bend fibers between simulation steps or simulate crystal growth by adding objects. The generation of new GAD files in GeoLab is also possible, so even the programming of your own structure generators in MATLAB is possible.

For further information about the GAD format, see the GadGeo chapter.

GeoLab's GAD class allows to load and edit GAD files. Objects of the structure can be edited, added or deleted with the functionality of the GAD class. The domain itself can be edited, like changing the number of voxels in X, Y and Z direction or changing the voxel length. Other properties of GAD files, like the constituent materials, can be accessed and modified as well.

For detailed information about working with objects of the GAD class in GeoLab, see the GeoLab documentation in MATLAB.

1.3.2 GDT Class

GeoDict structure files in GDT format contain voxel structures. They store information about the materials in the structure and each voxel in the structure is assigned to one of the materials. With GeoLab, it is possible to load such a GDT file in MATLAB, modify it, and save changes for later visualization in GeoDict. Also, analytic object data can be contained in a GDT file and can be accessed and modified in GeoLab.

Since GeoDict 2023, 256 different materials (with IDs from 0 to 255) can be handled, where ID0 usually stands for the background material. GeoLab allows also to set material IDs from 0 to 255.

GDT files are compressed to save space on the hard disk and to allow faster loading and writing speeds in GeoDict. When loading them to GeoLab, they need to be unpacked, when saving them, they need to be packed again. This is done automatically, so you usually don't need to take care of this.

GeoLab's GDT class allows to load and edit GDT files. All properties of the GDT class, like the number of voxels in each direction, voxel length, etc. can be accessed. It is possible to edit the domain size as well as the structure data or part of it. Furthermore, it is also possible to change the constituent materials.

The capabilities and properties of the GDT class are explained in detail in the GeoLab documentation in MATLAB.

1.3.3 GUF Class

GUF stands for **G**eoDict **U**niversal **F**ile format. This file format is used for many different applications in GeoDict, mainly for result files.

Three different data types are contained in a GUF file, and are handled differently in GeoLab:

- Tree-structured metadata, that contains the header information of the file.
- Image data (or Volume Fields)
- Array data

Examples for *Images* are *.**vap**-files (**v**elocity **a**nd **p**ressure, e.g., from FlowDict) or *.**das**-files (**d**isplacement **a**nd **s**tresses, from **ElastoDict**). *Images* contain information on the voxel grid. *Arrays* contain vector data, which might not be directly related to the voxel grid, as, e.g., particle data: *.**gpt** (**G**eoDict **p**article **t**rajectories) or *.**gpp** (**G**eoDict **p**article **p**ositions). Images and arrays can also be used to store user-defined data, as, e.g., a scalar field describing structural damage.

GeoLab's GUF class allows to load and edit all GUF files. Images and arrays available in the GUF file can be accessed and modified and new images and arrays can be added to the file. For a full list, and a detailed description of the functions, check the GeoLab documentation.

1.3.4 GDR Class

During computations and operations with GeoDict, result files (GDR files) are created and saved in the project folder, together with a folder of the same name. More information on the GDR file format (and other GeoDict file formats) can be found in the Result Viewer chapter.

A GDR file contains information on finished GeoDict operations, like the simulation results. The corresponding result folder contains solver results, the geometry, and some intermediate solver files. Most of these files can be loaded to MATLAB with GeoLab.

The GDR class in GeoDict allows to import a result file (GDR file) in MATLAB. The information that it contains can be accessed or manipulated.

Results from GeoDict operations, saved in .gdr-files, can be loaded and edited with GeoLab's GDR class. Nearly all information in the .gdr file can be accessed via the Data property. More functions, like LoadFlowField which allows loading result data directly from the .gdr-file folder, are available.

Additionally, new Gdr files can be created with GeoLab and imported to GeoDict.

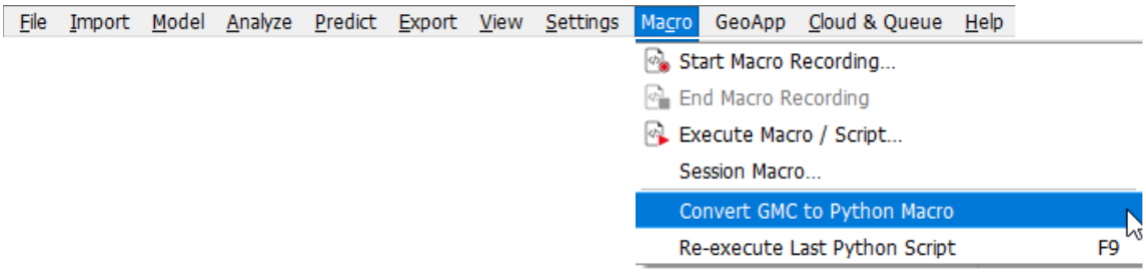
For a description and examples of the functions, check the GeoLab documentation in MATLAB.

1.3.5 GMC Class

GeoDict Macro files (.py or .gmc) contain a list of commands to steer GeoDict automatically. Each step in GeoDict can be recorded in a macro file and be replayed later. The real power of macro files is automation: parameters in the macro file can be changed automatically, and so a job can be executed multiple times with different settings. That way, parameter studies can be easily performed without a need for user interaction.

GMC Macros

Since GeoDict 2021, the recording of macros in GMC format is no longer supported in GeoDict. Macro files are stored in the newer GeoPy (*.py) format, that was introduced in GeoDict 2015. The GeoPy format provides more powerful capabilities since it allows using the full power of python to control GeoDict. GeoPy files cannot be edited with GeoLab, but it is possible to run GeoPy macros from GeoLab and to change the variables defined in the python macros.



GMC macros of GeoDict versions before GeoDict 2021 can be converted to a Python macro in GeoDict using the **Convert GMC to Python Macro** functionality.

Nevertheless, the older *.gmc format is still supported in GeoLab. GeoLab can load and edit GMC files, and all parameters in the macro file can be changed in MATLAB with the GeoLab GMC class. The information of the GeoDict commands contained in the macro file can be accessed and modified. Furthermore, new GMC files can be created from available templates and the macros can be automatically executed with GeoDict.

For a description and examples of the functions available, check the GeoLab documentation in MATLAB. The process of connecting with GeoDict and executing the macros automatically is explained there as well.

GeoPy Files

GeoPy files cannot be edited in GeoLab. However, it is possible to execute such Python macros in GeoDict from GeoLab. Furthermore, the variables which are defined in the macro can be set from GeoLab. Likewise, it is possible to automate the run of such macros with GeoLab as well. Variables can be changed every time the macro is called, and the result files can be further analyzed with GeoLab.

For the definition of variables in a GeoPy file, see the Automation chapter.

Run a GeoDict Python Macro with GeoLab

To run the macro from GeoLab, one of the options is to use `geodict.m`. When running it without any argument, information such as the GeoDict executable file path, version, etc. is shown.

When running it with `geodict(filename)`, a python file can be used. The GeoDict command line options can also be considered by using `geodict(filename, options)`. Possible options are those that can be used when running GeoDict in command line mode. For example, if the GeoDict GUI is desired, `'--stayopen'` should be used for the options. Then GeoDict stays open after the python file is executed.

The other possibilities to run the macro from GeoLab are to use the function `runMacro.m` or `runGeoPythonMacro.m`. At least the filename of the GeoDict macro is necessary as input of the two functions. It is also possible to give a list of variable

names and their desired values. With these two functions, only the variable inputs for macro files can be set, but not the general GeoDict command line options.

For example, to change the result file name, when a macro file is called from GeoLab, two steps are necessary:

1. Define the result file name as variable in the GeoPy macro:

```
Variables = {
   'NumberOfVariables' : 1,
   'Variable1' : {
     'Name'           : 'ResultFile',
     'Type'           : 'string',
     'BuiltinDefault' : 'FiberGeoExample.gdr',
     },
   }
```

   Set the variable in the function call for the 'ResultFileName':

```
'RandomSeed'                : 47,
'ResultFileName'            : ResultFile,
'MatrixDensity'             : (0, 'g/cm^3'),
```

2. Now, define this filename, when calling the macro from GeoLab. This can be done in two different ways:

```
>> geodict('GeoPythonExample.py','-v ResultFile Result1.gdr')
```

   or

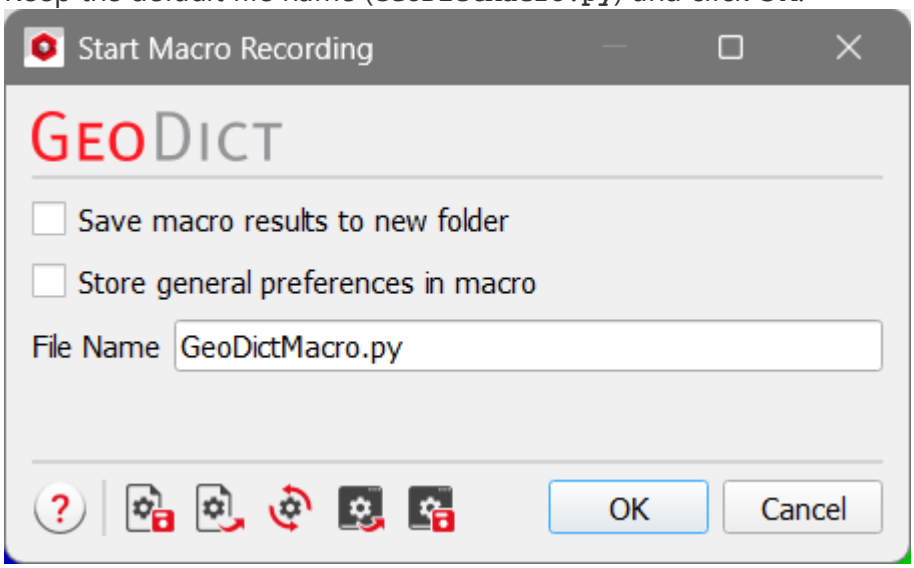
```
>> runMacro('GeoPythonExample.py','ResultFile','Result1.gdr')
```

Recording Macro files in GeoDict

As an example, here a short GeoDict macro is recorded while generating a fiber structure with FiberGeo:

1. Start GeoDict.
2. Select **Macro** → **Start Macro Recording...** in the menu bar.

3. Keep the default file name (`GeoDictMacro.py`) and click **OK**.

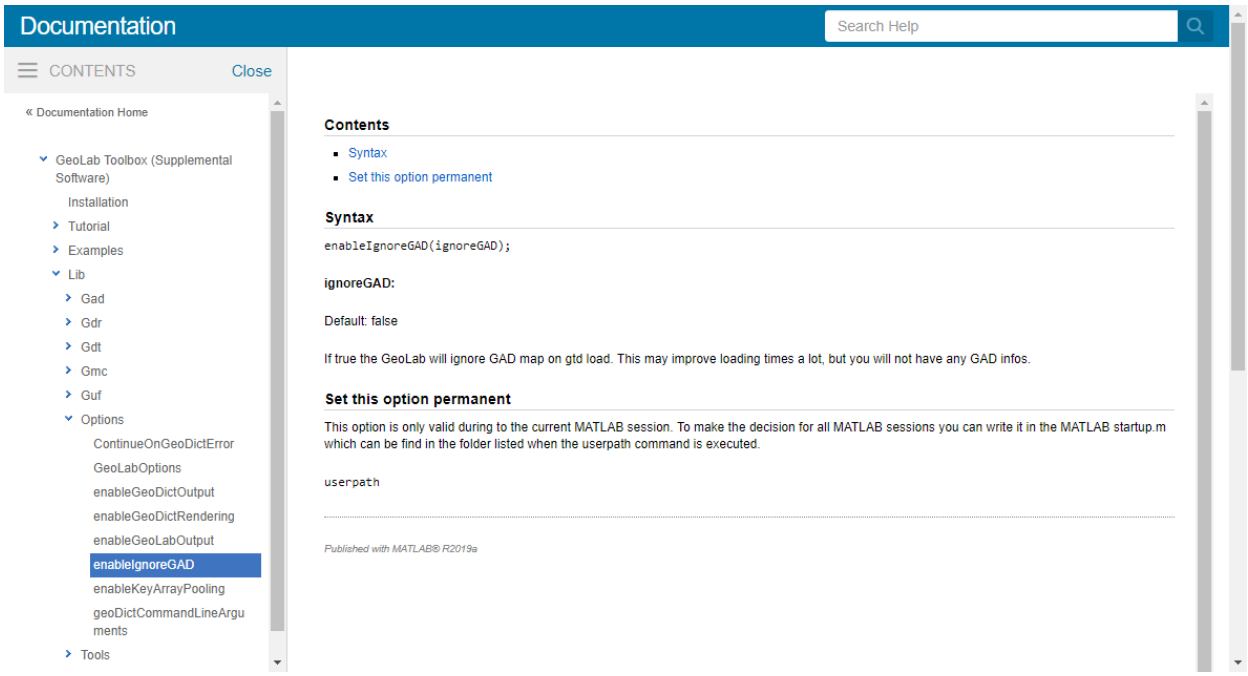


4. Choose **Model** → **FiberGeo**, select **Create Fibers** from the pull-down menu in the FiberGeo section of the GUI and click **Generate & Record** to generate a fiber structure with the default settings.
5. Select **Macro** → **End Macro Recording** in the menu bar. The `GeoDictMacro.py` file is now found in the project folder.

See the Automation chapter for more information on recording macros in GeoDict.

1.3.6 Options

Functions in the folder **Options** make it possible to change the default settings of GeoLab. For example, by default, when GeoLab reads a gdt file, the GAD map in it is loaded, too. If the GAD, however, is not needed, *enableIgnoreGAD(true)* can be used to speed up the loading. For a description and examples of the functions, check the GeoLab documentation in MATLAB in the section **Options**.

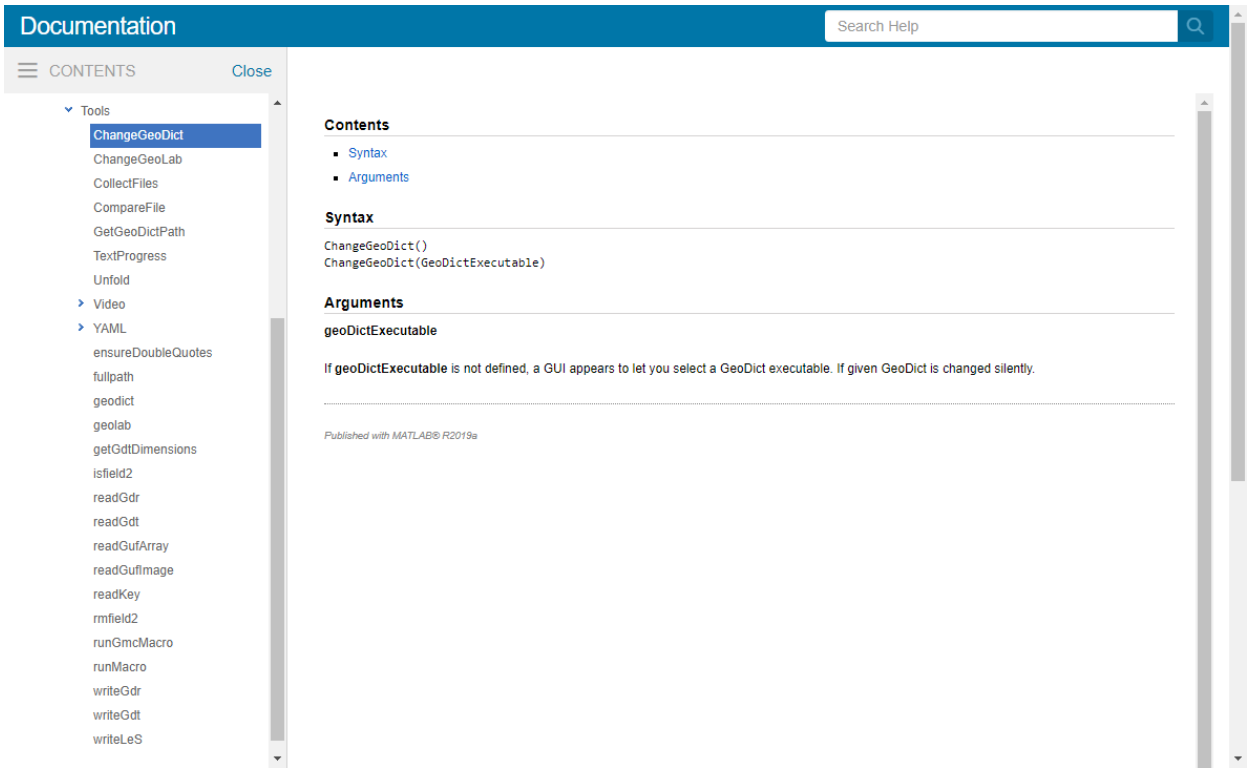


1.3.7 Tools

Functions in the folder **Tools** offer some functionality to work with GeoDict files in MATLAB without calling the class functions explicitly.

They offer easier access to frequently used tasks. This includes, e.g., importing and creating GDR and GDT files, importing data of arrays and images of GUF files and running GeoDict macros from GeoLab.

For a description and examples of the functions, check the GeoLab documentation in MATLAB in the section **Tools**.

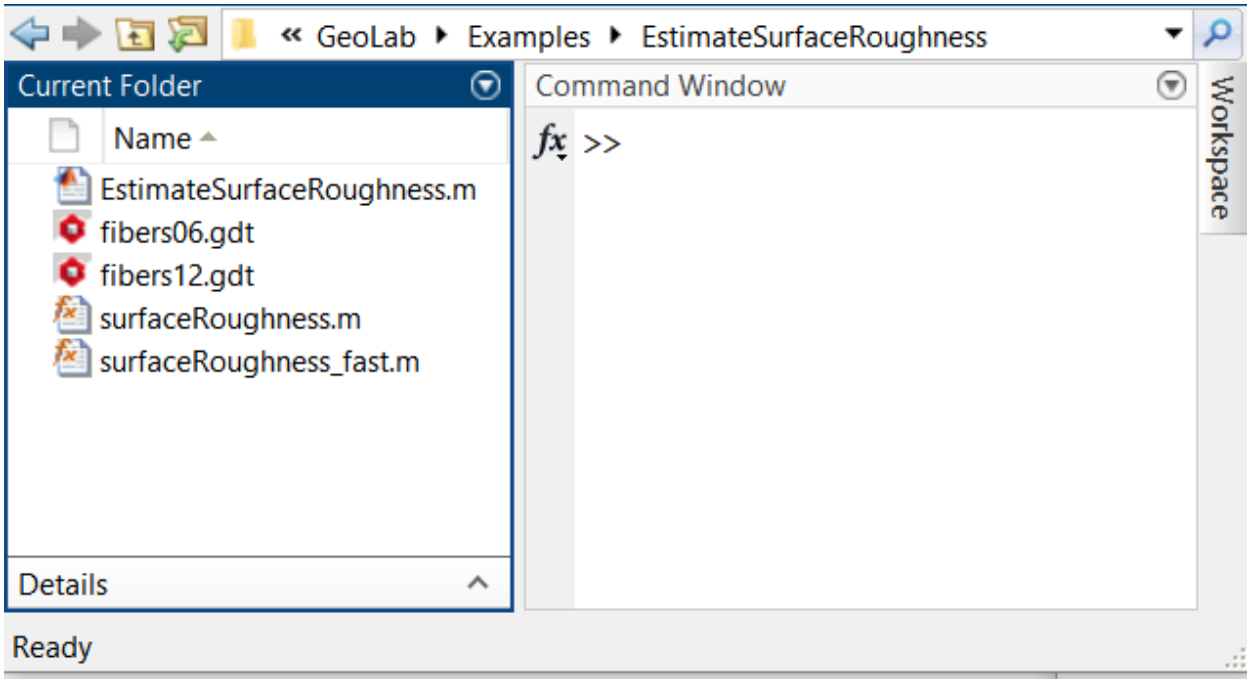

Documentation
Search Help
CONTENTS
Close
Tools
ChangeGeoDict
ChangeGeoLab
CollectFiles
CompareFile
GetGeoDictPath
TextProgress
Unfold
Video
YAML
ensureDoubleQuotes
fullpath
geodict
geolab
getGdtDimensions
isfield2
readGdr
readGdt
readGufArray
readGufImage
readKey
rmfield2
runGmcMacro
runMacro
writeGdr
writeGdt
writeLeS
Contents
Syntax
Arguments
Syntax
ChangeGeoDict()
ChangeGeoDict(GeoDictExecutable)
Arguments
geoDictExecutable
If geoDictExecutable is not defined, a GUI appears to let you select a GeoDict executable. If given GeoDict is changed silently.
Published with MATLAB® R2019a

1.4 GeoLab Examples

Examples showing the possibilities of working with GeoLab are provided in the subfolder **`GeoLab\Examples`** of your GeoDict installation. Each example is contained in a separate subfolder.

To run a GeoLab example you need writing permissions for the example folder. If you do not have such access rights in the folder of your GeoLab installation, you need to create a copy of the example folder first. However, even if you have writing permissions in the folder, it is recommended to create a copy of the folder as well.

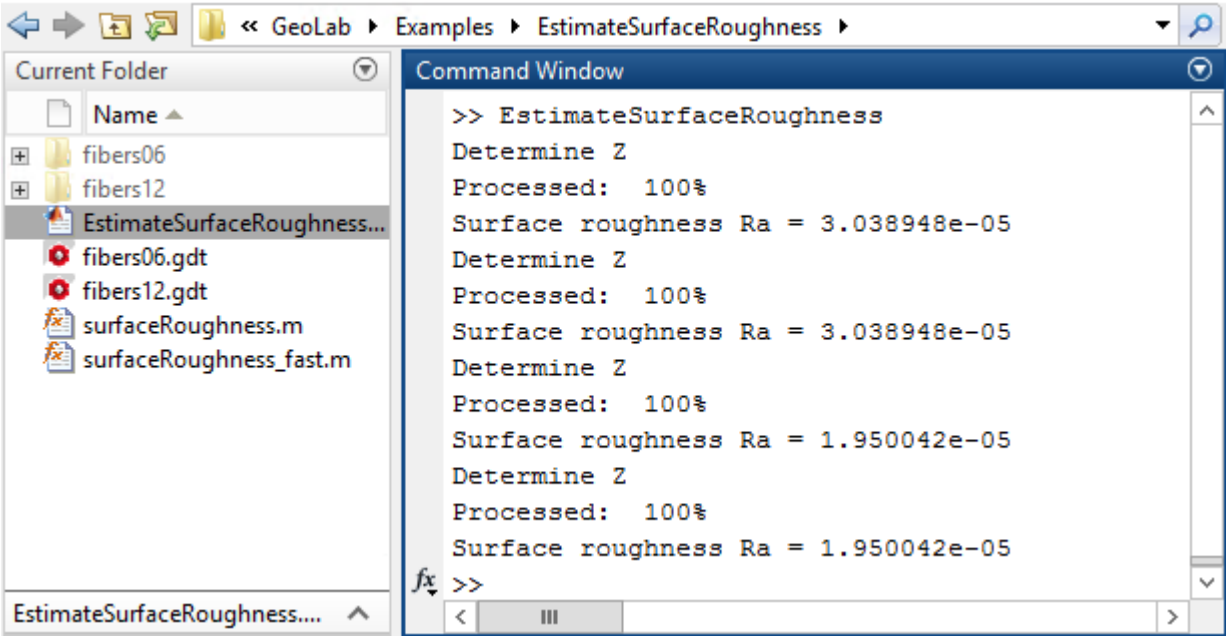
The output of EstimateSurfaceRoughness of MatDict is shown as example below.

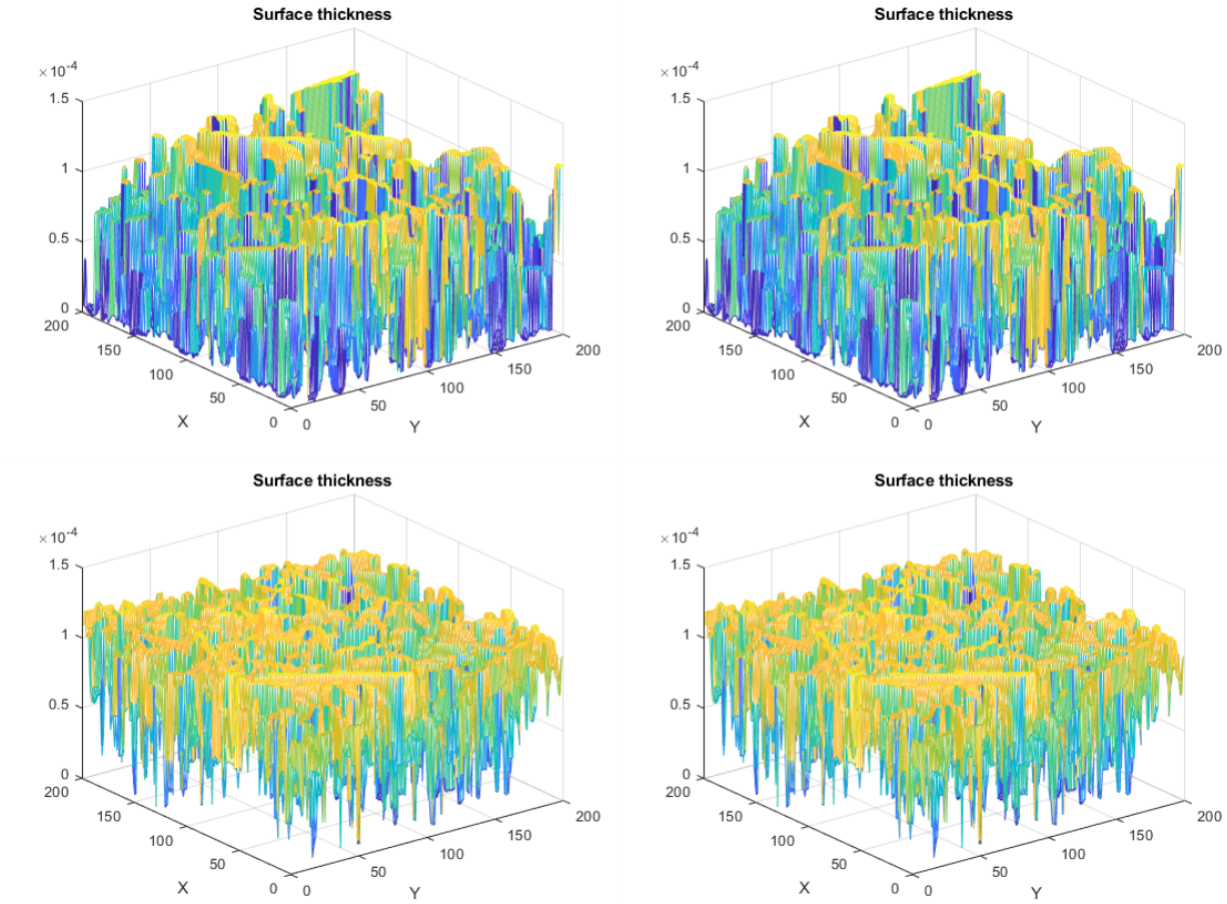


Call the MATLAB script **`EstimateSurfaceRoughness.m`** in the example folder to run the example. This file contains a call to subfunctions.

The example **`EstimateSurfaceRoughness`** calculates the surface roughness in z-direction for two different structures and with two different implementations for the calculation. The first one is a slower implementation looping over x- and y-direction of the structure. The second implementation is a faster one, showing how MATLAB functionality is used to speed up computation. The example therefore creates four figures, showing for each structure that both calculations produce the same result.

The MATLAB Command Window output for the example, as well as the four figures are shown below.


« GeoLab ▸ Examples ▸ EstimateSurfaceRoughness ▸
Current Folder
Name
fibers06
fibers12
EstimateSurfaceRoughness...
fibers06.gdt
fibers12.gdt
surfaceRoughness.m
surfaceRoughness_fast.m
EstimateSurfaceRoughness....
Command Window
>> EstimateSurfaceRoughness
Determine Z
Processed: 100%
Surface roughness Ra = 3.038948e-05
Determine Z
Processed: 100%
Surface roughness Ra = 3.038948e-05
Determine Z
Processed: 100%
Surface roughness Ra = 1.950042e-05
Determine Z
Processed: 100%
Surface roughness Ra = 1.950042e-05
>>


Surface thickness
Surface thickness
Surface thickness
Surface thickness
X
Y

Citation: Math2Market GmbH, 2026: GeoDict 2026 User Guide, GeoLab, Math2Market GmbH, Germany, doi.org/10.30423/userguide.geodict

MATH 2 MARKET

Math2Market GmbH
Richard-Wagner-Str. 1, 67655 Kaiserslautern / Germany
www.math2market.com