Using miniSoilLAB

Dominik Zobel

version: December 8, 2020

Contents

1 Introduction

miniSoilLAB is a program to read spreadsheets with results of laboratory tests of soils, integrate the data into a predefined structure and visualize selected results. It can either be used with a Graphical User Interface (see section 4) or loaded as a Python module in the command line (see section 5). Although the Graphical User Interface (GUI) is more intuitive to use and already gives a good impression of the loaded data, loading miniSoilLAB as a module offers more possibilites and access to the internal structure for custom usage. miniSoilLAB is written in Python and requires a working Python 3.x environment. A simple function documentation (in german) created with pydoc can be found at »https://d-zo.github.io/abapys/abapys.html to aid using miniSoilLAB as module.

2 Installation

miniSoilLAB uses some packages from the standard library, especially tkinter for creating the Graphical User Interface (GUI) and json for reading/writing data structures. Additionally, the following packages are required

- openpyxl for parsing `xlsx` spreadsheets and
- xlrd for reading older `xls` files
- matplotlib for plotting the results

miniSoilLAB can be downloaded and run with Python. In Linux the following commands can be used in a shell (the first command has to be adjusted to point to the right directory).

```
cd /path/to/miniSoilLAB
python3 miniSoilLAB.pyz
```

In Windows a batch file can be created to provide access to miniSoilLAB from anywhere. Therefore, the path to the Python interpreter and the path to miniSoilLAB have to be defined in the following code.

```
@echo off
pushd C:\path\to\miniSoilLAB
C:\path\to\Python\python.exe C:\path\to\miniSoilLAB\miniSoilLAB.pyz
pause
```

If the dependencies listed above are not present, they can be provided in some way or installed with `pip`. The Linux command to install them is

```
python3 -m pip install -U openpyxl xlrd matplotlib
```

and for Windows (paths have to be adjusted):

```
C:\path\to\Python\python.exe -m pip install -U openpyxl xlrd matplotlib
```

3 Files

miniSoilLAB differentiates between four different kinds of input files, raw data, spreadsheets, data set files and soil pattern files.

- *Raw data* files are simple text files with the measurements of laboratory tests. Those files are assumed to have a csv/table-like structure with a newline ending a row and a common separator (like a comma) between two entries. A header row has to be present for assign the values of each column to a corresponding name. Currently, there is only a very limited support for raw files since not all required information will be present there but in spreadsheets (sometimes with a copy of the raw data).
- *Spreadsheets* will be read to an internal structure based on the templates used in miniSoilLAB. By default, a folder called `Vorlagen` contains all templates known to miniSoilLAB as JSON files (can be changed/adjusted when calling miniSoilLAB). The structure of templates and how to modify them will be discussed in section 6. For each spreadsheet a matching template has to be found or the file won't be read. If a valid template is found, the specified cell contents are read into an internal structure of the program for this soil material. Some of the data can be plotted and examined while the whole data structure of the soil material (possibly with additions from other read files) can be saved as a data set.
- *Data sets* represent the internal structure of laboratory data for a single soil material. Those data sets can be saved as JSON files and load again afterwards. Since they already contain all read data from spreadsheets (and raw data), the original files mustn't be read again. One data set resembles all relevant data for one soil material. This does not render the original files useless (always keep original files) but enables to collect all relevant laboratory data in a single file for easy reference. The JSON format makes it easy to inspect the file content in a text editor or even load the saved structure in another program (like Octave or Matlab, see also section 7). Sometimes it might be a good idea to load all corresponding files and save the relevant data in a new files for future reference (i. e. creating data sets for soil materials). Sometimes it might be preferable to store only the instructions (name, search pattern and path) for loading soil materials (i. e. creating a soil pattern file).

- *Soil pattern files* are used to automatically load one or more soil materials from their respective original files. These files provide a comfortable way to specify name for the soil material and the (absolute) path to read the appropriate spreadsheets/raw data files. One advantage of using soil pattern files is that the data (if updated) is always read from the linked original files. A collection of soil materials is short and can easily be adjusted, if new files are added to the soil material. To control which files are to be loaded (e. g. if files for an other material are present as well), a search pattern (regular expression) can be defined to select only files matching the pattern. An empty search pattern includes all files found in the given path. Each line of a soil pattern file can either be an empty line, a comment line (starting with #) or a soil pattern line. A soil pattern line has three entries separated by a semicolon, i. e.

```
soil name  ;  search pattern  ;  path
```

4 Graphical User Interface

The Graphical User Interface (GUI) contains much of the functionality miniSoilLAB has to offer (for more control see section 5). After starting the GUI, everything below the menu bar is inactive until at least one file is successfully loaded and activated (see fig. 4.1). The name of the currently active soil material is shown below the menu bar next to *Boden:* and empty if none is active.

4.1 Loading data from files

To load a soil material, go to

- `Datei -> Datensatz oeffnen` to load a previously saved data set file in JSON format,
- `Datei -> Dateien einlesen` to load one or more spreadsheets/raw data files based on a path and a search pattern (name, search pattern and path constitute a soil pattern to identify all relevant files for a soil material) or
- `Datei -> Bodenmusterdatei einlesen` to load a soil pattern file and select afterwards which of the predefined soil materials to load.

More about the different kind of files can be found in section 3. The search pattern is a regular expression and some examples on using them are given in section 5.2. If any problems occur while reading or parsing the files, warnings and useful debug information is written to the console output. Make sure to check it when miniSoilLAB is not loading soil materials as expected.

After successfully loading a soil material, it can be activated by selecting it in the drop-down menu next to *Boden:* below the main menu. Information about loaded soil materials is divided in several categories (*Kornverteilung*, *Lagerung/Zustandsgrenzen*, *Auswertung* ...) as shown on the left hand side below the selected soil material. If all required information for a category has been loaded the category becomes active. Clicking on an active category expands it and shows information and plot commands. When a plot command is selected, the corresponding plot is shown in the right hand side of the window (cf. fig. 4.2).

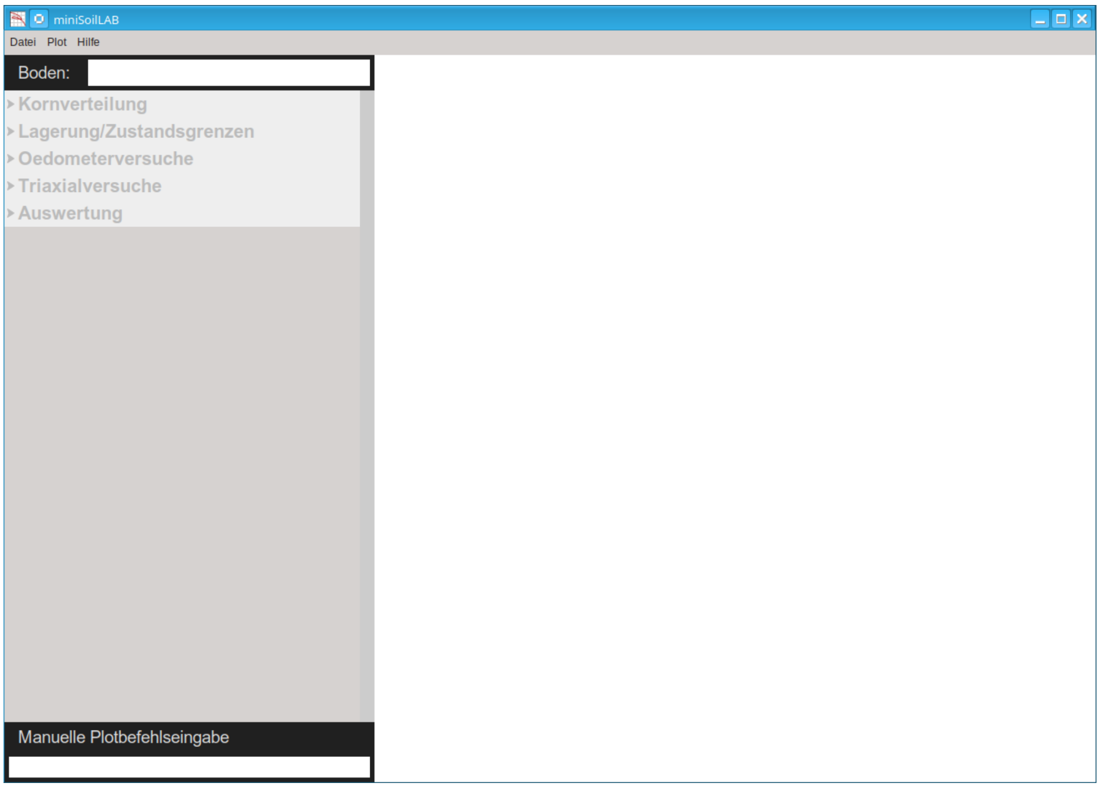

Figure 4.1: The default view after starting the GUI of miniSoilLAB.

4.2 Plotting and saving data

There are several predefined plot commands in each category for many common plots. It is also possible to create plots based on key words entered in the input below *Manuelle Plotbefehlseingabe* (and press `Enter`). This allows not only to show some of the predefined plots but also has some experimental features to add more data in one plot. The currently implemented plot commands are

```
kvs
oedo locker,dicht
oedo-crs-visko [setzung]
oedo-crs [locker,dicht] [stauchung]
oedo-crl [fit]
oedo-crl wurzel|logarithmus|porenzahl [laststufe 1|laststufe 2|laststufe 3|laststufe 4|
                                       laststufe 5|laststufe 6|laststufe 7|laststufe 8]
triax-d locker,dicht volume|reibungswinkel|porenzahl|mittelspannung|
                     eff.spannung|bruchgerade|spannungskreis
triax-cu hauptspannungsverhaeltnis|mittelspannung|porenwasserdruck|
         eff.spannung|bruchgerade|spannungskreis
triax-p-q [dehnung|e-modul|stauchung]
```

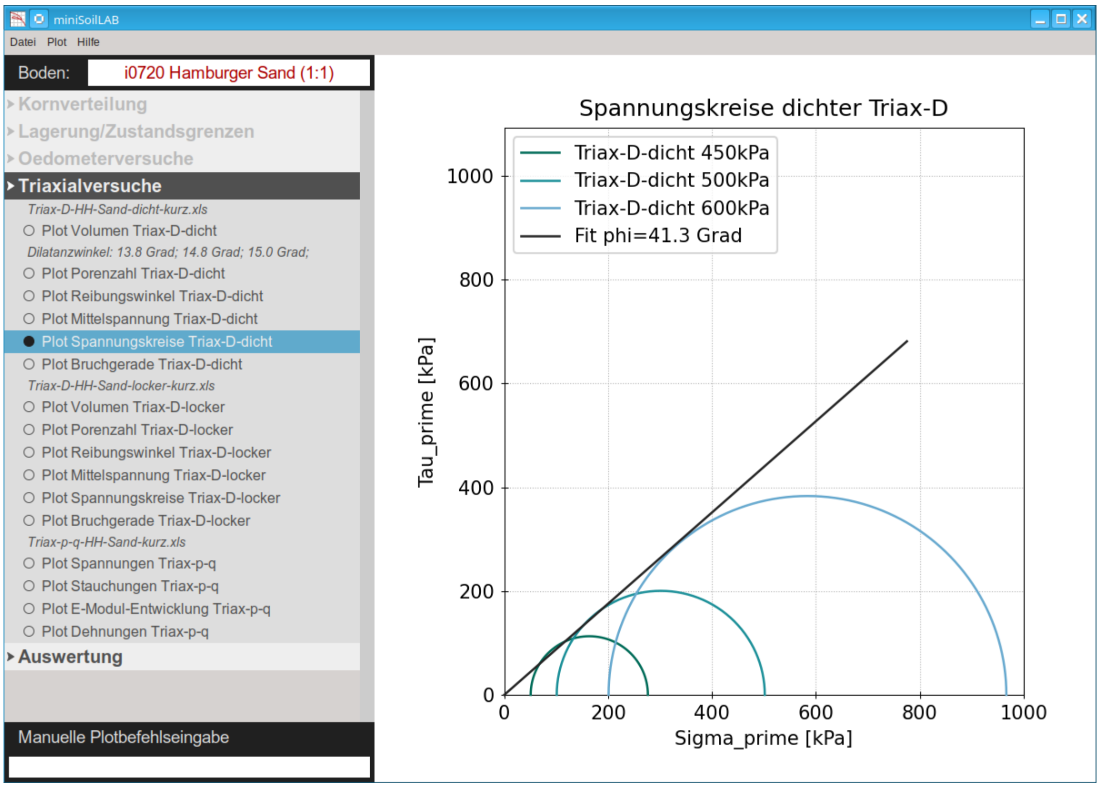
miniSoilLAB
Datei Plot Hilfe
Boden: i0720 Hamburger Sand (1:1)
Kornverteilung
Lagerung/Zustandsgrenzen
Oedometerversuche
Triaxialversuche
Triax-D-HH-Sand-dicht-kurz.xls
Plot Volumen Triax-D-dicht
Dilatanzwinkel: 13.8 Grad; 14.8 Grad; 15.0 Grad;
Plot Porenzahl Triax-D-dicht
Plot Reibungswinkel Triax-D-dicht
Plot Mittelspannung Triax-D-dicht
Plot Spannungskreise Triax-D-dicht
Plot Bruchgerade Triax-D-dicht
Triax-D-HH-Sand-locker-kurz.xls
Plot Volumen Triax-D-locker
Plot Porenzahl Triax-D-locker
Plot Reibungswinkel Triax-D-locker
Plot Mittelspannung Triax-D-locker
Plot Spannungskreise Triax-D-locker
Plot Bruchgerade Triax-D-locker
Triax-p-q-HH-Sand-kurz.xls
Plot Spannungen Triax-p-q
Plot Stauchungen Triax-p-q
Plot E-Modul-Entwicklung Triax-p-q
Plot Dehnungen Triax-p-q
Auswertung
Manuelle Plotbefehlseingabe
Spannungskreise dichter Triax-D
Triax-D-dicht 450kPa
Triax-D-dicht 500kPa
Triax-D-dicht 600kPa
Fit phi=41.3 Grad
Tau_prime [kPa]
Sigma_prime [kPa]
0
200
400
600
800
1000

Figure 4.2: The material *Hamburger Sand* was loaded and activated. One of the available plot commands was selected and the corresponding plot can be seen on the right hand side.

where each line represents all options for a valid command (arguments for `oedo-crl`, `triax-d` and `triax-cu` are continued in a second line). Optional arguments are within square brackets. `a|b` means either `a` or `b`. `a,b` means either `a|b` or `a b`. The order of the keywords can be changed and capitalization can be used.

A plot command uses the active soil material. If the keyword `alle:` is added at the beginnning of the plot command, it uses the data of all loaded soil materials instead. To compare only a selection of materials but not all loaded materials, all irrelevant materials must be unloaded first (e. g. `Datei -> Datensatz entfernen`).

Any visualized plot in the area on the right hand side can be saved as an image by choosing `Plot -> Als ... speichern` in the menu bar (where `...` is either `pdf` or `png`).

Loaded soil materials can be saved as data sets by `Datei -> Datensatz speichern`. The files are saved in JSON format and can also be viewed by any text editor. It is possible to save either single materials or all loaded materials. Similar to manual plotting, if all materials should be saved except a few, consider unloading those few materials before saving.

5 Console access

To load miniSoilLAB as a module, open a Python console and enter

```
from miniSoilLAB import *
```

If miniSoilLAB is not installed but just downloaded, you might consider to include its current directory to the path variable before loading it. The command should be similar to the following:

```
import sys
sys.path.insert(0, '/path/to/miniSoilLAB.pyz');
from miniSoilLAB import *
```

The documentation (in german) for the available function can be found at »https://d-zo.github.io/abapys/abapys.html. The next subsections give some examples on how they can be used.

5.1 Loading data from files

Single raw data files or spreadsheets can be loaded with `DateiEinlesen()`. The distinction between both will be made based on the file extension. Before reading the first spreadsheet, the necessary template-files will be loaded, so that the spreadsheet can be matched upon a template. One example for loading a spreadsheet and a raw data file are

```
testbodentriaxd = DateiEinlesen(dateiname='Testboden-Triax-D.xls');
kornverteilung = DateiEinlesen(dateiname='Testboden.kvs');
```

The function for reading the grain size distribution of a `.kvs` file can also be called explicitly with `LeseKVSDaten()`. miniSoilLAB provides four other functions to read selected raw data files, namely `LeseEAXDaten()`, `LeseDTADaten()`, `LeseGDSDaten()` and `LeseTVCDaten()`. These four functions are currently not used when loading soil material files (in the GUI). But they can be used in a similar manner like

```
triaxcu = LeseTVCDaten(dateiname='Testtriax.tvc');
```

Similarly spreadsheets can be read directly with the `LeseXLSDaten()` function if a matching template is available (the templates are loaded automatically at the first call of this function or `DateiEinlesen()` with a spreadsheet).

Templates are expected to be located in a `Vorlagen/` directory if not specified otherwise (the template files are discussed in section 6). When using the GUI, the location has to be provided at startup of the program. For console access, the template path can be adjusted with the function call

```
Standardpfad(pfad=path/to/templates);
```

Although templates can be used to read differently structured spreadsheets, miniSoilLAB expects the read contents to match certain laboratory test reports. Currently the following test reports (partially based on german standards) are supported:

- *Atterberg* based on DIN 18122 or DIN EN ISO 17892-12 respectively (determination of liquid limit and plastic limit)
- *Auswertung-Hypoplastisch* as a compilation of all test results required to determine hypoplastic parameters
- *LoDi* based on DIN 18126 (determination of density of non-cohesive soils for maximum and minimum compactness)
- *Oedo-CRL* based on DIN 18135 or DIN EN ISO 17892-5 respectively (oedometer consolidation test/one-axial compression tests with constant load steps)
- *Oedo* as a minimal version of *Oedo-CRL* for dense or loose soil configuration
- *Oedo-CRS* as an oedometric test with constant rate of strain
- *Oedo-CRS-Visko* is similar to *Oedo-CRS* but for cohesive soils and results required to determine viscohypoplastic parameters
- *Triax-CU* based on DIN 18137 or DIN EN ISO 17892-8/9 respectively (consolidated and undrained triaxial test)
- *Triax-D* based on DIN 18137 or DIN EN ISO 17892-8/9 respectively (consolidated and drained triaxial test)
- *Triax-p-q* as a triaxial test with a special loading path to determine the extended hypoplastic parameters

Details on the required values for a template to each of these test reports are listed in section 6. For each template the predefined checks are tested until a match is found or all templates have been tried.[1] The data of a file will be read to an internal structure as described in the matching template (nothing will be read if no template matches). For *Triax-D*- and *Oedo*-files the file name is also important in determining if the soil material is loose or dense

[1] Before the first spreadsheet is read, miniSoilLAB loads all templates and keeps them in memory for future comparisons.

(i. e. Oedo-locker and Oedo-dicht or Triax-D-locker and Triax-d-dicht respectively)[2]

Saved data sets can be loaded with `DatensatzEinlesen()`

```
bodendaten = DatensatzEinlesen(dateiname='gespeicherter_datensatz.txt');
```

Data sets are JSON files with arbitrary names (more on data sets in section 3).

The soil pattern lines in a soil pattern file can be loaded with

```
bodenzuweisungen = BodenmusterdateiEinlesen(dateiname='bodenmuster.txt');
```

Each line represents a soil pattern with name, search pattern and path (separated by semikolons) unless the line is empty or starting with a comment character (#). If the file is read and at least one soil pattern is present, the corresponding files for this soil pattern can be loaded with

```
bodendaten = BodendatenMitSchluesselAusMusterEinlesen(muster=bodenzuweisungen);
```

The return value is only assigned a non-`None` value, if the read was successfull. This is the intended behaviour for one soil material, but it would also return `None` if any error occurs while processing multiple materials (and not returning the completely loaded materials). It is therefore recommended to iteratively read multiple files in a structure[3], so that each successfully loaded soil materials will be available for use.

```
bodendaten = Datenstruktur();
for schluessel in bodenzuweisungen.keys():
   tempboden = BodendatenMitSchluesselAusMusterEinlesen(muster=dict([(schluessel,
      bodenzuweisungen[schluessel])]));
   if (not (tempboden is None)):
      bodendaten.update(tempboden);
```

5.2 Soil patterns and regular expressions

If a soil material is to be loaded from its original files, a soil pattern is used (soil name, search pattern, path). While soil name and path are easy to adjust, the complexity of the regular expression depends on the naming/location of the files to be loaded. Ideally, if all files in the given path are belonging to one soil material and should be read by miniSoilLAB,

[2]The name of these files is expected to have ”locker“,”_lo“ or ”-lo“ in their name when the soil material is loose and ”dicht“,”_di“ or ”-di“ when it is dense. Using the long form is recommended to have a unique distinction. This method fails (and dense is assumed) if neither or both are present as in ”Testsand_Triax-D_lo_direkt_an_GOK.xls“.

[3]Instead of using `Datenstruktur()` a simple `dict()` can also be used. It will just display the results a bit differently.

the regular expression is empty. In the next best scenario, the files can be renamed/moved to have a consistent representation. However, very often the collection is either read-only or should be treated like that. Naming might not be as consistent as desired and even multiple soil materials belonging to one project might be present in the given path. With the right regular expression only the intended files will be used.

The following example shows some simple use cases for regular expressions. Consider the following files present in the directory specified by path:

```
Sand1.kvs
Lodi_sand1.xls
Sand1_Triax-D_dicht.xls
Sand1_Triax-D_locker.xls
Sand2_Triax-D_dicht.xls
Sand2_Triax-D_locker.xls
CRS_oedo-Sand1.xls
Auswertung_hypoplastisch_sand1.xls
```

- An empty regular expression will load all files. The file names suggest that both `Sand2_`-files might belong to another soil material (loading them anyway might not be intended).
- Using `Sand1` as regular expression will ignore both `Sand2`-files, but also two files with lowercase `sand1` (which might also not be intended).
- Using `[sS]and1` includes all files with either `sand1` or `Sand1` (this might also be achieved with `sand1|Sand1`).

Many regular expressions to match the intended files can already be created with `|` as ”or“ condition, `[]` as ”any one of“ and `[^x]` as ”not x“. More possible conditions and a thorough discussion of regular expressions can be found at »https://docs.python.org/3/library/re.html.

To check if all intended files are loaded with miniSoilLAB, the following code (with adjusted path and search pattern) can be used.

```
zielordner = '/pfad/zu/labordaten/';
liste = ZieldateienFinden(zielordner=zielordner, dateimuster='[Ss]and1')
for eintrag in liste:
   print(eintrag[len(zielordner):]);
```

5.3 Navigating the internal structure

miniSoilLAB is using a modified `dict` class named `Datenstruktur` to represent the internal data structure. All loaded data is saved in the internal structure as lists and numbers (and some informative text). Values can be obtained by their keys as usual, e. g. by `value = bodendaten[key]`. The differences/advantages of `Datenstruktur` objects are

- Returned keys are always sorted (as in `bodendaten.keys()`).
- When using `print()` on a `Datenstruktur` object, only one key-value pair will be printed per line

```
print(bodendaten);
```

- If the value (list) of a key has many entries, using `print()` only shows the first values and a continuation symbol, so that the screen is not cluttered.
- Using `print()` does not expand all substructures automatically (again to prevent cluttering). Only the key of the substructure will be printed.

These changes allow to easily browse through the structure in the console. It is therefore recommended to use `print()` when displaying data structures.

5.4 Plotting

Plots with `matplotlib` can also be created from the console. The same plot commands described for use in the GUI with *Manuelle Plotbefehlseingabe:* (see section 4.2) can also be used here. More precisely the GUI is forwarding the input to the functions `VordefiniertePlots()` and `VordefinierteMultiPlots()`. Consequently, a possible plot command can look like

```
VordefiniertePlots(boden=testbodentriaxd, plotname='dichter volumetrischer Triax-D Versuch');
```

Note that the command is converted to lowercase and only the keywords `triax-d`, `dicht` and `volume` are required/recognized. So this command is equivalent to

```
VordefiniertePlots(boden=testbodentriaxd, plotname='volume triax-d dicht');
```

In these examples `testbodentriaxd` represents the internal structure of a single soil material. If any required keywords are missing or a problem occured, a console output should be generated.

To plot data from multiple soil materials `VordefinierteMultiPlots` has to be used (which is invoked when the `alle:` keyword in the GUI). The function requires the data sets of all soil materials which should be plotted.

5.5 Determination of parameters

miniSoilLAB supports the determination of material parameters for one hypoplastic and viscohypoplastic constitutive model based on laboratory data. The hypoplastic constitutive model is described in von Wolffersdorff (1996). The extended hypoplastic parameters are used in the intergranular strain concept from Niemunis and Herle (1997). The viscohypoplastic constitutive model used is described in Niemunis (2002).

miniSoilLAB provides three functions to determine the parameters for these constitutive models (optional control variables to adjust the parameter determination are described in the function documentation). If all relevant data is present, typical function calls are

```
hypoparam = HypoplastischeParameterFuerBoden(boden=bodendaten, referenzspannungen=[50, 1200]);
erwhypoparam = ErweiterteHypoplastischeParameterFuerBoden(boden=bodendaten);
viskohypoparam = ViskohypoplastischeParameterFuerBoden(boden=bodendaten);
```

The following files (matched templates) are required for a material to determine a set of parameters:

- *hypoplastic parameters:* Auswertung-Hypoplastisch (and optionally all of LoDi, Oedo-CRS, Triax-D (locker), Triax-D (dicht))
- *extended hypoplastic parameters:* Triax-p-q
- *viscohypoplastic parameters:* Oedo-CRL, Oedo-CRS-Visko, Triax-CU

5.6 Saving data sets

The internal structure can be saved in JSON files, e. g. by using

```
DatensatzSpeichern(datensatz=bodendaten, dateiname='gespeichert.txt');
```

The saved data sets can be loaded in miniSoilLAB/Python or other programs afterwards. Some notes on using the data in Octave/Matlab can be found in section 7.

6 Templates

6.1 Concept

Templates allow to read differently organized spreadsheets representing certain laboratory tests (see section 5.1 for an overview and section 6.3 for details). The should be saved within a given template folder, but using subfolders is optional. It is also possible to use multiple templates for one laboratory test (e. g. if the format was changed or they have different origins). Templates for the same test are expected to have the same name with a different two digit sequence number in the file name and within the template structure. Spreadsheets are matched against the (alphabetically sorted) templates and uses the first match. It is therefore recommended to assign templates with more specific checks a lower number than more general templates.

The amount of different templates for one type of laboratory test should be kept as minimal as possible.

6.2 Structure of a template file

A template is a valid JSON file (with `json` suffix in the file name). The first and only child element should be the name of the template as the file name itself (with corresponding sequence number). The name has to be unique or the template may be ignored/overwritten (console output will show a warning).

Its children are the names of the tables in a spreadsheet, from which data is to be read, which will typically include checks, regular entries and substructures. Therefore the basic structure of a template is

```
{
   "Name of the template with two digit sequence number": {
      "Name of table sheet to read values from": {
         "[Checks]": {

            ... cell content comparisons ...

         },

         ... regular entries ...

         "Substructure": {

            ... regular entries in substructure ...

         },

         "[Gruppe group name]": {

            ... regular entries read as group ...

         }

      },

      ... possible other table sheets to read values from ...

   }
}
```

Checks

Before any data is read into the internal structure, two kinds of checks are performed to evaluate if the template matches the current spreadsheet. First, all names of table sheets defined in the template file must also be present in the spreadsheet. Second, for each table sheet, all entries defined in a called `[Checks]` are tested. In this block the keys represent valid cell identifiers (e. g. "B4" for the fourth cell in the second row) and the values represent which text is expected in this cell. They usually represent static text and can also be defined to be empty (" ").

Overall by having a unique structure of table sheets and checks, each supported spreadsheet should match exactly one template. If no template matches the spreadsheet will not be read at all and a note will be printed in the console output. Otherwise, all entries defined below `[Checks]` are read into the internal structure. There are regular entries and substructures/groups to consider.

Regular Entries

Regular entries define keys whose values store data from the spreadsheet. The keys represent the name of the variables and the values where to get the data from. Single cells can be defined (like "B15"), areas by the upper left and the lower right cell with a colon (like"C15:C30") or multiple cells/cell areas separated by a semikolon. Areas may only be defined along one column or one row, not both (i. e."B5:C8" is not allowed). Since variables of measurement data are often represented as columns (sometimes with variable length), areas can also be defined until the end of valid data by using "-1" as the last row identifier (e. g. "B6:B-1"). Values are interpreted as text unless a valid range (min./max. value) is given as a second value. The numbers also have to be within the given range (or a warning is issued and the values are not read at all).

If one or more cells are to be read, all values have to be valid or the whole list will not be saved in the internal structure. Sometimes it might be useful if all available data is read and cut automatically based on a defined range of values (e. g. when there might be invalid entries at beginning of measurement data). In this scenario an optional parameter `min_schnitt` can be specified as third value, so that all data before the first valid data greater than the minimum specified as min. value will simply be cut off (values outside of the given range after the first valid value will still drop all values).

Substructures and groups

Successfully read regular values will be saved as defined in the internal structure. Sometimes it might be useful (or necessary) to organize values in a given substructure. To create a substructure, simply add a new key and move the definitions for regular value(s) in it.

It is also possible to read multiple values which have to have the same length/interval. All values which should be read together have to be defined in a group with its name in square brackets after the keyword `Gruppe` (i. e. `[Gruppe group name]`). Although this is similar to defining a substructure, it will not create an additional structure and all values will be read into the structure the group is defined in.

The values of all members in a group will be read until an empty cell is found for at least one value. Groups also behave a bit different in regard to invalid values. If the values of at least one group member is outside the valid range, all previously read values of all group members will be discarded. The next valid value for all group members will be the first value to be read again. It is therefore important to ensure that group data does not end on invalid values (empty cells just stop the reading process for the group without discarding data).

6.3 Required values

The template files provided with miniSoilLAB read more values than strictly required (i. e. are not minimal). It is possible to include even more values in the templates which will be read into the internal structure and saved/loaded with it. But the following minimal structure/values are required for miniSoilLAB to work correctly.[1]

Atterberg

- `Wassergehalt [%]`
- `Ueberkornanteil [%]`
- `Daten-Fliessgrenze`
  - `Schlaege-Anzahl [-]`
  - `Feuchtmasse mit Behaelter [g]`
  - `Trockenmasse mit Behaelter [g]`
  - `Behaeltermasse [g]`
- `Daten-Ausrollgrenze`
  - `Feuchtmasse mit Behaelter [g]`
  - `Trockenmasse mit Behaelter [g]`
  - `Behaeltermasse [g]`

Remark: The members of each `Daten-...` group are lists with the same length (enforced by group structure in the default template).

Auswertung-Hypoplastisch

- `Schuettkegel`
  - `Korndichte [g/cm^3]`
  - `Porenzahl-max [-]`
  - `Porenzahl-min [-]`
  - `Reibungswinkel-krit [Grad]`
- `Oedo-locker`
  - `Masse [g]`

[1]Currently some values can also be defined with a different dimension (e. g. `mm` instead of `cm` or `N` instead of `kN`) which still has to be streamlined in the code to work for all values.

  - `Hoehe [mm]`
  - `Durchmesser [mm]`
  - `Setzung [mm]`
  - `Spannung [kN/m^2]`

- `Oedo-dicht`
  - `Masse [g]`
  - `Hoehe [mm]`
  - `Durchmesser [mm]`
  - `Setzung [mm]`
  - `Spannung [kN/m^2]`

- `Triax-D`
  - `Porenzahl-Peak [-]`
  - `Spannung-Peak-eff [kN/m^2]`
  - `Dilatanzwinkel [Grad]`
  - `Reibungswinkel-Peak-locker [Grad]`
  - `Reibungswinkel-Peak-dicht [Grad]`

- `Parameter`

Remark: Entry `Parameter` might be empty, but has to exist. `Setzung [mm]` and `Spannung [kN/m^2]` of each oedometric test are lists and have to have the same lengths (enforced by group structure in the default template).

LoDi

- `Korndichte [g/cm^3]`
- `Lockerste-Lagerung`
  - `Zylindervolumen [cm^3]`
  - `Probenmasse [g]`

- `Dichteste-Lagerung`
  - `Zylindergrundflaeche [cm^2]`
  - `Zylinderhoehe [cm]`
  - `Plattenhoehe [cm]`
  - `Probenmasse [g]`
  - `Setzung [g]`

Remark: `Setzung [g]` is expected to be a list, `Probenmasse [g]` of `Lockerste-Lagerung` can be either a value or a list.

Oedo

- `Korndichte [g/cm^3]`
- `Hoehe [mm]`
- `Durchmesser [mm]`
- `Trockenmasse mit Behaelter [g]`
- `Behaeltermasse [g]`
- `Kopfplatte`
  - `Last-ref [kg]`
  - `Spannung-ref [kN/m^2]`
- `Erstbelastung`
  - `Setzung [mm]`
  - `Spannung [kN/m^2]`

Remark: The entries `Entlastung` (and `Wiederbelastung`) can also be present but have to have the same structure as `Erstbelastung`. Especially each `Setzung [mm]` and `Spannung [kN/m^2]` thereof are lists and have to have the same lengths (enforced by group structure in the default template).

Oedo-CRL

- `Korndichte [g/cm^3]`
- `Hoehe [mm]`
- `Durchmesser [mm]`
- `Trockenmasse mit Behaelter [g]`
- `Behaeltermasse [g]`
- `Kopfplatte`
  - `Last-ref [kg]`
  - `Spannung-ref [kN/m^2]`
- `Erstbelastung`
  - `Setzung [mm]`
  - `Spannung [kN/m^2]`

- `Laststufe 1`
    - `Zeitwerte`
    - `Setzung [mm]`
    - `Laststufen [kN/m^2]`
- `Laststufe 2`
    - `Zeitwerte`
    - `Setzung [mm]`
    - `Laststufen [kN/m^2]`

Remark: The entries `Entlastung` (and `Wiederbelastung`) can also be present but have to have the same structure as `Erstbelastung`. Especially each `Setzung [mm]` and `Spannung [kN/m^2]` thereof are lists and have to have the same lengths (enforced by group structure in the default template).

The three entries listed for `Laststufe 1` and `Laststufe 2` have to be repeated for each additional loading step (`Laststufe 3`, ...). At least `Laststufe 1` and `Laststufe 2` are expected (up to `Laststufe 8` allowed). `Zeitwerte` and `Setzung [mm]` of each loading step are lists and have to have the same lengths (enforced by group structure in the default template).

Oedo-CRS

- `Zeitwert`
- `Oedo-dicht`
    - `Hoehe [mm]`
    - `Durchmesser [mm]`
    - `Masse [g]`
    - `Schergeschwindigkeit [mm/min]`
    - `Kraft [kN]`
    - `Weg [mm]`
- `Oedo-locker`
    - `Hoehe [mm]`
    - `Durchmesser [mm]`
    - `Masse [g]`
    - `Schergeschwindigkeit [mm/min]`
    - `Kraft [kN]`

  – `Weg [mm]`

Remark: `Kraft [kN]` and `Weg [mm]` of `Oedo-locker` and `Oedo-dicht` are lists and have to have the same lengths (enforced by group structure in the default template).

Oedo-CRS-Visko

- `Hoehe [mm]`
- `Durchmesser [mm]`
- `Korndichte [g/cm^3]`
- `Trockenmasse mit Behaelter [g]`
- `Behaeltermasse [g]`
- `Parameter`
- `Zeit [s]`
- `Kraft [kN]`
- `Stauchung [mm]`

Remark: Entry `Parameter` might be empty, but has to exist. `Zeit [s]`, `Kraft [kN]` and `Stauchung [mm]` are lists and have to have the same lengths (enforced by group structure in the default template).

Triax-CU

- `Trockenmasse mit Behaelter [g]`
- `Behaeltermasse [g]`
- `1-Probenherstellung`
  – `Hoehe [mm]`
  – `Durchmesser [mm]`
  – `Feuchtmasse [g]`
- `2-Saettigung`
  – `Zelldruck [kN/m^2]`
  – `Saettigungsdruck [kN/m^2]`
- `3-Konsolidation`
  – `Delta Hoehe [mm]`
- `Versuch 1`

  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Zeit [s] (oder Datum und Uhrzeit)

- Versuch 2
  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Zeit [s] (oder Datum und Uhrzeit)

- Versuch 3
  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Zeit [s] (oder Datum und Uhrzeit)

Remark: `Versuch 1`, `Versuch 2` and `Versuch 3` all have the same structure. All listed values are expected to be lists.

Triax-D

- Korndichte [g/cm^3]
- 1-Probenherstellung
  - Hoehe [mm]
  - Durchmesser [mm]
  - Trockenmasse [g]

- 2-Saettigung
  - Zelldruck [kN/m^2]
  - Saettigungsdruck [kN/m^2]
  - Backvolume-Ende [mm^3]

- 3-Konsolidation
  - Backvolume-Ende [mm^3]

- 5-Abscheren
  - Backvolume-Ende [mm^3]
- Versuch 1
  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Porenwasservolumen [mm^3]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Zeit [s] (oder Datum und Uhrzeit)
- Versuch 2
  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Porenwasservolumen [mm^3]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Zeit [s] (oder Datum und Uhrzeit)
- Versuch 3
  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Porenwasservolumen [mm^3]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Zeit [s] (oder Datum und Uhrzeit)

Remark: `Versuch 1`, `Versuch 2` and `Versuch 3` all have the same structure. All listed values except `Korndichte [g/cm^3]` are expected to be lists.

Triax-p-q

- Korndichte [g/cm^3]
- Parameter
- 1-Probenherstellung
  - Hoehe [mm]
  - Durchmesser [mm]

  - Trockenmasse [g]
- 2-Saettigung
  - Zelldruck [kN/m^2]
  - Saettigungsdruck [kN/m^2]
  - Backvolume-Ende [mm^3]
- 3-Konsolidation
  - Backvolume-Ende [mm^3]
- 5-p-q-Pfad
  - Segment 2
    - Porenwasserdruck [kN/m^2]
    - Druck-isotrop-eff [kN/m^2]
    - Hauptspannungsdifferenz [kN/m^2]
  - Segment 4
    - Porenwasserdruck [kN/m^2]
    - Druck-isotrop-eff [kN/m^2]
    - Hauptspannungsdifferenz [kN/m^2]
- Versuch 1
  - Stage
  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Porenwasservolumen [mm^3]
- Versuch 2
  - Stage
  - Radialdruck [kN/m^2]
  - Porenwasserdruck [kN/m^2]
  - Axialkraft [kN]
  - Stauchung [mm]
  - Porenwasservolumen [mm^3]
- Versuch 3
  - Stage

  - `Radialdruck [kN/m^2]`
  - `Porenwasserdruck [kN/m^2]`
  - `Axialkraft [kN]`
  - `Stauchung [mm]`
  - `Porenwasservolumen [mm^3]`

Remark: Entry `Parameter` might be empty, but has to exist. All other listed values except `Korndichte [g/cm^3]` are expected to be lists.

7 Octave and Matlab

Data sets saved as JSON files can also be read by Octave/Matlab. To import a data set saved as `testboden.json` run the following command in Octave/Matlab.

```
testboden = jsondecode(fileread('testboden.json'));
```

The data will be read as a *struct* and and can be navigated with the dot-notation. Since Matlab does not allow as many special characters in identifiers as Python some entries might have a slightly different name. For example the key `Porenzahl [-]` in the JSON file will become `Porenzahl___` in Matlab.

Bibliography

Niemunis, A. (2002). “Extended hypoplastic models for soils”. Habilitation. Ruhr-Universität Bochum.

Niemunis, A. and I. Herle (Oct. 1997). “Hypoplastic model for cohesionless soils with elastic strain range”. In: *Mechanics of Cohesive-Frictional Materials* 2.4, pp. 279–299. doi: »10.1002/(SICI)1099-1484(199710)2:4<279::AID-CFM29>3.0.CO;2-8.

von Wolffersdorff, P.-A. (1996). “A hypoplastic relation for granular materials with a predefined limit state surface”. In: *Mechanics of Cohesive-Frictional Materials* 1.3, pp. 251–271. doi: »10.1002/(SICI)1099-1484(199607)1:3<251::AID-CFM13>3.0.CO;2-3.