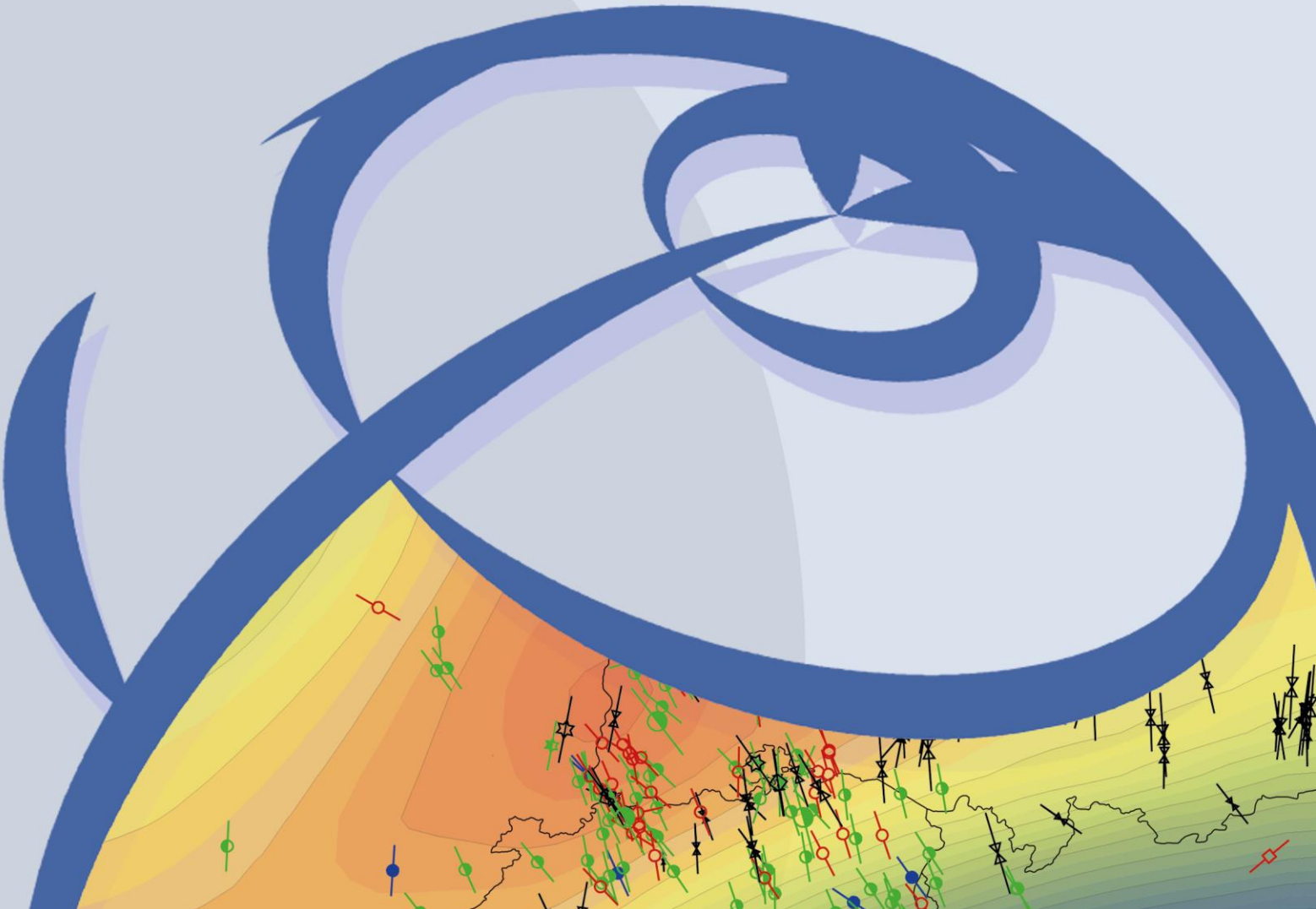
WSM WORLD STRESS MAP

WSM Technical Report 19-01

Manual of the Python Script Apple PY v1.0

Moritz Ziegler, Malte Ziebarth and Karsten Reiter



Recommended citation of the report:

Ziegler, M. O., Ziebarth, M., and Reiter, K. (2019). Manual of the Python Script Apple PY v1.0. World Stress Map Technical Report 19-01, GFZ German Research Centre for Geosciences.
DOI: http://doi.org/10.2312/wsm.2019.001

The software described in this report, including data examples, is published as:

Ziegler, M. O., Ziebarth, M., and Reiter, K. (2019). Python Script Apple PY v1.0. GFZ Data Services.
DOI: http://doi.org/10.5880/wsm.2019.001

Imprint

World Stress Map Project
GFZ German Research Centre for Geosciences

Telegrafenberg
D-14473 Potsdam
Published in Potsdam, Germany
February 2019
http://doi.org/10.2312/wsm.2019.001

WSM Technical Report 19-01

Manual of the Python Script Apple PY v1.0

Moritz Ziegler, Malte Ziebarth
GFZ German Research Centre for Geosciences, Potsdam, Germany
Karsten Reiter
Institute of Applied Geosciences, Technical University Darmstadt

GFZ
Helmholtz-Zentrum
POTSDAM

Table of Contents

List of Tables

List of Figures

1 Abstract

In geosciences the discretization of complex 3D model volumes into finite elements can be a time-consuming task and often needs experience with a professional software. Especially outcropping or out-pinching geological units, i.e. geological layers that are represented in the model volume, pose serious challenges. Changes in the geometry of a model may occur well into a project at a point, when re-meshing is not an option anymore or would involve a significant amount of additional time to invest.

In order to speed up and automate the process of discretization, Apple PY (Automatic Portioning Preventing Lengthy manual Element assignment for PYthon) separates the process of mesh-generation and unit assignment. It requires an existing uniform mesh together with separate information on the depths of the interfaces between geological units (herein called horizons). These two pieces of information are combined and used to assign the individual elements to different units. The uniform mesh is created with a standard meshing software and contains no or only very few and simple structures. The mesh has to be available as an Abaqus input file. The information on the horizons depths and lateral variations in the depths is provided in a text file. Apple PY compares the element location and depth with that of the horizons in order to assign each element to a corresponding geological unit below or above a certain horizon.

The script files are provided for download at http://github.com/MorZieg/APPLE_PY. Table 0-1 gives an overview of the folder structure and input files with a short explanation.

Table 0-1 Structure of the GitHub repository

The folders and files in the GitHub repository http://github.com/MorZieg/APPLE_PY. The page numbers (if available) direct to the documentation in this manual.

File Name	Explanation
apple.py	Main script with auxiliary functions. See page 6.
create_horizon_file.py	Additional script that converts single horizon depth files into readable input for Apple PY. See page 10.
CITATION.bib	The recommended citation for the software.
LICENSE	The full GPL v3.0 license text.
README.md	Readme file that contains relevant information on the usage of the software.
examples/	Folder that contains example files for both scripts (Table 0-2).

Table 0-2 Structure of the folder “examples”

Short explanation of the files provided for an exemplified usage of create_horizon_file.py and apple.py. Geometry is stored in Abaqus input files (.inp) and the horizon depths in text files (.txt). The example geometry for create_horizon_file.py is stored in text files (.dat).

Folder and File Name	**Explanation**
horizon_example/ 01_Quaternary.dat 02_Cretaceous.dat 03_Jurassic.dat 04_Basement.dat	Single horizon files as example for create_horizon_file.py. See page 10.
simple/ simple.inp simple.txt	Apple PY input files for a simple scenario. See page 15.
salt_dome/ salt_dome.inp salt_dome.txt	Apple PY input files for a scenario that includes a salt dome. See page 16.
graben/ graben.inp graben.txt horst.txt	Apple PY input files for a horst and graben scenario. See page 17.
intrusion/ intrusion.inp intrusion.txt	Apple PY input files for a scenario with a hard-coded intrusion. See page 19.
mixed_elems/ mixed_elems.inp mixed_elems.txt	Apple PY input files for different element types. See page 20.

1 Introduction

The finite element method (FEM) is often used for the solution of partial differential equations in geosciences due to two key advantages: It allows unstructured meshes to discretize complex model volumes and it allows to assign individual sub-volumes (usually geological units) different rock properties. However, the discretization of model volumes into finite elements can be time consuming. This holds especially for complex stratigraphic environments with faults and several geological units, i.e. geological layers represented in the model volume that have different rock properties. It is further complicated if these units pinch out, crop out, have a large variability in thickness, or are very thin in some places.

The standard procedure is that geological units (sub-volumes) are discretized separately and that the interface between two geological units is coincident with the location of nodes of the finite elements (Figure 1-1). This can be time-consuming when the number of geological units (sub-volumes) is large or their geometry complex. Still, a perfect connectivity of all nodes is required in the final finite element mesh.

With Apple PY (Automatic Portioning Preventing Lengthy manual Element assignment for PYthon) an alternative is provided which can speed up the mesh generation significantly. The previously mentioned standard workflow is turned upside-down by first creating the mesh and then assigning the finite elements to geological units (Figure 1-2).

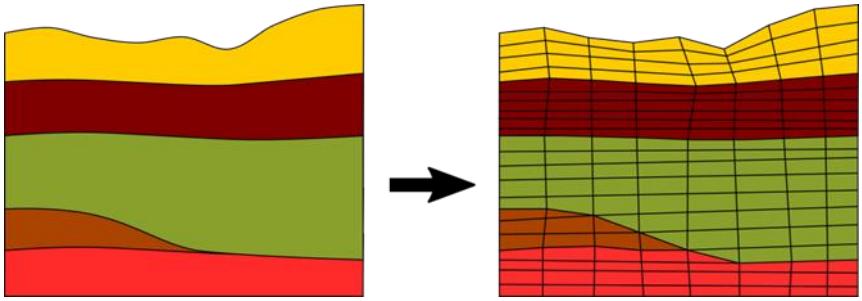


Figure 1-1 Standard approach to the discretization of geonumerical models

The subsurface geometries are directly used as control surfaces for discretization.

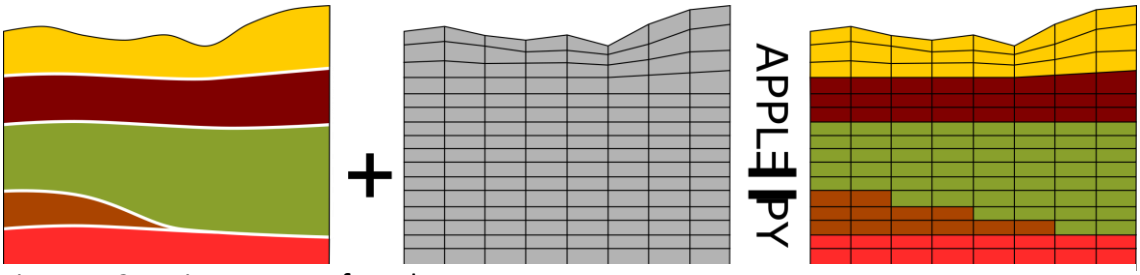


Figure 1-2 Basic concept of Apple PY

The geological model and a uniform mesh of the entire model volume are provided. Apple PY combines the information and assigns the elements to the corresponding rock units.

2 Concept

A typical 3D geological model geometry comprises of different geological units of interest distinguished by properties relevant for the addressed aspects of the geomechanical-numerical model. This lithology is based on various data sources such as wellbore data, geological maps, seismic lines, 3D seismic surveys, or results from other geophysical sounding methods. The information from all available datasets is harmonised and a 3D geologic model is created. This is often an evenly spaced lateral grid which contains information on the depth of the interfaces between the geological units (herein called horizons) at each grid point.

In the traditional approach to numerical modelling of the subsurface, finite elements constitute the model volume between horizons. Therefore, the relevant subsurface geometries are introduced as surfaces and volumes in a 3D geologic model. Then these entities are discretized using specialized software (Figure 1-1). However, the construction and discretization of the individual units is cumbersome, leads to errors, and takes a lot of time, especially if the model contains faults, units that crop or pinch out, or thin units that display great variations in thickness. A lot of labour intensive manual work is required in order to get acceptable results.

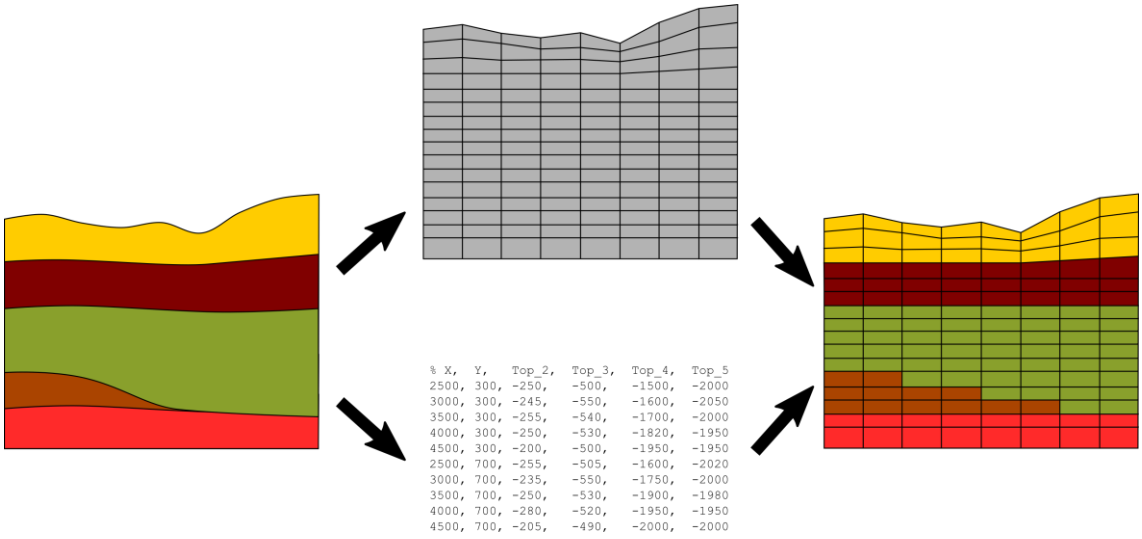


Figure 2-1 Apple PY approach

Bottom: The geologic model is transferred to a text-file that describes vertical profiles. Top: The entire model volume is uniformly meshed as a single sub-volume and provided as an Abaqus input file. Apple PY uses the information from these two files and provides the final mesh with assigned geological units.

Apple PY is designed to provide an easy and fast discretization approach that limits the manual work to the absolute minimum. Therefore, the traditional approach is split up in two parts (Figure 2-1). That means that the entire model volume is discretized in a single instance. No sub-volumes or geological units are assigned individually. In addition, the

3D geological model is processed in a way that the horizons that separate the sub-volumes can be represented by the text file in Apple PY syntax. Then, Apple PY uses these information to assign the elements to the according sub-volumes or geological units.

In other words, the Apple PY approach directly uses the 3D geological model in its configuration as a lateral grid with depth information to populate single elements in the geomechanical-numerical model with appropriate material properties. It does not require an individual discretization of each geological unit. This saves time and effort in the discretization process. Exemplified comparison between the traditional approach and Apple PYs approach are presented in Figure 2-2.

To use Apple PY the following three steps are executed: (1) A text file that contains the depth and variation of the interface between relevant geological units (herein called horizon) throughout the model is created. Such a file can be exported from some geological modelling software. Alternatively, the tool create_horizon_file.py is used. (2) A mesh that may contain as few as only one sub-volume/element set (and potential faults) is created. (3) Apple PY uses the information provided by the horizons file to assign the elements of the mesh to the corresponding rock units.

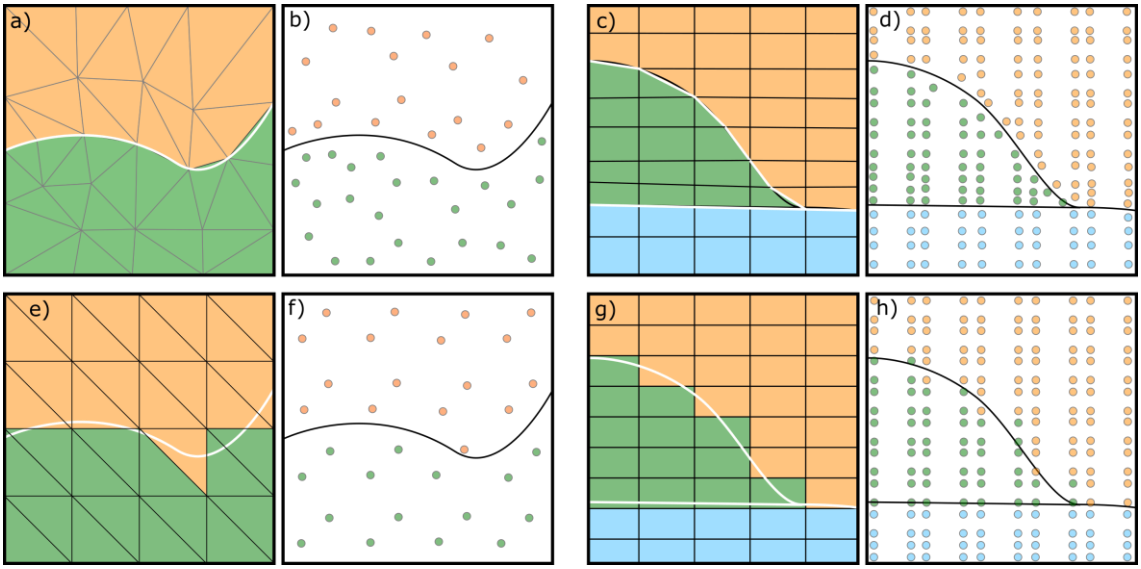


Figure 2-2 The elements and the resulting distribution of integration points

The traditional tetrahedron approach (a, b), the traditional hexahedron approach (c, d), the proposed tetrahedron approach (e, f), and the proposed hexahedron approach (g, h).

3 Application of the tool

Apple PY can be downloaded from https://github.com/MorZieg/Apple_Py. As a Python 2.7 script several functions required for script execution are provided in one file. The additional Python packages *numpy*, *scipy.spatial*, and *collections* are required. The general execution procedure is as follows:

1. The available geometry is reviewed and the input file for Apple PY with the horizon depths is created. The additional script create_horizon_file.py can be used (See page 10). The grid spacing is assessed.
2. The model volume is discretized as a single volume/element set in a standard meshing software. The geometry of potential faults is included. If applicable the topography is included as an undulating surface.
3. The entire model is discretized with a mesh size that corresponds both laterally and vertically to the grid size of the provided geologic information. Apple PY does not assess whether the resolution of the mesh compared to the horizon data is reasonable. Therefore, the user has to ensure that a sufficient high vertical resolution is provided.
4. The mesh is exported from the pre-processor as an Abaqus input file format (*.inp).
5. The user-defined variables are set in the Python file.
6. Apple PY is run in the command line by navigating to the folder that contains the script and the input files and executing `python apple.py`. An output file that contains element sets that correspond to the geological units is created.
7. Optionally the resulting mesh can be reviewed in Abaqus CAE or meshing tools such as Hypermesh. In addition to the input mesh the Apple PY output file is imported. Then the elements from the sets are individually assigned to different components.
8. The Apple PY output file is included in the Abaqus input file after the geometry definition (*NODES and *ELEMENTS) with the command `*include,INPUT=elements.set` and the analysis is run.

3.1 User input

Apple PY requires two input files: The mesh as an Abaqus input file (*.inp) and the depths of the horizons in a standard text file. Both files have to provide data in the same coordinate system. Furthermore, the resolution of the mesh should correspond to the resolution of the provided horizon data (Figure 3-1). In order to adjust the mesh size to the horizon data it is beneficial to use a regular grid for the horizons. Even though, Apple PY also works with unevenly distributed data. In addition, the files are required to fulfil certain criteria concerning their syntax and structure that are discussed in section 3.2 and 3.3.

In addition to the two input files which are explained in the following, the user needs to set five variables that control the input and execution of the script. The variables are contained in the main function of the Python script and are designed as shown in Table 3-1.

Table 3-1 Description of variables in Apple PY

Variable	Description
geometry	The name (and path) to a file that contains the input geometry in Abaqus input format. Example: `geometry = 'simple_geometry.inp'`
horizons	The name (and path) to the file that contains the information on the horizon depths. Example: `horizons = 'geom_test.txt'`
strata	The names of the geological units that are represented in the 3D geological model and the horizon file from top to bottom. Example: `strata = ('Unit_1','Unit_2','Unit_3')`
twodelem	Controls if 2D elements are also processed or if they are omitted. This is especially useful when 2D faults are present in the model. Example: `twodelem = 'omit'` or `'yes'`
elems_exclude (optional)	The names of rock units (element sets) that should be ignored by Apple PY. If you want to process all element sets comment this line out. Example: `elems_exclude = ('basement','intrusion')`
fname	The filename for the output element set. If a file with the same name already exists in the directory it will be overwritten without any prior notification. Example: `fname = 'elements.set'`

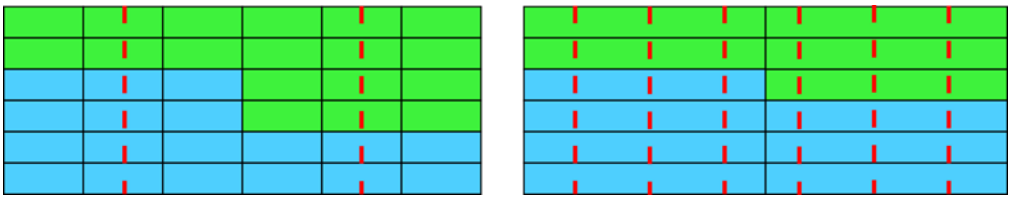


Figure 3-1 Mesh with two assigned units and corresponding horizon profiles

Vertical horizon profiles are red dashed vertical lines, element edges are indicated by black lines, and different units are blue and green. Left: A resolution of 6 x 6 elements. The resolution of the mesh is too fine since the spacing between the horizon profiles is clearly larger than the element size. Right: A resolution of 2 x 6 elements. The resolution of the mesh is too coarse. Within each element more than one vertical horizon profile is situated.

3.2 Mesh input file

The input file for the mesh is required to be in Abaqus *.inp file format. If the file is exported e.g. from Hypermesh usually Apple PY works out of the box. In the Abaqus input file four types of lines relevant for Apple PY that can be distinguished:

1. Comment lines are marked by `**` and are ignored by Apple PY
2. Keyword lines define nodes, elements, sets, etc. and begin with `*` and a following keyword. Only the keyword lines for node (`*NODE`) and element (`*ELEMENT`) definitions are regarded here.
    a. `*NODE` definitions usually occur only once at the beginning of the file and only one keyword line is present.
    b. `*ELEMENT` initiates the definition of element in sets and follows the `*NODE` definitions. They occur more or less frequently, depending on the number of element sets and element types present in the model. Apple PY requires at least one element set but supports more than one. In the keyword line the name of the element set and the type of elements defined (brick elements, tetrahedrons, wedges) is described.
3. Data lines follow keywords. They contain the information that is extracted by this script.
    a. A node is defined by comma-separated numbers: the node ID followed by the x, y, and z coordinates.
    b. An element is defined by comma-separated numbers: The element number is followed by a number of node IDs according to the number of nodes required for such an element type.
4. The file end is marked by `*****`. If the end of the file is not marked Apple PY will end up in an infinite loop.

An example for an input file follows:

```
**   Template:  ABAQUS/STANDARD 3D
**
*NODE
      1,    0.0,  1.0,  0.0
      2,    1.0,  1.0,  0.0
...
*ELEMENT,TYPE=C3D8R,ELSET=Quaternary
      100,  1,    2,    3,    4,    5,    6,    7,
      8
      101,  5,    6,    7,    8,    11,   12,   13,
      14
...
*ELEMENT,TYPE=C3D4,ELSET=Cretaceous
      200,  51,   52,   53,   54
```

```
      201,  55,   56,   57,   58
...
*ELEMENT,TYPE=S3,ELSET=fault
      300,  51,   52,   53
      301,  55,   56,   57
*****
```

Apple PY works with either uniform or refined meshes. Furthermore, the common 3D element types are supported (e.g. tetrahedron, brick, pyramid, or wedge). Per default Apple PY requires only a single set of elements as input. However, it also processes several element sets, such as in a model that is already portioned into different rock units and may also contain faults or intrusions. In order for Apple PY to work as designed, the following requirements of the input file are crucial:

- Each node and element definition needs to end without a comma. If a comma is used, the software assumes that the definition of the element continues in the next line.
- In order to be recognised as such, 2D elements have to be assigned an element type that begins with `TYPE=S3` or `TYPE=S4`.
- If entire element sets should be disregarded (elems_exclude) their name should be exactly as defined in the variable and initiated by `ELSET=`. No blank space is allowed in between the term `ELSET=` and the name.

3.3 Horizon input file

The depth of the interface between geological units (herein called horizons) is read from a text file. The data is provided in the comma separated text file as a point cloud. Each line represents a lateral point defined by the first two values x (easting) and y (northing). The following numbers indicate the depth of each horizon from top to bottom. Note that every horizons depth needs to be specified at each lateral point.

In case of discontinued units two possibilities exist. (1) If the unit crops or pinches out towards the top it is assigned a depth of 9999 in areas where it is discontinued. (2) If the unit pinches out towards the bottom it is assigned the depth of the next deeper horizon in areas where the original unit is not present. An example of the syntax of the input file that corresponds to the stratigraphy in Figure 3-2 is provided here:

```
% x,  y,    Top1, Top2, Top3,      Top4
1,    0,    -1,   -2,   -10000,    -10000
2,    0,    -1,   9999, -4,        -4
3,    0,    -1,   9999, -3,        -4
1,    1,    -1,   -2,   -4,        -4
2,    1,    -1,   -2,   -4,        -4
3,    1,    -1,   9999, -4,        -4
```

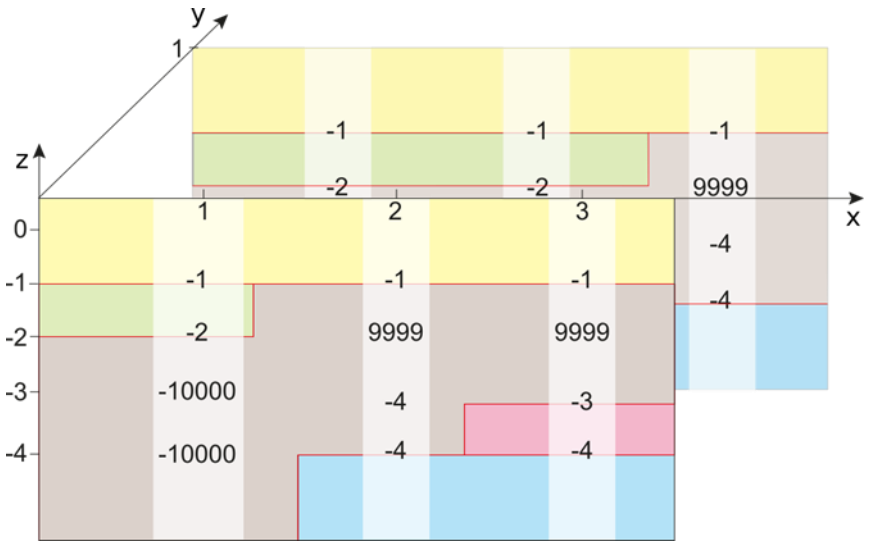


Figure 3-2 Description of the stratigraphy with Apple PY syntax

3.4 Create horizon file (optional)

Often software used for geological modelling does not provide the option to export a file that fulfils the exact syntax required for the horizon file. In the following section the auxiliary script create_horizon_file.py is presented that merges several files that contain depth information of different horizons. The output is suitable for Apple PY. The execution of the script requires the following input information (Table 3-2).

Table 3-2 Description of variables in create_horizon_file.py

Variable	Description
files	The names (and paths) to the files that contain the single horizons from top to bottom. Example: `files = ['Quaternary.txt','Cretaceous.txt']`
comment	The character that indicates a comment line in the horizon files. Lines that are beginning with a comment character are ignored. Example: `comment = '#'`
fname	The name of the output file that is suitable for Apple PY. Example: `fname = 'horizons.txt'`

The input files that each defines a single horizon necessarily have the same grid spacing and resolution. If horizons with different resolutions should be used they are either interpolated to a common resolution in external software or Apple PY is run twice or even more often. Furthermore, especially for large models, it is beneficial for the runtime of create_horizon_file.py if the grid points are in the same order. In addition, it is beneficial (but not necessary) that outcropping or out-pinching horizons in areas where they are not present are also assigned to grid points and depth according to the Apple PY scheme. Generally, horizons that are not specified at a certain grid point are assumed to be not present at this grid point.

The input file requires at least the lateral coordinates (Easting and Northing) and the depth of the horizon. Additional columns in the file are ignored. An example for a possible input file is provided in the following:

```
# Cretaceous
# Easting Northing Depth Row Column
732500.00 5316500.00 -743.97 1  1
732550.00 5316500.00 -744.59 1  2
732600.00 5316500.00 -745.22 1  3
```

Furthermore, a working example that responds to the default input in create_horizon_file.py and corresponds to Figure 3-2 is available.

create_horizon_file.py executes the following routine in order to create an input file for Apple PY.

1. The uppermost horizon file is read into the variable `horizons` and the variable is converted to a Python *numpy* array.
2. The following steps are executed for each horizon once.
    a. A new column is added to the `horizons` variable. The column is filled with the value 9999.
    b. The file is read line-by-line. For each line the coordinates and depth are extracted and the following action ensues.
        i. In the optimal case the coordinates of the grid point in the current line match the coordinates of the current row in the `horizons` variable. Then the depth value is added to the new column of the `horizons` variable.
        ii. If the coordinates are not the same, the matching coordinates are sought in the `horizons` variable and the depth is added to the new column of the `horizons` variable.

iii. If the coordinates do not exist yet in the `horizons` variable the coordinates and depth of the current line are added as a new row to the `horizons` variable. All previous horizons are assigned the depth 9999 as it is assumed that they are not present above the current horizon.

3. The depth values in each row in the `horizons` variable are processed from bottom to top. In each row until the first unit has a depth other than 9999 (i.e. not present) the arbitrary value of -100000 is assigned. This is done in order to make sure that Apple PY will not assign any element to a unit that is not present at the bottom. Please note that create_horizon_file.py does not use 9999 in the final horizon file. The output of Apple PY is not affected thereby.
4. The `horizons` variable is written to the file suitable for Apple PY.

4 Script description

Apple PY is a Python script which consists of a main function and four auxiliary functions. The objective is to provide a fully automated assignment of elements from a finite element mesh to various geological units based on the location and depth of the element and pointwise geological information.

4.1 Reading geometry

The script reads the Abaqus input file that contains the discretized geometry (the mesh) and extracts the nodes and element definitions. Therefore, the two additional functions `nodearray` and `elemarray` which are provided in the Python file are called. The names of element sets that are loaded are printed to the screen. In order to distinguish between the data lines, it is mandatory for the mesh input file to strictly follow the syntax.

In some models 2D elements exist, e.g. active faults. In such case these elements should remain in the original element set and should not be distributed to another set by Apple PY. This is done by specifying `twodelem = 'omit'` which prevents any 2D elements of type S3 or S4 and derivatives such as S4R to be loaded. (The script searches for element types with a definition that starts with S3 or S4.) The names of excluded 2D element sets are printed to the screen with an according note.

Furthermore, (2D or 3D) element sets which should not be sorted and reassigned can be specified. For instance, if in an uncertainty analysis only some horizons should be varied while the others are fixed, the element sets that should not be reassigned are specified in the variable `elems_exclude`. The names of excluded element sets are printed to the screen with an according note.

In the next step the file that specifies the depth of the horizons is loaded to a variable. Then it is verified that the number of geological units in the horizon file is in correspondence to the number of specified horizons (i.e. the number of units equals the number of horizons plus one). If this is not the case an error is printed to the screen and the script is stopped.

4.2 Assign elements to rock units

For each node in the numerical model the closest horizon depth information is sought. Therefore, the location of the node in 2D (defined by x and y coordinates) is compared to the x and y locations of the depth information in the horizon file. Each line in the horizons file defines a vertical 1D profile. Each node is assigned to the closest 1D horizon profile (Figure 4-1a).

Each element is characterized by a certain number of nodes. The 1D profile that occurs most often among the constituting nodes of an element is chosen to be representative for the element (Figure 4-1b). If two or more 1D profiles occur with the same frequency, the profile that is listed first in the input file is assigned to the element. In addition, for each element a mean depth is computed using the depths of the constituting nodes (Figure 4-1b).

Elements are then assigned to a certain rock unit depending on the mean depth of the element and the depth of the horizon as specified by the 1D profile that is most common for this element. This is done by the function `assign_elems`. For each element the function iterates over the number of horizons. For each horizon depth other than the value of 9999 it is tested whether the mean depth of the element is smaller than the horizon depth. If that is not the case the according geological unit is assigned. In the very unlikely case that the mean depth of the element is exactly equal to the depth of the horizon the element is assigned to the upper geological unit.

Eventually the affiliation of the elements to different rock units is exported using Abaqus element sets. Each set contains the numbers of the elements that belong to a certain rock unit. The sets bear the name of the rock unit and `_new` is added. This is helpful in distinguishing the new element sets from potentially previously existing element sets. The element sets are exported in a single output file which is added to the Abaqus input file with the command `*include,INPUT=file.set` by the user. The `*solidsection` command in the input file needs to be adapted accordingly to direct the material properties to the newly defined element sets.

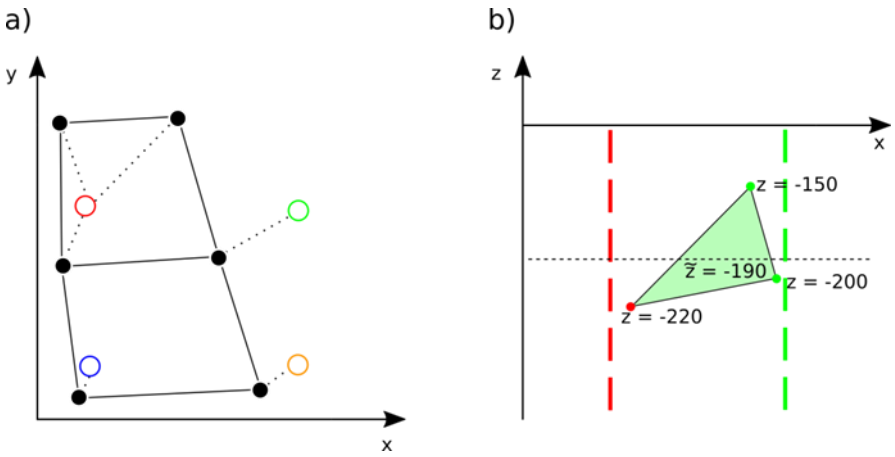


Figure 4-1 Assignment of vertical profiles and horizon depth to elements

Nodes and elements in a 2D view. a) Each node (black point) is assigned the laterally (x-y plane) closest 1D profile (coloured circle). b) A single element in x-z view that consists of three nodes and two 1D profiles (dashed lines). The element is assigned a mean depth (dashed black line) dependent on the depths of the nodes and the 1D profile that is assigned to the majority of the nodes (green).

5 Examples

In the following several examples are provided in order to show the features of Apple PY. Most examples are semi-2D with only a single column of 3D elements in one lateral direction. This is to facilitate the presentation and increase the understanding. The Abaqus input files and horizon files are provided as supplementary material to the script. The script settings, the horizon definitions, and the result are shown in the following. If not stated otherwise only a single element set is present in the initial model.

5.1 Simple units

The first example provides an overview of the most simple and basic operations comparable to the syntax presentation in Figure 3-2. Results are shown in Figure 5-1.

```
geometry = 'simple.inp'
horizons = 'simple.txt'
strata = ('dark_blue','green','yellow','light_blue','pink')
twodelem = 'yes'
#elems_exclude = ()
fname = 'simple_units_elements.set'


0.5, 0.5, -1,   -2, -3, -4
1.5, 0.5, -1,   -2, -3, -4
2.5, 0.5, -0.7, -2, -3, -4.3
3.5, 0.5, -0.6, -2, -3, -4.5
4.5, 0.5, -0.5, -2, -3, -4.7
5.5, 0.5, -0.3, -2, -3, -4.9
6.5, 0.5, -0.1, -2, -3, -5.1
7.5, 0.5, 9999, -2, -3, -5.3
8.5, 0.5, 9999, -2, -3, -5.5
9.5, 0.5, 9999, -2, -3, -5.7
```

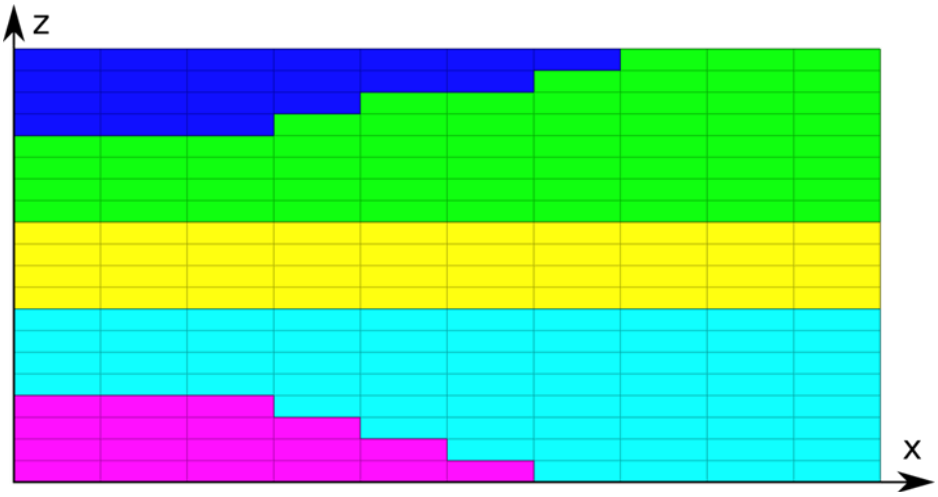


Figure 5-1 Example: Simple stratigraphy

x-z plane view of the pseudo-2D example for simple outcropping and outpinching units.

5.2 Salt dome

In case of a salt dome structure the challenge of the thin chimney with a broader “head” is addressed by using two different units for the salt dome: one for the lower part and the chimney (light blue) and the other for the “head” (dark blue). In Abaqus both units are assigned the same rock properties. Results are shown in Figure 5-2.

```
geometry = 'salt_dome.inp'
horizons = 'salt_dome.txt'
strata = ('green','dark_blue','yellow','light_blue','pink')
twodelem = 'yes'
#elems_exclude = ()
fname = 'salt_dome_elements.set'

0.5, 0.5, -2,   -2,   -3.5,-4
1.5, 0.5, -2,   -2,   -3.5,-4
2.5, 0.5, -1.5, -2.8, -3.5,-4
3.5, 0.5, -1,   -2.5, -3.5,-4
4.5, 0.5, -1,   9999, -1,  -4
5.5, 0.5, -1,   9999, -1,  -4
6.5, 0.5, -1,   -2.5, -3.5,-4
7.5, 0.5, -1.5, -2.8, -3.5,-4
8.5, 0.5, -2,   -2,   -3.5,-4
9.5, 0.5, -2,   -2,   -3.5,-4
```

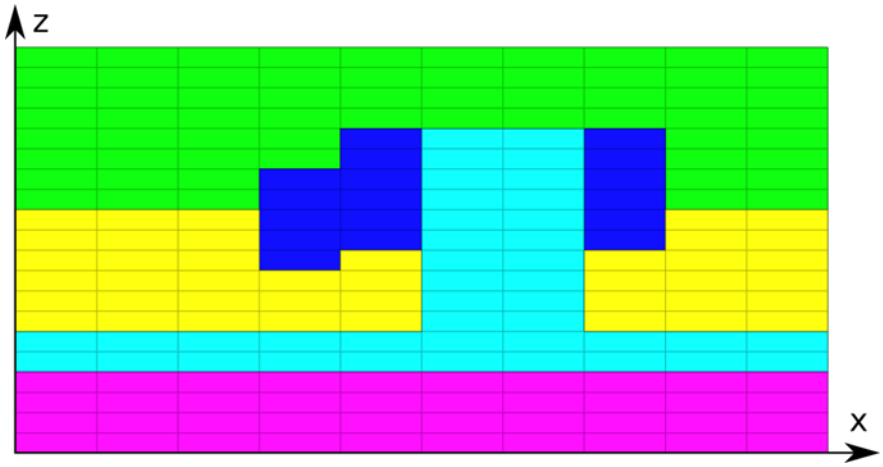


Figure 5-2 Example: Salt dome

x-z plane view of the pseudo-2D example for a salt dome. The salt layer is coloured dark and light blue.

5.3 Graben structure

The challenge in the case of a graben structure is to assign the elements at the boundary faults correctly if the fault displays a vertical throw. Therefore, two initial element sets, horst and graben, are created (Figure 5-3) and Apple PY is run twice. In the first run only the units within the graben are assigned, in the second run only the units of the horst are assigned (Figure 5-4). Two different horizons files are required and two different element files are created. Note that the element file produced by the first run will be overwritten by the second run if they have the same name.

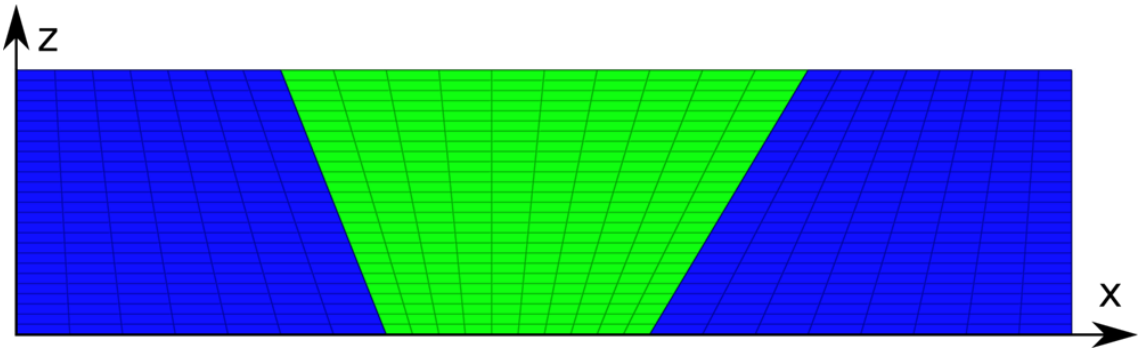


Figure 5-3 Example: Initial graben structure

x-z plane view of the initial element sets in a pseudo-2D example for a graben structure.

First run:

```
geometry = 'graben.inp'
horizons = 'graben.txt'
strata = ('red_graben','bordeaux_graben','blue_graben')
twodelem = 'omit'
elems_exclude = ('horst')
fname = 'graben_elements.set'


1,    0.5, -2, -4
3,    0.5, -2, -4
5,    0.5, -2, -4
7,    0.5, -2, -4
9,    0.5, -2, -4
11,   0.5, -2, -4
13,   0.5, -2, -4
15,   0.5, -2, -4
17,   0.5, -2, -4
19,   0.5, -2, -4
```

Second run:

```
geometry = 'graben.inp'
horizons = 'horst.txt'
strata = ('yellow_horst','blue_horst','pink_horst')
twodelem = 'omit'
elems_exclude = ('graben')
fname = 'horst_elements.set'


1,    0.5, -1, -3
3,    0.5, -1, -3
5,    0.5, -1, -3
7,    0.5, -1, -3
9,    0.5, -1, -3
11,   0.5, -1, -3
13,   0.5, -1, -3
15,   0.5, -1, -3
17,   0.5, -1, -3
19,   0.5, -1, -3
```

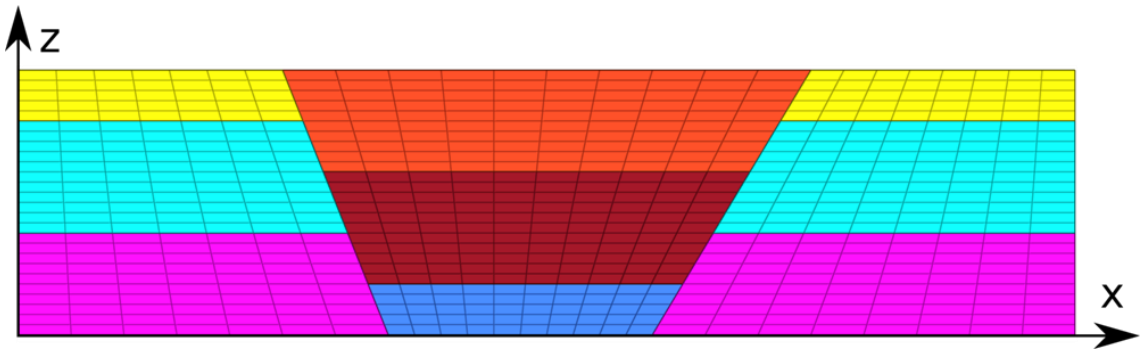


Figure 5-4 Example: Graben structure

x-z plane view of the final element distribution in a pseudo-2D example for a graben structure.

5.4 Intrusion & basement

In this example the basement and an intrusion are “hard-coded” in the mesh, i.e. their geometry is considered while building the mesh (Figure 5-5). In addition, three Units are present which are not regarded while building the mesh. They are included using Apple PY (Figure 5-6).

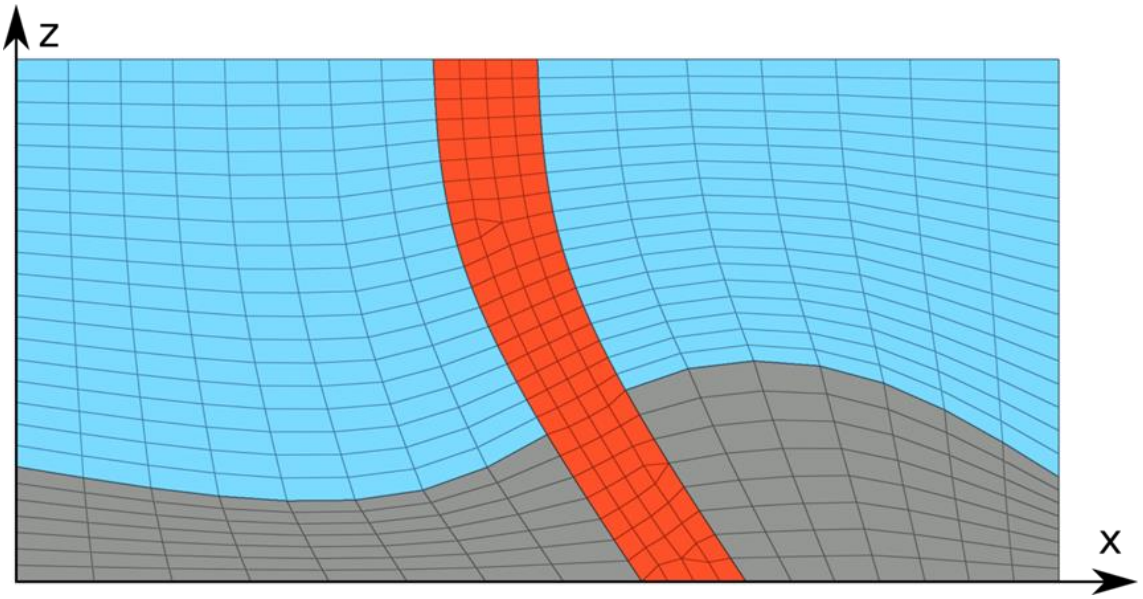


Figure 5-5 Example: Initial intrusion

x-z plane view of the initial mesh in a pseudo-2D example for hard-coded and soft-coded stratigraphy.

```
geometry = 'intrusion.inp'
horizons = 'intrusion.txt'
strata = ('blue','pink','green')
twodelem = 'omit'
elems_exclude = ('basement','intrusion')
fname = 'intrusion_elements.set'
```

```
0.5, 0.5, -1,   -2.5
1.5, 0.5, -0.9, -2.5
2.5, 0.5, -0.8, -2.75
3.5, 0.5, -0.7, -3
4.5, 0.5, -0.6, -3.5
5.5, 0.5, -0.5, -3.5
6.5, 0.5, -0.4, -5
7.5, 0.5, -0.3, -5
8.5, 0.5, -0.2, -5
9.5, 0.5, -0.1, -5
```

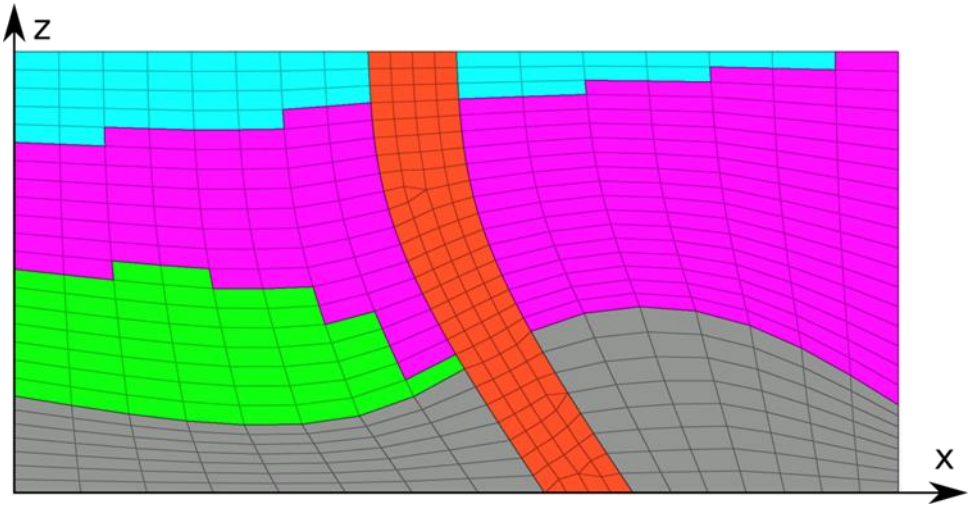


Figure 5-6 Example: Final intrusion

x-z plane view of the final mesh in a pseudo-2D example for hard-coded and soft-coded stratigraphy.

5.5 Mixed elements 3D

In this 3D example several different element types (hexahedron, wedge, and tetrahedron) are in use.

```
geometry = 'mixed_elems.inp'
horizons = 'mixed_elems.txt'
strata = ('green','blue','pink','red','brown')
twodelem = 'omit'
#elems_exclude = ()
fname = 'mixed_elements.set'


0.5, 1,    -1, -2,    -4, -4
1.5, 1,    -1, -2,    -4, -4
2.5, 1,    -1, 9999,  -3, -4
3.5, 1,    -1, 9999,  -3, -4
4.5, 1,    -1, 9999,  -3, -4
0.5, 2.5,  -1, -2,    -4, -4
1.5, 2.5,  -1, -2,    -4, -4
2.5, 2.5,  -1, -2,    -3, -4
3.5, 2.5,  -1, 9999,  -3, -4
4.5, 2.5,  -1, 9999,  -3, -4
0.5, 4,    -1, -2,    -4, -4
1.5, 4,    -1, -2,    -4, -4
2.5, 4,    -1, -2,    -4, -4
3.5, 4,    -1, -2,    -4, -4
4.5, 4,    -1, -2,    -3, -4
```

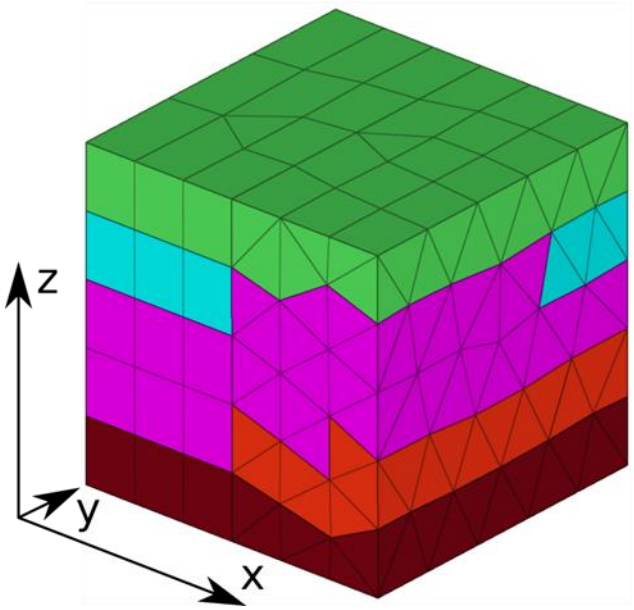


Figure 5-7 Example: Mixed element types

3D view of an example for different element types with outpinching units.

6 Acknowledgement

The script founds on the Lego Technique used by Tobias Hergert und Karsten Reiter. The authors would like to thank Solveig Strutzke for typesetting and Kirsten Elger for final checks and supporting the publication.

The work leading to these results has received funding from Germany's Federal Ministry for Education and Research under the FONA/GEO:N programme (Project SUBI) and from the Initiative and Networking Fund of the Helmholtz Association through the project "Integrity of nuclear waste repository systems - Cross-scale system understanding and analysis (iCross)". The content is the sole responsibility of the authors and it does not represent the opinion of the Helmholtz Association, and the Helmholtz Association is not responsible for any use that might be made of the information contained.