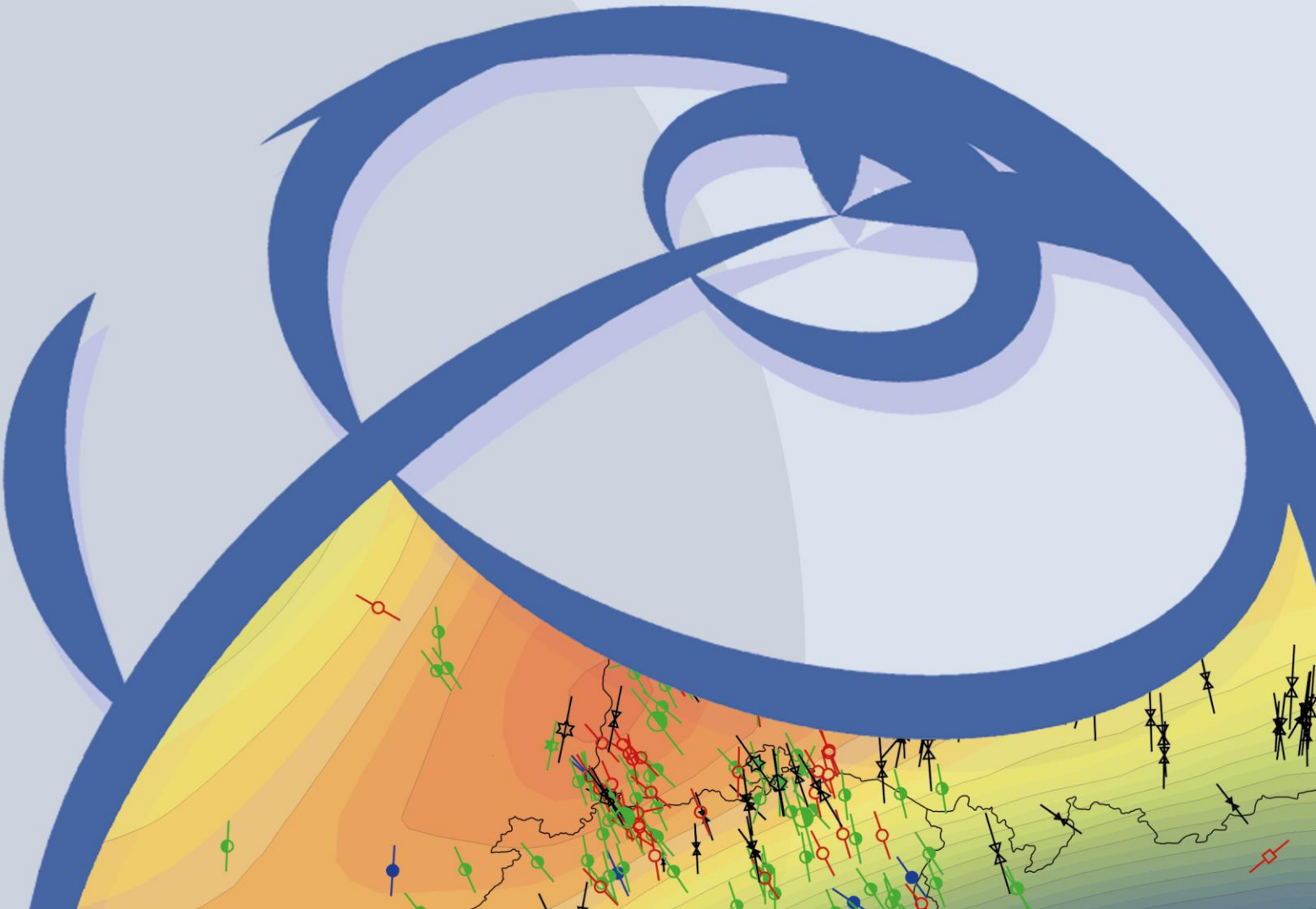
WSM WORLD STRESS MAP

WSM Technical Report 23-01

Manual of the Python Script FAST Estimation *v1.0*

Moritz O. Ziegler



Recommended citation of the report:

Ziegler, M. O. (2023). Manual of the Python Script FAST Estimation v1.0. World Stress Map Technical Report 23-01, GFZ German Research Centre for Geosciences.
DOI: https://doi.org/10.48440/wsm.2023.001

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

Ziegler, M. O. (2023). Python Script FAST Estimation v1.0. GFZ Data Services. DOI: https://doi.org/10.5880/wsm.2023.001

The software is available for download on GitHub:

https://github.com/MorZieg/FAST_Estimation

Imprint

World Stress Map Project
GFZ German Research Centre for Geosciences

Telegrafenberg
D-14473 Potsdam
Published in Potsdam, Germany
February 2023
https://doi.org/10.48440/wsm.2023.001

WSM Technical Report 23-01

Manual of the Python Script FAST Estimation v1.0

Moritz O. Ziegler
GFZ German Research Centre for Geosciences, Potsdam, Germany

GFZ
Helmholtz Centre
POTSDAM

Table of Contents

List of Tables

List of Figures

Abstract

The classical way to model the stress state in a rock volume is to estimate displacement boundary conditions that minimize the deviation of the modelled stress state with respect to model-independent stress information such as stress magnitude data. However, these data records are usually subject to significant uncertainties and measurement errors. Hence, it has to be expected that not all stress magnitude data records are representative and can be used in a model. In order to identify unreliable stress data records, the stress state that is based on individual data records is solved and compared with observations at a few discrete locations.

While this method works, it is not efficient in that most of the solved model scenarios will be discarded. The solving of the entire model consumes immense amount of computation time for a high-resolution model. Yet, the stress state is required at only a very limited number of locations. For linear geomechanical models it is sufficient to estimate the stress state from three model scenarios with arbitrary, but different displacement boundary conditions. These three results can be used to estimate analytically using a linear regression at discrete points stress states based on user-defined boundary conditions.

The tool Fast Automatic Stress Tensor Estimation (FAST Estimation) is a Python function that automatizes this approach. FAST Estimation provides very efficiently the stress states at pre-defined locations for all possible boundary conditions. It does not provide the continuous stress field as provided by a solved geomechanical model. Instead, it is a cost-efficient solution for the rapid assessment of stress states at a limited number of discrete locations based on pre-defined boundary conditions.

The script files are provided for download at:

http://github.com/MorZieg/FAST_Estimation

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.

Folders/Files in GitHub repository http://github.com/MorZieg/FAST_Estimation. The page numbers (if available) direct to the documentation in this manual.

File Name	Explanation	Page
fast_estimation.py	Python script for the estimation of the stress state at discrete locations in a model based on the applied boundary conditions.	5
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 an estimation using either Moose or Abaqus.	16
cellcent2nodal.mcr	Tecplot macro file that converts cell-centred to nodal variables.	7
rename_stress.mcr	Tecplot macro that renames the stress state variables from a Moose solver to be compatible with the Tecplot Add-on GeoStress.	7

1 Introduction

3D geomechanical-numerical models provide a continuous description of the stress state in a rock volume of interest. Besides the implementation of the gravitational volume forces the lateral surface forces are imposed by displacement boundary conditions. In a rectangular model these are applied normal and parallel to the prevailing horizontal stress orientations at the model boundaries. The boundary conditions are adapted in a way that the resulting stress state minimizes the deviation with respect to stress magnitude data at discrete points within the model volume (Hergert et al., 2015; Reiter & Heidbach, 2014; Ziegler & Heidbach, 2021a).

To find the best-fit model scenario a large number of model scenarios that only differ in their displacement boundary condition need to be solved. In addition, when also the uncertainties of the stress magnitude data are considered the number of model scenarios increases further (Ziegler and Heidbach, 2021a) and for models with high spatial resolution the computational time needed becomes an issue. Furthermore, it is inefficient since it has to be expected that many of the model scenarios will be discarded right away for being unrealistic.

However, due to the linear elastic rheology it is sufficient to estimate the stress state from three model scenarios with arbitrary, but different displacement boundary conditions. These three results span in the stress space planes that can be used to estimate analytically at discrete points stress states that are a result of all possible boundary conditions using a linear regression. The tool FAST Estimation (Fast Automatic Stress Tensor Estimation) is using this approach to increase speed and efficiency in the estimation of the best-fit model scenario.

FAST Estimation is a derivative of the model calibration approach FAST Calibration (Ziegler & Heidbach, 2021b). The tool runs in Python 3.x in conjunction with the visualization software Tecplot 360 EX 2019 R1 and its Add-on GeoStress (Heidbach et al., 2020; Stromeyer et al., 2020). It supports both Abaqus output files and output databases from the Moose Framework. A fully automatized application of FAST Estimation is possible using the solvers Abaqus or the Moose Framework together with PyTecplot, the Python extension of Tecplot. The user should be familiar with 3D geomechanical-numerical modelling, Python, Tecplot 360 EX, including a basic knowledge of Tecplot 360 EX macro functions, and the Tecplot 360 EX Add-on GeoStress provided by Stromeyer et al. (2020) and documented by Heidbach et al. (2020).

This FAST Estimation manual provides an overview of the scripts and is designed to help the user to adapt the scripts for their own needs. FAST Estimation is briefly validated in Section 2 and basic guidance sufficient to successfully run FAST Estimation are provided in Section 3. The mathematical background is described in Section 4. Section 5 provides information on the individual Python functions that come with FAST Estimation and is mainly dedicated to advanced users who might want to reuse or adapt functions. Eventually, common errors and pitfalls are addressed (Section 6) and basic usage examples are provided (Section 7).

2 Validation of the approach

FAST Estimation aims at a massive reduction of computation time by reducing the amount of model scenarios that have to be solved. Instead of solving, the stress state at a discrete number of locations is estimated in Python using a linear regression analysis. This significantly speeds up the process. However, it needs to be ensured that no significant errors or uncertainties are introduced by this approach. Therefore, the results of FAST Estimation are validated in the following.

Three generic models with increasing complexity are used in order to validate the accuracy of FAST Estimation. Each generic model has a volume of 10 x 10 x 5 km³ and each is solved for three combinations of arbitrary, but different displacement boundary conditions parallel to the x and y axes of the rectangular model. The resulting three stress states are the basis for FAST Estimation to setup a linear regression. The equation system is setup for 100 random locations throughout the model volume where the three sets of boundary conditions and the resulting stress states are available. Then, the stress state that results from application of another set of reference boundary conditions is estimated at the 100 locations. To test the accuracy of this approach the obtained estimated stress state is compared to a stress state that results from model solution provided by a solver that is using the same reference kinematic boundary conditions.

The differences between the actually solved stress states and the stress states estimated by FAST are visualized using boxplots for each component of the stress tensor. Both, the six independent components of the 3D stress tensor and the reduced stress tensor with only four independent components are regarded. The orientation of the maximum horizontal stress S_{Hmax} is left out in the comparison.

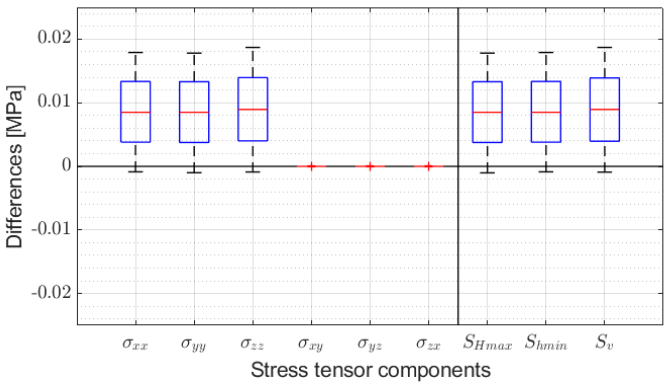


Figure 2-1 Accuracy of FAST estimation for a one-unit model

Differences between the classically modelled stress state and the stress state using FAST estimation for a model with homogeneous rock properties.

The first generic model has homogeneous linear elastic rock properties. A maximum difference of < 0.02 MPa is observed for the normal stress components of the stress tensor and the reduced stress tensor with a median around 0.008 MPa (Figure 2-1). No differences are observed for the shear stress components of the stress tensor. In the regarded homogeneous setting the shear components are expected to be very small to a point where they can be neglected.

The second generic model contains five lithological units with different rock properties that are partly out-cropping, out-pinching, and folded (Figure 2-2). The differences between modelled stress state and stress state estimated by FAST for such a realistic model geometry are indicated in Figure 2-2. While the differences for the normal stress tensor components and the reduced

stress tensor are significantly increased to < 0.04 MPa, the medians remain almost the same with around 0.01 MPa. It is noteworthy that now differences exist for the shear stress components. Even though they are still negligibly small, both in absolute values and in comparison to the normal components.

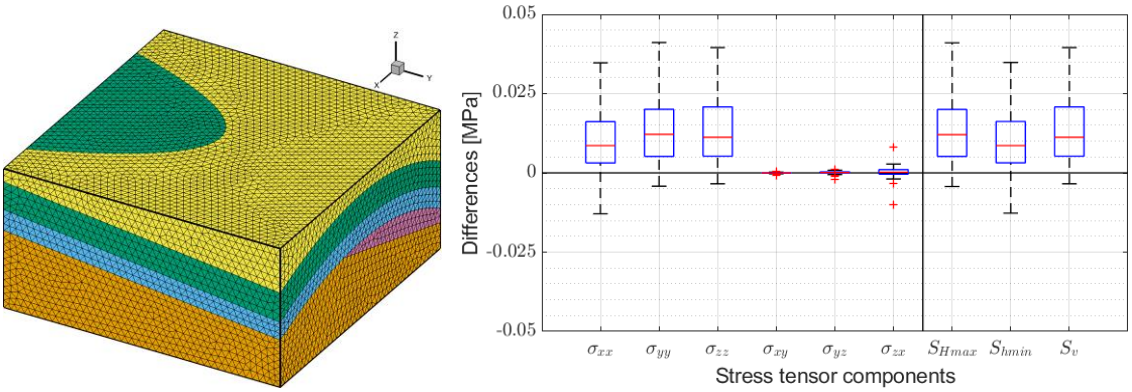


Figure 2-2 Accuracy of FAST Estimation for a realistic model

Left: Generic model setup with five units with different linear elastic rock properties (colour-coded). Right: Differences between classically modelled stress state and the stress state using FAST Estimation.

The third generic model is using arbitrarily distributed rock properties (Figure 2-3). The three elastic rock properties, i.e. the Young's modulus E, the Poisson ratio ν, and the Density ρ, are randomly assigned from a broad distribution using HIPSTER v1.3 (Ziegler, 2019, 2022). This results in a maximum heterogeneity of the model (Figure 2-3). The resulting differences indicate no significant increase in the median and the whiskers of the boxplots remain < 0.05 MPa. However, the outliers are up to < 0.1 MPa. Furthermore, the shear components of the stress tensor are also significantly more affected but still smaller than the normal components. Nevertheless, FAST Estimation succeeds in providing generally good results, in particular in lights of the magnitude of uncertainties in the stress state which is at least one magnitude higher.

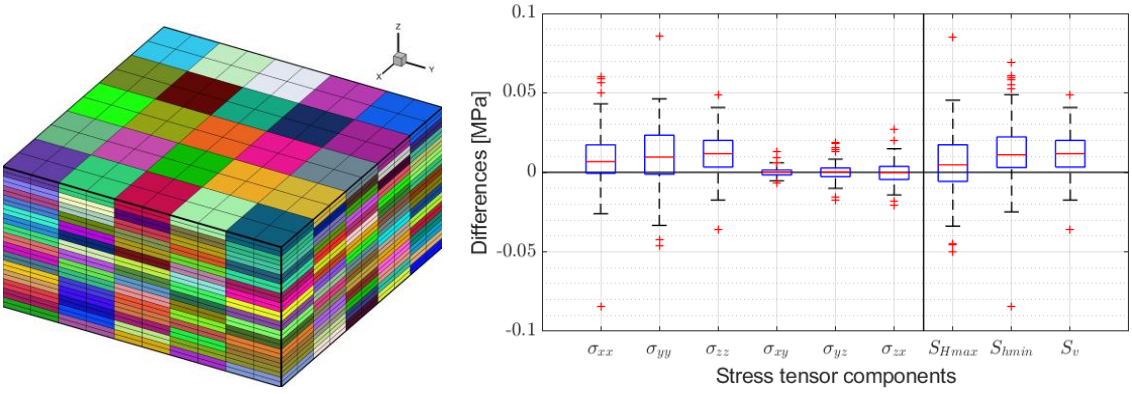


Figure 2-3 Accuracy of FAST Estimation for a maximum complexity

Left: Generic model with randomly distributed rock properties (colour-coded). Right: Differences between classically modelled stress state and the stress state using FAST Estimation.

3 Application of FAST Estimation

FAST Estimation runs with full functionality both on Windows and Linux computers and supports output files from two solvers – Abaqus and the Moose Framework. A full automatization is achieved using the commercial Python extension of Tecplot – PyTecplot. As an alternative to PyTecplot, as an intermediate manual step the user needs to run a Tecplot macro. FAST Estimation is designed and tested for usage with Python 3.11, Tecplot 360 EX 2019 R1, and PyTecplot 1.4.2. In particular, if newer Tecplot and PyTecplot versions are used adaptations may be required.

As a Python 3 script several functions required for script execution are provided in one file, which can be copied and configured for each project. Alternatively, FAST Estimation is used as a Python package with all required variables transmitted by the caller function.

3.1 General procedure

The FAST Estimation approach requires at least one set of displacement boundary conditions, one in x and one in y direction, to solve for a stress state at one location specified in x, y, and z model coordinates. Furthermore, information on the test boundary conditions (three different sets of boundary conditions), the name of the solved model with test boundary conditions, and (if PyTecplot is not used) the full path of the working folder are required. Whether the full stress tensor (all six independent components), the reduced stress tensor (S_{Hmax}, S_{hmin}, and S_v), or only individual components are estimated is left to the user. A compilation of all required variables and examples is presented in Table 3-1.

FAST Estimation in connection with Tecplot and PyTecplot supports output files from both Abaqus and the Moose Framework. In case of Abaqus two different output files have to be supported, depending on whether Tecplot is run on a Windows (*.odb file) or Linux (*.fil file) operating system. For some combinations of OS, Solver, and whether PyTecplot is used or not, certain additional aspects need to be considered which are explained in the following subsection.

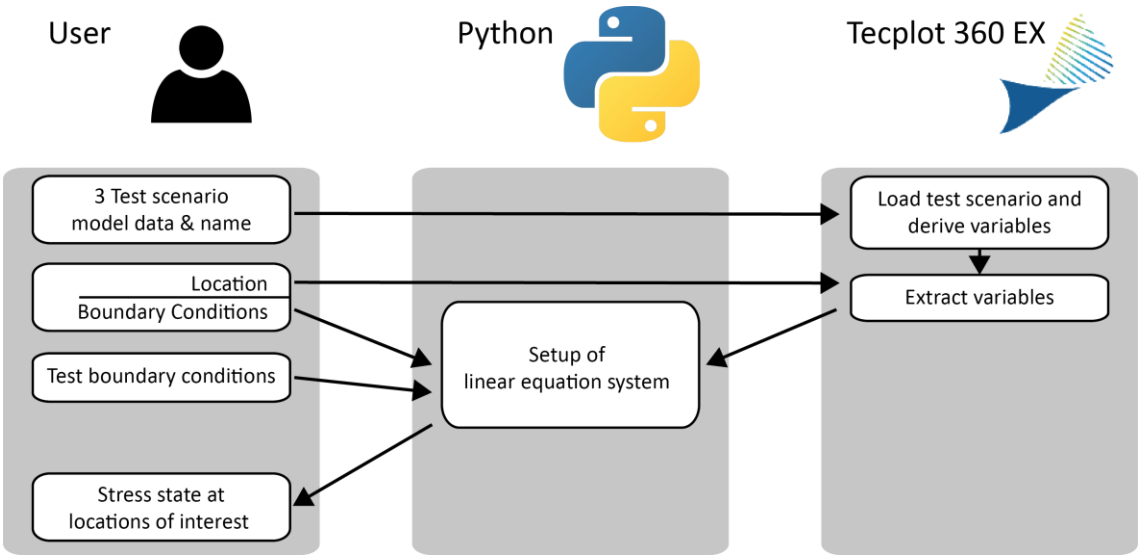


Figure 3-1 FAST Estimation procedure using PyTecplot.
Three columns indicate the three involved parties: The user, Python and Tecplot.

The application of FAST Estimation using PyTecplot works automatically from loading the output database files to the computation and display of boundary conditions in a single step illustrated in Figure 3-1. Most procedures specific to certain operating systems and solvers are implemented. Nonetheless, if running in a Linux environment with an Abaqus solver, the command for the generation of a *.fil file needs to be included to the Abaqus input file. This is done by including the following lines at the end of Abaqus input files first *STEP section.

```
*EL FILE
S
*NODE FILE
COORD, U
```

Table 3-1 Input variables required by FAST Estimation.
Examples are provided in the script file.

Variable	Description
folder (without PyTecplot)	Provide the directions to the folder in your system which contains the script data. It is important to include the full path for Tecplot 360 EX (which does *not* support relative paths). Example (Windows): folder = 'd:\\Data\\Project\\FAST\\Test' Example (Linux): folder = '/home/user/Project/FAST/Test'
name	Provide the file name of the output database file with three independent stress states derived from three different boundary conditions. This is also the name of the data-files that will be created by the Tecplot 360 EX macro. Example: name = 'test_scenarios'
bcs	Enter the displacements in x' (S_{hmin} or S_{Hmax}) and y' (S_{Hmax} or S_{hmin}) direction that are prescribed at the different test scenarios. Make sure to assign the values in the correct order, i.e. as in the solver. Example: bcs = [[4, -4],[4,-5],[5, -5]]
stress_vars	Provide the names of the stress variables in the model that are estimated. Note that the variables are case-sensitive. Example: stress_vars = ['SHmax','Shmin', 'Sv']
loc	Location where the stress state is estimated. If the variable is a string, it is assumed that it specifies a *.csv file which contains the coordinates. Example: shmax = [[5050,-3500,-800],[8000,2500,-3402]]
bc_eval	Boundary conditions used to estimate the resulting stress state at the locations specified in loc. If the variable is a string, it is assumed that it specifies a *.csv file which contains the boundary conditions. Example: bc_eval = [[-4, 3],[-2, -2.5]]
solver	The name of the used solver, i.e. either Abaqus or Moose Example: solver = 'abaqus'
pytecplot	Whether PyTecplot should be used or not. If set to 'off' a macro for the use in Tecplot 360 EX is created. Example: pytecplot = 'off'

3.2 Specific procedure without PyTecplot

The application of FAST Estimation without using PyTecplot requires several steps (including two steps performed within the Python script) which are illustrated in Figure 3-2.

1. Prepare your working directory with a subfolder called “data” and set the variables in the FAST Estimation script.

2. Load the model output database in Tecplot.

   Abaqus/Linux Instead of loading the standard *.odb file, only the *.fil file is supported by the Tecplot loader on Linux systems. To generate a *.fil file include the following lines in the Abaqus input files first *STEP section before running the model.

   ```
   *EL FILE
   S
   *NODE FILE
   COORD, U
   ```

   Moose The output database from the solver comes in three consecutive files each for one boundary condition scenario. These files need to be loaded consecutively into one Tecplot database. Make sure to load the files in the correct order.

3. Optionally run GeoStress (Heidbach et al. 2020) or a similar tool.

   Abaqus/Linux The stress tensors variables in the *.fil file are cell-centered instead of nodal variables as required for a successful application of GeoStress. Running the supplemented Tecplot macro *cellcent2nodal.mcr* converts the stress tensor variables to nodal variables. This has to be done before execution of GeoStress. Otherwise running GeoStress will result in an error.

   Moose The stress tensor output variables of Moose are called stress_xx etc. in contrast to XX Stress etc. expected by GeoStress. The same holds for the displacements that are called disp_x etc. in contrast to X Displacement. Running the supplemented Tecplot macro *rename_stress.mcr* fixes this issue so that GeoStress works. If this step is omitted, the GeoStress GUI fails to load.

4. Run FAST Estimation in the working directory. A Tecplot macro is written to your working directory. You will be prompted to execute the macro in Tecplot.

5. Run the macro in Tecplot to export the modelled stress state from the test scenarios at the locations of the stress data records to *.csv files in the data folder.

6. Press enter in the Python command window. The *.csv files will be loaded to Python variables and the stress states according to the specified boundary conditions are computed and written to the file estimated_stress_states.dat.

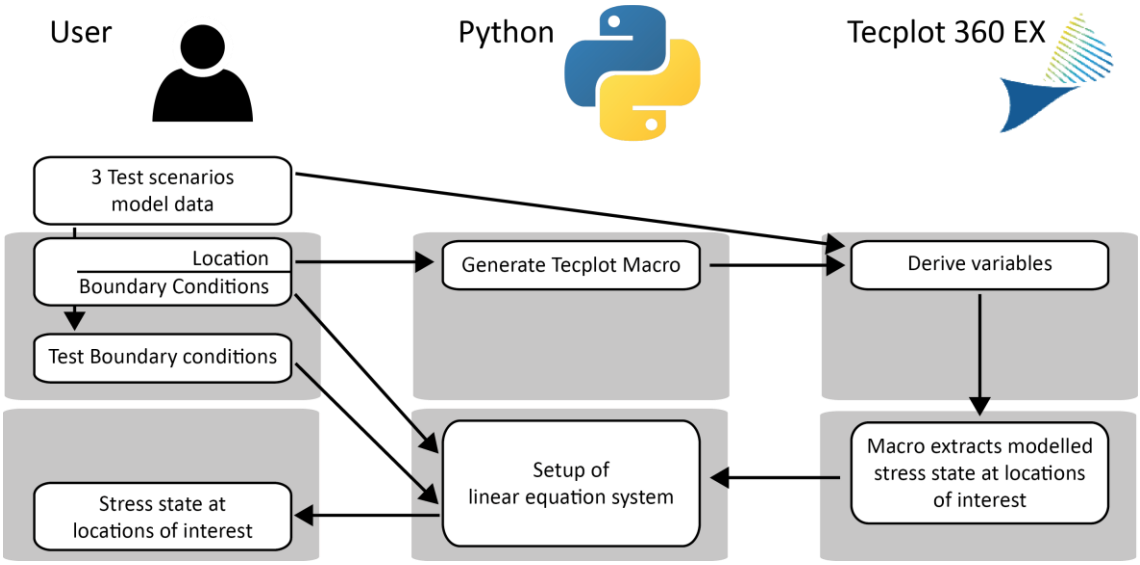


Figure 3-2 FAST Estimation procedure including an intermediate manual step.

Three columns indicate the three involved parties: The user, Python and Tecplot. The two steps are indicated by the two vertical boxes in each of the three columns.

3.3 Output

The output is either returned to the caller function as a variable or written to the file estimated_stress_state_YYYMMDD_HHMMSS.dat. The timestamp ensures that the file is not overwritten by a subsequent execution of FAST Estimation. Both, variable and file are structured following the boundary condition scenarios as follows and shown in the example:

1. Boundary displacements in x and y direction.
2. Is a stress rotation observed? If yes, at how many of the specified locations?
3. The components of the stress state in lines according to the locations.

The output variable can be used for an assessment of the stress state in a caller function. The output file can be read manually or automatically by another script or function.

```
Stress states estimated by FAST Estimation v1.0
Stress components: SHmax Shmin Sv
# Boundary Condition Scenario # 1
-4.400000, 2.000000, SHmax azimuth remains.
# Estimated Stress States
9.8989, 30.3757, 79.6037,
-8.3961, 12.0808, 24.7179,
#
# Boundary Condition Scenario # 2
…
```

4 Basic principle

The core of FAST Estimation is implemented in the function *solve* that sets up a system of linear equations which are used to estimate the stress state at a certain location as a result of user-defined boundary conditions. Therefore, the boundary conditions (bc) and resulting stress states (exemplified $\mathrm{S_{Hmax}}$) at this location $(\mathrm{x, y, z})$ of three different combinations of boundary conditions are required. An example of the available information is shown in Table 4-1.

Table 4-1 Example for the information required for FAST Estimation.

Three model scenarios with different boundary condition are required. For each scenario the displacement in x' and y' and $\mathrm{S_{Hmax}}$ are available.

model scenario	x' displacement	y' displacement	S_{Hmax}
1	-10	5	-12.3
2	-30	25	2.5
3	-30	5	-14.4

Three vectors are now available, one for each of the three model scenarios that each takes the form

$$\overrightarrow{v} = \begin{pmatrix} x \\ y \\ S_{Hmax} \end{pmatrix}$$

with the boundary displacement x in x' direction and y in y' direction as well as $\mathrm{S_{Hmax}}$, the resulting stress state.

It can be pictured that this information is used to set up the equation of the plane that is defined by the displacement boundary conditions in x' and y' direction and the stress magnitude. Hence, the planes are set up in e.g. $\mathbb{R}^3(\mathrm{x'}, \mathrm{y'}, \widehat{\Delta \mathrm{S}}_{\mathrm{Hmax}})$ (Figure 4-1). The planes equation in parameter form is

$$\vec{x} = \vec{p} + s\,\overrightarrow{r_1} + t\,\overrightarrow{r_2}$$

with the position vector $\vec{p} = \overrightarrow{v_1}$, the parameters s and t, and the direction vectors $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ which are computed by

$$\overrightarrow{r_1} = \overrightarrow{v_2} - \overrightarrow{v_1}$$
$$\overrightarrow{r_2} = \overrightarrow{v_3} - \overrightarrow{v_1}$$

Then the parameter form is transferred to the coordinate form of the planes equation which is defined as

$$n_1\,x' + n_2\,y' + n_3\,S_{Hmax} = d$$

with $\vec{\mathrm{n}}$ as the normal vector of the plane derived by

$$\vec{n} = \overrightarrow{r_1} \times \overrightarrow{r_2}$$

and d as

$$d = \vec{p} \cdot \vec{n}$$

with $\vec{\mathrm{p}}$ the planes position vector.

Then the planes equation can be transferred to solve for the stress magnitude and thus reads

$$S_{Hmax} = (d - n_1 x' - n_2 y') n_3{}^{-1}$$

Which enables estimation of the stress magnitude when the desired boundary conditions x' and y' are included.

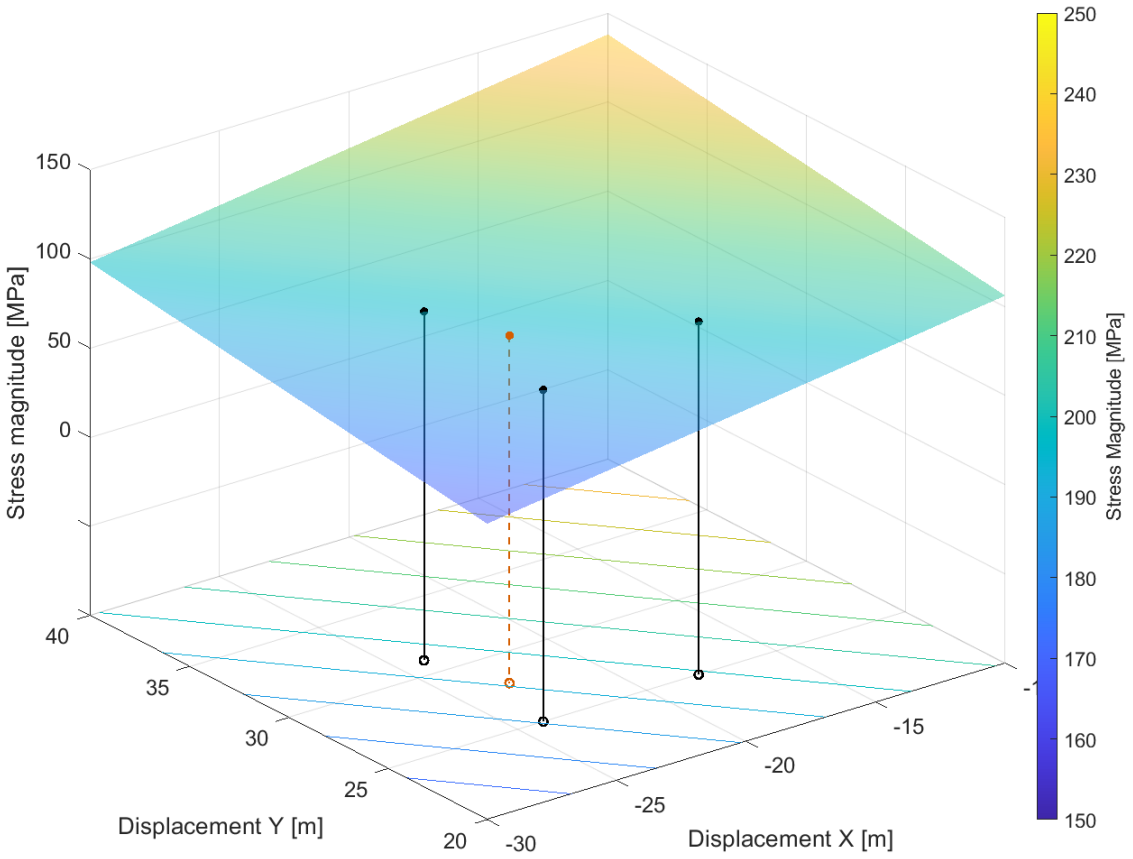


Figure 4-1 Concept of FAST Estimation.

Results of the three model scenarios provide three stress magnitudes that result from three different sets of displacement boundary conditions for each model. Black circles indicate for an arbitrarily chosen point in the model volume the values of the model results for the boundary conditions specified on the x- and y-axes. In a linear system these three different model results span a stress plane in $\mathbb{R}^3(\mathrm{x}', \mathrm{y}', \mathrm{S_{Hmax}})$ for any pair of displacement boundary conditions.. Solving analytically the planes equation for the desired boundary conditions results in a new stress magnitude value (red circle) without solving the model again.

5 Individual functions

The technical implementation of FAST Estimation is described in this chapter, mainly designated for advanced users that may want to adapt the code for their own demands. Therefore, the Python functions and their intentions are described. While the more important functions are described in detail, a short summary is provided for the less important functions.

5.1 Writing Tecplot 360 EX macro function (write_macro)

Only required if PyTecplot is NOT used. The function write_macro generates a Tecplot 360 EX macro that exports the modelled stress state from given locations in the model, usually at locations where the stress state is estimated. Since these locations are not necessarily at nodes the variables are interpolated from the nodes to the exact coordinates in the volume. The function requires four input variables (Table 5-1) which are automatically provided and transmitted by the script.

Table 5-1 Input variables required by the function write_macro.

Variable	Description
coords	Locations in model coordinate system where the stress state is estimated. Example: coords = [[1407,2016,-600],[511,2018,-400],[506,2021,-100]
stress_vars	Provide the names of the stress variables in the model that should be estimated. Example: stress_vars = ['SHmax','Shmin']
name	The desired name of the macro without the file type extension ".mcr".
folder	Provide the directions to the folder in your system which contains the script data. It is important to include the full path since this information is not used in Python but for the Tecplot 360 EX macro which does not support relative paths. Make sure of the correct usage of slash/backslash depending on the operating system and that the correct escape characters are used on Windows. Example (Windows): folder = 'd:\\Data\\Project\\FAST\\Test' Example (Linux): folder = '/home/user/Project/FAST/Test'

With these variables the function creates a Tecplot 360 EX macro with the following structure.

1. The Tecplot macro header is written which may be sensitive with regards to the Version of Tecplot. The according line is clearly marked in the code and may need an adaptation. Per default a Macro header suitable for Tecplot 360 EX 2019 R1 is used (#!MC 1410). Other headers are e.g. #!MC 1120 for Tecplot 360 EX 2013 R1

2. The internal number of the variables that are used for stress estimation in Tecplot 360 EX are sought and stored in according macro variables.

```
$!GETVARNUMBYNAME |VAR0|
NAME = "SHmax"
```

3. For each data record location specified in the variable stress an individual 1D zone (point) is created. The number of zones created per data record location depends on the number of model scenarios, usually three. The following syntax (with an exemplified location) is repeated accordingly often.

```
$!CREATERECTANGULARZONE
IMAX = 1
```

```
JMAX = 1
KMAX = 1
X1 = 6.621267e+05
Y1 = 5.300777e+06
Z1 = -1.259692e+02
X2 = 6.621267e+05
Y2 = 5.300777e+06
Z2 = -1.259692e+02
```

4. The variables are linearly interpolated from the source zones (i.e. one of the model steps) to the zones defined in step 1. The following code is repeated accordingly often.

```
$!LINEARINTERPOLATE
SOURCEZONES =  [1]
DESTINATIONZONE = 4
VARLIST =  [|VAR0|,|VAR1|]
LINEARINTERPCONST = 0
LINEARINTERPMODE = DONTCHANGE
```

5. The variable values in the 1D zones are exported to comma-separated data files, for each variables an individual file.

```
$!EXTENDEDCOMMAND
COMMANDPROCESSORID = 'excsv'
COMMAND = 'FrOp=1:ZnCount=6:ZnList=[4-
9]:VarCount=1:VarList=[VAR0]:ValSep=",":FNAME="D:\Data\Project\FA
ST\data\shmax.csv"'
```

6. The 1D zones created in step 3 are deleted from the Tecplot 360 EX file.

```
$!DELETEZONES [4-34]
```

5.2 Load output database (load_abq, load_mse)

Only required is PyTecplot IS used. Depending on the solver (Abaqus or Moose) the according version of the function is used. As only argument the name of the output database without file extension is provided. PyTecplot currently does not support loading of Abaqus or Moose output databases via a specific command. Thus, it is realized using a macro command. Then the file is saved as a *.plt file native to Tecplot.

The loading of Abaqus output databases is dependent on the operating system since Tecplot on Windows supports only reading Abaqus *.odb files while Tecplot on Linux supports only Abaqus *.fil files. The check for the operating system works automatically with the Python platform function.

The Moose Framework provides the three different sets of boundary conditions in three separate files. They are consecutively loaded and appended to the Tecplot *.plt file using macro commands.

5.3 Extract variables from *.plt file (extract_tp)

Only required is PyTecplot IS used. The *.plt files created by the functions load_abq or load_mse are loaded in Tecplot. The arguments of the function are listed and described in Table 5-2. Once

the *.plt file is loaded, specific actions need to be taken dependent on the solver type and operating system.

If a Linux operating system is used the cell-centred stress tensor variables from the *.fil file are converted to nodal variables using the function cell2nodal. In case of using Moose as solver, the solution time is assigned and the variables of the stress tensor are renamed to comply with the variable names expected by the GeoStressCmd Add-on using the function rnm_vrbls. If desired, the reduced stress tensor (i.e. S_{Hmax} and S_{hmin}) is derived using the GeoStressCmd Add-on.

Eventually, the stress state is extracted at the specified locations. Therefore, the additional function strextract is used. Its arguments are the location and zone of the stress component, the PyTecplot model handle, and the name of the stress component. The modelled stress states are returned to the caller function.

Table 5-2 Input variables required by the function extract_tp.

Variable	Description
name	The name of the *.plt file that is to be loaded without the file extension.
solver	The name of the used solver, i.e. either Abaqus or Moose Example: solver = 'abaqus'
loc	Locations where the stress is estimated in model coordinates. Example: loc = [[1407,2016,-600],[511,2018,-400],[506,2021,-100]
stress_vars	Names of the stress variables in the model that should be estimated. Example: stress_vars = ['SHmax','Shmin']

5.4 Estimation of the stress state (solve)

The function solve is the core of FAST Estimation that sets up the plane's equation defined by three sets of boundary conditions and the resulting modelled stress states (see Section 4). Then, the equation is solved for the previously specified boundary conditions in order to obtain the newly estimated stress state. The arguments required for the function solve and examples are described in Table 5-3Table .

Table 5-3 Input variables required by the function solve.

Variable	Description
moss	Stress state of the three modelled test stress scenarios Example: moss = [22.5, 15.4, 23.2]
bcs	The boundary conditions of the three test scenarios. Example: bcs = [[4, -4],[4,-5],[5, -5]]
bcnx	Boundary conditions for estimated stress state in x' direction. Example: bcnx = -4.4
bcny	Boundary conditions for estimated stress state in y' direction. Example: bcny = 6.3

5.5 Additional functions

In the following, functions of less importance and simpler structure are summarized.

cell2nodal When operating on a Linux system with Abaqus, Tecplot is only able to load *.fil files which has cell-centred variables. This function works with PyTecplot in order to change the variables from cell-centred to nodal.

independence_check Checks whether the test boundary conditions and the resulting stress states are chosen in a way that they are linearly independent, i.e. that three independent supporting points are available to set up a plane. This is required in order to setup a meaningful plane equation.

load_csv Reads the variable files that are exported by the Tecplot macro into Python. The variables are then sorted in the correct order.

load_bc Reads a file that contains the boundary conditions (displacement in x' direction, displacement in y' direction) for which the stress state is estimated.

load_loc Reads a file that contains the location (x, y, z, in model coordinates) where a stress state is estimated.

rnm_vrbls Output databases from Moose need to be adapted in order to be compatible with the GeoStressCmd Addon. A solution time needs to be assigned. Furthermore, the stress tensors components and displacement variables need to be renamed. This is achieved in this function if PyTecplot is used.

strextract Function that extracts specified variables using PyTecplot. The *.plt file already needs to be loaded using PyTecplot and the model handle is transmitted to the function.

stress_rotation If the variables S_{Hmax} and S_{hmin} are estimated a possible switching between them is investigated. If S_{hmin} becomes larger than S_{Hmax} this may be an indicator for an incorrect stress state (in particular if the orientation of S_{Hmax} is well constrained). This function is used to notify the user of such an occurrence.

write_output The variable that contains the results is written to an output file. Each output files name contains a timestamp in order to prevent the loss of data that occurs due to overwriting a previous results file.

6 Troubleshooting

Several cases of possible errors are covered by error messages and instructions to resolve the problem that are printed to the screen. In the following, several additional possible errors are listed and possible solutions are explained.

Without PyTecplot an error occurs in the beginning of the second step	• Is the file path set to the correct location? Also check in the macro file itself. • Are the escape characters in the file path set correctly? • Does the folder “data” exist? • Do the requested variables exist with the correct names?
When opening the GeoStress GUI stress tensor variables are missing	• Rename the stress tensor and displacement variables to the names expected by GeoStress, e.g. using the auxiliary Tecplot macro rename_stress.mcr
An error occurs during execution of GeoStress GUI	• Are the stress tensor variables nodal variables? Convert cell-centered variables to nodal ones using the auxiliary Tecplot macro cellcent2nodal.mcr
Using PyTecplot an error occurs during execution of GeoStressCmd	• Are you using a *.plt file that was read from a *.fil file on a Windows computer? Convert cell-centered variables to nodal ones using the auxiliary Tecplot macro cellcent2nodal.mcr
The Tecplot macro does not run/returns an error	• Check whether the macro header is correct. (You can find the appropriate header for your Tecplot version by recording a simple macro and opening it in a text editor.) • Are any locations outside the model volume? • Does the folder /data exist? Is the path specified in the macro correct? • Are there sufficient steps/zones in Tecplot?
An error shows after running the macro in Tecplot and pressing enter in the Python command prompt	• Re-run the Python script and press enter without leaving the commando prompt. • Make sure the macro exported the stress states to the correct folder.

7 Examples

In the supplemented examples a basic estimation procedure is presented. All files are available in the examples folder as Abaqus® solver input file (*.inp) and output file (*.odb and *.fil, Table 7-1) and as Moose input file (*.i) and output database (*.dat, Table 7-2). Three model scenarios with arbitrary, but different displacement boundary conditions (4, -4; 2, -5; 4, -3) are included. The geometry is in a separate file (*.geom and *.inp respectively). The exemplary parameters that are provided in the Python script matches these examples.

Table 7-1 Abaqus example files

Short explanation of the files provided for an exemplified estimation.

File Name	Explanation
test_scenarios.fil	Three model scenarios with different displacement boundary conditions. Abaqus® output database for loading in Tecplot on Linux.
test_scenarios.inp	Three model scenarios with different displacement boundary conditions. Abaqus® input file.
test_scenarios.odb	Three model scenarios with different displacement boundary conditions. Abaqus® output database for loading in Tecplot on Windows.

Table 7-2 Moose example files

Short explanation of the files provided for an exemplified estimation.

File Name	Explanation
test_scenarios_1.i	First (out of three) model scenario Moose input file.
test_scenarios_1_out_0001.dat	First (out of three) solved model scenario.
test_scenarios_2.i	Second (out of three) model scenario Moose input file.
test_scenarios_2_out_0001.dat	Second (out of three) solved model scenario.
test_scenarios_3.i	Third (out of three) model scenario Moose input file.
test_scenarios_3_out_0001.dat	Third (out of three) solved model scenario.

The output databases can be directly loaded in Tecplot 360 EX and FAST Estimation can hence be tested without using Abaqus® or the Moose Framework to solve the model. Please note that the models were solved using Abaqus® 2019. Output files from this version of Abaqus® can only be read from Tecplot 360 EX 2019 R1 onwards. For compatibility with older Tecplot 360 EX versions the input files can be rerun in an older Abaqus® version (older Tecplot 360 EX versions up to 2017 require Abaqus® 6.11 output files, later on Abaqus ® 6.14).

Acknowledgements

The authors would like to thank Kirsten Elger and Dorothea Hansche for supporting the publication.

The work leading to these results has been supported in the framework of the project Spannungsmodell Endlagerung Deutschland SpannEnD 2.0 by the federal company for radioactive waste disposal BGE.

References

Heidbach, O., Ziegler, M. O., & Stromeyer, D. (2020). *Manual of the Tecplot 360 Add-on GeoStress v2.0*. https://doi.org/10.2312/WSM.2020.001

Hergert, T., Heidbach, O., Reiter, K., Giger, S. B., & Marschall, P. (2015). Stress field sensitivity analysis in a sedimentary sequence of the Alpine foreland, northern Switzerland. *Solid Earth*, *6*(2), 533–552. https://doi.org/10.5194/se-6-533-2015

Morawietz, S., Heidbach, O., Reiter, K., Ziegler, M. O., Rajabi, M., Zimmermann, G., Müller, B., & Tingay, M. (2020). An open-access stress magnitude database for Germany and adjacent regions. *Geothermal Energy*, *8*(1). https://doi.org/10.1186/s40517-020-00178-5

Reiter, K., & Heidbach, O. (2014). 3-D geomechanical-numerical model of the contemporary crustal stress state in the Alberta Basin (Canada). *Solid Earth*, *5*(2), 1123–1149. https://doi.org/10.5194/se-5-1123-2014

Stromeyer, D., Heidbach, O., & Ziegler, M. O. (2020). *Tecplot 360 Add-on GeoStress v2.0* (V2.0). GFZ Data Services. https://doi.org/10.5880/wsm.2020.001

Ziegler, M. O. (2019). *Manual of the Python Script HIPSTER v1.3*. https://doi.org/10.48440/wsm.2021.001

Ziegler, M. O. (2022). Rock Properties and Modelled Stress State Uncertainties: A Study of Variability and Dependence. *Rock Mechanics and Rock Engineering*. https://doi.org/10.1007/s00603-022-02879-8

Ziegler, M. O., & Heidbach, O. (2020). The 3D stress state from geomechanical–numerical modelling and its uncertainties: a case study in the Bavarian Molasse Basin. *Geothermal Energy*, *8*(1). https://doi.org/10.1186/s40517-020-00162-z

Ziegler, M. O., & Heidbach, O. (2021a). *Manual of the Matlab Script FAST Calibration v2.0*. https://doi.org/10.48440/wsm.2021.002

Ziegler, M. O., & Heidbach, O. (2021b). *Manual of the Python Script PyFAST Calibration v1.0*. https://doi.org/10.48440/wsm.2021.003

Ziegler, M. O., & Heidbach, O. (2023). Bayesian Quantification and Reduction of Uncertainties in 3D Geomechanical-Numerical Models. *Journal of Geophysical Research: Solid Earth*, *128*(1). https://doi.org/10.1029/2022JB024855

Ziegler, M. O., Heidbach, O., Reinecker, J., Przybycin, A. M., & Scheck-Wenderoth, M. (2016). A multi-stage 3-D stress field modelling approach exemplified in the Bavarian Molasse Basin. *Solid Earth*, *7*(5), 1365–1382. https://doi.org/10.5194/se-7-1365-2016