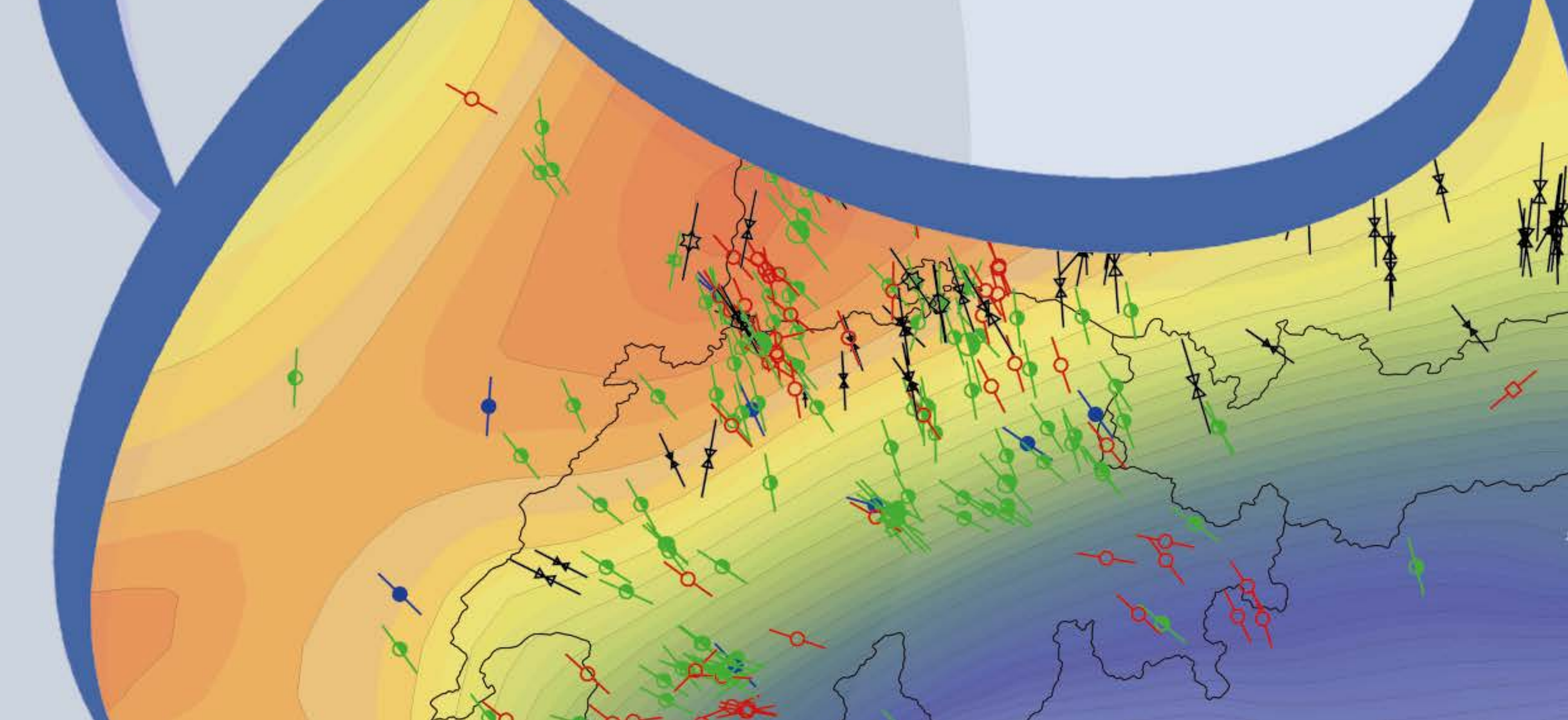
WSM WORLD STRESS MAP

WSM Technical Report 21-03

Manual of the Python Script PyFAST Calibration *v1.0*

Moritz O. Ziegler and Oliver Heidbach



Recommended citation of the report:

Ziegler, M. O. & Heidbach, O. (2021). Manual of the Python Script PyFAST Calibration v1.0. World Stress Map Technical Report 21-03, GFZ German Research Centre for Geosciences.
DOI: https://doi.org/10.48440/wsm.2021.003

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

Ziegler, M. O. & Heidbach O.(2021). Python Script PyFAST Calibration v1.0. GFZ Data Services. DOI: https://doi.org/10.5880/wsm.2021.003

The software is available for download on GitHub:

https://github.com/MorZieg/PyFAST_Calibration

Imprint

World Stress Map Project
GFZ German Research Centre for Geosciences

Telegrafenberg
D-14473 Potsdam
Published in Potsdam, Germany
April 2021
https://doi.org/10.48440/wsm.2021.003

WSM Technical Report 21-03

Manual of the Python Script PyFAST Calibration v1.0

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

Table of Contents

List of Tables

List of Figures

Abstract

The 3D geomechanical-numerical modelling of the in-situ stress state aims at a continuous description of the stress state in a subsurface volume. It requires observed stress information within the model volume that are used as a reference. Once the modelled stress state is in agreement with the observed reference stress data the model is assumed to provide the continuous stress state in its entire volume.

The modelled stress state is fitted to the reference stress data records by adaptation of the displacement boundary conditions. This process is herein referred to as calibration. Depending on the amount of available stress data records and the complexity of the model the manual calibration is a lengthy process of trial-and-error modelling and analysis until best-fit boundary conditions are found.

The Fast Automatic Stress Tensor Calibration (FAST Calibration) is a Python function that facilitates and speeds up this calibration process. By using a linear regression it requires only three model scenarios with different boundary conditions. The stress states from the three model scenarios at the locations of the reference stress data records are extracted. The differences between the modelled and observed stress states are used for a linear regression that allows to compute the displacement boundary conditions required for the best-fit modelled stress state. If more than one reference stress state is provided, the influence of the individual observed stress data records on the best-fit boundary conditions can be weighted.

The script files are provided for download at:
http://github.com/MorZieg/PyFAST_Calibration

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/PyFAST_Calibration. The page numbers (if available) direct to the documentation in this manual.

File Name	Explanation	Page
fast_calibration.py	Python script for the calibration of a geomechanical-numerical model on stress data records.	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 a calibration using either Moose or Abaqus.	16
cellcent2nodal.mcr	Tecplot macro file that converts cell-centred to nodal variables.	6
rename_stress.mcr	Tecplot macro that renames the stress state variables from a Moose solver to be compatible with the Tecplot Add-on GeoStress.	6

1 Introduction

3D geomechanical-numerical modelling of the stress state depends on stress data records that are available within the model volume. Dirichlet displacement boundary conditions are altered until a fit of the modelled to these observed data is achieved. Herein, this process of fitting the model on observables is referred to as model calibration. Once the boundary conditions are found with which the model is best fitted to the stress data records the model is calibrated. The assumption is made that the stress state in the rest of the model is then also a legitimate representation of the in-situ stress state. A manual adaptation of the boundary conditions towards a best fit is a lengthy procedure that involves a lot of trial-and-error.

PyFAST Calibration (Python Fast Automatic Stress Tensor Calibration) uses a linear regression to speed up and automatize the process of calibration. It is a transfer to Python of the Matlab tool with the same name (Ziegler & Heidbach, 2021) and follows the calibration approach described by Reiter and Heidbach (2014), Hergert et al. (2015), and Ziegler et al. (2016). 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 PyFAST Calibration 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 PyFAST Calibration manual provides an overview of the scripts and is designed to help the user to adapt the scripts for their own needs. It contains basic technical information on 3D geomechanical-numerical modelling (Section 2). The input data and the syntax as well as the execution of the script are presented in Section 3. Section 4 provides information on the individual Python functions that come with PyFAST Calibration and is mainly dedicated to advanced users. Common error messages and according help is provided in Section 5. Examples are provided in Section 6.

Note to users of FAST Calibration v2.0 on Matlab:

PyFAST Calibration is a transfer of only the main functions of FAST Calibration v2.0 to Python. For sanity checks of the stress state, additional calibration data types, or a multistage approach FAST Calibration v2.0 should be used (Ziegler & Heidbach, 2021; 2021a).

2 Background of geomechanical modelling

3D geomechanical-numerical modelling requires a static geological model that describes the geometry of the geologic units and faults, the density distribution and the elastic rock properties. For the calibration process stress data records are needed at individual points within the model volume. To solve the partial differential equation of the equilibrium of forces the model volume is discretized into finite elements. The discretized geometry is populated with rock properties, i.e. the density, Young’s module, and Poisson ratio. Fitting the magnitude of the vertical stress S_V is easy as the density distribution is relatively well known. In contrast fitting the magnitudes of maximum and minimum horizontal stress, S_{Hmax} and S_{hmin} respectively, is challenging as stress data of these two components is sparse (we assume here for simplicity that S_V and thus S_{Hmax} and S_{hmin} are principal stresses). Information on the orientation of the stress tensor by means of the orientation of S_{Hmax} is provided by the World Stress Map (WSM) database (Heidbach et al., 2016). However, stress orientations are helpful for the model calibration any differential displacement boundary condition will lead to an instantaneous adjustment of the stress tensor orientation parallel and perpendicular to the model boundary regardless the amount of displacement. In contrast, information on the magnitudes of principal horizontal stresses is generally sparse and incomplete (Morawietz et al., 2020).

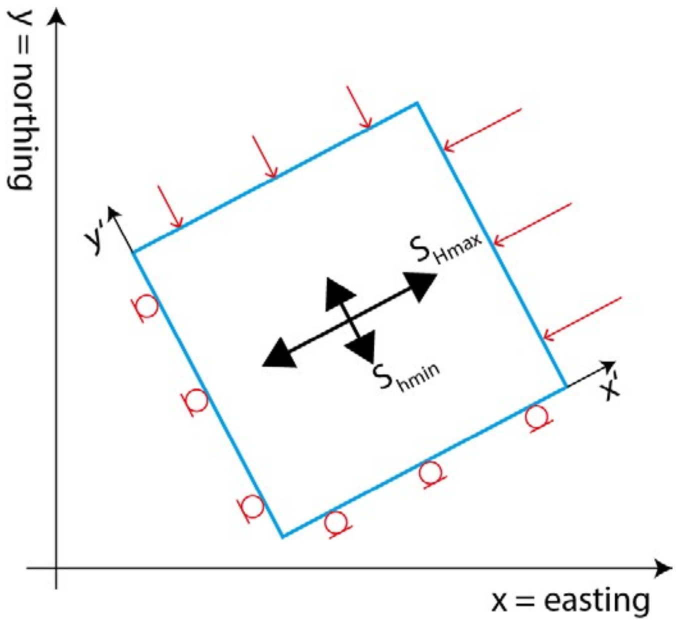


Fig. 2-1 Map view of a model area (blue box) in a geographical coordinate system.

The boundary conditions (red) are applied in a rotated coordinate system (x’ and y’) parallel to the model boundaries which are oriented parallel to the prevailing orientations of the principal horizontal stress axes. Information on the orientation is available from data of the World Stress Map database (Heidbach et al., 2016). The application of even small displacement boundary conditions (Dirichlet type) perpendicular to the model boundaries are usually sufficient to achieve a good fit of the overall stress orientation.

To calibrate the geomechanical-numerical model information on the S_{Hmax} and S_{hmin} magnitudes is essential. The model calibration is achieved by a comparison of the modelled stress to the observed S_{Hmax} and S_{hmin} magnitudes in dependence of the Dirichlet displacement boundary condition (Fig. 2-1). Their values are altered until a good fit of the modelled stress state to the observed stress information at the calibration points is achieved. Thus, the models calibration depends on only two displacements which facilitates the calibration process as the best-fit boundary conditions can be found by a system of linear equations. The automated setup and solving of this linear equation system is the core of the FAST-Calibration tool. Note that two implicit assumption are that 1) the rheology is linear elastic and 2) that the stress state is controlled by a lateral displacement that is not changing with depth

The calibration procedure is illustrated in Fig. 2-2. Three different test model scenarios with different arbitrary, but reasonable displacement boundary conditions are solved (Fig. 2-2a). At several locations within the model volume stress data records are available (Fig. 2-2b). For each test model scenarios the modelled stress state is compared with these stress data records. Each comparison provides a mean deviation for S_{Hmax} and S_{hmin}, respectively (Fig. 2-2c). A system of linear equations is set up in the domain of the displacement boundary conditions (x and y) and the deviations of observed and modelled data (z) (Fig. 2-2d). For the smallest deviation of each, S_{Hmax} *or* S_{hmin}, an infinite number of corresponding displacement boundary conditions exist. However, only one set satisfies both the requirements for the smallest deviation in observed and modelled S_{Hmax} *and* S_{hmin}. This set of displacement boundary conditions is applied to compute the best-fit model (see Ziegler et al., 2016).

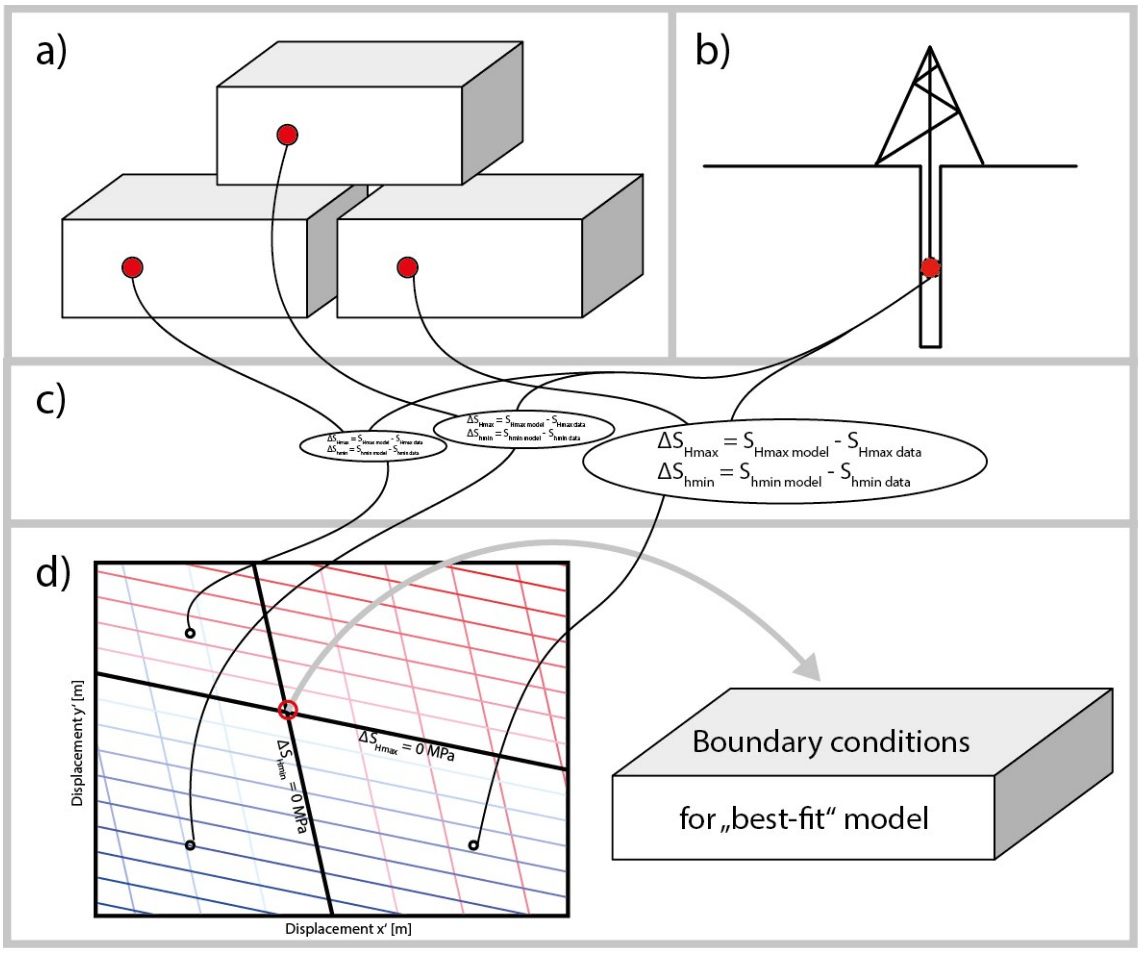

a)
b)
c)
$\Delta S_{Hmax} = S_{Hmax\ model} - S_{Hmax\ data}$
$\Delta S_{hmin} = S_{hmin\ model} - S_{hmin\ data}$
d)
$\Delta S_{Hmax} = 0$ MPa
$\Delta S_{hmin} = 0$ MPa
Displacement y' [m]
Displacement x' [m]
Boundary conditions
for „best-fit“ model

Fig. 2-2 Basic workflow of the tool FAST Calibration.

The stress state from three test model scenarios with arbitrary displacement boundary conditions (a) and observed stress data records (b) are compared (c). A set of linear equations is set up that provide the best-fit boundary conditions (d).

3 Calibration of a model

PyFAST Calibration 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 PyTecplot extension of Tecplot. Alternatively, as an intermediate manual step the user needs to run a Tecplot macro.

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, PyFAST Calibration is used as a Python package with all required variables transmitted by the caller function.

3.1 General procedure

The FAST Calibration approach requires at least one data record for S_{Hmax} and S_{hmin} magnitude each. Each data record consists of a location defined in the model coordinate system (x, y, z), a magnitude (in MPa), and is assigned a confidence between 0 (low) and 1 (high). Furthermore, FAST Calibration requires information on the test 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. Eventually, the name of the two stress components from the model results used for calibration need to be specified, e.g. to be *SHmax* and *Shmin* or *XX Stress* and *YY Stress*. A compilation of all required variables and examples is presented in Table 3-1.

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

Variable	Description
folder	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) if PyTecplot is *not* used. Example (Windows): folder = 'd:\\Data\\Project\\FAST\\Test' Example (Linux): folder = '/home/user/Project/FAST/Test'
name	Provide the file name with three independent stress states derived from three different boundary conditions. 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. Example: bcs = [[4, 2, 4],[-4, -5, -3]]
stress_vars	Provide the names in Tecplot of the two stress variables in the model that should be calibrated. Example: stress_vars = ['SHmax','Shmin']
shmax	Location, magnitude, and weight of the S_{Hmax} magnitude data records that should be used for calibration. Example: shmax = [[5050,-3500,-800,22.5,1.0]]
shmin	Location, magnitude, and weight of the S_{hmin} magnitude data records that should be used for calibration. Example: shmin = [[5050,-3500,-800,12.7,0.5],[5100,-3450,-2745,41.9,1.0]]
solver	The name of the used solver, i.e. either Abaqus or Moose Example: solver = 'abaqus'
pytecplot	Whether PyTecplot should be used or not. Example: pytecplot = 'off'

3.2 Specific procedure

PyFAST Calibration in connection with Tecplot and PyTecplot supports output files from both Abaqus and the Moose Framework. In case of Abaqus even two different output files are 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. The detailed procedure is presented in the following dependent whether PyTecplot is used or not.

3.2.1. Without PyTecplot

The application of FAST Calibration without using PyTecplot requires several steps (including two steps performed within the Python script) which are illustrated in Figure 3-1.

1. Prepare your working directory with a subfolder called “data” and set the variables in the PyFAST Calibration script.
2. Load the model output database in Tecplot. Optionally run GeoStress (Heidbach et al. 2020) or a similar tool.
3. Run PyFAST Calibration in the working directory. A Tecplot macro is written to your working directory. You will be prompted to execute the macro in Tecplot.
4. 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 older.
5. Press enter in the Python command window. The *.csv files will be loaded to Python variables and the best-fit boundary conditions will then be computed and displayed.

Deviations and additions to these steps are listed in the following.

Abaqus/Windows

No additional steps required

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.

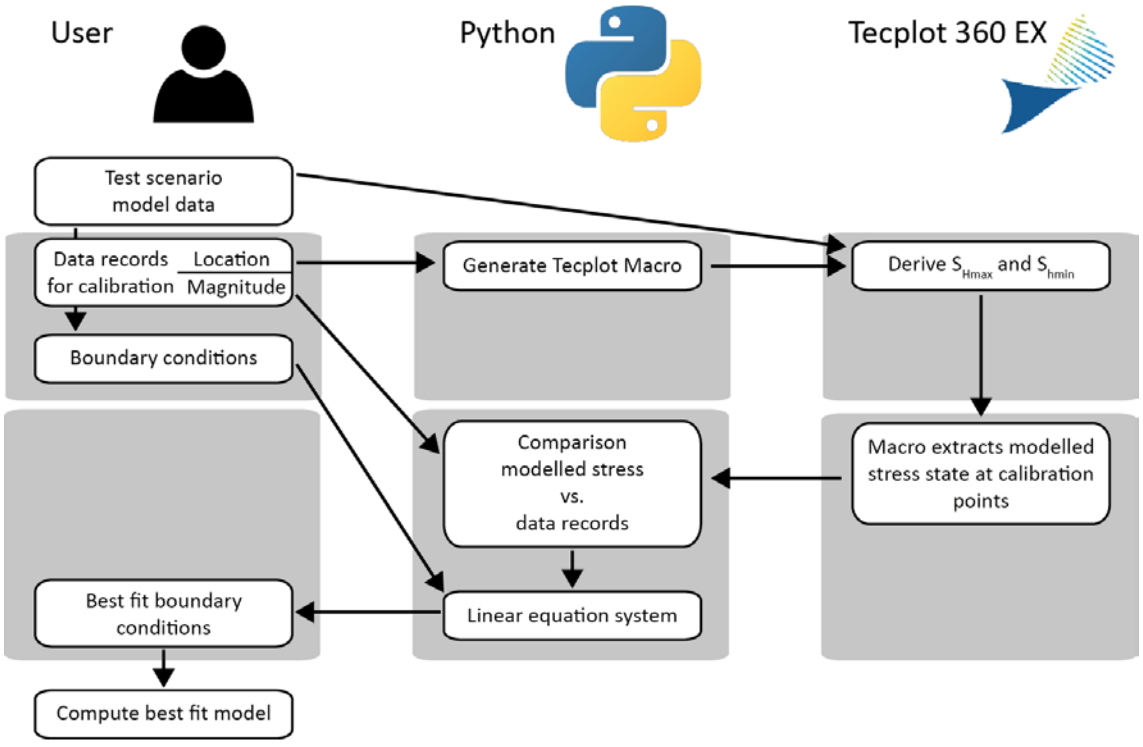
```
*EL FILE
S
*NODE FILE
COORD, U
```

- The stress tensors variables in the *.fil file are cell-centred 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 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.

- 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.



User
Python
Tecplot 360 EX
Test scenario model data
Data records for calibration
Location
Magnitude
Boundary conditions
Generate Tecplot Macro
Derive S_{Hmax} and S_{hmin}
Comparison modelled stress vs. data records
Macro extracts modelled stress state at calibration points
Best fit boundary conditions
Linear equation system
Compute best fit model

Figure 3-1 FAST Calibration 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.

3.2.2. Using PyTecplot

The application of FAST Calibration 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 3-2. 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.

Abaqus/Windows

No additional steps required

Abaqus/Linux

- Instead of loading the standard *.odb file only the *.fil file is supported by the Linux Tecplot loader. To generate a *.fil file include the following lines in the Abaqus input files first *STEP section.

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

Moose

No additional steps required

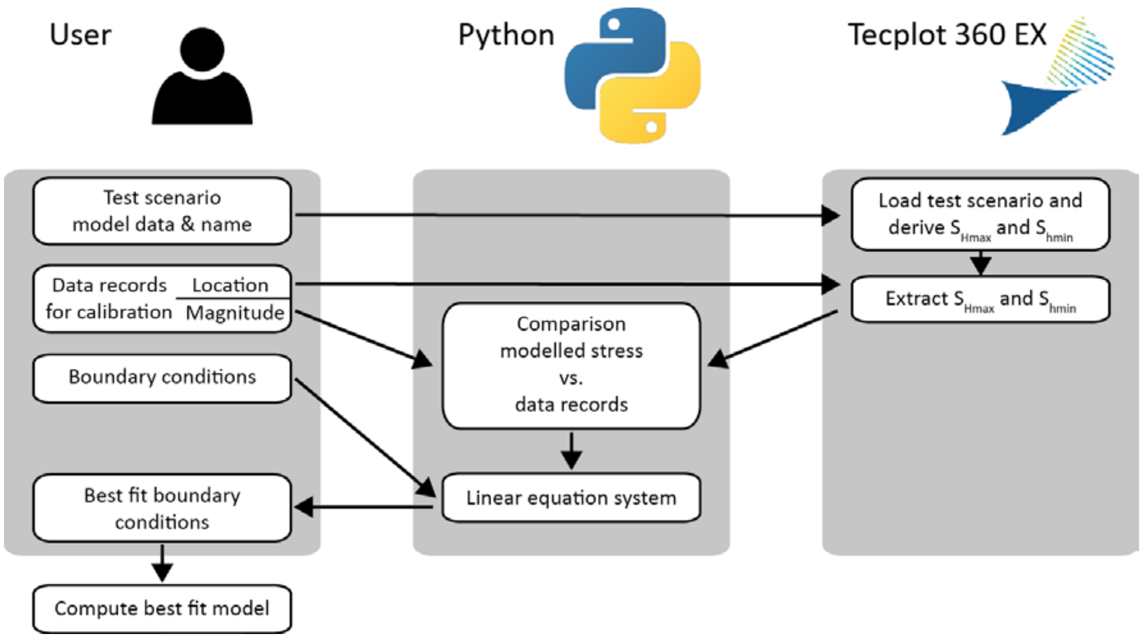

User
Python
Tecplot 360 EX
Test scenario model data & name
Data records for calibration
Location
Magnitude
Boundary conditions
Best fit boundary conditions
Compute best fit model
Comparison modelled stress vs. data records
Linear equation system
Load test scenario and derive S_{Hmax} and S_{hmin}
Extract S_{Hmax} and S_{hmin}

Figure 3-2 FAST Calibration procedure using PyTecplot.
Three columns indicate the three involved parties: The user, Python and Tecplot.

4 Basic principles

The technical implementation of FAST Calibration 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.

4.1 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.

4.2 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 are listed and described in Table 4-1.

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 GeoStress Add-on using the function rnm_vrbls. If desired, the reduced stress tensor (i.e. S_{Hmax} and S_{hmin}) is derived using the GeoStress Add-on.

Eventually, the stress state are extracted at the locations where stress data is available for comparison. 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. Eventually, the modelled stress states are returned to the caller function.

Table 4-1 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’
shmax	Location, magnitude, and weight of the S_{Hmax} magnitude data records that should be used for calibration. Example: shmax = [[5050,-3500,-800,22.5,1.0]]
shmin	Location, magnitude, and weight of the S_{hmin} magnitude data records that should be used for calibration. Example: shmin = [[5050,-3500,-800,12.7,0.5],[5100,-3450,-2745,41.9,1.0]]
stress_vars	Provide the names of the two stress variables in the model that should be calibrated. Example: stress_vars = ['SHmax','Shmin']

4.3 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 calibration points. Since the calibration points are not necessarily at nodes the variables are interpolated from the nodes to the exact coordinates in the volume. The function requires five input variables (Table 4-2) which are automatically provided and transmitted by the script.

Table 4-2 Input variables required by the function write_macro.

Variable	Description
shmax	Location, magnitude, and weight of the S_{Hmax} magnitude data records that should be used for calibration. Example: shmax = [[5050,-3500,-800,22.5,1.0]]
shmin	Location, magnitude, and weight of the S_{hmin} magnitude data records that should be used for calibration. Example: shmin = [[5050,-3500,-800,12.7,0.5],[5100,-3450,-2745,41.9,1.0]]
stress_vars	Provide the names of the two stress variables in the model that should be calibrated. 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.
2. The internal number of the variables S_{Hmax} and S_{hmin} in Tecplot 360 EX are sought and stored in the macro variable SHMAX and SHMIN.

```
$!GETVARNUMBYNAME |SHMAX|
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 two 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 =  [|SHMAX|,|SHMIN|]
LINEARINTERPCONST = 0
LINEARINTERPMODE = DONTCHANGE
```

5. The variable values in the 1D zones are exported to two comma-separated data files. Each file contains all instances of one of the two variables.

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

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

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

4.4 Load data exported by macro (load_csv)

Only required if PyTecplot is NOT used. The content of the *.csv files written by the Tecplot macro are loaded as Python variables. The variables from all three steps are sorted accordingly in order to be comparable. Please note that both variables are exported at all calibration points. Thus, calibration points with only SHmax or only Shmin data are still exported for both stress components. Thus, the first (SHmax) values are discarded in the following. Eventually, the modelled stress data is converted into numpy arrays and returned to the caller function. The required arguments are described in Table 4-3.

Table 4-3 Input variables required by the function load_csv.

Variable	Description
name	The file name with three independent stress states derived from three different boundary conditions. Example: name = 'test_scenarios'
leshmax	Number of S_{Hmax} calibration points. (Length of the variable shmax)
leshmin	Number of S_{hmin} calibration points. (Length of the variable shmin)
stress_vars	Provide the names in Tecplot of the two stress variables in the model that should be calibrated. Example: stress_vars = ['SHmax','Shmin']

4.5 Estimation of best-fit boundary conditions (calibrate)

The core of FAST Calibration is the function *calibrate* that sets up a system of linear equations which are used to estimate the best-fit boundary conditions. Therefore, the boundary displacements of at least three different combinations of boundary conditions are required together with the solved model scenarios. The functions arguments are listed and described in Table 4-4.

Table 4-4 Input variables required by the function calibrate.

Variable	Description
shmax	Location, magnitude, and weight of the S_{Hmax} magnitude data records that should be used for calibration. Example: shmax = [[5050,-3500,-800,22.5,1.0]]
shmin	Location, magnitude, and weight of the S_{hmin} magnitude data records that should be used for calibration. Example: shmin = [[5050,-3500,-800,12.7,0.5],[5100,-3450,-2745,41.9,1.0]]
shmax_calib	The modelled S_{Hmax} stress state at the calibration points from the three test scenarios provided by load_csv or extract_tp.
shmin_calib	The modelled S_{hmin} stress state at the calibration points from the three test scenarios provided by load_csv or extract_tp.
bcs	The boundary conditions of the three test scenarios.

The deviation of the modelled stress state from the observed data record for each boundary condition scenario (bc) at each location $(\mathrm{x}, \mathrm{y}, \mathrm{z})$ with available observed stress data records is computed for S_{Hmax} and S_{hmin} by

$$\Delta S_{Hmax}(x, y, z, bc) = S_{Hmax,modelled}(x, y, z, bc) - S_{Hmax,observed}(x, y, z)$$

exemplified here for S_{Hmax}. The weighted mean deviation of the modelled and observed S_{Hmax} and S_{hmin} magnitudes are computed for each boundary condition scenario by

$$\widetilde{\Delta S}_{Hmax}(bc) = \frac{\sum_i w_i \, \Delta S_{Hmax}(x_i, y_i, z_i, bc)}{\sum_i w_i}$$

with i the number of data records and $\mathrm{w_i}$ the individual weighting of each data record. Now for each boundary condition scenario a mean deviation of the modelled from the observed S_{Hmax} and S_{hmin} magnitudes exists. An example of the available information is shown in Table 4-5.

Table 4-5 Example for the information required for the generation of the planes.

At least three different boundary condition scenarios are required. For each scenario the displacement in x' and y' and $\widetilde{\Delta \mathrm{S}}_{\mathrm{Hmax}}$ and $\widetilde{\Delta \mathrm{S}}_{\mathrm{hmin}}$ are available.

BC scenario	x' displacement	y' displacement	$\widetilde{\Delta S}_{Hmax}$	$\widetilde{\Delta S}_{hmin}$
1	-10	5	-12.3	8.7
2	-30	25	2.5	-3.2
3	-30	5	-14.4	-4.8

This information is used to derive the boundary conditions that fulfil the requirement that there is no deviation between the observed and modelled $\widetilde{\Delta \mathrm{S}}_{\mathrm{Hmax}}$ and $\widetilde{\Delta \mathrm{S}}_{\mathrm{hmin}}$ magnitudes. Note that no deviation between the average S_{Hmax} and S_{hmin} magnitudes is sought. If more than one data record is available for S_{Hmax} or S_{hmin} it is expected that deviations are observed for the individual data records. The assigned weights control which data records are preferred and should have a smaller deviation. The resulting system of linear equations can be visualised as two intersecting planes (Figure 4-1).

Two vectors are now available for each of the three test model scenarios that take the form

$$\vec{v} = \begin{pmatrix} x \\ y \\ \widetilde{\Delta S}_{Hmax} \end{pmatrix}$$

with the boundary displacement x in x' direction and y in y' direction as well as $\widetilde{\Delta S}_{\text{Hmax}}$ or $\widetilde{\Delta S}_{\text{hmin}}$, the resulting difference of the modelled and expected stress state in the corresponding component.

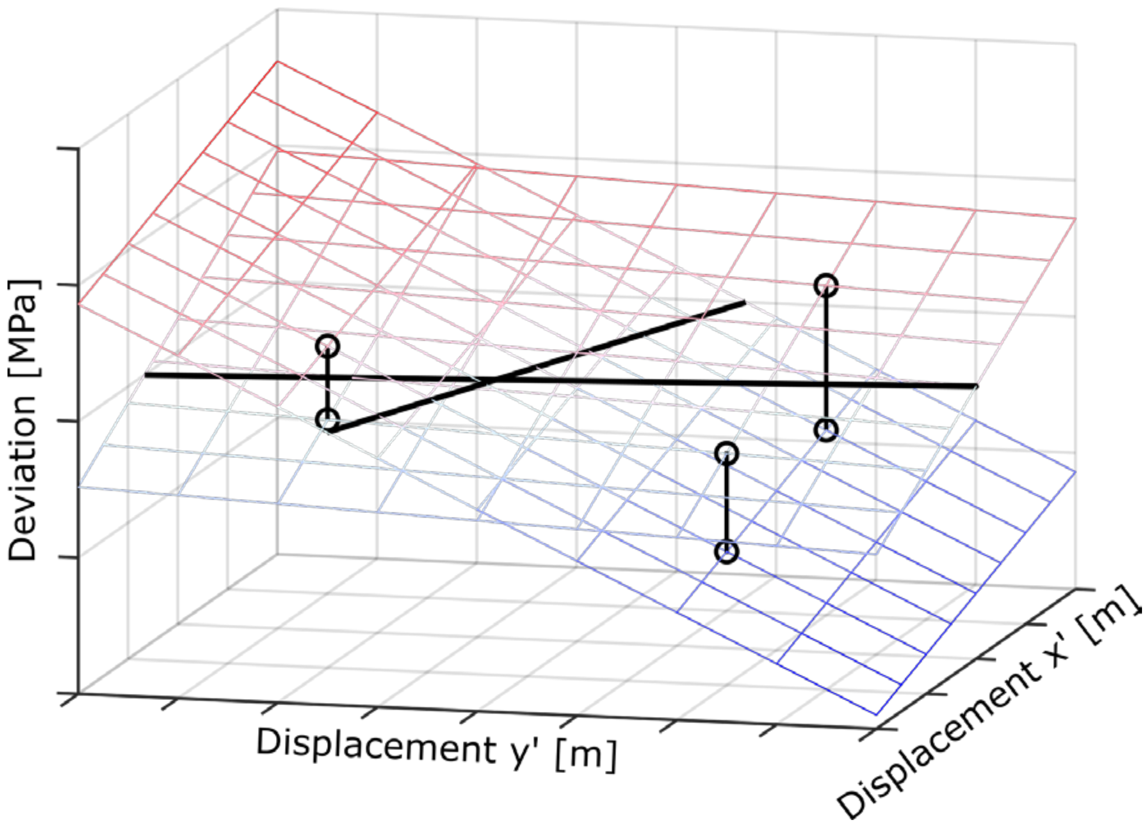


Figure 4-1 Best-fit boundary conditions as planes in 3D.

Three test scenarios (vertically connected black circles) provide mean deviations of modelled and observed S_{Hmax} and S_{hmin} magnitudes for specific boundary conditions. For both, S_{Hmax} and S_{hmin}, the plane spanned by the boundary conditions and the deviations in $\mathbb{R}^3(x', y', \widetilde{\Delta S}_{\text{Hmax}})$ and $\mathbb{R}^3(x', y', \widetilde{\Delta S}_{\text{hmin}})$, respectively, are sought. They are colour coded with blue as a negative deviation and red as a positive deviation. The best-fit boundary conditions are found where the contour lines z=0 (black lines) of the two planes intersect.

It can be pictured that for each, S_{Hmax} and S_{hmin}, 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 mean deviation of S_{Hmax} and S_{hmin}, respectively. Hence, the planes are set up in $\mathbb{R}^3(x', y', \widetilde{\Delta S}_{\text{Hmax}})$ and $\mathbb{R}^3(x', y', \widetilde{\Delta S}_{\text{hmin}})$, respectively (Figure 4-1). The planes equation in parameter form is

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

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

$$n_1\, x' + n_2\, y' + n_3\, \widetilde{\Delta S}_{Hmax} = d$$

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 $\widetilde{\Delta \mathrm{S}}_{\mathrm{Hmax}}$ is set to $\widetilde{\Delta \mathrm{S}}_{\mathrm{Hmax}} = 0$ which represents the line with no deviation between observed and modelled stress state (black solid lines in Figure 4-1). The coordinate form of the equation now reads

$$n_1\, x' + n_2\, y' \qquad\qquad = d$$

Then the equation is transformed to represent this line in $\mathbb{R}^2(\mathrm{x}', \mathrm{y}')$.

$$y' = \frac{d - n_1 x'}{n_2}$$

The same equation is setup for S_{hmin}. Then, $\widetilde{\Delta \mathrm{S}}_{\mathrm{hmin}}$ is set to $\widetilde{\Delta \mathrm{S}}_{\mathrm{hmin}} = 0$ which defines the according line of zero deviation of modelled from observed $\widetilde{\Delta \mathrm{S}}_{\mathrm{hmin}}$ magnitude. Both lines of zero deviation are now defined in $\mathbb{R}^2(\mathrm{x}', \mathrm{y}')$. At their intersection both, $\widetilde{\Delta \mathrm{S}}_{\mathrm{Hmax}}$ and $\widetilde{\Delta \mathrm{S}}_{\mathrm{hmin}}$, are zero and hence x' and y' at the intersection define the best-fit boundary conditions (indicated by the red circle and dashed red lines in Figure 4-2).

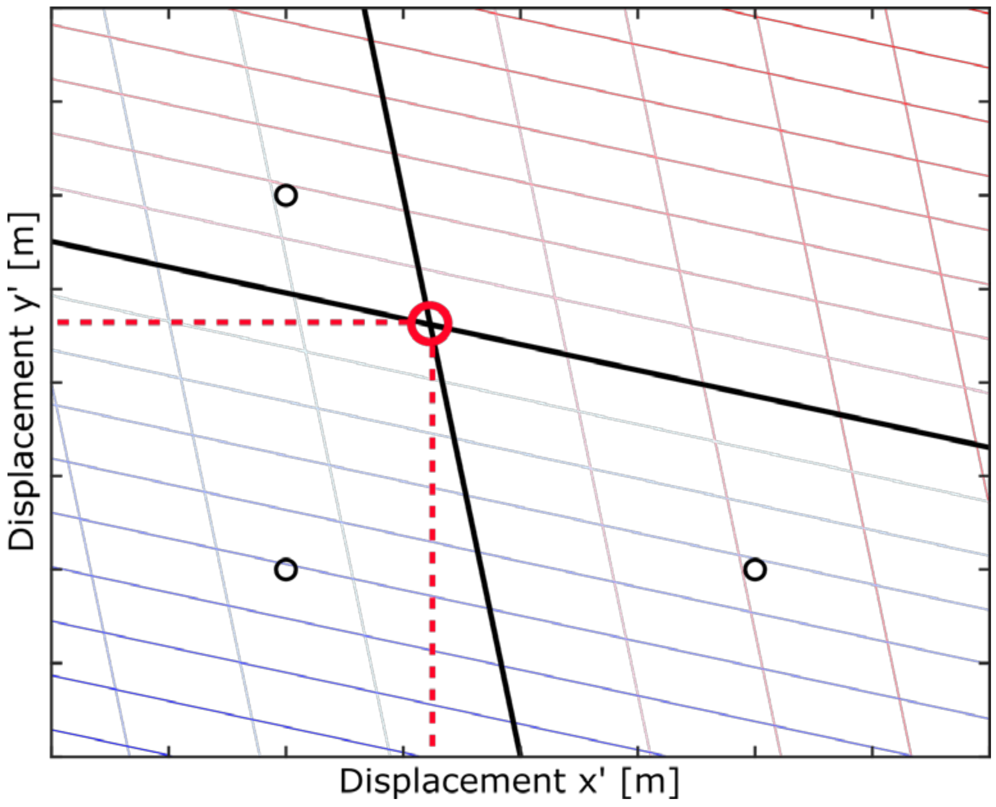


Figure 4-2 Best-fit boundary conditions in 2D.

Red dashed line and red circle are displayed in $\mathbb{R}^2(x', y')$. Three test scenarios (black circles) provide mean deviations of modelled and observed S_{Hmax} and S_{hmin} magnitudes for specific boundary conditions. For both, S_{Hmax} and S_{hmin}, the plane spanned by the boundary conditions and the deviations in $\mathbb{R}^3(x', y', \widetilde{\Delta S}_{Hmax})$ and $\mathbb{R}^3(x', y', \widetilde{\Delta S}_{hmin})$, respectively, are sought. They are represented here by the contour lines with blue as a negative deviation and red as a positive deviation. The best-fit boundary conditions are found where the contour lines z=0 of the two planes intersect.

5 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? If not rename.
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

6 Examples

In the supplemented examples a basic calibration procedure is presented. All files are available in the examples folder as Abaqus® solver input file (.inp) and output file (.odb and *.fil, Table 6-1) and as Moose input file (*.i) and output database (*.dat, Table 6-2). Three test scenario with arbitrary boundary condition scenarios, and a final model are presented. 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 6-1 Abaqus example files

Short explanation of the files provided for an exemplified calibration.

File Name	Explanation
test_calibration.fil	Three model scenarios with different boundary conditions. Abaqus® output database for loading in Tecplot on Linux.
test_calibration.inp	Three model scenarios with different boundary conditions. Abaqus® input file.
test_calibration.odb	Three model scenarios with different boundary conditions. Abaqus® output database for loading in Tecplot on Windows.
test_model.geom	Geometry of the geomechanical-numerical root model.

Table 6-2 Moose example files

Short explanation of the files provided for an exemplified calibration.

File Name	Explanation
test_calibration_1.i	First (out of three) model scenario Moose input file.
test_calibration_1_out_0001.dat	First (out of three) solved model scenario.
test_calibration_2.i	Second (out of three) model scenario Moose input file.
test_calibration_2_out_0001.dat	Second (out of three) solved model scenario.
test_calibration_3.i	Third (out of three) model scenario Moose input file.
test_calibration_3_out_0001.dat	Third (out of three) solved model scenario.
test_model.inp	Test model geometry provided as Abaqus input file that can be read by Moose.

The output databases can be directly loaded in Tecplot 360 EX and FAST Calibration 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 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 received funding 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)" project by the Federal Ministry of Education and Research (project number 02NUK053D), the Helmholtz Association and the Helmholtz Centre Potsdam - Deutsches GeoForschungsZentrum GFZ.

References

Heidbach, O., Rajabi, M., Reiter, K., Ziegler, M., & the WSM Team. 2016. World Stress Map Database Release 2016. GFZ Data Services. https://doi.org/10.5880/WSM.2016.001

Heidbach, O., Rajabi, M., Cui, X., Fuchs, K., Müller, B., Reinecker, J., Reiter, K., Tingay, M., Wenzel, F., Xie, F., Ziegler, M., Zoback, M.-L., & Zoback, M. 2018. The World Stress Map database release 2016: Crustal stress pattern across scales. Tectonophysics, 744, 484-498. https://doi.org/10.1016/j.tecto.2018.07.007

Heidbach, O., Stromeyer, D. & Ziegler, M.O. 2020. Manual of the Tecplot 360 Add-on GeoStress v2.0. World Stress Map Technical Report 20-01, GFZ Data Services. 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.; Rajabi, M.; Zimmermann, G.; Müller, B. & Tingay, M. 2020. An open-access stress magnitude database for Germany and adjacent regions. Geothermal Energy. 8. 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. GFZ Data Services. https://doi.org/10.5880/wsm.2020.001

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

Ziegler, M. O. & Heidbach, O. 2021. Manual of the Matlab Script FAST Calibration v2.0. World Stress Map Technical Report 21-02, GFZ German Research Centre for Geosciences. https://doi.org/10.48440/wsm.2021.002

Ziegler, M. O. & Heidbach, O. 2021. Matlab Script FAST Calibration v2.0. GFZ Data Service. https://doi.org/10.5880/wsm.2021.002