4a Summer School Day 8

Python: a powerful tool with VALIMAT®, user defined material cards/specimen

B. Hirschmann, H. Pothukuchi, Ch. Schober
Traboch, 17.07.2020

Summer School

2nd week - Advanced topics

14. July - Evaluating and checking test data
interpretation of typical results

15. July - general yield surface (*MAT_187) and other material models,
failure approaches and comprehensive Autofit setup

16. July - Fiber reinforced plastics and their modelling approach
an extensive guide

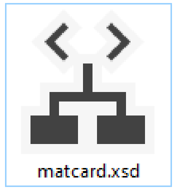
17. July - Python: a powerful tool with VALIMAT®,
user defined material cards/specimen

4a Summer School

Content of session 8

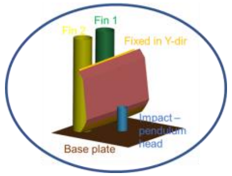
Python Interface

User Defined Material Cards



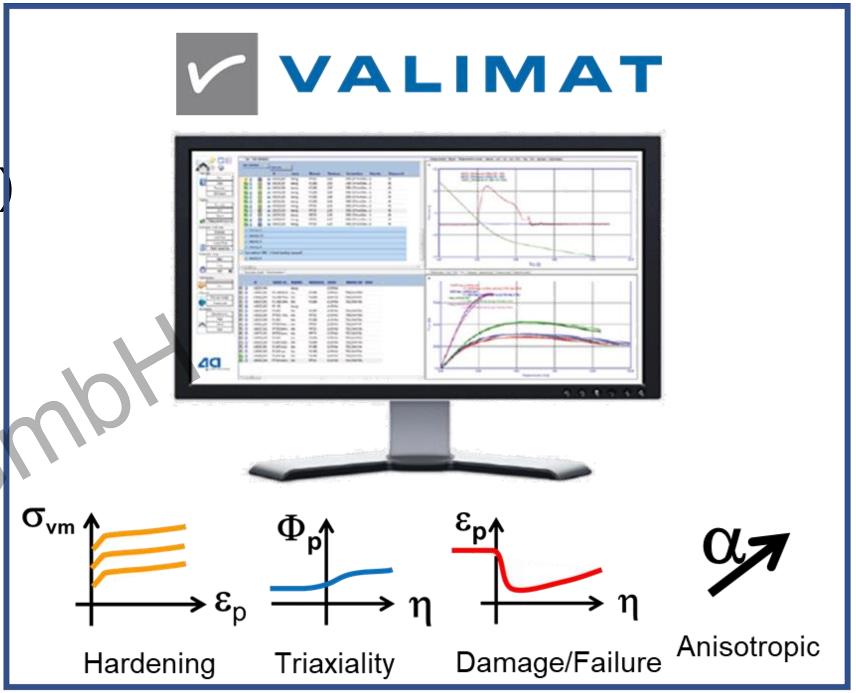
User Defined Specimen

User Defined Input deck



What is possible with the Python interface

- Access all values that are stored in the VALIMAT® database (read-only)
- Read the raw measurement data for custom evaluation
- Read the evaluated data
- Read the simulation results
- Add custom packages that are not distributed with Valimat



Setting up python in VALIMAT®

Valimat ver. 3.7

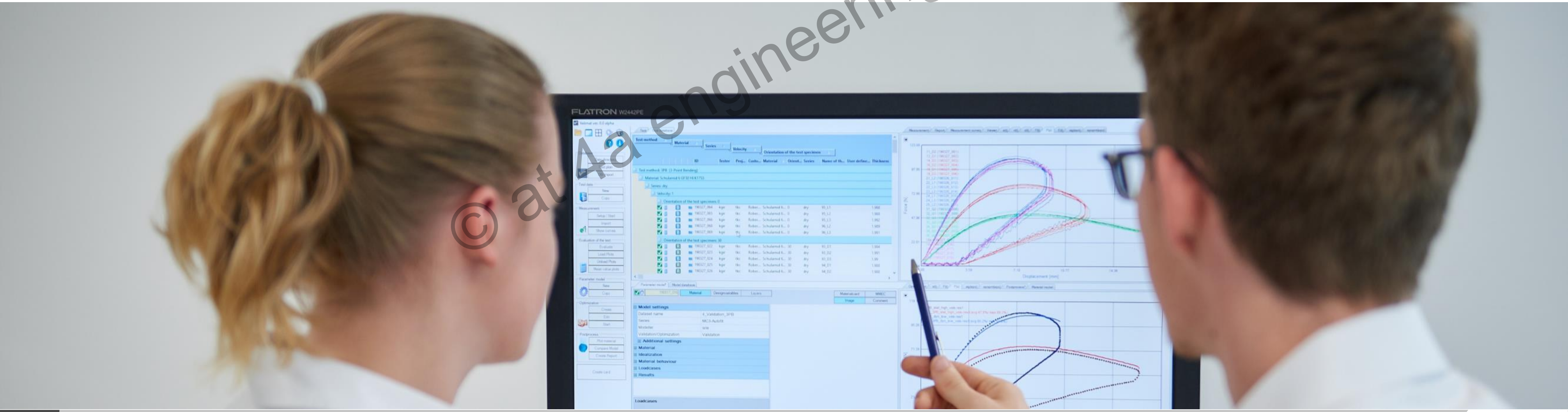
Links/references	
System links	
Temp	C:\Temp
Python executable	C:\Python27\python.exe
Python executable for user-scripts	python
Python scripts for Windows	..\settings\python
Python scripts for Linux	path/to/global_settings/python
Folder for Templatedatabases	

External programs	
Video Temp	C:\EXF1TEMP
7zip	C:\Programme\7-Zip\7z.exe
Batchrunscript	
Test python packages	.\python\evaluation_scripts_tests
Model python packages	.\python\evaluation_scripts_model

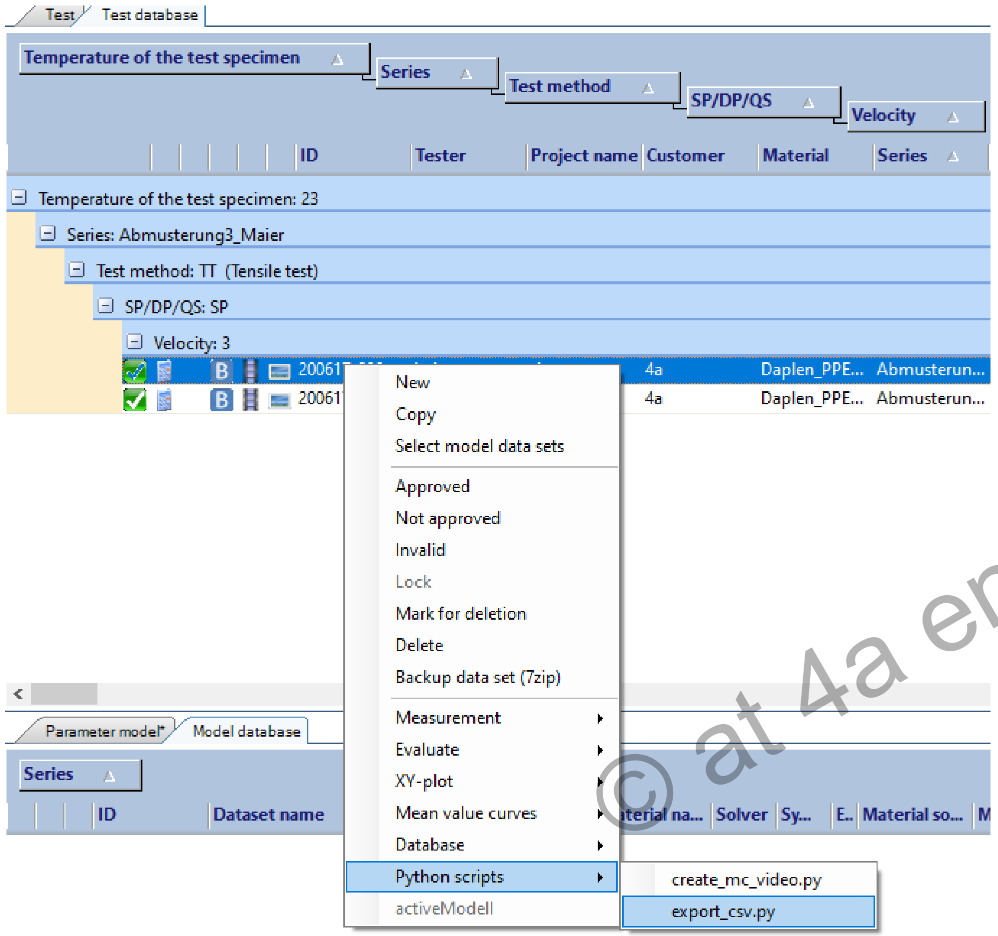
Überschrift

- Make sure that a Python 2.7 executable is properly linked in System links.
- Python scripts can be made callable from the context menu for the test cases and the model cases.
- The path to the script directory is set at:
  - Test python packages
  - Model python packages:
- In **VALIMAT 3.8** Python will be included in the local software installation.
- No additional installation necessary!

Live demo Example export_csv.py



Setting up python in VALIMAT®



Überschrift

To execute a python script:

1. Select the Tests or Models of interest
2. Open the context menu (RMB)
3. The available Python files can be found at "Python scripts"

General structure of a python script for VALIMAT®

- Basic structure can be seen in the prototype file (_prototype.py_)
- Edit the function main according to your needs
- The part that should be altered is enclosed in a block of hash letters

Import & Definitions → **Main** → **Parameter parsing**

```
import os
import platform
import inspect
import sys
import math
import subprocess
```

```
##############################################################################
##############################################################################
def main(base_path, table_type, table_name, work_dir, db_path, ids):

    # Please remove this part if you do not need any Database values
    rows = readDBData(base_path, table_type, table_name, work_dir, db_path, ids)

    if table_name == "ANALYSIS":
        for row in rows:
            print(row.id)
            print(row.density)
            print(row.name)
    elif table_name == "TESTS":
        for row in rows:
            print(row.id)
            print(row.moisture)
            print(row.temperatur)

##############################################################################
##############################################################################
```

```
##############################################################################
##############################################################################

if __name__ == "__main__":

    error = 0

    # check arguments
```

check and convert command line arguments for the main function call

Main function call

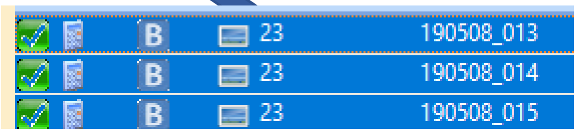
command line arguments

- VALIMAT® MDB directory path
- tabletype (either curvestore or model)
- VALIMAT® database name
- ids

The main function:

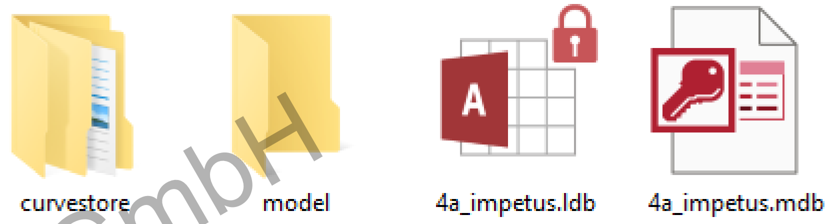
```
def main(base_path, table_type, table_name, work_dir, db_path, ids):
```

- base_path: location of VALIMAT® MDB
- table_type: either TESTS or ANALYSIS
- table_name: VALIMAT® database name “4a_impetus.mdb”
- work_dir: base_path+table_type
- db_path: base_path+table_name
- ids: list of test or model ids ([‘190508_013’, ‘190508_014’, ‘190508_015’])



VALIMAT® database structure overview

- curvestore
  - Raw data of measurement
  - Channels
  - Evaluated test curves
  - Measurement videos/pictures
- model
  - input files
  - material cards for optimization with lsopt
  - average curves for optimization (oavg_”casename”.”specifier”, in simulation units)
  - 4a_impetus_sampling directory (lsopt results)
  - case_” casename” (lsopt results)
    - StageResults
    - directories containing the simulation models
      - .xy simulation curve files, in simulation units



VALIMAT® database access

Be sure to have the following executable in your script directory:

- extract_values_from_db.exe

- In your script do the following:

```
def main(base_path, table_type, table_name, work_dir, db_path, ids):
    extract_call = [db_extract, db_path, base_path, table_type]
    extract_call.extend(ids)
    subprocess.call(extract_call)
    data = readDBData(base_path, table_name, ids)

    for curr_test in data:
        sssr=curr_test.stressstrainstrainrate
```

VALIMAT® 3.8 database access

The new VALIMAT® module allows read only access to the database

Database Access for tests:

- Import the VALIMAT® module

```
from Valimat import *
from Valimat.DatabaseAccess import *
```

- Examplary database access for a test script

```
def main(base_path, table_type, table_name, work_dir, db_path, ids):
    '''Access Database values and curves for TESTS
    '''
    DB = Database(db_path) #create a database object of the given Valimat Database
    tests = DB.GetTests(ids) #Make a list containing the reference to the given test objects
    for test in tests:
        curr_tc_force_curve=test.Curves.Force  #measurement curve of current test in current case
        print('Young\'s modulus of '+str(test.ID)+' is '+str(test.RES_MODUL))
```

VALIMAT® 3.8 database access

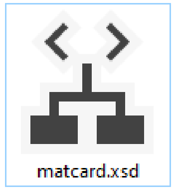
- Examplary database access for a model script

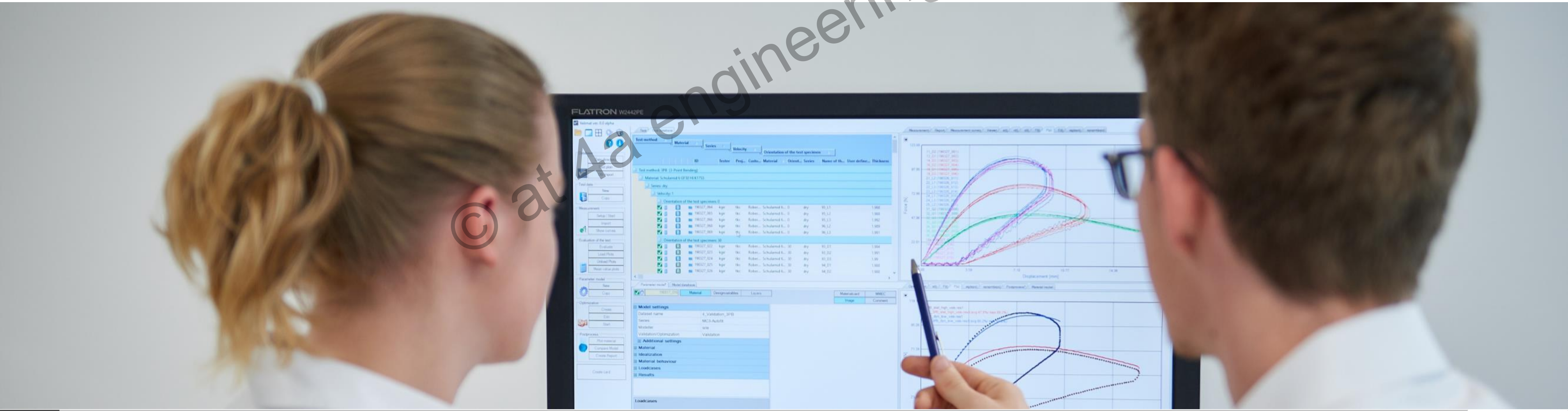
```
def main(base_path, table_type, table_name, work_dir, db_path, ids):
    '''Access Database values and curves for ANALYSIS
    '''
    DB = Database(db_path) #create a database object of the given Valimat Database
    models = DB.GetModels(ids) #Make a list containing the reference to the given model objects
    for model in models:
        for case in model.CASES:
            curr_sc_force_curve=case.Simulation.CurveForce #simulation curve of current case
            for test in case.Tests:
                curr_tc_force_curve=test.Curves.Force  #measurement curve of current test in current case
                print('Young\'s modulus of '+str(test.ID)+' is '+str(test.RES_MODUL))
```

Summary

- Python can be used to extend Valimat
- Almost no limit for the creativity of the user
- 2 Examples are delivered with Valimat
  - export_csv.py
  - create_mc_video.py
- Prototype file for a rapid start into the development
- Easy to start the script out of Valimat

User Defined Material Cards in VALIMAT®



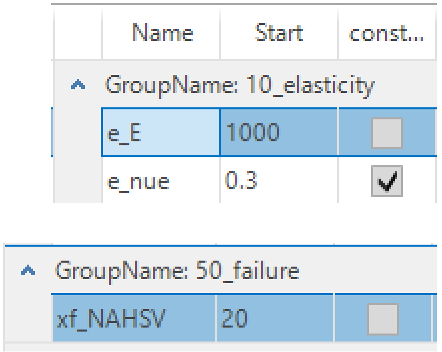


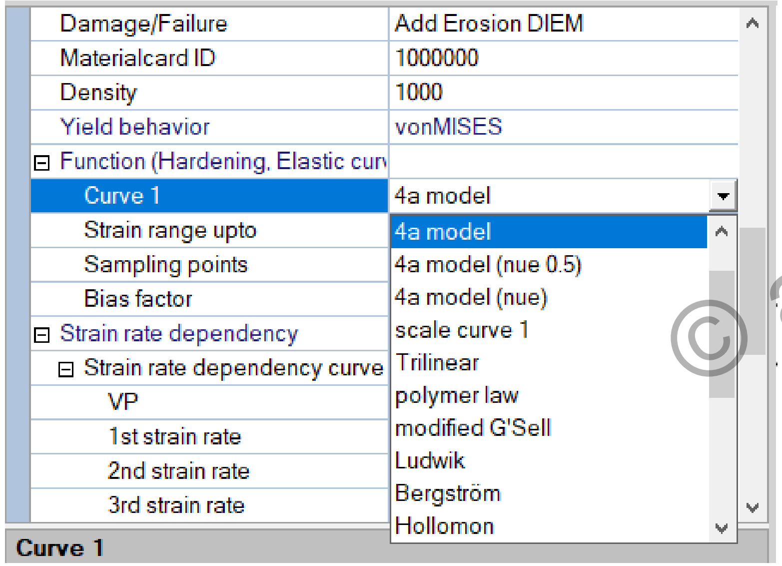
User Defined Material Cards - Objective

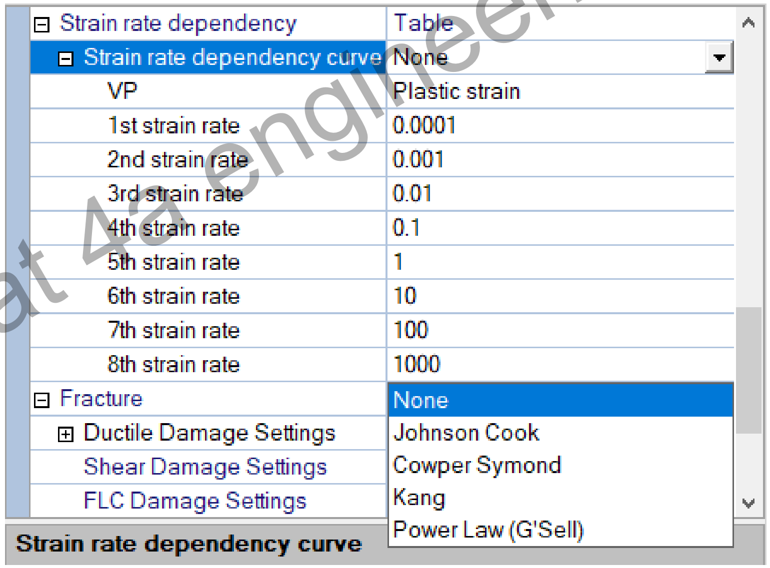
- Provide an overview of the capabilities of VALIMAT® user material card feature
- Show how to use user defined material cards
- Describe the elements of a user defined material card
- Provide some tips for implementing your own user defined material card

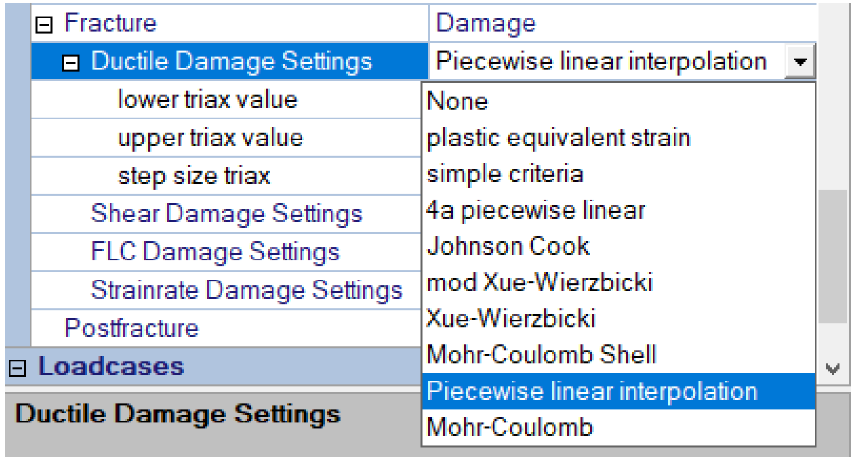
Capabilities of VALIMAT® .xml material card format

- Add other material models
- Use standard VALIMAT® Design variable groups
  - Transfer from model to another (Change solver, material card,…)
- Enter user defined variables
- Use implemented curves/tables for hardening, failure,…







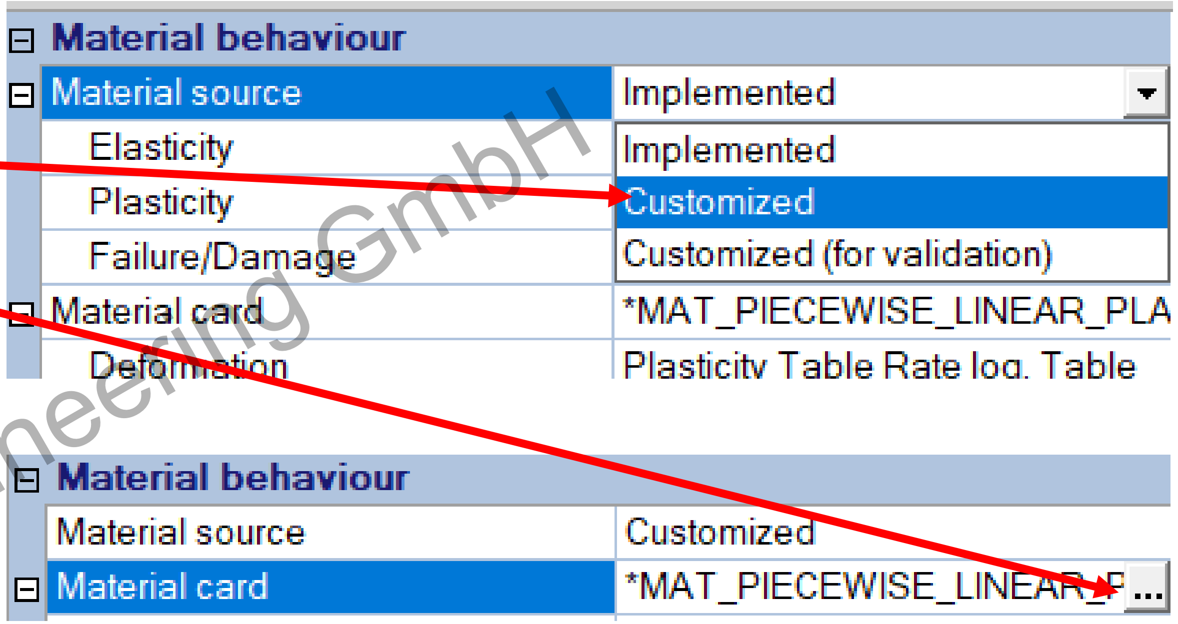


How to use .xml material cards

In Material behavior:

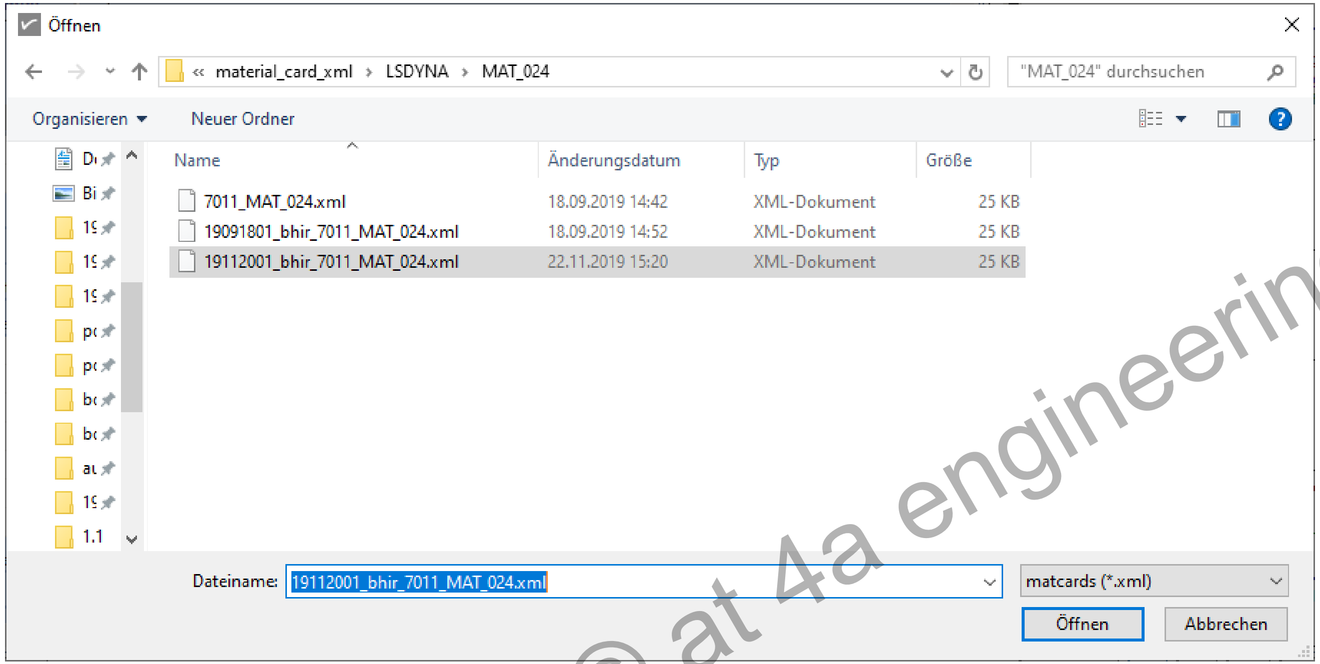
1.) Set Material source to customized

2.) next click on Material card field



How to use .xml material cards

3.) select the .xml material card file

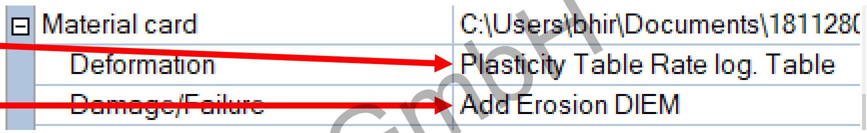


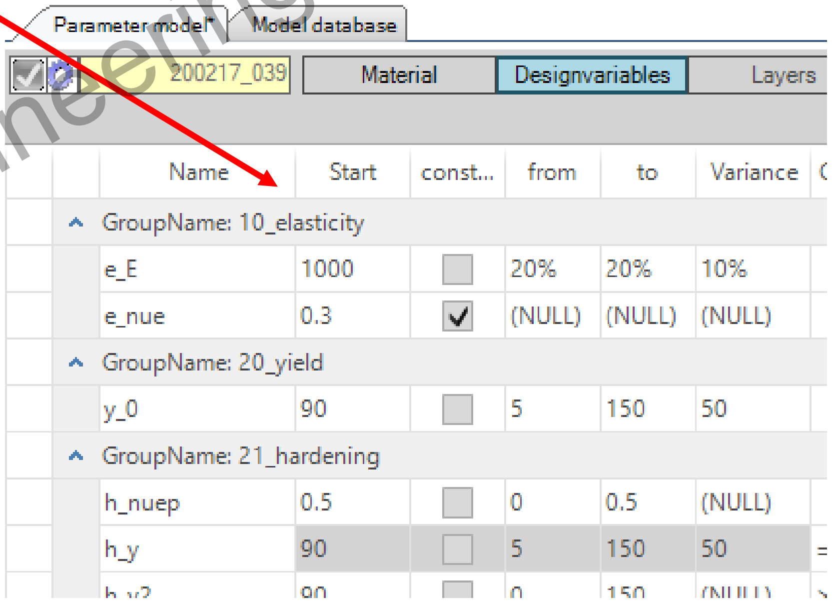
How to use .xml material cards (tips for switching)

4.) Set the correct settings for user defined material card (**<u>old settings are unfortunately lost</u>**)

Do the following:

1. Set Deformation
2. Set Damage/Failure
3. Then paste old Design variables





xml schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Material: MAT1
Author: Bernhard Jilka
Date: 03.08.2016
Version:1
-->
<matcard xmlns="urn:4a:impetus:matcard"
 xmlns:xsi="http://www.w3.org/2001/XMLSc
 xsi:schemaLocation="urn:4a:impetus:matc
 name="*MAT_ELASTIC (*MAT_001)" A_MAT_TY
<matcard cases>
<matcard output>
<damage cases>
<damage output>
</matcard>
```

Encoding information

Comments

XML Schema Definition

Variables for the whole material card

case specific variables

what is written into the material.inp file

failure case specific variables

what is written into the material.inp file

IMPETUS® vs ls_opt formula

impetus_formula are used to create the **static** part of a material card (no changes in the optimization runs)

- Use **<u>only</u>** VALIMAT® database variables
- Examples:

```
<impetus_formula formula="db_mattyp"/>
<impetus_formula formula="ID_MAT" format="0D8S"/>
<impetus_formula formula="db_rho" format="3D10S"/>
```

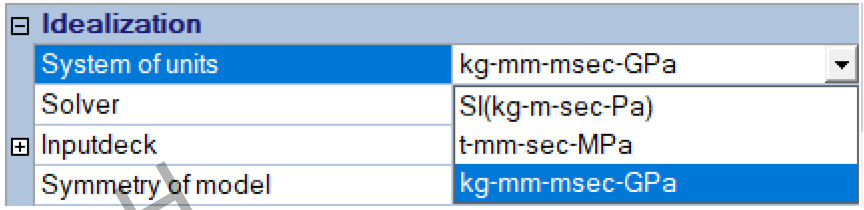

ls_opt formula create the **dynamic** part of a material card (lsopt replaceable code, dependant from design variables)

- Use **<u>only</u> LS-Opt variables**
- Examples:

```
<ls_opt formula="e_E*US_stress" format="0D10S"/>
<ls_opt formula="e_nue" format="0D10S"/>
```

unit systems

- In VALIMAT® we support 3 types of unit systems (Variables are declared in t-mm-sec-MPa):



- db_vars are always converted to current unit system!

- The variables are dependent from the unit system and the time scaling.

conversion factors	
US_length	US_stiffness
US_time	US_force
US_density	US_energy
US_strainrate	US_stress
US_velocity	

- Example: Young’s modulus conversion:

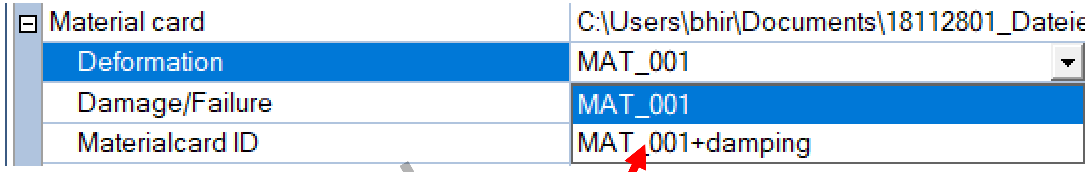
```
<ls_opt formula="e_E*US_stress" format="0D10S"/>
```

user variable feature

Add user variable to the LS-Opt variables

- Define in matcard_vars
  - case:
  - name: variable name (naming convention xm_(matcard) or xf_ (failure))
  - description: Description
  - group: GroupName
  - position: unique position for ordering
  - static:constant either ”true” or “false”
  - startvalue: Start
  - lowerbound: from
  - upperbound: to
  - optimizationwindow: Variance
  - boundary_condition: Condition
- Use in ls_opt formula by name

```
<ls_opt formula="xm_da" format="0D10S"/>
```




```
<matcard_vars>
<designvar case="2" name="xm_da" de
<designvar case="2" name="xm_db" de
</matcard_vars>
<matcard_output>
```

Name	Start	const...	from	to	Variance	Condition	Description
GroupName: 10_elasticity							
GroupName: 90_damping							
xm_da	0.0	✓	(NULL)	(NULL)	(NULL)		axial damping factor
xm_db	0.0	✓	(NULL)	(NULL)	(NULL)		Bending damping factor

Click here to add a new row

table input (arrays)

- epp (equivalent plastic/total strain)
  - strain range upto defines the endpoint of the curve
  - Sampling points defines number of points in the curve
  - Bias factor defines a bias to the front end of the curve
    - Bas factor=1: equally distributed points
- triax (stress triaxiality)
  - lower triax value to upper triax value with step size triax
  - typical values: plane stress state [-2/3;2/3;1/9]
- strain rate dependency:
  - db_epspkt1➔db_epspkt8
  - typical values: (LS-DYNA/PAMCRASH [0.001;1000;0;…]; ABAQUS [0.0;0.001;1000;…])

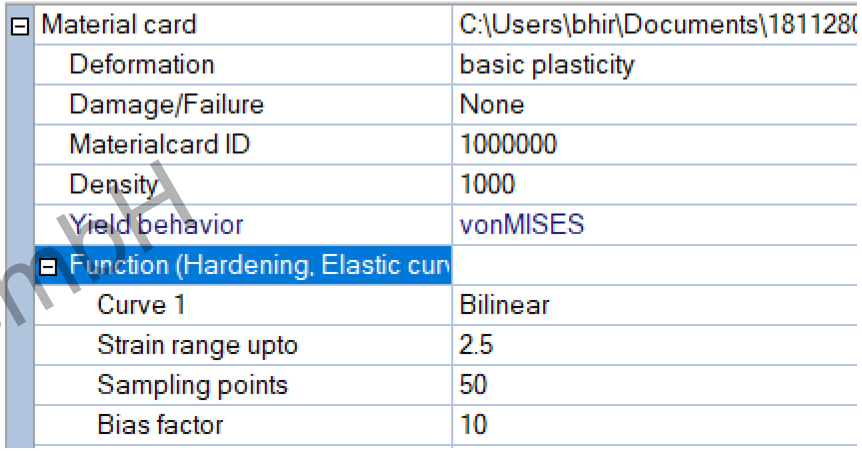
Yield behavior	vonMISES
⊟ Function (Hardening, Elastic cur	
Curve 1	Bilinear
Strain range upto	2.5
Sampling points	50
Bias factor	10

⊟ Fracture	Damage
⊟ Ductile Damage Settings	Mohr-Coulomb
lower triax value	-0.66
upper triax value	0.66
step size triax	0.11

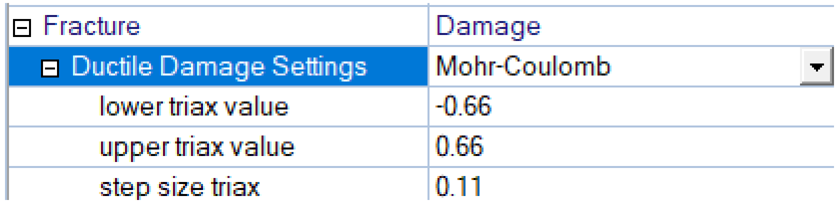
⊟ Strain rate dependency	Table
⊟ Strain rate dependency curve	None
VP	Plastic strain
1st strain rate	0.0001
2nd strain rate	0.001
3rd strain rate	0.01
4th strain rate	0.1
5th strain rate	1
6th strain rate	10
7th strain rate	100
8th strain rate	1000

curve definition (arrays)

- hardening curve: sig; s2g; s3g 🡐 result of Curve 1;2;3 (epp)
  - number of curves: “A_MAT_TYPE_PLASTIC_enum”=
    - 0: “none_0”
    - 1: “vonMises_11”; “vonMises_12”; ”Hillr2D_51”; ”HillR3D_52”; “Hill3D_53”; “Hill2D_54”; “RaghavaHill2D_55”
    - 2: “DruckerPrager_21”; “Raghava_22”
    - 3: “GenYLD3_31”; ”GenYLD5_32”



- ductile damage: fail_ductile 🡐 Ductile Damage Settings (triax)
  - availability depends on “A_MAT_FRAC_DIEM_DUCTILE”



*See “matcard.xsd” for available options and “dv_and_curve_def.xml” for VALIMAT names, variables and function definitions.

impetus_material curve feature

impetus_material curve definitions allow the creation of curves

xVal: arithmetical expression with an array

yVal: arithmeticel expression with an array

```
<impetus_materialcurve xVal="epp" yVal="sig" format="10D20S"/>
```

⊟ Function (Hardening, Elastic curv	
Curve 1	Bilinear
Strain range upto	2.5
Sampling points	50
Bias factor	10

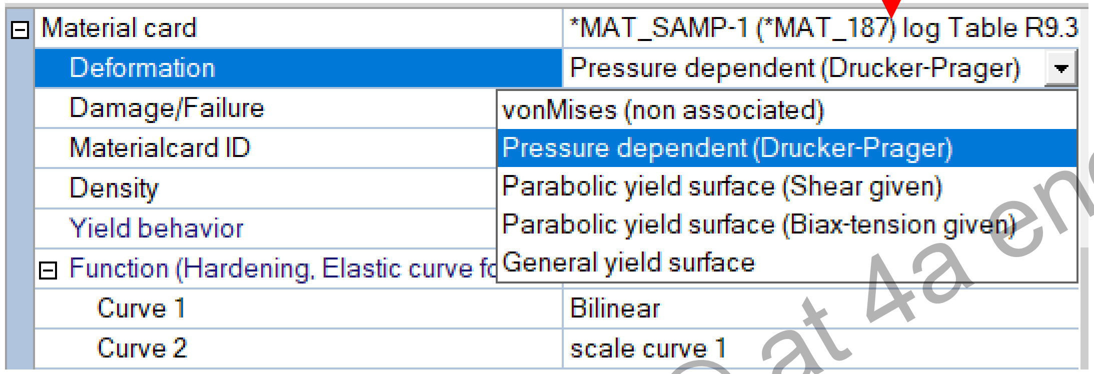
```
*DEFINE_CURVE
$#     lcid       sidr        sfa        sfo       offa       offo     dattyp
   1000001          0          1          1          0          0
$#                  al                    ol
<<(0)*1+0:20>><<((h_y+h_ET*0)*(1+1/v_p*log(max(0.0001,v_epspkt)/v_epspkt)))*1+0:20>>
```

impetus_material curve feature

- Example: MAT_SAMP-1

```
name="*MAT_SAMP-1 (*MAT_187) log Table R9.3+" A_MAT_TYPE_ELASTIC="linearElastic_0" A_MOD_IDEALIZATION="all_2" A_SOLVER="LSDYN
<matcard_cases>
<case id="1" name="vonMises (non associated)" A_MAT_ELASTIC_CURVE="linearElastic_1" A_MAT_TYPE_PLASTIC="vonMises_11" A_MAT_TYE
<case id="2" name="Pressure dependent (Drucker-Prager)" A_MAT_ELASTIC_CURVE="linearElastic_1" A_MAT_TYPE_PLASTIC="Raghava_22"
<case id="3" name="Parabolic yield surface (Shear given)" A_MAT_ELASTIC_CURVE="linearElastic_1" A_MAT_TYPE_PLASTIC="GenYLD3_31
<case id="5" name="Parabolic yield surface (Biax-tension given)" A_MAT_ELASTIC_CURVE="linearElastic_1" A_MAT_TYPE_PLASTIC="Ger
<case id="4" name="General yield surface" A_MAT_ELASTIC_CURVE="linearElastic_1" A_MAT_TYPE_PLASTIC="GenYLD5_32" A_MAT_TYPE_VIS
</matcard_cases>
```

activates




```
<impetus_materialtable/>
<XMLIF mcase="2-4">
*DEFINE_CURVE
$#    lcid       sidr        sfa        sfo       offa       offo     dattyp
<impetus_formula formula="ID_FUNC10" format="0D10S"/>          0        1.0        1.0        0.0        0.0          0
$#                  a1                  o1
<impetus_materialcurve xVal="epp" yVal="s2g" format="10D20S"/>
</XMLIF>
```

adds this curve to card

impetus_materialtable feature

material table definitions allows for fast viscoplasticity definition

It creates a table definition with strain rates and hardening curve ids

The curves are a combination of the first material curve and the strain rate dependency model

Example: Bilinear hardening and Johnson Cook strain rate dependency

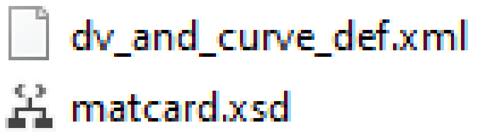
<impetus_materialtable/>

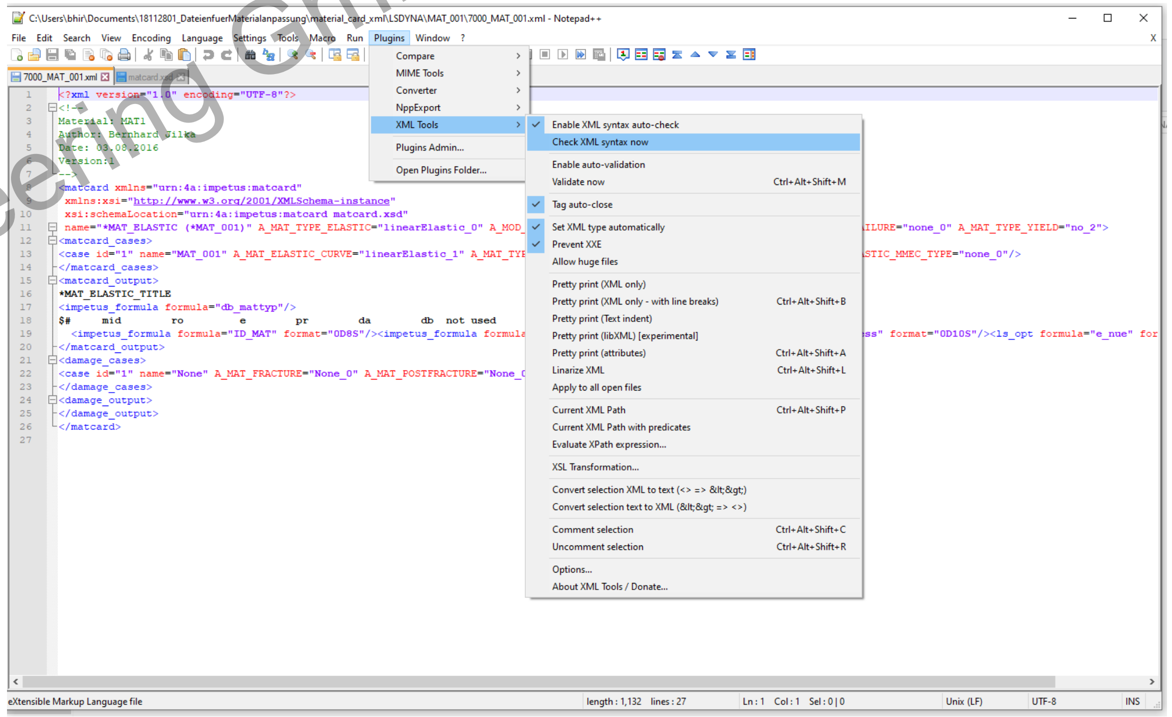
⊟ Function (Hardening, Elastic curv	
Curve 1	Bilinear
Strain range upto	2.5
Sampling points	50
Bias factor	10
⊟ Strain rate dependency	Table
⊟ Strain rate dependency curve	Johnson Cook
VP	Plastic strain
1st strain rate	0.0001
2nd strain rate	1000
3rd strain rate	0

```
*DEFINE_TABLE
$#     tbid
   1000000
$#               value      lcid
<<log(0.0001*1/US_time):20>>   1000001
<<log(1000*1/US_time):20>>   1000002
*DEFINE_CURVE
$#     lcid      sidr       sfa       sfo      offa      offo    dattyp
   1000001         0         1         1         0         0
$#                al                  ol
<<(0)*1+0:20>><<((h_y+h_ET*0)*(1+1/v_p*log(max(0.0001,v_epspkt)/v_epspkt)))*1+0:20>>
```

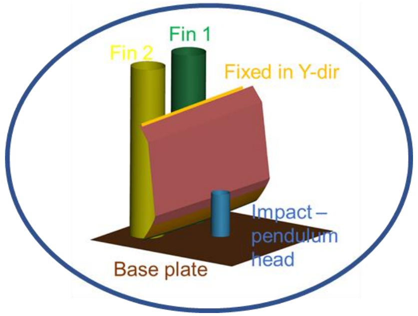
Tips for implementing a new material card for VALIMAT®

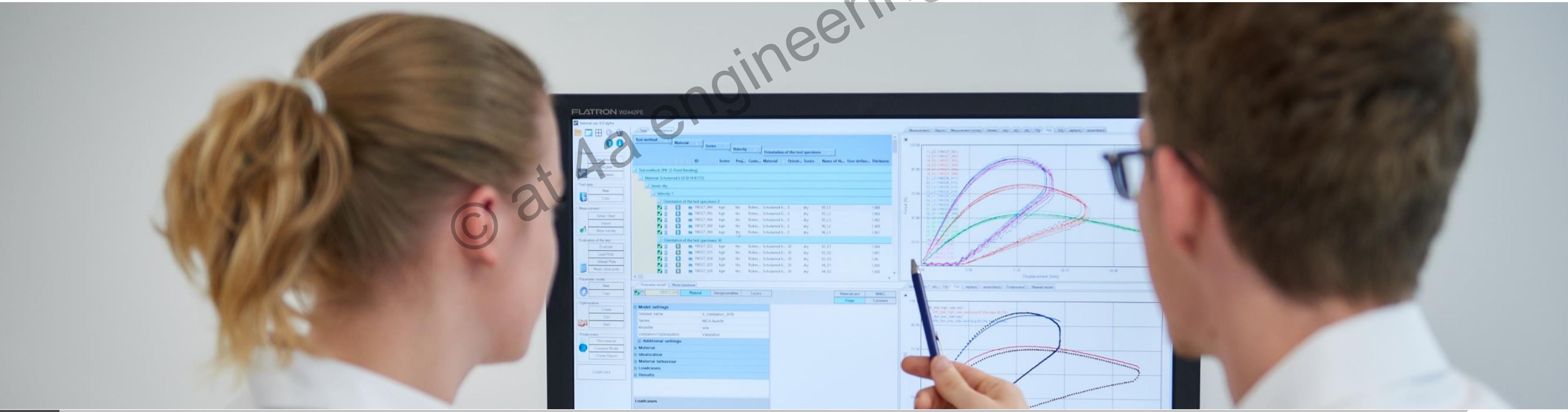
- For the text editor we use Notepad++, which has the plugin “XML tools” that allows to check the file for compliance with the schema file (Have a copy of “matcard.xsd” in the working directory).
- Doesn’t detect all problems!
- Variable definitions are in the dv_and_curve_def.xml





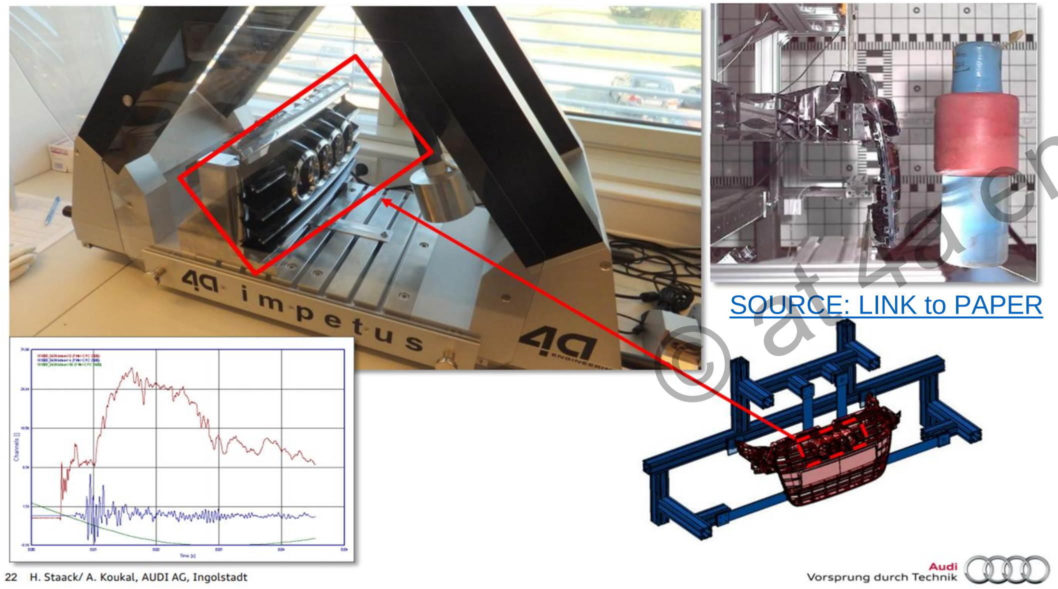
User Defined Specimen/Input Deck in VALIMAT®



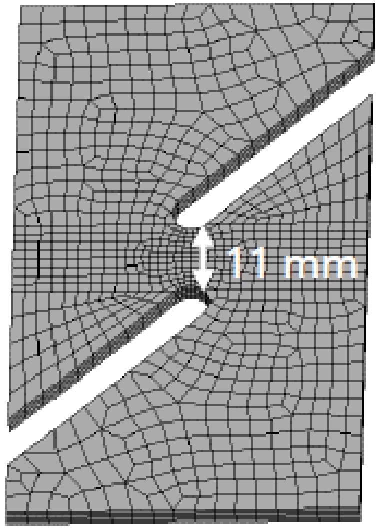


User Defined Specimen/Input Deck - Introduction

- Why do we need a user-defined specimen / input deck
  - To work with specimens that are not already implemented in VALIMAT®
  - User-defined specimen → Only the specimen type is new and needs to be incorporated in VALIMAT®
  - User-defined input deck → Flexible → for component tests as an example



Component test

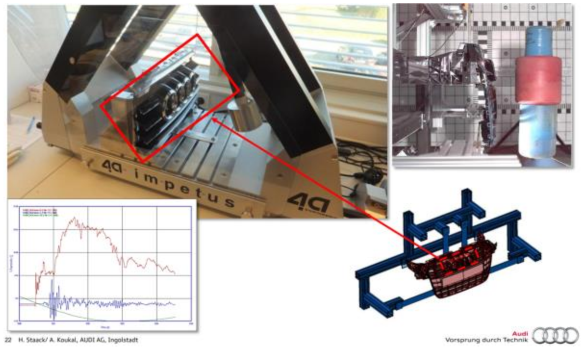


Shear tension specimen

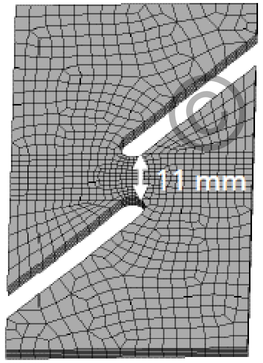
Introduction

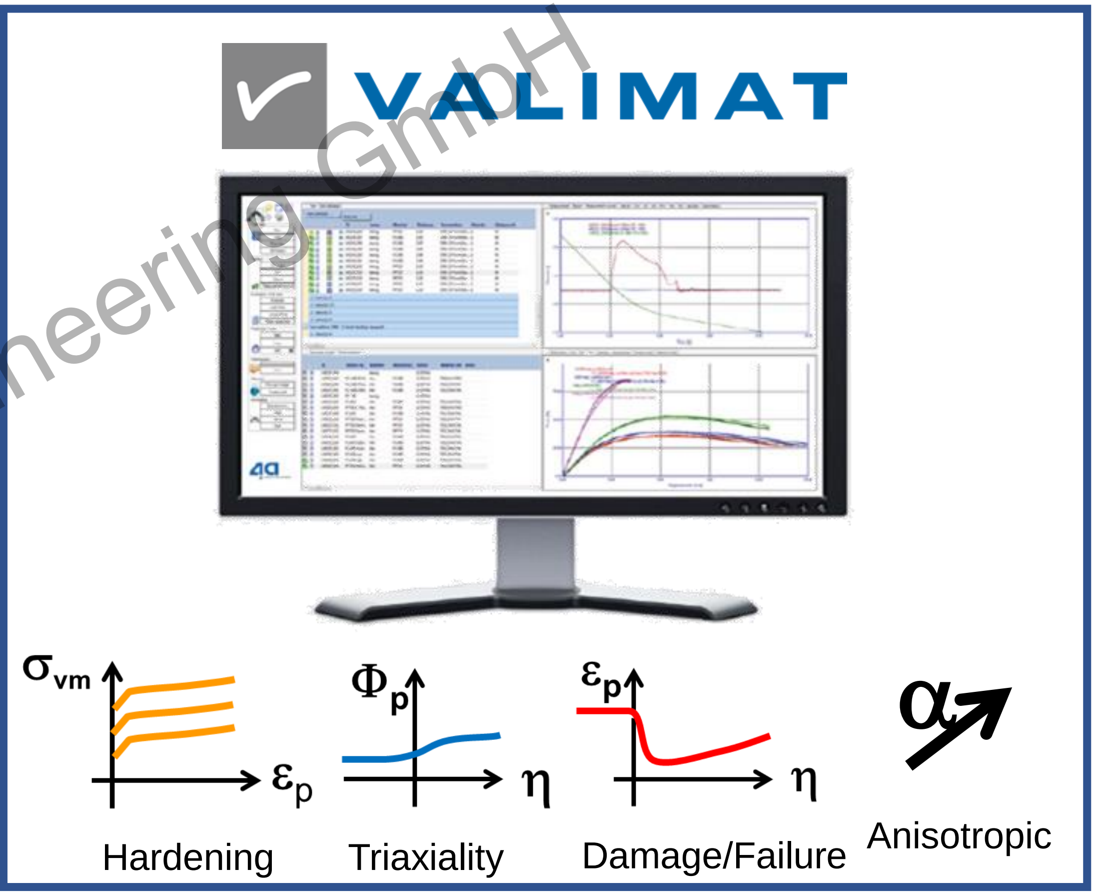
- How do we include new specimen types or custom test setups in VALIMAT®?

Component test



Shear tension specimen



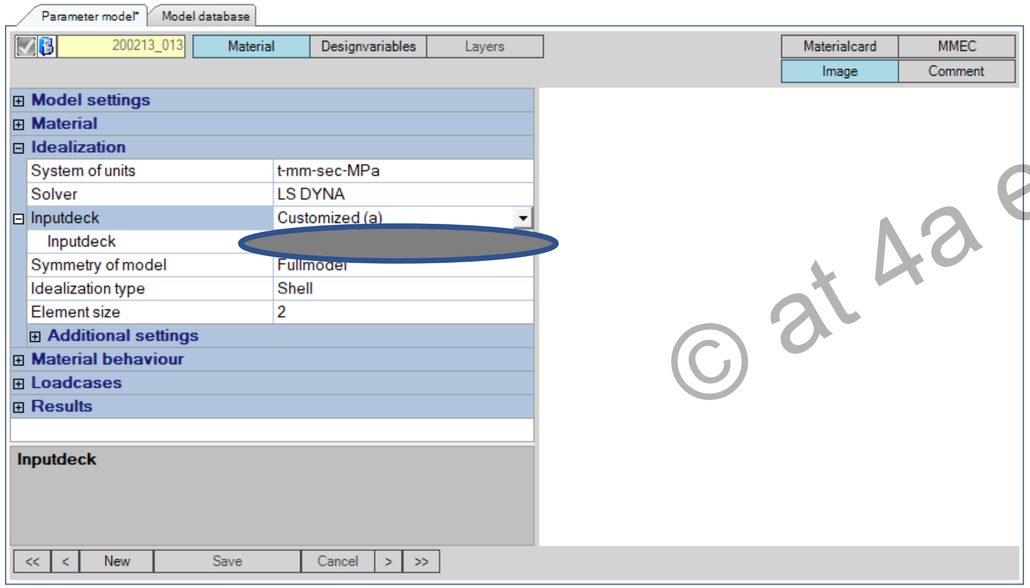


Implementation in VALIMAT®

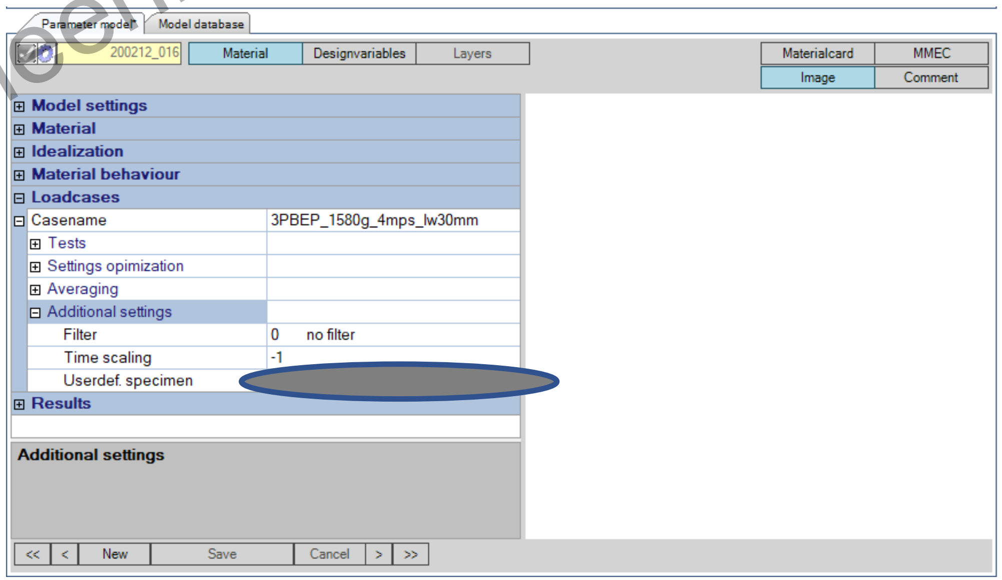
- In addition to the manual (chapter 5.4.3), the following presentation should help the user to create a user defined input deck and a user defined specimen
- You will need the following files:
  - File with geometry, boundary conditions, etc. (can be split into several files)
  - .Conf-file with commands for VALIMAT® for user defined inputdeck

Implementation in VALIMAT®

- Idealization → Inputdeck switch to 'customized' and name the .Conf file as Inputdeck
- All other files (Material, Geometry, Scripts) also need to be in this directory
- For a User-defined specimen → Switch to user defined specimen under Loadcases → Additional Settings



User defined input deck

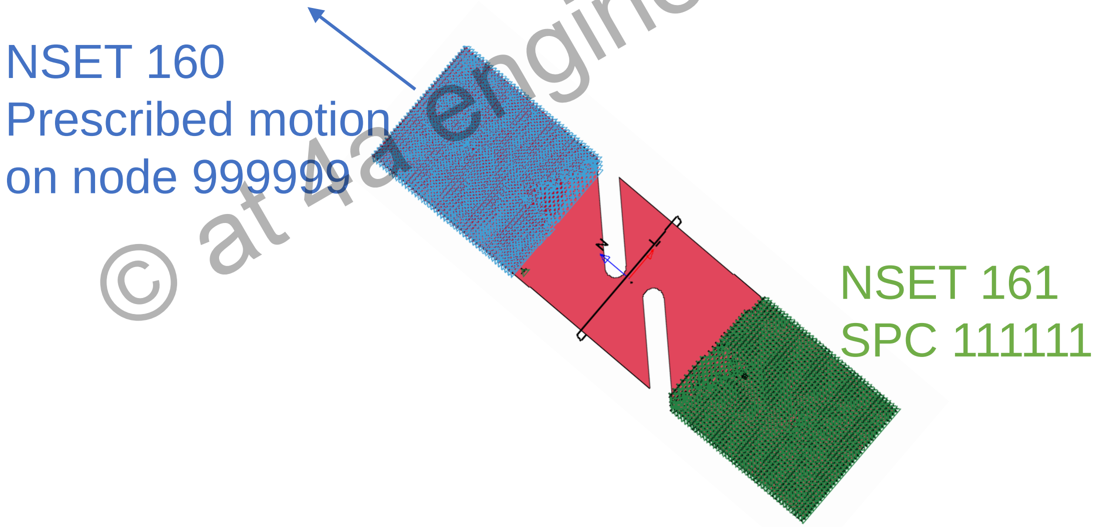


User defined specimen

Overview – User defined input specimen

General structure

- The test database is updated with the required fields from the tests
- The tests are linked in the model database with the correct settings for the averaging parameters
- Check the optimization curve generated from all the test results
- Carefully check for the entry in write part/section in the Idealization → Additional settings
- The elements in the user defined specimen mesh are renumbered and the right node set IDs are referenced in the *DATABASE_OUTPUT → displayed in VALIMAT®

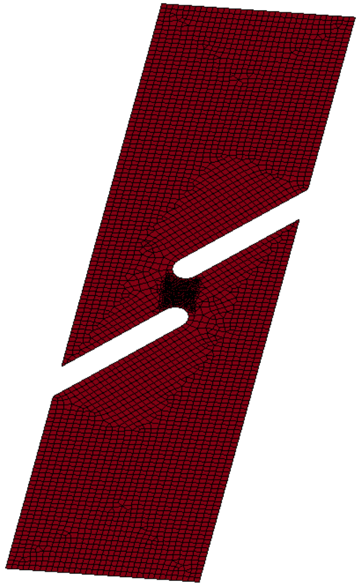


Overview – User defined input specimen

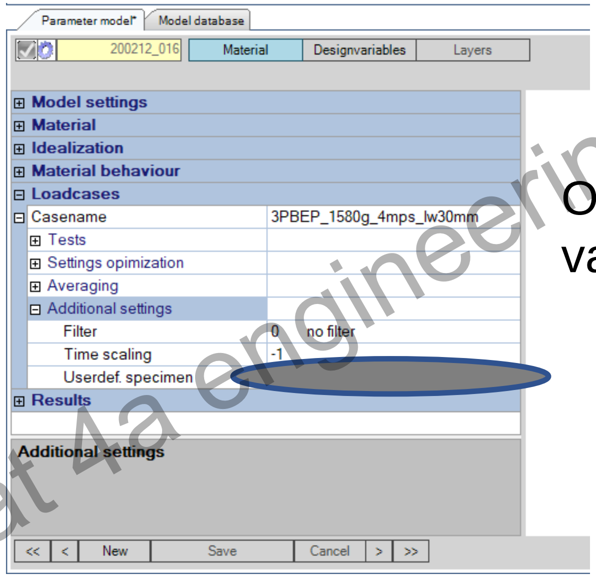
General structure

- A shear tension specimen that is not implemented in VALIMAT® → How do we use the user defined specimen feature to incorporate it in the software.

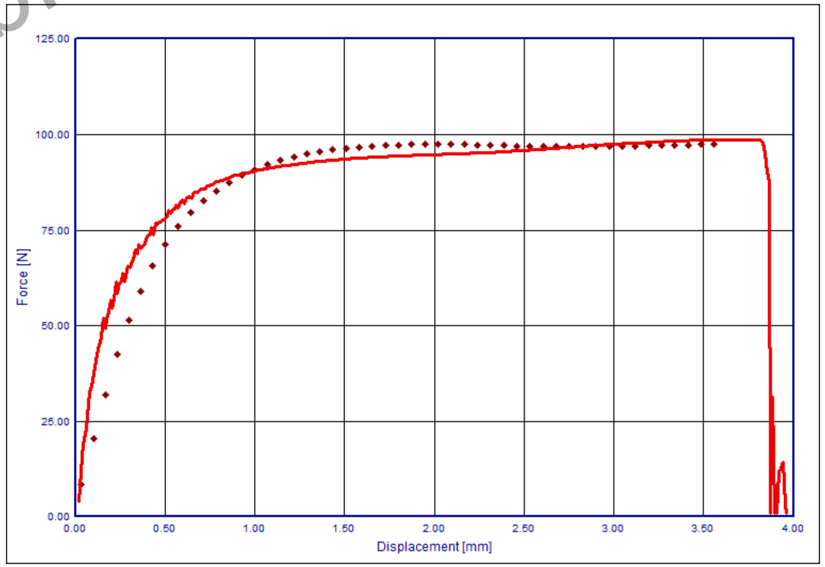
Step 1 : mesh, nodes, elements, sets



Step 2



Optimization/ validation



Renumbered PIDs, setIDs
VALIMAT® manual

Summary

- VALIMAT® plots depending on the load case and settings the following simulation results:
- For 3-point- bending tests:
  - Displacement of the node with the id 200000
  - Force: the contact force between the fin and the sample
  - stress/strain/strain rate results from the element with id 1 000 000 (Works for the implemented material models. For other materials the stored history variables might differ.)
  - Necessary Sets 140 & 141 (half model 150, 151; quarter model 152)

- Tensile test:
  - global displacement of the node with the id 200000 or 999999
  - local displacement: difference between node 999997 and 999998 for full model, 2*displacement 999998 for half and quarter model
  - Force: solver dependent either spc reaction forces or cross section forces
  - stress/strain/strain rate results from the element with id 1000000 (Works for the implemented material models. For other materials the stored history variables might differ.)
  - Necessary Sets 140,141, 160 & 161 (half model 150, 151; quarter model 152)

Overview – User defined input deck

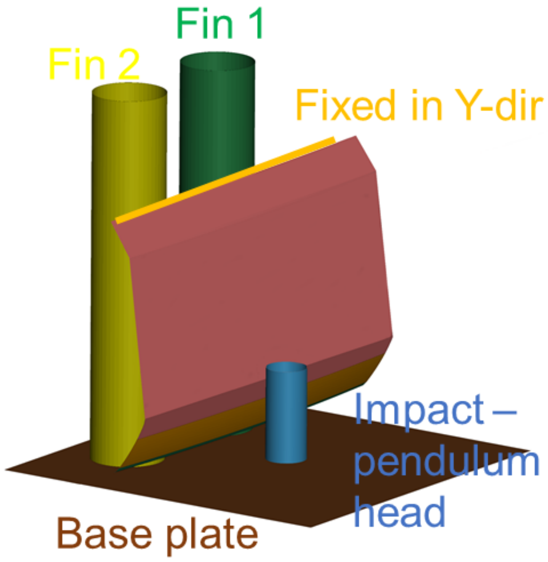
General structure

- A template folder containing:
  - A configuration file which will be selected in VALIMAT®(*.conf)
  - The main keyword file (suffix not a condition, but for readability a solver specific suffix is advised: *.k or *.key (LS-DYNA); *inp (ABAQUS), *.pc (PAMCRASH))
  - Conditional include files with the suffix *.inc. Commands in the main keyword file will lead to the inclusion of a subset of all the *.inc files in the main keyword file. *.inc references in *.inc references will have no effect. This allows for example to handle solid and shell idealization of the specimen.
  - Other input files with no conditions/parameters. For example meshes.
  - VALIMAT® python script for running the job and copying the files (Requires no modification and can be copied from any other model).

Overview – User defined input deck

General structure

- A component bending test with a non-standard setup that is not implemented in VALIMAT®.



Model

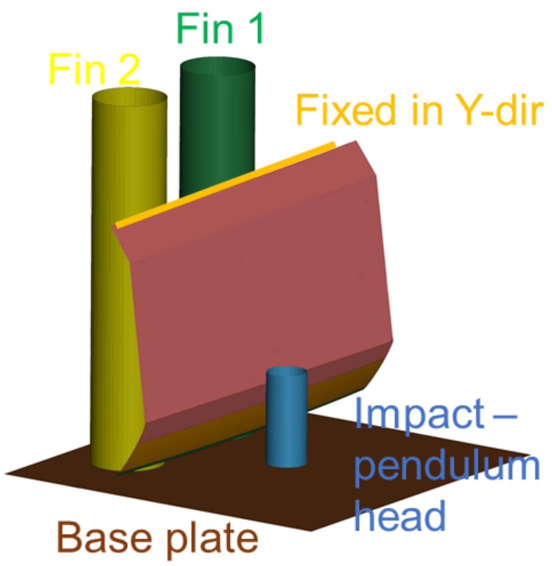
User defined input deck - content

Name	Änderungsdatum	Typ	
20020501_bhir_R40TubeSupport.conf	06.02.2020 15:07	CONF-Datei	Configuration file
vel_initial.inc	06.07.2017 19:24	INC-Datei	Supporting files
vel_prescribed.inc	06.07.2017 18:06	INC-Datei	
20020501_bhir_R40TubeSupport	06.02.2020 15:49	INP-Datei	Main keyword file
20020501_bhir_Fin	06.02.2020 12:15	KEY-Datei	Supporting files
20020501_bhir_R40TubeSupport	07.02.2020 12:33	KEY-Datei	
20020501_bhir_ulc_plate	06.02.2020 12:07	KEY-Datei	
20020501_bhir_ulc_tube1	06.02.2020 12:26	KEY-Datei	
20020501_bhir_ulc_tube2	06.02.2020 12:27	KEY-Datei	
20020502_bhir_hpot_ulc_fullmodel-nosides	06.02.2020 14:51	KEY-Datei	
copy_file	22.04.2016 12:28	Python File	python scipts for 4a IMPETUS®
run	14.06.2016 17:50	Python File	

Overview – User defined input deck

Example

- A component bending test with a non-standard setup that is not implemented in VALIMAT®.



Model

.conf file - definition

LS-DYNA solver command

Main keyword file

Material card (implemented or customized)

Supporting files – excluding the main file

Extraction of results

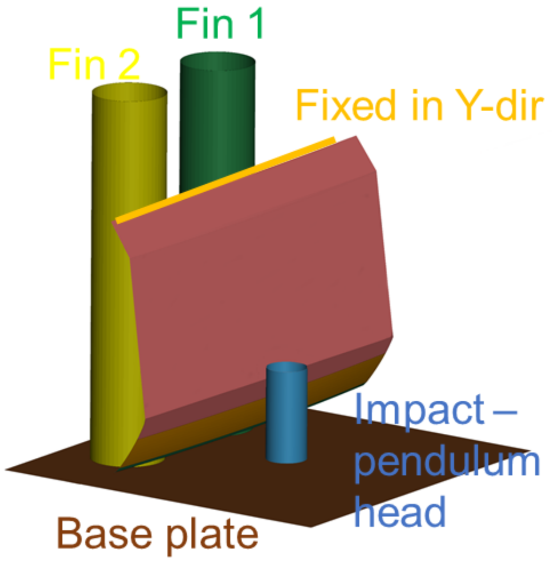
*Renumber PIDs, node and element sets in a VALIMAT® friendly format!

For optimization/validation/pre-simulation → the test database is updated and contains the necessary fields referenced in the loadcases tab under the model database

Overview – User defined input deck

Example

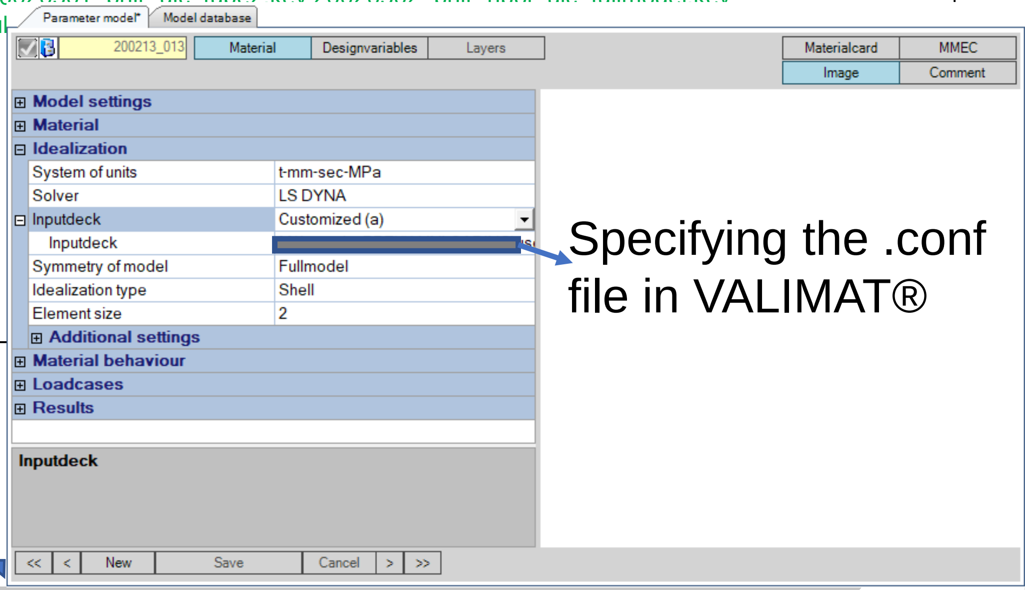
- A component bending test with a non-standard setup that is not implemented in VALIMAT®.



Model

.conf file - definition

```
#LSDYNA
#DEFINTION
impetus_command:
impetus_parameter:
solver_command:python ../../run.py -i UserOpt.inp -testtype 100 -oneElement <<db_oneElementSim>> -elmodell 1 -L0 <<db_s0>> -height <<db_h>> -width
<<if(db_l_Ten==0,db_b,db_b_Ten)>> -length <<db_l>> -pm <<db_pm>> -trueStressStrain <<db_trueStressStrain>> -filter
<<if(db_pm==3&db_timescale==0,0,db_filter)>>
solver_input:main.inp
solver_append:
solver_extraInpFiles:material.inp
solver_copy:copy_file.py run.py 20020501_bhir_R40TubeSupport.key 20020501_bhir_ulc_tube2.key 20020502_bhir_hpot_ulc_fullmodel.key
20020501_bhir_Fin.key 20020501_bhir_ulc_plate.key 20020501_bhir_ul
pre_command:
#-------------
#     force:
#-------------
res_force:1:python ../../copy_file.py force.xy LsoptHistory
#-------------
#     displ:
#-------------
res_disp:1:python ../../copy_file.py disp.xy LsoptHistory
#END FILE
```

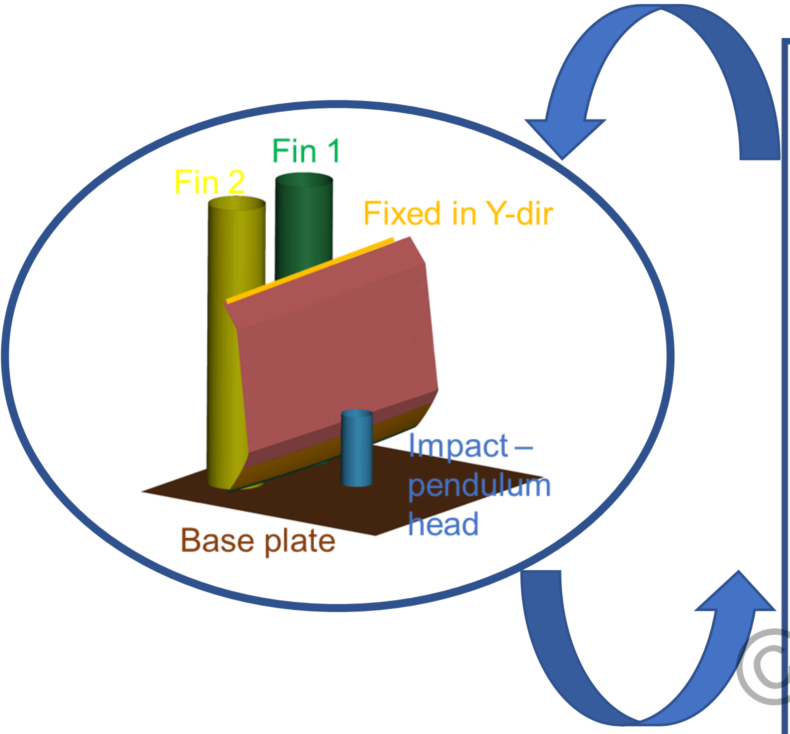
Specifying the .conf file in VALIMAT®

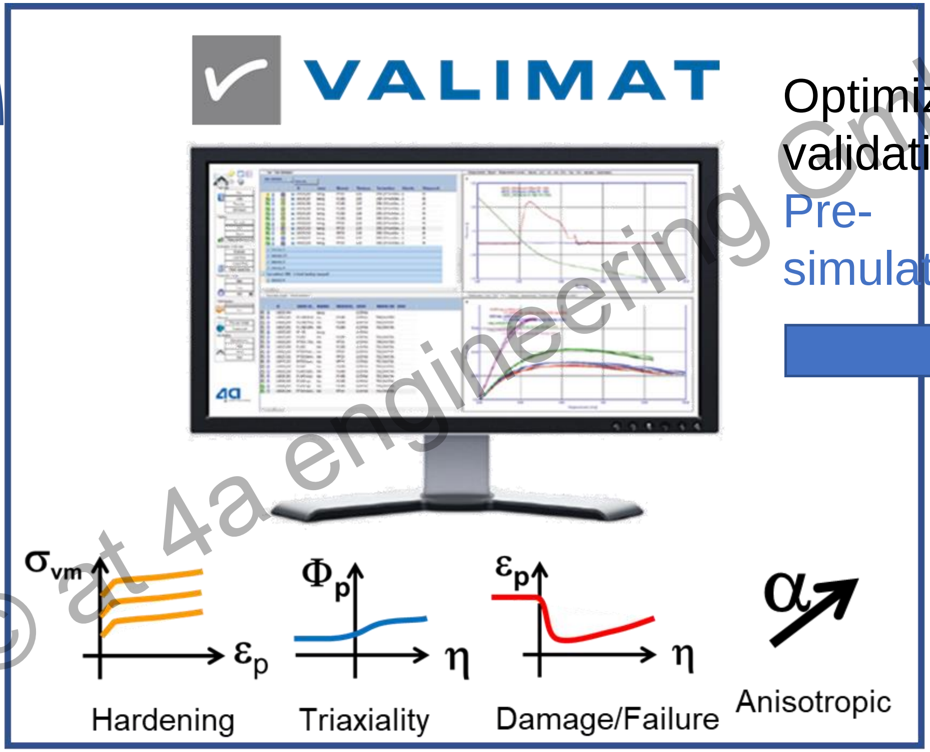
*ELEMENT_MASS → mass of the pendulum taken from the test database in VALIMAT®

Overview – User defined input deck

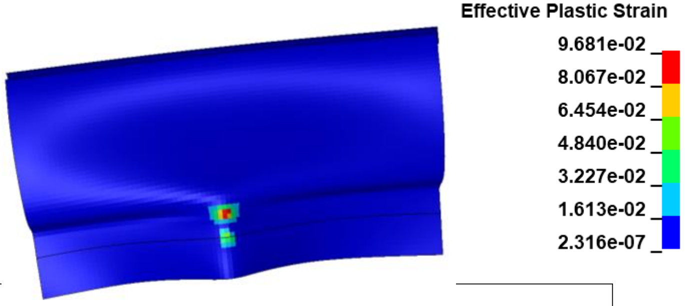
Example

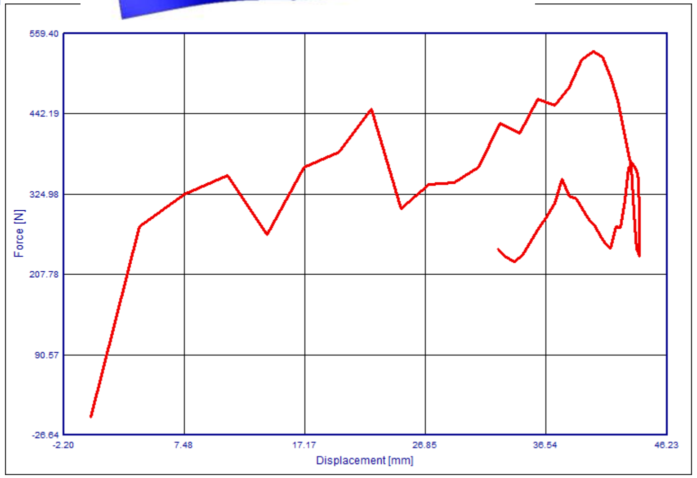
- A component bending test with a non-standard setup that is not implemented in VALIMAT®.





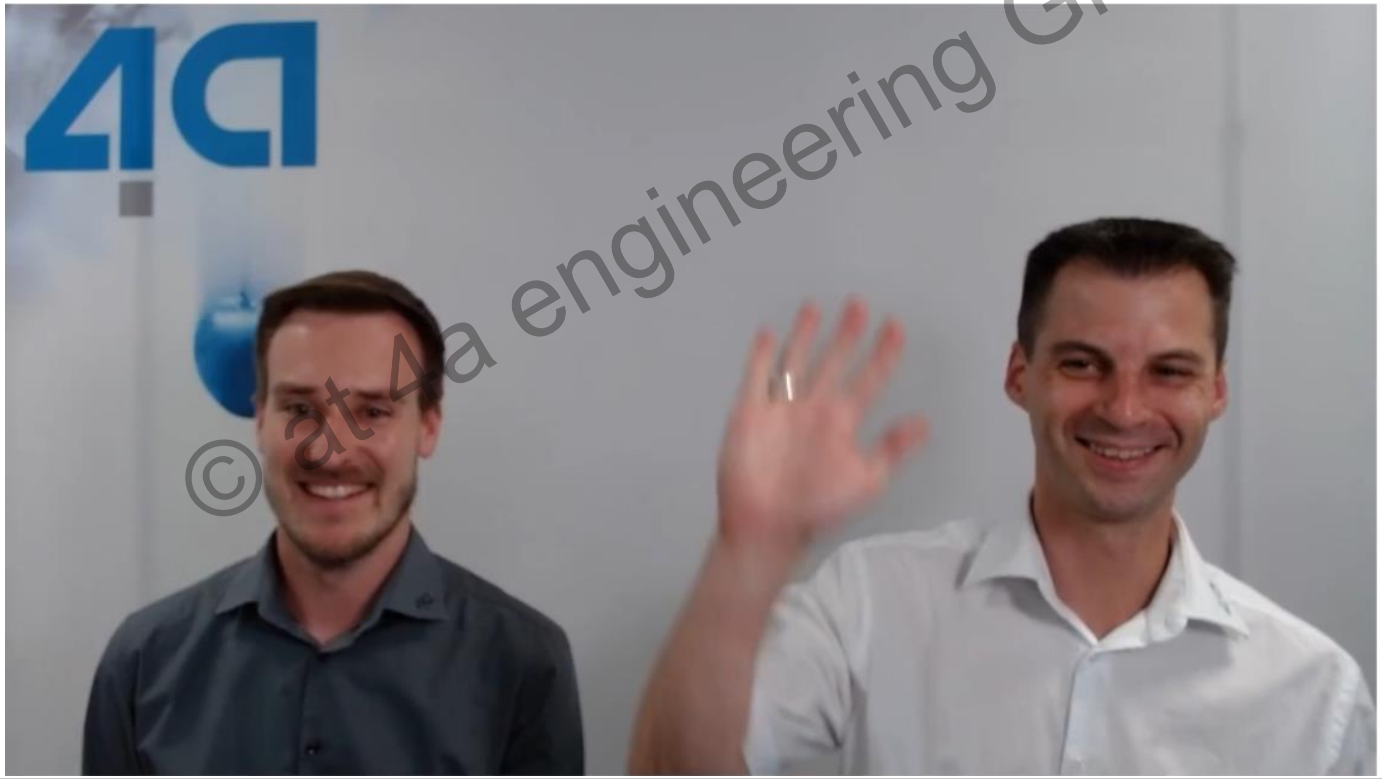
Optimization/ validation/ Pre-simulation





Summer School

Thank you very much for tuning in!



Thank you for your Attention!

Python: a powerful tool with VALIMAT®, user defined material cards/specimen

survey: Please give us your personal feedback

VALIMAT

YouTube

IMPETUS