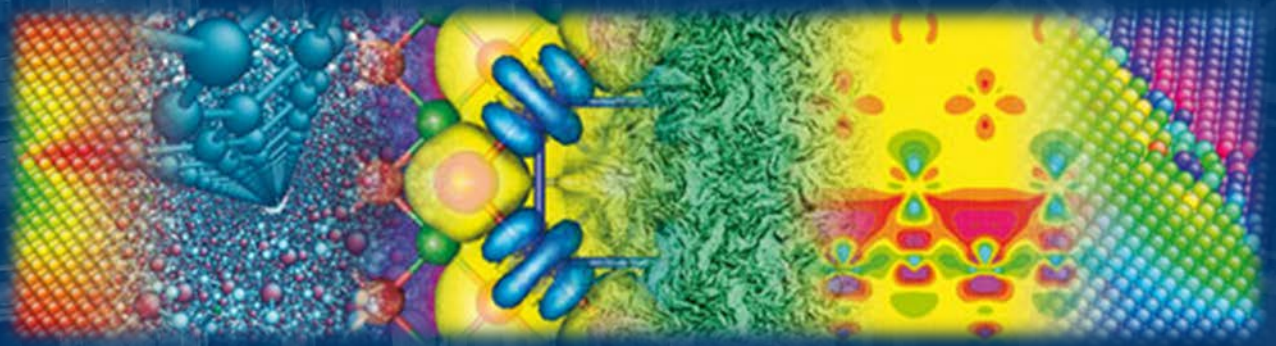
Multiscale Simulation Methods I (MuSim I)

Inhomogeneous Material Properties in Solid Mechancial Systems



Dr. Stefan Sandfeld

Department für Werkstoffwissenschaften

Lehrstuhl für Werkstoffsimulation (WW8)

Dr.-Mack-Str. 77, 90762 Fürth

stefan.sandfeld@fau.de

www.matsim.techfak.uni-erlangen.de

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

I. Some continuum mechanical foundations and notations

II. Averaging elastic material properites with FEM

III. Modelling of shearband formation with FEM

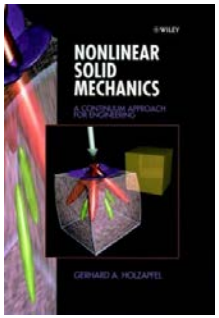
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Literature and Links

- WIKIPEDIA has many very nice sketches (which I shamelessly copied) and good summaries

- SCHAUM's outlines on Continuum Mechanics, McGraw Hill

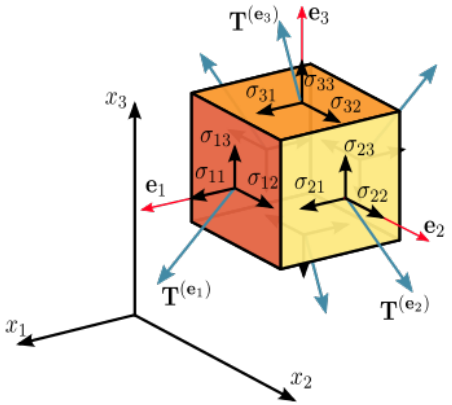


- HOLZAPFEL, Non-linear Solid Mechanics. Wiley
- http://www.colorado.edu/engineering/cas/courses.d/IFEM.d/ excellent webpage on FEM and continuum mechanics
- http://www.oofem.org/en/oofem.html
  The free FEM code which we will be using throuhout this lecture/tutorial

Die "von Mises Vergleichsspannung " – the Mises stress or 'equivalent tensile stress'

- How can we compare two stress states? … the Cauchy stress $\boldsymbol{\sigma}$ is a tensor with 9 components

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \equiv \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \equiv \begin{bmatrix} \sigma_{x} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{y} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{z} \end{bmatrix}$$



- The *Mises stress* or *equivalent tensile stress* $\boldsymbol{\sigma}_v$ is a scalar stress measure, which satisfies the property that two stress states with equal distortion energy have equal Mises stresses

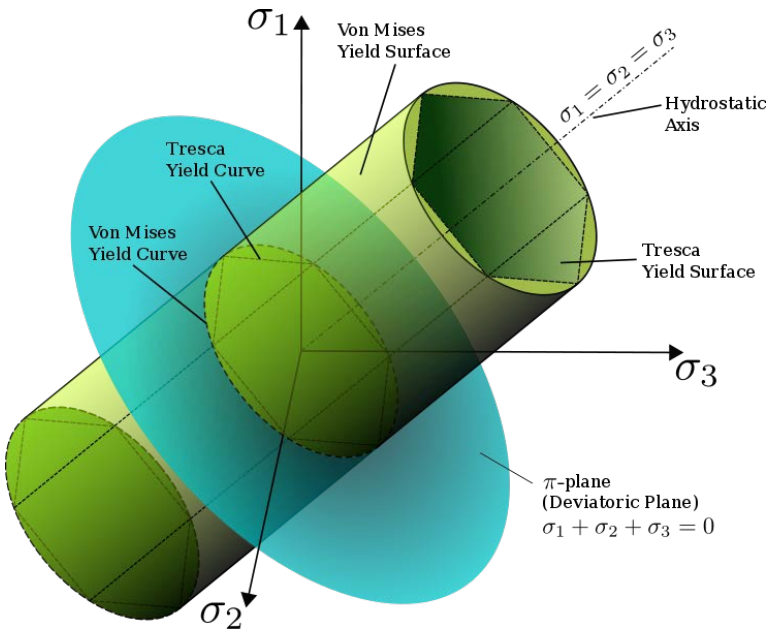
$$\sigma_v^2 = \tfrac{1}{2}[(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{11} - \sigma_{33})^2 + 6(\sigma_{23}^2 + \sigma_{31}^2 + \sigma_{12}^2)]$$

Die "von Mises Vergleichsspannung " – the Mises stress or 'equivalent tensile stress'

- The Mises stress is used for formulating the Mises yield criterion, which is used to predict yielding of materials.
- Mises yield criterion: yielding of materials begins when the second deviatoric stress invariant J_2 reaches a critical value:

$$\sigma_v = \sigma_y = \sqrt{3J_2}$$

- For smaller values, no yielding occurs and the material response is purely elastic
- Stress values larger than the yield stress cannot be reached
- The Mises yield surfaces (in principal stress space) is a cylinder with radius $\sqrt{\frac{2}{3}}\sigma_y$



The work-conjugate counterpart: the equivalent strain

- The equivalent strain (or the von Mises strain) is again a scalar quantity
- In plasticity, we use the definition

$$\varepsilon_{\mathrm{eq}} = \sqrt{\frac{2}{3}\boldsymbol{\varepsilon}^{\mathrm{dev}} : \boldsymbol{\varepsilon}^{\mathrm{dev}}} = \sqrt{\frac{2}{3}\varepsilon_{ij}^{\mathrm{dev}}\varepsilon_{ij}^{\mathrm{dev}}} \; ; \quad \boldsymbol{\varepsilon}^{\mathrm{dev}} = \boldsymbol{\varepsilon} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\varepsilon})\ \mathbf{1}$$

  where the trace of the strain, $\mathrm{tr}(\boldsymbol{\varepsilon})$, is defined as the 1st Invariant, J_1, of the strain tensor:

$$\mathrm{tr}(\boldsymbol{\varepsilon}) = \varepsilon_{ii} = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}$$

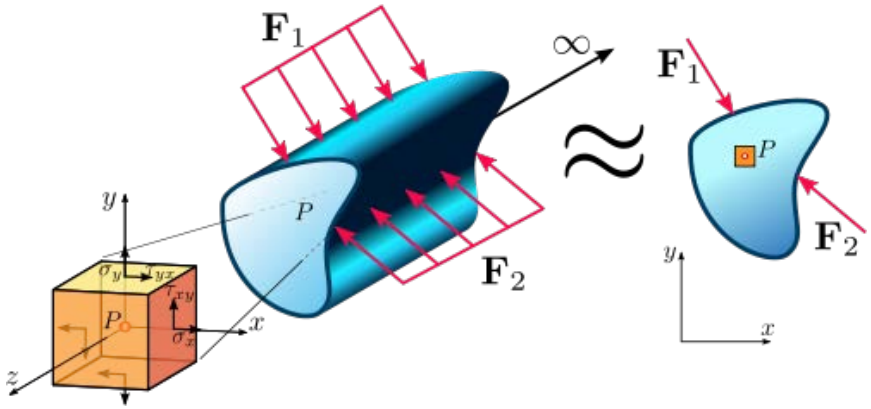
- The deviator of e.g. the stress tensor relates to its 2nd invariant through

$$J_2 = \frac{1}{2}\sigma^{\mathrm{dev}} : \sigma^{\mathrm{dev}}$$

- $\varepsilon_{\mathrm{eq}}$ is work conjugate to the equivalent stress, which also can be written in the same manner

$$\sigma_{\mathrm{eq}} = \sqrt{\frac{3}{2}\boldsymbol{\sigma}^{\mathrm{dev}} : \boldsymbol{\sigma}^{\mathrm{dev}}}$$

Special case: plane strain



- If the length of a structure into one dimension is much larger than the two other directions, one can simplify the system
- strains associated with the larger direction are constrained by nearby material and are small compared to the *cross-sectional strains* → plane strain approximation
- The strain and stress tensors for plane strain can be written as:

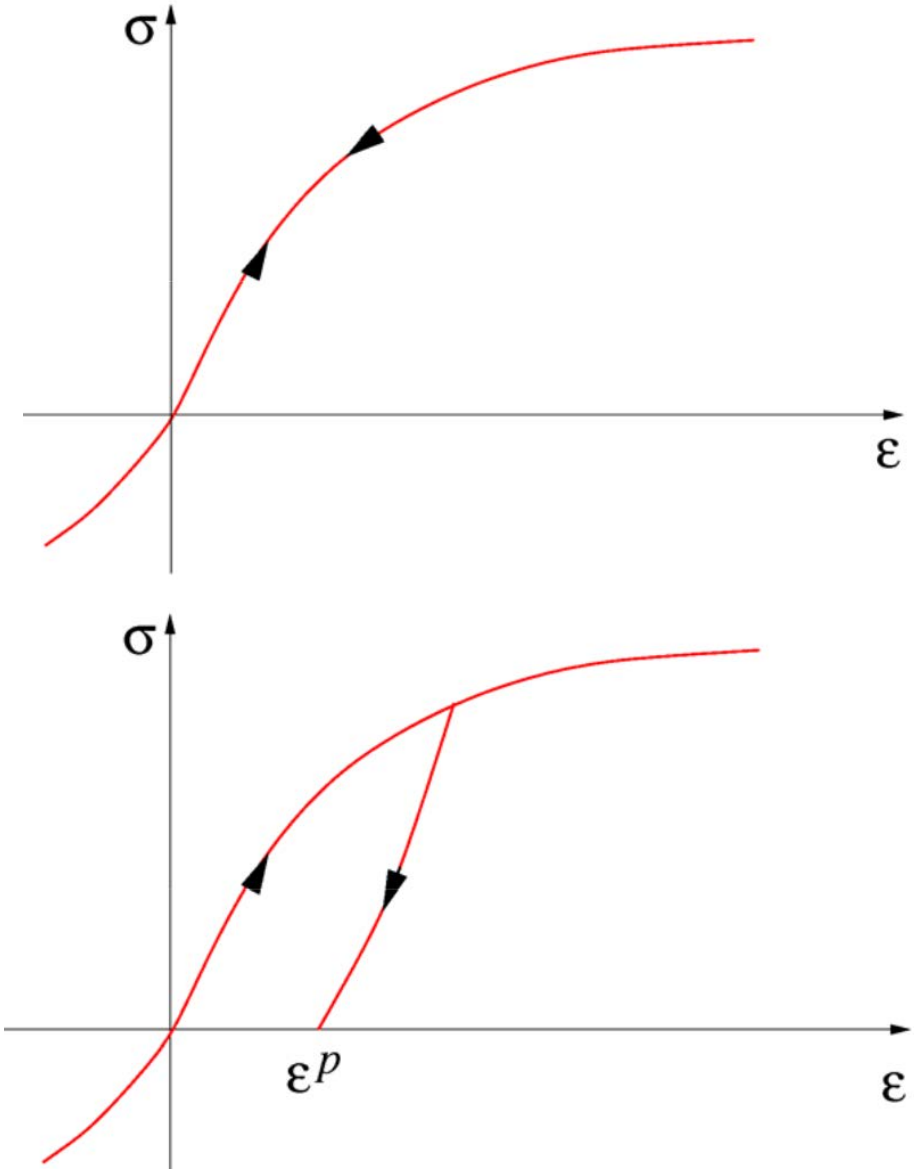
$$\underline{\underline{\boldsymbol{\varepsilon}}} = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & 0 \\ \varepsilon_{21} & \varepsilon_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \underline{\underline{\boldsymbol{\sigma}}} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & 0 \\ \sigma_{21} & \sigma_{22} & 0 \\ 0 & 0 & \sigma_{33} \end{bmatrix}$$

- The non-zero σ_{33} is needed to maintain the constraint $\varepsilon_{33} = 0$.
- The ε_{33}-component can be – during solution – neglected and hence the problem becomes a 2D problem.

Elastic and plastic behavior

- Elastic material response: the path during loading is also followed during unloading → reversible behavior.

- Plasticity: a residual irreversible deformation is typical for the loading – unloading behaviour



Ideal Plasticity

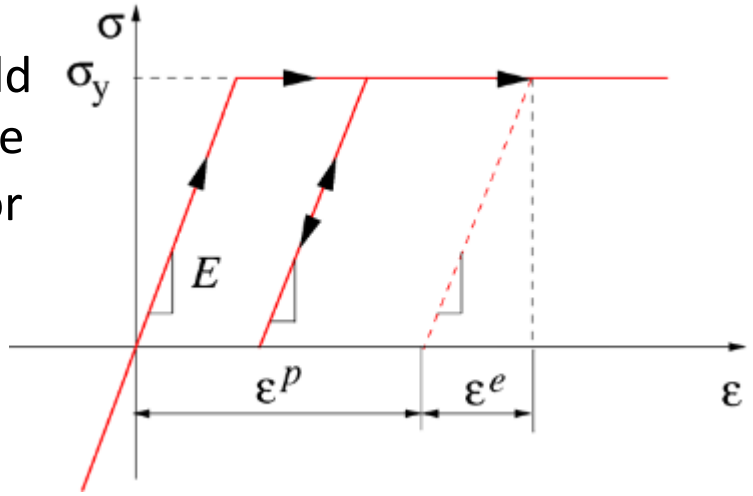
- Plastic deformation: when a stress threshold is reached (yield stress σ_y). Then, irreversible deformations occurs, otherwise the behavior is fully elastic. This is governed by the yield condition:

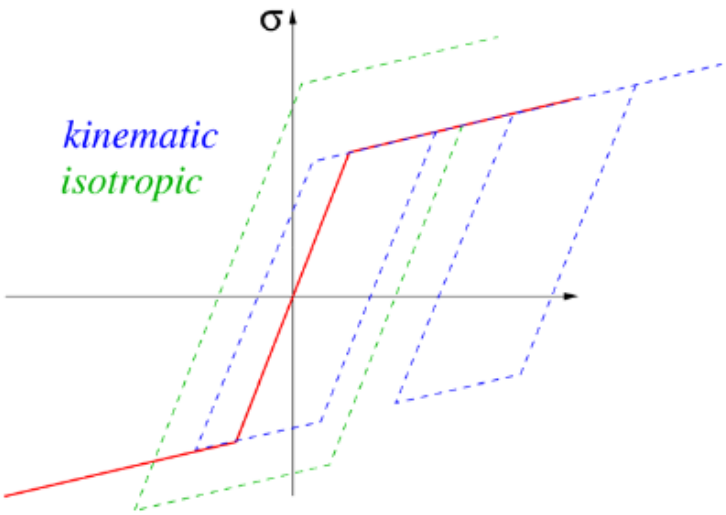
$$f(\sigma, \sigma_y) \leq 0$$

- Total strains are the sum of an elastic part (related to the stress) and a plastic part

$$\varepsilon = \frac{\partial u}{\partial x} = \varepsilon^e + \varepsilon^p$$

- perfect plasticity: no hardening, i.e. stress stays constant for increasing strain
- hardening: threshhold evolves with loading history ➔ additional (internal) variables are required!





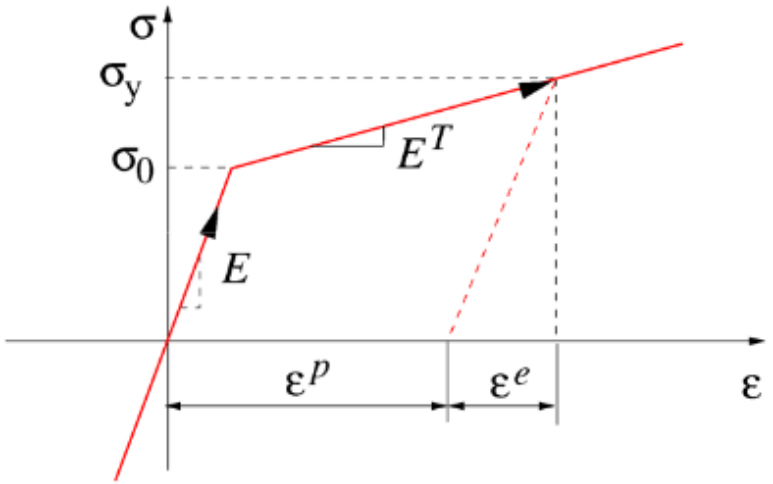
Linear Hardening

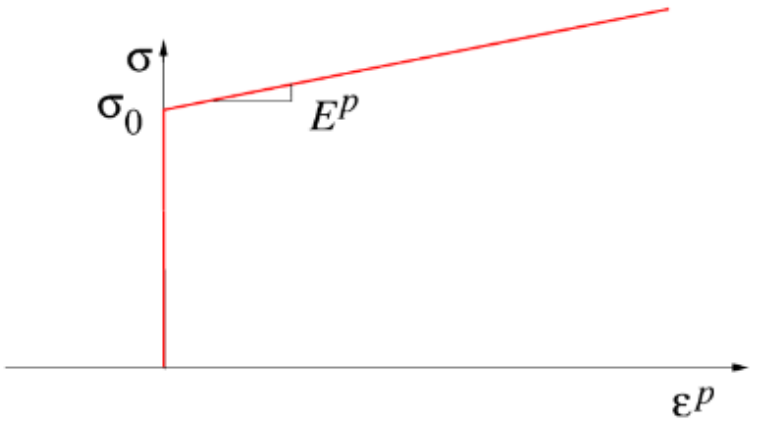
- For a material with isotropic hardening, the effective stress function reduces to a linear relationship between effective plastic strain and effective stress:

$$\Delta\bar{e}^p = \frac{\bar{\sigma}(t) - \sigma_y}{E^p}, \qquad E^p = \frac{E\,E_T}{E - E_T}$$

- By adding the flow-rule, obtain:

$$\bar{\sigma}(t) = \frac{2E^p d + 3\sigma_y}{E^p/\mu + 3}$$





- To described isotropic linear hardening, all we need is 1 additional variable, the so-called hardening modulus

II. Averaging elastic material properites

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

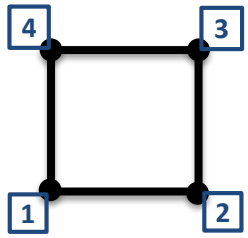
II. Averaging elastic material properites

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

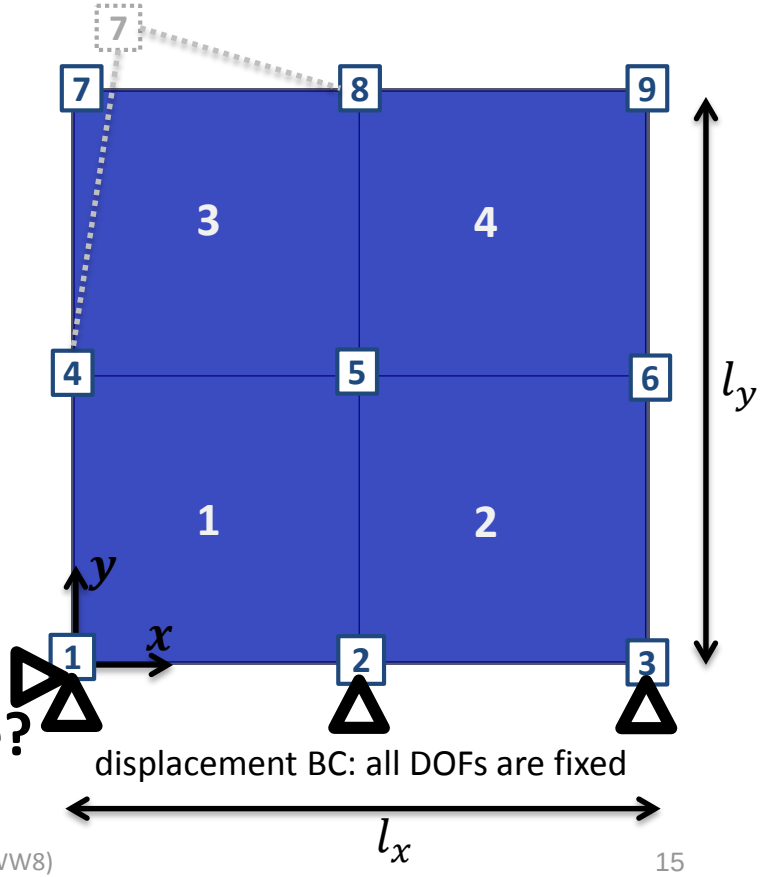
Objective: **What is the behaviour and mechanical response of a structure with random elastic material properties?**

- This part is a hands-on class, where you will write some small PYTHON scripts for creating FEM input files
- Material properties can then automatically be defined in a random fashion
- This is the prerequisite for Part III of this block, where we will investigate the clustering behaviour of inelastic materials with random microstructure

A solid mechanical system with oofem



- 2D (plane strain) geometry, $l_x = l_y = 3.0$
- material parameter: $E = 1.0,\ \nu = 0.3$
- discretization: 2x2 elements
- BCs: lower nodes (nodes 1,2,3) fixed
- node 7 has non-zero displacement u_x and u_y prescribed
- conventions:
  - element node numbering: counter clock wise, starting from bottom left node (cf. top picture)
  - element connectivity bottom left to top right
  - coordinate system located at bootm left node



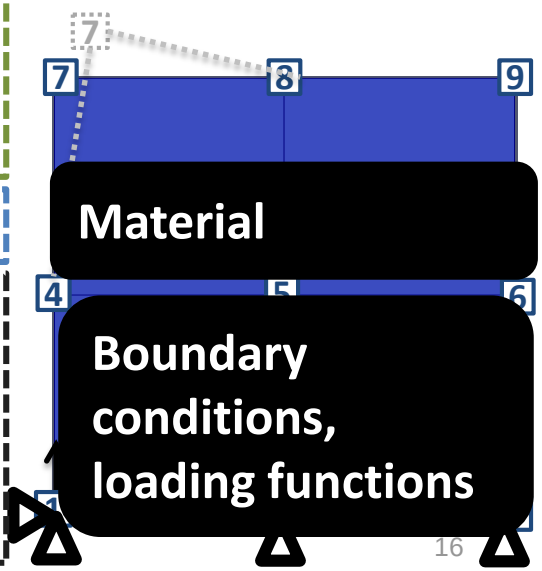
What is the overall mechanical response?

Inputfile structure: (careful – no blank lines, always use # for comments!)

```
test.in.out
der titel geht hierhin
NonLinearStatic nsteps 10 controllmode 1 deltaT 0.1 rtolv 1e-6  MaxIter 1000   nmodules 3
vtkxml tstep_all domain_all primvars 1 1 vars 4 1 2 4 5   stype 1
hom tstep_all
GPExportModule tstep_all domain_all ncoords 3 vars 3 1 4 27
domain PlaneStrain
OutputManager tstep_all dofman_all element_all
#
# ndofman=no. of nodes, nelem=no. elemn.,  ncrosssect=no. crossections
# nmat=no. of material models, nbc=no. BCs, nic=no. ICs,  nltf=no. of time functions
ndofman 9 nelem 4 ncrosssect 1  nmat 2 nbc 6 nic 0 nltf 2
node 1 coords 3              0              0              0   bc 2 1 2
node 2 coords 3            1.5              0              0   bc 2 0 3
node 3 coords 3              3              0              0   bc 2 0 4
node 4 coords 3              0            1.5              0
node 5 coords 3            1.5            1.5              0
node 6 coords 3              3            1.5              0
node 7 coords 3              0              3              0   bc 2 5 6
node 8 coords 3            1.5              3              0
node 9 coords 3              3              3              0
quad1planestrain      1 nodes 4      1      2      5      4 crossSect 1  mat 1
quad1planestrain      2 nodes 4      2      3      6      5 crossSect 1  mat 2
quad1planestrain      3 nodes 4      4      5      8      7 crossSect 1  mat 1
quad1planestrain      4 nodes 4      5      6      9      8 crossSect 1  mat 1
SimpleCS 1 thick 1
#
IsoLE 1 d 0. E 1.0 n 0.3 tAlpha 0
IsoLE 2 d 0. E 1.2 n 0.3 tAlpha 0
#
BoundaryCondition 1 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 2 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 3 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 4 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 5 loadTimeFunction 1 prescribedvalue 0.1
BoundaryCondition 6 loadTimeFunction 1 prescribedvalue 0.05
PiecewiseLinFunction 1 nPoints 2 t 2 0.0 1.0  f(t) 2 0. 1.0
PiecewiseLinFunction 2 nPoints 2 t 2 0.0 1.0  f(t) 2 0.0 1.0
```

Solver, Output, Filenames

node coordinates, element definition, connectivity



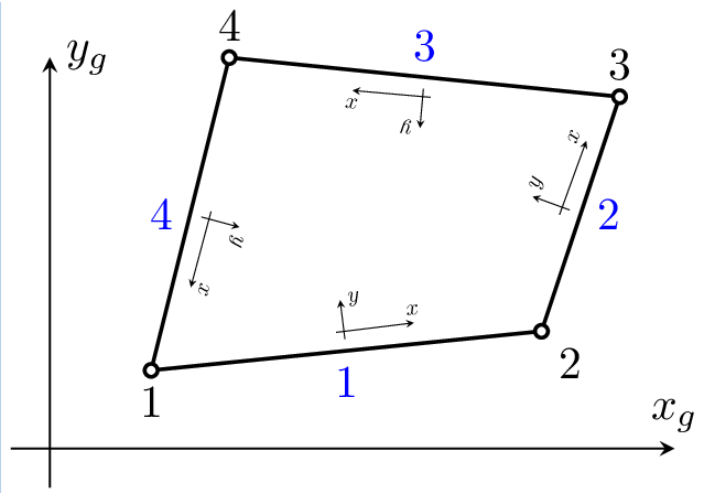
Material

Boundary conditions, loading functions

OOFEM: The Quad1PLaneStrain element

Represents isoparametric four-node quadrilateral plane-strain finite element. Each node has 2 degrees of freedom. Structure should be defined in x,y plane. The nodes should be numbered anti-clockwise (positive rotation around z-axis).

Keyword **Description** **Specific parameters**	**quad1planestrain** **2D linear quadrilateral plane-strain element** **[NIP #(in)]**
Parameters	NIP: allows to set the number of integration points for integration of membrane terms.
Unknowns	Two dofs (u-displacement, v-displacement) are required in each node.
Approxi-mation	Linear approximation of displacements and geometry.
CS properties	Cross section thickness is required.
Loads	Body loads are supported. Boundary loads are supported and computed using numerical integration.



Inputfile structure: (careful – no blank lines, always use # for comments!)

```
test.in.out
der titel geht hierhin
NonLinearStatic nsteps 10 controllmode 1 deltaT 0.1 rtolv 1e-6  MaxIter 1000   nmodules 3
vtkxml tstep_all domain_all primvars 1 1 vars 4 1 2 4 5   stype 1
hom tstep_all
GPExportModule tstep_all domain_all ncoords 3 vars 3 1 4 27
domain PlaneStrain
OutputManager tstep_all dofman_all element_all
#
# ndofman=no. of nodes, nelem=no. elemn.,  ncrosssect=no. crossections
# nmat=no. of material models, nbc=no. BCs, nic=no. ICs,  nltf=no. of time functions
ndofman 9 nelem 4 ncrosssect 1  nmat 2 nbc 6 nic 0 nltf 2
node 1 coords 3              0              0              0   bc 2 1 2
node 2 coords 3            1.5              0              0   bc 2 0 3
node 3 coords 3              3              0              0   bc 2 0 4
node 4 coords 3              0            1.5              0
node 5 coords 3            1.5            1.5              0
node 6 coords 3              3            1.5              0
node 7 coords 3              0              3              0   bc 2 5 6
node 8 coords 3            1.5              3              0
node 9 coords 3              3              3              0
quad1planestrain      1 nodes 4      1      2      5      4 crossSect 1  mat 1
quad1planestrain      2 nodes 4      2      3      6      5 crossSect 1  mat 2
quad1planestrain      3 nodes 4      4      5      8      7 crossSect 1  mat 1
quad1planestrain      4 nodes 4      5      6      9      8 crossSect 1  mat 1
SimpleCS 1 thick 1
#
IsoLE 1 d 0. E 1.0 n 0.3 tAlpha 0
IsoLE 2 d 0. E 1.2 n 0.3 tAlpha 0
#
BoundaryCondition 1 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 2 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 3 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 4 loadTimeFunction 1 prescribedvalue 0
BoundaryCondition 5 loadTimeFunction 1 prescribedvalue 0.1
BoundaryCondition 6 loadTimeFunction 1 prescribedvalue 0.05
PiecewiseLinFunction 1 nPoints 2 t 2 0.0 1.0  f(t) 2 0. 1.0
PiecewiseLinFunction 2 nPoints 2 t 2 0.0 1.0  f(t) 2 0.0 1.0
```

Material

OOFEM Material Models (http://www.oofem.org/resources/doc/matlibmanual/html)

Isotropic linear elastic material - IsoLE

Description	Linear isotropic elastic material
Record Format	**IsoLE** num(in) # d(rn) # E(rn) # n(rn) # tAlpha(rn) #
Parameters	- *num* material model number
	- *d* material density
	- *E* Young modulus
	- *n* Poisson ratio
	- *tAlpha* thermal dilatation coefficient
Supported modes	3dMat, PlaneStress, PlaneStrain, 1dMat, 2dPlateLayer, 2dBeamLayer, 3dShellLayer, 2dPlate, 2dBeam, 3dShell, 3dBeam, PlaneStressRot
Features	Adaptivity support

Example: IsoLE 1 d 1.0 E 1. n 0.2 tAlpha 0.000012

first: ". /usr/net/bin/module_oofem21"

Task 1:

1.1) Download the input file **task1.in** from StudOn into a directory **task1**

1.2) run the FEM simulation by typing

>oofem –f task.in

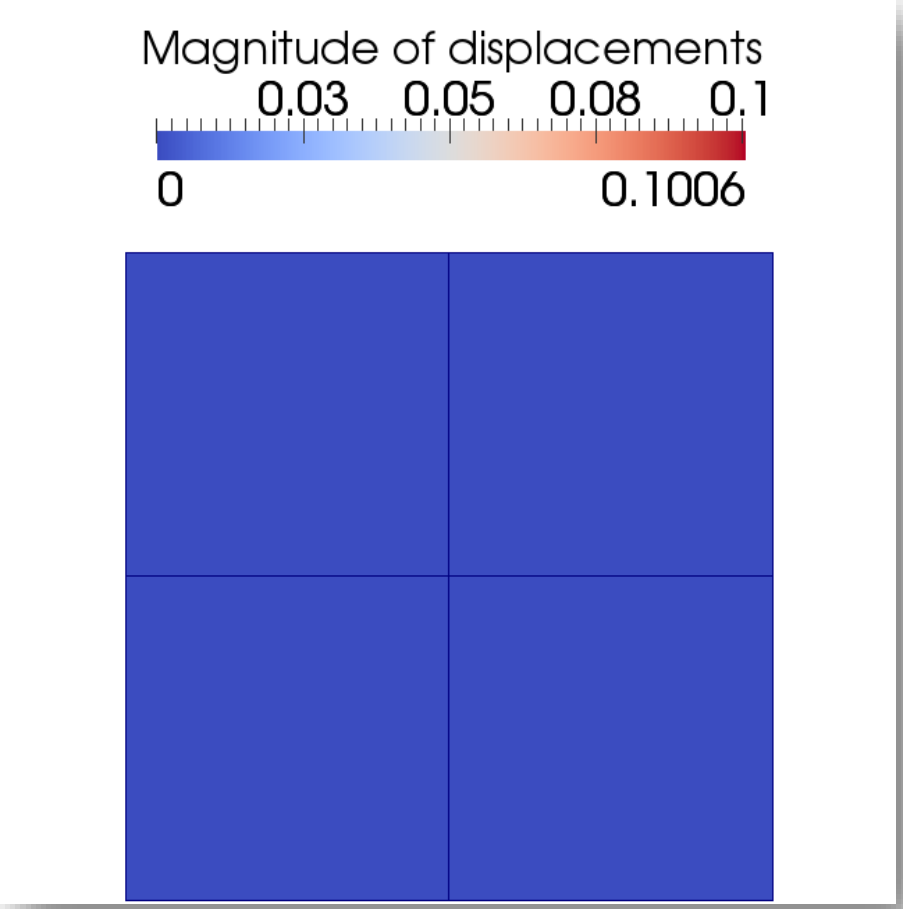
This produces the files

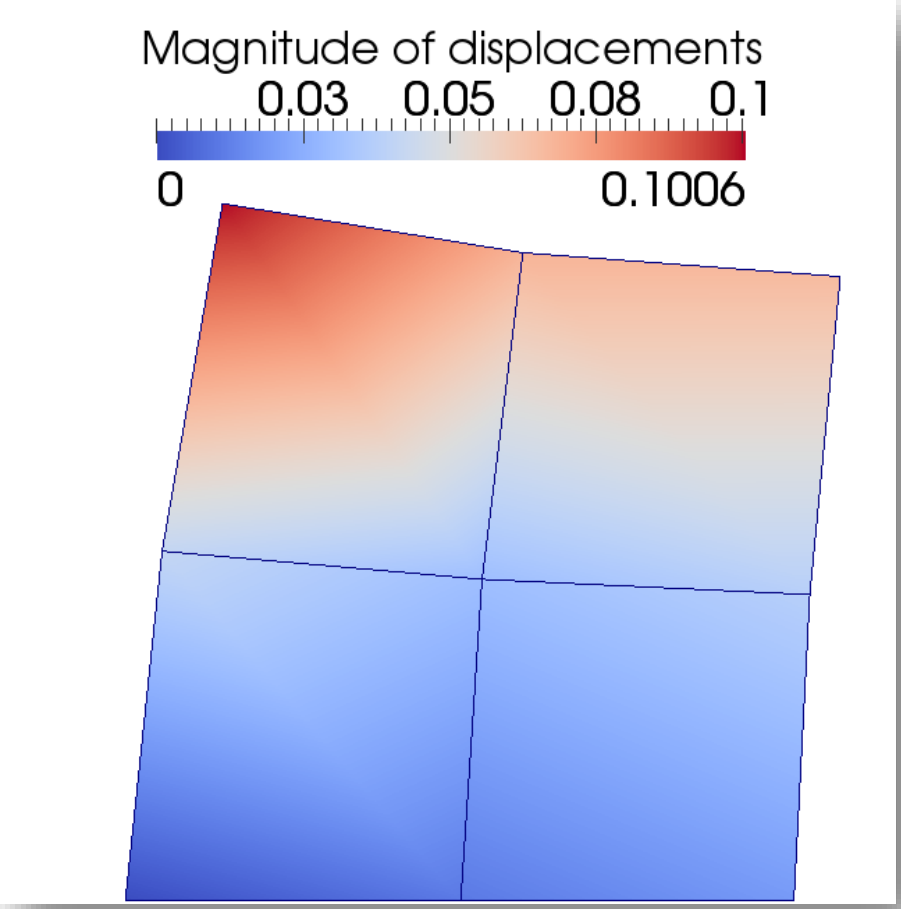
- **task1.in.out**
  (stresses and strains at GPs)
- **task1.in.out.hom**
  (averaged stresses and strains)
- **task1.in.out.m0.1.vtu**
- **task1.in.out.m2.1.gp**
  (stresses and strains at GPs in compact notation)

1.3) view the .vtu file with paraview

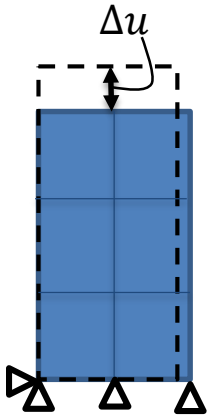
```
ssandfeld@ww8:~/MuSim2/StochasticFEM/task1$ oofem -f task1.in
_____________________________________________________________
           OOFEM - Finite Element Solver
        Copyright (C) 1994-2010 Borek Patzak
_____________________________________________________________
Total number of solution steps 1
Instanciating domain                 1
Instanciated nodes & sides           9
Instanciated elements                4
Instanciated cross sections          1
Instanciated materials               2
Instanciated BCs                     6
Instanciated ICs                     0
Instanciated load-time fncts         2
Consistency check                   ok
Assembling load
Assembling tangent stiffness matrix
NonLinearStatic info: user time consumed by assembly: 0.00s
Solving [step number     1.0]
Time       Iteration       ForceError      DisplError
_____________________________________________________________
Skyline info: neq is 12, nwk is 67
Skyline info: user time consumed by factorization: 0.00s
0          1               7.950242e-16    1.000000e+00
0          2               3.549430e-16    2.916910e-16

  Quasi reaction table:
node  dof   displacement         force
=======================================
Equilibrium reached at load level = 1.000000 in 2 iterations
Updating domain                      1
Updated nodes & sides                9
Updated Elements                     4
Updated Materials                    4
NonLinearStatic: fixed load level
Reseting load level
EngngModel info: user time consumed by solution step 1: 0.00s
_____________________________________________________________
ANALYSIS FINISHED (real time consumed: 000h:00m:00s)
                  (user time consumed: 000h:00m:00s)

_____________________________________________________________
Total 0 error(s) and 0 warning(s) reported
```

Total deformation (w/o and w displacement of node 7)

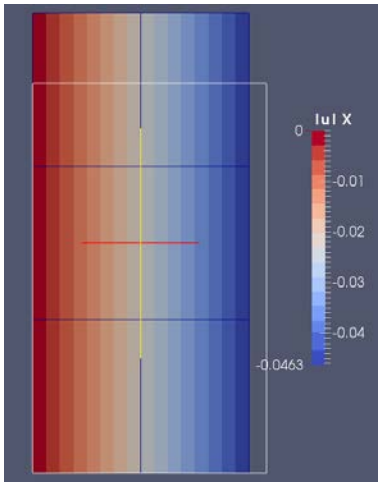


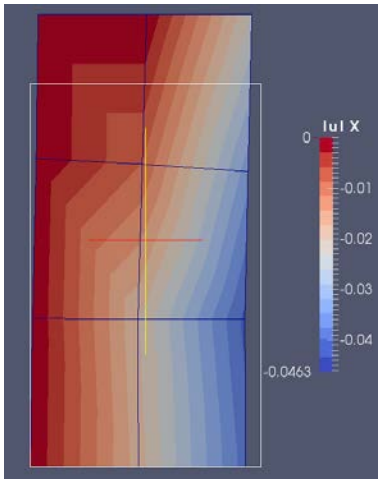


Task 2



- write a PYTHON script **`task2.py`** that creates an input file **`task2.in`** for a tensile test with
  - 2x3 elements, $l_x = 3, l_y = 5$
  - only displacement BCs, where the Δu changes from 0..0.2 in 10 steps
  - each element should have its own linear elastic material law IsoLE 1 .. 10 (initially set all $E = 1.0$)
- use the provided StudOn template to start your python script
- change the material properties slightly, e.g. set E=0.5…1.0 randomly for each element and view the displacement, stresses and strains with paraview





Careful – OOFEM numbering is 1-based!

```
task2.in.out
Title
NonLinearStatic nsteps 10 controllmode 1 deltaT 0.1 rtolv 1e-6  MaxIter 1000   nmodules 3
vtkxml tstep_all domain_all primvars 1 1 vars 4 1 2 4 5   stype 1
hom tstep_all
GPExportModule tstep_all domain_all ncoords 3 vars 3 1 4 27
domain PlaneStrain
OutputManager tstep_all dofman_all element_all
ndofman 12 nelem 6 ncrosssect 1 nmat 6 nbc 2 nic 0 nltf 1
node 1 coords 3 0.0 0.0 0 bc 2 1 1
node 2 coords 3 1.5 0.0 0 bc 2 0 1
node 3 coords 3 3.0 0.0 0 bc 2 0 1
node 4 coords 3 0.0 1.66666666667 0
node 5 coords 3 1.5 1.66666666667 0
node 6 coords 3 3.0 1.66666666667 0
node 7 coords 3 0.0 3.33333333333 0
node 8 coords 3 1.5 3.33333333333 0
node 9 coords 3 3.0 3.33333333333 0 bc 2 0 2
node 10 coords 3 0.0 5.0 0 bc 2 0 2
node 11 coords 3 1.5 5.0 0 bc 2 0 2
node 12 coords 3 3.0 5.0 0 bc 2 0 2
quad1planestrain 1 nodes 4 1 2 5 4 crossSect 1 mat 1
quad1planestrain 2 nodes 4 2 3 6 5 crossSect 1 mat 2
quad1planestrain 3 nodes 4 4 5 8 7 crossSect 1 mat 3
quad1planestrain 4 nodes 4 5 6 9 8 crossSect 1 mat 4
quad1planestrain 5 nodes 4 7 8 11 10 crossSect 1 mat 5
quad1planestrain 6 nodes 4 8 9 12 11 crossSect 1 mat 6
SimpleCS 1 thick 1
IsoLE 1 d 0. E 0.973320858783 n 0.3 tAlpha 0
IsoLE 2 d 0. E 1.19445995486 n 0.3 tAlpha 0
IsoLE 3 d 0. E 0.712106575439 n 0.3 tAlpha 0
IsoLE 4 d 0. E 0.7025746199 n 0.3 tAlpha 0
IsoLE 5 d 0. E 0.99073485378 n 0.3 tAlpha 0
IsoLE 6 d 0. E 0.55510456231 n 0.3 tAlpha 0
BoundaryConditions 1 loadTimeFunction 1 prescribedvalue 0
BoundaryConditions 2 loadTimeFunction 1 prescribedvalue 1
PiecewiseLinFunction 1 nPoints 2 t 2 0.0 1.0 f(t) 2 0.0 1.0
```

Task 2: … the template --> StudOn

```
24
25  import numpy as np
26  import numpy.random as rnd
27
28
29  def main():
30      
31      # size of the system and number of nodes
32      lx,ly = 3., 5.
33      nodes_x,nodes_y = 3,4
34      
35      dofs = nodes_x*nodes_y
36      Nele=(nodes_x-1)*(nodes_y-1)
37      
38      
39      oofem_input = list()
40      oofem_input.append('task2.in.out')
41      oofem_input.append('Title')
42      oofem_input.append('NonLinearStatic nsteps 10 controllmode 1 deltaT 0.1 rtolv 1e-6  MaxIter 1000  nmodules 3')
43      oofem_input.append('vtkxml tstep_all domain_all primvars 1 1 vars 4 1 2 4 5  stype 1')
44      oofem_input.append('hom tstep_all')
45      oofem_input.append('GPExportModule tstep_all domain_all ncoords 3 vars 3 1 4 27')
46      oofem_input.append('domain PlaneStrain')
47      oofem_input.append('OutputManager tstep_all dofman_all element_all')
48      oofem_input.append('ndofman {} nelem {} ncrosssect 1 nmat {} nbc 2 nic 0 nltf 1'.format(dofs,Nele,Nele))
49
50      node_id = 0
51      
52      # ------------------------------------------------------
53      # ------- TODO: define nodes with 2 for-loops ----------
54      # ------------------------------------------------------
55      # hint: if-elif-... takes care of BCs at bottom and top
56      # number of nodes can be counted by incrementing node_id
57      # the 2 loops should go over all node numbers in x and y direction
58      # node numbers start at 1 and not at 0 !
59      # ------------------------------------------------------
60      
```

```
61
62  # ----------------------------------------------------
63  # ------- TODO: define elements with 2 for-loops -------
64  # ----------------------------------------------------
65  # define: node numbers of the element (lowerleft, lower right, upper left, upper right)
66  # each element should get its own material id
67  # element numbers start at 1 and not at 0 !
68  # ----------------------------------------------------
69
70
71  # ----------------------------------------------------
72  # ------- define CS ----------------------------------
73  # ----------------------------------------------------
74  oofem_input.append('SimpleCS 1 thick 1')
75  # ----------------------------------------------------
76
77
78  # ----------------------------------------------------
79  # ------- TODO: define materials
80  # ----------------------------------------------------
81  # use loop over all elements
82  # Young's modulus should be random
83  # ----------------------------------------------------
84
85
86  # ----------------------------------------------------
87  # ------- define 2 BCs -------------------------------
88  # ----------------------------------------------------
89  oofem_input.append('BoundaryConditions 1 loadTimeFunction 1 prescribedvalue 0')
90  oofem_input.append('BoundaryConditions 2 loadTimeFunction 1 prescribedvalue 1')
91  # ----------------------------------------------------
92
```

```
93
94      # ------------------------------------------------------
95      # ------- define time dependent function ---------------
96      # ------------------------------------------------------
97      oofem_input.append('PiecewiseLinFunction 1 nPoints 2 t 2 0.0 1.0 f(t) 2 0.0 1.0')
98      # ------------------------------------------------------
99
100
101
102     fname = 'task2.in'
103     with open(fname, 'w') as f:
104         f.write('\n'.join(oofem_input))
105
106     return 0
107
```

Task 3: Analyze the stress-strain response

Use **`task2.out.hom`** to extract stresses and strains for each time step

- Option 1: write a MATLAB or PYTHON script to read the average data from file and plot the von Mises stress/equivalent strain using the formula from the previous slides
- Option 2: =quick alternative: use GNUPLOT ('gnuplot strestra.gp')– don't waste time on details ;-)

[get this file on StudOn!]

```
set xlabel 'eps_v'
set ylabel 'sig_v'
sv(sxx,syy,szz,sxy,syz,sxz)=sqrt(  0.5*( (sxx-syy)**2 + (syy-szz)**2 +
                          (sxx-szz)**2) + 6.*(syz**2+sxz**2+sxy**2)  )

plot 'task3.1.out.hom' \
         using (sv($2,$3,$4,$5,$6,$7)):(sv($8,$9,$10,$11,$12,$13)) \
         with line     title'elastic'
pause -1

p 'task3.1.out.hom' u (sv($2,$3,$4,$5,$6,$7))*(sv($8,$9,$10,$11,$12,$13)) \
         w l t'total (av.) strain energy density'
pause -1
```

- Why are the two lines straight?
- What determines the inclination? Write down the equation for the resulting homogenized relationship between σ_v and ε_v.
- Under which circumstances might the stress-strain curve with inhomogeneous material properties no longer stay straight?

solution…

HOMEWORK: extending your FEM simulation framework

- So far you have writtne a Python script that creates a text-inputfile for oofem. Executing this (oofem – f task2.in) produces result files
- The task is now, to read one of the result files (e.g. taskX.out) and to store the stress and strain data into a PYTHON numpy array so that we can postprocess the data subsequently
- Point of departure is the file task4.py in the directory 'task4'. The tasks are also described in details there (search for the TODO sections)
- HINT: you should write a function in the file oofem_utils.py. This will be then imported by the line import oofem_utils as oofem
  A function of this file can be used by 'oofem.my_func(...)'

Overview over the HOMEWORK tasks:

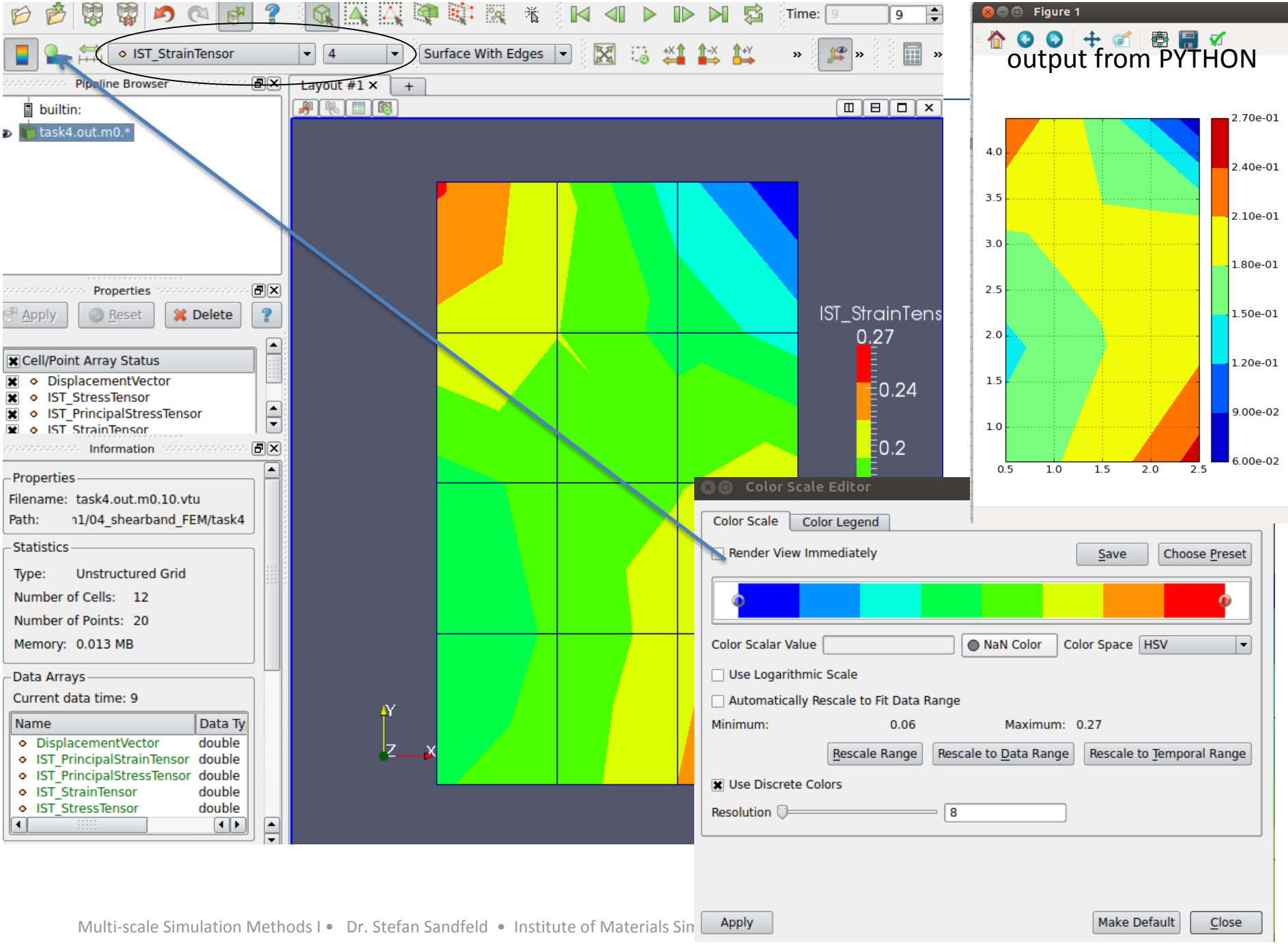
1. convert your script that generates the inputfile into a function and store it in the file oofem_utils.py. Add the function parameter and use them in the function
2. Test the function and make yourself familiar with the data structure 'sim' that is returned from oofem.readOutputfile. Manuel Leimberger (who wrote this – thanks!) also added documentation in the file
3. Extract strain data avstrain.yy (=ε_{yy}) from sim and store it into a numpy array.
4. plot the data from avstrain.yy and check that everything is fine by comparing it with the paraview output (set the data type to IST_StrinaTensor, 4, Surface with edges) (Hint: 0:epsxx, 1:epsxy, 2:epsxz, 3:epsyx, 4:epsyy,…)
5. Add another function for writing oofem inputfile write_J2plasti_inputfile(lx,ly,nodes_x,nodes_y,dv, infile, outfile). This should use a plasticity material model (see next slide). All you have to do is to change material model from
   IsoLE to **J2mat 2 d 1. Ry 0.3 E 1.0 n 0.33 IHM 0.04 tAlpha 0.0**

OOFEM Material Models (http://www.oofem.org/resources/doc/matlibmanual/html)

Mises plasticity model with isotropic and kinematic hardening

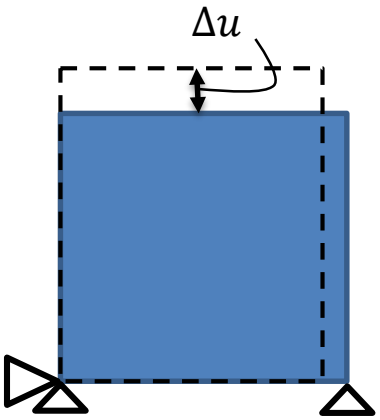
Description	Linear isotropic elastic material
Record Format	**J2mat** num(in) # d(rn) # Ry # E(rn) # n(rn) # IHM # tAlpha(rn) #
Parameters	- *num* material model number - *d* material density - Ry yield stress - E Youngs modulus - N Poisson ratio - IHM isotropic hardening modulus - KHM kinematic hardening modulus - *tAlpha* thermal dilatation coefficient
Supported modes	3dMat, PlaneStress, PlaneStrain, 1dMat, 2dPlateLayer, 2dBeamLayer, 3dShellLayer, 2dPlate, 2dBeam, 3dShell, 3dBeam, PlaneStressRot
Features	Adaptivity support

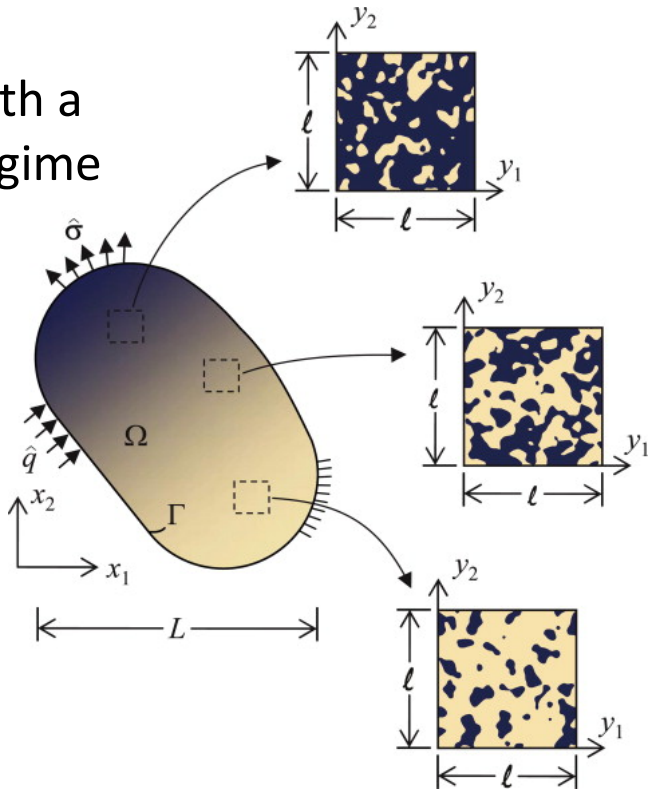
Example: J2mat 2 d 1. Ry 1.7 E 1.0 n 0.33 IHM 0.4 tAlpha 0.0


Time:
IST_StrainTensor
Surface With Edges
Pipeline Browser
builtin:
task4.out.m0.*
Layout #1
Properties
Apply
Reset
Delete
Cell/Point Array Status
DisplacementVector
IST_StressTensor
IST_PrincipalStressTensor
IST_StrainTensor
Information
Properties
Filename: task4.out.m0.10.vtu
Path: n1/04_shearband_FEM/task4
Statistics
Type: Unstructured Grid
Number of Cells: 12
Number of Points: 20
Memory: 0.013 MB
Data Arrays
Current data time: 9
Name
Data Ty
DisplacementVector double
IST_PrincipalStrainTensor double
IST_PrincipalStressTensor double
IST_StrainTensor double
IST_StressTensor double
IST_StrainTens
0.27
0.24
0.2
Y
Z
X
Color Scale Editor
Color Scale
Color Legend
Render View Immediately
Save
Choose Preset
Color Scalar Value
NaN Color
Color Space
HSV
Use Logarithmic Scale
Automatically Rescale to Fit Data Range
Minimum: 0.06
Maximum: 0.27
Rescale Range
Rescale to Data Range
Rescale to Temporal Range
Use Discrete Colors
Resolution
8
Apply
Make Default
Close
Figure 1
output from PYTHON
2.70e-01
2.40e-01
2.10e-01
1.80e-01
1.50e-01
1.20e-01
9.00e-02
6.00e-02
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1.0
1.5
2.0
2.5

Part II: Study of the 'clustering behavior' of a non-linear material with randomly distributed properties

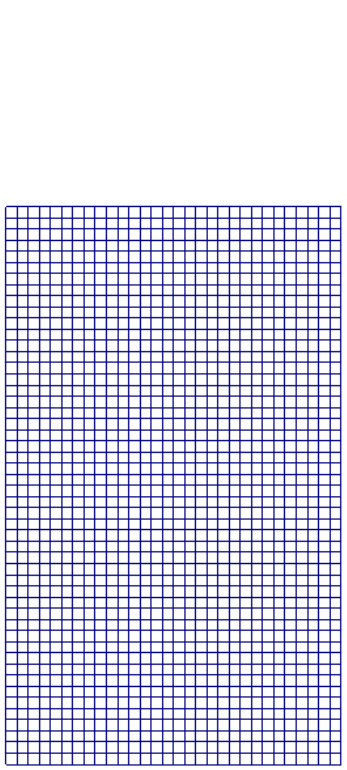
- Assume the same tensile test with a non-linear constitutive material law, e.g. a J2 plasticity law with a linear elastic regime, followed by a ideal-plastic regime
- Task 1: plot the stress-strain curve for 1 element
  Material parameter: E=1, nu=0.3, τ_y =0.1, hardening modulus $H = 0$. and compare it with the linear elastic response.



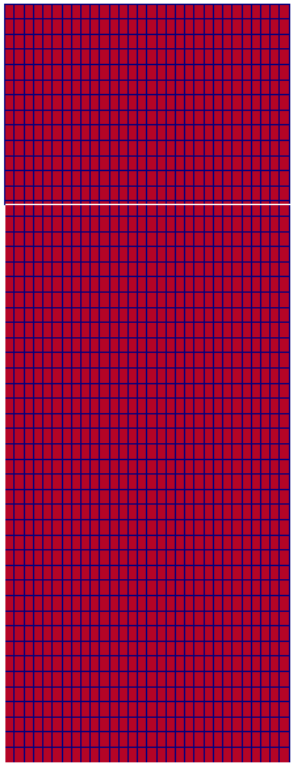


S. Vel and A. Goupee,
Comp. Mater. Sci. 48, 2010

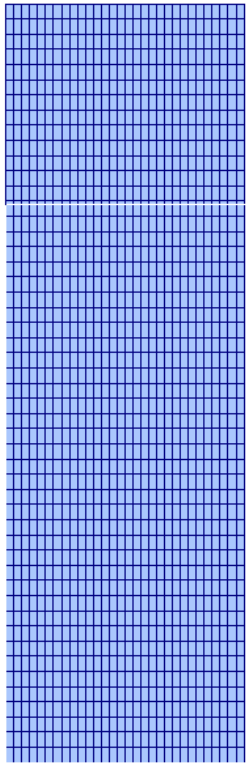
Random Material Properties: 2 different flow stresses



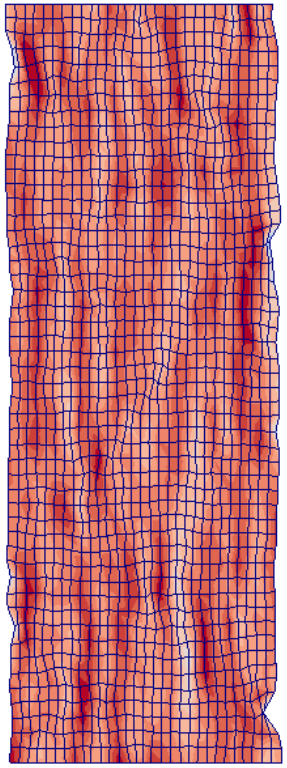
undeformed state



purely elastic



purely plastic



50% elastic, 50% plastic elements

The Material Model

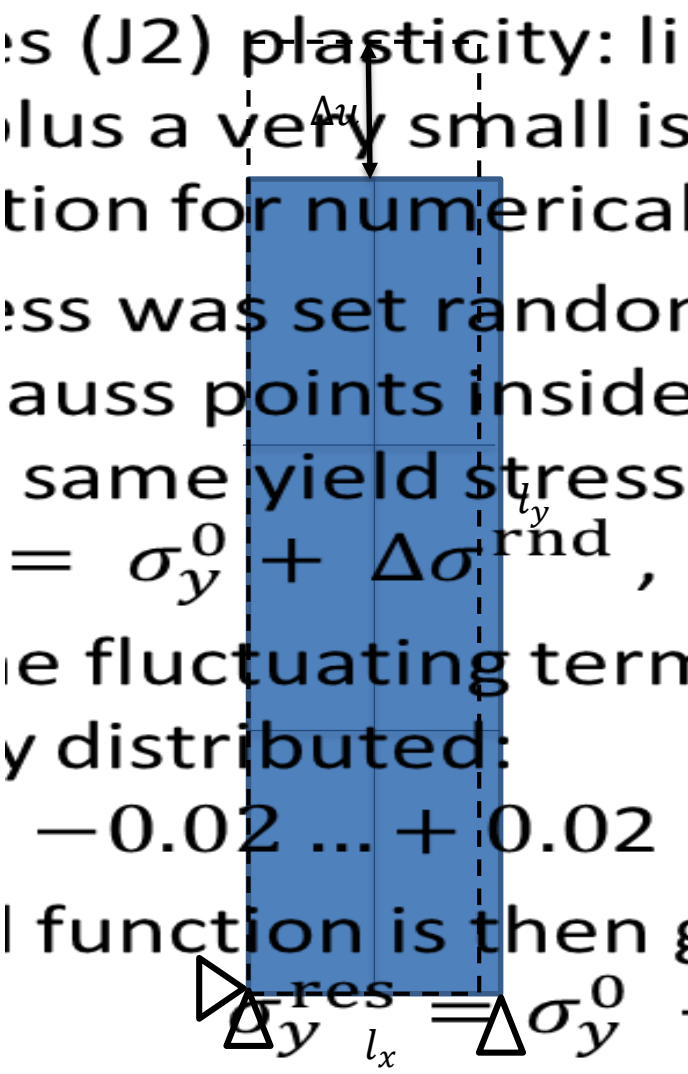
- von Mises (J2) plasticity: linear elastic, ideally plastic (plus a very small isotropic hardening contribution for numerical stability reasons)
- yield stress was set randomly for *each element* (i.e. all Gauss points inside an finite element have the same yield stress)

$$\sigma_y = \sigma_y^0 + \Delta\sigma^{\mathrm{rnd}}, \quad \text{where} \quad \sigma_y = 0.2$$

- where the fluctuating term $\Delta\sigma^{\mathrm{rnd}}$ is uniform randomly distributed:
  $\Delta\sigma^{\mathrm{rnd}} = -0.02 \ldots + 0.02$
- The yield function is then given by

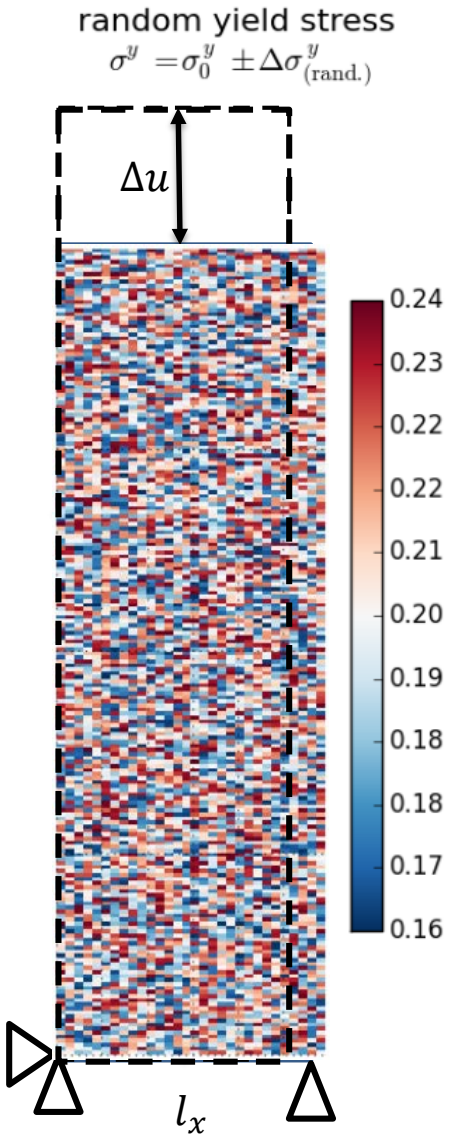
$$\sigma_y^{\mathrm{res}} = \sigma_y^0 + 0.01\varepsilon_{\mathrm{eq}}, \text{ where } \ldots$$

- $\varepsilon_{\mathrm{eq}}$ is the equivalent plastic strain (a scalar quantity)



Numerical aspects

- solution of this BVP by the finite element method
- approximation of displacements by quadratic shape functions
- number of elements: $N_x = 30,\ N_y = 200$
- → element size $h_x = h_y = 0.1$
- same number/size of elements for the yield stress as for approximating the geometrie
- we assume small strains and small rotations
  - This assumption is somewhat pushed to its limits in the following examples...
  - … but acceptable approximation, since all important features (surface roughening due to shear bands etc) already occur at small strains and are merely amplified by increasing the total strain)



Output from a typical FEM simulation: the displacement field

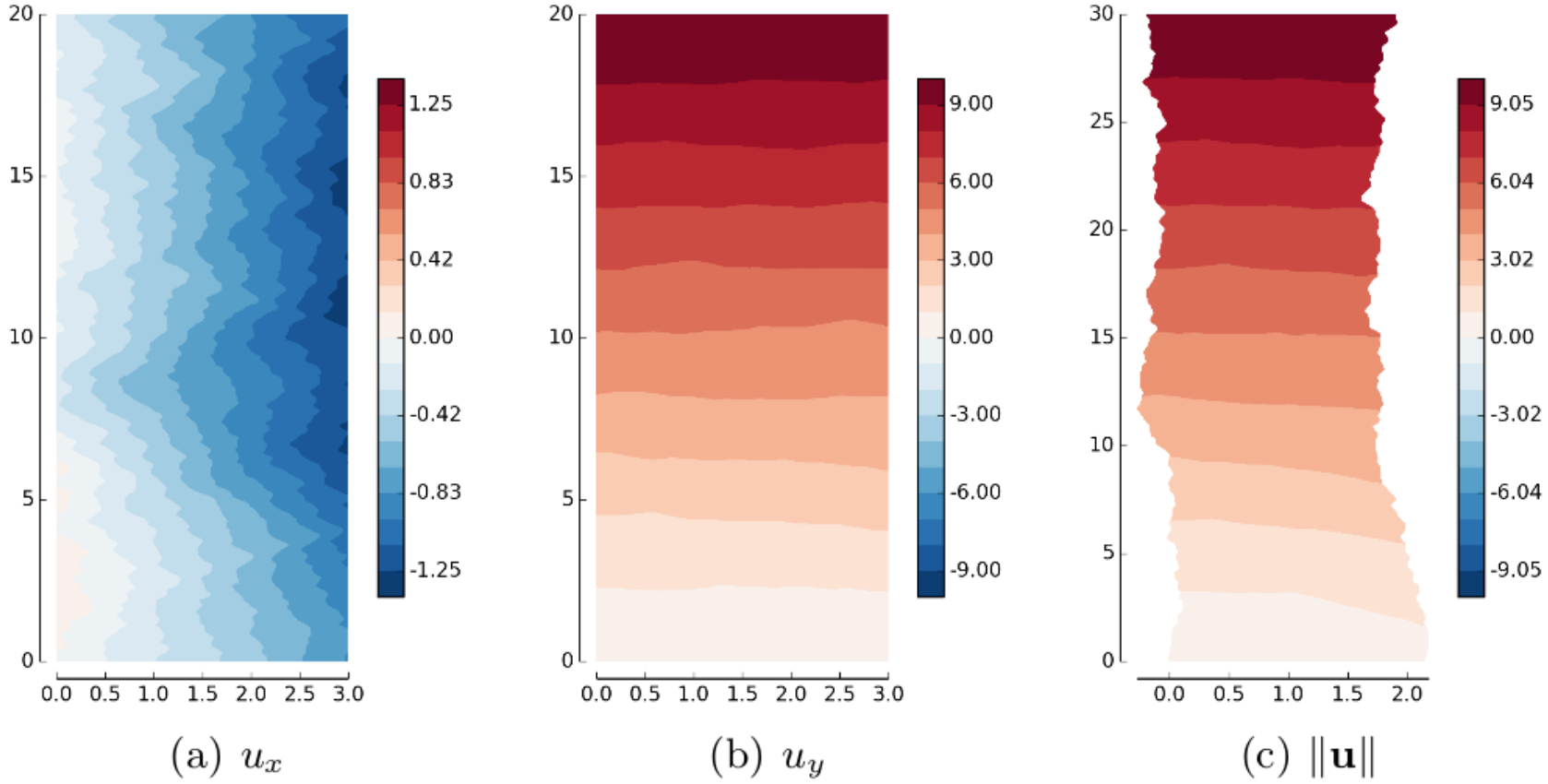


Fig. 2 Local field quantities obtained from a FEM simulation at maximum tensile strain $\varepsilon_{yy} = 0.5$: (a) horizontal displacements, (b) vertical displacements, (c) norm of the displacements plotted in deformed geometry (note the different vertical length scale as compared to plots (a) and (b)).

Output from a typical FEM simulation: strain field $\boldsymbol{\varepsilon} = \mathrm{sym}(\nabla \mathbf{u})$

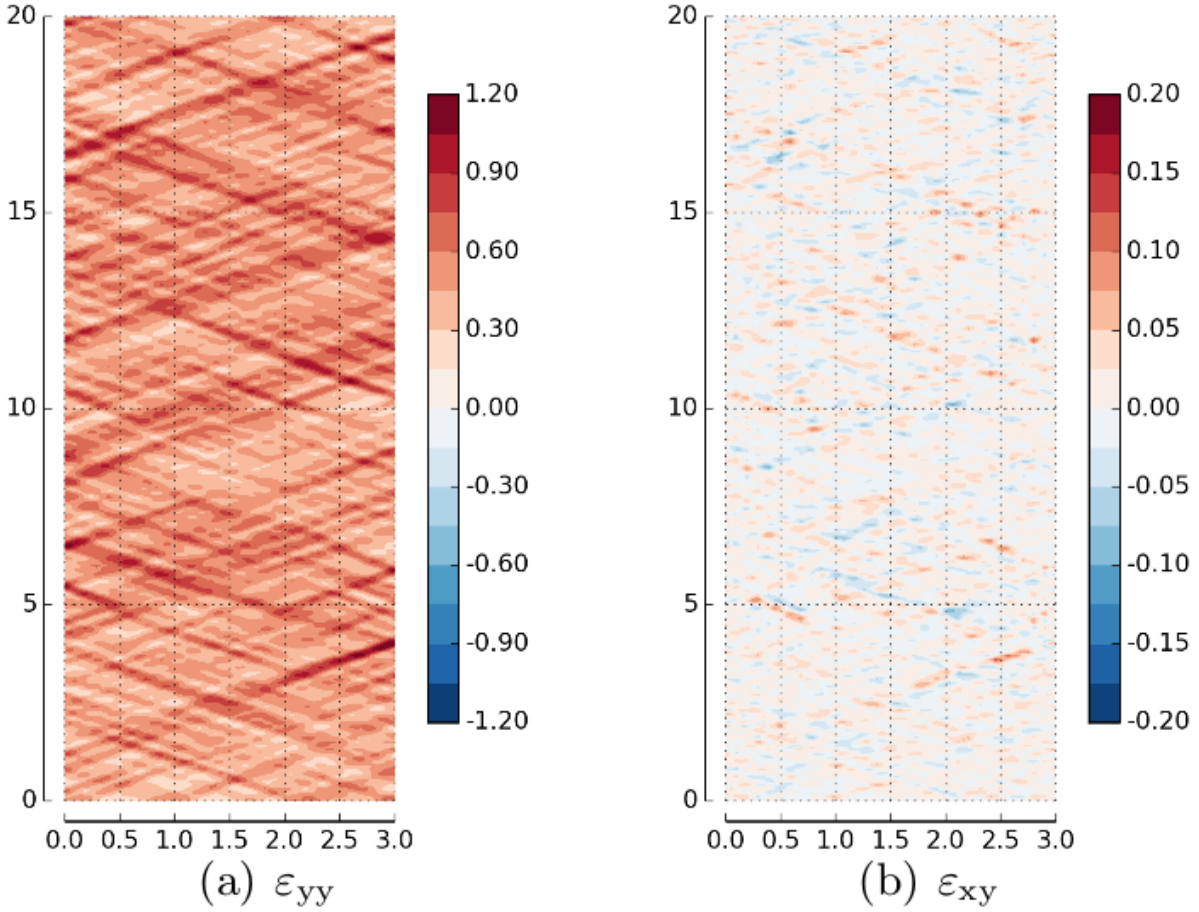
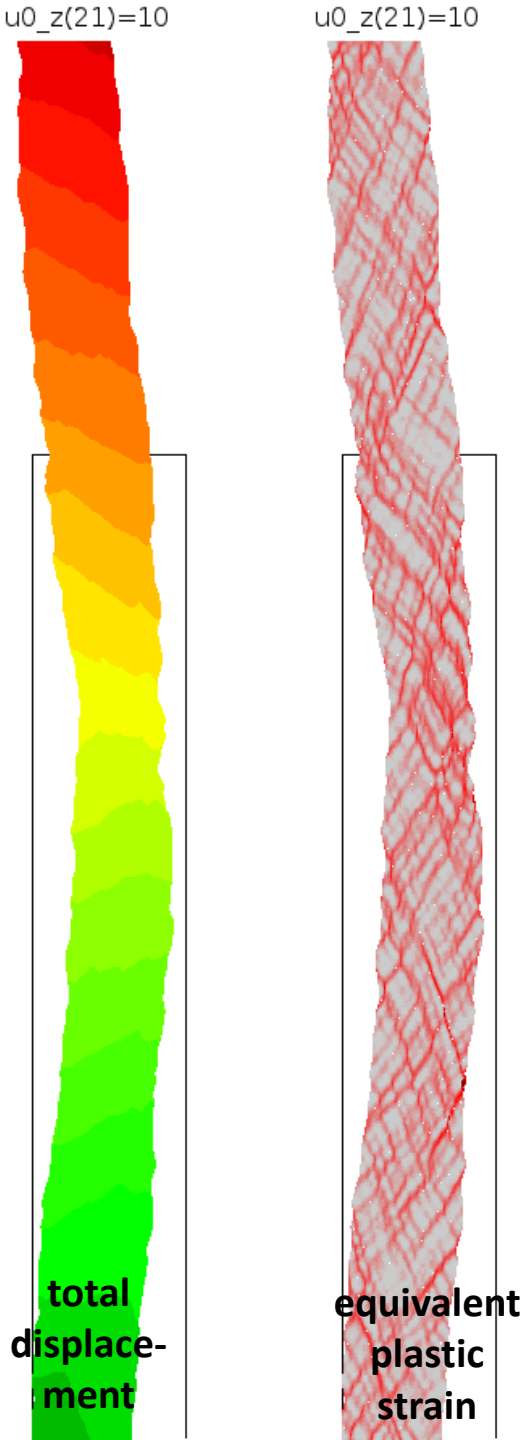
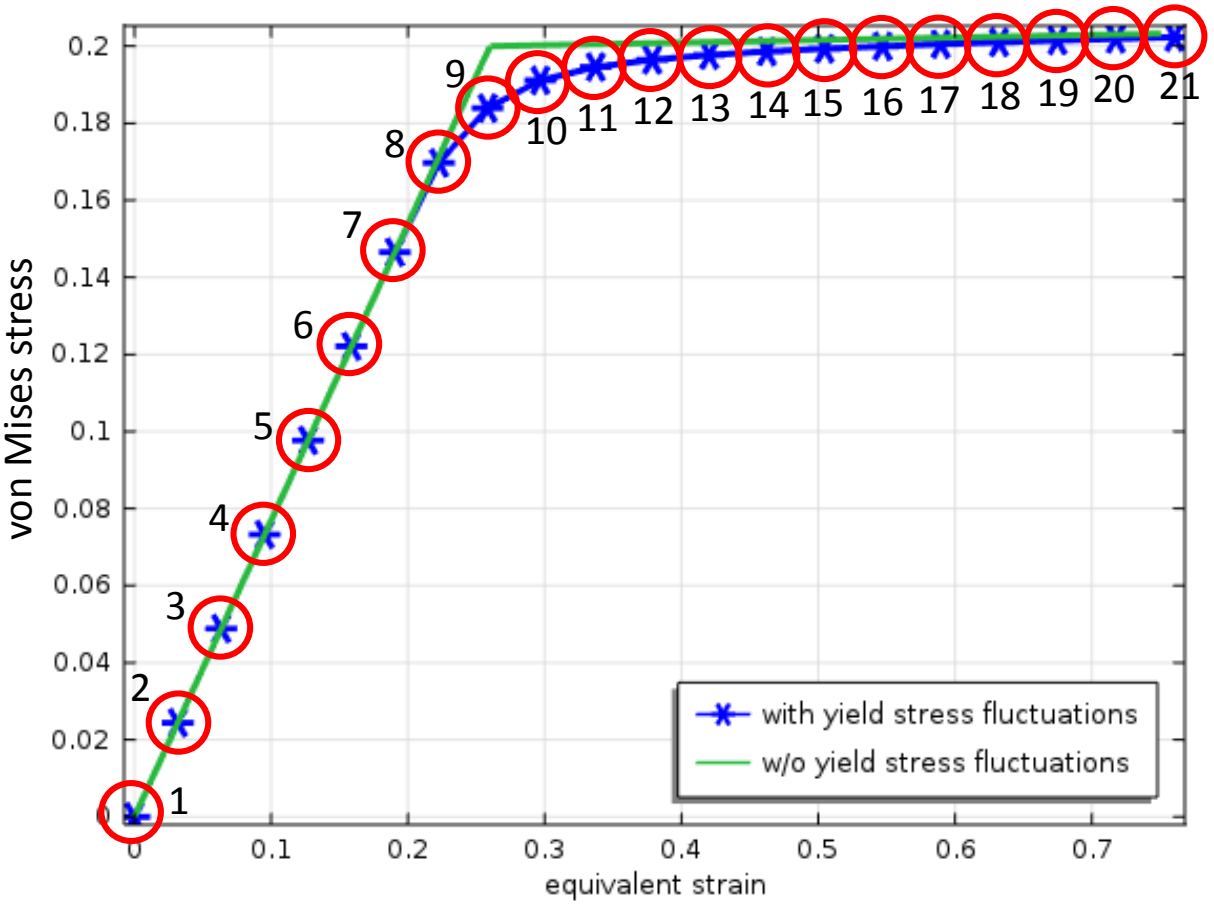


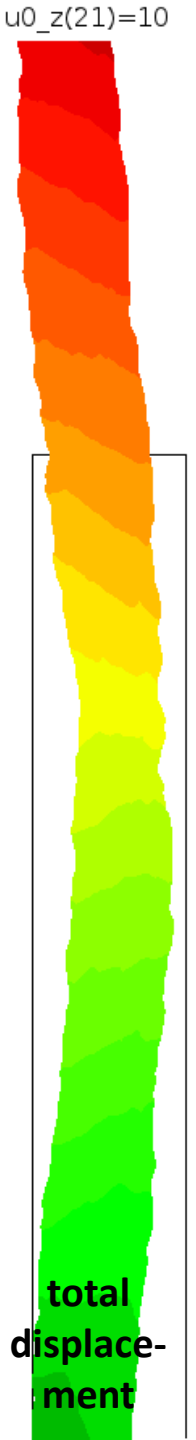
Fig. 3 Examples of local strain distributions at maximum tensile strain $\varepsilon_{yy} = 0.5$: (a) axial strain $\varepsilon_{xx} \approx -\varepsilon_{yy}$ and (b) shear strain ε_{xy}. Shear bands occur under an angle of $\pi/4$ w.r.t. the horizontal axis. Note that the geometry is not true to scale.



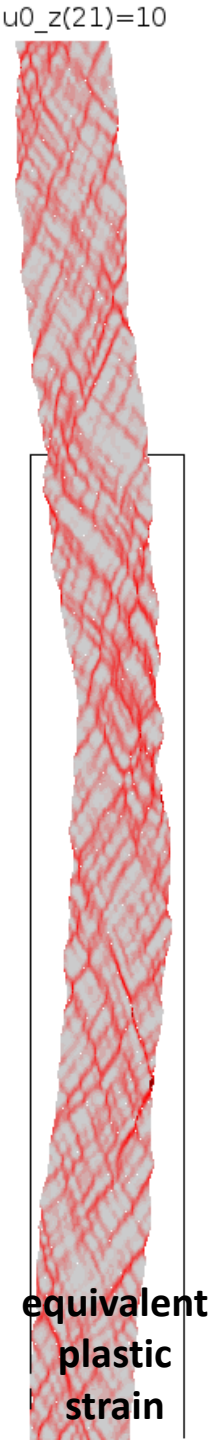
Output from a typical FEM simulations: average stress-strain diagramm



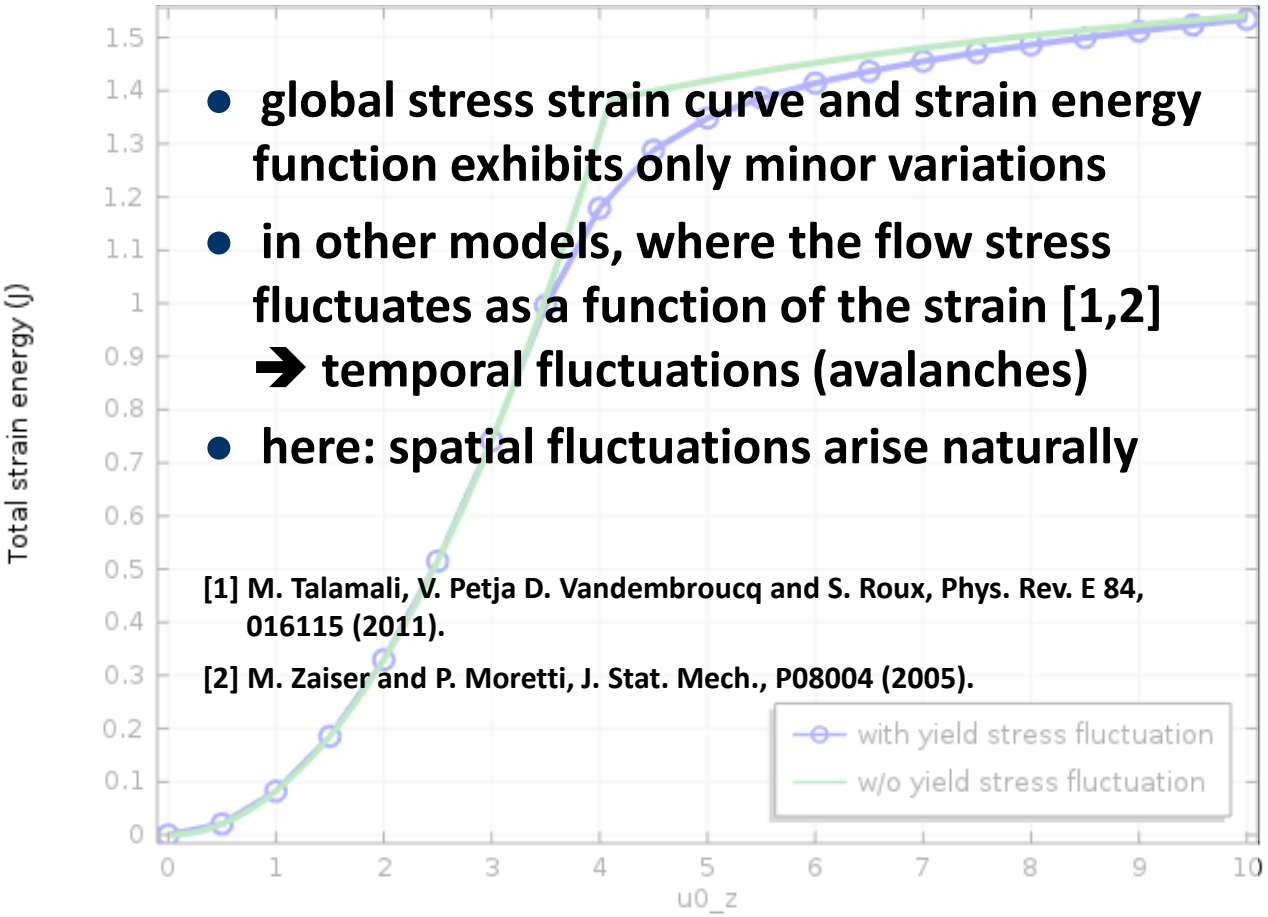
Output from a typical FEM simulation: total elastic strain energy



total displace-ment



equivalent plastic strain

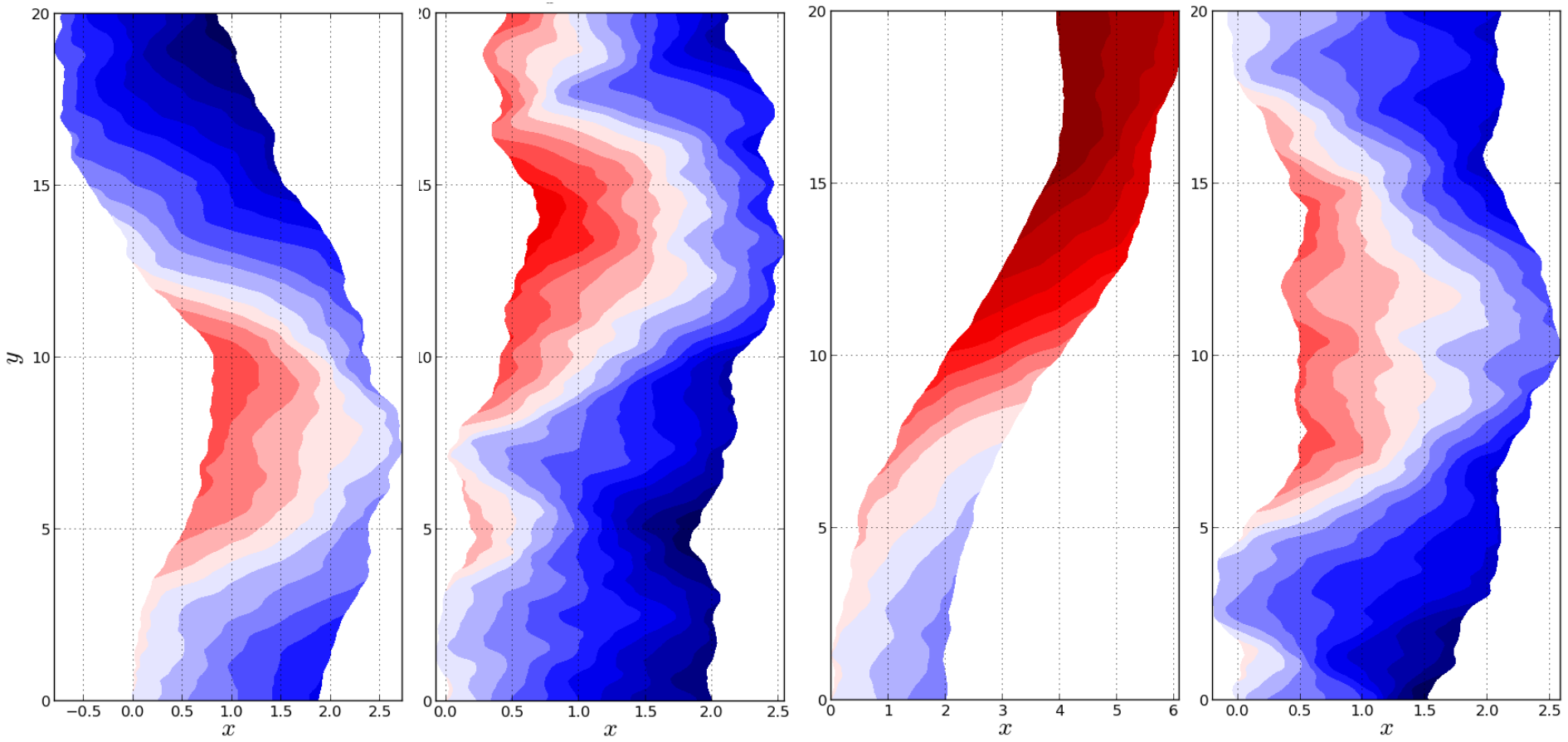


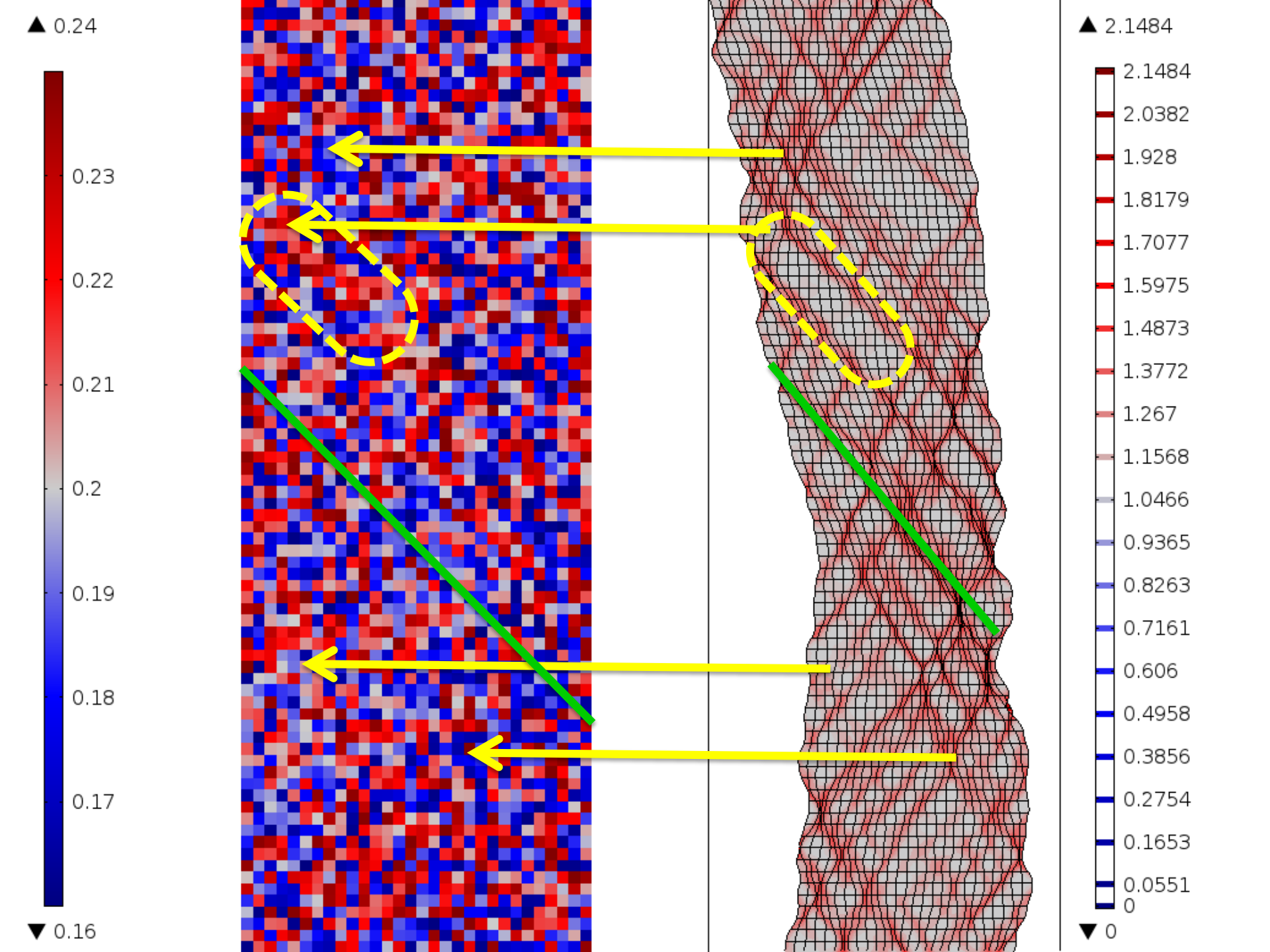
- global stress strain curve and strain energy function exhibits only minor variations
- in other models, where the flow stress fluctuates as a function of the strain [1,2] ➔ temporal fluctuations (avalanches)
- here: spatial fluctuations arise naturally

[1] M. Talamali, V. Petja D. Vandembroucq and S. Roux, Phys. Rev. E 84, 016115 (2011).

[2] M. Zaiser and P. Moretti, J. Stat. Mech., P08004 (2005).

Some typical deformation patterns

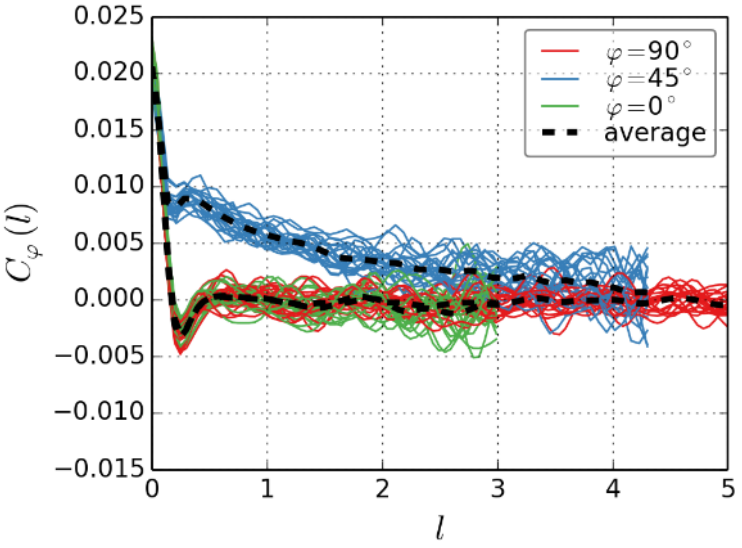



▲ 0.24
0.23
0.22
0.21
0.2
0.19
0.18
0.17
▼ 0.16
▲ 2.1484
2.1484
2.0382
1.928
1.8179
1.7077
1.5975
1.4873
1.3772
1.267
1.1568
1.0466
0.9365
0.8263
0.7161
0.606
0.4958
0.3856
0.2754
0.1653
0.0551
0
▼ 0

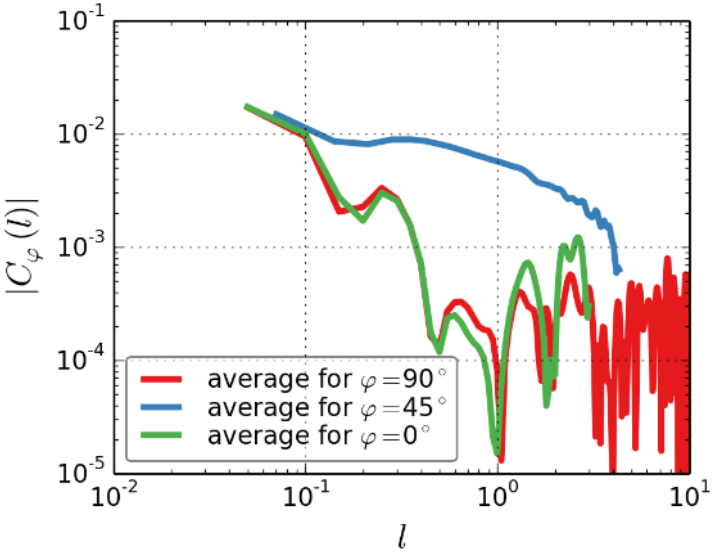
Strain correlation functions of local strains

Analyze the strain localization occuring in preferred directions by a spatial correlation function

$$C_\varphi(l) := \left\langle \left(\varepsilon(\mathbf{r}) - \langle\varepsilon\rangle\right)\left(\varepsilon(\mathbf{r}+\mathbf{l}_\varphi) - \langle\varepsilon\rangle\right)\right\rangle_{\mathbf{r}}$$

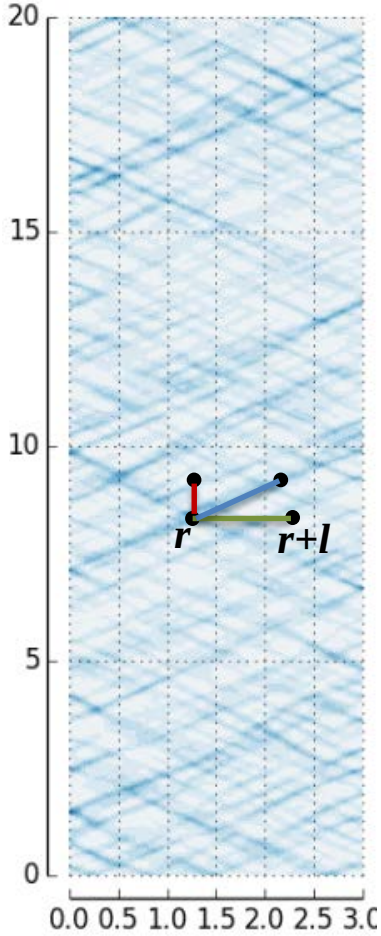


(a) $C_\varphi(l)$ plotted on linear axis



(b) absolute value of $C_\varphi(l)$ plotted on double logarithmic axis

$(\varepsilon_{11} + \varepsilon_{22})/2$



$$C_\varphi(l) := \left\langle (\varepsilon(\mathbf{r}) - \langle\varepsilon\rangle)\,(\varepsilon(\mathbf{r}+\mathbf{l}_\varphi) - \langle\varepsilon\rangle) \right\rangle_{\mathbf{r}}$$

- $\langle\varepsilon\rangle$ is the average strain:

$$\langle\varepsilon\rangle := \sum_{nx,ny} \text{strain}[nx, ny]$$

- $\varepsilon(\boldsymbol{r})$ is the strain at point $\boldsymbol{r}$, e.g. $\text{strain}[nx, ny]$
- $\varepsilon(\boldsymbol{r} + \boldsymbol{l}_\varphi)$ is the strain at another point
- $\langle \quad \rangle_r$ denotes the average of a quantity over all points $\boldsymbol{r}$

```
# NC is the max. nodes of half the corr. function
for ny in range(NC, dimY-NC):
   for nx in range(NC, dimX-NC):

      t1=(eps[ny][nx]-av_eps)

      for j in range(-NC,NC+1):
         for i in range(-NC,NC+1):

            t2 = eps[ny+j][nx+i]-av_eps
            C2d[j+NC,i+NC] += t1*t2

C2d /= float((dimY-(2*NC-1))*(dimX-(2*NC-1)))
```