Data-Driven Identification

Adrien Leygue, Michel Coret, Erwan Verron, Rian Seghir, Julien Réthoré

Research Institute in Civil Engineering and Mechanic (**GeM**)
Centrale Nantes, France

Find $(\mathrm{u}, \boldsymbol{\sigma})$ such that

Admissibility for u

$$\mathrm{u} = \mathrm{u}_d \text{ on } \Gamma_u + \text{regularity}$$

Balance of momentum

$$\mathrm{div}(\boldsymbol{\sigma}) = 0$$

Compatibility

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \mathrm{u} + \nabla^T \mathrm{u})$$

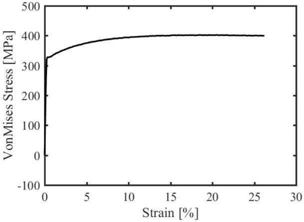
Balance of external forces

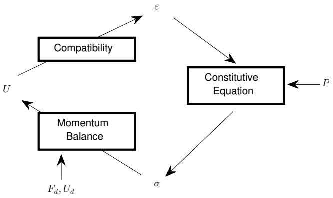
$$\boldsymbol{\sigma}(\mathrm{n}) = \mathrm{F}_d \text{ on } \Gamma_F$$

Regularization

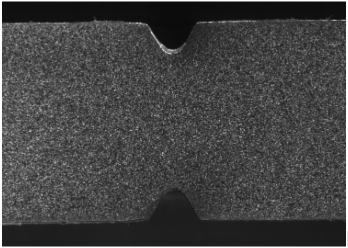
Sample with a geometry such that the solution is unique.
Usual specimen geometry for uni-, bi-, tri-axial testing,...

- "Engineering" approach



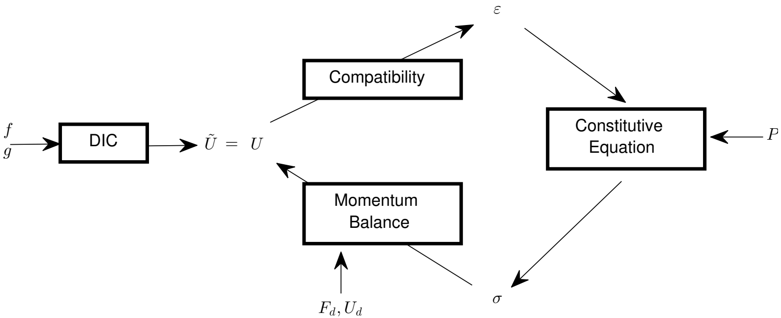


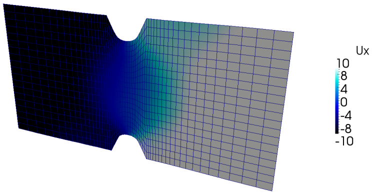
- Validation ?



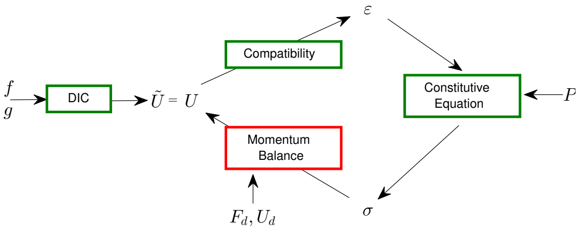
by G. Portemont at ONERA Lille

- Photomechanics

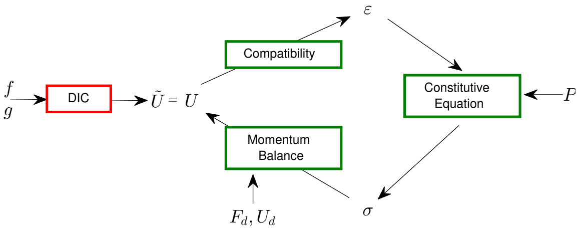




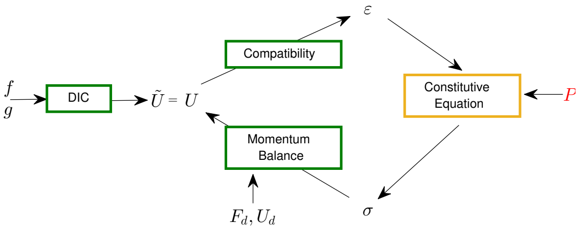
- Stress calculation



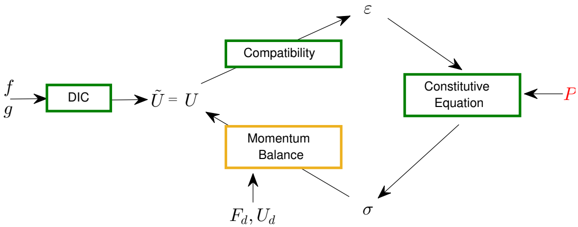
- Numerical simulation



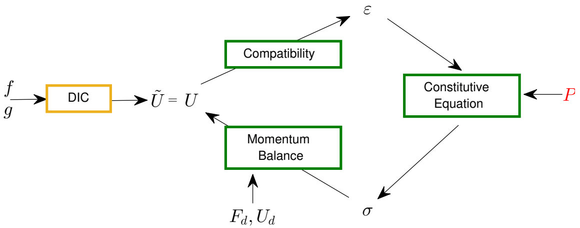
- Constitutive Equation Gap [Chrysochoos *et al.*]

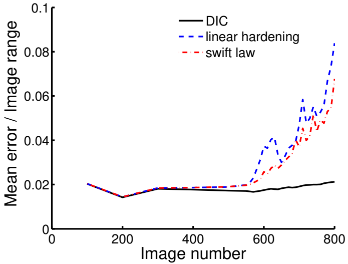


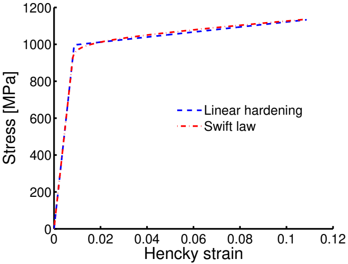
- Equilibrium gap [Claire et al., 2004], VFM [Grédiac et al., 2006]



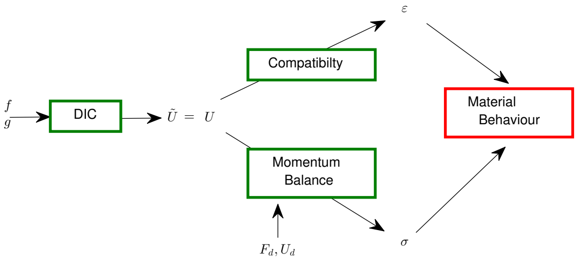
- FEMU [Lecompte et al., 2007, Leclerc et al., 2009]...





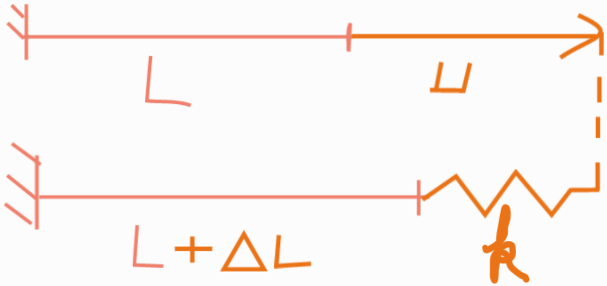


- Parametric techniques (using a constitutive equation)
  - \+ provide for the optimal set of parameters
  - +/- tell that the constitutive equation is not correct
  - \- how to improve it
  - \- kinematically consistent direct FEA
- Non-parametric (without using a constitutive equation)



SIMPLE PROBLEM

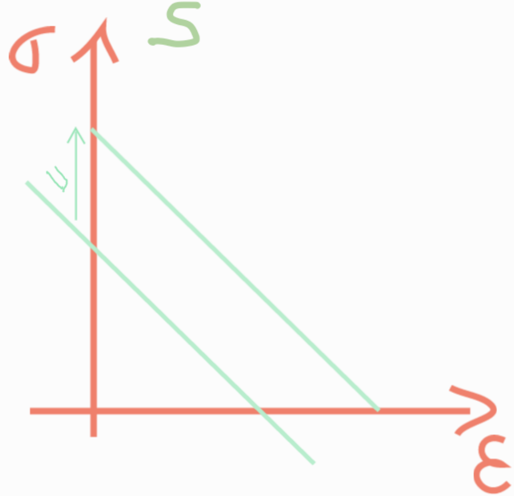
Problem definition



Balance of momentum

$$\sigma = (u - \Delta L)\frac{k}{S}$$

$$\sigma = \frac{uk}{S} - \varepsilon \frac{Lk}{S}$$

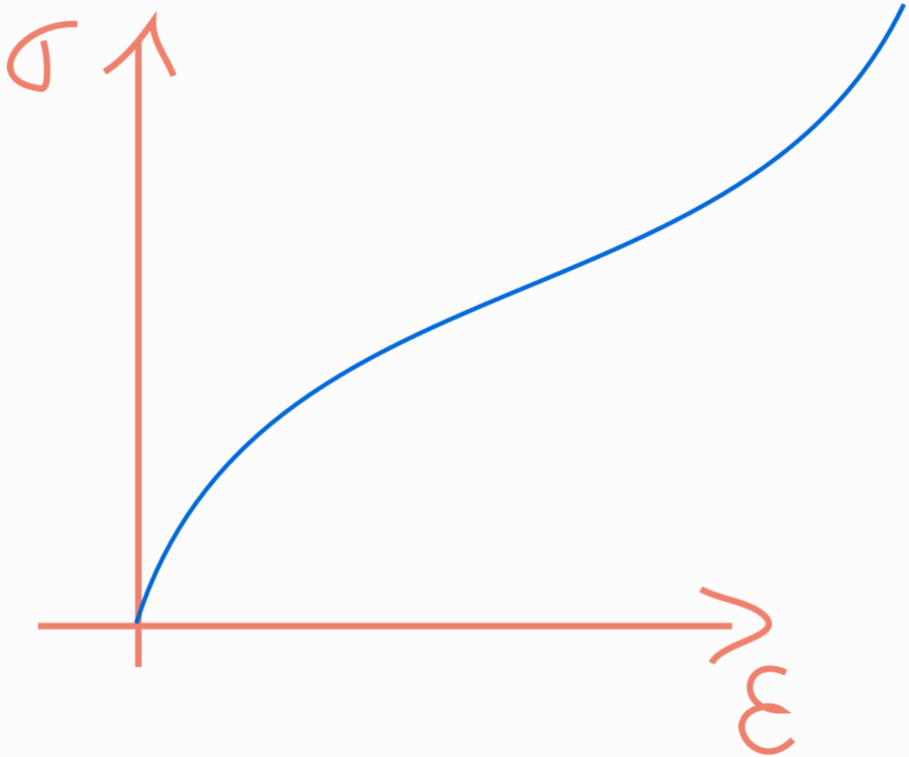


Compatibility

$$\varepsilon = \frac{\Delta L}{L}$$

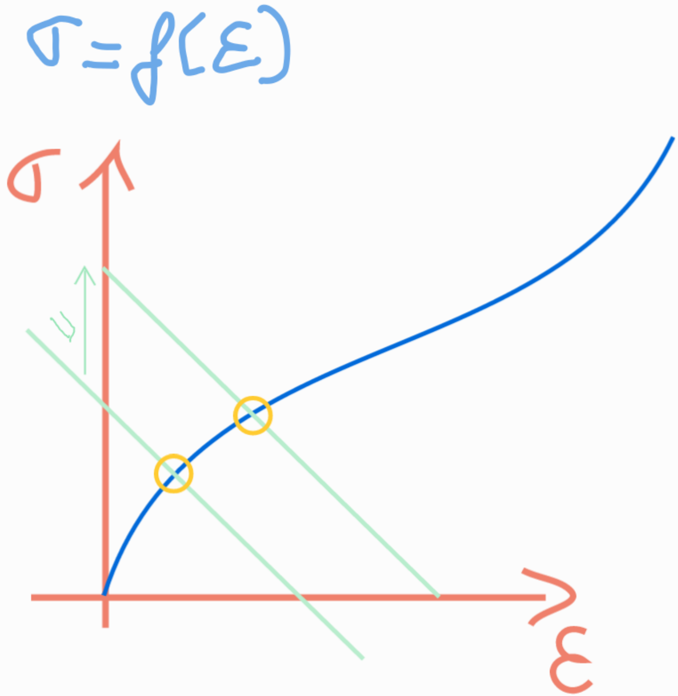
CONSTITUTIVE LAW

$\sigma = f(\varepsilon)$

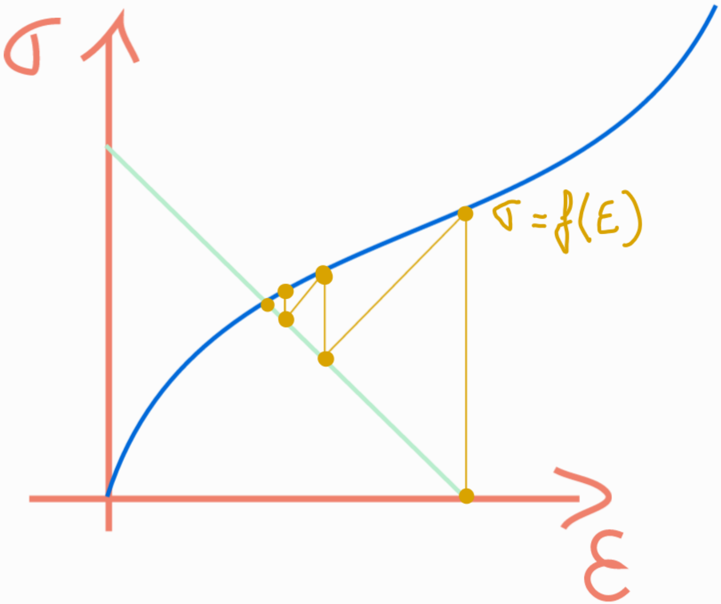


RESOLUTION

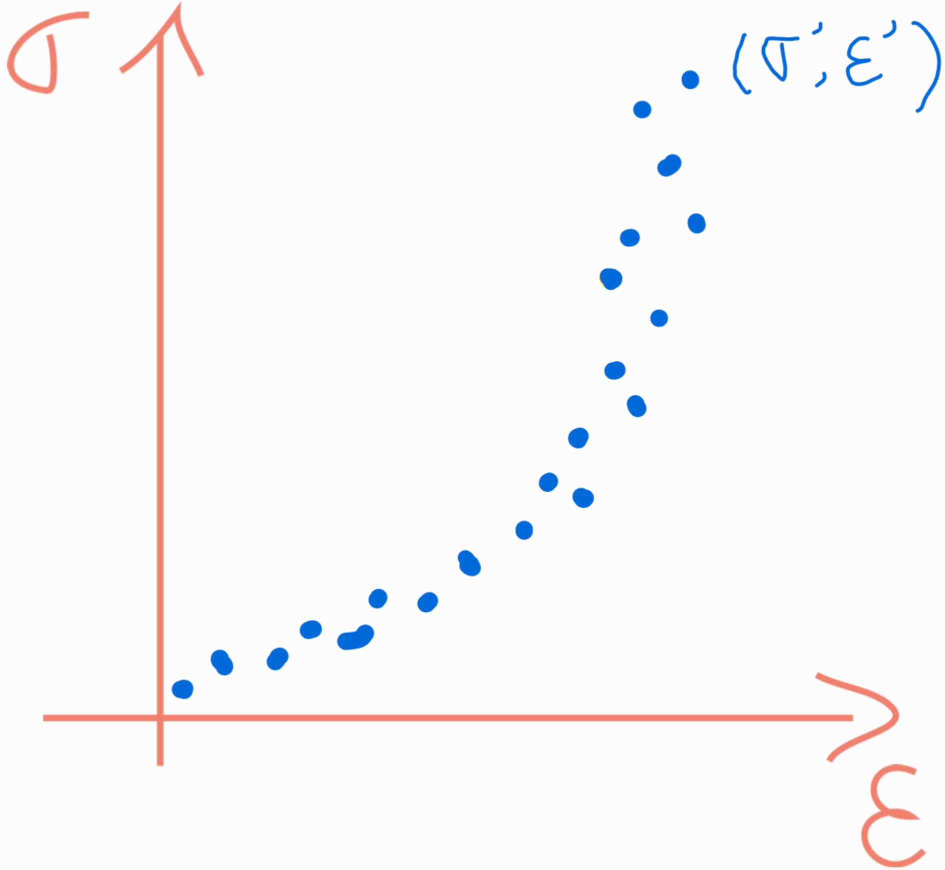
Expected solution



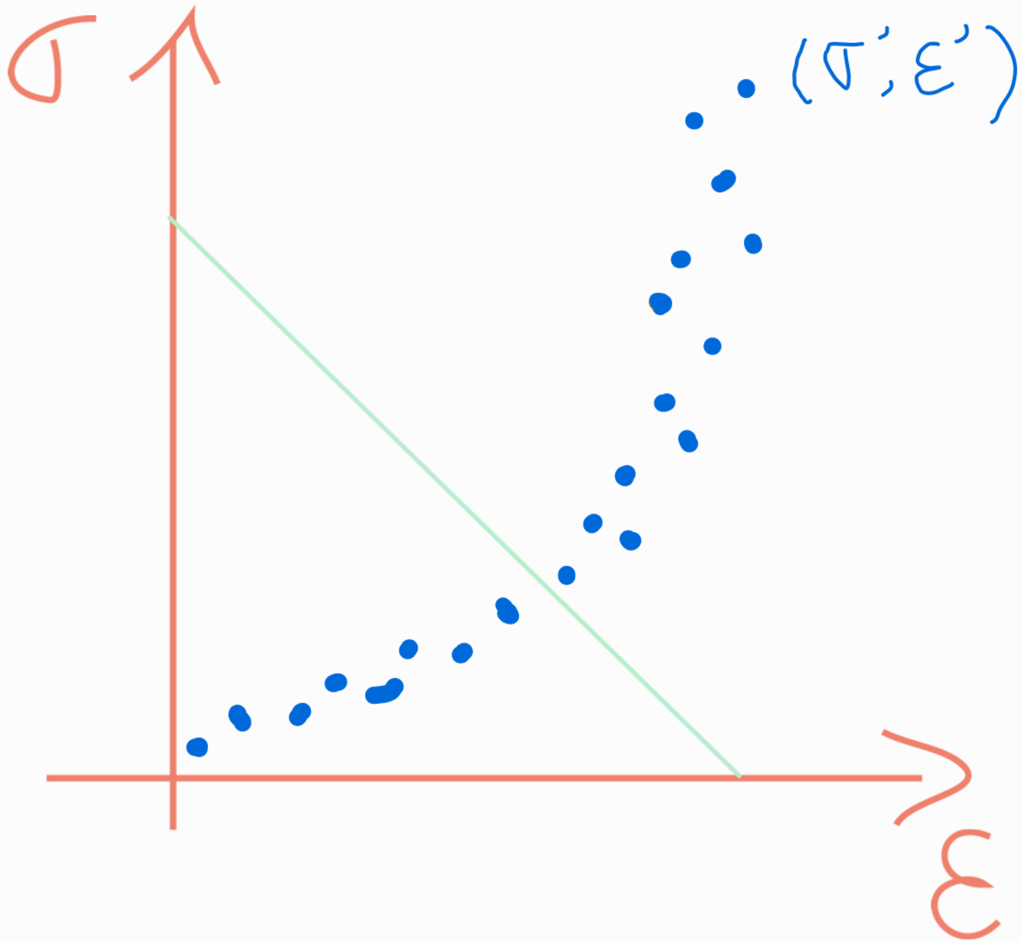
Non linear problem resolution



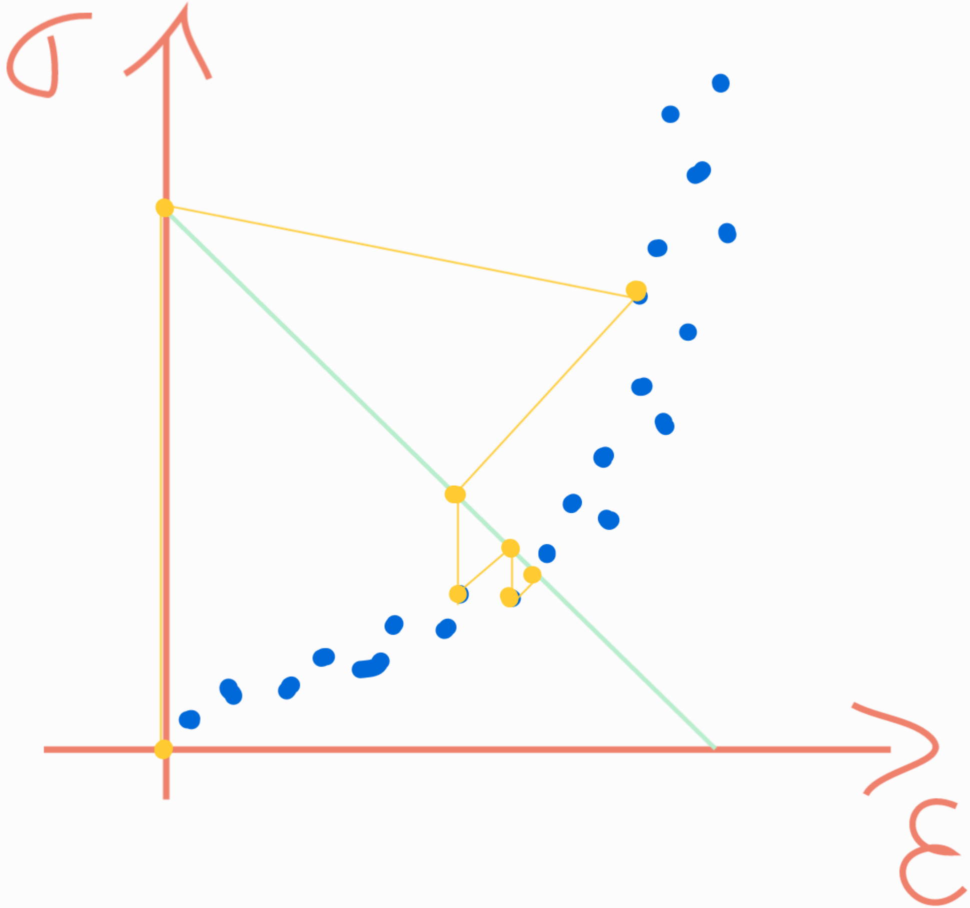
CONSTITUTIVE DATA



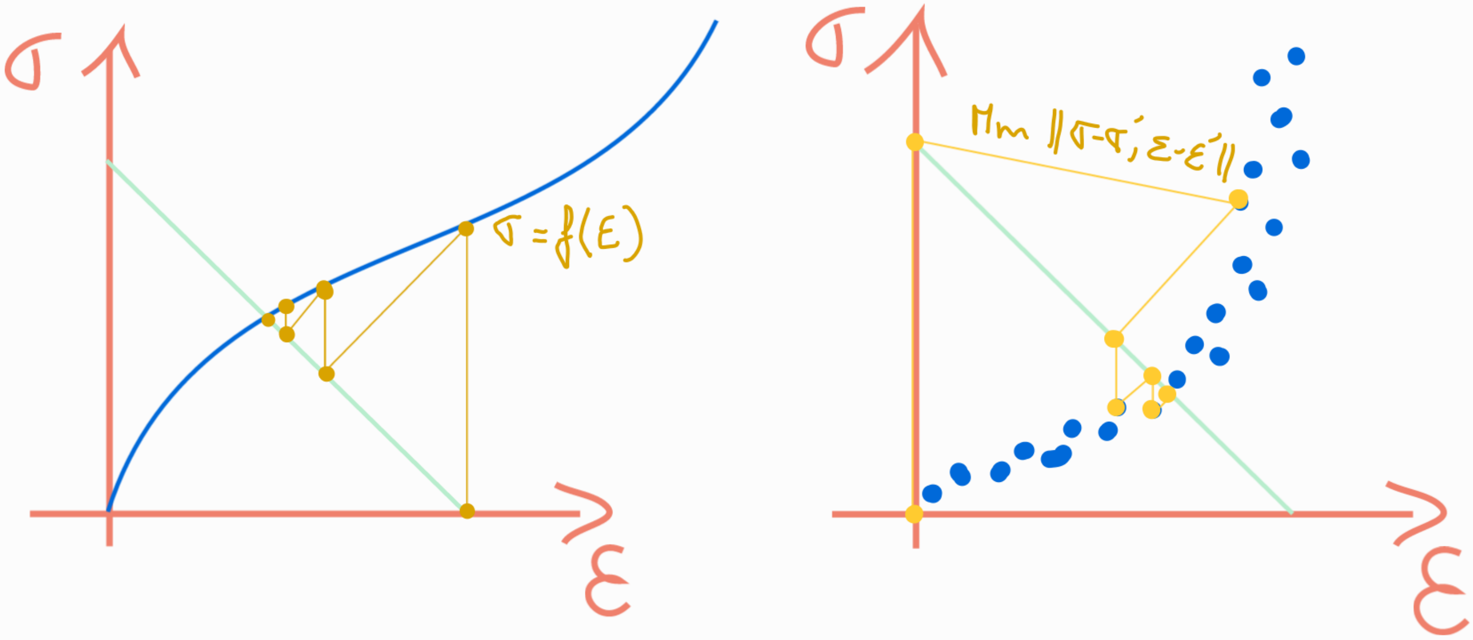
DATA-DRIVEN COMPUTATIONAL MECHANICS



DATA-DRIVEN SOLVER



INTRISINC DIFFERENCE



Find $(\mathbf{u}, \boldsymbol{\sigma})$ such that:

Admissibility for u

$$\mathbf{u} = \mathbf{u}_d \text{ on } \Gamma_u + \text{regularity}$$

Balance of momentum

$$\operatorname{div}(\boldsymbol{\sigma}) = 0$$

Compatibility

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \mathbf{u} + \nabla^T \mathbf{u})$$

Balance of external forces

$$\boldsymbol{\sigma}(\mathbf{n}) = \mathbf{F}_d \text{ on } \Gamma_F$$

Regularization

Sample with a geometry such that the solution is unique.
Usual specimen geometry for uni-, bi-, tri-axial testing,...

Find $(\mathbf{u}, \boldsymbol{\sigma})$ such that:

Admissibility for u

$$\mathbf{u} = \mathbf{u}_d \text{ on } \Gamma_u + \text{regularity}$$

Balance of momentum

$$\mathrm{div}(\boldsymbol{\sigma}) = 0$$

Compatibility

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \mathbf{u} + \nabla^T \mathbf{u})$$

Balance of external forces

$$\boldsymbol{\sigma}(\mathbf{n}) = \mathbf{F}_d \text{ on } \Gamma_F$$

Regularization

Constitutive relation

$$\boldsymbol{\sigma} = f(\boldsymbol{\varepsilon}, \ldots)$$

Non-parametric identification

From a measured **u**, find $(\boldsymbol{\sigma})$ such that:

Admissibility for u

$$\mathbf{u} = \mathbf{u}_d \text{ on } \Gamma_u \text{ + regularity}$$

Balance of momentum

$$\mathrm{div}(\boldsymbol{\sigma}) = 0$$

Compatibility

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \mathbf{u} + \nabla^T \mathbf{u})$$

Balance of external forces

$$\boldsymbol{\sigma}(\mathbf{n}) = \mathbf{F}_d \text{ on } \Gamma_F$$

Regularization

Minimizing the deviation (energy norm in the phase space) from the mean response (ϵ^*, σ^*).

Implementation

DDI_implementation

October 9, 2025

1 How DDI Works ?

Authors : Outlaw Group, Centrale Nantes, France

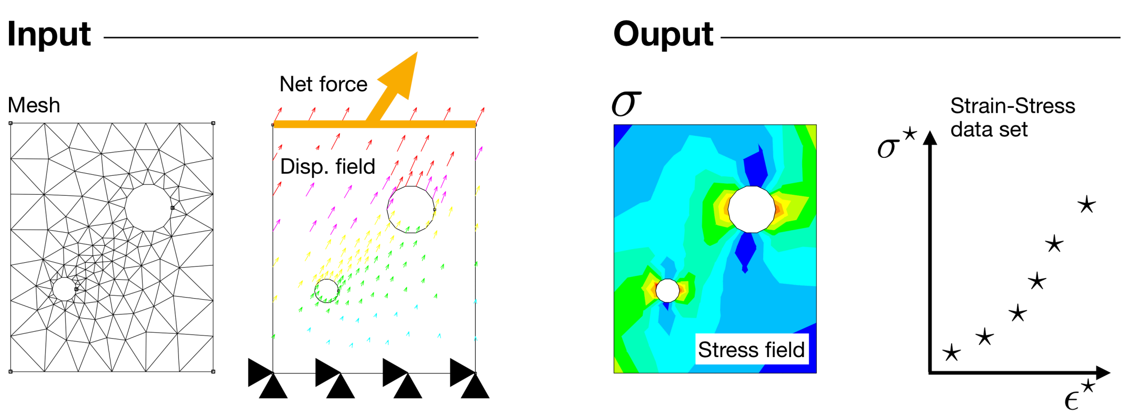
[Leygue et al., Data-based derivation of material response, CMAME 2018]

1.1 Basics and an illustrative example

1.1.1 DDI at a glance

Data Driven Identification (DDI) aims at computing stresses on a structure where displacement field (u) and applied forces (f) are known but not the mechanical behaviour law. DDI gives us, first, the **balanced stress field** (σ) in the structure (stress tensor in each element) and second, a **Strain-Stress data set** $(\varepsilon^\star, \sigma^\star)$. The Strain-Stress data set is needed to solve the ill-posed problem of equilibrium in the absense of a material law, its size, (N_{ss}), is smaller than the number of elements (N_e) of the structure. This data set can also be seen as an atractor allowing the minimisation of the variance of the stress field.

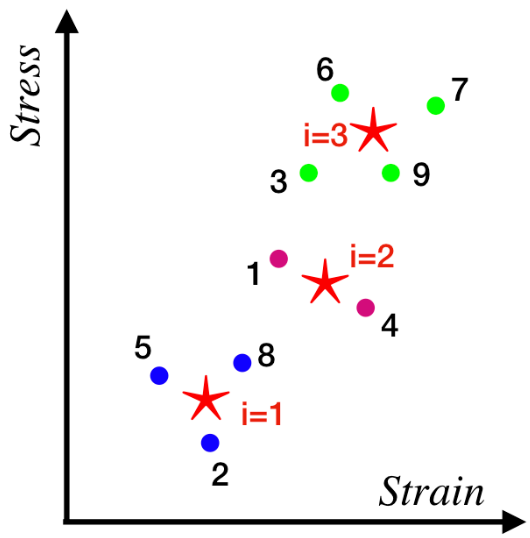
> **Input** * 2D mesh of a structure * Node displacement field u of size N_e - Typically from DIC * Applied net forces - Typicaly from a load cell
>
> **Output** * Stress field (σ) - preserving balance of momentum * Clusturized Strain-Stress data base $(\varepsilon^\star, \sigma^\star)$ of size N_{ss}
>
> **Parameters** 1. r : clustering ratio [no unit] $\left(r = \frac{N_e}{N_{ss}}\right)$ 2. $\mathcal{C}_0$: Algorithmic stiffness [Pa]



1.2 Definitions, Problem & Equations to be solved

1.2.1 1. Definitions:

- **The Clustering** is the pairing between each couple $(\varepsilon, \sigma)_e$ of an element e with a couple i of the data-set $(\varepsilon^\star, \sigma^\star)_i$. $(\varepsilon^\star, \sigma^\star)$ is a weighted average of a subset of (ε, σ). Obviously, $N_e > N_{ss}$.
- Clusterization is defined by the the pairing operator S such as $\sigma_e^\star = P_{ie}\sigma_i^\star$. S is an unknown of the problem.
- Illustration of S :



$$S = \begin{bmatrix} 0 & {\color{magenta}1} & 0 \\ {\color{blue}1} & 0 & 0 \\ 0 & 0 & {\color{green}1} \\ 0 & {\color{magenta}1} & 0 \\ {\color{blue}1} & 0 & 0 \\ 0 & 0 & {\color{green}1} \\ 0 & 0 & {\color{green}1} \\ {\color{blue}1} & 0 & 0 \\ 0 & 0 & {\color{green}1} \end{bmatrix}$$

- $\mathcal{C}_0$ **norm** is define on a couple $(\mathbf{a}, \mathbf{b})$ of tensors of order 2 (typically a strain and a stress) by the definition :

$$\|\mathbf{a}, \mathbf{b}\|_{\mathcal{C}_0} = \left[\mathbf{a} : \mathcal{C}_0 : \mathbf{a} + \mathbf{b} : \mathcal{C}_0^{-1} : \mathbf{b}\right]^{\frac{1}{2}}, \quad \mathcal{C}_0 \text{ a 4th order symmetric positive definite tensor.}$$

1.2.2 2. Problem:

ε and f being known, find $(\sigma, \varepsilon^\star, \sigma^\star, S)$ minimizing the $\mathcal{C}_0$ norm between (ε, σ) and $(\varepsilon^\star, \sigma^\star)$ and which preserve the equilibrium of the structure.

- $$\|\varepsilon - S\varepsilon^\star, \sigma - S\sigma^\star\|_{\mathcal{C}_0} = \left[(\varepsilon - S\varepsilon^\star) : \mathcal{C}_0 : (\varepsilon - S\varepsilon^\star) + (\sigma - S\sigma^\star) : \mathcal{C}_0^{-1} : (\sigma - S\sigma^\star)\right]^{\frac{1}{2}},$$
- $$\text{and} \quad div\, \sigma = f, \quad \forall M \in \text{Structure}$$

$\mathcal{C}_0$ has the dimension of a stiffness tensor.

That can be rewritten in a variational form:

$$\text{Find, } ({\color{green}\sigma}, {\color{red}\varepsilon^\star}, {\color{red}\sigma^\star}, {\color{orange}\eta}) \text{ that make the following functional stationary, for a given } {\color{blue}S}$$

$$\mathcal{E}(\sigma, \varepsilon^\star, \sigma^\star, S, \eta) = \frac{1}{2}\int_V \left[\|\varepsilon - S\varepsilon^\star, \sigma - S\sigma^\star\|^2_{\mathcal{C}_0} - (div\, \sigma - f)\, \eta \right] dV$$

With η a Lagrange Multiplier (dimension of a displacement vector in [m]).

1.2.3 3. Derivation of variational equation:

The derivation of the above fonctional \$ $= 0$ \$ gives us the following equations needed to find the unknowns $(\sigma, \varepsilon^\star, \sigma^\star, \eta)$:

$$\begin{aligned}
&\delta\varepsilon^\star && \Rightarrow \int_V \mathcal{C}_0 : (\varepsilon - S\varepsilon^\star)\, dV && = 0 \quad \forall V \quad (1)\\
&\delta\sigma^\star && \Rightarrow \int_V \mathcal{C}_0^{-1} : (\sigma - S\sigma^\star)\, dV && = 0 \quad \forall V \quad (2)\\
&\delta\eta && \Rightarrow \int_V (div\, \sigma - f)\, dV && = 0 \quad \forall V \quad (3)\\
&\delta\sigma && \Rightarrow (\sigma - S\sigma^\star) = \mathcal{C}_0 : grad^s(\eta) && \forall M \quad (4)
\end{aligned}$$

- Equations 1 and 2 state that $\varepsilon^\star$ (resp. $\sigma^\star$) is a weighted average of a cluster of ε (resp. σ)
- Equation 3 is the standard balance of momentum equation on the whole structure.
- Equation 4 states the distance between σ and $\sigma^\star$ is proportional to the gradient of \$ grad^s( )\$ (a kind of strain)

1.3 Discrete format

In 1D using bar elements (instead 2D elements): > - (ε, σ) become scalar (instead of tensors) > - $\mathcal{C}_0$ is just a scalar C_0 > - Only one snapshot is considered > - the bar elements are assigned a unit heigh > - the unit for length is pixel

The variational form can be rewritten as:

$$\begin{aligned}
\mathcal{E}(S_e, E^\star, S^\star, S, L) = \quad & \frac{1}{2} C_0 \left[E_e - S\ E^\star\right]^T\ W\ \left[E_e - S\ E^\star\right] \\
& + \frac{1}{2} C_0^{-1} \left[S_e - S\ S^\star\right]^T\ W\ \left[S_e - S\ S^\star\right] \\
& - \left[B^T\ W\ S_e - F_{ext}\right]^T\ L
\end{aligned}$$

where B is such that $E_e = B\, U$, U being a vector collecting the nodal displacements (data), W is a diagonal matrix collecting the surface of each bar element (E_e, S_e) are vectors collecting the strain and stress in each element, $(S^\star, S^\star)$ are vectors containing the values of $(\varepsilon^\star, \sigma^\star)$ and L a vector collecting the value of nodal Lagrange multipliers. The stationarity conditions recast as:

$$
\begin{array}{llr}
\delta E^\star & \Rightarrow \quad (S^T\ S)E^\star = S^T\ E_e & (I) \\
\delta S^\star & \Rightarrow \quad (S^T\ S)S^\star = S^T\ S_e & (II) \\
\delta L & \Rightarrow \quad B^T\ W\ S_e = F_{ext} & (III) \\
\delta S_e & \Rightarrow \quad C_0^{-1}W\,(S_e - S\ S^\star) + W\ B\ L = 0 & (IV)
\end{array} \tag{1}
$$

In pratice the full F_{ext} vector is not known. Only one component of the resulting force applied on one side of the sample can be measured. A two matrices D and D_c are introduced.

$$
D = B^T\ W \qquad D_c = \Lambda B^T\ W
$$

where Λ is such that: >- the lines of D corresponding to vanishing nodal forces are kept, >- the lines of D corresponding to non-vanishing nodal forces are removed, >- lines corresponding to measuring resulting force (sum of one component of nodal forces along one edge e.g.) are added. The external force vector is modified accordingly and named $\bar{F}_{ext}$.

The first two systems of equations (I, II):

$$
(S^T\ S)E^\star = S^T\ E_e \tag{2}
$$

$$
(S^T\ S)S^\star = S^T\ S_e \tag{3}
$$

$$
\tag{4}
$$

are easily solved independently as $(S^T\ S)$ is diagonal. In the following they will be solved together when updating S the selection matrix obtained from the labelling of a k-means algorithm.

The two next lines (III, IV) are included in the following linear system

$$
\begin{bmatrix} C_0^{-1}W & D_c^T \\ D_c & 0 \end{bmatrix} \begin{bmatrix} S_e \\ L \end{bmatrix} = \begin{bmatrix} C_0^{-1}W\ S\ S^\star \\ \bar{F}_{ext} \end{bmatrix}
$$

The resolution is performed by computing first the Lagrange multipliers L:

$$
(Dc\ W^{-1}\ Dc^T)L = Dc\ S\ S^\star - \bar{F}_{ext}
$$

and then the stress:

$$
S_e = S\ S^\star - C_0\ W^{-1}\ D_c^T\ L
$$

The following algorithm is used latter on >- #### While > - Update S using k-means > - Solve $(S^T\ S)E^\star = S^T\ E_e$ > - Solve $(S^T\ S)S^\star = S^T\ S_e$ > - Solve $(Dc\ W^{-1}\ Dc^T)L = Dc\ S\ S^\star - \bar{F}_{ext}$ > - Update \$S_e=S$_{S\text{-C_0}_{W^{\{-1\}}}\text{D_c}_T}$L \$

Note that the three steps could be performed in a single one but, usually k-means function use a L_2 norm for evaluating the distance between clusters and samples. Here the norm for this distance is $\|\mathbf{a}, \mathbf{b}\|_{C_0}$. K-means is thus called with $(E_e\sqrt{C_0}, S_e/\sqrt{C_0})$ instead of (E_e, S_e) as input. The coordinates of the clusters $(E^\star, S^\star)$ are thus not a direct output of kmeams. Their are udpated by solving systems (I, II) suing the new S.

1.4 Numerical implementation

[1]:
```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy import ndimage
from scipy.sparse import csr_matrix as smatrix
import scipy.sparse.linalg as splinalg
import h5py
import scipy.io as sio
from sklearn.cluster import KMeans
import fem
```

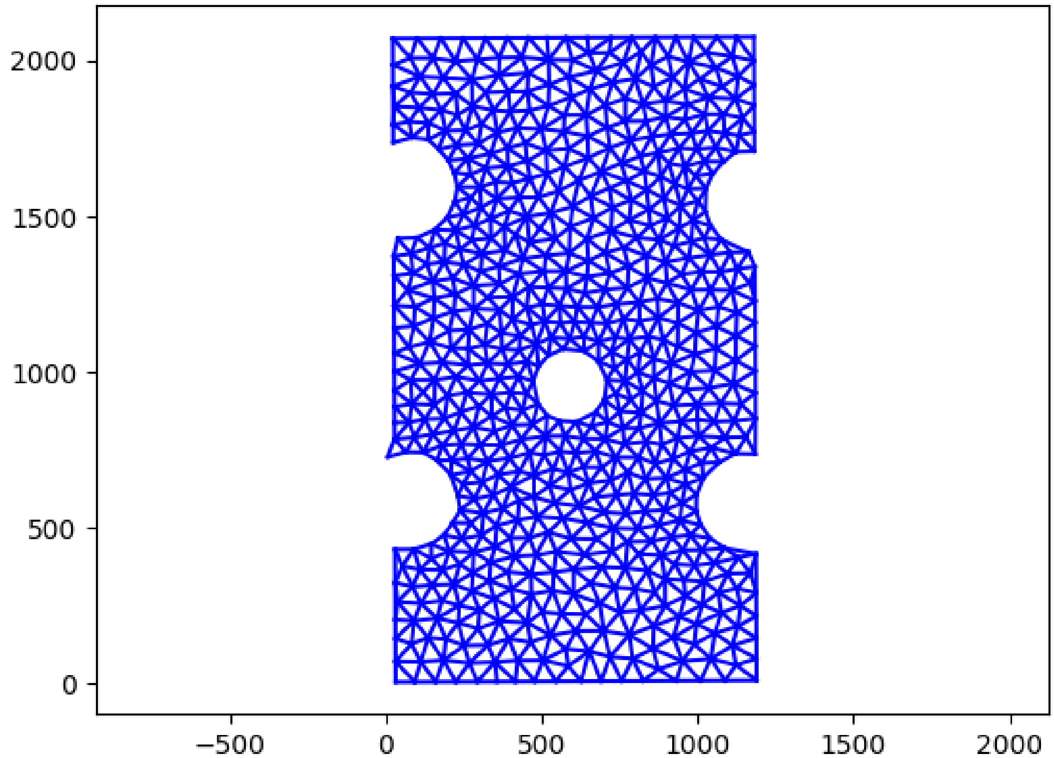
1.4.1 Application parameters

[2]:
```
# Units m->pixel=m/pix2m
# Pa=kg/m/s^2->kg/pixel/s^2=Pa*pix2m
# N = kg.m/s^2->kg.pixel/s^2=N/pix2m
# N/m = kg/s^2->kg/s^2=N/m
stdu=0.1 # noise level displacement in pixel
stdf=.0 # N noise level on force
pix2m=25.e-6; # pixel to m conversion
thickness=3e-3 # specimen thicness in m
```

1.4.2 DDI parameters

[3]:
```
Ns=50 # number of material states
inp='dic-coarse.res'# input file
(X,conn)=fem.readDICmesh(inp)# loading the mesh
model=fem.FEModel() # instantiating a model
model.X=X # Nodes
model.conn=conn # Connectivity
Nnodes=X.shape[0]
Nelems=conn.shape[0]

model.Assemble() # Assembly
W=model.W # Weighting matrix
B=model.B # B matrix for computing strain from displacement

npz=np.load('fem-from-dic.npz')
U=npz['U'] # Input displacement from FE simulation
Fres=(npz['Fres']+stdf)/thickness # Input force from FE simulation
Sref=npz['Sref']*pix2m # Stress field from the FE simulation used as imput data
E_e=B.dot(U) # Input strain
Eref=E_e # considered as the reference
dE_e=B.dot(stdu*np.random.randn(U.size))# noise
E_e=Eref+dE_e # strain to be considered as imput for DDI
```

```python
# Setting the algorithmic stiffness
L=max(X[:,1])-min(X[:,1])
dL=np.max(U[Nnodes::])-np.min(U[Nnodes::])
section=max(X[:,0])-min(X[:,0])
Co=(Fres/section)/(dL/L)*pix2m

# Display the sample
plt.plot(X[conn,0].T,X[conn,1].T,'b-');
plt.axis('equal');
```

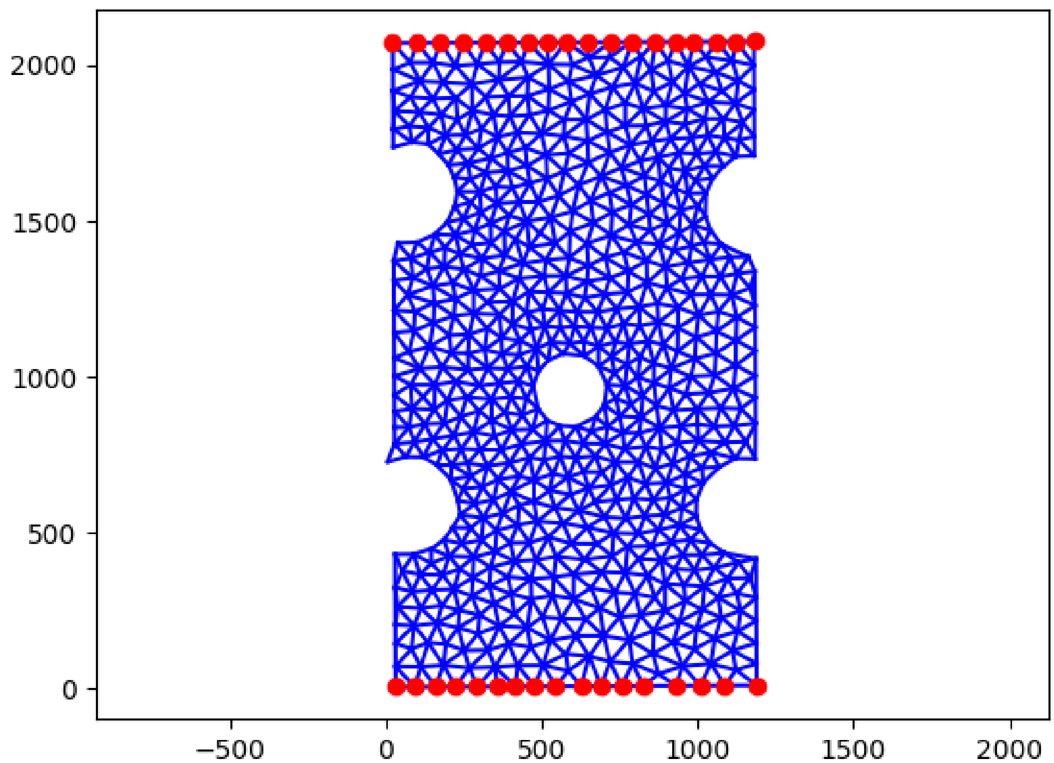


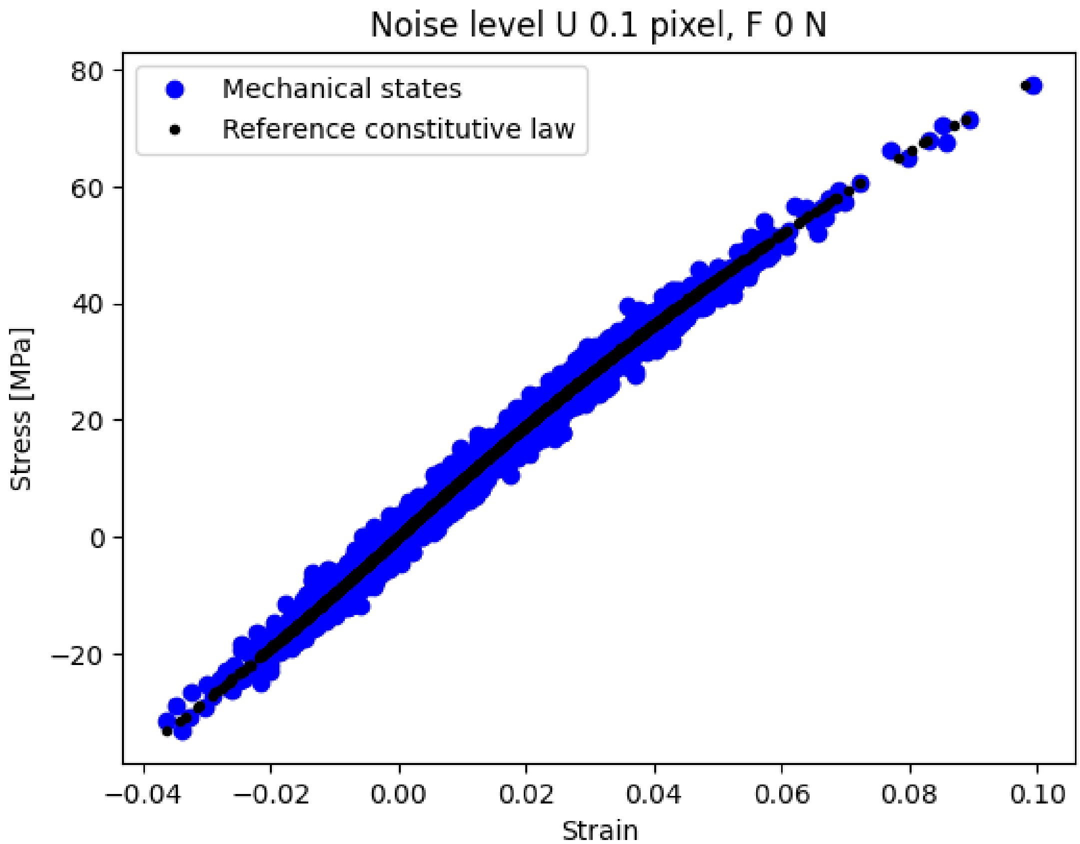
1.4.3 Boundary conditions

The internal force vector is 0 everywhere except: >- on the bottom where the distribution of its x component is unknown >- on the bottom where the distribution of its y component is unknown >- on the top where the distribution of its x component is unknown >- on the top where the sum of the distribution of its y component equals the measured load

[4]:
```python
top=X[:,1]>max(X[:,1])*0.99
bot=X[:,1]<max(X[:,1])*0.01

nodes_index=np.arange(Nnodes)
top_nodes=nodes_index[top]
```

```
ntop=top_nodes.size
free=np.logical_not(np.logical_or(top, bot))
free_nodes=nodes_index[free]
nfree=free_nodes.size
Free_x=smatrix((np.ones(nfree),(np.
 ↪arange(nfree),free_nodes)),shape=(2*nfree+1,2*Nnodes))
Free_y=smatrix((np.ones(nfree),(np.
 ↪arange(nfree)+nfree,free_nodes+Nnodes)),shape=(2*nfree+1,2*Nnodes))
Const_y=smatrix((np.ones(ntop),(2*nfree*np.
 ↪ones(ntop),top_nodes+Nnodes)),shape=(2*nfree+1,2*Nnodes))


plt.plot(X[conn,0].T,X[conn,1].T,'b-');
plt.plot(X[top,0],X[top,1].T,'ro');
plt.plot(X[bot,0],X[bot,1].T,'ro');


plt.axis('equal');


ff=plt.figure()
plt.plot(E_e,Sref/pix2m*1.e-6,'bo',label='Mechanical states')
plt.plot(Eref,Sref/pix2m*1.e-6,'k.',label='Reference constitutive law');
plt.xlabel('Strain')
plt.ylabel('Stress [MPa]');
plt.title('Noise level U %g pixel, F %g N' % (stdu,stdf))
plt.legend();
```


2000
1500
1000
500
0
−500
0
500
1000
1500
2000



[5]:
```
# Operator assembly
D=B.T*W
Dc=(Free_x+Free_y+Const_y)*(B.T*W)
Fext=np.zeros(2*nfree+1)
Fext[-1]=Fres

iW=scipy.sparse.spdiags(1/W.diagonal(),0,Nelems,Nelems)
C=Dc*(iW*Dc.T)
LU=splinalg.splu(C)
```

```
/tmp/ipykernel_81593/876515232.py:9: SparseEfficiencyWarning: splu converted its
input to CSC format
  LU=splinalg.splu(C)
```

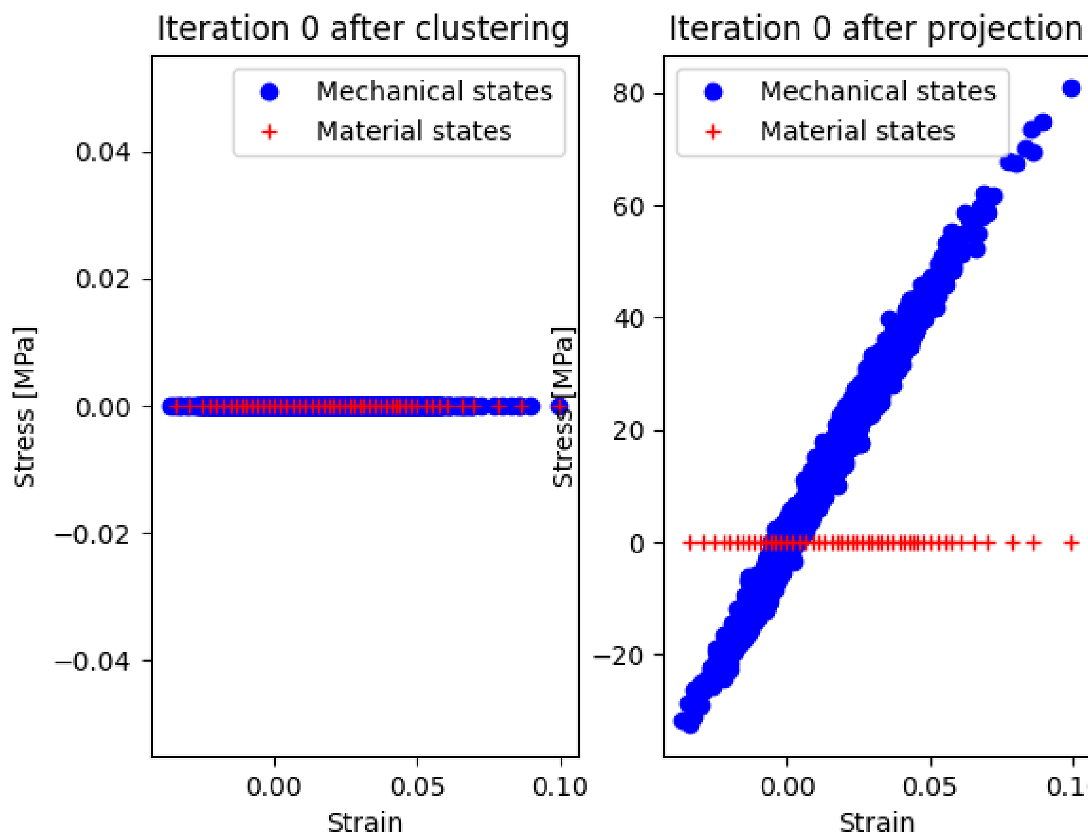
1.4.4 Resolution

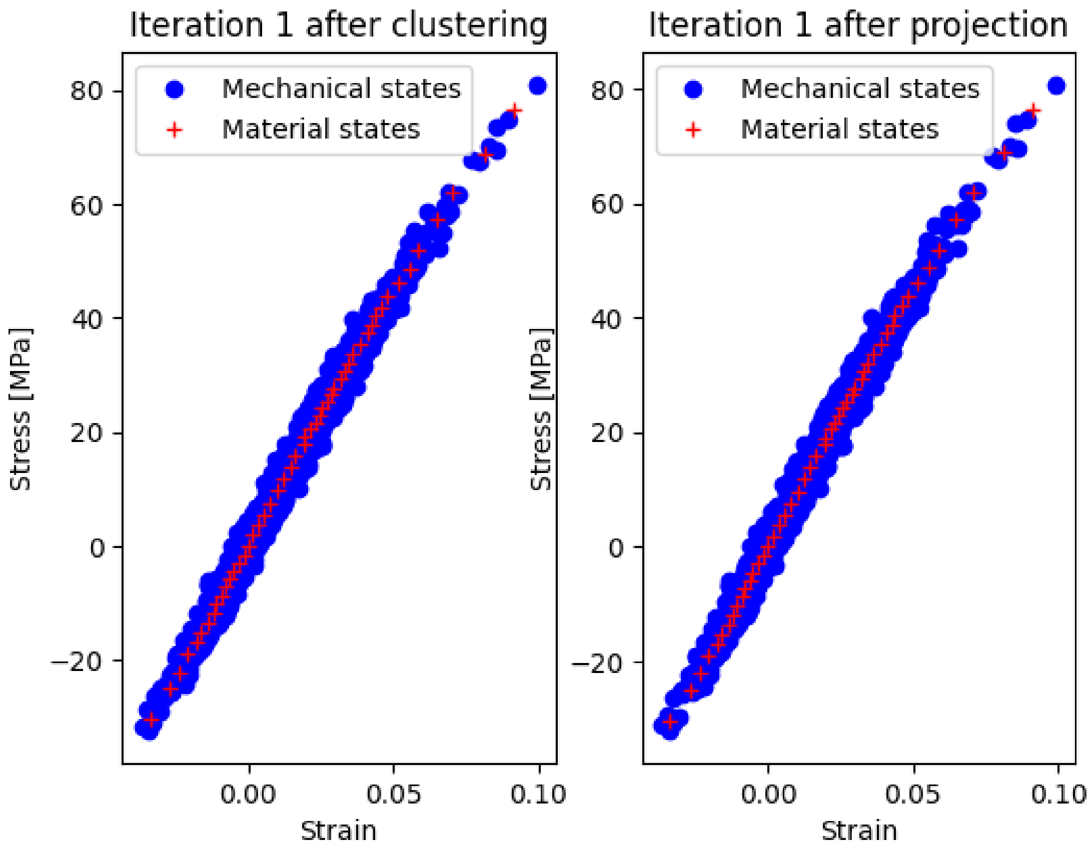
[6]:
```

# Initialisation
E_e=B.dot(U)
S_e=np.zeros(Nelems)
```

```
E_e=Eref+dE_e
#ff=plt.figure()
#plt.yscale('log')
#plt.ylabel('DDI norm')
#plt.xlabel('Number of iteration')
ic=0
for resampling in range(5):
    ## selection matrix from k-means
    samples=np.c_[np.squeeze(E_e*np.sqrt(Co)),np.squeeze(S_e/np.sqrt(Co))]
    kmeans = KMeans(Ns).fit(samples)
    #.reshape(-1,1))
    ie=kmeans.labels_
    val=np.ones(Nelems)
    ii=np.arange(Nelems)
    S=smatrix((val,(ii,ie)),shape=(Nelems,Ns))
    STS=S.T*S
    STS=STS.diagonal()

    ## Material states
    Estar=S.T.dot(E_e)/STS
    Sstar=S.T.dot(S_e)/STS
    ff=plt.figure()
    plt.subplot(121)
    plt.plot(E_e,S_e/pix2m*1.e-6,'bo',label='Mechanical states')
    plt.plot(Estar,Sstar/pix2m*1.e-6,'r+',label='Material states');
    plt.xlabel('Strain')
    plt.ylabel('Stress [MPa]');
    plt.title('Iteration %d after clustering' % (resampling))
    plt.legend();

    # Projection
    Estar_e=S.dot(Estar)
    Sstar_e=S.dot(Sstar)
    b=Dc.dot(Sstar_e)-Fext
    Lag=LU.solve(b)
    S_e=Sstar_e-iW*(Dc.T.dot(Lag))

    Estar_e=S.dot(Estar)
    Sstar_e=S.dot(Sstar)
    ddi_norm=0.5*(Co*np.dot(E_e-Estar_e,W.dot(E_e-Estar_e))+1/Co*np.
 ↪dot(S_e-Sstar_e,W.dot(S_e-Sstar_e)))
    print('***DDI loop Iteration %02d: DDI norm %6.3e ***' %␣
 ↪(resampling,ddi_norm))
    plt.subplot(122)
    plt.plot(E_e,S_e/pix2m*1.e-6,'bo',label='Mechanical states')
    plt.plot(Estar,Sstar/pix2m*1.e-6,'r+',label='Material states');
    plt.xlabel('Strain')
```

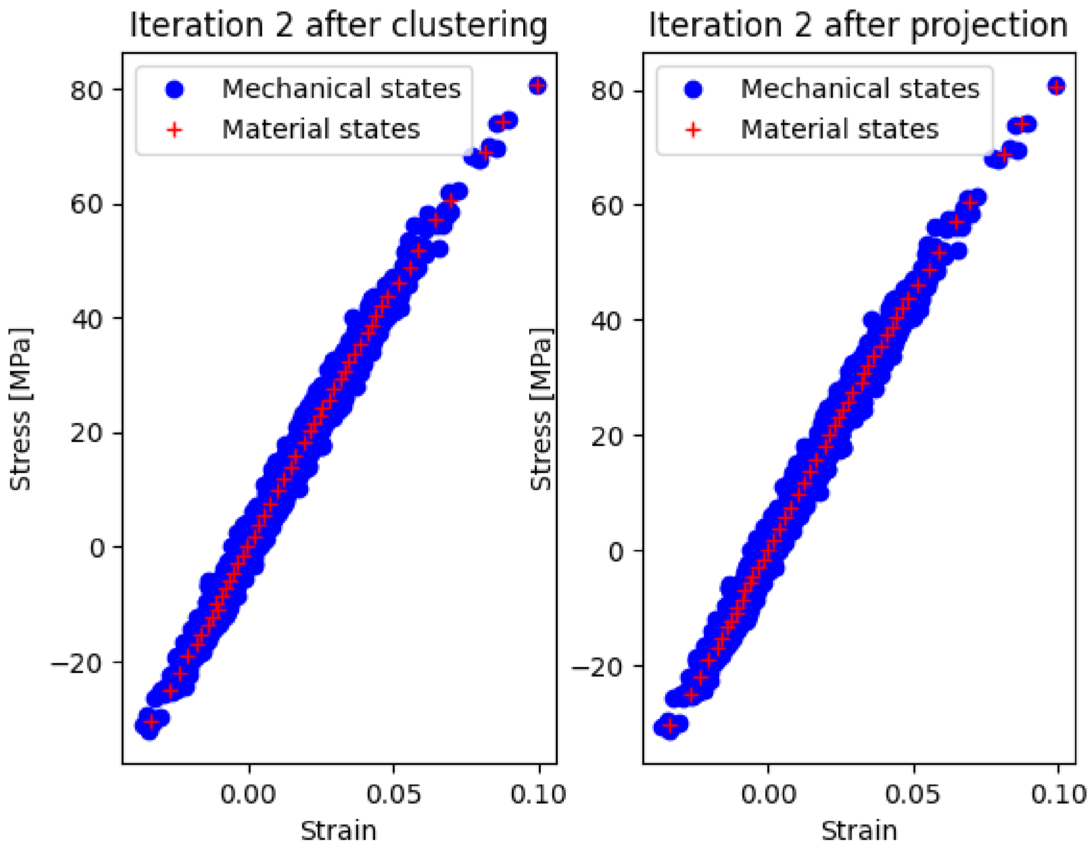
```
    plt.ylabel('Stress [MPa]');
    plt.title('Iteration %d after projection' % (resampling))
    plt.legend();
```

```
***DDI loop Iteration 00: DDI norm 2.619e+12 ***
***DDI loop Iteration 01: DDI norm 9.214e+08 ***
***DDI loop Iteration 02: DDI norm 7.138e+08 ***
***DDI loop Iteration 03: DDI norm 6.544e+08 ***
***DDI loop Iteration 04: DDI norm 6.437e+08 ***
```

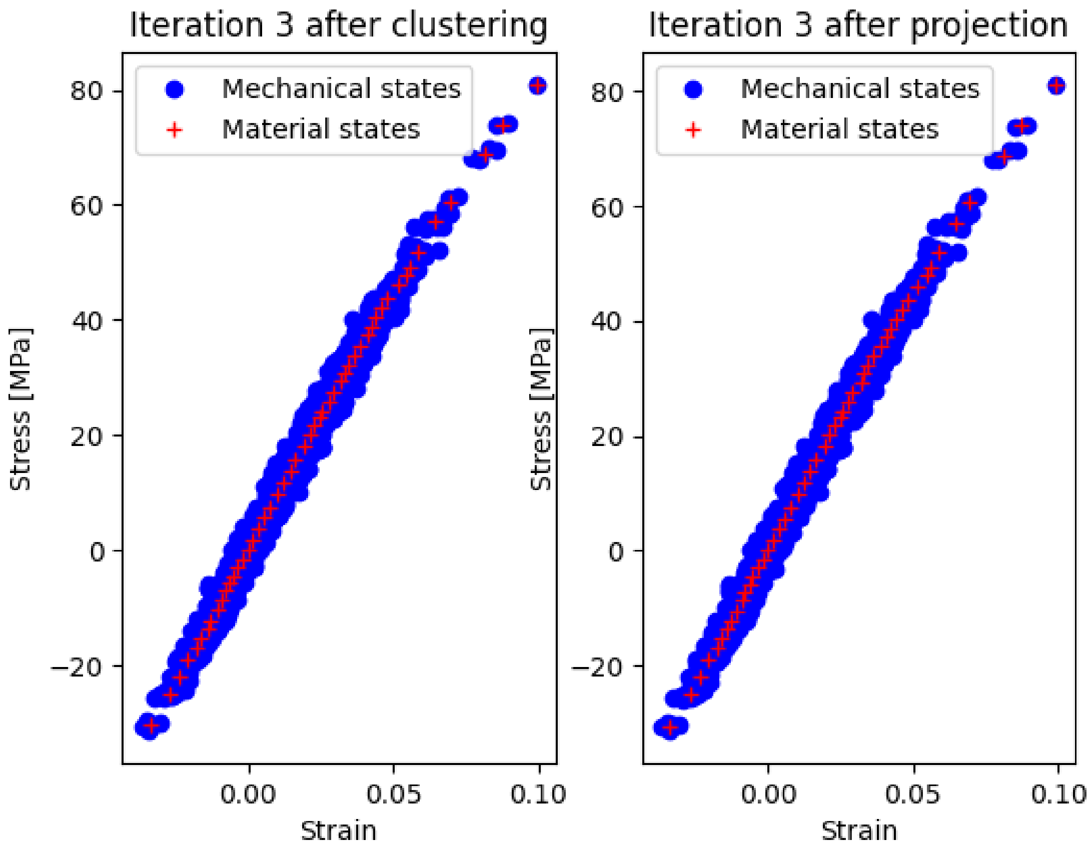




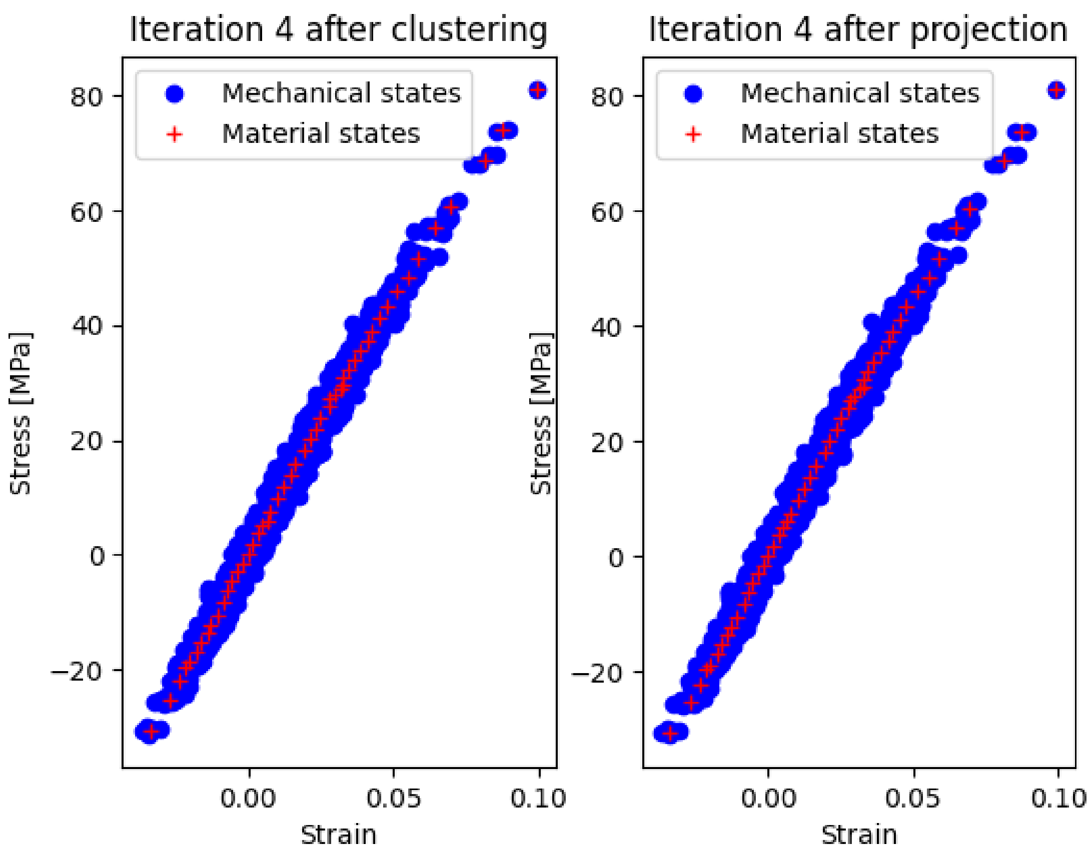
Iteration 1 after clustering
Mechanical states
Material states
Stress [MPa]
Strain
80
60
40
20
0
−20
0.00
0.05
0.10
Iteration 1 after projection
Mechanical states
Material states
Stress [MPa]
Strain
80
60
40
20
0
−20
0.00
0.05
0.10



Iteration 2 after clustering
Iteration 2 after projection
Mechanical states
Material states
Stress [MPa]
Strain
80
60
40
20
0
−20
0.00
0.05
0.10

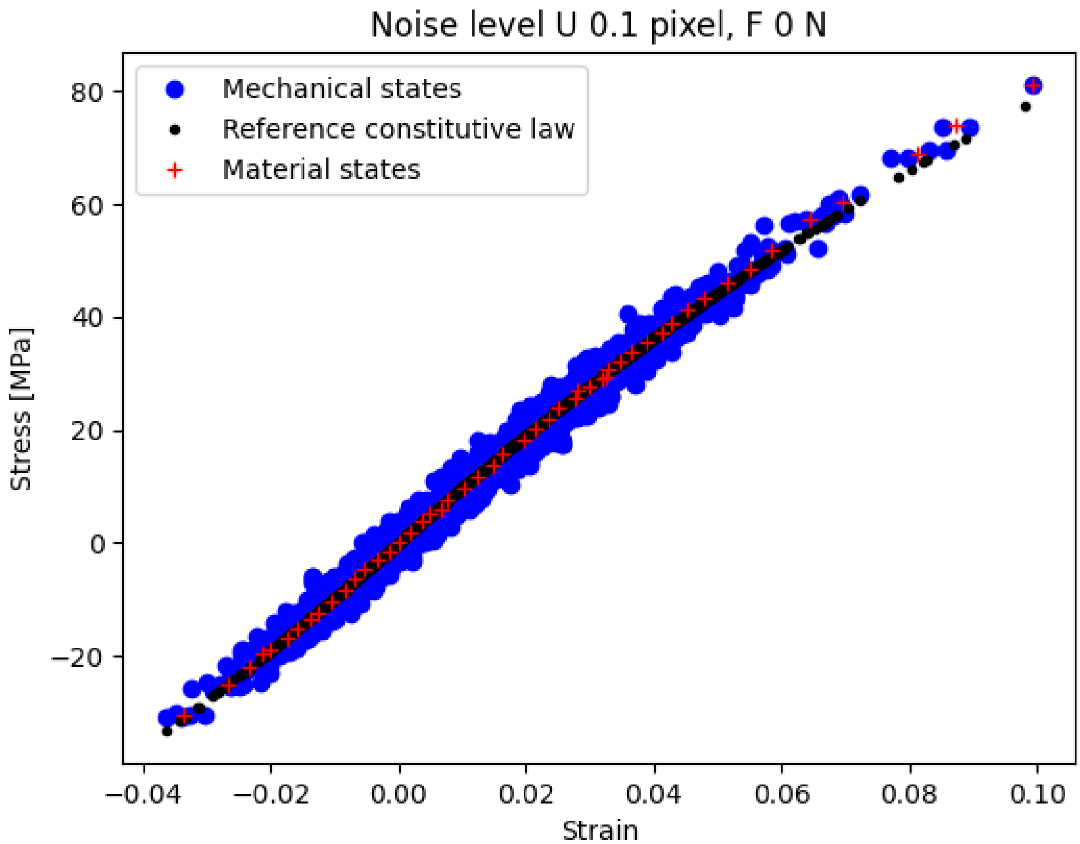


Iteration 3 after clustering
Mechanical states
Material states
Stress [MPa]
Strain
Iteration 3 after projection
Mechanical states
Material states
Stress [MPa]
Strain



2 Comparison with refernece

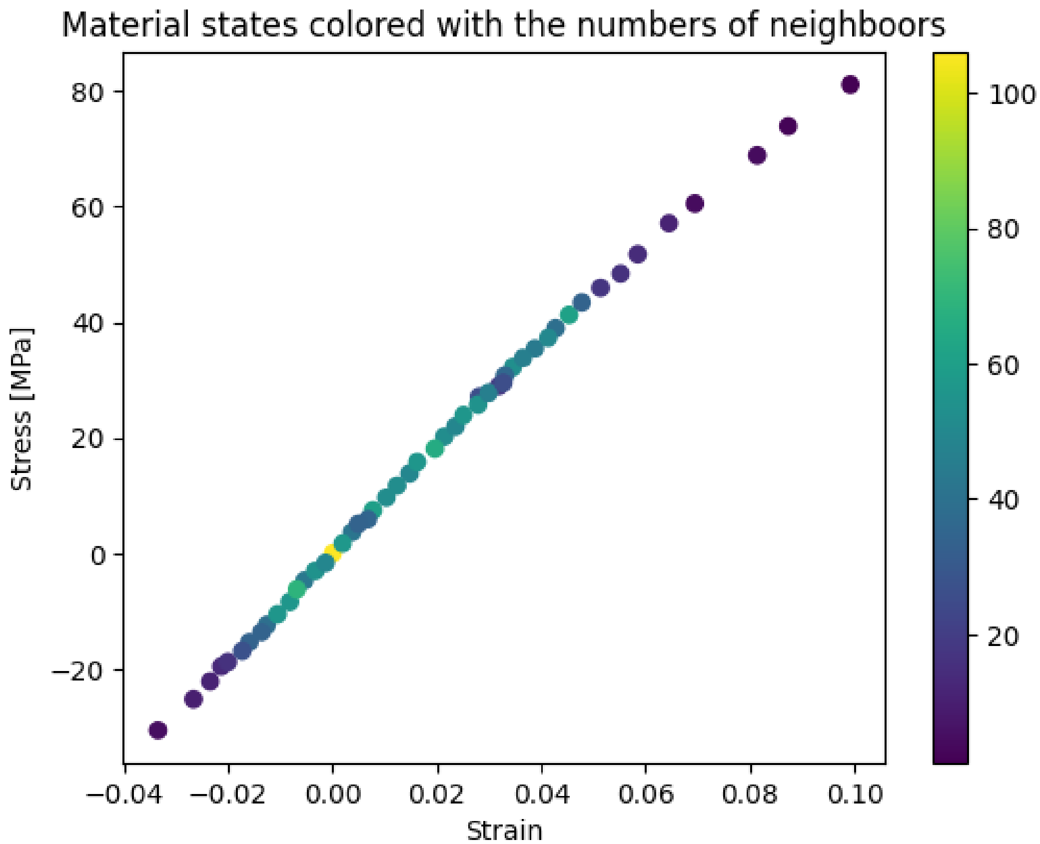
[7]:
```
ff=plt.figure()
plt.plot(E_e,S_e/pix2m*1.e-6,'bo',label='Mechanical states')
plt.plot(Eref,Sref/pix2m*1.e-6,'k.',label='Reference constitutive law');
plt.plot(Estar,Sstar/pix2m*1.e-6,'r+',label='Material states');
plt.xlabel('Strain')
plt.ylabel('Stress [MPa]');
plt.title('Noise level U %g pixel, F %g N' % (stdu,stdf))
plt.legend();
```

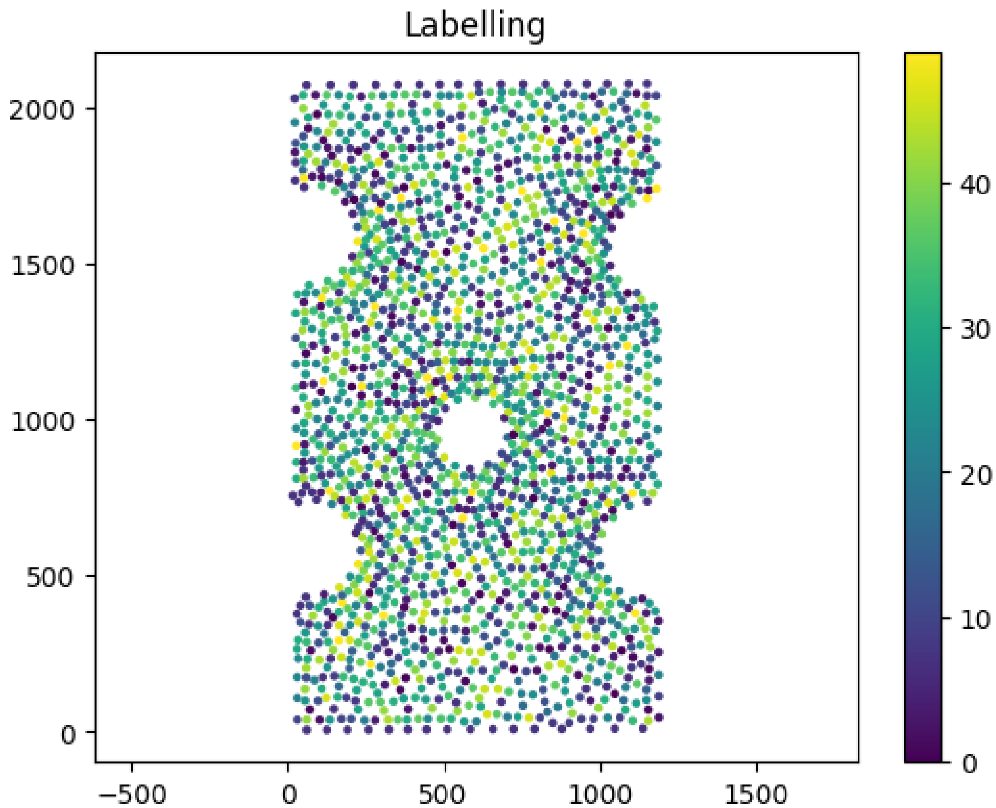


2.1 Illustrations

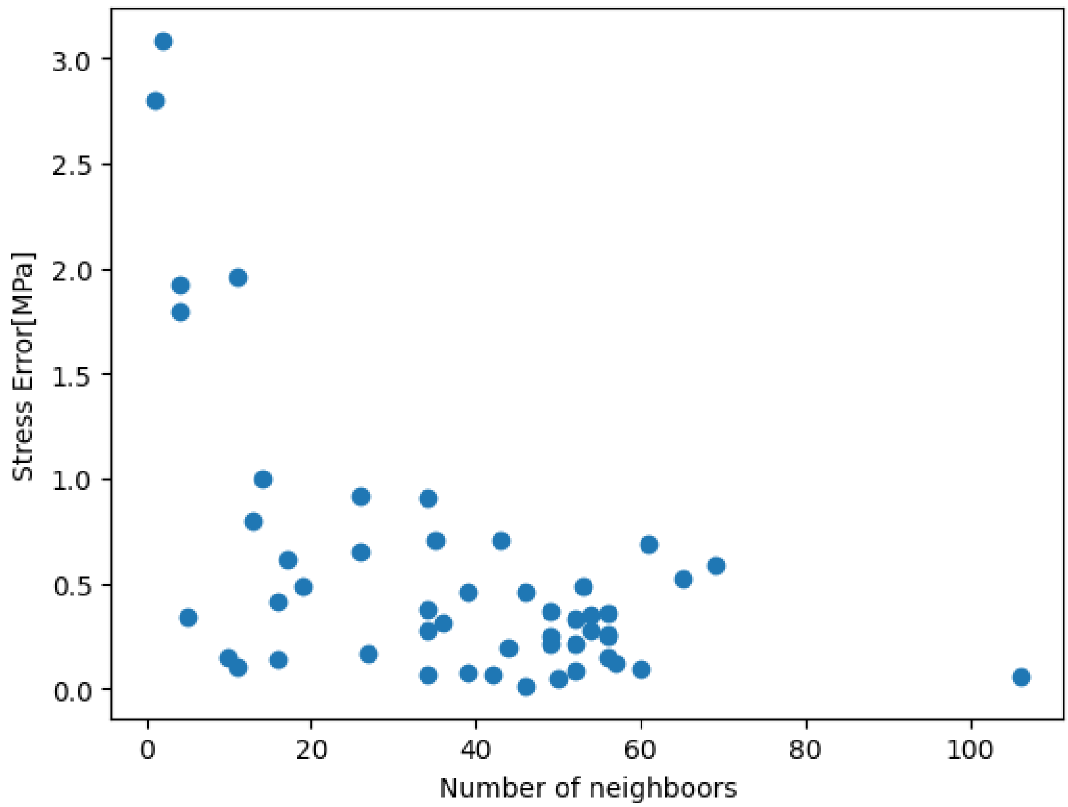
[8]:
```
STS=S.T*S
iSTS=STS.diagonal()
ff=plt.figure()
plt.scatter(Estar,Sstar/pix2m*1.e-6,c=iSTS);
plt.xlabel('Strain')
plt.ylabel('Stress [MPa]');
plt.title('Material states colored with the numbers of neighboors')
plt.colorbar();


Xg=0.5*(X[conn[:,0]]+X[conn[:,1]])
ff=plt.figure();
plt.scatter(Xg[:,0],Xg[:,1],c=ie,s=5);
plt.colorbar();
plt.axis('equal');
plt.title('Labelling')
```

[8]:
```
Text(0.5, 1.0, 'Labelling')
```


Material states colored with the numbers of neighboors
Stress [MPa]
Strain
80
60
40
20
0
−20
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
0.10
100
80
60
40
20



[9]:
```
Sref=2e8*(1-np.exp(-np.abs(Estar)/0.2))*np.sign(Estar)*pix2m
plt.scatter(iSTS,np.abs(Sref-Sstar)/pix2m*1.e-6);
plt.xlabel('Number of neighboors')
plt.ylabel('Stress Error[MPa]');
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
Stress Error[MPa]
0
20
40
60
80
100
Number of neighboors

```
[ ]:
```

Class's references i

- Claire, D., Hild, F., and Roux, S. (2004). **A finite element formulation to identify damage fields: The equilibrium gap method.** International Journal for Numerical Methods in Engineering, 61:189–208.
- Dalémat, M., Coret, M., Leygue, A., and Verron, E. (2019). **Measuring stress field without constitutive equation.** Mechanics of Materials, 136:103087.
- Grédiac, M., Pierron, F., Avril, S., and Toussaint, E. (2006). **The virtual fields method for extracting constitutive parameters from full-field measurements: a review.** Strain, 42(4):233–253.
- Kirchdoerfer, T. and Ortiz, M. (2016). **Data-driven computational mechanics.** Computer Methods in Applied Mechanics and Engineering, 304:81–101.
- Langlois, R., Coret, M., and Réthoré, J. (2022). **Non-parametric stress field estimation for history-dependent materials: Application to ductile material exhibiting piobert–lüders localization bands.** Strain, page e12410.
- Leclerc, H., Perie, J., Roux, S., and Hild, F. (2009). **Computer Vision/Computer Graphics CollaborationTechniques, chapter Integrated Digital Image Correlation for the Identification of Mechanical Properties.** Springer, Berlin.

Class's references ii

Lecompte, D., Smits, A., Sol, H., Vantomme, J., and Van Hemelrijck, D. (2007). **Mixed numerical–experimental technique for orthotropic parameter identification using biaxial tensile tests on cruciform specimens.** International Journal of Solids and Structures, 44(5):1643–1656.

Leygue, A., Coret, M., Réthoré, J., Stainier, L., and Verron, E. (2018). **Data-based derivation of material response.** Computer Methods in Applied Mechanics and Engineering, 331:184–196.

Leygue, A., Seghir, R., Réthoré, J., Coret, M., Verron, E., and Stainier, L. (2019). **Non-parametric material state field extraction from full field measurements.** Computational Mechanics, 64(2):501–509.