MTEX

an open source texture analysis toolbox

Ralf Hielscher and Florian Bachmann

TU Chemnitz, TU Freiberg, Germany

ICOTOM, Dresden 2014

Chemnitz MTEX Workshop 2015

Scope:	bring together MTEX users to share their ideas and applications
Tutorial:	what's new in MTEX 4.0?
Invited Speakers:	D. Mainprice, K. Kunze, G. Nolze

Date:	9th to 13th of February 2015
Location:	Chemnitz University of Technology, Germany
Organizer:	R. Hielscher

What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

```
MTEX 4.0.beta1 (show documentation)
Import pole figure data
Import EBSD data
Import ODF data

Uninstall MTEX

>> s = sin([0 30 60]*degree)

s =

         0    0.5000    0.8660

>>
```

What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

why scripts?

- *reproducible results*
- *templates for common tasks*
- *extensively customizable*
- *batch processing of many data sets*
- *repeated calculations with different parameters*

What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

```
ebsd = loadEBSD('mylonite.txt') % load data

grains = calcGrains(ebsd)       % reconstruct grains

[m,index] = max(grains.area)    % find largest grain

plot(grains(index).boundary)    % plot largest grain
```

What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

```
ebsd = loadEBSD('mylonite.txt')
```

```
ebsd = EBSD (show methods, plot)

  Properties: x, y
  Phase Orientations     Mineral Symmetry Crystal reference frame
      1          3444    Andesina       -1                X||a*, Z||c
      2          3893      Quartz      -3m      X||a*, Y||b,  Z||c*
      3           368     Biotite      2/m      X||a*, Y||b*, Z||c
      4          4781  Orthoclase      2/m      X||a*, Y||b*, Z||c
```

What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

```
ebsd = loadEBSD('mylonite.txt')
grains = calcGrains(ebsd)
```

```
grains = grain2d (show methods, plot)

  Phase  Grains     Mineral  Symmetry  Cryst. reference frame
      1    1951    Andesina        -1              X||a*, Z||c
      2     776      Quartz       -3m    X||a*, Y||b,  Z||c*
      3     305     Biotite       2/m    X||a*, Y||b*, Z||c
      4    1641  Orthoclase       2/m    X||a*, Y||b*, Z||c
```

What is MTEX?

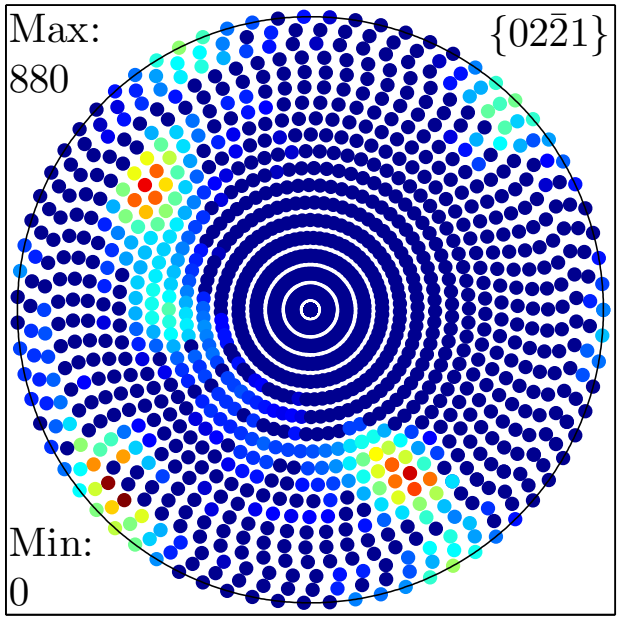
1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

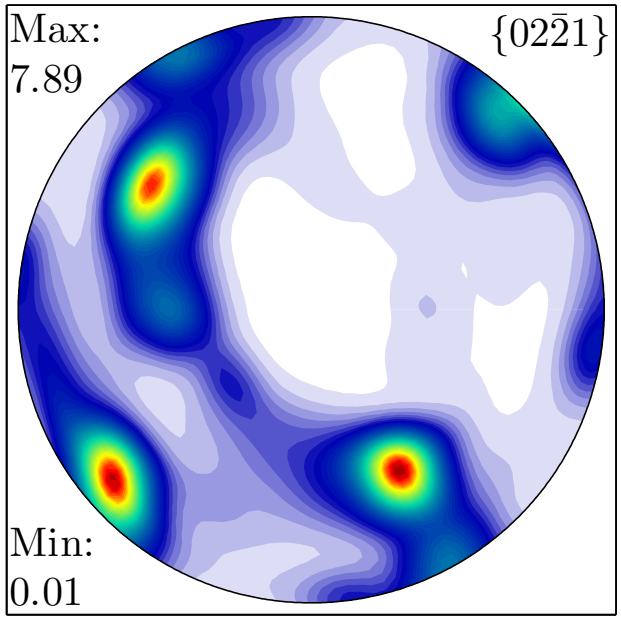
```
ebsd = loadEBSD('mylonite.txt')
grains = calcGrains(ebsd)
[m,index] = max(grains.area)
```

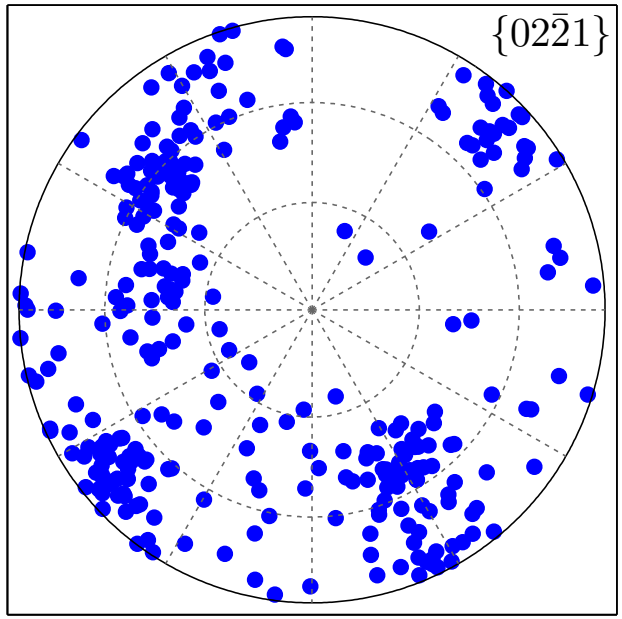
```
m =
   1.4985e+06
index =
        1795
```

What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify







What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

- *7 years of development*
- *1000 functions*
- *40 000 lines of code, 33 percent comment lines*
- *14 reference paper, about 200 references*
- *1500 downloads per version*
- *1000 help pages*
- *compatible toolboxes: MSAT, PolyLx*

What is MTEX?

1. a MATLAB toolbox for quantitative texture analysis
2. a scripting language
3. a tool for generating publication ready plots
4. large, well documented and exhaustively tested
5. free to use, to extend, to modify

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
ori1 = orientation('Euler',10*degree,20*degree,0,...
       loadCIF('quartz'))
```

```
ori1 = orientation (show methods, plot)
  size: 1 x 1
  crystal symmetry: Quartz (P 32 2 1, X||a*, Y||b, Z||c*)
  sample symmetry : triclinic

  Bunge Euler angles in degree
  phi1   Phi  phi2
    10    20     0
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
ori1 = orientation('Euler',10*degree,20*degree,0,...
       loadCIF('quartz'))
```

```
r = ori1 * Miller(1,1,-2,0,symmetry('quartz'))
```

```
r = vector3d (show methods, plot)
  size: 1 x 1
          x         y         z
  0.305126  0.176165  0.203417
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
ori2 = orientation('Euler',50*degree,0,20*degree,...
       loadCIF('olivin'))
mori = inv(ori1) * ori2
```

```
mori = misorientation (show methods, plot)
  size: 1 x 1
  crystal symmetry: IRON TETRATHIOSILICATE (P n m a)
  crystal symmetry: Quartz (P 32 2 1, X||a*, Y||b, Z||c*)

  Bunge Euler angles in degree
  phi1  Phi phi2
    60   90   40
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
odf = 0.8 * unimodalODF(ori1) + ...
      0.2 * uniformODF(symmetry('quartz'))
```

```
odf = ODF (show methods, plot)
  crystal symmetry: Quartz (P 32 2 1, X||a*, Y||b, Z||c*)
  sample symmetry : triclinic

  Radially symmetric portion:
    kernel: de la Vallee Poussin, hw = 10
    center: (30  ,90  ,0  )
    weight: 0.8

  Uniform portion:
    weight: 0.2
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

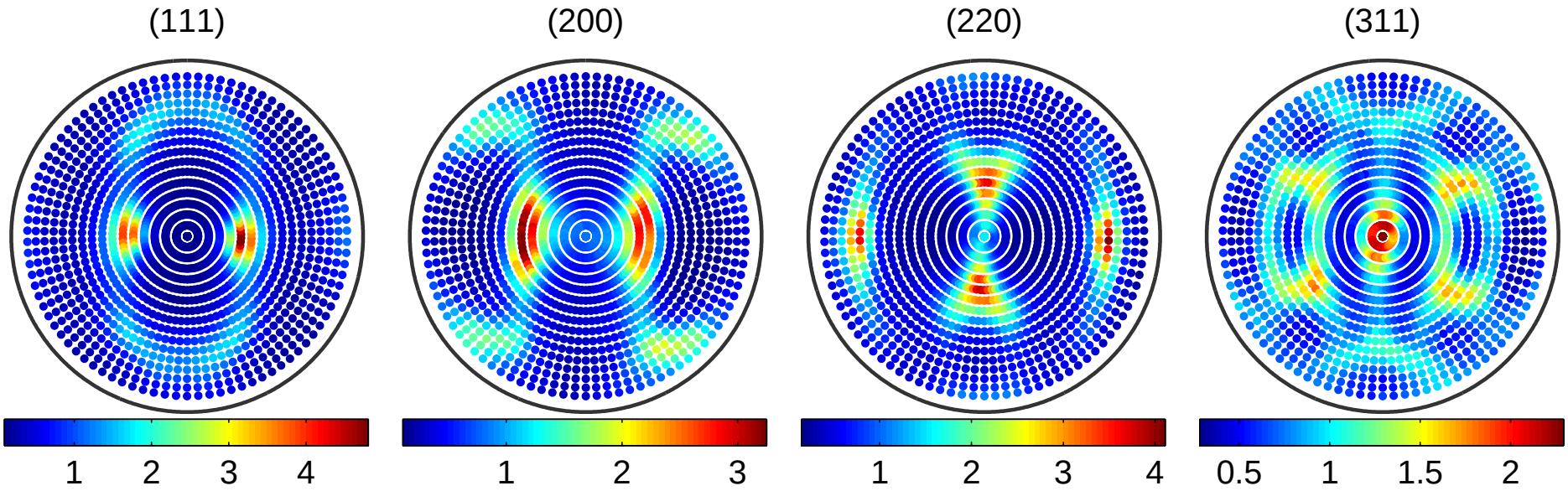
```
pf = loadPoleFigure('Queens_alu.plf')
```

```
pf = PoleFigure (show methods, plot)
  file name: Queens_alu.plf
  crystal symmetry : cubic (m-3m)
  specimen symmetry: orthorhombic

  h = {111}, r = 90x17 points
  h = {200}, r = 90x17 points
  h = {220}, r = 90x17 points
  h = {311}, r = 90x17 points
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
pf = loadPoleFigure('Queens_alu.plf')
plot(pf)
```



Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

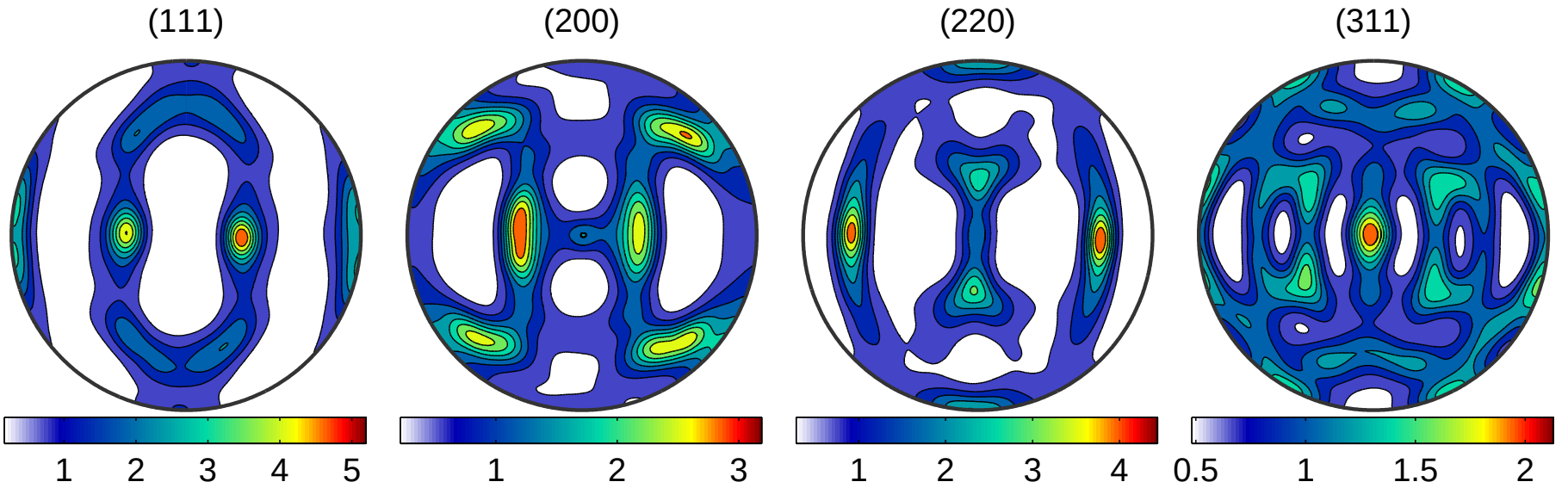
```
odf = calcODF(pf)
```

```
odf = ODF (show methods, plot)
  file name: Queens_alu.plf
  crystal symmetry: cubic
  sample symmetry : orthorhombic

  Uniform portion:
    weight: 0.14633

  Radially symmetric portion:
    kernel: de la Vallee Poussin, hw = 4
    center: 2444 orientations, resolution: 3.9
    weight: 0.85367
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
odf = calcODF(pf)
plotPDF(odf,pf.h)
```



Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

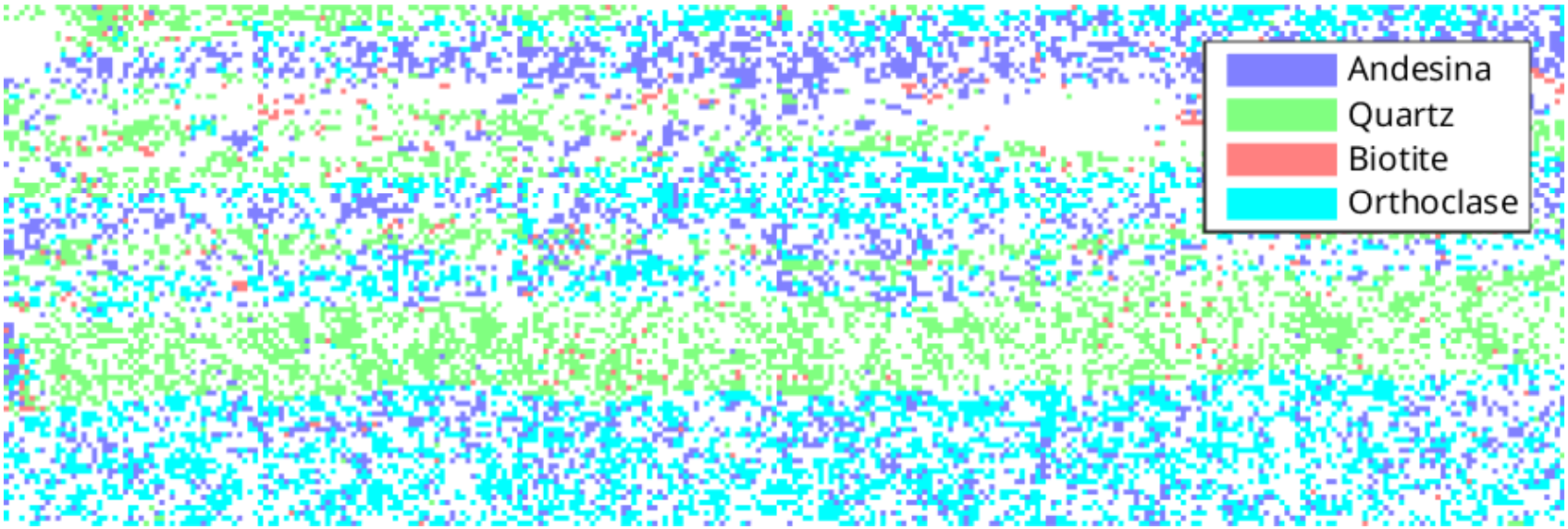
```
ebsd = loadEBSD('mylonite.txt')
```

```
ebsd = EBSD (show methods, plot)
  file name: mylonite.txt
  Properties: x, y
  Phase Orientations     Mineral Symmetry Crystal reference frame
      1          3444   Andesina       -1                X||a*, Z||c
      2          3893     Quartz      -3m      X||a*, Y||b,  Z||c*
      3           368    Biotite      2/m      X||a*, Y||b*, Z||c
      4          4781 Orthoclase      2/m      X||a*, Y||b*, Z||c
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
ebsd = loadEBSD('mylonite.txt')
plot(ebsd)
```



Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

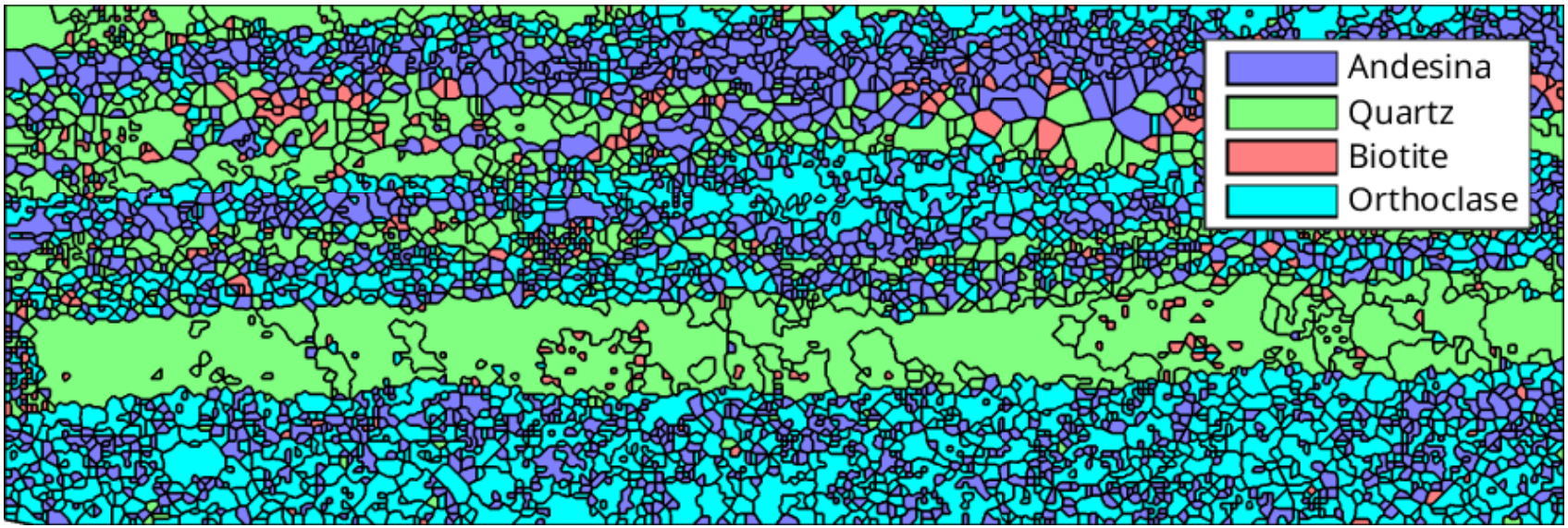
```
grains = calcGrains(ebsd)
```

```
grains = grain2d (show methods, plot)

  Phase  Grains     Mineral  Symmetry  Cryst. reference frame
      1    1951    Andesina        -1              X||a*, Z||c
      2     776      Quartz       -3m    X||a*, Y||b,  Z||c*
      3     305     Biotite       2/m    X||a*, Y||b*, Z||c
      4    1641  Orthoclase       2/m    X||a*, Y||b*, Z||c
```

Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
grains = calcGrains(ebsd)
plot(grains)
```



Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

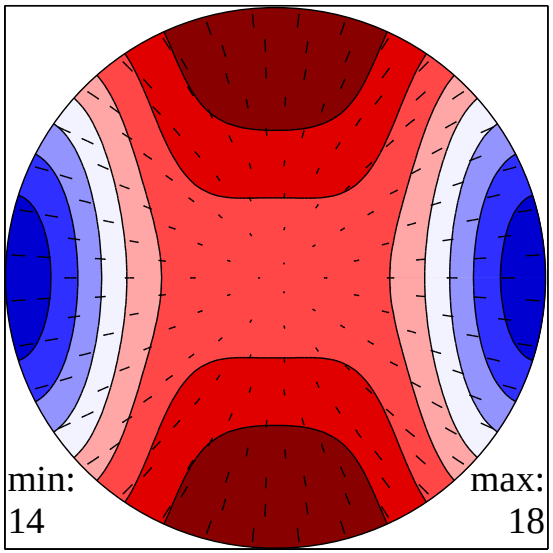
```
cs = symmetry('Olivin');
C  = loadTensor('Olivine1997PC.GPa',cs,...
     'propertyname','elastic stiffness','unit','Pa')
```

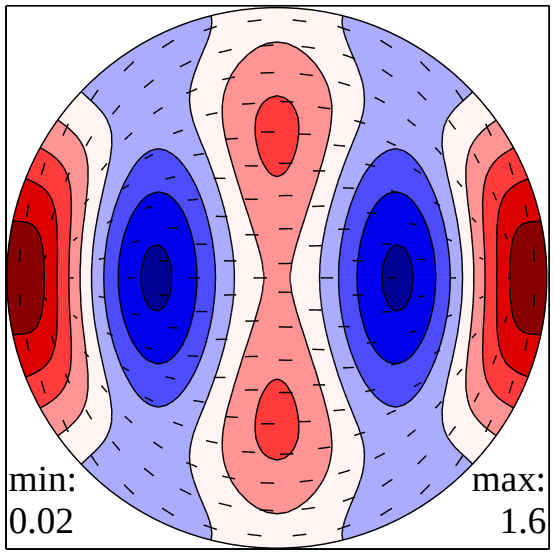
```
C = elastic stiffness tensor
  unit          : Pa
  mineral       : Olivin (mmm)

 320.5   68.2   71.6      0      0      0
  68.2  196.5   76.8      0      0      0
  71.6   76.8  233.5      0      0      0
     0      0      0     64      0      0
     0      0      0      0     77      0
     0      0      0      0      0   78.7
```

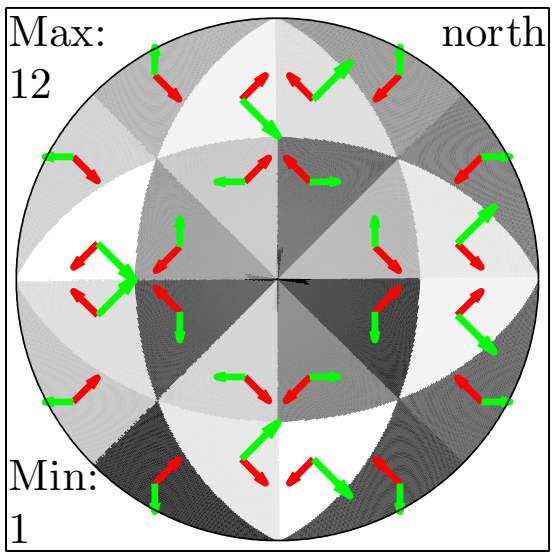
Feature Overview

- crystal geometry
- odf modeling
- pole figure measurements
- individual orientation measurements
- elastic and plastic deformations

```
plot(C,'PlotType','velocity','vp','density')
plot(C,'PlotType','velocity','pp','density')
```







Benefits / Challenges of Big Projects

- everything (needs to) works with everything
- less code duplication - bugs have greater impact
- additional features are simple / difficult to include
- you can (must) plot everything with the same engine

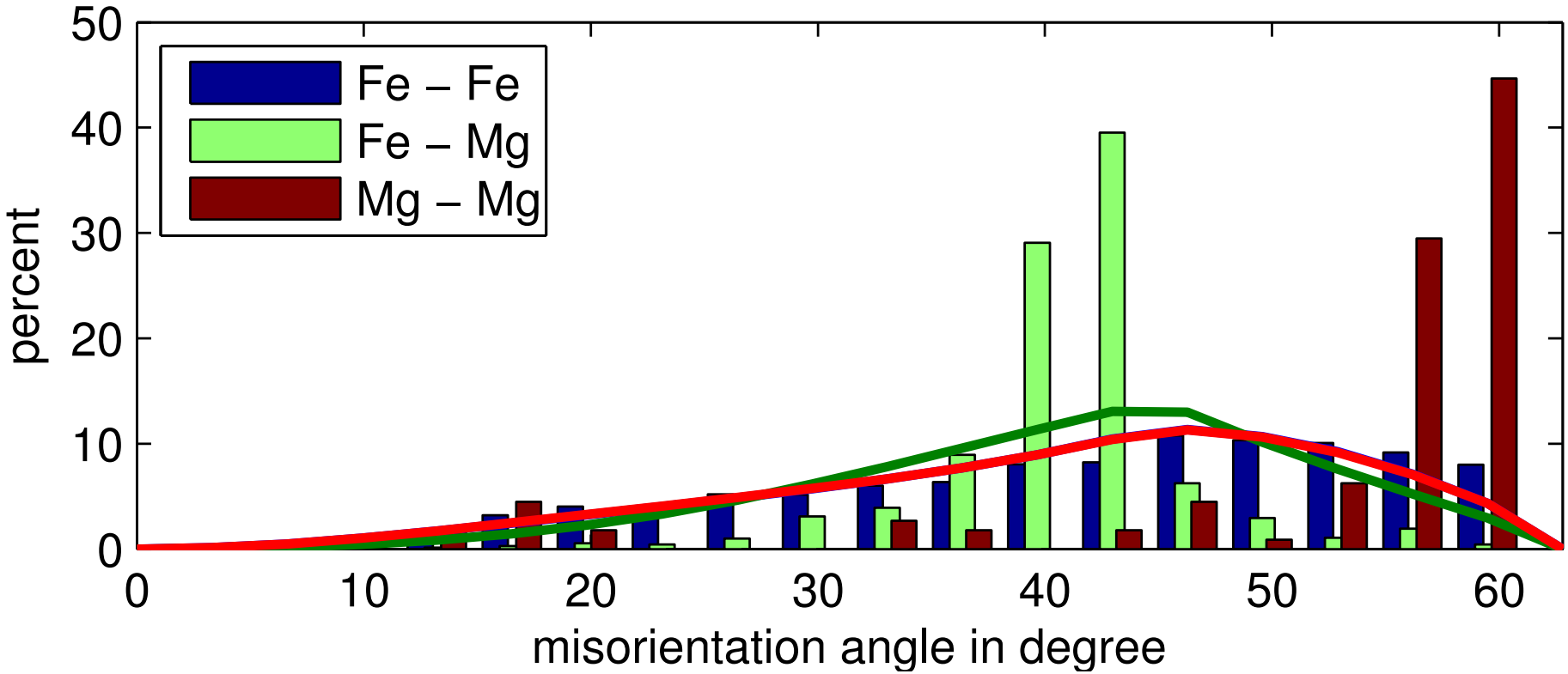
Benefits / Challenges of Big Projects

- everything (needs to) works with everything
- less code duplication - bugs have greater impact
- additional features are simple / difficult to include
- you can (must) plot everything with the same engine

```
density = calcAngleDistribution(ebsd)
density = calcAngleDistribution(grains)
density = calcAngleDistribution(sym)
density = calcAngleDistribution(mdf)
```

Benefits / Challenges of Big Projects

- everything (needs to) works with everything
- less code duplication - bugs have greater impact
- additional features are simple / difficult to include
- you can (must) plot everything with the same engine



Benefits / Challenges of Big Projects

- everything (needs to) works with everything
- less code duplication - bugs have greater impact
- additional features are simple / difficult to include
- you can (must) plot everything with the same engine

```
function nu = PoissonRatio(C,x,y)
% compute Poisson ration transverse x and axial y

S = inv(C); % compute the complience

% compute lateral and longitudinal strain
a = EinsteinSum(S,[-1 -2 -3 -4],x,-1,x,-2,y,-3,y,-4)
b = EinsteinSum(S,[-1 -2 -3 -4],x,-1,x,-2,x,-3,x,-4)

% negative quotient
nu = -double(a) ./ double(b);
```

Benefits / Challenges of Big Projects

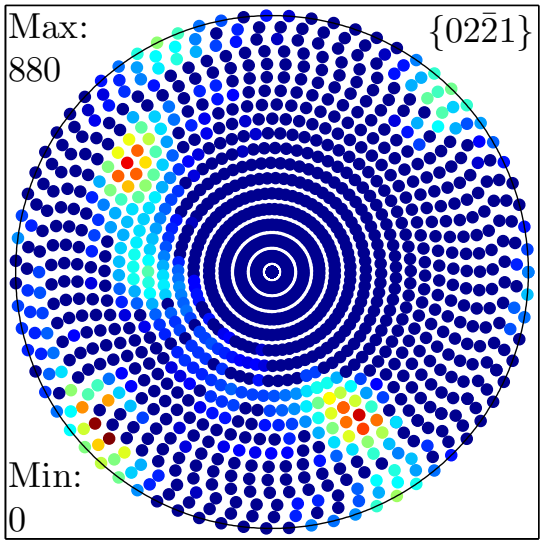
- everything (needs to) works with everything
- less code duplication - bugs have greater impact
- additional features are simple / difficult to include
- you can (must) plot everything with the same engine

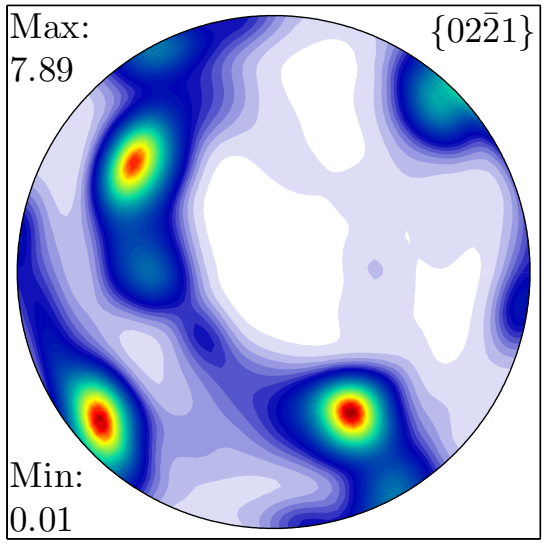
```
function nu = PoissonRatio(C,x,y)
% compute Poisson ration transverse x and axial y

S = inv(C); % compute the complience

% compute lateral and longitudinal strain
a = EinsteinSum(S,[-1 -2 -3 -4],x,-1,x,-2,y,-3,y,-4)
b = EinsteinSum(S,[-1 -2 -3 -4],x,-1,x,-2,x,-3,x,-4)

% negative quotient
nu = -double(a) ./ double(b);
```

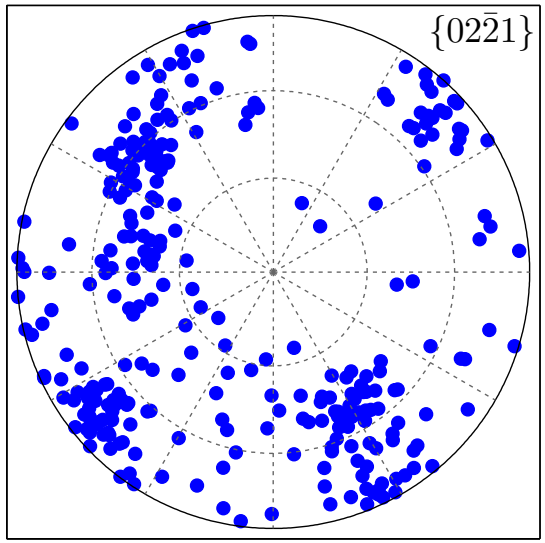
Benefits / Challenges of Big Projects

- everything (needs to) works with everything
- less code duplication - bugs have greater impact
- additional features are simple / difficult to include
- you can (must) plot everything with the same engine

```
plot(pf)
plotpdf(odf,Miller(0,2,-2,1))
plotpdf(ebsd,Miller(0,2,-2,1))
```







Design Principle - Everything is a Vector

```
[grains,ebsd] = calcGrains(ebsd)
```

```
grains = grain2d (show methods, plot)

  Phase  Grains     Mineral  Symmetry  Crystal reference frame
      1    1951    Andesina        -1                X||a*, Z||c
      2     776      Quartz        -3m       X||a*, Y||b,  Z||c*
      3     305     Biotite       2/m        X||a*, Y||b*, Z||c
      4    1641  Orthoclase       2/m        X||a*, Y||b*, Z||c
```

Design Principle - Everything is a Vector

```
[grains,ebsd] = calcGrains(ebsd)
```

```
grains(1795)
```

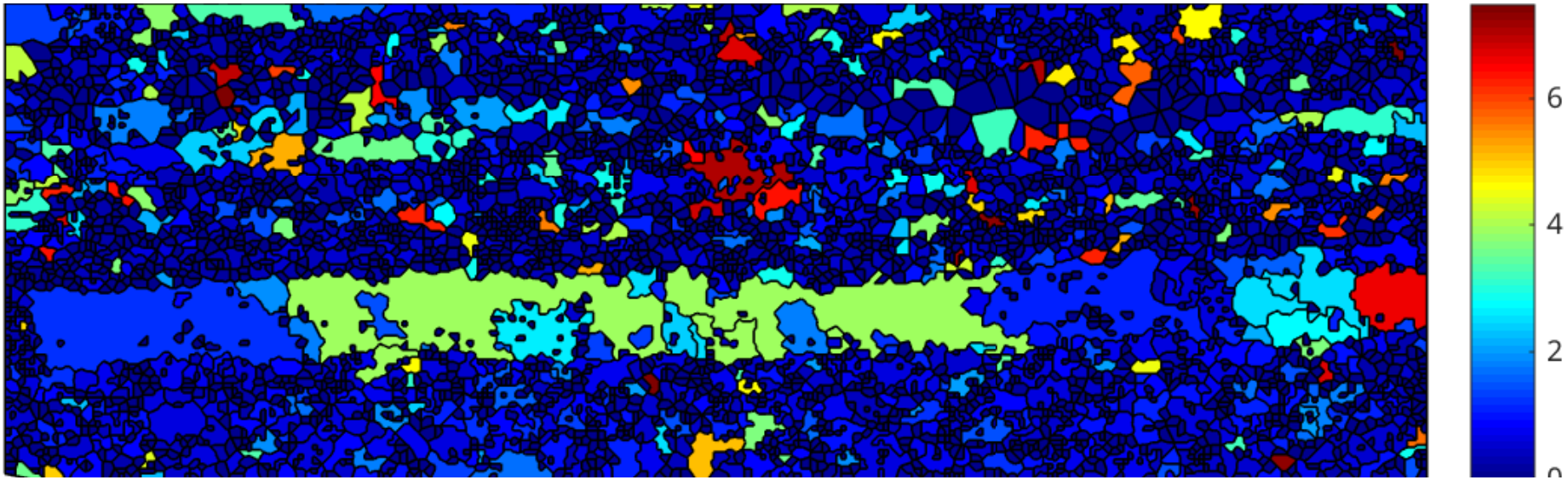
```
ans = grain2d (show methods, plot)

  Phase Grains Mineral Symmetry Crystal reference frame
      2      1  Quartz      -3m      X||a*, Y||b, Z||c*
```

Design Principle - Everything is a Vector

```
[grains,ebsd] = calcGrains(ebsd)
```

```
grains(1795)
```

```
for i = 1:length(grains)
  mori = ebsd(grains(i)).mis2mean;
  oriSpread(i) = sqrt(mean(mori.angle.^2));
end
plot(grains,oriSpread ./ degree)
```



Design Principle - Everything is a Vector

```
[grains,ebsd] = calcGrains(ebsd)
```

```
grains(1795)
```

```
for i = 1:length(grains)
  mori = ebsd(grains(i)).mis2mean;
  oriSpread(i) = sqrt(mean(mori.angle.^2));
end
plot(grains,oriSpread ./ degree)
```

```
grains(oriSpread>5*degree)
```

```
ans = grain2d (show methods, plot)

  Phase  Grains     Mineral  Symmetry  Crystal reference frame
      1      14    Andesina        -1                X||a*, Z||c
      2      17      Quartz       -3m       X||a*, Y||b,  Z||c*
      3       1     Biotite       2/m       X||a*, Y||b*, Z||c
      4      12  Orthoclase       2/m       X||a*, Y||b*, Z||c
```

Design Principle - Everything is a Vector II

```
sigma = tensor([[0 0 0];[0 0 0];[0 0 1]],'stress')
```

```
sigma = stress tensor (show methods, plot)
  rank: 2 (3 x 3)

 0 0 0
 0 0 0
 0 0 1
```

Design Principle - Everything is a Vector II

```
sigma = tensor([[0 0 0];[0 0 0];[0 0 1]],'stress')
```

```
ori = grains('Quartz').meanOrientation;
```

```
ori = orientation (show methods, plot)
  size: 776 x 1
  crystal symmetry: Quartz (-3m, X||a*, Y||b, Z||c*)
  sample symmetry : triclinic
```

Design Principle - Everything is a Vector II

```
sigma = tensor([[0 0 0];[0 0 0];[0 0 1]],'stress')
```

```
ori = grains('Quartz').meanOrientation;
```

```
sigmaCS = rotate(sigma,inv(ori))
```

```
sigmaCS = stress tensor (show methods, plot)
  size    : 776 x 1
  rank    : 2 (3 x 3)
  mineral: Quartz (-3m, X||a*, Y||b, Z||c*)
```

Design Principle - Everything is a Vector II

```
sigma = tensor([[0 0 0];[0 0 0];[0 0 1]],'stress')
```

```
ori = grains('Quartz').meanOrientation;
```

```
sigmaCS = rotate(sigma,inv(ori))
```

```
m = Miller(0,0,0,1,loadCIF('quartz'))
n = Miller(1,1,-2,0,loadCIF('quartz'))
tauMax = calcShearStress(sigmaCS,m,n,'symmetrise')
```

Design Principle - Everything is a Vector II

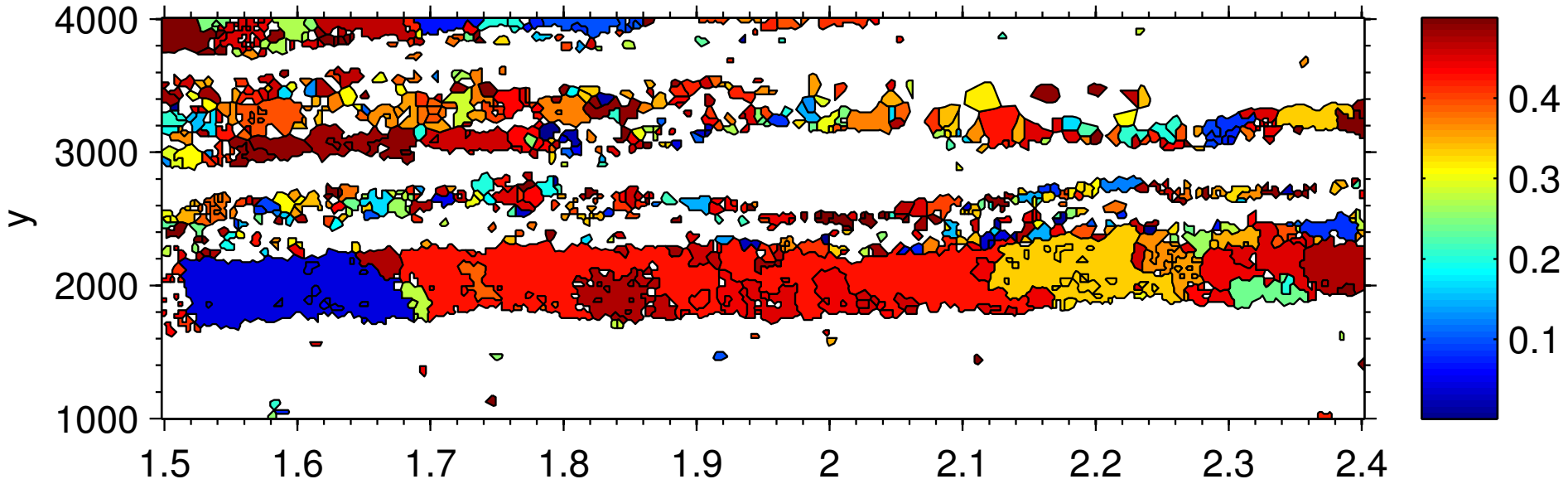
```
sigma = tensor([[0 0 0];[0 0 0];[0 0 1]],'stress')
```

```
ori = grains('Quartz').meanOrientation;
```

```
sigmaCS = rotate(sigma,inv(ori))
```

```
m = Miller(0,0,0,1,loadCIF('quartz'))
n = Miller(1,1,-2,0,loadCIF('quartz'))
tauMax = calcShearStress(sigmaCS,m,n,'symmetrise')
```

```
plot(grains('quartz'),tauMax)
```



Geometry data types