MTEX

an open source texture analysis toolbox

Ralf Hielscher

TU Chemnitz, Germany

Düsseldorf, 2014

An Appetizer

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

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

 Phase   Orientations   Mineral   Symmetry
     1   154107 (100)      iron       m-3m

 Properties: ci, error, iq, x, y
 Scan unit : um
```

An Appetizer

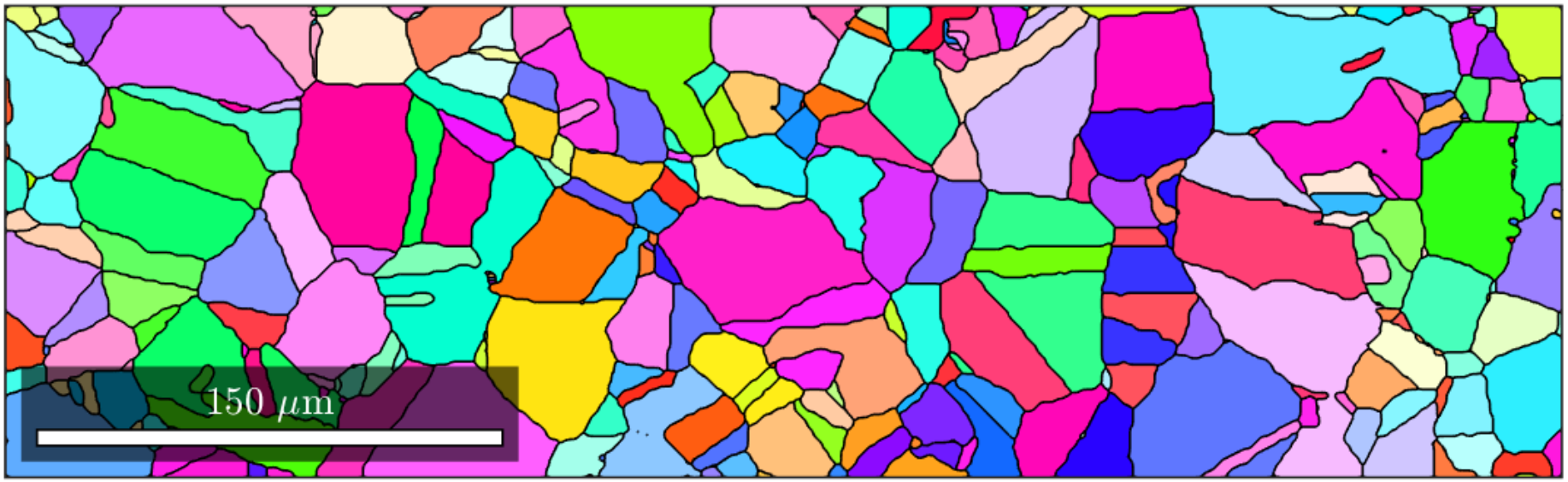
```
ebsd = loadEBSD('CSL.txt')
```

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

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

 Phase   Grains   Pixels   Mineral   Symmetry
    -1      465    75651      iron       m-3m

 Properties: GOS, meanRotation
```

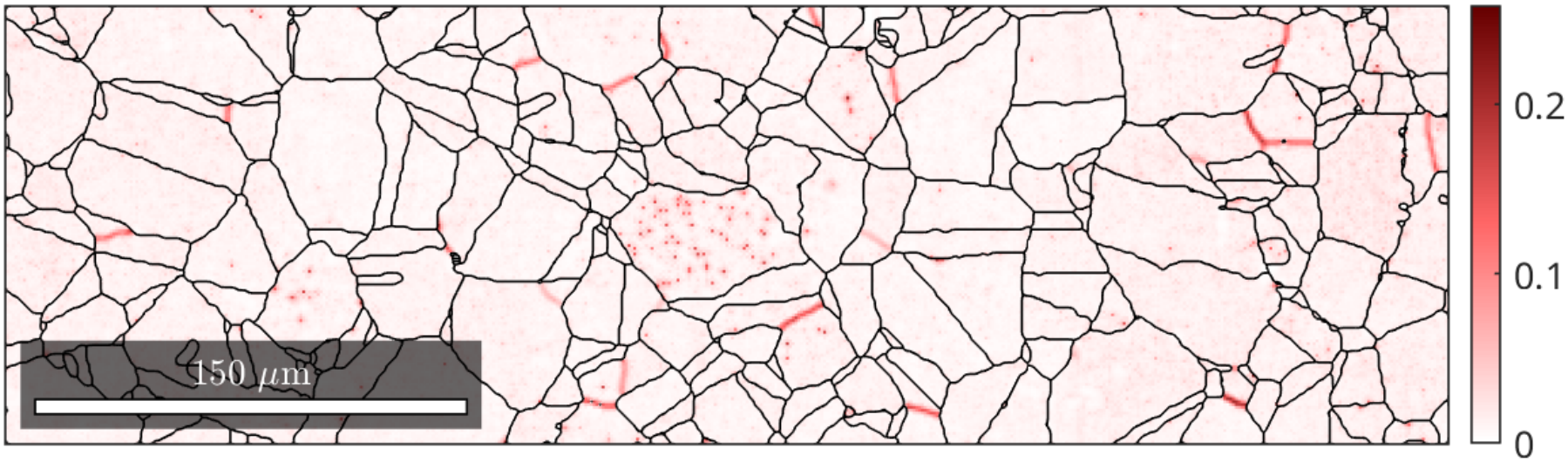


An Appetizer

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

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

```
plot(ebsd,ebsd.KAM)
hold on
plot(grains.boundary)
hold off
```



An Appetizer

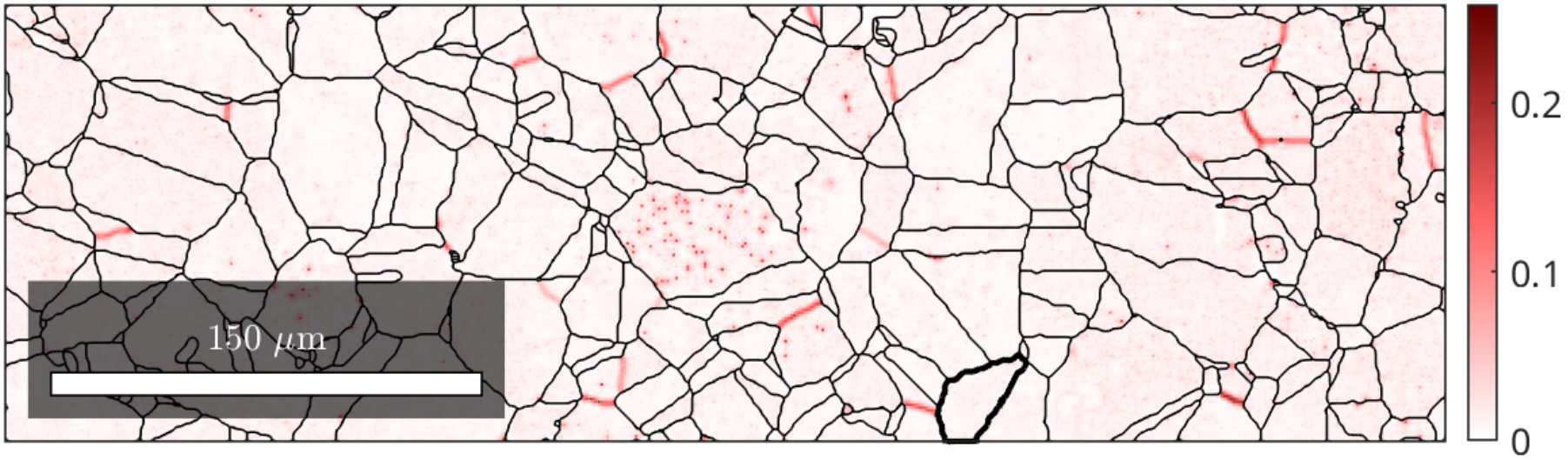
```
ebsd = loadEBSD('CSL.txt')
```

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

```
grains(90).boundary
```

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

 Segments     mineral 1   mineral 2
       11   not indexed        iron
      111          iron        iron
```



An Appetizer

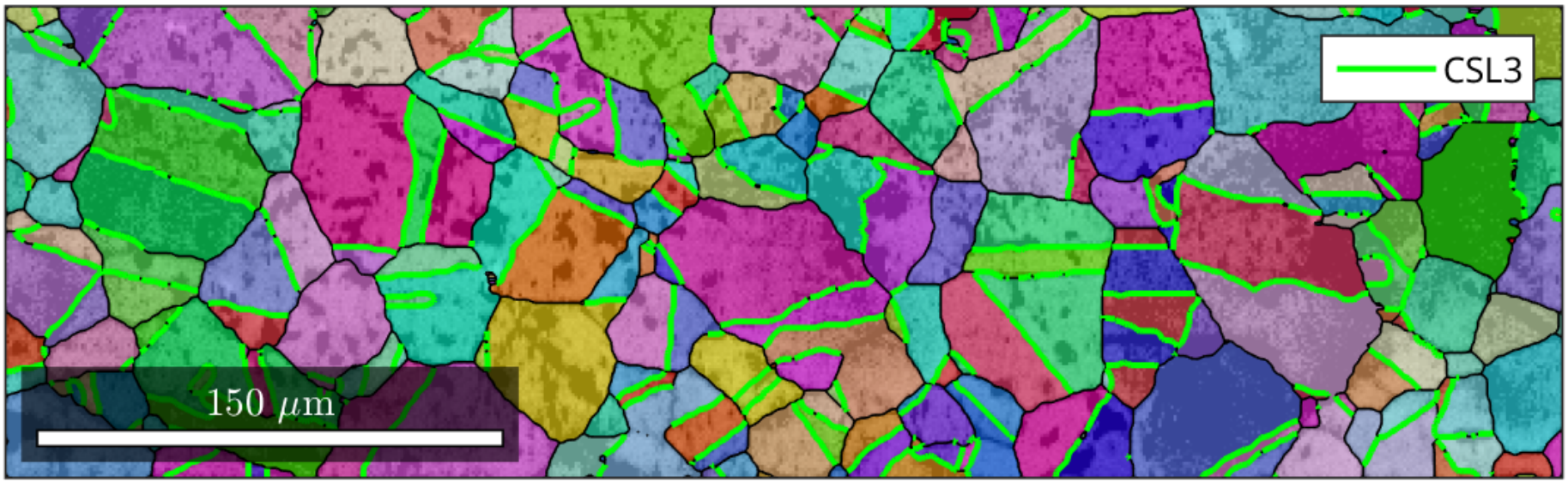
```
ebsd = loadEBSD('CSL.txt')
```

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

```
gB = grains.boundary('indexed');
gB3 = gB(angle(gB.misorientation,CSL(3)) < 3*degree)
```

```
gB3 = grainBoundary (show methods, plot)

 Segments  mineral 1  mineral 2
     4152        iron        iron
```



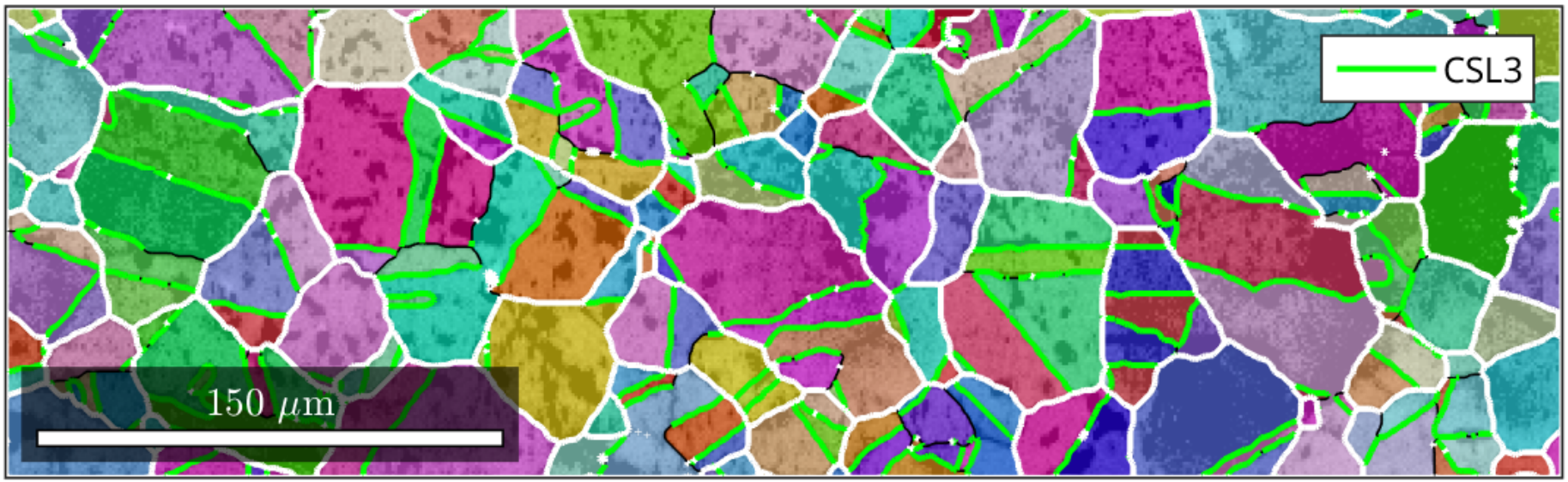
An Appetizer

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

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

```
gB = grains.boundary('indexed');
gB3 = gB(angle(gB.misorientation,CSL(3)) < 3*degree)
```

```
[mergedGrains,parentIds] = merge(grains,gB3);
```



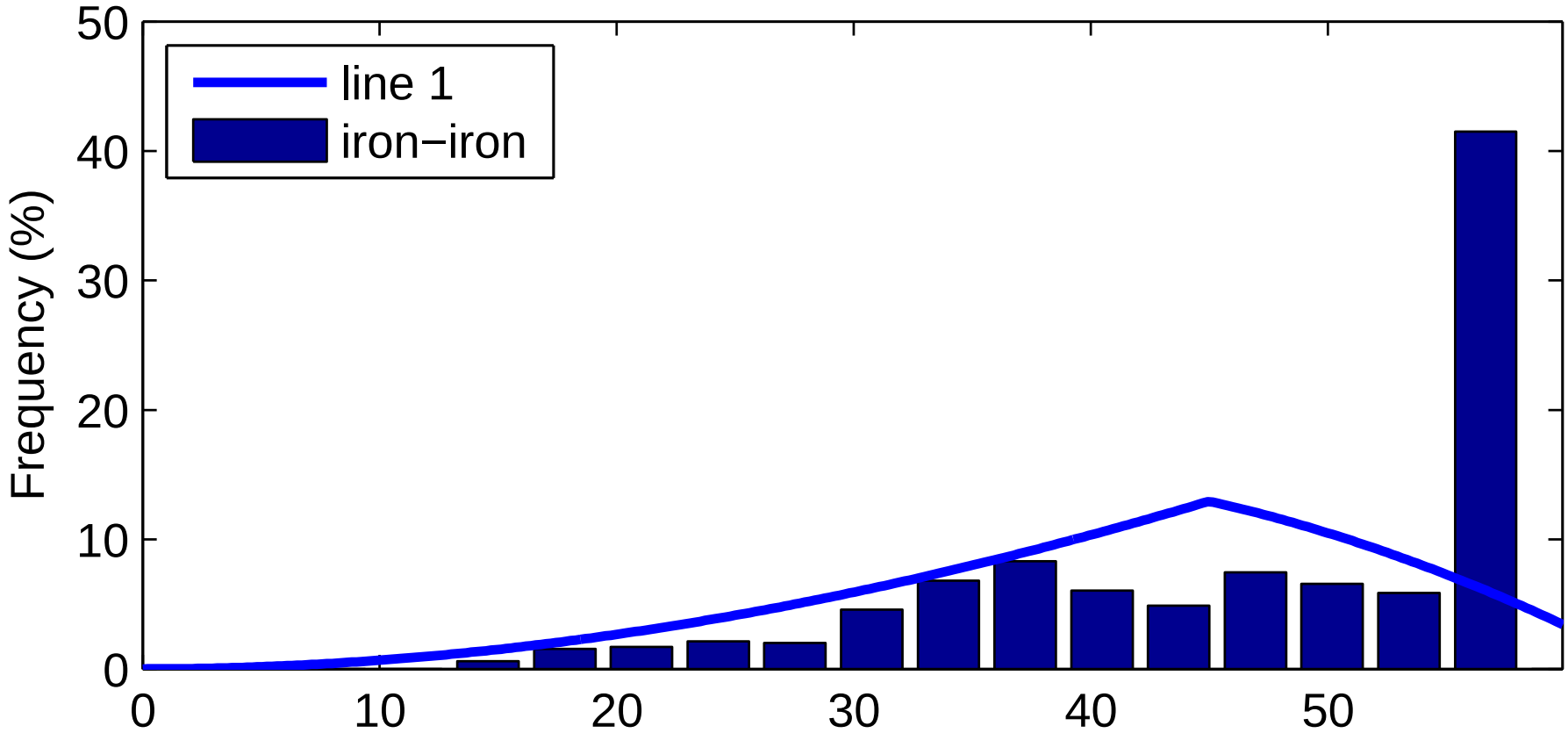
An Appetizer

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

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

```
gB = grains.boundary('indexed');
gB3 = gB(angle(gB.misorientation,CSL(3)) < 3*degree)
```

```
plotAngleDistribution(gB.misorientations)
```



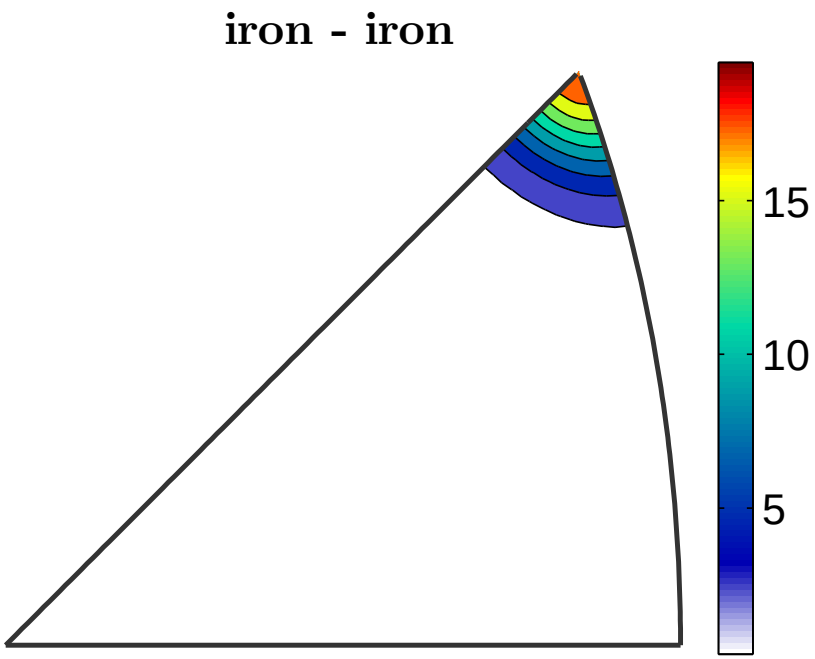
An Appetizer

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

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

```
gB = grains.boundary('indexed');
gB3 = gB(angle(gB.misorientation,CSL(3)) < 3*degree)
```

```
plotAxisDistribution(gB.misorientations,'contourf')
```



An Appetizer

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

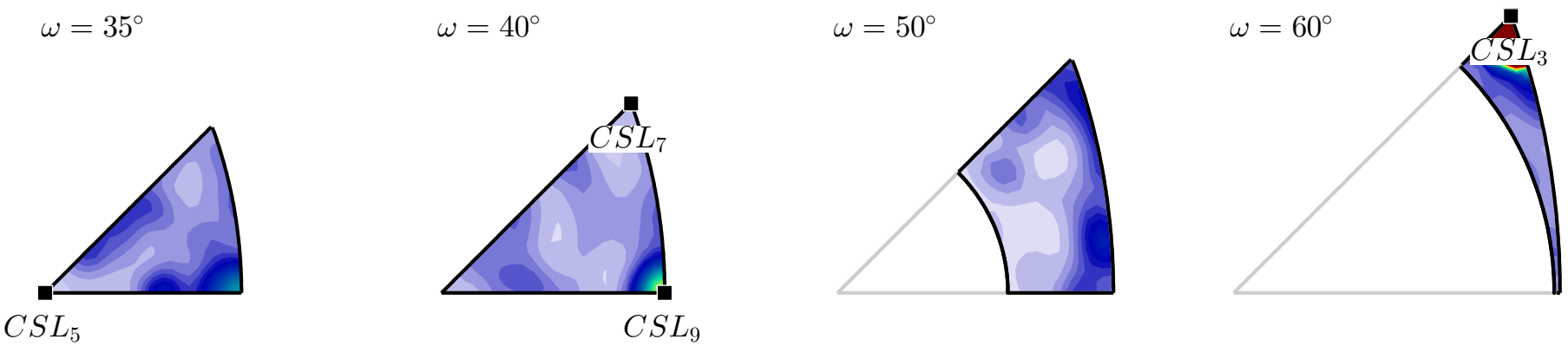
```
[grains,ebsd.grainId] = calcGrains(ebsd)
```

```
gB = grains.boundary('indexed');
gB3 = gB(angle(gB.misorientation,CSL(3)) < 3*degree)
```

```
mdf = calcMDF(gB.misorientations)
```

```
mdf = MDF (show methods, plot)
  crystal symmetry : iron (m-3m)
  crystal symmetry : iron (m-3m)

  Radially symmetric portion:
    kernel: de la Vallee Poussin, halfwidth 2.5
    center: 3436 orientations, resolution: 1.2
```



An Appetizer

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

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

```
gB = grains.boundary('indexed');
gB3 = gB(angle(gB.misorientation,CSL(3)) < 3*degree)
```

```
mdf = calcMDF(gB.misorientations)
```

```
mori = calcModes(mdf,2)
```

```
mori = misorientation (show methods, plot)
  size: 1 x 2
  crystal symmetry : iron (m-3m)
  crystal symmetry : iron (m-3m)

  Bunge Euler angles in degree
     phi1      Phi     phi2     Inv.
  62.7843 48.0359 333.939        0
  103.129 26.8018  284.91        0
```

What is MTEX?

A toolbox for texture computations

- crystal geometry
- pole figure data
- EBSD data
- ODFs, MDFs
- tensorial properties
- elastic, plastic deformation
- grains, grain boundaries

What does MTEX provide?

- basic functionality
- reliable, fast and general implementation
- a scripting language with a simple and unified syntax
- free to use, to extend, to modify
- scriptable graphics
- good documentation
- long lifetime
- big community

Why no GUI but Scripts?

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

Phase Transitions

We want to model the phase transition from magnetite to hematite given by the orientation relationship $\{111\}_m||\{0001\}_h$ and $\{\bar{1}01\}_m||\{00\bar{1}0\}_h$

```
CS_Mag = loadCIF('Magnetite')
CS_Hem = loadCIF('Hematite')
```

```
cs_Magnetite = crystalSymmetry (show methods, plot)
  mineral : Magnetite
  symmetry: m-3m
  a, b, c : 8.4, 8.4, 8.4

cs_Hematite = crystalSymmetry (show methods, plot)
  mineral : Hematite
  symmetry : -3m1
  a, b, c : 5, 5, 14
  reference frame: X||a*, Y||b, Z||c
```

```
Mag2Hem = orientation('map',...
  Miller(1,1,1,CS_Mag),Miller(0,0,0,1,CS_Hem),...
  Miller(-1,0,1,CS_Mag),Miller(1,0,-1,0,CS_Hem))
```

Phase Transitions

We want to model the phase transition from magnetite to hematite given by the orientation relationship $\{111\}_m||\{0001\}_h$ and $\{\bar{1}01\}_m||\{00\bar{1}0\}_h$

```
CS_Mag = loadCIF('Magnetite')
CS_Hem = loadCIF('Hematite')
```

```
Mag2Hem = orientation('map',...
   Miller(1,1,1,CS_Mag),Miller(0,0,0,1,CS_Hem),...
   Miller(-1,0,1,CS_Mag),Miller(1,0,-1,0,CS_Hem))
```

```
Mag2Hem = misorientation (show methods, plot)
  size: 1 x 1
  crystal symmetry : Magnetite (m-3m)
  crystal symmetry : Hematite (-3m1, X||a*, Y||b, Z||c)

  Bunge Euler angles in degree
  phi1 Phi phi2 Inv.
  120 54.7356 45 0
```

Phase Transitions

For an initial magnetite orientation

```
ori_Mag = orientation('Euler',0,0,0,CS_Mag)
```

```
ori_Mag = orientation (show methods, plot)
  size: 1 x 1
  crystal symmetry : Magnetite (m-3m)
  specimen symmetry: 1

  Bunge Euler angles in degree
  phi1   Phi  phi2  Inv.
     0     0     0     0
```

we can compute the resulting hematite orientation by

```
ori_Hem = ori_Mag * inv(Mag2Hem)
```

We should care about symmetrically equivalent orientations

```
ori_Hem = ori_Mag * symmetrise(inv(Mag2Hem))
```

Phase Transitions

For an initial magnetite orientation

```
ori_Mag = orientation('Euler',0,0,0,CS_Mag)
```

we can compute the resulting hematite orientation by

```
ori_Hem = ori_Mag * inv(Mag2Hem)
```

```
ori_Hem = orientation (show methods, plot)
  size: 1 x 1 crystal
  symmetry : Hematite (-3m1, X||a*, Y||b, Z||c)
  specimen symmetry: 1

  Bunge Euler angles in degree
  phi1 Phi phi2 Inv.
   135  54   60    0
```

We should care about symmetrically equivalent orientations

```
ori_Hem = ori_Mag * symmetrise(inv(Mag2Hem))
```

Phase Transitions

For an initial magnetite orientation

```
ori_Mag = orientation('Euler',0,0,0,CS_Mag)
```

we can compute the resulting hematite orientation by

```
ori_Hem = ori_Mag * inv(Mag2Hem)
```

We should care about symmetrically equivalent orientations

```
ori_Hem = ori_Mag * symmetrise(inv(Mag2Hem))
```

```
ori_Hem = orientation (show methods, plot)
  size: 1 x 575
  crystal symmetry : Hematite (-3m1, X||a*, Y||b, Z||c)
  specimen symmetry: 1
```

MTEX keeps track about the symmetries throughout all computations and warns in case of mismatch.

Phase Transitions

For an initial magnetite orientation

```
ori_Mag = orientation('Euler',0,0,0,CS_Mag)
```

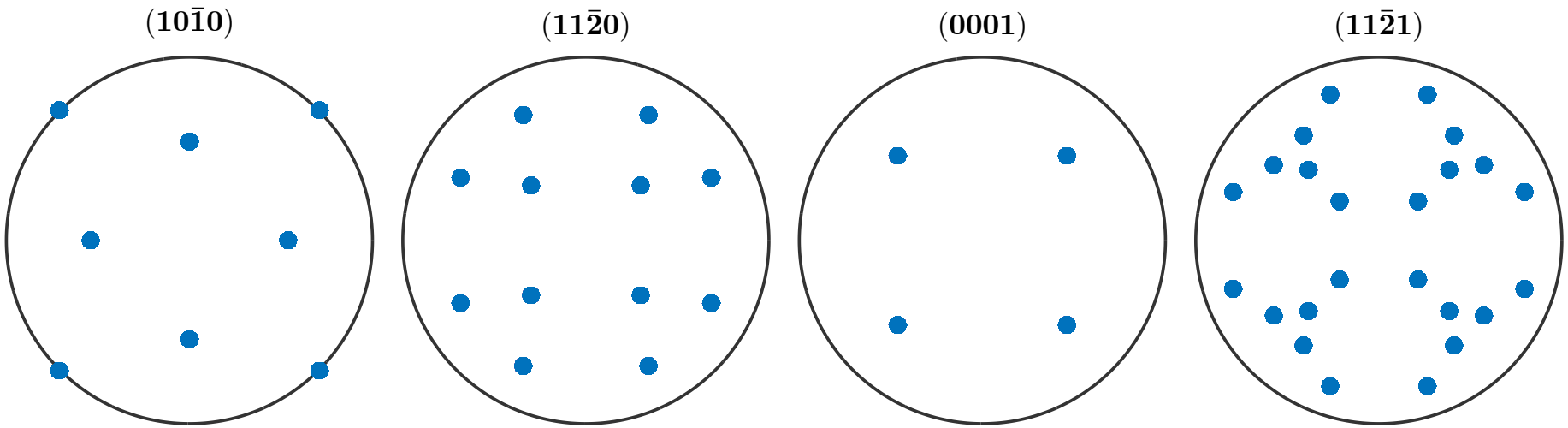
we can compute the resulting hematite orientation by

```
ori_Hem = ori_Mag * inv(Mag2Hem)
```

We should care about symmetrically equivalent orientations

```
ori_Hem = ori_Mag * symmetrise(inv(Mag2Hem))
```

```
plotPDF(ori_Mag, Miller(1,0,-1,1,CS_Hem))
```



Plastic Deformation

```
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
```

Plastic Deformation

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

```
sigmaCS = rotate(sigma,inv(grains.meanOrientation))
```

```
sigmaCS = stress tensor (show methods, plot)
  size    : 465 x 1
  rank    : 2 (3 x 3)
  mineral: iron (m-3m)
```

Plastic Deformation

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

```
sigmaCS = rotate(sigma,inv(grains.meanOrientation))
```

```
m = Miller(0,0,0,1,grains.CS)
n = Miller(1,1,-2,0,grains.CS)
[resStress,mActive,nActive,tau,active] = ...
  calcShearStress(sigmaCS,m,n,'symmetrise');
```

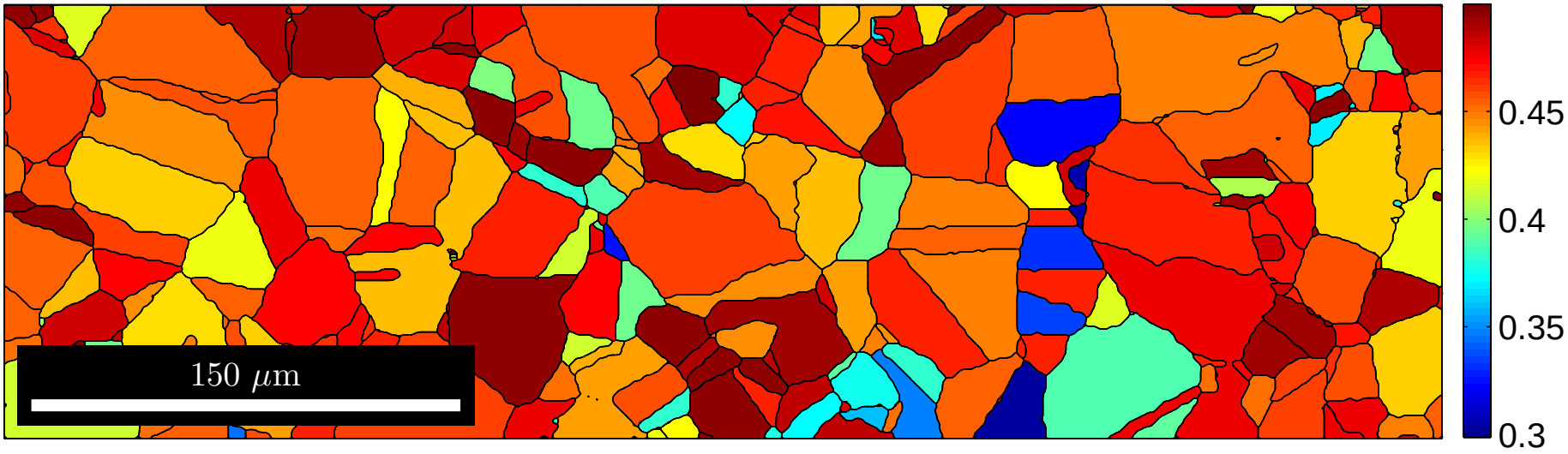
Plastic Deformation

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

```
sigmaCS = rotate(sigma,inv(grains.meanOrientation))
```

```
m = Miller(0,0,0,1,grains.CS)
n = Miller(1,1,-2,0,grains.CS)
[resStress,mActive,nActive,tau,active] = ...
  calcShearStress(sigmaCS,m,n,'symmetrise');
```

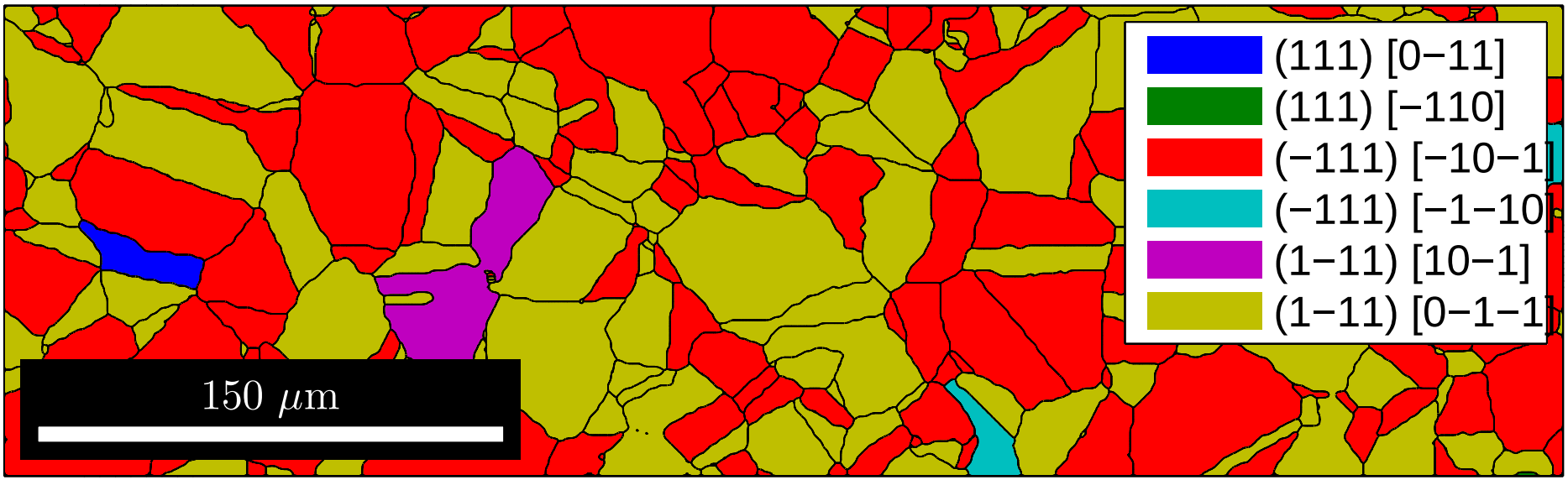
```
plot(grains,resStress)
```



Plastic Deformation

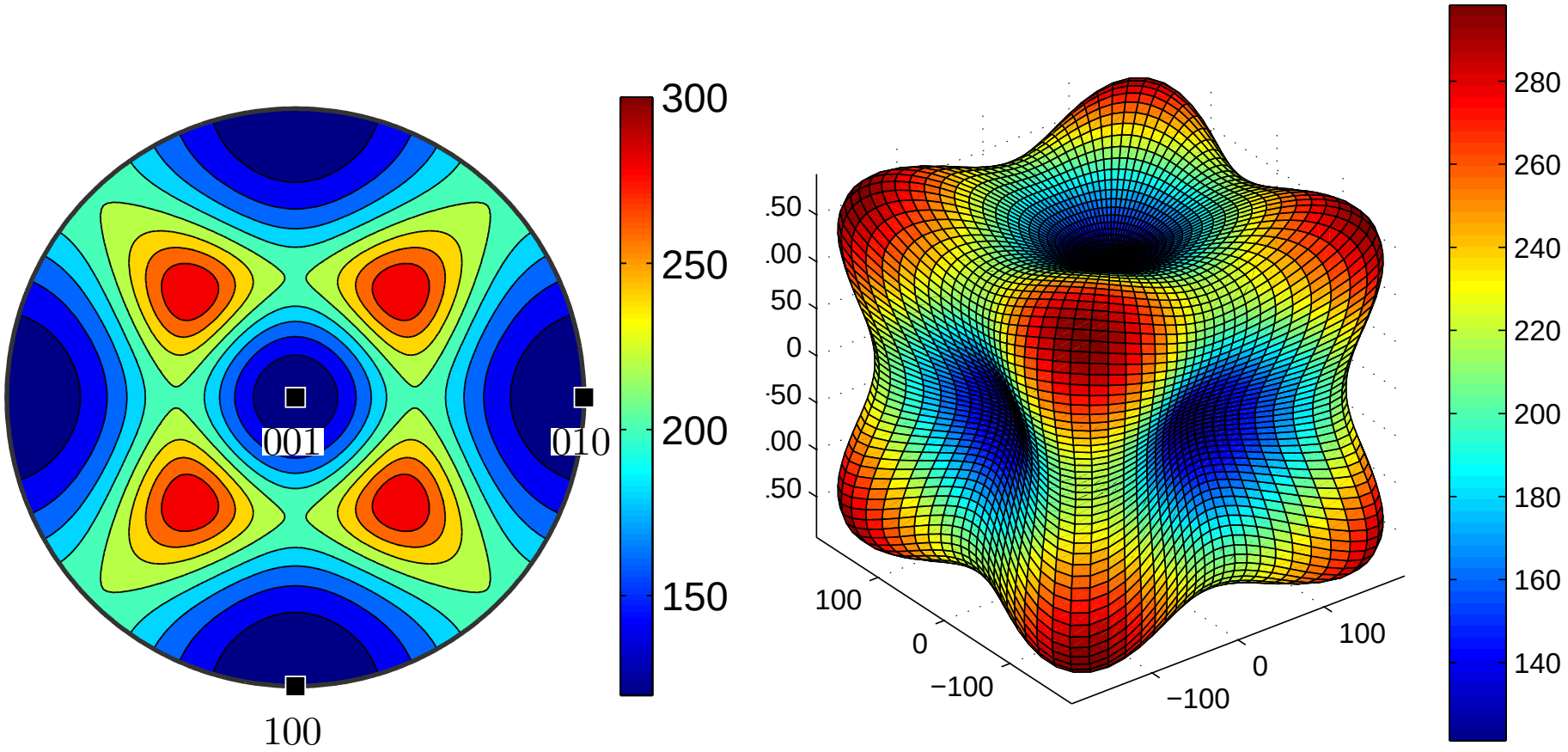
```
m = Miller(0,0,0,1,grains.CS)
n = Miller(1,1,-2,0,grains.CS)
[resStress,mActive,nActive,tau,active] = ...
  calcShearStress(sigmaCS,m,n,'symmetrise');
```

```
for id = unique(active)
  plot(grains(active==id),'FaceColor',color{id},...
  'DisplayName',[char(nSym(id)) '␣' char(bSym(id))])
end
```



Elastic Deformation

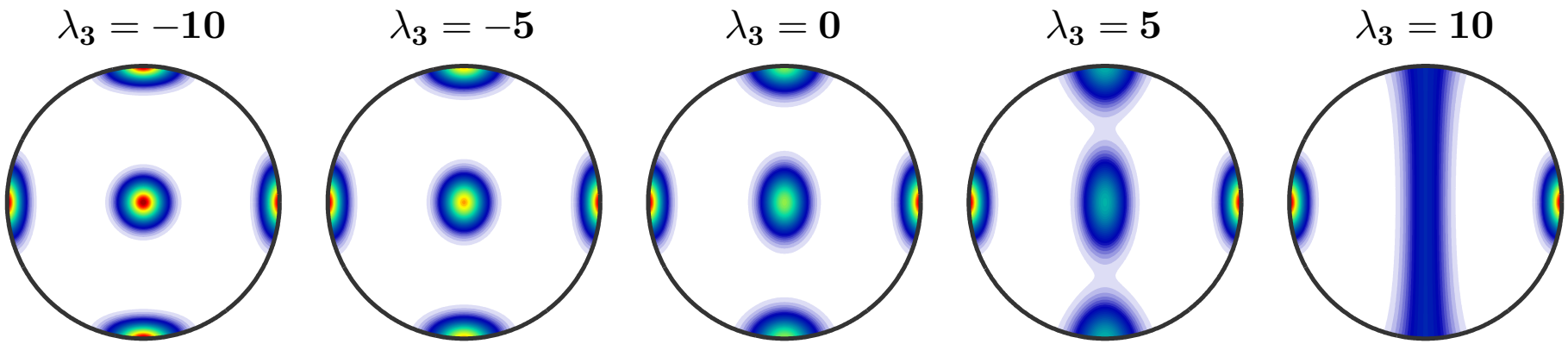
```
cs = crystalSymmetry('432','mineral','Nickel');
C = loadTensor('IN739LC.GPa',cs,'unit','GPa',...
                    'name','elastic stiffness')
```



Elastic Deformation

```
cs = crystalSymmetry('432','mineral','Nickel');
C = loadTensor('IN739LC.GPa',cs,'unit','GPa',...
                    'name','elastic stiffness')
```

```
odf{lambda3} = BinghamODF([-10,-10,lambda3,10],cs);
```



Elastic Deformation

```
cs = crystalSymmetry('432','mineral', 'Nickel');
C = loadTensor('IN739LC.GPa',cs,'unit','GPa',...
                    'name','elastic stiffness')
```

```
odf{lambda3} = BinghamODF([-10,-10,lambda3,10],cs);
```

```
for k = 1:length(odf)
  [C_v,C_r] = calcTensor(odf{k},C);
  psr_v(k) = C_v.YoungsModulus(zvector);
  psr_r(k) = C_r.YoungsModulus(zvector);
end
```