Tensor Calculations

R. Hielscher

Faculty of Mathematics,
Chemnitz University of Technology, Germany

MTEX Workshop 2016

Table of Content

What is a Tensor?

Tensors are used to described linear interactions between physical properties.

rank zero tensor scalar property, e.g. temperature

rank one tensor directional depended property, e.g. wave velocity

rank two tensors relationship between two vector fields, e.g. stress, strain, conductivity

rank three tensor relationship between a one and a two rank tensor, e.g. piezoelectricity

rank four tensor relationship between two two rank tensor, e.g. elasticity,

A tensor T of rank $s + t$ maps a tensor A of rank s onto a tensor B of rank t by the formula

$$B_{k_1,\ldots,k_t} = T_{k_1,k_2,\ldots,k_t,j_1,\ldots,j_s} A_{j_1,\ldots,j_s}$$

What is a Tensor?

Tensors are used to described linear interactions between physical properties.

rank zero tensor scalar property, e.g. temperature

rank one tensor directional depended property, e.g. wave velocity

rank two tensors relationship between two vector fields, e.g. stress, strain, conductivity

rank three tensor relationship between a one and a two rank tensor, e.g. piezoelectricity

rank four tensor relationship between two two rank tensor, e.g. elasticity,

A tensor T of rank $s + t$ maps a tensor A of rank s onto a tensor B of rank t by the formula

$$B_{k_1,\ldots,k_t} = T_{k_1,k_2,\ldots,k_t,j_1,\ldots,j_s} A_{j_1,\ldots,j_s}$$

What is a Tensor?

Tensors are used to described linear interactions between physical properties.

rank zero tensor scalar property, e.g. temperature

rank one tensor directional depended property, e.g. wave velocity

rank two tensors relationship between two vector fields, e.g. stress, strain, conductivity

rank three tensor relationship between a one and a two rank tensor, e.g. piezoelectricity

rank four tensor relationship between two two rank tensor, e.g. elasticity,

A tensor T of rank $s + t$ maps a tensor A of rank s onto a tensor B of rank t by the formula

$$B_{k_1,\ldots,k_t} = T_{k_1,k_2,\ldots,k_t,j_1,\ldots,j_s} A_{j_1,\ldots,j_s}$$

A Simple Example

```
M = [[1.45  0.00  0.19];...
     [0.00  2.11  0.00];...
     [0.19  0.00  1.79]];

sigma = tensor(M,'name','stress','unit','MPa');
```

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

  1.45     0  0.19
     0  2.11     0
  0.19     0  1.79
```

A Simple Example

```
M = [[1.45  0.00  0.19];...
     [0.00  2.11  0.00];...
     [0.19  0.00  1.79]];

sigma = tensor(M,'name','stress','unit','MPa');
```

```
n = vector3d.X % normal direction
```

```
n = vector3d (show methods, plot)
  size: 1 x 1
  x y z
  1 0 0
```

A Simple Example

```
M = [[1.45  0.00  0.19];...
     [0.00  2.11  0.00];...
     [0.19  0.00  1.79]];

sigma = tensor(M, 'name', 'stress', 'unit', 'MPa');
```

```
n = vector3d.X % normal direction
```

the stress vector $T^{\vec{n}}$ of plane $\vec{n} = \{1, 0, 0\}$, is computed by $T_j^{\vec{n}} = \sigma_{ij}\vec{n}_i$.

A Simple Example

```
M = [[1.45  0.00  0.19];...
     [0.00  2.11  0.00];...
     [0.19  0.00  1.79]];

sigma = tensor(M, 'name', 'stress', 'unit', 'MPa');
```

```
n = vector3d.X % normal direction
```

the stress vector $T^{\vec{n}}$ of plane $\vec{n} = \{1, 0, 0\}$, is computed by $T_j^{\vec{n}} = \sigma_{ij}\vec{n}_i$.

```
T = EinsteinSum(sigma,[-1  1],n,-1,'unit','MPa')
```

```
T = tensor (show methods, plot)
  unit: MPa
  rank: 1 (3)

 1.45
    0
 0.19
```

Einstein Summation

The scalar magnitudes of the normal stress σ_N and the shear stress σ_S are given as

$$\sigma_N = T_i^{\vec{n}} \vec{n}_i = \sigma_{ij} \vec{n}_i \vec{n}_j \quad \text{and} \quad \sigma_S = \sqrt{T_i^{\vec{n}} T_i^{\vec{n}} - \sigma_N^2}.$$

```
sigmaN = double(EinsteinSum(T,-1,n,-1))
sigmaS = sqrt(double(EinsteinSum(T,-1,T,-1))...
         - sigmaN^2)
```

```
sigmaN =

    1.4500

sigmaS =

    0.1900
```

Einstein Summation

The scalar magnitudes of the normal stress σ_N and the shear stress σ_S are given as

$$\sigma_N = T_i^{\vec{n}} \vec{n}_i = \sigma_{ij} \vec{n}_i \vec{n}_j \quad \text{and} \quad \sigma_S = \sqrt{T_i^{\vec{n}} T_i^{\vec{n}} - \sigma_N^2}.$$

```
sigmaN = double(EinsteinSum(T,-1,n,-1))
sigmaS = sqrt(double(EinsteinSum(T,-1,T,-1))...
         - sigmaN^2)
```

```
sigmaN =

    1.4500

sigmaS =

    0.1900
```

Visualization

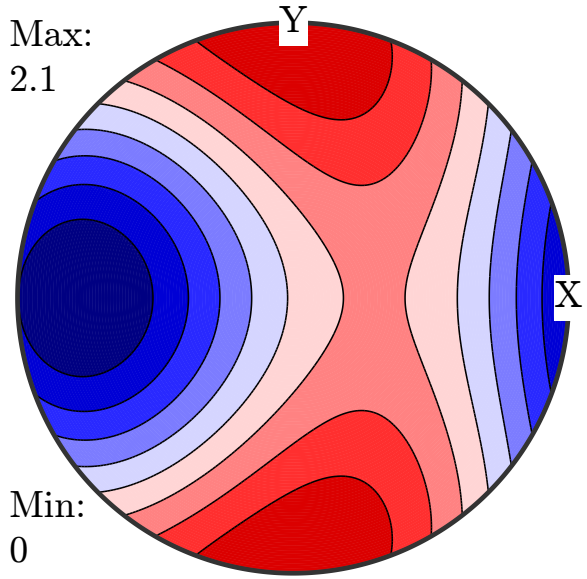
For a second order tensor σ_{ij} its directional magnitude $R(\vec{x})$ is defined as

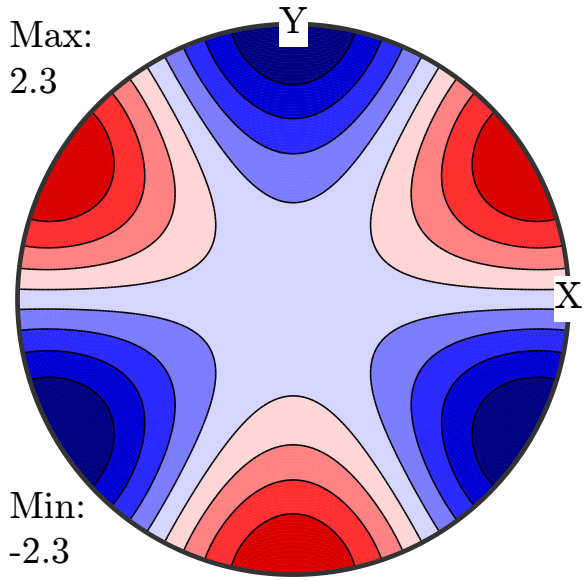
$$R(\vec{x}) = \sigma_{ij}\vec{x}_i\vec{x}_j.$$

```
R = EinsteinSum(sigma,[-1 -2],...
  x,-1,x,-2)
```

```
R = directionalMagnitude(sigma,x)
```

```
plot(sigma,'minmax')
mtexColorMap blue2red
```





Visualization

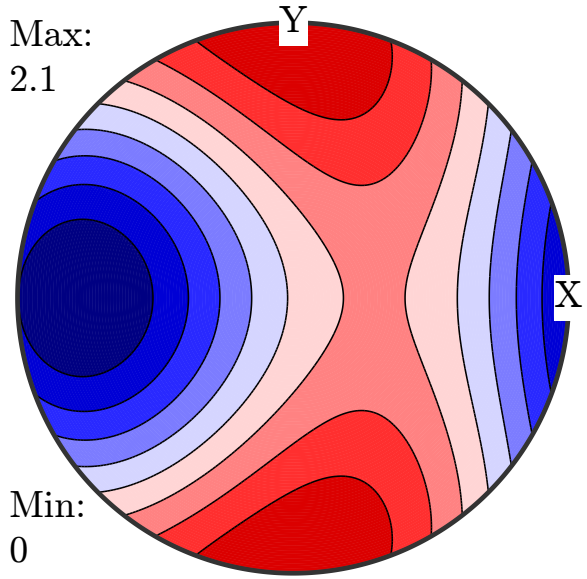
For a second order tensor σ_{ij} its directional magnitude $R(\vec{x})$ is defined as

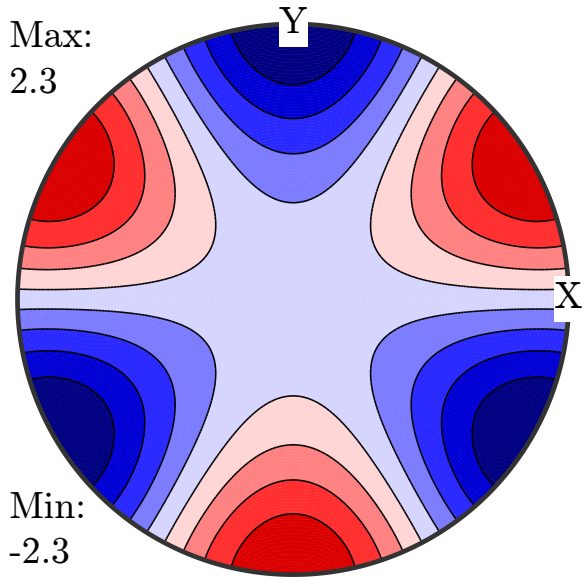
$$R(\vec{x}) = \sigma_{ij}\vec{x}_i\vec{x}_j.$$

```
R = EinsteinSum(sigma,[-1 -2],...
  x,-1,x,-2)
```

```
R = directionalMagnitude(sigma,x)
```

```
plot(sigma,'minmax')
mtexColorMap blue2red
```





Visualization

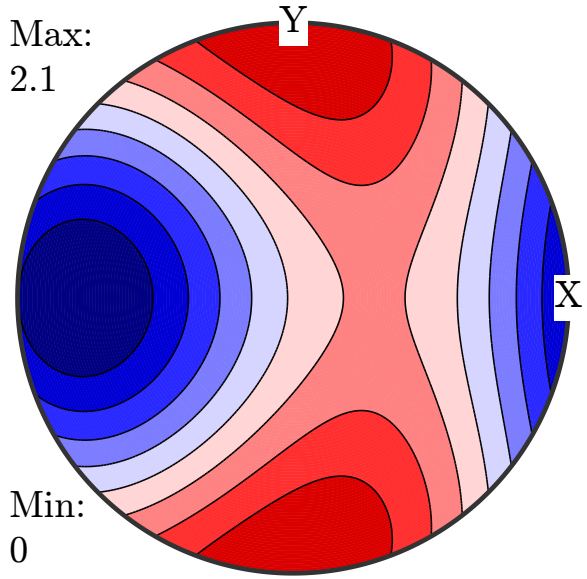
For a second order tensor σ_{ij} its directional magnitude $R(\vec{x})$ is defined as

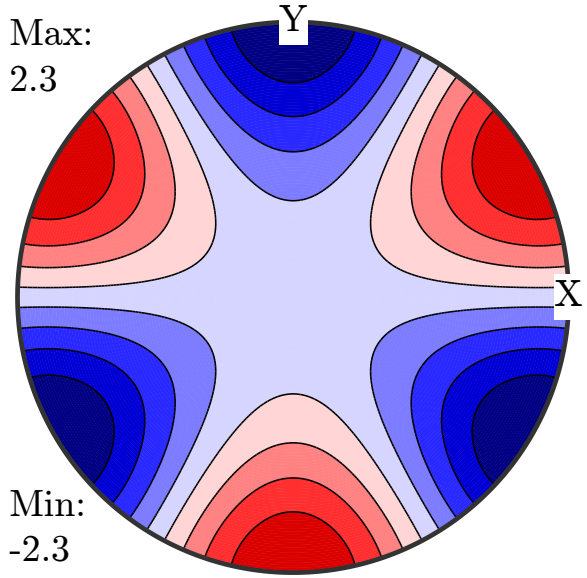
$$R(\vec{x}) = \sigma_{ij}\vec{x}_i\vec{x}_j.$$

```
R = EinsteinSum(sigma,[-1 -2],...
  x,-1,x,-2)
```

```
R = directionalMagnitude(sigma,x)
```

```
plot(sigma,'minmax')
mtexColorMap blue2red
```





Visualization

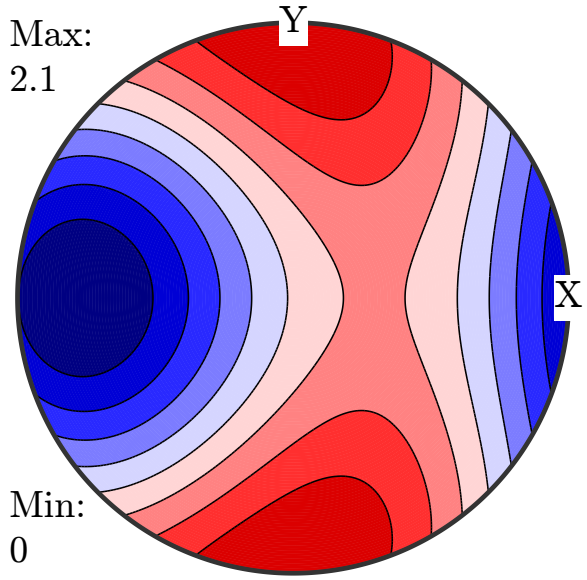
For a second order tensor σ_{ij} its directional magnitude $R(\vec{x})$ is defined as

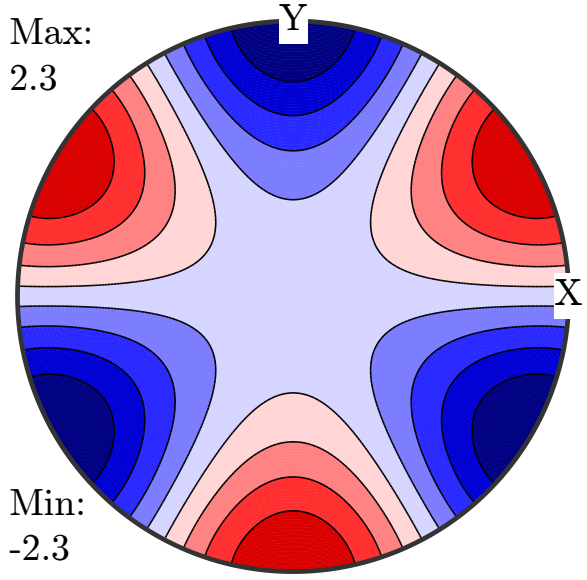
$$R(\vec{x}) = \sigma_{ij}\vec{x}_i\vec{x}_j.$$

```
R = EinsteinSum(sigma,[-1 -2],...
    x,-1,x,-2)
```

```
R = directionalMagnitude(sigma,x)
```

```
plot(sigma,'minmax')
mtexColorMap blue2red
```



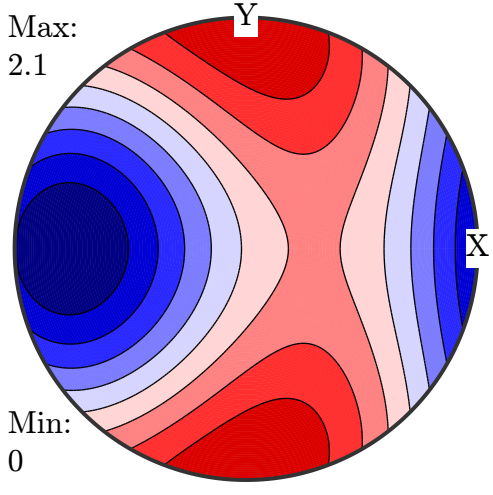


Field Tensors vs. Matter Tensors

field tensors:

- in specimen coordinates
- describe applied forces like: stress, electric field

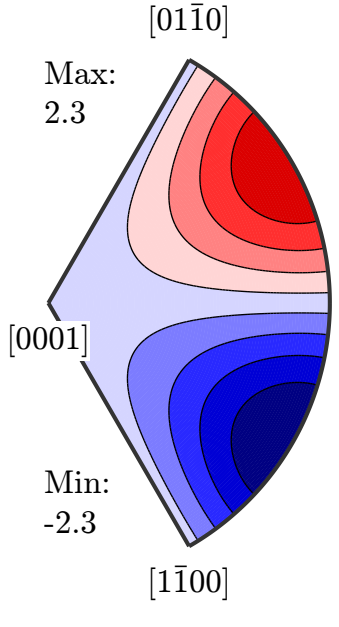
```
sigma = tensor(M, 'name', 'stress', ...
                  'unit', 'MPa');
```



matter tensors:

- in crystal coordinates
- describe physical properties like: electrical or thermal conductivity, magnetic permeability

```
CS = crystalSymmetry('321', ...
 'mineral', 'Quartz', 'X||a*', 'Z||c')
P = tensor(M,CS, 'name', ...
     'piecoelectricity', 'unit', 'C/N')
```

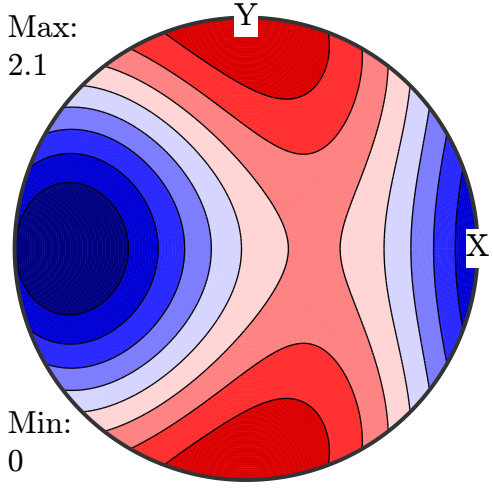


Field Tensors vs. Matter Tensors

field tensors:

- in specimen coordinates
- describe applied forces like: stress, electric field

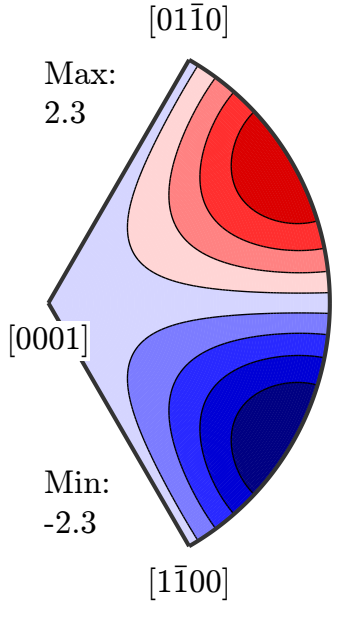
```
sigma = tensor(M, 'name', 'stress', ...
                  'unit', 'MPa');
```



matter tensors:

- in crystal coordinates
- describe physical properties like: electrical or thermal conductivity, magnetic permeability

```
CS = crystalSymmetry('321', ...
'mineral', 'Quartz', 'X||a*', 'Z||c')
P = tensor(M, CS, 'name', ...
      'piecoelectricity', 'unit', 'C/N')
```



Rotating Tensors

Consider the piezoelectricity tensor

```
P=tensor(M,CS,'name','piecoelectricity','unit','C/N')
```

```
P = tensor (show methods, plot)
  propertyname     : piecoelectricity
  unit             : C/N
  rank             : 3 (3 x 3 x 3)
  doubleConvention: true
  mineral          : Quartz (321, X||a*, Y||b, Z||c)

  tensor in compact matrix form:
     0      0      0 -0.67      0    4.6
   2.3   -2.3      0      0   0.67      0
     0      0      0      0      0      0
```

Remember orientations transforms crystal into specimen coordinates

```
ori = orientation('Euler',10*degree,20*degree,0,CS)
rotate(P,ori)
```

Contrary, an inverse orientation transforms specimen coordinates into

Rotating Tensors

Consider the piezoelectricity tensor

```
P=tensor(M,CS,'name','piecoelectricity','unit','C/N')
```

Remember orientations transforms crystal into specimen coordinates

```
ori = orientation('Euler',10*degree,20*degree,0,CS)
rotate(P,ori)
```

```
ans = tensor (show methods, plot)
  propertyname     : piecoelectricity
  rank             : 3 (3 x 3 x 3)
  doubleConvention: true

  tensor in compact matrix form:
  -1.08   1.25  -0.17  -0.01   1.47   3.72
   1.92  -1.63  -0.29  -1.29   1.10   1.86
   0.76  -0.66  -0.09  -0.46   0.30   0.43
```

Contrary, an inverse orientation transforms specimen coordinates into crystal coordinates.

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

Rotating Tensors

Consider the piezoelectricity tensor

```
P=tensor(M,CS,'name','piecoelectricity','unit','C/N')
```

Remember orientations transforms crystal into specimen coordinates

```
ori = orientation('Euler',10*degree,20*degree,0,CS)
rotate(P,ori)
```

Contrary, an inverse orientation transforms specimen coordinates into crystal coordinates.

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

```
ans = stress tensor (show methods, plot)
  unit    : MPa
  rank    : 2 (3 x 3)
  mineral: Quartz (321, X||a*, Y||b, Z||c)

  1.47   0.17   0.14
  0.17   2.03  -0.12
  0.14  -0.12   1.85
```

Average Tensors

The average tensorial property of a specimen is the mean of matter tensors rotated according to each grain orientation o_m, $m = 1, \dots, M$.

The Voigt and the Reuss averages of a tensor T are defines as

$$\langle T \rangle^{\text{Voigt}} = \sum_{m=1}^{M} V_m T(o_m), \quad \langle T \rangle^{\text{Reuss}} = \left[\sum_{m=1}^{M} V_m T^{-1}(o_m) \right]^{-1}.$$

For EBSD data this is computed by

```
[TVoigt, TReus, THill] = calcTensor(ebsd,T)
```

and for an ODF by

```
[TVoigt, TReus, THill] = calcTensor(odf,T)
```

Average Tensors

The average tensorial property of a specimen is the mean of matter tensors rotated according to each grain orientation o_m, $m = 1, \dots, M$.

The Voigt and the Reuss averages of a tensor T are defines as

$$\langle T \rangle^{\text{Voigt}} = \sum_{m=1}^{M} V_m T(o_m), \quad \langle T \rangle^{\text{Reuss}} = \left[\sum_{m=1}^{M} V_m T^{-1}(o_m) \right]^{-1}.$$

For EBSD data this is computed by

```
[TVoigt, TReus, THill] = calcTensor(ebsd,T)
```

and for an ODF by

```
[TVoigt, TReus, THill] = calcTensor(odf,T)
```

Average Tensors

The average tensorial property of a specimen is the mean of matter tensors rotated according to each grain orientation o_m, $m = 1, \dots, M$.

The Voigt and the Reuss averages of a tensor T are defines as

$$\langle T \rangle^{\text{Voigt}} = \sum_{m=1}^{M} V_m T(o_m), \quad \langle T \rangle^{\text{Reuss}} = \left[\sum_{m=1}^{M} V_m T^{-1}(o_m) \right]^{-1}.$$

For EBSD data this is computed by

```
[TVoigt, TReus, THill] = calcTensor(ebsd,T)
```

and for an ODF by

```
[TVoigt, TReus, THill] = calcTensor(odf,T)
```

Average Tensors

The average tensorial property of a specimen is the mean of matter tensors rotated according to each grain orientation o_m, $m = 1, \dots, M$.

The Voigt and the Reuss averages of a tensor T are defines as

$$\langle T \rangle^{\text{Voigt}} = \sum_{m=1}^{M} V_m T(o_m), \quad \langle T \rangle^{\text{Reuss}} = \left[\sum_{m=1}^{M} V_m T^{-1}(o_m) \right]^{-1}.$$

For EBSD data this is computed by

```
[TVoigt, TReus, THill] = calcTensor(ebsd,T)
```

and for an ODF by

```
[TVoigt, TReus, THill] = calcTensor(odf,T)
```

Elastic Deformation

Imort some stiffness tensor

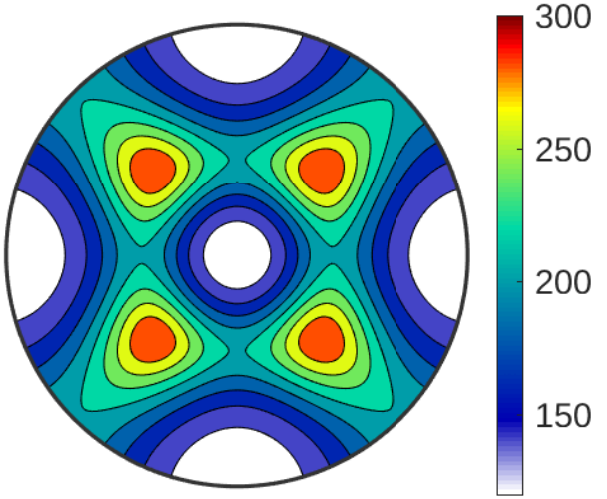
```
cs = loadCIF('Nickel')
C = loadTensor('IN739LC.GPa',...
    cs,'name','elastic stiffness')
```

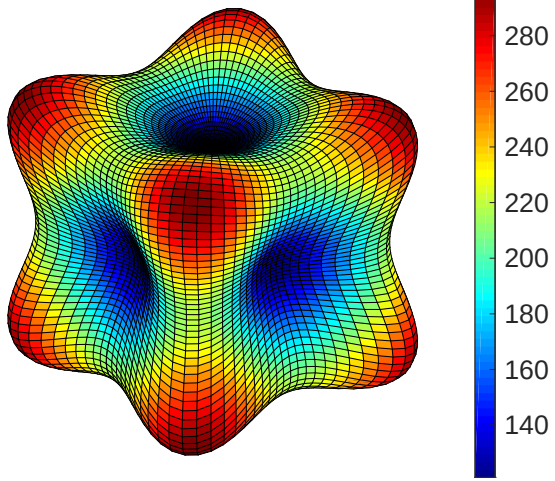
```
C = elastic stiffness tensor (show methods)
  unit   : GPa
  rank   : 4 (3 x 3 x 3 x 3)
  mineral: Ni (432)

  tensor in Voigt matrix representation:
 235.16  147.67  147.67       0       0       0
 147.67  235.16  147.67       0       0       0
 147.67  147.67  235.16       0       0       0
      0       0       0  122.53       0       0
      0       0       0       0  122.53       0
      0       0       0       0       0  122.53
```

The inverse of the stiffness is the compliance

```
S = inv(C)
```

Consider unaxial stress σ





Elastic Deformation

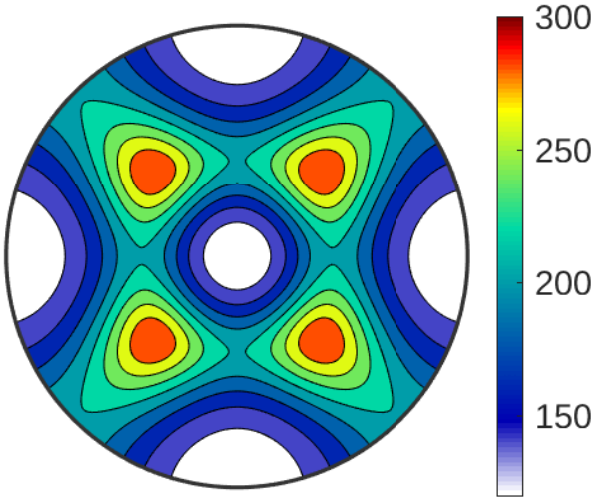
Imort some stiffness tensor

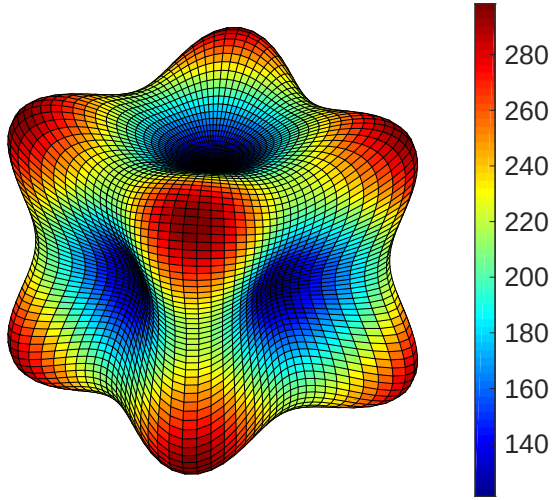
```
cs = loadCIF('Nickel')
C = loadTensor('IN739LC.GPa',...
    cs,'name','elastic stiffness')
```

The inverse of the stiffness is the compliance

```
S = inv(C)
```

```
S = elastic compliance tensor
  unit              : 1 / GPa
  rank              : 4 (3 x 3 x 3 x 3)
  doubleConvention: true
  mineral           : Ni (432)

  tensor in Voigt matrix represent.: *10^-4
  82.48 -31.82 -31.82      0      0      0
-31.82  82.48 -31.82      0      0      0
-31.82 -31.82  82.48      0      0      0
     0      0      0  81.61      0      0
     0      0      0      0  81.61      0
     0      0      0      0      0  81.61
```





Consider uniaxial stress σ

Elastic Deformation

Imort some stiffness tensor

```
cs = loadCIF('Nickel')
C = loadTensor('IN739LC.GPa',...
    cs,'name','elastic stiffness')
```

The inverse of the stiffness is the compliance

```
S = inv(C)
```

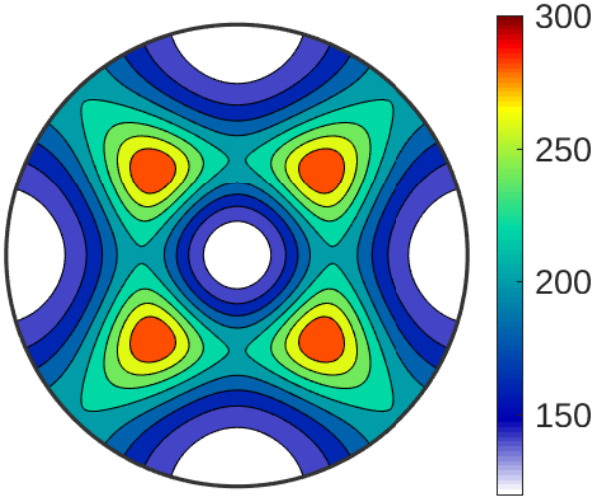
Consider unaxial stress σ

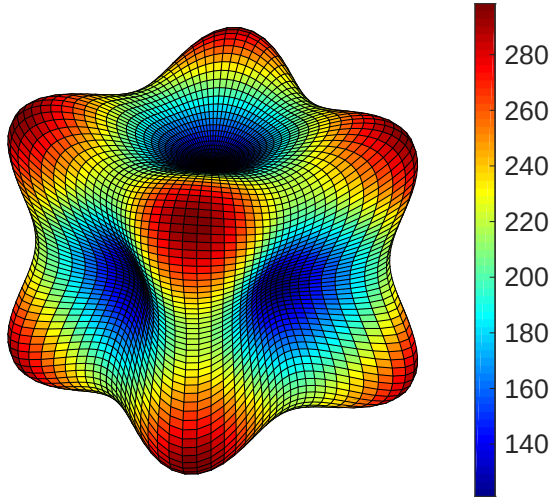
```
sigma = tensor(M,'name','stress')
```

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

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

strain $\varepsilon_{ij} = S_{klij}\sigma_{kl}$

```
eps = EinsteinSum(rotate(S,ori),...
    [-1 -2 1 2],sigma,[-1 -2])
```





Elastic Deformation

Imort some stiffness tensor

```
cs = loadCIF('Nickel')
C = loadTensor('IN739LC.GPa',...
    cs,'name','elastic stiffness')
```

The inverse of the stiffness is the compliance

```
S = inv(C)
```

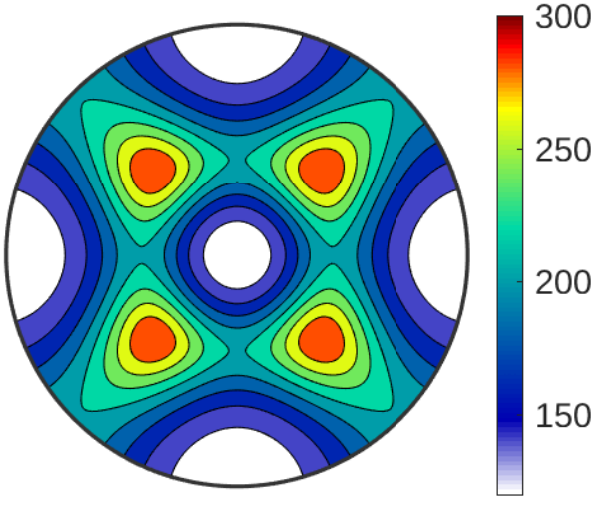
Consider unaxial stress σ

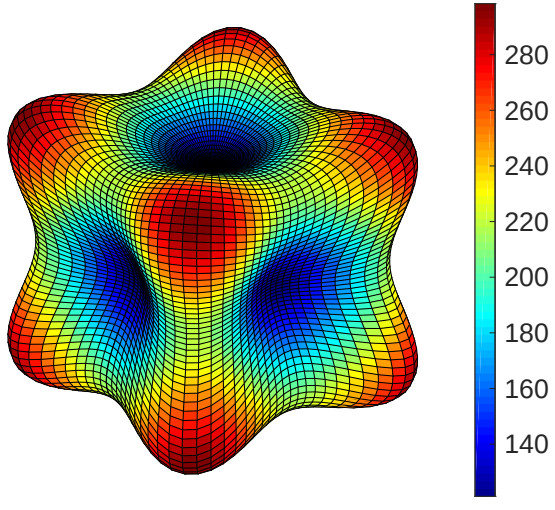
```
sigma = tensor(M,'name','stress')
```

strain $\varepsilon_{ij} = S_{klij}\sigma_{kl}$

```
eps = EinsteinSum(rotate(S,ori),...
    [-1 -2 1 2],sigma,[-1 -2])
```

```
eps = strain tensor (show methods, plot)
  rank: 2 (3 x 3)

 147.67       0       0
      0  147.67       0
      0       0  235.16
```





Elasticity Tensors

derived elastic properties

```
beta = volumeCompressibility(C)
beta = linearCompressibility(C,x)
E    = YoungsModulus(C,x)
G    = shearModulus(C,h,u)
nu   = PoissonRatio(C,x,y)
T    = ChristoffelTensor(C,n)
```

```
x = plotS2Grid('upper');
nu = C.PoissonRatio(x,xvector);
contourf(x,nu,'minmax')
mtexColorMap blue2red
```

```
[value,id] = max(nu)
```

```
annotate(x(id),'label','max')
```

Elasticity Tensors

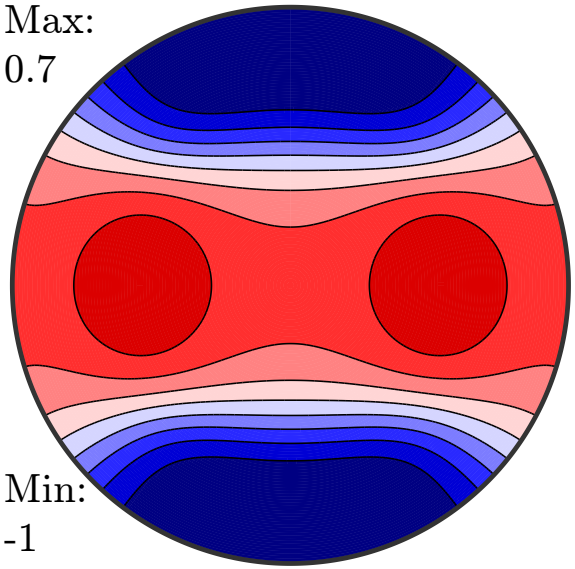
derived elastic properties

```
beta = volumeCompressibility(C)
beta = linearCompressibility(C,x)
E    = YoungsModulus(C,x)
G    = shearModulus(C,h,u)
nu   = PoissonRatio(C,x,y)
T    = ChristoffelTensor(C,n)
```

```
x = plotS2Grid('upper');
nu = C.PoissonRatio(x,xvector);
contourf(x,nu,'minmax')
mtexColorMap blue2red
```

```
[value,id] = max(nu)
```

```
annotate(x(id),'label','max')
```



Elasticity Tensors

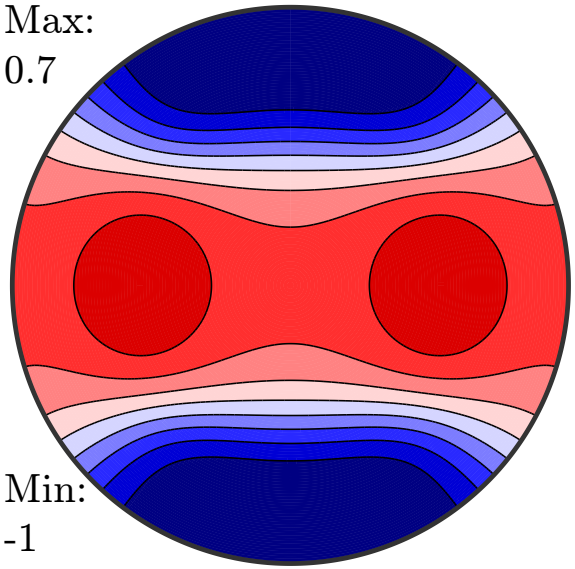
derived elastic properties

```
beta = volumeCompressibility(C)
beta = linearCompressibility(C,x)
E    = YoungsModulus(C,x)
G    = shearModulus(C,h,u)
nu   = PoissonRatio(C,x,y)
T    = ChristoffelTensor(C,n)
```

```
x = plotS2Grid('upper');
nu = C.PoissonRatio(x,xvector);
contourf(x,nu,'minmax')
mtexColorMap blue2red
```

```
[value,id] = max(nu)
```

```
value =
    0.6956
id =
        8236
```



Elasticity Tensors

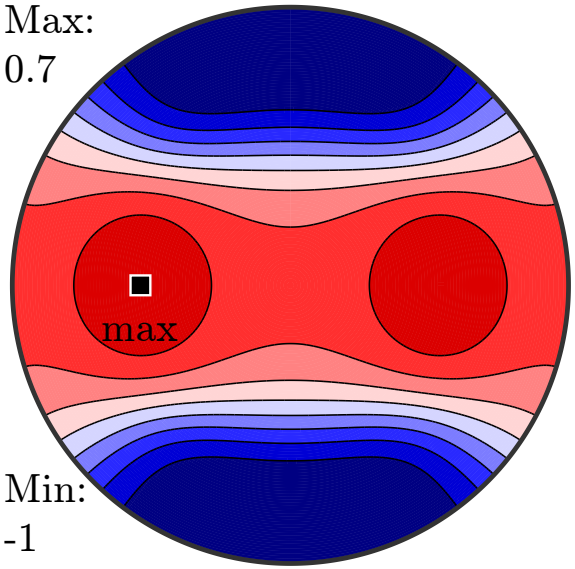
derived elastic properties

```
beta = volumeCompressibility(C)
beta = linearCompressibility(C,x)
E    = YoungsModulus(C,x)
G    = shearModulus(C,h,u)
nu   = PoissonRatio(C,x,y)
T    = ChristoffelTensor(C,n)
```

```
x = plotS2Grid('upper');
nu = C.PoissonRatio(x,xvector);
contourf(x,nu,'minmax')
mtexColorMap blue2red
```

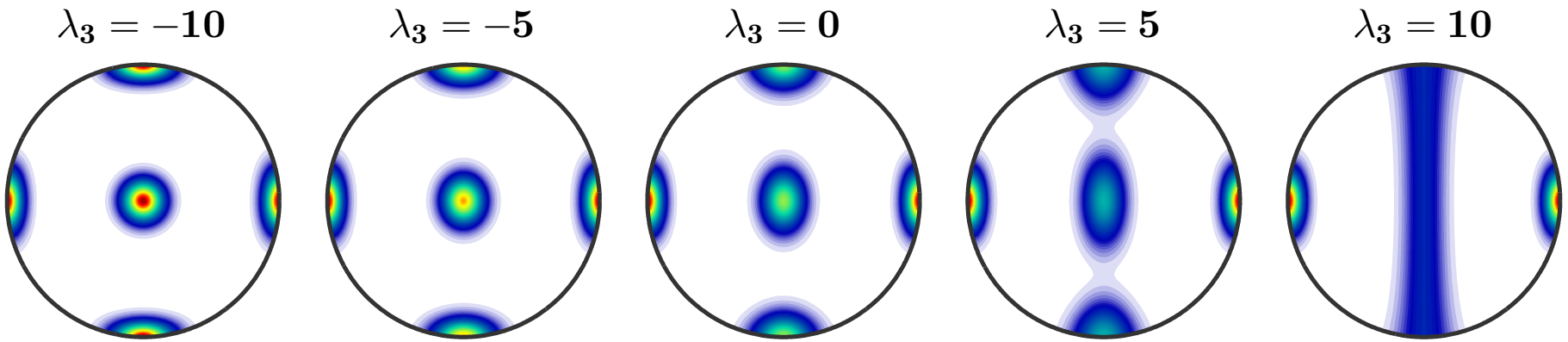
```
[value,id] = max(nu)
```

```
annotate(x(id),'label','max')
```



Evolution of Elasticity

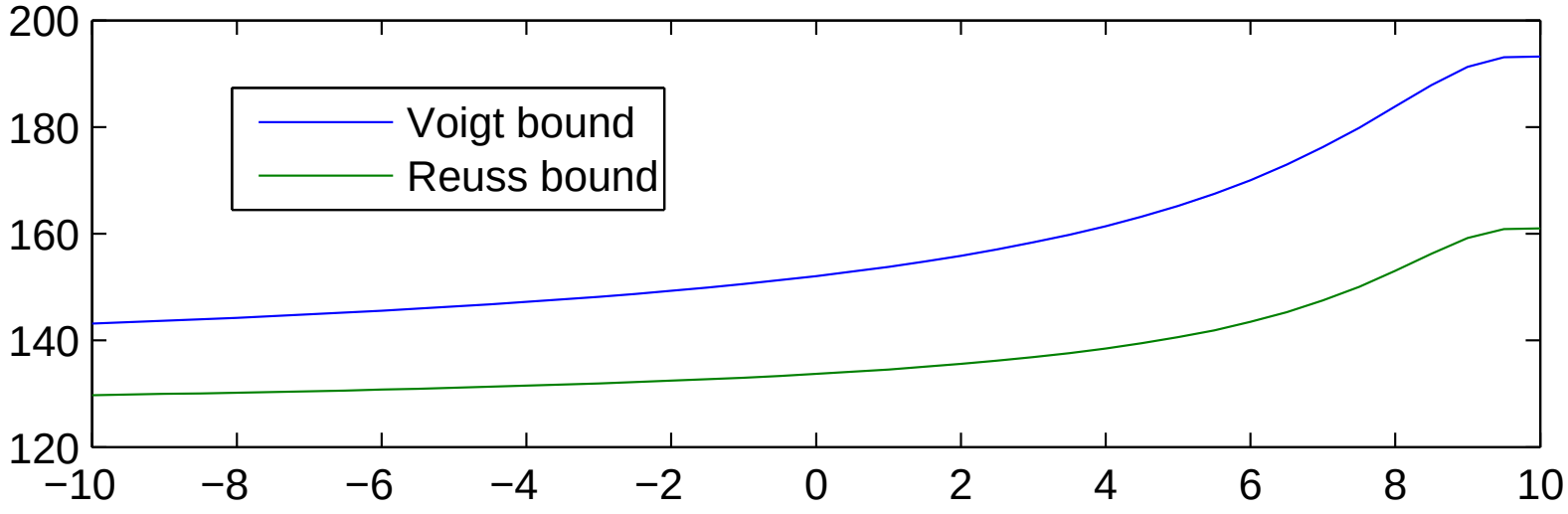
```
for k = 1:5
  odf{k} = BinghamODF([-10,-10,5*k-15,10],cs);
end
```



Evolution of Elasticity

```
for k = 1:5
  odf{k} = BinghamODF([-10,-10,5*k-15,10],cs);
end
```

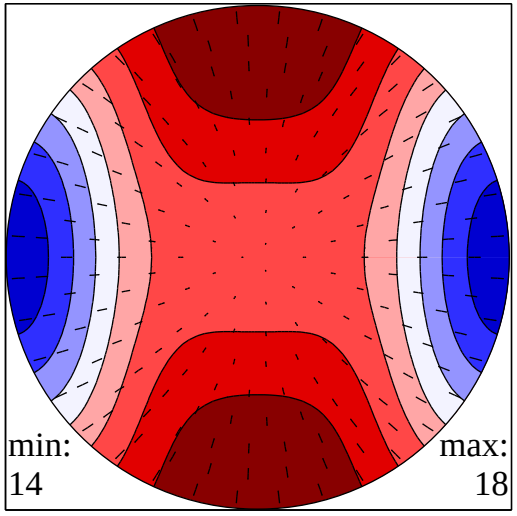
```
for k = 1:length(odf)
  [C_v,C_r] = calcTensor(odf{k},C);
  psr_v(k) = C_v.YoungsModulus(vector3d.Z);
  psr_r(k) = C_r.YoungsModulus(vector3d.Z);
end
```



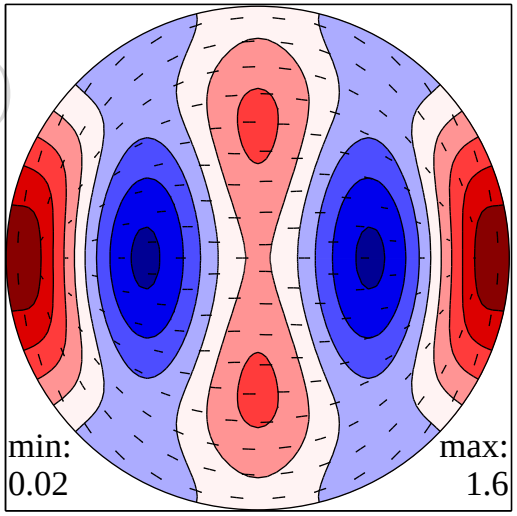
Wave Velocities

```
[vp,vs1,vs2,pp,ps1,ps2] = velocity(C,vector3d.X,rho)
```

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

hold on
plot(C,'PlotType','velocity','pp')
hold off
```



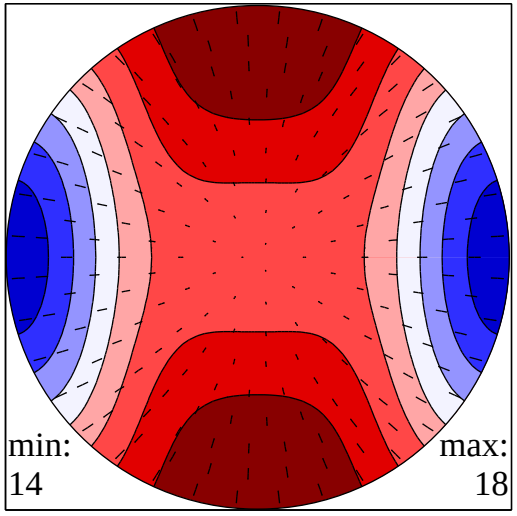

```
plot(C,'PlotType','velocity','vs1-vs2')

hold on
plot(C,'PlotType','velocity','ps1')
hold off
```



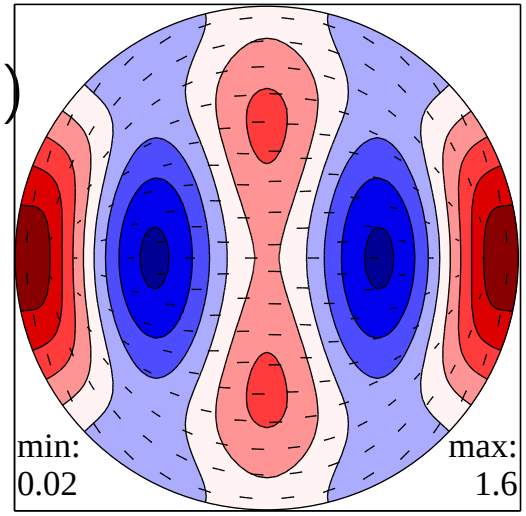
Wave Velocities

```
[vp,vs1,vs2,pp,ps1,ps2] = velocity(C,vector3d.X,rho)
```

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

hold on
plot(C,'PlotType','velocity','pp')
hold off
```




```
plot(C,'PlotType','velocity','vs1-vs2')

hold on
plot(C,'PlotType','velocity','ps1')
hold off
```



Schmidt Factor

Consider the slip system

```
b = Miller(0,-1,1,cs,'uvw');   % slip direction
n = Miller(1,1,1,cs,'hkl');    % slip plane normal
```

and the extension direction

```
r = vector3d(0,0,1)            % extension direction
```

Then the Schmid factor in direction **r** is given by

$$SF = \cos\theta \cos\rho$$

In **MTEX** this can be computed by

```
SF = dot(n,r) * dot(b,r)
```

Schmidt Factor

Consider the slip system

```
b = Miller(0,-1,1,cs,'uvw');   % slip direction
n = Miller(1,1,1,cs,'hkl');    % slip plane normal
```

and the extension direction

```
r = vector3d(0,0,1)            % extension direction
```

Then the Schmid factor in direction **r** is given by

$$SF = \cos\theta \cos\rho$$

In **MTEX** this can be computed by

```
SF = dot(n,r) * dot(b,r)
```

Schmidt Factor

Consider the slip system

```
b = Miller(0,-1,1,cs,'uvw');   % slip direction
n = Miller(1,1,1,cs,'hkl');    % slip plane normal
```

and the extension direction

```
r = vector3d(0,0,1)            % extension direction
```

Then the Schmid factor in direction **r** is given by

$$SF = \cos\theta \cos\rho$$

In MTEX this can be computed by

```
SF = dot(n,r) * dot(b,r)
```

Schmidt Factor

Consider the slip system

```
b = Miller(0,-1,1,cs,'uvw');   % slip direction
n = Miller(1,1,1,cs,'hkl');    % slip plane normal
```

and the extension direction

```
r = vector3d(0,0,1)            % extension direction
```

Then the Schmid factor in direction **r** is given by

$$SF = \cos\theta \cos\rho$$

In **MTEX** this can be computed by

```
SF = dot(n,r) * dot(b,r)
```

The Schmid Tensor

In tensor language the slip system may expressed as

```
ST = SchmidTensor(n,b)
```

```
ans = symmetric Schmid tensor
  rank: 2 (3 x 3)

  0    0.5  0
  0.5  0    0
  0    0    0
```

The Schmid Tensor

In tensor language the slip system may expressed as

```
ST = SchmidTensor(n,b)
```

and the force to the specimen can be described by a stress tensor, e.g.

```
sigma = tensor(
  [[ 0.0  0.0  0.0 ];...
   [ 0.0  0.0  0.0 ];...
   [ 0.0  0.0  1.0 ]],'name','stress');
```

The Schmid Tensor

In tensor language the slip system may expressed as

```
ST = SchmidTensor(n,b)
```

and the force to the specimen can be described by a stress tensor, e.g.

```
sigma = tensor(
  [[ 0.0  0.0  0.0 ];...
   [ 0.0  0.0  0.0 ];...
   [ 0.0  0.0  1.0 ]],'name','stress');
```

Then the Schmid factor is

```
SF = EinsteinSum(ST,[-1,-2],sigma,[-1,-2])
```

The Schmid Tensor

In tensor language the slip system may expressed as

```
ST = SchmidTensor(n,b)
```

We can do this also for a list of directions

```
r = plotS2Grid('upper')
```

The Schmid Tensor

In tensor language the slip system may expressed as

```
ST = SchmidTensor(n,b)
```

We can do this also for a list of directions

```
r = plotS2Grid('upper')
```

```
r = vector3d (show methods, plot)
 size: 91 x 361
 resolution: 1
 plot: true
 region: upper hemisphere
 theta: 91 x 361 double
 rho: 91 x 361 double
```

The Schmid Tensor

In tensor language the slip system may expressed as

```
ST = SchmidTensor(n,b)
```

We can do this also for a list of directions

```
r = plotS2Grid('upper')
```

```
sigma = EinsteinSum(...
          tensor(r),1,...
          tensor(r),2)
```

```
sigma = tensor (show methods, plot)
  size: 5365 x 1
  rank: 2 (3 x 3)
```

The Schmid Tensor

In tensor language the slip system may expressed as

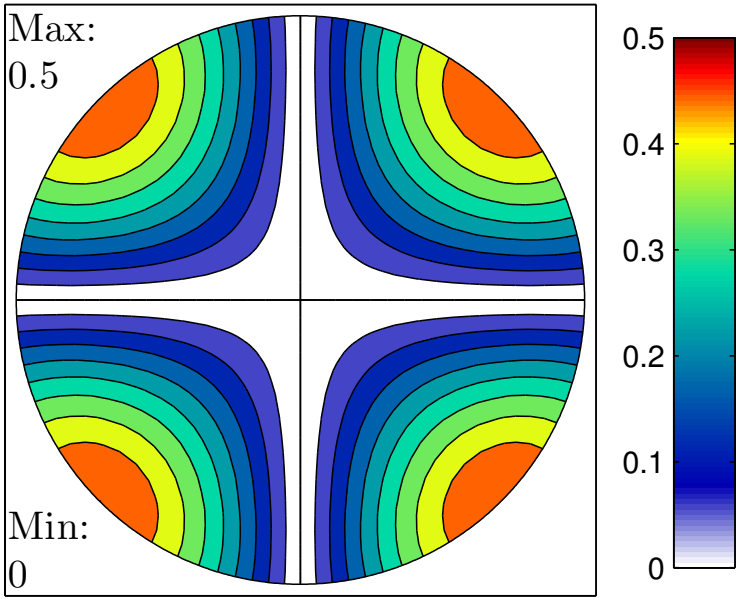
```
ST = SchmidTensor(n,b)
```

We can do this also for a list of directions

```
r = plotS2Grid('upper')
```

```
sigma = EinsteinSum(...
            tensor(r),1,...
            tensor(r),2)
```

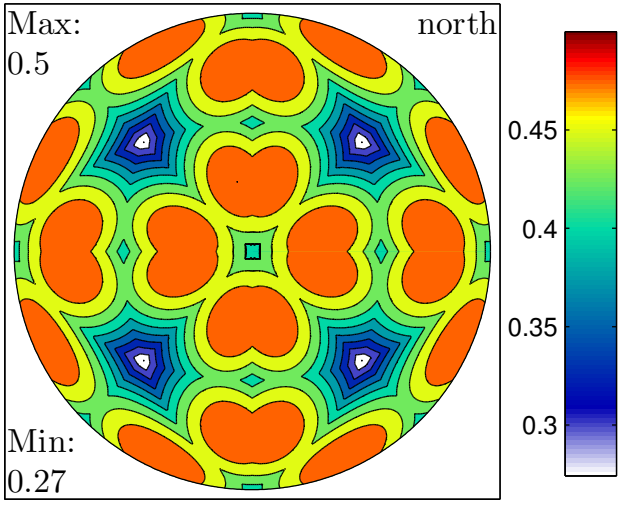
```
SF = EinsteinSum(...
       ST,[-1,-2],sigma,[-1,-2])

contourf(r,double(SF))
```



Finding the Active Slip Systems - Shortcut

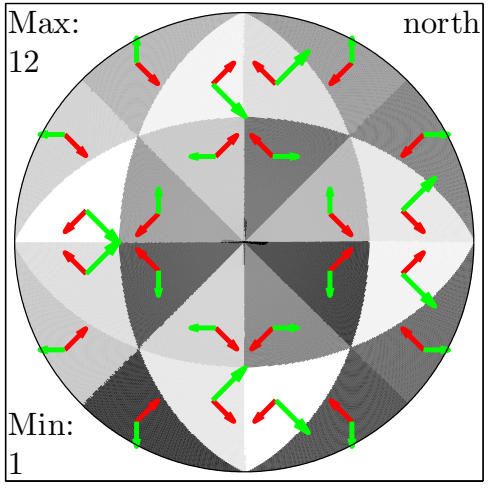
Usually there is more the one slip system in a crystal.
The maximum Schmid factor can be computed by a single command

```
[SFMax,n,b,tau,ind] = ...
  calcShearStress(sigma,n,b)
```



As usual it is applicable to a list of stress tensors

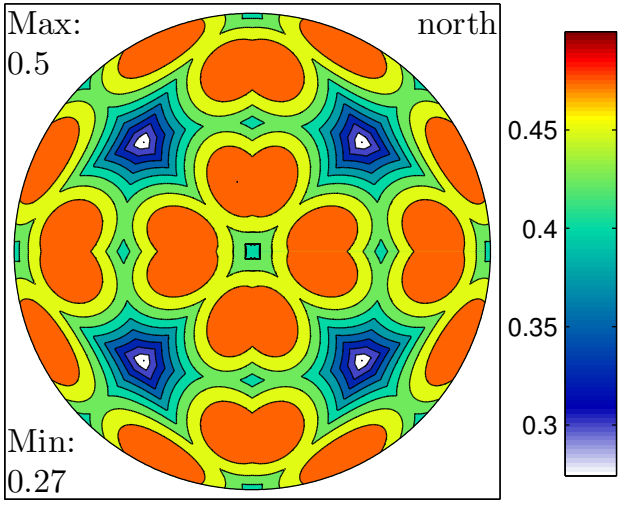
```
contourf(r,SFMax);
colorbar
```



Finding the Active Slip Systems - Shortcut

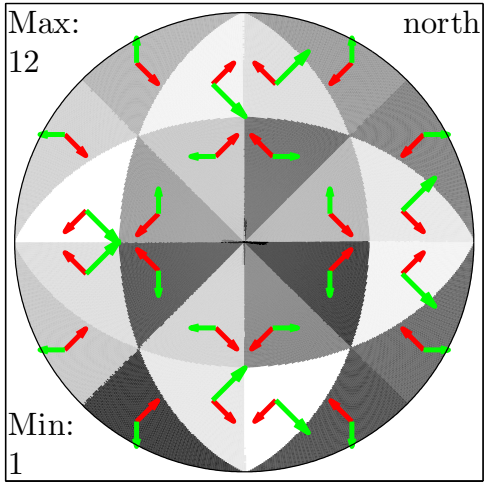
Usually there is more the one slip system in a crystal.
The maximum Schmid factor can be computed by a single command

```
[SFMax,n,b,tau,ind] = ...
   calcShearStress(sigma,n,b)
```



As usual it is applicable to a list of stress tensors

```
contourf(r,SFMax);
colorbar
```



Schmidfactor for grains

```
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
```

Schmidfactor for grains

```
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)
```

Schmidfactor for grains

```
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');
```

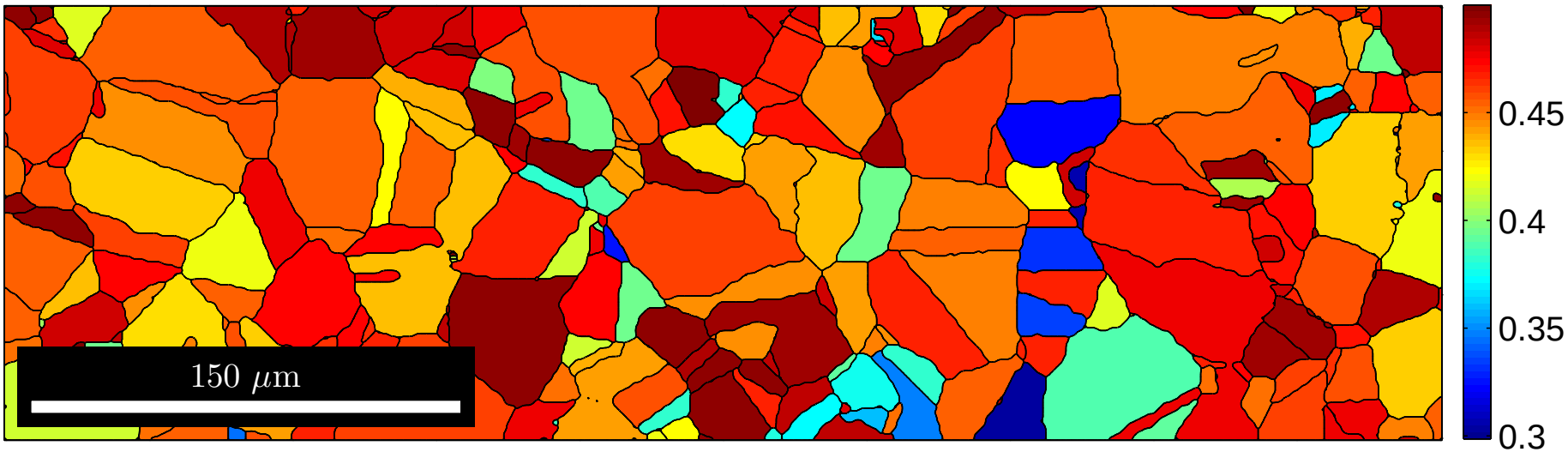
Schmidfactor for grains

```
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)
```



Schmidfactor for grains

```
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
```