Supplementary material:MATLAB®-codes

Roland Schierholz [a]*

[a] *Technische Universität Darmstadt Germany. E-mail: rolandschierholz@gmx.de*

1. Introduction

This material contains the MATLAB®-codes for review process and/or to be provided for interested readers. The codes are divided into different parts to display in shorter form for readability. So the position, where a part should be included, is marked with >> Part. Other calculations also repeat often so in this case only the comment line and subsequent ... are shown to imply that the similar calculation as already described above follows. Following abbrevations are used: twin operation (TO), Nanodomain (ND), nanodomain wall (NDW), microdomain wall (MDW).

2. MATLAB-code PART I
Calculation of twins

2.1. Tetragonal 90°-wall in (110)

```
function zolz=tetra(h1,k1,l1,h2,k2,l2)
% lattice parameters
a = 4.04; b = a; c = 4.14;
% lattice vectors in cartesian space
c=[c; 0; 0];
b=[0; 0; b];
a=[0; a; 0];
reciprocal lattice
ar = cross(b,c)/(dot(a,cross(b,c)));
...
```

```
% normal vector of (101)-domain wall
n=ar+cr
% rotation about [001] to align n||[110]
delta=atan((n(1)/n(2)-1)/(n(1)/n(2)+1));
rot1 = [cos(delta) -sin(delta) 0;...
        sin(delta) cos(delta) 0;...
        0 0 1];
% lattice of domain 1
a1=rot1*a;
...
% TO 2fold rotation about [110]
R110 = [0 1 0; 1 0 0; 0 0 -1];
% lattice of domain 2
...
% reciprocal lattice for domain 1 and 2
...
% coordinates in ZOLZ in this form
% !only for perpendicular base vectors!
x=[h1 k1 l1]/norm([h1 k1 l1]);
y=[h2 k2 l2]/norm([h2 k2 l2]);
% zone axis [uvw]
uvw = cross(x,y);
```

2.2. Rhombohedral 71°-wall in (110)

```
function zolz = rhombo110(h1,k1,l1,h2,k2,l2)
% lattice parameters (monoclinic set up)
a = [5.79]; b = [5.75]; c = [4.08];
beta = [90.56];
% domain 1 (100)m||(-1-10)c
a1 = [-a/sqrt(2)*sin(beta/180*pi);...
     -a/sqrt(2)*sin(beta/180*pi);...
      a * cos(beta/180*pi)];
b1 = [b/sqrt(2); -b/sqrt(2); 0];
c1 = [0; 0; c];
% 2fold rotation about [110]
R110 = [0 1 0; 1 0 0; 0 0 -1];
% lattice of domain 2
a2 = R110*a1; ...
% reciprocal lattice
...
% zone axis [uvw]
...

function zolz = rhombo100(h1,k1,l1,h2,k2,l2)
% lattice parameters in monoclinic setup
...
```

```
% n1 = normal of (-110)m
n1 = (-a1r+b1r)/norm(-a1r+b1r)
delta=acos(dot(n1,[1 0 0]'));
% rotation about [001] so (-110)m||(100)
r1 =[cos(delta) -sin(delta) 0;...
    sin(delta) cos(delta) 0;...
    0 0 1];
% lattice of domain 1
a1 = r1*a1;
...
% twofold rotation about [100]
R100 = [1 0 0; 0 -1 0; 0 0 -1];
% lattice of domain2
a2 = R100*a1;
...
% reciprocal lattice
...
% zone axis
...
```

3. Part II
Calculation of reflection positions in ZOLZ for Fig. 2, 3 and 4

```
% loops to calculate reciprocal
% lattice points up order h, k and l
m = 1;
for l = -5:5
 for h = -5:5
  for k = -5:5
   % C-centering monoclinic setting
   for n = -10:1:10
   if (n == h+k)
    % reflections in cartesian space
    r1 = h*a1r + k*b1r + l*c1r;
    r2 = h*a2r + k*b2r + l*c2r;
    % excitation error = r_||[uvw]
    s1 = dot(uvw',r1);
    s2 = dot(uvw',r2);
    % limit to ZOLZ
     if abs(s1) <= 0.05
     % position of r in uvw-zone
     x1 = dot([h1;k1;l1],r1)/...
```

```
      sqrt(h1.^2+k1.^2+l1.^2);
      y1 = dot([h2;k2;l2],r1)/...
      sqrt(h2.^2+k2.^2+l2.^2);
      else x1 = 0;
      y1 = 0;
      end
      if abs(s2) <= 0.05
      x2 = dot([h1 k1 l1],r2)/...
      sqrt(h1.^2+k1.^2+l1.^2);
      y2 = dot([h2 k2 l2],r2)/...
      sqrt(h2.^2+k2.^2+l2.^2);
      else x2 = 0;
      y2 = 0;
      end
     zolz(m,:) = [h k l x1 y1 x2 y2];
    m = m+1;
    end
   end
  end
 end
% plot
axis equal % equal aspect ratio
% limit for axis A^-1 (|g100|~0.25A^-1)
axis([-1.2 1.2 -1.2 1.2]); grid off;
set(gca,'ytick',[])
set(gca,'xtick',[])
% reflections of domain 1 (green circles)
plot(zolz(:,4),zolz(:,5),'o',
'MarkerEdgeColor',[0.17 0.51 0.34],...
'MarkerSize',6)
% reflections of domain 2 (yellow dots)
hold on; plot(zolz(:,6),zolz(:,7),'.',...
'MarkerEdgeColor',[0.87 0.49 0],...
'MarkerSize',6)
end
```

4. Part III: calculation of s for Fig. 5

4.1. Tetragonal 90°*-wall*

```
function dg=splitting_tetra()
m = 1
for dc = 0.0:0.005:0.15;
```

```
a = 4.0733-dc/2;
b = a;
c = 4.0733+dc;
covera = c/a
...
>> Part I
...
% loops to calculate s vs. c/a-ratio
% [001] zone axis
base1 = [1;0;0]; % cartesian space
base2 = [0;1;0]; % cartesian space
uvw = cross(base1,base2); % zone axis
% normalisation of [uvw]
uvw1 = uvw/norm(uvw);
ref1 = [1;0;-1]; % refl. of domain 1
ref2 = [-1;0;1]; % refl. of domain 2
r1 = ref1(1)*a1r...
        +ref1(2)*b1r...
        +ref1(3)*c1r;
r2 = ...
% direction of splitting in ZOLZ
ds1 = ((r2-r1)-...
        dot((r2-r1),uvw1)*uvw1)/norm(r1);
% value of s
s1 = norm((r2-r1)-...
        dot((r2-r1),uvw1)*uvw1)/norm(r1);
% generate matrix for plot s vs. c/a
s001(m,:) = [covera s1];
% [0-10] and other zone axis
...
m = m+1;
% large matrix for all incidences
dg=[s001(:,1) s001(:,2) s0-10(:,2) ...
        s0-11(:,2) s111(:,2) s1-11(:,2)];
% plot s vs. c/a-ratio for all zone
plot(dg(:,1),dg(:,2),dg(:,1),dg(:,3),...
         dg(:,1),dg(:,4),dg(:,1),dg(:,5),...
         dg(:,1),dg(:,6));
end



function dg=splitting_rhombo110()
m = 1;
for r=0:0.05:1;
alpha = (90-r)/180*pi;
```

```
alphadegree =90-r;
 a_r=4.0733;
% transformation to monoclinic lattice
a = 2*a_r*cos(alpha/2);
b = 2*a_r*sin(alpha/2);
c = a_r;
beta = acos(-sqrt(2)*(cos(alpha))/...
         (1+cos(alpha)));
>> Part I
...
>> Part III with for zones and refl. in Fig. 3
...
```

4.1.2. Rhombohedral 109°*-wall (100)* For rhombohedral (100)-domain walls s can be calculated in a similar way for the reflections indexed in Fig. 4. Only the base vectors, indices of reflections and the twin operation, according to Part I need to be adapted.

5. Part IV
Multidomain configurations

This is a very simple approach. Only geometrical stiff lattices are transformed by twin operations and the mismatch angles are calculated. The first type of angles given expresses the roughness of the microdomain wall (MDW). This is the angle between the surface normals of crystallographic planes that represent the MDW for in the different nanodomains (ND). The second angle gives the missing wedge that is left inside the microdomain. This is calculated as the angle between the normals of the crystallographic planes of the nanodomains, that ideally should be parallel, after a series of twin operations (TO).

5.1. Monoclinic {110}-mirror twins inside tetragonal domains

```
function omega = nanodomains4mm_011
% lattice parameters in monoclinic setup
```

```
...
% nanodomain with P1 b||[0 -11]; c||[1 0 0]
a1=[a*cos(beta);...
       1/sqrt(2)*a*sin(beta);...
       1/sqrt(2)*a*sin(beta)];
b1=[0; -b/sqrt(2); b/sqrt(2)];
c1=[c ; 0; 0];
% reciprocal lattice
...
% MDW1 (110)c (1-11)m
mdw1 = (a1r-b1r+c1r)/norm(a1r-b1r+c1r);
% normal of nanodomain wall
ndw1 = a1r/norm(a1r);
% twin operation (mirror)
ndw13 = [1-2*ndw1(1)^2...
        -2*ndw1(2)*ndw1(1)...
        -2*ndw1(3)*ndw1(1);...
        -2*ndw1(1)*ndw1(2)...
        1-2*ndw1(2)^2...
        -2*ndw1(3)*ndw1(2);...
    -2*ndw1(1)*ndw1(3)...
    -2*ndw1(2)*ndw1(3)...
    1-2*ndw1(3)^2];
% nanodomain with P3
a3=ndw13*a1;
b3=-ndw13*b1; % right hand system
c3=ndw13*c1;
% reciprocal lattice
...
% MDW 3 (110)c (-111)m
mdw3 = (-a3r+b3r+c3r)/norm(-a3r+b3r+c3r);
omega = 180/pi*acos(dot(mdw1,mdw3));
 end
```

5.2. Monoclinic {100}-mirror twins inside tetragonal domains

```
function omega = nanodomains4mm_010
% lattice parameters in monoclinic setup
...
gamma = atan(a/b);
% orientation with [110]m||z
a1=[0; a*cos(gamma); a*sin(gamma)];
b1=[0; -b*sin(gamma); b*cos(gamma)];
c1=[c*sin(beta);...
       1/sqrt(2)*c*cos(beta);...
       1/sqrt(2)*c*cos(beta)];
% reciprocal lattice
```

```
...
% MDW1 (110)c (1-11)m
n1 = a1r-b1r+c1r;
n1 = n1/norm(n1);
% normal of NDW 1|2
ndw1 = (a1r+b1r)/norm(a1r+b1r);
% twin operation (mirror)
...
% nanodomain 2
...
% MDW 2 (110)c (111)m
n2 = (a2r+b2r+c2r)/norm(a2r+b2r+c2r);
% normal of NDW 2|3
ndw2 = (a2r+b2r)/norm(a2r+b2r);
% twin operation (mirror)
...
% MDW3 (110)c (-111)m
n3 = (-a3r+b3r+c3r)/norm(-a3r+b3r+c3r);
% normal of NDW 3|4
ndw3 = (a3r+b3r)/norm(a3r+b3r);
% twin operation (mirror)
...
% MDW4 (110)c (-1-11)m
n4 = (-a4r-b4r+c4r)/norm(-a4r-b4r+c4r);
% NDW 4|1
ndw4 = (a4r+b4r)/norm(a4r+b4r);
% NDW 1|4
ndw14 = (-a1r+b1r)/norm(-a1r+b1r);
% missing wedge within microdomain
phi = 180/pi*acos(dot(ndw4,ndw14))
% angles between MDW sections
omega12 = 180/pi*acos(dot(n1,n2));
...
end
```

5.3. 90°*-rotational nanotwins inside tetragonal domains*

```
function omega = nanodomains4mm_001R
% lattice parameters in monoclinic setup
gamma = atan(a/b);
% orientation with c||[100]
% so twin operation is roation about [100]
a1=[0; a*cos(gamma); a*sin(gamma)]
b1=[0; -b*sin(gamma); b*cos(gamma)]
bn=-b1/norm(b1); % for rotation matrix
c1=[ c*sin(beta); 1/sqrt(2)*c*cos(beta); 1/sqrt(2)*c*cos(beta)];
% reciprocal lattice
```

```
...
% MDW1 (110)c
n1 = (a1r-b1r+c1r)/norm(a1r-b1r+c1r);
% twin operation fourfold rot. about [100]
R100 =[1 0 0; 0 0 1; 0 -1 0];
% nanodomain 2
a2=R100*a1;
...
% reciprocal lattice
...
% MDW2 (110)c
n2 = (a2r+b2r+c2r)/norm(a2r+b2r+c2r);
% nanodomain 3
...
% MDW3 (110)c
n3 = (-a3r+b3r+c3r)/norm(-a3r+b3r+c3r);
% nanodomain 4
...
% MDW4 (110)c
n4 = (-a4r-b4r+c4r)/norm(-a4r-b4r+c4r);
omega12 = 180/pi*acos(dot(n1,n2));
...
end
```

5.4. {110}-mirror nanotwins in rhombohedral domains

The programs are written for both $(100)_c$ and $(110)_c$ MDWs. The result not wanted should be commented.

```
function omega = nanodomains3m_S
% lattice in monoclinic setup
% nanodomain 1
a1=[-1/sqrt(2)*a*sin(beta);...
          -1/sqrt(2)*a*sin(beta);...
     a*cos(beta)];
b1=[b/sqrt(2); -b/sqrt(2); 0];
c1=[0; 0; c];
% reciprocal lattice
...
% MDW (110)c for ND1
% n1 = (-a1r)/norm(-a1r);
% MDW (100)c for ND 1
n1 = (-a1r+b1r)/norm(-a1r+b1r);
% NDW 1|2 (1-11)m-plane
ndw1 = (a1r-b1r+c1r)/...
          norm(a1r-b1r+c1r);
```

```
% NDW 1|3 (111)m−plane
ndw13 =(a1r+b1r+c1r )/...
        norm(a1r+b1r+c1r );
% mirror operation on 1=>2
 ...
% MDW 2 (110)c−plane
n2 = (−a2r+b2r+c2r )/...
        norm(−a2r+b2r+c2r );
% MDW 2 (100)c−plane
n2 = (c2r)/norm(c2r );
% normal of NDW 2|3
ndw2 = (a2r−b2r+c2r)/norm(a2r−b2r+c2r );
% ndw21 = (a2r+b2r+c2r)/norm(a2r+b2r+c2r );
% mirror operation 2=>3
 ...
% MDW (110)c for ND 3
% n3 = (−a3r−b3r+c3r)/norm(−a3r−b3r+c3r );
% MDW (100)c for ND 3
n3 = (−a3r−b3r)/norm(−a3r−b3r );
% NDW 3|1
ndw3 = (a3r−b3r+c3r)/norm(a3r−b3r+c3r );
% missing wedge
phi = 180/pi∗acos(dot(−ndw3,ndw13))
% angles between sections of MDW
omega12 = 180/pi∗acos(dot(n1,n2));
 ...
```

5.5. 120°*-rotational nanotwins*

```
function nanodomains3m_R()
% lattice in monoclinic setup
 ...
a1=[−1/sqrt(2)∗a∗sin(beta);...
        −1/sqrt(2)∗a∗sin(beta);...
     a∗cos(beta)];
b1=[b/sqrt(2); −b/sqrt(2); 0];
c1=[0; 0; c];
% reciprocal lattice
 ...
% normal of (−201)m plane (111)c
n = (−2∗ar+cr)/norm(−2∗ar+cr );
% [−101]m=[111]c
% rotation about [1−10]
r=[1/sqrt(2); −1/sqrt(2); 0];
% to aligne (−201)m||(111)
```

```
delta=-acos(dot(n,sd));
Rotation(r1,delta)
% nanodomain 1
a1=Rb*a;
...
% reciprocal lattice
...
n1=(-2*a1r+c1r)/norm(-2*a1r+c1r);
%MDW(110) for nd 1
mdw1=(-a1r)/norm(-a1r); % nd1
%MDW(100) for nanodomain 1
% mdw1=(-a1r+b1r)/norm(-a1r+b1r);
% threefold rotation for TO
sd =[1/sqrt(3); 1/sqrt(3); 1/sqrt(3)];
rho = 2*pi/3;
R3 = rotation(sd,2*pi/3)
% TO nd1=>nd2
    a2=R3*a1;
    ...
% reciprocal lattice
a2r = cross(b2,c2)/(dot(a2,cross(b2,c2)));
...
n2=(-2*a2r+c2r)/norm(-2*a2r+c2r);
%MDW(110) for nanodomain 2
mdw2 = (-a2r+b2r+c2r)/norm(-a2r+b2r+c2r);
%MDW (100) for nanodomain 2
% mdw2 = (c2r)/norm(c2r);
    a3=R3*a2;
    ...
% reciprocal lattice
...
%MDW (110) for nd 3
mdw3=(-a3r-b3r+c3r)/norm(-a3r-b3r+c3r);
%MDW (100) for nd3
% mdw3=(-a3r-b3r)/norm(-a3r-b3r);
% angles between the MDW normals
omega12 = 180/pi*acos(dot(mdw1,mdw2));
...
end
```