Appendix A.

Here the algorithm is presented in a pseudo-code version in a simplified Rietveld program. The function “background_calculation” delivers the background calculated as function of the the parameters of a polynom. From the function “Phase_calculation” the modelled diffractogram from austenite and martensite for Chromium Kα radiation is estimated. The LP factor and the structure factor are combined in “R”-values according to (ASTM, 2013).

The third program describes the whole Gibbs sampling algorithm used in the paper.

```
function [ybackground]=background_calculation(x_,CounterX,B,CounterP)
//calculates the background of the modelled diffractogram
// Input:
// x_:   2Theta values of the diffractogram (vector)
//CounterX: number of 2Theta values
// B: values of the parameters
//CounterP: number of polynoms
// Output:
// ybackground: values of the background (vector), same dimension as x_
For i=1 to CounterX   //loop: calculation of the background for every
        BKPOS=x_(1)  //starting point of 2Theta
        Ybi=0  //start value is zero
        For j=0 to CounterP-1  //loop: calculation of the polynom
                Ybi(i)=ybi(i)+B(j+1)*(x_(i)/BKPOS-1)^j  //calculation: starts with order 0 of the polynom
        End j
        ybackground(i)=ybi //calculated background value is saved to the output variable
End i
End function background_calculation
```

```
function [yPhase]=Phase_calculation(x_,CounterX,Kl,Phase)
// calculates the diffractogram for austenite (fcc) and martensite (bcc)
// Input:
// x_:   2Theta values of the diffractogram (vector)
//CounterX: number of 2Theta values
//Kl: wavelength (average of Kalpha1 and Kalpha2)
//Phase: parameters of austenite and martensite
// Output:
// yPhase: calculated diffractogram without background
yPhase=0  //start point is zero-

//first phase: austenite
a1=Phase(1).a  //lattice parameter austenite
//Bragg: lambda=2*d*sinTheta
//first phase, first peak austenite: hkl: (1 1 1)
d11=a1/sqrt(3)  //d-value from first phase, first peak
sinTheta11=Kl/2/d11  //Bragg equation for first phase, first peak
theta2_11=2*asin(sinTheta11)*360/2/pi  //2Theta for first phase, first peak
//first phase, second peak austenite: hkl: (2 0 0)
d12=a1/sqrt(4)  //d-value from first phase, second peak
sinTheta12=Kl/2/d12  //Bragg equation for first phase, second peak
theta2_12=2*asin(sinTheta12)*360/2/pi  //2Theta for first phase, second peak
//first phase, third peak austenite: hkl: (2 2 0)
d13=a1/sqrt(8)  )  //d-value from first phase, third peak
sinTheta13=Kl/2/d13  //Bragg equation for first phase, third peak
```

```
theta2_13=2*asin(sinTheta13)*360/2/pi  //2Theta for first phase, third peak
Phase(1).theta2(1)=theta2_11
Phase(1).theta2(2)=theta2_12
Phase(1).theta2(3)=theta2_13
//second phase: martensite
a2=Phase(2).a  //lattice parameter martensite
//Bragg: lambda=2*d*sinTheta
//second phase, first peak martensite: hkl: (1 1 0)
d21=a2/sqrt(2)  //d-value from second phase, first peak
sinTheta21=Kl/2/d21  //Bragg equation for second phase, first peak
theta2_21=2*asin(sinTheta21)*360/2/pi  //2Theta for second phase, first peak
//second phase, second peak martensite: hkl: (2 0 0)
d22=a2/sqrt(4)  //d-value from second Phase, second peak
sinTheta22=Kl/2/d22  //Bragg equation for second phase, second peak
theta2_22=2*asin(sinTheta22)*360/2/pi  //2Theta for second phase, second peak
Phase(2).theta2(1)=theta2_21
Phase(2).theta2(2)=theta2_22

//calculation difractogram austenite and martensite
for i=1 to Phase(1).number  //i=1 (austenite) and i=2 (martensite)
   for j=1 to Phase(i).peaks  //j=1..3 peaks for austenite, j=1..2 peaks for martensite
         FWHM=Phase(i).FWHM(j)  //Full Width at Half Maximum
         I=Phase(i).Int*Phase(i).R(j)  //I: global intensity for austenite and martensite
         xk=Phase(i).theta2(j)  //2Theta: Position of maximum
         for k=1 to CounterX  //2Theta values diffractogram
               //calculation Gauss-Peak
```

```
                    yPhase(k)=yPhase(k)+I*(4*log(2)/pi)^0.5/FWHM*exp(-4*log(2)*(x_(k)-
xk)^2/FWHM^2);
            end k  //2Theta values diffractogram
    end j  //peaks
end i  //phases
end function Phase_Calculation
```

```
//Rietveld with Gibbssampling
//example for fcc austenite and bcc martensite, X-rays: Chromium Kalpha
X_  //2Theta values of the diffractogram
Y_ //Intensity values of the diffractogram
CounterX  //number of 2Theta-values
Kl=2.29108  //wavelength in Angstrom: average Kalpha1 und Kalpha2 Chromium
CounterP //Background polynom with e.g. 3 coefficients
B(1)  //define starting value first coefficient of the background polynom
B(2)  //define starting value second coefficient of the background polynom
B(3)  //define starting value third coefficient of the background polynom
BI(1)  //define: change from run to run of first coefficient of the background polynom
BI(2)  // define: change from run to run of second coefficient of the background polynom
BI(3)  // define: change from run to run of third coefficient of the background polynom
tinySqur=0.0001  //smallest acceptable variance (positive value)
tinyFWHM=0.001  //smallest acceptable FWHM of the peaks (positive value)
tinyI=0.0001;

Phase(1).Anzahl=2  //two phases
//phase1 - austenite
Phase(1).a=3.5987  //lattice parameter austenite: start value
Phase(1).aI=0.001  //change from run to run of lattice parameter austenite
Phase(1).Int=0.2  //global intensity austenite: start value
Phase(1).IntI=0.001  //change from run to run of global intensity austenite
Phase(1).peaks=3  //number of austenite Peaks
Phase(1).R(1)=75.24  //first austenite Peak: L*P*Structure factor for austenite for chromium Ka
```

```
Phase(1).R(2)=34.78  //second austenite Peak: L*P*Structure factor for austenite for chromium Ka

Phase(1).R(3)=47.88  //third austenite Peak: L*P*Structure factor for austenite for chromium Ka

Phase(1).FWHM(1)=0.3928  //first austenite peak:start value FWHM

Phase(1).FWHMI(1)=0.1  //change from run to run of first austenite peak FWHM

Phase(1).FWHM(2)=0.7333  //second austenite peak: start value FWHM

Phase(1).FWHMI(2)=0.1  //change from run to run of second austenite peak FWHM

Phase(1).FWHM(3)=1.6412  //third austenite peak: start value FWHM

Phase(1).FWHMI(3)=0.1  //change from run to run of third austenite peak FWHM

//phase2 - martensite

Phase(2).a=2.8678  //lattice parameter martensite: start value

Phase(2).aI=0.003  //change from run to run of lattice parameter martensite

Phase(2).Int=3.069  //global intensity martensite: start value

Phase(2).IntI=0.01  //change from run to run of global intensity austenite

Phase(2).peaks=2  //number of martensite Peaks

Phase(2).R(1)=101.5  //first martensite Peak: L*P*Structure factor for austenite for chromium Ka

Phase(2).R(2)=20.73  //second martensite Peak: L*P*Structure factor for austenite for chromium Ka

Phase(2).FWHM(1)=0.3061  //first martensite peak: start value FWHM

Phase(2).FWHMI(1)=0.02 //change from run to run of first martensite peak FWHM

Phase(2).FWHM(2)=1.2328  //second martensite peak: start value FWHM

Phase(2).FWHMI(2)=0.02  //change from run to run of second martensite peak FWHM


for runs=1 to 10000 //number of runs (Monte Carlo)

yPhase=Phase_calculation(x_,CounterX,KI,Phase)  //calculation model diffractogram


//squ0: calculation of the mean variance: equation 7

squ0=0;
```

```
ybackground=background_calculation(x_CounterX,B,CounterP)
yPhase=Phase_calculation(x_,CounterX,Kl,Phase)+ybackground;
squ0=squ0+sum((y_-yPhase).^2);
squ0=squ0/CounterX;
// end calculation of the mean variance
          for indexz=1 to CounterP+5+4  //loop for Gibbs sampling, variation of all parameters
            Br=B  //new state for background polynom parameters
            Phaser=Phase  //new state of parameter of phases
            squr=squ  //new state of parameter of variance
             if indexz<=CounterP  //Gibbs sampling background polynom
               //randn(1)…randomized value from gauss distribution with standarddeviation=1, mean=0
              Br(indexz)=B(indexz)+Bl(indexz)*randn(1)  //new state background polynom
            End
            //Gibbs sampling phases
            //some values should be positive, e.g. FWHM – abs(x) calculates the absolute value from x
            //austenite
             case indexz=CounterP+1: Phaser(1).a=abs(Phase(1).a+randn(1,1)*Phase(1).al) //aust. a
             case indexz=CounterP+2: Phaser(1).Int=abs(Phase(1).Int+randn(1,1)*Phase(1).Intl)
//intensity austenite
             case indexz=CounterP+3:
Phaser(1).FWHM(1)=abs(Phase(1).FWHM(1)+randn(1,1)*Phase(1).FWHMl(1))  //first peak
             case indexz=CounterP+4:
Phaser(1).FWHM(2)=abs(Phase(1).FWHM(2)+randn(1,1)*Phase(1).FWHMl(2))  //second peak
             case indexz=CounterP+5:
Phaser(1).FWHM(3)=abs(Phase(1).FWHM(3)+randn(1,1)*Phase(1).FWHMl(3))  //second peak
             //martensite
```

```
            case indexz=CounterP+5+1: Phaser(2).a=abs(Phase(2).a+randn(1,1)*Phase(2).aI)
//martensite a
            case indexz=CounterP+5+2: Phaser(2).Int=abs(Phase(2).Int+randn(1,1)*Phase(2).IntI)
//intensity martensite
            case indexz=CounterP+5+3:
Phaser(2).FWHM(1)=Phase(2).FWHM(1)+randn(1,1)*Phase(2).FWHMI(1)
            case indexz=CounterP+5+4:
Phaser(2).FWHM(2)=abs(Phase(2).FWHM(2)+randn(1,1)*Phase(2).FWHMI(2))


        //probability new state
        ybackgroundr=background_calculation(x_,CounterX,Br,CounterP)  //calculation background
        yPhaser=Phase_calculation(x_,CounterX,KI,Phaser)  //calculation diffractogram new state
        yr=ybackgroundr+yPhaser  //calculation model diffractogram


       sumdifference=0 //starts with zero
        for k=1 to CounterX
          sumdifference=sumdifference+(y_(k)-yr(k))^2
        end
        f1=((-1/2/squ0*sumdifference))  //approximation: squ0 instead of squ0r


        %probability old state
        ybackground=background_calculation(x_,CounterX,B,CounterP)  //calculation background
        yPhase=Phase_calculation(x_,CounterX,KI,Phase)  //calculation diffractogram old state
        y=ybackground+yPhase  //calculation model diffractogram
       sumdifference=0 //starts with zero
        for k=1 to CounterX
```

```
        sumdifference=sumdifference+(y_(k)-y(k))^2
      end
      f2=((-1/2/squ0*sumdifference))  //squ0 = variance
      prob = f1-f2; //division of probabilities is consistent with a subtraction of the logarithm of
probabilities
      prob=exp(prob)
      if prob=inf  //if there is a problem with the exp-function (e.g. overflow)
        prob=1.2;
      end

      if prob >1 //Alpha>1; take the new state  (eq.9)
       squ=squr;B=Br;h=hr;
       Phase=Phaser;
      elseif rand(1,1) > (1-prob)  //take it with a probability (according to eq. 9)
       squ=squr;B=Br;h=hr;
       Phase=Phaser;
      end
      Phase  //save results
      B  //save results
      end //loop: indexz
end //loop: runs
```