```
(* +---------------------------------- evaluate_density.nb ----------------------------+ *)
(* +                                                                                   + *)
(* +           evaluate_density.nb calculates for a data set with xyz coordinates     + *)
(* +           the local density of "atoms" around each site.                         + *)
(* +           evaluate_order.nb creates a three dimensional temperature map for the  + *)
(* +           best match of each "atom" in then xyz data file with a database item   + *)
(* +           The "lower" the temperature in the map the better the match.            + *)
(* +                                                                                   + *)
(* +           INPUT:  xyz coodinates of the cluster                                  + *)
(* +           OUTPUT: Graphics file with a color map that shows the density        + *)
(* +                   for each "atom" in the xyz data file.                          + *)
(* +           USAGE:  (1) Delete All Output (from the 'Cell' menu)                    + *)
(* +                   (2) Fill the section between 'BEGIN USER INPUT' and             + *)
(* +                      'END USER INPUT'                                             + *)
(* +                   (3) Evaluate Notebook (from the'Evaluate' menu)                + *)
(* +           DEPENDENCIES: None                                                    + *)
(* +                                                                                   + *)
(* +           AUTHOR:  L. Houben, Weizmann Institute of Science                      + *)
(* +                    lothar.houben(at)weizmann.ac.il                               + *)
(* +           COPYRIGHT: This software is licensed under the GNU GENERAL PUBLIC      + *)
(* +                      LICENSE Version 3                                          + *)
(* +-------------------------------------------------------------------------------+ *)
(* +--------------------------- BEGIN USER INPUT ------------------------------------+ *)
(* +-------------------------------------------------------------------------------+ *)
(* baseDir is the base folder for program input and output.                           *)
(* dataDir is a subfolder that holds the data file.                                   *)
(* outDir is a subfolder for the program output files.                                  *)
(* Please make sure that the folders exist.                                           *)
baseDir = StringJoin[$HomeDirectory, "/Desktop/diOPA"];
dataDir = "data";
outDir = "out";
(* datafileName is name of the data file with the cluster xyz coordinates *)
datafileName = "Au-crystal.cel";
Nfcc = {1.6, 18.};(* shell radius in units of the nearest neighbour distance, number of nearest neighbours in the shell in the bulk structure *)
(* Nfcc[[1]] is the cutoff in units of the nearest neighbour distance for the density calculation, i.e. the resolution of the density map *)
(* Nfcc[[2]] is required to normalize the result to the bulk density of the material *)
(* Note that Nfcc[[2]] depends on the symmetry of the coordination *)
(*                                                                                    *)
(* +-------------------------------------------------------------------------------+ *)
(* +--------------------------- END USER INPUT --------------------------------------+ *)
(* +-------------------------------------------------------------------------------+ *)
```

In[84]:=
```
celfile = StringJoin[baseDir, "/", dataDir, "/", datafileName]
```

Out[84]=
```
/Users/lothar/Desktop/diOPA/data/Au-crystal.cel
```

In[85]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
(* (3) define output files                                                                                                                                      *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
```

In[86]:=
```
graphfileprefix = StringJoin[baseDir, "/", outDir, "/", FileBaseName[datafileName], "_Density"]
logfilename = StringJoin[baseDir, "/", outDir, "/", FileBaseName[datafileName], "_Density.log"]
```

Out[86]= /Users/lothar/Desktop/diOPA/out/Au-crystal_Density

Out[87]= /Users/lothar/Desktop/diOPA/out/Au-crystal_Density.log

In[88]:=
```
(* define plot parameters                                          *)
axespos = {0, 0, 0};
viewvc = {50, -300, 100};
imsz = {512, 512};
axstyle = {Thick, Thick, Thick};
labSz = 14; (* set to zero if no labels, otherwise 14 or alike *)
colscalelabSz = 14;
ImMagfact = 1.5;
plotopac = 0.5; (* bubble plot opacity *)
```

In[95]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
(*                                                                                                                                                                   *)
(* END PARAMETER DEFINITION                                                                                                                                          *)
(*                                                                                                                                                                   *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
```

In[96]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
(*                                                                                                                                                                   *)
(* START FUNCTION DEFINITION                                                                                                                                          *)
(*                                                                                                                                                                   *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
```

In[97]:=
```
(* Function for Reading a celfile *)
(* J. Barthel, RWTH-Aachen University, Germany *)
(* Email: ju.barthel-at-fz-juelich.de *)
(* 28.08.2015 *)
(*  *)
(* Reads super cell structure data in CEL file format from *)
(* The specified file and stores the information in global arrays: *)
(*   celdims = dimensions of the super-cell  *)
(*   celangs = angles between the super-cell axes  *)
(*   natoms  = number of atoms in the super-cell *)
(*  *)
ReadCelFile[filename_] := Block[{celstrs},
  (* - init *)
  Clear[celdims, celangs, celatms, natoms];
  celdims = {0., 0., 0.}; celangs = {90., 90., 90.}; celatms = {}; natoms = 0;
  (* - read data from file *)
  celstrs = ReadList[filename, "String"];
  (* - analyse / extract numeric data *)
  celdims = Take[ReadList[StringToStream[celstrs[[2]]], "Number"], {2, 4}];
  celangs = Take[ReadList[StringToStream[celstrs[[2]]], "Number"], {5, 7}];
  celatms = Table[
    ReadList[StringToStream[celstrs[[i]]], Join[{"Word"}, Table["Number", {8}]]][[1]]
    , {i, 3, Length[celstrs] - 1}];
  natoms = Length[celatms];
  (* - report *)
  Print["- INPUT  CEL file: ", InputForm[filename]];
  Print["- OUTPUT (celdims) dimensions  (a,b,c)  [nm]: ", celdims];
  Print["- OUTPUT (celangs) axes angles (α,β,γ) [deg]: ", celangs];
  Print["- OUTPUT (natoms)  number of atoms          : ", natoms];
  If[natoms > 0,
   Print["- OUTPUT (celatms[[1]]) first atom          : ", celatms[[1]]];
  ];
  If[natoms > 1,
   Print["- OUTPUT (celatms[[", natoms, "]]) last  atom          : ", celatms[[natoms]]];
  ];
 ]
```

In[98]:=
```
QuickDistance[model_, nlist_] := Block[{SortedEntry = {}, tmplist = model, selected = {}, x = {}},
  (* - init *)
  Clear[EuclDist];
  EuclDist = 0;
  (* - analyse / extract numeric data *)
  For[k = 1, k <= coordination, k++,
   x = Nearest[tmplist → Automatic, nlist[[k]], 1];
   selected = N[tmplist[[x[[1]]]]];
   AppendTo[SortedEntry, selected]; tmplist = Drop[tmplist, x]
  ];
  EuclDist = EuclideanDistance[nlist, SortedEntry];
 ]
```

In[99]:=
```
GetNeighbours[atomindex_, nneighbours_] := Block[{k = {}},
  (* Extract atom atomindex and evaluate its n nearest neighbours *)
  (* the function nearest returns the test element itself, therefore we start with n+1 neighbours *)
  (* centre the list around the selected atom, the selected atom is the origin of the coordinate system *)
  (* *)
  (* Initialize global variable: neighbours , type: list *)
  Clear[neighbours];
  neighbours = {};
  (* *)
  k = Nearest[q, q[[atomindex]], nneighbours + 1];
  neighbours = k[[2 ;; nneighbours + 1]];
  neighbours[[All, 1]] = neighbours[[All, 1]] - k[[1, 1]];
  neighbours[[All, 2]] = neighbours[[All, 2]] - k[[1, 2]];
  neighbours[[All, 3]] = neighbours[[All, 3]] - k[[1, 3]];
 ]
```

In[100]:=
```
GetDistMeas[atomindex_] := Block[{DistMap = {}},
  (* Calculate a distance measure to the closest model item in the DB *)
  (* *)
  (* Returns a global variable: Dist, type: list *)
  (* Dist = {atomnr, DBindex,{angle,angle,distance}} *)
  (* Example: *)
  (* In:   GetDistMeas[41]; Dist *)
  (* Out: {41,256,{π/5,π/10,0.468556698631839}} *)
  Clear[Dist];
  Dist = {};
  (* *)
  (* Create a distance map *)
  (* Here we need permutations because the list of nearest neighbours is not sorted *)
  (* Finding the minimum over all possible permutations is the right way to go, yet very slow *)
  (* therefore we sort the list of data base model atoms *)
  (* for each atom in the neighbour list we search for the closest in the model *)
  (* note that this gives the minimum euclidian distance is achieced only if model and neighbour list are close *)
  (* there might be a solution with a better compromise if the match is not close *)
  GetNeighbours[atomindex, coordination]
   For[ind = 1, ind ≤ Length[ModelDB], ind++, QuickDistance[ModelDB[[ind, 3]], neighbours]; AppendTo[
     DistMap, {ModelDB[[ind, 2, 1]], ModelDB[[ind, 2, 2]], EuclDist}]];
  WhereMin = Ordering[DistMap[[All, 3]], 1];
  Dist = {atomindex, WhereMin[[1]], DistMap[[WhereMin]][[1]]};
]
```

In[101]:=
```
(* BoundingBox *)
(* returns box coordinates around the centre *)
(* in a list variable cutoff *)
(* INPUT: xyz_= {{x1,y1,z1},{x2, y2, z2}, ...}  *)
(*        margin = margin size to add               *)
(*                                  *)
(* OUTPUT: bbox ={{xmin-margin,xmax+margin},{ymin-mrgin,ymax+margin}{zmin-margin,zmax+margin}}  *)
BoundingBox[xyz_, margin_] := Block[{tmp},
  (* - init *)
  Clear[bbox];
  bbox = {{0, 0}, {0, 0}, {0, 0}};
  bbox[[1, 1]] = Min[xyz[[All, 1]]] - margin;
  bbox[[2, 1]] = Min[xyz[[All, 2]]] - margin;
  bbox[[3, 1]] = Min[xyz[[All, 3]]] - margin;
  bbox[[1, 2]] = Max[xyz[[All, 1]]] + margin;
  bbox[[2, 2]] = Max[xyz[[All, 2]]] + margin;
  bbox[[3, 2]] = Max[xyz[[All, 3]]] + margin;
  axespos = {bbox[[1, 1]], bbox[[2, 1]], bbox[[3, 1]]};
  Print["- BoundingBox         : ", bbox];
  Print["- Axes Position       : ", axespos];
 ]
```

In[102]:=
```
(* CentreBox *)
(* Centres xyz around 0 *)
(* in a list variable cutoff *)
(* INPUT: xyz_= {{x1,y1,z1},{x2, y2, z2}, ...}  *)
(*                                              *)
(*                                  *)
(* OUTPUT: centrelist = list with vectors centred around 0 *)
CentreBox[xyz_] := Block[{tmp},
  (* - init *)
  Clear[centrelist];
  centrelist = xyz;
  centrelist[[All, 1]] -= Min[xyz[[All, 1]]] + 0.5 * (Max[xyz[[All, 1]]] - Min[xyz[[All, 1]]]);
  centrelist[[All, 2]] -= Min[xyz[[All, 2]]] + 0.5 * (Max[xyz[[All, 2]]] - Min[xyz[[All, 2]]]);
  centrelist[[All, 3]] -= Min[xyz[[All, 3]]] + 0.5 * (Max[xyz[[All, 3]]] - Min[xyz[[All, 3]]]);
  Print["- Centred List "];
 ]
```

In[103]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
(*                                                                                                                                                                               *)
(* END FUNCTION DEFINITION                                                                                                                                                      *)
(*                                                                                                                                                                               *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
```

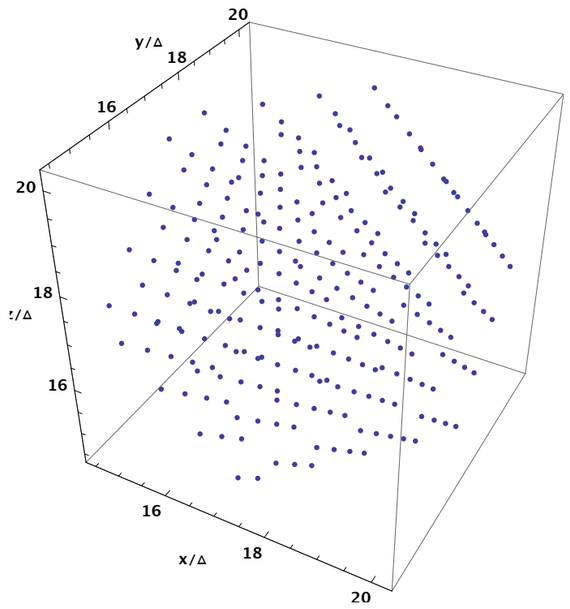
In[104]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
(*                                                                                                                                                                               *)
(* START PROCESSING                                                                                                                                                             *)
(*                                                                                                                                                                               *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
```

In[105]:=
```
(* Open log file *)
logfile = OpenWrite[logfilename];
WriteString[logfile, "Filename:", celfile, "\n"];
```

In[107]:=
```
ReadCelFile[celfile];
Clear[coordatms, f, q];
(* Extract relative atomcoordinates only *)
coordatms = celatms[[All, {2, 3, 4}]];
(* multiply atom relative coordinates with cell dimensions *)
(* multiply atom relative coordinates with cell dimensions *)

f[x_] := {celdims[[1]] * x[[1]], celdims[[2]] * x[[2]], celdims[[3]] * x[[3]]}
q = Map[f, coordatms];
(* Plot atoms *)
ListPointPlot3D[q, BoxRatios → {1, 1, 1}, PlotStyle → PointSize[0.01], AxesLabel → {"x/Δ", "y/Δ", "z/Δ"}, Lighting → "Neutral", LabelStyle → Directive[Bold]]
WriteString[logfile, "Number of atoms:", ToString[Length[q]], "\n"];
```

```
- INPUT  CEL file: "/Users/lothar/Desktop/diOPA/data/Au-crystal.cel"
- OUTPUT (celdims) dimensions  (a,b,c)  [nm]: {34.655, 34.655, 34.655}
- OUTPUT (celangs) axes angles (α,β,γ) [deg]: {90., 90., 90.}
- OUTPUT (natoms)  number of atoms          : 260
- OUTPUT (celatms[[1]]) first atom          : {Au, 0.4428, 0.495314, 0.584261, 1., 0.005, 0.1, 0.1, 0.1}
- OUTPUT (celatms[[260]]) last  atom          : {Au, 0.5572, 0.504686, 0.415739, 1., 0.005, 0.1, 0.1, 0.1}
```

Out[112]=




```
In[114]:= (* Color Map Definitions *)
          colortbl = "Rainbow";
          gamma = 1.6; (* gama factor to stretch the contrast *)
          ContrastEnhancement = 1.;
          colnorm[x_] := ((x - CScaleMin) / (CScaleMax - CScaleMin))^gamma; (* color normailzation, map a ramge of values CMin..CMaxx to 0..1 *)
          revcolnorm[x_] := (1 - (x - CScaleMin) / (CScaleMax - CScaleMin))^gamma; (* inverted color normailzation map a ramge of values CMin..CMaxx to 0..1 *)

In[119]:=

In[120]:= (* +------------------------------------------------------------------------------------------------------------------------------------------------ + *)
          (* DENSITY MAP calculation                                                                                                                        *)
          (* +------------------------------------------------------------------------------------------------------------------------------------------------ + *)
          (* INPUT: q=atomlist  *)

In[121]:= (* use the cutoff list and calculate for each point of it the euclidian distances to the points in the reference list *)
          (* this will give Length[cutofflist] times Length[xyz] values *)
          distances = With[{tr = Transpose[q]}, Function[point, Sqrt[Total[(point - tr)^2]]] /@ q]; // AbsoluteTiming

Out[121]= {0.005235, Null}
```

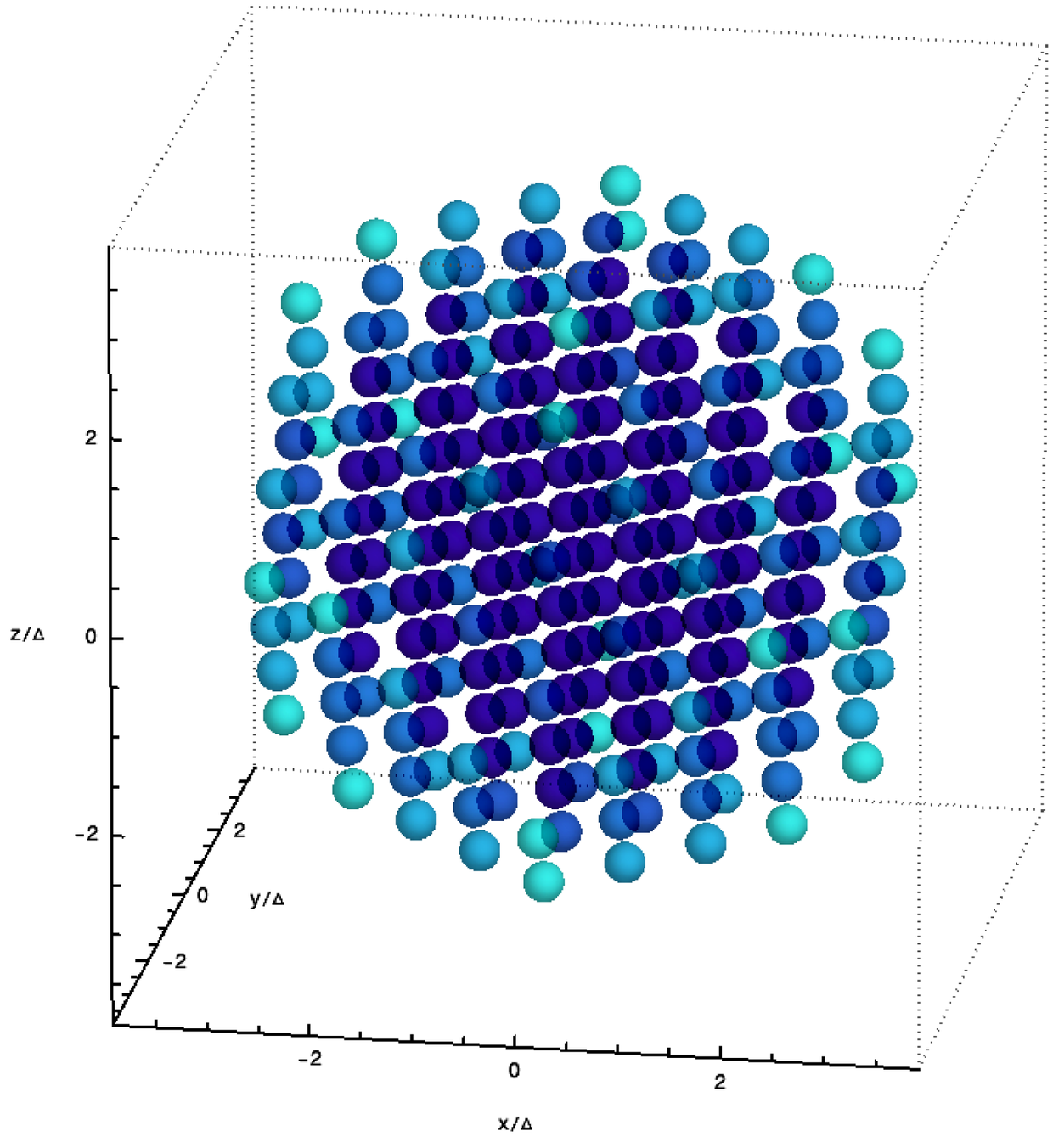
In[122]:=
```
ddata = {};
For[l = 1, l ≤ Length[q], l++, neighbourcount = BinCounts[distances[[l]], Nfcc[[1]]]; AppendTo[
   ddata, {q[[l, 1]], q[[l, 2]], q[[l, 3]], (neighbourcount[[2]] - 1) / Nfcc[[2]]}]];
CentreBox[ddata]
ddata = centrelist;
BoundingBox[ddata, 1]
CScaleMin = 0;
CScaleMax = 1;
CMean = Mean[ddata[[All, 4]]];
CMedian = Median[ddata[[All, 4]]];
CMin = Min[ddata[[All, 4]]];
CMax = Max[ddata[[All, 4]]];
```

```
- Centred List

- BoundingBox         : {{-3.92006, 3.92006}, {-3.92235, 3.92235}, {-3.92006, 3.92006}}

- Axes Position       : {-3.92006, -3.92235, -3.92006}
```

In[133]:=
```
colortbl = "BlueGreenYellow";
colddata = ddata;
colddata[[All, 4]] = Map[revcolnorm, ddata[[All, 4]]];
ldp1 = ListPointPlot3D[List /@ Most /@ colddata, PlotRange → bbox, BoxStyle → Directive[Thick, Dotted],
   AxesOrigin → axespos, AxesLabel → {"x/Δ", "y/Δ", "z/Δ"}, Lighting → "Neutral", AxesStyle → axstyle, BoxRatios → {1, 1, 1}, ViewCenter → {0.5, 0.5, 0.5},
   LabelStyle → Directive[Bold, FontFamily → "Helvetica", FontSize → labSz], PlotStyle → ({Opacity[plotopac], AbsolutePointSize[1], ColorData[colortbl][(# + 0) / 1]} & /@ Last /@ colddata),
   ViewCenter → {0.5, 0.5, 0.5}, ViewVector → viewvc, ViewVertical → {0, 0, 1}, ImageSize → imsz * ImMagfact];
ldp2 = ldp1 /. Point[x__] :> (Sequence@{EdgeForm[], Lighting → "Neutral", Sphere[#, .2]} &@({x} /. {{a_, b_, c_}} :> {{a, b, c}}));
```

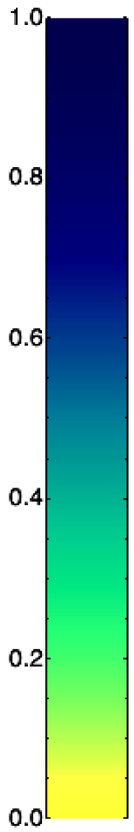
In[138]:=
```
ldp2adj = ImageAdjust[ldp2, ContrastEnhancement]
```

Out[138]=



In[139]:=
```
colourBar3 = DensityPlot[(1 - v)^gamma, {u, 0, 1}, {v, CScaleMin, CScaleMax}, ColorFunction → ColorData[colortbl], Frame → {False, True},
   LabelStyle → Directive[Bold, FontFamily → "Helvetica", FontSize → colscalelabSz], AspectRatio → 10, PlotRangePadding → 0, ImagePadding → 25, ImageSize → imsz, FrameTicks → {None, Automatic}];
colourBar3adj = ImageAdjust[colourBar3, ContrastEnhancement]
```

Out[140]=



In[141]:=
```
Print["- Average density (% of bulk): ", CMean]
Print["- Median density  (% of bulk): ", CMedian]
Print["- Minimum density (% of bulk): ", CMin]
Print["- Maximum density (% of bulk): ", CMax]
Print["- Contrast cut off (Min,Max): (", CScaleMin, ",", CScaleMax, ")"]
```

```
- Average density (% of bulk): 0.767949

- Median density  (% of bulk): 0.722222

- Minimum density (% of bulk): 0.444444

- Maximum density (% of bulk): 1.

- Contrast cut off (Min,Max): (0,1)
```

In[146]:=
```
WriteString[logfile, "- Average density (% of bulk): ", ToString[CMean], "\n"];
WriteString[logfile, "- Median density  (% of bulk): ", ToString[CMedian], "\n"];
WriteString[logfile, "- Minimum density (% of bulk): ", ToString[CMin], "\n"];
WriteString[logfile, "- Maximum density (% of bulk): ", ToString[CMax], "\n"];
WriteString[logfile, "- Contrast cut off (Min,Max): (", ToString[CScaleMin], ",", ToString[CScaleMax], ")", "\n"];
```

In[151]:=
```
(* Combine the order parameter map and the colour scale bar into one graphics file for file export *)
exportgraph = Graphics[{Inset[colourBar3adj, {2, 0.}, Center, 1.5], Inset[ldp2adj, {0, 0.5}, Center, 3]}, PlotRange → 3, ImageSize → 1024];
goutfile = StringJoin[graphfileprefix, ".png"]
Export[goutfile, exportgraph, "AllowRasterization" → True, ImageSize → imsz * ImMagfact, ImageResolution → 600]
```

Out[152]= /Users/lothar/Desktop/diOPA/out/Au-crystal_Density.png

Out[153]= /Users/lothar/Desktop/diOPA/out/Au-crystal_Density.png

In[154]:=
```
Close[logfile];
```

In[155]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
(*                                                                                                                                                                               *)
(* END PROCESSING                                                                                                                                                                *)
(*                                                                                                                                                                               *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+ *)
```