In[799]:=
```
(* +---------------------------------- evaluate_order.nb -----------------------------+ *)
(* +                                                                                  + *)
(* +           evaluate_order.nb calculates for a data set with xyz coordinates       + *)
(* +           the match of a given coordination, i.e. the distance norm against     + *)
(* +           rotation variants in a database file                                  + *)
(* +           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:  Database file, created with create_db                          + *)
(* +                   xyz coodinates of the cluster                                  + *)
(* +           OUTPUT: A text file that contains the distance norm between the        + *)
(* +                   database coordination and the coordination of each "atom"      + *)
(* +                   in the xyz data file.                                          + *)
(* +                   A graphics file with a temperature map that shows the          + *)
(* +                   distance order parameters in a three-dimensional list plot.    + *)
(* +           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                                                     + *)
(* +           NOTES:  The calculation may take some time in the order of minutes     + *)
(* +                   in the case of a large number of atoms (few hundred) and       + *)
(* +                   and a large number of data base ortientations (several         + *)
(* +                   hundred). It is recommended to use a coarse data base          + *)
(* +                   with only a few hundred database items for test runs.          + *)
(* +                                                                                  + *)
(* +           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.                    *)
(* dbDir is a subfolder that holds the database file.                          *)
(* 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"];
dbDir = "db";
dataDir = "data";
outDir = "out"
(* dbfileName is name of the database file                  *)
dbfileName = "rotation-database_Fcc_30_2.db";
(* datafileName is name of the data file with the cluster xyz coordinates *)
datafileName = "Au-crystal.cel";
(* +-------------------------------------------------------------------------------+ *)
(* +-------------------------- END USER INPUT -------------------------------------+ *)
(* +-------------------------------------------------------------------------------+ *)
```

Out[802]= out

```
In[805]:= (*  define input files                                                *)
DBfile = StringJoin[baseDir, "/", dbDir, "/", dbfileName]
celfile = StringJoin[baseDir, "/", dataDir, "/", datafileName]
```

Out[805]= /Users/lothar/Desktop/diOPA/db/rotation-database_Fcc_30_2.db

Out[806]= /Users/lothar/Desktop/diOPA/data/Au-crystal.cel

```
In[807]:= (*  define output files                                               *)
graphfileprefix = StringJoin[baseDir, "/", outDir, "/", FileBaseName[datafileName], "_OrderParameter"]
logfilename = StringJoin[baseDir, "/", outDir, "/", FileBaseName[datafileName], "_OrderParameter.log"]
```

Out[807]= /Users/lothar/Desktop/diOPA/out/Au-crystal_OrderParameter

Out[808]= /Users/lothar/Desktop/diOPA/out/Au-crystal_OrderParameter.log

```
In[809]:= (* 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[817]:=
```
ModelDB =.;
ModelDB = Get[DBfile];
coordination = Length[ModelDB[[1, 3]]];
NDBItems = Length[ModelDB[[All, 1]]];
Print["- INPUT  Database file: ", InputForm[DBfile]];
Print["- Coordination: ", coordination];
Print["- Number of Database items: ", NDBItems];
```

```
- INPUT  Database file: "/Users/lothar/Desktop/diOPA/db/rotation-database_Fcc_30_2.db"

- Coordination: 12

- Number of Database items: 225
```

In[824]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                  -----------+ *)
(*
          *)
(* END PARAMETER
 DEFINITION                                                                                                                                          *)
(*
          *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                  -----------+ *)
```

In[825]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                  -----------+ *)
(*
          *)
(* START FUNCTION
 DEFINITION                                                                                                                                          *)
(*
          *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                  -----------+ *)
```

In[826]:=
```
(* ReadCelFile: Function for Reading a celfile *)
(* courtesy of J. Barthel, RWTH-Aachen University, Germany *)
(* Email: ju.barthel-at-fz-juelich.de *)
(* 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[827]:=
```
(* QuickDistance: Function for calculating the minimum distance between a coordination polyhedron and all database rotaion variants *)
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[828]:=
```
(* GetNeighbours: Function for sorting the atom list according to their distance with respect to a central atom *)
(* and isolates a given number of nearest neighbours *)
(* INPUT:  index of centre atom, number of nearest neighbours to return *)
(* OUTPUT: nearest neighbour list *)
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[829]:=
```
(* GetDistMeas: Calculate a distance measure to the closest model item in the DB *)
(* This fuction does the same as QuickDistance, it is much slower but in rare cases more accurate *)
(* Returns a global variable: Dist, type: list *)
(* Dist = {atomnr, DBindex,{angle,angle,distance}} *)
(* Example: *)
(* INPUT:   GetDistMeas[41]; Dist *)
(* OUTPUT: {41,256,{π/5,π/10,0.468556698631839}} *)
GetDistMeas[atomindex_] := Block[{DistMap = {}},
  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[830]:=
```
(* 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[831]:=
```
(* CentreBox: Centres xyz coordinates around a common centre *)
(* 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[832]:=
```
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                -----------+ *)
(*
          *)
(* END FUNCTION
 DEFINITION                                                                                                                                                  *)
(*
          *)
(* +-------------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                -----------+ *)
```

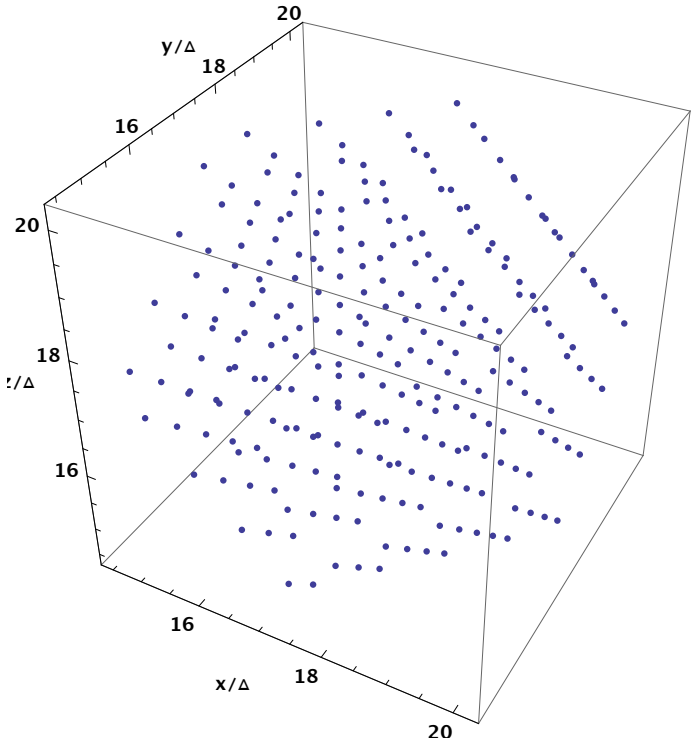
In[833]:=
```
(* +-----------------------------------------------------------------------------------------------------------------
                                                                         -----------+ *)
(*
         *)
(* START
 PROCESSING                                                                                                                  *)
(*
         *)
(* +-----------------------------------------------------------------------------------------------------------------
                                                                         -----------+ *)
```

In[834]:=
```
(* Open log file and start logging *)
logfile = OpenWrite[logfilename];
WriteString[logfile, "Filename:", celfile, "\n"];
WriteString[logfile, "- INPUT  Database file: ", InputForm[DBfile], "\n"];
WriteString[logfile, "- Coordination: ", ToString[coordination], "\n"];
WriteString[logfile, "- Number of Database items: ", ToString[NDBItems], "\n"];
```

In[839]:=
```
(* Open data file and plot data *)
ReadCelFile[celfile];
Clear[coordatms, f, q];
(* Extract relative atomcoordinates only *)
coordatms = celatms[[All, {2, 3, 4}]];
(* 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]]
```

```
- 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[844]=



In[845]:=

```
(* Calculate distance order parameter for each atom in the list *)
(* This step may take some time depending on the sampling density in the database, i.e. the number of database items,  *)
(* and the number of atoms in the cluster. *)
(* For each atom DBDist will get a new entry with the  *)
(* matching database entry, the rotation of the database corrdination polyhedron and the distance orer parameter *)
Clear[DBDist]
DBDist = {};
For[l = 1, l ≤ Length[q], l++, GetDistMeas[l]; AppendTo[
   DBDist, {Dist}]];
```

In[848]:=
```
goutfile = StringJoin[graphfileprefix, "_Values.txt"];
Export[goutfile, DBDist, "Table"] (* import later with ddata=Import["goutfile","Table"] *)
```

Out[849]=
```
/Users/lothar/Desktop/diOPA/out/Au-crystal_OrderParameter_Values.txt
```

In[850]:=
```
(* 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[855]:=
```
(* +------------------------------------------------------------------------------------------------------------------------------------------------------ + *)
(* DISTANCE ORDER PARAMTER
 rendering                                                                                                                                                *)
(* +------------------------------------------------------------------------------------------------------------------------------------------------------ + *)
```

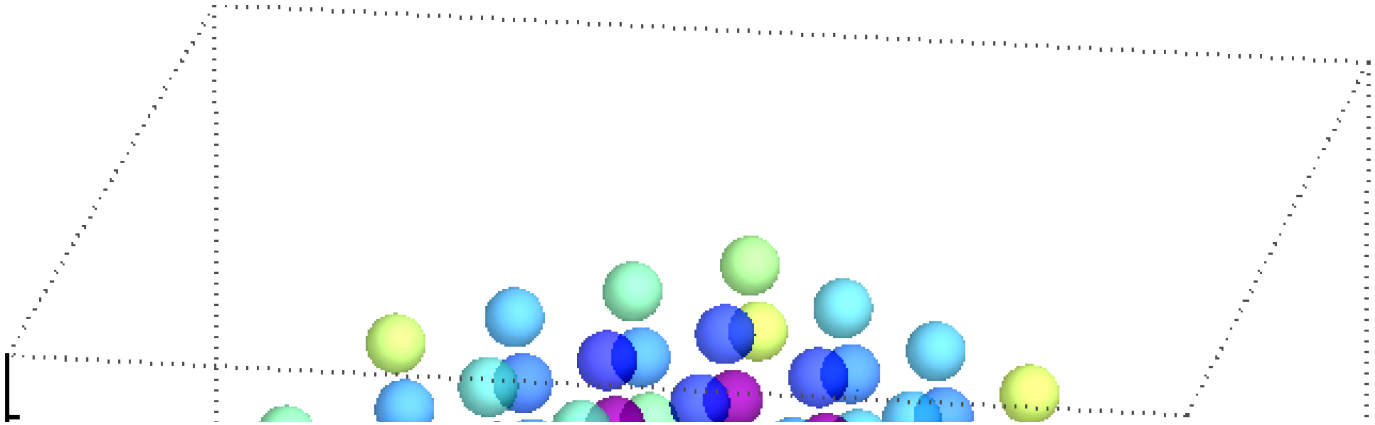
In[856]:=
```
(* INPUT: q=atomlist,
DBDist=list containing the rotation of the coordination vector in the data base and the distance order paramater value  *)
(* OUTPUT: data = vector tupel x,y,z; distance order paramater value *)
(*                              x,y,z are centred *)
(* q, DBDist remain antouched *)
data = {};
plotcol = 3; (* correlation fig *)
For[l = 1, l ≤ Length[DBDist], l++, AppendTo[
   data, {q[[l, 1]], q[[l, 2]], q[[l, 3]], DBDist[[l]][[1, 3]][[plotcol]]}]];
CentreBox[data]
data = centrelist;
BoundingBox[data, 1];
CScaleMin = 0;
CScaleMax = coordination / 2;
CMin = Min[data[[All, 4]]];
CMax = Max[data[[All, 4]]];
CMean = Mean[data[[All, 4]]];
CMedian = Median[data[[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}
```

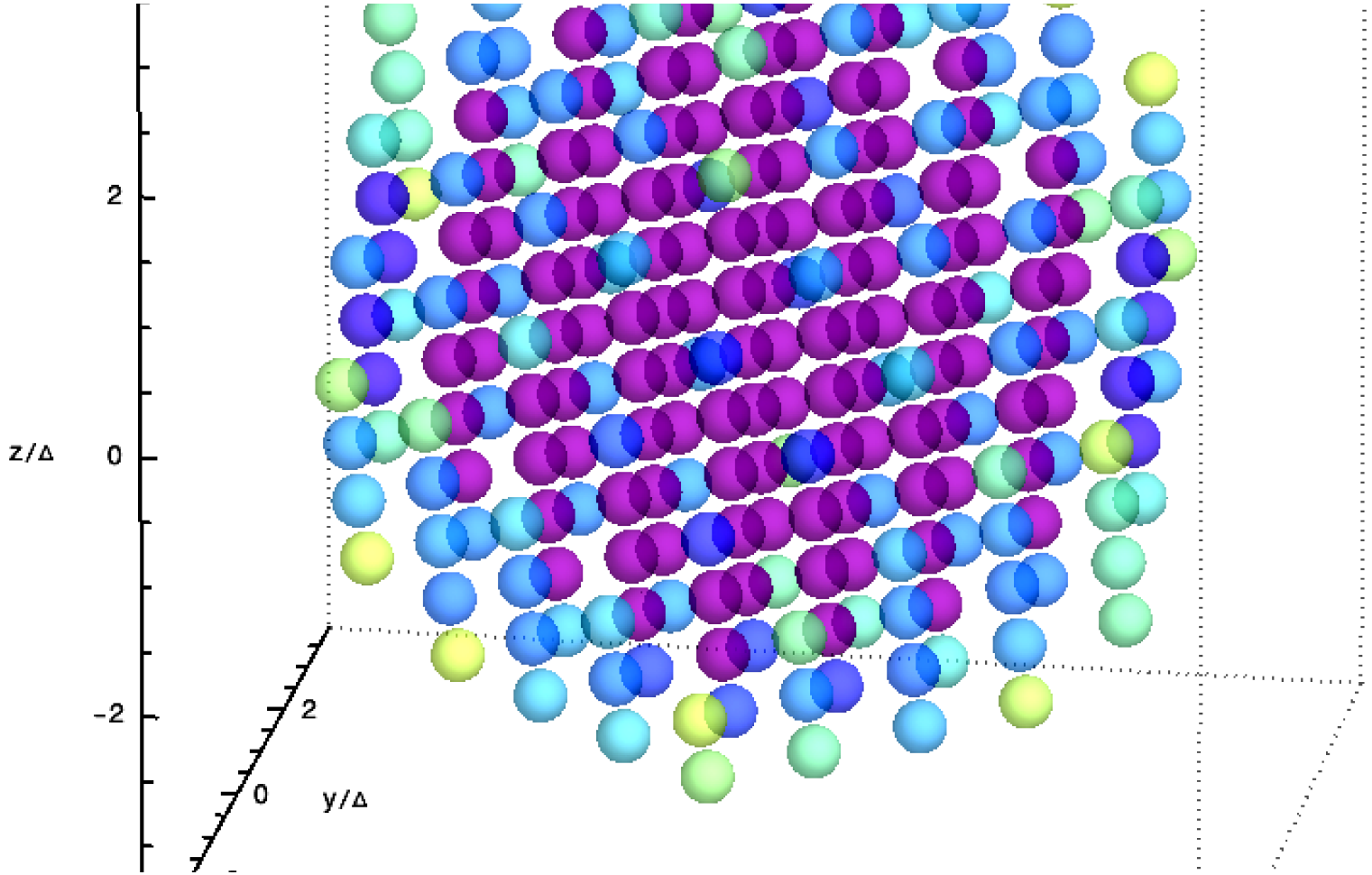
In[868]:=
```
WriteString[logfile, "Bounding box:", ToString[bbox], "\n"];
```

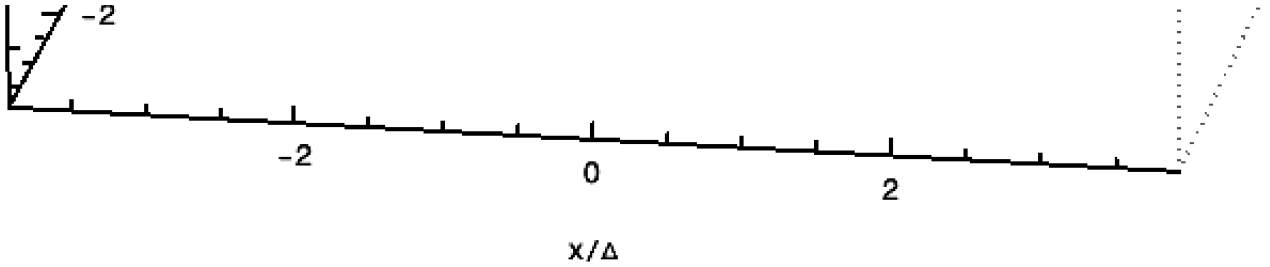
In[869]:=
```
coldata = data;
coldata[[All, 4]] = Map[colnorm, data[[All, 4]]];
lpp1 = ListPointPlot3D[List /@ Most /@ coldata, PlotRange → bbox, BoxStyle → Directive[Thick, Dotted],
   AxesOrigin → axespos, AxesLabel → {"x/Δ", "y/Δ", "z/Δ"}, AxesStyle → axstyle, Lighting → "Neutral",
   BoxRatios → {1, 1, 1}, LabelStyle → Directive[Bold, FontFamily → "Helvetica", FontSize → labSz],
   PlotStyle → ({AbsolutePointSize[1], ColorData[colortbl][(# + 0) / 1]} & /@ Last /@ coldata),
   ViewCenter → {0.5, 0.5, 0.5}, ViewVector → viewvc, ViewVertical → {0, 0, 1}, ImageSize → imsz * ImMagfact];
pp2 = lpp1 /. Point[x__] :→ (Sequence@{EdgeForm[], Lighting → "Neutral", Opacity[plotopac], Sphere[#, 0.2]} &@
      ({x} /. {{a_, b_, c_}} :→ {{a, b, c}}));
```

In[873]:=
```
pp2adj = ImageAdjust[pp2, ContrastEnhancement]
```



Out[873]=





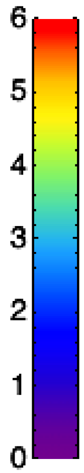
In[874]:=
```
(* Print some statistics about the order parameter *)
Print["- Average distance: ", CMean]
Print["- Median distance: ", CMedian]
Print["- Minimum distance: ", CMin]
Print["- Maximum distance: ", CMax]
Print["- Contrast cut off (Min,Max): (", CScaleMin, ",", CScaleMax, ")"]
WriteString[logfile, "- Average distance: ", ToString[CMean], "\n"];
WriteString[logfile, "- Median distance: ", ToString[CMedian], "\n"];
WriteString[logfile, "- Minimum distance: ", ToString[CMin], "\n"];
WriteString[logfile, "- Maximum distance: ", ToString[CMax], "\n"];
WriteString[logfile, "- Contrast cut off (Min,Max): (", ToString[CScaleMin], ",", ToString[CScaleMax], ")", "\n"];
```

```
- Average distance: 1.61202

- Median distance: 1.80632

- Minimum distance: 0.13955

- Maximum distance: 4.25757

- Contrast cut off (Min,Max): (0,6)
```

In[884]:=
```
colourBar2 = DensityPlot[v^gamma, {u, 0, 1}, {v, CScaleMin, CScaleMax}, ColorFunction → ColorData[colortbl],
    LabelStyle → Directive[Bold, FontFamily → "Helvetica", FontSize → colscalelabSz], AspectRatio → 10, Frame → {False, True},
    PlotRangePadding → 0, ImagePadding → 25, ImageSize → imsz / 2, FrameTicks → {False, Automatic, False, Automatic}];
```

In[885]:=
```
colourBar2adj = ImageAdjust[colourBar2, ContrastEnhancement]
```

Out[885]=



In[886]:=
```
(* Combine the order parameter map and the colour scale bar into one graphics file for file export *)
exportgraph = Graphics[{Inset[colourBar2adj, {2, 0.}, Center, 1.5], Inset[pp2adj, {0, 0.5}, Center, 3]}, PlotRange → 3, ImageSize → 1024];
```

In[887]:=
```
goutfile = StringJoin[graphfileprefix, ".png"];
Export[goutfile, exportgraph, "AllowRasterization" → True, ImageSize → imsz * ImMagfact, ImageResolution → 600]
```

Out[888]= /Users/lothar/Desktop/diOPA/out/Au-crystal_OrderParameter.png

In[889]:=
```
Close[logfile];
```

In[890]:=
```
(* +-----------------------------------------------------------------------------------------------------------------------------------------
-----------+ *)
(*
           *)
(* END
 PROCESSING                                                                                                                             *)
(*
           *)
(* +-----------------------------------------------------------------------------------------------------------------------------------------
-----------+ *)
```