In[64]:=
```
(* +---------------------------------- create_db -------------------------------------+ *)
(* +                                                                                   + *)
(* +           create_db is a program to create a database that contains rotation    + *)
(* +           variants of a coordination polyhedron.                                + *)
(* +           The database is used for the local order parameter determination      + *)
(* +           on a dataset of "atoms", which can be a crystalline condensate or a    + *)
(* +           an unordered cluster.                                                 + *)
(* +                                                                                 + *)
(* +           INPUT:  Coordination polyhedron, rotation sampling and filename       + *)
(* +                   for output data. Further explanantions below                  + *)
(* +           OUTPUT: Data base file containing a list of entries                   + *)
(* +                   Each entry has an index, the rotation angles and the          + *)
(* +                   rotated coordination vector.                                   + *)
(* +                   A log file is created for each data base file as well.         + *)
(* +           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 ------------------------------------+ *)
(* +-------------------------------------------------------------------------------- + *)
(* outDir is the base folder for program input and output.                           *)
(* subDir is a subfolder to hold the database file.                                  *)
(* Please make sure that the folders exist.                                          *)
outDir = StringJoin[$HomeDirectory, "/Desktop/OrderParameterAnalysis"];
subDir = "db";
(* fileName is a prefix name for the database file and the log file                  *)
fileName = "rotation-database";
(* Samp is the number of discrete angular increments over a roation of 360 degrees. *)
(* The rotation is around two orthogonal axis. A typical value for samp is 100,     *)
(* resulting in 2500 rotation variants                                               *)
Samp = 50;
(* Symmetry can be used when the coordination has a high symmetry. For cubic coordination *)
(* a rotation of 90 degrees around each orientation is sufficient *)
(* For lower symmetry a factor of 1 is safe. *)
Symmetry = 1;
(* Coord is a list of labels for predefined coordination vectors                    *)
(* A coordination vector describes a local order around a central "atom". It is an *)
(* n-tupel of three-dimensional vectors that define the nearest neighbour cordination. *)
(* The n-tupel may contain more than the first coordination shell. *)
(* The distance length to the nearest neighbour is normalised to 1. *)
(* Three coordination polehedra are predefined here: Oh, Bcc and Fcc. *)
(* You can add more coordination vectors according to your needs by  *)
(* adding a label to Coord and defining the coordination vector below. *)
(* *)
(* *)
(* Simple cubic order Oh. The Octahedron symmetry with six nearest neighbours. *)
Coord = {"Oh", "Bcc", "Fcc"};
Oh = {{1, 0, 0}, {-1, 0, 0}, {0, 1, 0}, {0, -1, 0}, {0, 0, 1}, {0, 0, -1}};
```

```
(* A base centred cubic (bcc) crystal would have eight nearest neighbours. *)
Bcc = {{1/√3, 1/√3, 1/√3}, {-1/√3, -1/√3, -1/√3}, {1/√3, 1/√3, -1/√3},
   {-1/√3, -1/√3, 1/√3}, {1/√3, -1/√3, 1/√3}, {-1/√3, 1/√3, -1/√3}, {-1/√3, 1/√3, 1/√3}, {1/√3, -1/√3, -1/√3}};
(* A face centred cubic (fcc) crystal would have twelve nearest neighbours. *)
Fcc = {{√2/2, √2/2, 0}, {-√2/2, -√2/2, 0},
   {√2/2, -√2/2, 0}, {-√2/2, √2/2, 0},
   {√2/2, 0, √2/2}, {-√2/2, 0, -√2/2},
   {√2/2, 0, -√2/2}, {-√2/2, 0, √2/2},
   {0, √2/2, √2/2}, {0, -√2/2, -√2/2},
   {0, √2/2, -√2/2}, {0, -√2/2, √2/2}
  };
(* +-------------------------------------------------------------------------------  + *)
(* +---------------------------- END USER INPUT -------------------------------------+ *)
(* +------------------------------------------------------------------------------- + *)
```

In[73]:=

```
(* +---------------------------- Processing -----------------------------------+ *)
```

In[74]:=

```
(* +                                                                          + *)
(* General rotation around an axis in three dimensions                          *)
(*                                                                              *)
(* 3 dimensional rotation around a normalized tilt axis (wx,wy,wz)              *)
(* The script uses the Rodriguez rotation formula                               *)
(* A rotation around the angle θ is given by                                      *)
 (* u=(IdentityMatrix[3]+ωtilde Sin[θ]+ωtilde.ωtilde(1-Cos[θ]))*v                *)
(* note that (ωx,ωy,ωz) is a unit vector pointing in the direction of the tilt axis *)
(* *)
Clear[ωX]
Clear[ωY]
Clear[ωZ]
ωX =.
ωY =.
ωZ =.
MatrixForm[ωtilde = {{0, -ωZ, ωY}, {ωZ, 0, -ωX}, {-ωY, ωX, 0}}] // TraditionalForm
m1 = FullSimplify[ComplexExpand[MatrixExp[θ ωtilde] /. ωZ^2 → 1 - ωX^2 - ωY^2]]
(* RotVector does the matrix multiplication the rotation matrix with a vector v *)
RotVector[v_] = Dot[m1, v] /. {ωY → Sqrt[1 - ωX^2], ωZ → 0.}
```

Out[80]//TraditionalForm=

$$\begin{pmatrix} 0 & -\omega Z & \omega Y \\ \omega Z & 0 & -\omega X \\ -\omega Y & \omega X & 0 \end{pmatrix}$$

Out[81]= $\left\{\left\{\omega X^2 + \text{Cos}[\theta] - \omega X^2 \text{Cos}[\theta],\ \omega X\,\omega Y - \omega X\,\omega Y \text{Cos}[\theta] - \omega Z \text{Sin}[\theta],\ \omega X\,\omega Z - \omega X\,\omega Z \text{Cos}[\theta] + \omega Y \text{Sin}[\theta]\right\},\right.$
$\left.\left\{\omega X\,\omega Y - \omega X\,\omega Y \text{Cos}[\theta] + \omega Z \text{Sin}[\theta],\ \omega Y^2 + \text{Cos}[\theta] - \omega Y^2 \text{Cos}[\theta],\ \omega Y\,\omega Z - \omega Y\,\omega Z \text{Cos}[\theta] - \omega X \text{Sin}[\theta]\right\},\ \left\{\omega X\,\omega Z - \omega X\,\omega Z \text{Cos}[\theta] - \omega Y \text{Sin}[\theta],\ \omega Y\,\omega Z - \omega Y\,\omega Z \text{Cos}[\theta] + \omega X \text{Sin}[\theta],\ 1 - \omega X^2 - \omega Y^2 + \left(\omega X^2 + \omega Y^2\right) \text{Cos}[\theta]\right\}\right\}$

Out[82]= $\left\{\left\{\omega X^2 + \text{Cos}[\theta] - \omega X^2 \text{Cos}[\theta],\ \omega X \sqrt{1 - \omega X^2} - \omega X \sqrt{1 - \omega X^2} \text{Cos}[\theta] + 0. \text{Sin}[\theta],\ 0. \omega X + 0. \omega X \text{Cos}[\theta] + \sqrt{1 - \omega X^2} \text{Sin}[\theta]\right\},\right.$
$\left.\left\{\omega X \sqrt{1 - \omega X^2} - \omega X \sqrt{1 - \omega X^2} \text{Cos}[\theta] + 0. \text{Sin}[\theta],\ 1 - \omega X^2 + \text{Cos}[\theta] - \left(1 - \omega X^2\right) \text{Cos}[\theta],\ 0. \sqrt{1 - \omega X^2} + 0. \sqrt{1 - \omega X^2} \text{Cos}[\theta] - \omega X \text{Sin}[\theta]\right\},\ \left\{0. \omega X + 0. \omega X \text{Cos}[\theta] - \sqrt{1 - \omega X^2} \text{Sin}[\theta],\ 0. \sqrt{1 - \omega X^2} + 0. \sqrt{1 - \omega X^2} \text{Cos}[\theta] + \omega X \text{Sin}[\theta],\ \text{Cos}[\theta]\right\}\right\}.v$

In[83]:=
```
(* DB construction *)
(* (1) create a list of rotation vectors to cover the unit sphere θ,ω *)
(*     each index in the list corresponds to a single orientation *)
(* (2) for each element (=rotation vector) do calculate the object transformation and store the transformation with the index of rotation *)
(* Step (1): Prepare rotation vectors *)
da = 2 * π / Samp;
maxa = 2 * π / Symmetry;
imax = maxa / da;
DBTable = Table[{i * da, j * da}, {i, 0, imax - 1}, {j, 0, imax - 1}];
DBIndexTable = Flatten[DBTable, 1];
(* Step (2): Construct Object Rotation data bases, this is done in a loop for all the coordination polyhedra in Coord *)
(* The database will hold the index, the rotation angle and the transformed vectors *)
(* ObjRotDB[[index,1]] = index *)
(* ObjRotDB[[index,2]] = angular coordinates (tilt axis x component, azimuth rotation) *)
(* ObjRotDB[[index,3]] = transformed coordinates *)
For[ind = 1, ind <= Length[Coord], ind++,
  Print["Preparing database for the coordination ", ToString[Coord[[ind]]], " ..."];
  Obj = ToExpression[Coord[[ind]]];
  DBfile = StringJoin[outDir, "/", subDir, "/", fileName, "_", ToString[Coord[[ind]]], "_", ToString[Samp], "_", ToString[Symmetry], ".db"];
  ObjRotDB = {};
  For[i = 1, i ≤ Length[DBIndexTable], i++, AppendTo[ObjRotDB, {i, DBIndexTable[[i]], Map[RotVector, Obj] /. {ωX → Sin[DBIndexTable[[i, 1]]], θ → DBIndexTable[[i, 2]]}}]];
  Put[ObjRotDB, DBfile]; Print["finished."]];

(* Write some documentation output *)

For[ind = 1, ind <= Length[Coord], ind++,
  Print["Writing log file for the coordination ", ToString[Coord[[ind]]], " ..."];
  logfile = StringJoin[outDir, "/", subDir, "/", fileName, "_", ToString[Coord[[ind]]], "_", ToString[Samp], "_", ToString[Symmetry], ".log"];
  OpenWrite[logfile];
  WriteString[logfile, "Coordination rotation variants created by create_db.nb", "\n"];
  WriteString[logfile, "-----------------------------------------------------", "\n"];
  WriteString[logfile, "coordination label         : ", Coord[[ind]], "\n"];
  WriteString[logfile, "coordination vectors       : ", ExportString[ToExpression[Coord[[ind]]], "Text"], "\n"];
  WriteString[logfile, "number of neighbours       : ", Length[ToExpression[Coord[[ind]]]], "\n"];
  WriteString[logfile, "sampling                   : ", Samp, "\n"];
  WriteString[logfile, "symmetry                   : ", Symmetry, "\n"];
  WriteString[logfile, "angular tilt increment (rad): ", ExportString[da, "Text"], "\n"];
  WriteString[logfile, "angular range (rad)        : ", maxa, "\n"];
  WriteString[logfile, "number or database items   : ", Length[DBIndexTable], "\n"];
  Close[logfile];
  Print["finished."]
 ];
```

```
Preparing database for the coordination Oh ...

finished.

Preparing database for the coordination Bcc ...

finished.

Preparing database for the coordination Fcc ...

finished.

Writing log file for the coordination Oh ...

finished.

Writing log file for the coordination Bcc ...

finished.

Writing log file for the coordination Fcc ...

finished.
```