Program SYMPRJS and SYMPWS

Contents

Programs SYMPRJS and SYMPWS

Authors Per-Olof Jansson, Esko Blokker and Stig Flodmark

2006-08-18

The **SYMPRJS** and **SYMPWS** Matlab programs calculates the symmetry projection matrices for the **double** space groups and thereby taking different spin directions into account. **SYMPRJS** and **SYMPWS** are functionally, largely based on the FORTRAN programs **SYMPRJ** and **SYMPW** (QCPE no. 259), which calculates the symmetry projection matrices for the **single** space groups.

1. Prodats

Program **Prodats** creates 7 output files which are stored on disc; to be read by **SYMPRJS/SYMPWS**.

prodats.dat is the input file for **prodats.m** execution.

After defining the variables and the files, the program reads the Euler angles of the first 24 elements of the double point group ${}^2O_h$ into the matrix **Oh(1:3,1:24)**. The inversion parameter, **Oh(4,1:24)**, of these first 24 elements is set equal to 0, denoting that no inversion is involved. Euler angles for elements 25:48 are defined by **Oh(1:4,25:48) = Oh(1:4,1:24)** The Euler angles of the elements 49:96 of ${}^2O_h$ are equal to the first 1:48, but the inversion parameter is set to equal to 1, denoting that the operations include inversion. The input angles are given in units of π, so the program multiplies the input angles by π and then prints them, together with the inversion parameters.

Similarly the Euler angles and inversion parameters of the double point group ${}^2D_{6h}$ are read into **D6h(1:3,1:12)**, extended to the set **D6h(1:4,1:48)** and printed.

Our conventions for Euler angles and enumeration of the point group elements are described in chapter 2.

The next input is the upper left quarter of the multiplication table of ${}^2O_h$, which is read into **MOh(1:48,1:48)**. From this quarter the complete multiplication table **MOh(1:96,1:96)** is calculated and then printed.

The same procedure is followed for the multiplication table of ${}^2D_{6h}$.

The multiplication tables are the same as in reference [2], since we have enumerated the group elements in the same way.

The matrix **npgo(1:2,1:36)** is read, where **npgo(1,1:36)** contains the orders of the 36 point groups (36 and not 32, since some point groups are defined in two different ways). The enumeration of the point groups is the same as in Table 3 of chapter 2. **npgo(2,K)** contains the index that denotes where the elements of group K are stored in the matrix **nge(1:736)**. Then the indices of each point group, as listed in chapter 2, Table 3, Subsequently the program will calculate the rotation matrices ${}^l\mathbf{D}(\phi_i,\theta_i,\psi_i)$ for $l = 0,1,2,3$ for all different rotations. For operations including inversion the matrices are multiplied by $(-1)^l$.

All matrix elements ${}^lD(L1,L2)$ of group element I, for $l = 0,1,2,3$ are stored into matrix **ldrmm(I,1:84)**.

The matrices ${}^1\mathbf{D}(l=1)$ are also stored into **rcgr3(1:3,1:3,I)** for later use.

Our convention for spherical harmonics is the following:

$$Y_{lm}(\theta,\phi) = (-1)^m \left(\frac{2l+1}{4\pi}\cdot\frac{(l-m)!}{(l+m)!}\right)^{\frac{1}{2}} \frac{1}{2^l l!} e^{im\phi} \sin^m\theta\left(\left(\frac{d}{d\cos\theta}\right)^{l+m}(\cos^2\theta-1)^l\right) \qquad (1)$$

The rotations (ϕ,θ,ψ) are represented in this basis by

$${}^lD(\phi,\theta,\psi)_{m',m} = \left(\sum_{\kappa=\kappa_1}^{\kappa_2}(-1)^\kappa \frac{((l-m')!(l+m')!(l-m)!(l+m)!)^{\frac{1}{2}}}{(l-m-\kappa)!(l+m'-\kappa)!\kappa!(\kappa+m-m')!}\left(\cos\frac{\theta}{2}\right)^{2l-2\kappa-m+m'}\left(\sin\frac{\theta}{2}\right)^{2\kappa+m-m'}\right)\cdot$$

$$e^{-im'\phi}e^{-im\psi} \qquad (2)$$

and $\kappa_1 = \max(0,m'-m), \kappa_2 = \min(l+m',l-m)$

So if

$$\mathbf{Y}_l = (Y_{l,-l}, Y_{l,-l+1},\ldots,Y_{l,l}) \qquad (3)$$

Then for a point group element $P(\phi,\theta,\psi,\lambda)$, including rotation $R(\phi,\theta,\psi)$ and inversion $I^\lambda(\lambda =$ 0 or 1):

$$P(\phi,\theta,\psi,\lambda)\mathbf{Y}_l = \mathbf{Y}_l(-1)^{\lambda l}\,{}^l\mathbf{D}(\phi,\theta,\psi) \qquad (4)$$

The matrices ${}^l\mathbf{D}$ are calculated by function **dmatr**. Since the highest value of l is only 3, no recursive techniques have been used and formula (2) has been programmed, making use of the factorial function **fac**. The angles and the ${}^{0-3}\mathbf{D}$ matrices are printed.

The rotation of a function $\zeta(\bar{r}) = \zeta((x,y,z)) = \zeta\left(\begin{pmatrix}x\\y\\z\end{pmatrix}\right)$ is defined as follows:

$$P_R\zeta(\bar{r}) = \zeta(R^{-1}\,\bar{r}) = \zeta\left(\mathbf{R}^{-1}\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right) = \zeta((x,y,z)\mathbf{R}) \tag{5}$$

By this we mean that the function $P_R\zeta$ has at point $\bar{r}$ the same value as function ζ at point $R^{-1}\,\bar{r}$.

Consider the functions $\zeta_1(\bar{r}),\zeta_2(\bar{r}),\zeta_3(\bar{r})$, defined as follows:

$$\zeta_1(\bar{r}) = \zeta_1((x,y,z)) = \text{the first argument of } \zeta_1 = x \tag{6a}$$

$$\zeta_2(\bar{r}) = \zeta_2((x,y,z)) = \text{the second argument of } \zeta_2 = y \tag{6b}$$

$$\zeta_3(\bar{r}) = \zeta_3((x,y,z)) = \text{the third argument of } \zeta_3 = z \tag{6c}$$

Then

$$P_R\zeta_1(\bar{r}) = \zeta_1((x,y,z)\mathbf{R}) = xR_{11} + yR_{21} + zR_{31} \tag{7a}$$

$$P_R\zeta_2(\bar{r}) = \zeta_2((x,y,z)\mathbf{R}) = xR_{12} + yR_{22} + zR_{32} \tag{7b}$$

$$P_R\zeta_3(\bar{r}) = \zeta_3((x,y,z)\mathbf{R}) = xR_{13} + yR_{23} + zR_{33} \tag{7c}$$

$$P_R(\zeta_1,\zeta_2,\zeta_3) = P_R(x,y,z) = (x,y,z)\mathbf{R} = (\zeta_1,\zeta_2,\zeta_3)\mathbf{R} \tag{8}$$

When we rotate a vector $\bar{a}$, then its components change according to:

$$R\,\bar{a} = \mathbf{R}\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \tag{9}$$

This equation is used in the second and third terms of (5). Now the functions ζ_1,ζ_2,ζ_3 are simple linear combinations of $Y_{1,-1},Y_{1,0},Y_{1,1}$ and we can use this fact to calculate the rotation matrices **R** from the matrices ${}^1\mathbf{D}$.

$$Y_{1,-1}(\bar{r}) = \frac{1}{\sqrt{2}}(x - iy) = \sqrt{\frac{3}{8\pi}}\sin(\theta)e^{-i\phi} \tag{10a}$$

$$Y_{1,0}(\bar{r}) = z = \sqrt{\frac{3}{4\pi}}\cos(\theta) \tag{10b}$$

$$Y_{1,1}(\bar{r}) = -\frac{1}{\sqrt{2}}(x+iy) = -\sqrt{\frac{3}{8\pi}}\sin(\theta)e^{i\phi} \tag{10c}$$

In (10) we have restricted $\bar{r}$ to the unit sphere. Then:

$$(\zeta_1, \zeta_2, \zeta_3) = (x, y, z) = (Y_{1,-1}, Y_{1,0}, Y_{1,1}) \begin{pmatrix} 1/\sqrt{2} & i/\sqrt{2} & 0 \\ 0 & 0 & 1 \\ -1/\sqrt{2} & i/\sqrt{2} & 0 \end{pmatrix} = \mathbf{Y}_1\mathbf{Q} \tag{11}$$

$$P_R(x, y, z) = P_R\mathbf{Y}_1\mathbf{Q} = \mathbf{Y}_1{}^1\mathbf{DQ} = (x, y, z)\mathbf{Q}^{-1\,1}\mathbf{DQ} \tag{12}$$

So we conclude that:

$$\mathbf{R}(\phi, \theta, \psi) = \mathbf{Q}^{-1\,1}\mathbf{D}(\phi, \theta, \psi)\mathbf{Q} \tag{13}$$

The program calculates the rotation matrices this way.

The matrices ${}^1\mathbf{D}$, which were stored in **rcgr3(1:3,1:3,l)**, are all transformed by the $\mathbf{Q}$ matrix and the result is stored into **rgr3(1:3,1:3,l)**. These matrices are also printed.

Then the program produces the 100 smallest primes, larger than 3, by calling the function **primen**. The primes are stored in **NP(1:100)**. They will be used in **charac**, a function called by the function **Irrep**. The primes are also printed.

To extend the program to higher values of $l\ (l_{\max} = 3;\text{ at present})$, the dimension of **ldrmm** has to be changed to

$$ldrmm(1:144, 1: \sum_{l=0}^{l_{\max}} (2l+1)^2)$$

and of matrix $\mathbf{D}$ in function **dmatr** to $\mathbf{D}(2l_{\max}+1,\ 2l_{\max}+1)$. Further, statements have to be added, so that the matrices are stored correctly into **ldrmm**.

The spin matrices are calculated and stored in **spinm(1:2,1:2,1:144)**. The matrix elements of the spin matrices are calculated according to the expression:

$${}^{1/2}D_{s's}(\phi, \theta, \psi) = \begin{pmatrix} e^{-\frac{1}{2}i(\psi+\phi)}\cos\left(\frac{\theta}{2}\right) & -e^{\frac{1}{2}i(\psi-\phi)}\sin\left(\frac{\theta}{2}\right) \\ e^{-\frac{1}{2}i(\psi-\phi)}\sin\left(\frac{\theta}{2}\right) & e^{\frac{1}{2}i(\psi+\phi)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

After a proper execution; “Prodats has finished executing” is printed at the end of **prodats** execution.

See Figure 1 for the flowing scheme.

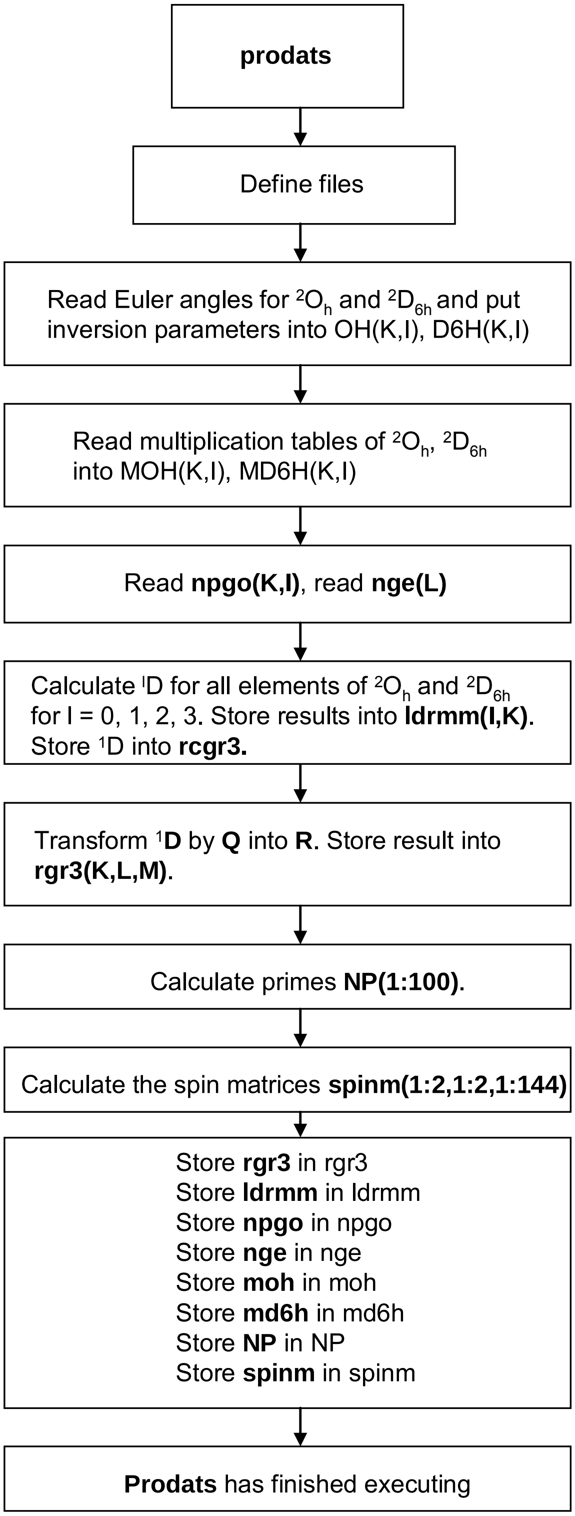
prodats
Define files
Read Euler angles for ^2O_h and $^2D_{6h}$ and put inversion parameters into OH(K,I), D6H(K,I)
Read multiplication tables of ^2O_h, $^2D_{6h}$ into MOH(K,I), MD6H(K,I)
Read **npgo(K,I)**, read **nge(L)**
Calculate ^lD for all elements of ^2O_h and $^2D_{6h}$ for l = 0, 1, 2, 3. Store results into **ldrmm(l,K)**. Store ^1D into **rcgr3.**
Transform ^{1}**D** by **Q** into **R**. Store result into **rgr3(K,L,M)**.
Calculate primes **NP(1:100)**.
Calculate the spin matrices **spinm(1:2,1:2,1:144)**
Store **rgr3** in rgr3
Store **ldrmm** in ldrmm
Store **npgo** in npgo
Store **nge** in nge
Store **moh** in moh
Store **md6h** in md6h
Store **NP** in NP
Store **spinm** in spinm
Prodats has finished executing

Figure 1: Flowing scheme of prodats

2. Conventions for Point Groups and Space Groups

The space group of a crystal with atomic positions $\bar{\mu}$, is the maximal set of operators $F_i = (P_i \,|\, \bar{u}_i + \bar{n})$, which, when acting on the atomic positions, according to

$$(P_i \,|\, \bar{u}_i + \bar{n})\,\bar{\mu} = P_i\,\bar{\mu} + \bar{u}_i + \bar{n} \qquad (i = 1, 2, \ldots) \tag{14}$$

map the crystal onto itself. P_i is a point group operator, $\bar{n}$ is a primitive lattice vector and $\bar{u}_i$ is a nonprimitive lattice vector. The point group operators P_i of the space group form a point group.

The point group operators can be described by the Euler angles ϕ_i, θ_i, ψ_i and the inversion label λ_i , which is equal to 1 if inversion is included and equal to 0 if there is no inversion. So:

$$P_i = (\phi_i, \theta_i, \psi_i, \lambda_i) \tag{15}$$

We define the Euler angles as follows:

Start with a right-handed coordinate system x, y, z. All rotations will also be right-handed (counter-clockwise). First make a rotation ϕ through the z-axis. This produces a rotated coordinate system x', y', z' from the original coordinate system. Then follows a rotation θ through the y'-axis, transforming (x', y', z') into (x", y", z"). Finally follows a rotation ψ through the z"-axis, transforming (x", y", z") into (x"', y"', z"'). See Figure 2.

All point groups are subgroups of either point group ${}^2O_h$ or point group ${}^2D_{6h}$. We have written a list of the Euler angles of these point groups in Table 1 and Table 2. For the other point groups we denote which elements are included in each of them (Table 3). Our enumeration of the point group operators of ${}^2O_h$ and ${}^2D_{6h}$ is the same as in reference [2], but we have another definition of Euler angles.

In Table 1 we list also the transformation of the Cartesian coordinates x, y, z by the point group operators of ${}^2O_h$. For point group ${}^2D_{6h}$ this is done in Table 2.

Figure 3 shows the points to which point 1 are mapped by the application of the point group operators of ${}^2O_h$. Figure 4 shows the points into which point 1 is transformed by the application of the operators of group ${}^2D_{6h}$. To this figure belongs a skew coordinate system (x, y, z). The z-axis is directed along a sixth-order axis and forms right angles with the x- and y-axes. Both of these point towards an edge of the prism and the angle between them is $2\pi/3$.

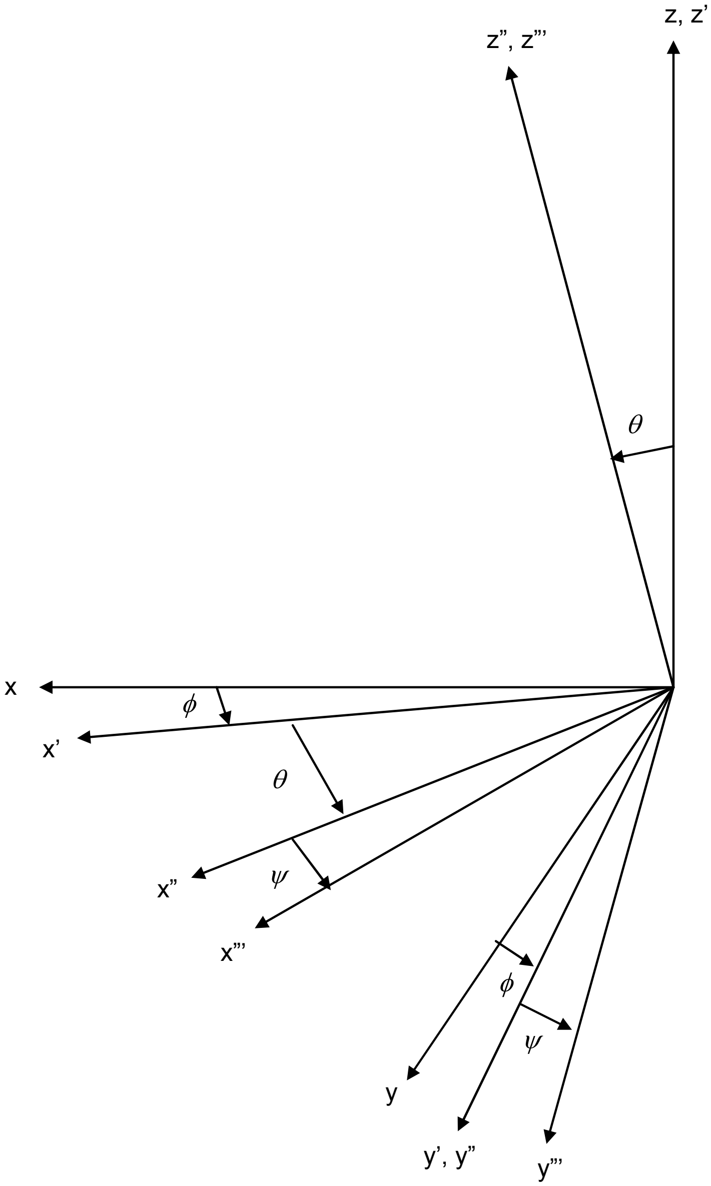

Figure 2: Definition of Euler angles

In Table 3 we list the point groups. In the first column we give the index number of the point group. In column two we give the name of the point group. In column three we give the point group operators (enumerated as in Table 1 and Table 2), which belong to this point group. In column four we give the names of the space groups, which have exactly this point group as their point group of the space group. We also give the index numbers of these space groups, according to the enumeration in reference [3]. In order to obtain complete correspondence with these tables, we had to write some of the point groups in two isomorphic forms. This resulted in a total of 36 point groups. Some of the space groups in reference [3] are given for two different systems of coordinates. For such groups we give an extra subscript 1 or 2, meaning the first or second setting in reference [3] to this form of the point groups. So ${}^2C^6_{3v,2}$ = 161 means that the second set of coordinate axes in reference [3] for space group ${}^2C^6_{3v}$, with space group index 161 corresponds to the given form of the point group. We write h(1,13) instead of h_1, h_{13} and we write h(1-4) instead of h_1, $h_{2,}$ h_3, h_4.

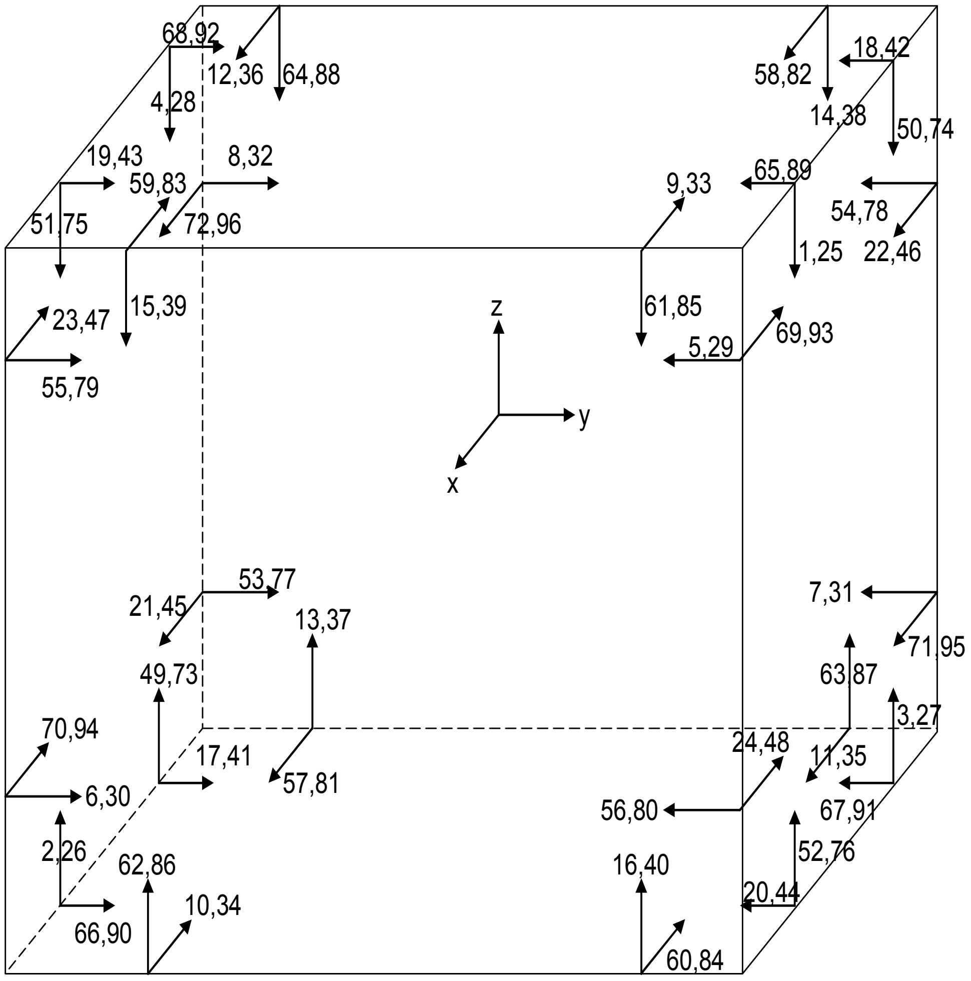

Figure 3: The operations of double point group O_h

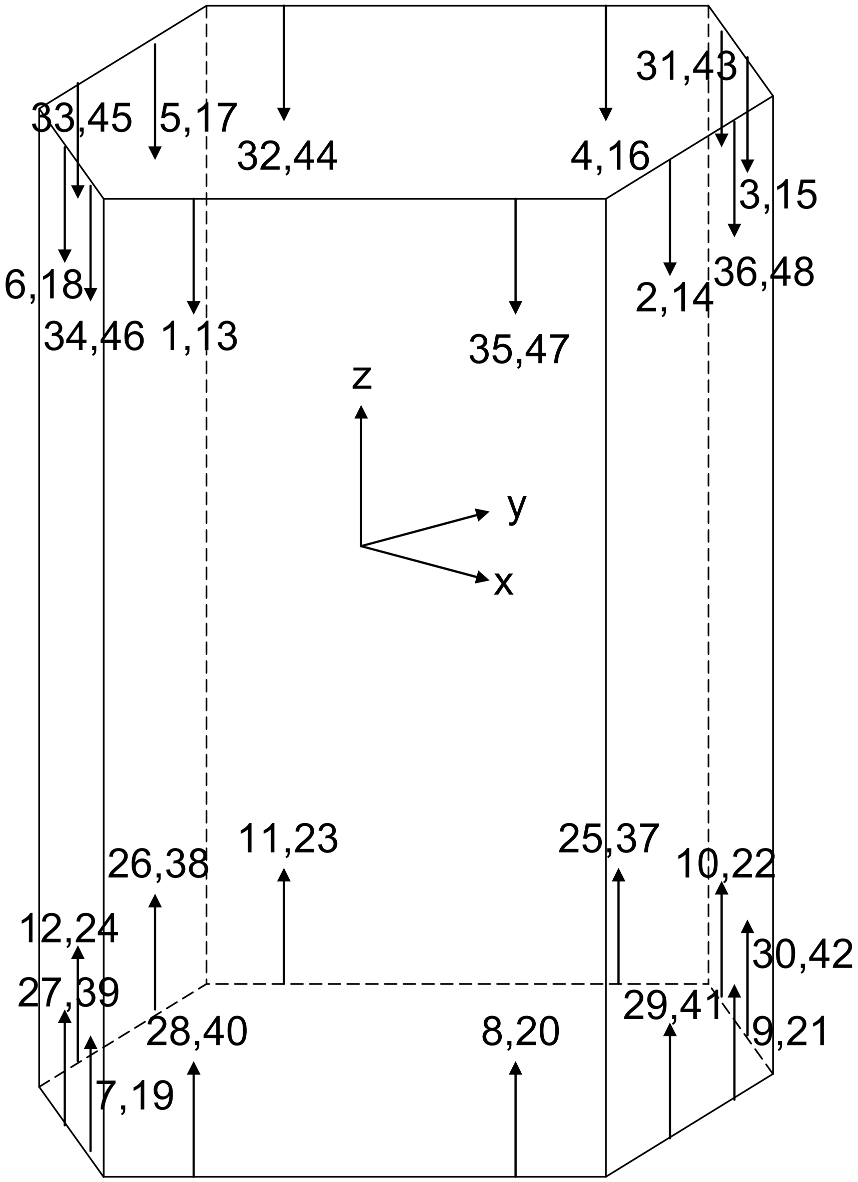

Figure 4: The operations of double point group D_{6h}

Table 1: Double point group operators of $^{2}O_{h}$

Element	Transformed Cartesian coordinates	Euler angles $\phi/\pi, \theta/\pi, \psi/\pi$	Inversion λ	Description of the double point group operation
h1	$x\,y\,z$	0 0 0	0	*identity transformation*
h2	$x\,\bar{y}\,\bar{z}$	0 1 1	0	$\pi -$ *rotation about* $(1,0,0)$
h3	$\bar{x}\,y\,\bar{z}$	0 1 0	0	$\pi -$ *rotation about* $(0,1,0)$
h4	$\bar{x}\,\bar{y}\,z$	1 0 0	0	$\pi -$ *rotation about* $(0,0,1)$
h5	$y\,z\,x$	3/2 3/2 0	0	$4\pi/3 -$ *rotation about* $(1,1,1)$
h6	$y\,\bar{z}\,\bar{x}$	3/2 1/2 0	0	$2\pi/3 -$ *rotation about* $(1,1,\bar{1})$
h7	$\bar{y}\,z\,\bar{x}$	1/2 1/2 0	0	$2\pi/3 -$ *rotation about* $(\bar{1},1,1)$
h8	$\bar{y}\,\bar{z}\,x$	1/2 3/2 0	0	$2\pi/3 -$ *rotation about* $(1,\bar{1},1)$
h9	$z\,x\,y$	0 1/2 1/2	0	$2\pi/3 -$ *rotation about* $(1,1,1)$
h10	$z\,\bar{x}\,\bar{y}$	0 1/2 3/2	0	$4\pi/3 -$ *rotation about* $(1,\bar{1},1)$
h11	$\bar{z}\,x\,\bar{y}$	0 3/2 1/2	0	$4\pi/3 -$ *rotation about* $(1,1,\bar{1})$
h12	$\bar{z}\,\bar{x}\,y$	0 3/2 3/2	0	$4\pi/3 -$ *rotation about* $(\bar{1},1,1)$
h13	$\bar{y}\,\bar{x}\,\bar{z}$	1/2 1 0	0	$\pi -$ *rotation about* $(\bar{1},1,0)$
h14	$\bar{y}\,x\,z$	1/2 0 0	0	$\pi/2 -$ *rotation about* $(0,0,1)$
h15	$y\,\bar{x}\,z$	3/2 0 0	0	$3\pi/2 -$ *rotation about* $(0,0,1)$
h16	$y\,x\,\bar{z}$	3/2 1 0	0	$\pi -$ *rotation about* $(1,1,0)$
h17	$\bar{x}\,\bar{z}\,\bar{y}$	3/2 1/2 3/2	0	$\pi -$ *rotation about* $(0,\bar{1},1)$
h18	$\bar{x}\,z\,y$	1/2 1/2 1/2	0	$\pi -$ *rotation about* $(0,1,1)$
h19	$x\,\bar{z}\,y$	3/2 1/2 1/2	0	$\pi/2 -$ *rotation about* $(1,0,0)$
h20	$x\,z\,\bar{y}$	3/2 3/2 1/2	0	$3\pi/2 -$ *rotation about* $(1,0,0)$
h21	$\bar{z}\,\bar{y}\,\bar{x}$	1 1/2 0	0	$\pi -$ *rotation about* $(\bar{1},0,1)$
h22	$\bar{z}\,y\,x$	0 3/2 0	0	$3\pi/2 -$ *rotation about* $(0,1,0)$
h23	$z\,\bar{y}\,x$	1 3/2 0	0	$\pi -$ *rotation about* $(1,0,1)$
h24	$z\,y\,\bar{x}$	0 1/2 0	0	$\pi/2 -$ *rotation about* $(0,1,0)$

$h_{24+i} = h_i$ for i=1…24				
h_{49}	$\bar{x}\,\bar{y}\,\bar{z}$	0 0 0	1	*inversion*
h_{50}	$\bar{x}\,y\,z$	0 1 1	1	*reflection in plane* (1,0,0)
h_{51}	$x\,\bar{y}\,z$	0 1 0	1	*reflection in plane* (0,1,0)
h_{52}	$x\,y\,\bar{z}$	1 0 0	1	*reflection in plane* (0,0,1)
h_{53}	$\bar{y}\,\bar{z}\,\bar{x}$	3/2 3/2 0	1	$4\pi/3-$ *rotation about* $(1,1,1)+$ *inv*.
h_{54}	$\bar{y}\,z\,x$	3/2 1/2 0	1	$2\pi/3-$ *rotation about* $(1,1,\bar{1})+$ *inv*
h_{55}	$y\,\bar{z}\,x$	1/2 1/2 0	1	$2\pi/3-$ *rotation about* $(\bar{1},1,1)+$ *inv*.
h_{56}	$y\,z\,\bar{x}$	1/2 3/2 0	1	$2\pi/3-$ *rotation about* $(1,\bar{1},1)+$ *inv*
h_{57}	$\bar{z}\,\bar{x}\,\bar{y}$	0 1/2 1/2	1	$2\pi/3-$ *rotation about* $(1,1,1)+$ *inv*.
h_{58}	$\bar{z}\,x\,y$	0 1/2 3/2	1	$4\pi/3-$ *rotation about* $(1,\bar{1},1)+$ *inv*
h_{59}	$z\,\bar{x}\,y$	0 3/2 1/2	1	$4\pi/3-$ *rotation about* $(1,1,\bar{1})+$ *inv*
h_{60}	$z\,x\,\bar{y}$	0 3/2 3/2	1	$4\pi/3-$ *rotation about* $(\bar{1},1,1)+$ *inv*.
h_{61}	$y\,x\,z$	1/2 1 0	1	*reflection in plane* $(\bar{1},1,0)$
h_{62}	$y\,\bar{x}\,\bar{z}$	1/2 0 0	1	$3\pi/2-$ *mirror rotation about* (0,0,1)
h_{63}	$\bar{y}\,x\,\bar{z}$	3/2 0 0	1	$\pi/2-$ *mirror rotation about* (0,0,1
h_{64}	$\bar{y}\,\bar{x}\,z$	3/2 1 0	1	*reflection in plane* (1,1,0)
h_{65}	$x\,z\,y$	3/2 1/2 3/2	1	*reflection in plane* $(0,\bar{1},1)$
h_{66}	$x\,\bar{z}\,\bar{y}$	1/2 1/2 1/2	1	*reflection in plane* (0,1,1)
h_{67}	$\bar{x}\,z\,\bar{y}$	3/2 1/2 1/2	1	$3\pi/2-$ *mirror rotation about* (1,0,
h_{68}	$\bar{x}\,\bar{z}\,y$	3/2 3/2 1/2	1	$\pi/2-$ *mirror rotation about* (1,0,0
h_{69}	$z\,y\,x$	1 1/2 0	1	*reflection in plane* $(\bar{1},0,1)$

h_{70}	$z\ \bar{y}\ \bar{x}$	0 3/2 0	1	$\pi/2-$ *mirror rotation about* $(0,1,0$
h_{71}	$\bar{z}\ y\ \bar{x}$	1 3/2 0	1	*reflection in plane* $(1,0,1)$
h_{72}	$\bar{z}\ \bar{y}\ x$	0 1/2 0	1	$3\pi-$ *mirror rotation about* $(0,1,0)$
$h_{24+i} = h_i$ for i=49…72				

Table 2: Double point group operators of $^2D_{6h}$

Element	Transformed skew coordinates	Euler angles $\phi/\pi, \theta/\pi, \psi/\pi$	Inversion λ	Description of the double point group operation
g_1	$x\ y\ z$	0 0 0	0	*identity transformation*
g_2	$x-y\ x\ z$	1/3 0 0	0	$\pi/3-$ *rotation about* $(0,0,1)$
g_3	$\bar{y}\ x-y\ z$	2/3 0 0	0	$2\pi/3-$ *rotation about* $(0,0,1)$
g_4	$\bar{x}\ \bar{y}\ z$	1 0 0	0	$\pi-$ *rotation about* $(0,0,1)$
g_5	$y\text{-}x\ \bar{x}\ z$	4/3 0 0	0	$4\pi/3-$ *rotation about* $(0,0,1)$
g_6	$y\ y-x\ z$	5/3 0 0	0	$5\pi/3-$ *rotation about* $(0,0,1)$
g_7	$\bar{x}\ y-x\ \bar{z}$	1/3 1 0	0	$\pi-$ *rotation about* $(0,1,0)$
g_8	$\bar{y}\ \bar{x}\ \bar{z}$	2/3 1 0	0	$\pi-$ *rotation about* $(1,\bar{1},0)$
g_9	$x\text{-}y\ \bar{y}\ \bar{z}$	1 1 0	0	$\pi-$ *rotation about* $(1,0,0)$
g_{10}	$x\ x\text{-}y\ \bar{z}$	4/3 1 0	0	$\pi-$ *rotation about* $(2,1,0)$
g_{11}	$y\ x\ \bar{z}$	5/3 1 0	0	$\pi-$ *rotation about* $(1,1,0)$
g_{12}	$y\text{-}x\ y\ \bar{z}$	0 1 0	0	$\pi-$ *rotation about* $(1,2,0)$
$g_{12+i} = g_i$ for i=1…12				
g_{25}	$\bar{x}\ \bar{y}\ \bar{z}$	0 0 0	1	*inversion*
g_{26}	$y-x\ \bar{x}\ \bar{z}$	1/3 0 0	1	$4\pi/3-$ *rotation about* $(0,0,1)$
g_{27}	$y\ y-x\ \bar{z}$	2/3 0 0	1	$5\pi/3-$ *rotation about* $(0,0,1)$
g_{28}	$x\ y\ \bar{z}$	1 0 0	1	*reflection in plane* $(0,0,1)$
g_{29}	$x-y\ x\ \bar{z}$	4/3 0 0	1	$\pi/3-$ *mirror rotation about* $(0,0,1)$
g_{30}	$\bar{y}\ x-y\ \bar{z}$	5/3 0 0	1	$2\pi/3-$ *mirror rotation about* $(0,0,1)$

g_{31}	$x\ x-y\ \bar{z}$	1/3 1 0	1	*reflection in plane* (0,1,0)
g_{32}	$y\ x\ z$	2/3 1 0	1	*reflection in plane* (1,1,0)
g_{33}	$y\ y-x\ z$	1 1 0	1	*reflection in plane* (1,0,0)
g_{34}	$\bar{x}\ y-x\ z$	4/3 1 0	1	*reflection in plane* (2,1,0)
g_{35}	$\bar{y}\ \bar{x}\ z$	5/3 1 0	1	*reflection in plane* (1,1,0)
g_{36}	$x-y\ \bar{y}\ z$	0 1 0	1	*reflection in plane* (1,2,0)
$g_{12+i} = g_i$ for i=25…36				

Table 3: The double point groups and the double space groups to which they belong

Double point group index	Double point group name	Double point group operators, belonging to the double point group	Names and labels of the double space groups to which this double point group belongs
1	$^2C_1$	h(1,25)	$^2C^1_1$ = 1
2	$^2C_i = {}^2S_2$	h(1,13,25,37)	$^2C^1_i$ = 2
3	$^2C_2$	h(1,4,25,28)	$^2C^1_{2,1} - {}^2C^3_{2,1}$ = 3-5
4	^2C_s	h(1,25,52,76)	$^2C^1_{s,1} - {}^2C^4_{s,1}$ = 6-9
5	$^2C_{2h}$	h(1,4,25,28,49,52,73,76)	$^2C^1_{2h,1} - {}^2C^6_{2h,1}$ = 10-15
6	$^2D_2$	h(1-4,25-28)	$^2D^1_2 - {}^2D^9_2$ = 16-24
7	$^2C_{2v}$	h(1,4,25,28,50,51,74,75)	$^2C^1_{2v} - {}^2C^{22}_{2v}$ = 25-46
8	$^2D_{2h}$	h(1-4,25-28,49-52,73-76)	$^2D^1_{2h} - {}^2D^{28}_{2h}$ = 47-74
9	$^2C_4$	h(1,4,14,15,25,28,38,39)	$^2C^1_4 - {}^2C^6_4$ = 75-80
10	$^2S_4$	h(1,4,25,28,62,63,86,87)	$^2S^1_4$ = 81, $^2S^2_4$ = 82
11	$^2C_{4h}$	h(1,4,14,15,25,28,38,39,49,52,62, 63,73,76,86,87)	$^2C^1_{4h} - {}^2C^6_{4h}$ = 83-88
12	$^2D_4$	h(1-4,13-16,25-28,37-40)	$^2D^1_4 - {}^2D^{10}_4$ = 89-98
13	$^2C_{4v}$	h(1,4,14,15,25,28,38,39,50,51,61, 64,74,75,85,88)	$^2C^1_{4v} - {}^2C^{12}_{4v}$ = 99-110
14	$^2D_{2d}$	h(1-4,25-28,61-64,85-88)	$^2D^1_{2d} - {}^2D^{12}_{2d}$ = 111-122
15	$^2D_{4h}$	h(1-4,13-16,25-28,37-40,49-52, 61-64,73-76,85-88)	$^2D^1_{4h} - {}^2D^{20}_{4h}$ = 123-142
16	$^2C_3$	g(1,3,5,13,15,17)	$^2C^1_3 - {}^2C^3_3$ = 143-145, $^2C^4_{3,2}$ = 146
17	$^2C_{3i} = {}^2S_6$	g(1,3,5,13,15,17,25,27,29,37,39,41)	$^2C^1_{3i}$ = 147, $^2C^2_{3i}$ = 148
18	$^2D_3$	g(1,3,5,8,10,12,13,15,17,20,22,24)	$^2D^1_3$ = 149, $^2D^3_3$ = 151, $^2D^5_3$ = 153
19	$^2D_3$	g(1,3,5,7,9,11,13,15,17,19,21,23)	$^2D^2_3$ = 150, $^2D^4_3$ = 152, $^2D^6_3$ = 154, $^2D^7_{3,2}$ = 155
20	$^2C_{3v}$	g(1,3,5,13,15,17,31,33,35,43,45,47)	$^2C^1_{3v}$ = 156, $^2C^3_{3v}$ = 158, $^2C^5_{3v,2}$ = 160, $^2C^6_{3v,2}$ = 161
21	$^2C_{3v}$	g(1,3,5,13,15,17,32,34,36,44,46,48)	$^2C^2_{3v}$ = 157, $^2C^4_{3v}$ = 159
22	$^2D_{3d}$	g(1,3,5,7,9,11,13,15,17,19,21,23,25	$^2D^3_{3d}$ = 164, $^2D^4_{3d}$ = 165,

		27,29,31,33,35,37,39,41,43,45,47)	$^2D^5_{3d}$ = 166, $^2D^6_{3d,2}$ = 167
23	$^2D_{3d}$	g(1,3,5,8,10,12,13,15,17,20,22,24 25,27,29,32,34,36,37,39,41,44, 46,48)	$^2D^1_{3d}$ = 162, $^2D^2_{3d}$ = 163
24	$^2C_6$	g(1-6,13-18)	$^2C^1_6 - {}^2C^6_6$ = 168-173
25	$^2C_{3h}$	g(1,3,5,13,15,17,26,28,30,38,40,42)	$^2C^1_{3h}$ = 174
26	$^2C_{6h}$	g(1-6,13-18,25-30,37-42)	$^2C^1_{6h}$ = 175, $^2C^2_{6h}$ = 176
27	$^2D_6$	g(1-24)	$^2D^1_6 - {}^2D^6_6$ = 177-182
28	$^2C_{6v}$	g(1-6,13-18,31-36,43-48)	$^2C^1_{6v} - {}^2C^4_{6v}$ = 183-186
29	$^2D_{3h}$	g(1,3,5,8,10,12,13,15,17,20,22,24, 26,28,30,31,33,35,38,40,42,43, 45,47)	$^2D^1_{3h}$ = 187, $^2D^2_{3h}$ = 188
30	$^2D_{3h}$	g(1,3,5,7,9,11,13,15,17,19,21,23,26, 28,30,32,34,36,38,40,42,44,46, 48)	$^2D^3_{3h}$ = 189, $^2D^4_{3h}$ = 190
31	$^2D_{6h}$	g(1-48)	$^2D^1_{6h} - {}^2D^4_{6h}$ = 191-194
32	^2T	h(1-12,25-36)	$^2T^1 - {}^2T^5$ = 195-199
33	^2T_h	h(1-12,25-36,49-60,73-84)	$^2T^1_h - {}^2T^7_h$ = 200-206
34	^2O	h(1-48)	$^2O^1 - {}^2O^8$ = 207-214
35	^2T_d	h(1-12,25-36,61-72,85-96)	$^2T^1_d - {}^2T^6_d$ = 215-220
36	^2O_h	h(1-96)	$^2O^1_h - {}^2O^{10}_h$ = 221-230

3. Program SYMPRJS

This is the main program, which calculates the coefficients to the spherical harmonics, so that these functions form irreducible bases for the given space group and wave vector.

This program uses all datasets prepared by program **prodats**.

3.1 Description of the input

The input to the program shall consist of a sequence of sets in the following order:

1. There shall be 20 logical 1's (true) and 0's (false). These logical parameters are read into an array **steer(1:20)** and the parameters steer the amount of output data that will be printed. See Table 4 for the meaning of **steer(I)** = 'true' or 'false' for each **I**.

Table 4: Conventions for input data steer

I	If **steer(I)** ~= 0	If **steer(I)** == 0
1	The multiplication table is printed for each group for which the irreducible representations are calculated (function Irrep).	No print
2	Calculate the irreducible representatives (function Irrep) and not only the irreducible characters. Using Irrep as a function of SYMPRJS/SYMPWS one should have **steer(2)** = 'true' in all cases.	Calculate only the irreducible characters.
3	Print the inverse group elements (function **inverse**)	No print.
4	Print the number of generators **nmberg**, the group indices of the generators **ngen(I)**, the map **map(1:G,1:2)** by which each group element can be constructed as a product of generators (function **genera**). Print loop structure (function **permu**) of the group element with a unique eigenvalue (function **repres**).	No print.
5	Print the number of classes and the group elements in each class (function **classes**).	No print.
6	Print the table of primes (function **primen**). Print the exponent **ex** of the group, the prime **P** used in the calculations of the present group, **Zprim**, the used primitive root of unity modulus **P**, the characters as sums of roots of unity (function **charac**).	No print.
7	Print the dimensions **lj(1:ncl)** of the irreducible representations and the irreducible characters **ch(1:ncl,1:ncl)** as complex numbers (function	No print

	charac). Print the numbering of the irreducible representations (function **repres**).	
8	Print the one-dimensional group representation (function **repres**).	No print.
9	Print the irreducible representations of the group of dimension higher than one (function **repres**).	No print.
10	Not used.	No print.
11	In the input, **steer(11)** should always be “true”. This means that no error has occurred so far for this group. The program may change the value of the **steer(11)** if it detects an error.	Stop execution for the present group and continue with the next wave vector, giving another group.
12-17	No effect. These parameters can be neglected or used for own purposes.	No effect.
18	Print messages about the tests on a non-symmorphic space group.	No print
19	Print the irreducible characters, where the irreducible representations are numbered in the same way the main output. When using SYMPRJS/SYMPWS, this output should be used to identify the irreducible representations, instead of that of **steer(7)**.	No print
20	The space group is symmorphic. Information to the program.	Nonsymmorphic space group.

2. There shall follow three lines with three numbers each, defining the three rows of a matrix $\mathbf{A}$. The matrix $\mathbf{A}$ describes the primitive unit cell vectors $\bar{a}_1, \bar{a}_2, \bar{a}_3$ according to $(\bar{a}_1, \bar{a}_2, \bar{a}_3) = (\bar{e}_x, \bar{e}_y, \bar{e}_z)\mathbf{A}$

   where $(\bar{e}_x, \bar{e}_y, \bar{e}_z)$ are the unit vectors in the Cartesian coordinate system.

3. Three numbers are read, **pgnr**, **nel** and **lmax**. **pgnr** is the index number of the point group which belongs to the space group, as described in section 2. The indices are given in Table 3, column 1. **nel** is the number of chemical elements in the crystal. **lmax** is the maximum value of orbital quantum number l, for which one wants to make calculations. Maximum value for **lmax** is 3.

4. There is a set of **nel** numbers, denoting the number of atoms per unit cell for each of the **nel** chemical elements. These numbers are stored into **nat(1:nel)**.

5. $\sum_{I=1}^{nel} nat(I)$ triplets, each triplet containing the coordinates of one atom. First the atoms of chemical element no. 1, then the atoms of chemical element no. 2 etc. The integer **atco** in the first column informs the program about the coordinate system in which the coordinates are given. If **atco** = 1, the numbers are coordinates to the Cartesian vectors $(\bar{e}_x, \bar{e}_y, \bar{e}_z)$. If **atco** = 0 the numbers are coordinates to the lattice

vectors $(\bar{a}_1, \bar{a}_2, \bar{a}_3)$.

6. This set of numbers must be given only for a nonsymmorphic space group. **steer(20)** must be set to 'false' (0) for a nonsymmorphic space group. If **steer(20)** = 'true' the program assumes that the space group is symmorphic and it will attempt to read the set of numbers in point 7, immediately after those of point 5.

   For nonsymmorphic space groups the input should consist of **order** numbers, where **order** is the number of elements in the point group of the space group, the index of which (**pgnr**) was given as input earlier (point 3). The nth set should contain the nonprimitive translation associated with the nth point group element, where the point group elements are given in the order of Table 3, column 3.

   For example, if **pgnr** = 9, the point group is $^2C_4$ and there should be 8 sets if the space group is nonsymmorphic. Then the second set for example, should give the coordinates of the nonprimitive translation to the second point group operator, which according to Table 3 is h_4. Table 1 informs that h_4 is the rotation through angle π about (0,0,1).

   Even when there is no nonprimitive translation associated with a particular point group operator, it should be given as a zero vector, in order to obtain as many sets as there are point group operators and to have them in the required order. The integer **uco** in the first column of each set informs the program about the coordinate system in which the nonprimitive translations are given. **uco** = 1 means Cartesian coordinates. **uco** = 0 means lattice coordinates.

7. The number of $\bar{k}$-vectors (**number_of_wave_vectors**) that follows shall be supplied. The following set contains the wave vectors $\bar{k}$ for which the symmetry projection matrices will be calculated. Each set contains five numbers:

   last, **wvco**, **rk(1)**, **rk(2)**, **rk(3)**, **nfacto**

   Here **last** = 1 means that this is the end of the input. For all wave vectors one should have **last** = 0

   wvco = 1 means that the wave vector is given in Cartesian coordinates, **wvco** = 0 that it is given in reciprocal lattice coordinates $(\bar{b}_1, \bar{b}_2, \bar{b}_3)$ with

$$\bar{b}_i = \frac{\bar{a}_j \times \bar{a}_k}{\bar{a}_i \cdot (\bar{a}_j \times \bar{a}_k)}$$

   The input coordinates should be given in units of 2π, since they are multiplied by 2π as soon as they have been read.

   The number **nfacto** must be set equal to zero if the symmetry projection should be done for only one wave vector in the same direction. Then the next wave vector set may follow immediately after this set.

If however, the symmetry projection should be repeated for several wave vectors with the same direction but different lengths, **nfacto** should be set equal to the number of such wave vectors (the first one included). In the case that **nfacto** > 0, there must be added a set of **nfacto** factors by which the given vector must be multiplied, in order to obtain the other wave vectors for which the symmetry adaptation should be made. The factors must be given in decreasing order and the first factor must be equal to 1. Then comes the set for the next wave vector.

The last set should contain the information **last** = 1 in the first column. A wave vector occurring in this set is not treated, the program stops executing after writing 'SYMPRJS has finished executing'.

You will have to update the following statements in SYMPRJS.m:

read_inputdata = fopen('xx.dat','r');	where xx is the name of your ASCII input data file.
fid = fopen('yy.m','a');	where yy is the name of your .m output data file.
write_proj = fopen('zz','a');	where zz is the name of your binary output data file.

3.2 Formula, which has been programmed

We refer to references [4] and [5] for the theory, which leads to the following formula for the projection matrix.

$$((((((^{j\bar{k}}S_{dd})_{cc})_{ll})_{\bar{\mu}\bar{\nu}})_{m'm})_{s's} = \frac{l_{j\bar{k}}}{g'_{\bar{k}}} \sum_{(P|\bar{u})\in G(\bar{\mu},\bar{\nu})} {}^{j\bar{k}}\Gamma_A((P\,|\,\bar{u}))^*_{dd}\, e^{-i\bar{k}\cdot\bar{n}_c(F,\bar{\mu},\bar{\nu})}(-1)^{\lambda l}\,{}^{l}D(\phi,\theta,\psi)_{m'm}\,{}^{1/2}D(\phi,\theta,\psi)_{s's}$$

The space group G of the crystal consists of the operators $F = (P_i\,|\,\bar{u}_i + \bar{n})$. $\bar{k}$ is the input wave vector. $G_{\bar{k}}$ is the little group of the second kind, or group of the wave vector, it consists of space group operators $(P_i\,|\,\bar{u}_i + \bar{n})$ with $P_i \in P_{\bar{k}}$. The little point group $P_{\bar{k}}$ consists of the point group operators P_i with $(P_i\,|\,\bar{u}_i) \in G$ and $P_i\bar{k} = \bar{k} + \bar{K}$, where $\bar{K}$ is a reciprocal lattice vector.

j is the index of the irreducible representation of $G_{\bar{k}}$. These irreducible representations are formed from those of the group $P_{\bar{k}}$ or, for the case $\bar{k}$ lies on the Brillouin zone boundary and $G_{\bar{k}}$ is nonsymmorphic, from the allowable irreducible representations of the factor group $G_{\bar{k}}/T_{\bar{k}}$. Here $T_{\bar{k}}$ is the group of lattice translations $(E\,|\,\bar{n})$ for which $e^{-i\bar{k}\cdot\bar{n}}=1$. An irreducible representation of $G_{\bar{k}}/T_{\bar{k}}$ is allowable if an element of the coset $(E\,|\,\bar{m})T_{\bar{k}}$ is represented by $e^{-i\bar{k}\cdot\bar{m}}$ times the unit matrix.

$l_{j\bar{k}}$ is the dimension of the irreducible representation.

${}^{j\bar{k}}\Gamma_A((P\,|\,\bar{u}))^*_{dd}$ is the complex conjugate of the dth diagonal element of the representative of $(P\,|\,\bar{u})$ in the allowable irreducible representation ${}^{j\bar{k}}\Gamma_A$ of $G_{\bar{k}}$. For symmorphic groups and nonsymmorphic groups with a symmorphic $G_{\bar{k}}$ or $\bar{k}$ within the first Brillouin zone boundary, all irreducible representations are allowable and the subscript A is superfluous. $G(\bar{\mu},\bar{\nu})$ is the set of elements $(P\,|\,\bar{u})\in G_{\bar{k}}$ with $(P\,|\,\bar{u})^{-1}\,\bar{\mu}=\bar{\nu}+\bar{n}(F,\bar{\mu},\bar{\nu})$ where $\bar{n}(F,\bar{\mu},\bar{\nu})$ is a lattice vector. c is the index for the chemical element. $g'_{\bar{k}}$ is the order of $G(\bar{\mu},\bar{\nu})$.
λ is the inversion label of the point group operator P and ${}^{l}D(\phi,\theta,\psi)_{m'm}$ is defined in section 1. This formula corresponds to formula (5.38) in reference [4] which we refer to for the theory.

3.3 Description of the program

The projection matrix defined in the previous section is block diagonal with respect to the subscripts c and l. For each particular c and l there is a diagonal block, which is a projection matrix by itself. One can construct a matrix ${}^{j\bar{k}}\mathbf{t}_{cl}$ of orthonormal columns from each of those sub-projection matrices. These matrices form the output of the program. They give the coefficients for the harmonics which are adapted to the symmetry of the space group. This output is given for all wave vectors of the input and all irreducible representations. The text in the output is explaining the details, so it is not necessary to read through this section to understand the form of the output.

This section gives a detailed description of the functioning of the program and comments to possible input or program errors.

The program SYMPRJS can be divided into 9 sections.

Section 1. Input (See chapter 3.1 for the sequence of input data).

1.1 Read input 1 (the numbers refer to the points in chapter 3.1).

1.2 Read input 2, matrix $\mathbf{A}$ with $(\bar{a}_1, \bar{a}_2, \bar{a}_3) = (\bar{e}_x, \bar{e}_y, \bar{e}_z)\mathbf{A}$.

1.3 Print input 2.

1.4 Calculate matrix $\mathbf{B}$ so that $(\bar{b}_1, \bar{b}_2, \bar{b}_3) = (\bar{e}_x, \bar{e}_y, \bar{e}_z)\mathbf{B}$ form the reciprocal lattice vectors. It is easy to prove that $\mathbf{B}$ is the inverse of the transpose of $\mathbf{A}$: $\mathbf{B} = (\mathbf{A}^{-1})^T$. Store $\mathbf{B}^{-1}$ in **bi** and $\mathbf{A}^{-1}$ in **ai**.

1.5 Print the matrix $\mathbf{B}$.

1.6 Read the input 3: **pgnr**, **nel**, **lmax.**

1.7 Print **nel**, the number of chemical elements.

1.8 Read input 4.

1.9 Read input 5 and transform, if necessary to Cartesian coordinates.

1.10 Print the number of atoms of each chemical element and their positions, according to 1.8 and 1.9.

1.11 Read **order**, the order of the point group with index **pgnr** and the position of the first element of this point group (**first**).

1.12 Read the elements of the point group into **gel(1:order)**. Read the primes into **npri(1:100)**.

1.13 For nonsymmorphic space groups (**steer(20)** = 'false'): Read the nonprimitive translations associated with each point group operator into **u(1:order,1:3)**. Transform if necessary to lattice coordinates.

1.14 If the point group is a subgroup of $^2D_{6h}$ (16 ≤ **pgnr** ≤ 31), go to 1.16.

1.15 Print "The point group no '**pgnr**' of the crystal is a subgroup of O_h". Set the reading index **K96** = 0. Read from the multiplication table of ^2O_h, the rows with row indices **gel(1:order)** into **mtab(I,K)**. If **pgnr** = 36 the point group is ^2O_h itself, the complete table is read into **mtab**, no selection of columns has to be made. Go in that case directly to 1.18. Otherwise, go to 1.17.

1.16 Read from the multiplication table of $^2D_{6h}$, the rows with row indices **gel(1:order)** into **mtab(I,K)**. Set the read index **K96** = 96. Print "The point group no '**pgnr**' of the crystal is a subgroup of D_{6h} with element nrs". If **pgnr** = 31 the point group is $^2D_{6h}$ itself, no selection of columns has to be made. In this case, go directly to 1.18.

1.17 Select in **mtab** the columns with column indices **gel(1:order)** and then shift to the left, so that the upper left **order** x **order** block of **mtab** becomes the multiplication table, where the elements are indexed as in ^2O_h or $^2D_{6h}$. Renumber the elements from 1 to **order** using the matrix **inver(I)** and obtain finally the multiplication table of the point group.

1.18 Print the element numbers **gel(1:order)**. This output follows immediately the printing under 1.15 or 1.16. Print the maximum number of orbital quantum number l, **lmax**, according to 1.6.

1.19 Read the three dimensional orthogonal rotation matrices (or rotation-inversion matrices) corresponding to the elements of the point group G_P of the space group. The position of these matrices is determined by the value of **K96** + **gel(1:order)**.

1.20 Read the number of wave vectors for which the projection matrices shall be calculated, into variable **number_of_wave_vectors**. This variable as well as **nel**, **nat(1:nel)** and **lmax(1:nel)** are written to the binary output file. Read a wave vector for which symmetry projection matrices must be calculated. Further read the number **nfacto** which is the number of wave vectors in the same direction for which the calculation should be repeated. If the index **last** in column 1 is equal to 1, go to section 9. Transform, if necessary the given coordinates of the wave vector to reciprocal lattice coordinates. If **nfacto** > 1, read the factors by which the original vector should be multiplied to obtain the successive wave vectors in the same direction. The original vector is stored into **ark(1:3)**.

1.21 The index **IV** enumerates the number of vectors in the same direction for which a symmetry projection has been made. Set **IV** = 1 and set the factor by which the original vector should be multiplied, for the first calculation equal to 1.

1.22 If all vectors for the present direction have been treated (**IV** > **nfacto, nfacto** > 1 or **IV >** 1**, nfacto =** 0) go to 1.20 to read a new wave vector.

1.23 If **IV** = 1, go to 2.1 for the formation of $P_{\bar{k}}$, the point group of the wave vector. If the original vector (**IV** = 1, **nfacto** > 1) lies on the Brillouin zone boundary, the next vector (**IV** = 2) may have lower symmetry: Go to 2.1 to form $P_{\bar{k}}$. If **IV** > 2 and **IV** ≤ **nfacto**, the new vector will have the same symmetry as the preceding one. $P_{\bar{k}}$ and its irreducible representations do not have to be calculated again. In this case, go to section 8. If $\bar{k} = \bar{0}$, $P_{\bar{k}} = G_P$, go to 2.2.

Section 2. Formation of the point group of the wave vector $P_{\bar{k}}$.

2.1 Multiply the original wave vector with **factor(IV)** and 2π, **kgord** will be the order of $P_{\bar{k}}$. **kgel(I)** will be the index of the Ith element of $P_{\bar{k}}$, where the index refers to the enumeration **1**:**order** of point group G_P. Transform the wave vector to Cartesian coordinates and operate on it successively with all the rotation matrices of group

G_P. Transform back to reciprocal lattice coordinates and subtract the original wave vector. The result is $P\bar{k}-\bar{k}$. If $P \in P_{\bar{k}}$ this difference will be a reciprocal lattice vector $\bar{K}$. Since $\bar{k}$ lies within or on the first Brillouin zone boundary, the components $\bar{K}$ (in reciprocal lattice coordinates) can only be 0 or $\pm 2\pi$. Test the components of $P\bar{k}-\bar{k}$ for these values. If each component is equal to 0 or $\pm 2\pi$, the element P belongs to $P_{\bar{k}}$ and is registered as such. When all elements P of G_P have been tested, go to 2.3.

2.2 If $\bar{k}=\bar{0}$ then $P_{\bar{k}} = G_P$ and we register the elements of $P_{\bar{k}}$ in this way into **kgel**. The indices of the elements in the enumeration of ${}^2O_h/{}^2D_{6h}$ are registered into **kkgel**. Further $\bar{k}$ lies of course within the first Brillouin zone. Set **ibz** = ‘true’. The multiplication table **mtab2** of $P_{\bar{k}}$ is in this case equal to the multiplication table **mtab** of G_P. Go to section 5.

2.3 Form the multiplication table **mtab2** of $P_{\bar{k}}$ from the multiplication table **mtab** of G_P, using the information in **kgel**. Register the indices of the elements of $P_{\bar{k}}$ in the enumeration of the elements of ${}^2O_h/{}^2D_{6h}$ into **kkgel**. If G is symmorphic (**steer(20)** = ‘true’), go to section 5.

Section 3. Tests for the nonsymmorphic space group.

The vector $\bar{k}$ is tested by function **bztest**.

3.1 If $\bar{k}$ lies within the first Brillouin zone, **ibz** is set to ‘true’.

If $\bar{k}$ lies on the first Brillouin zone boundary, **ibz** is set to ‘false’.

If $\bar{k}$ lies outside the first Brillouin zone, the user have supplied unallowed input, a message is printed and control goes to section 1.20.

bztest tests if $|\bar{k}\cdot\bar{c}| \leq \frac{1}{2}\bar{c}\cdot\bar{c}$ for $\bar{c} = \bar{b}_1, \bar{b}_2, \bar{b}_3, \bar{b}_1+\bar{b}_2, \bar{b}_1+\bar{b}_3, \bar{b}_2+\bar{b}_3, \bar{b}_1+\bar{b}_2+\bar{b}_3$.

If $\bar{k}$ itself does not lie in the first octant, then the appropriate signs are changed, for example $\bar{c} = -\bar{b}_1, \bar{b}_2, -\bar{b}_3$ etc.

If **steer(18)** = ‘true’ and **ibz** = ‘true’ print “Nonsymmorphic but within Bz”. If **ibz** = ‘true’, go to section 5.

3.2 The space group $G_{\bar{k}}$ associated with $P_{\bar{k}}$ may be symmorphic even when the complete space group is nonsymmorphic. If $G_{\bar{k}}$ is symmorphic, **ksym** is set to 'true', otherwise **ksym** = 'false'. $G_{\bar{k}}$ is symmorphic if the "nonprimitive" translations **u(I,K)**, associated with the point group operators of $P_{\bar{k}}$ are all equal to zero translations. If **steer(18)** = 'true' and **ksym** = 'true', print "Nonsymmorphic but symmorphic $G_{\bar{k}}$". If **ksym** = 'true', go to section 5.

Section 4. Formation of the factor group $G_{\bar{k}}/T_{\bar{k}}$.

If G is nonsymmorphic and $\bar{k}$ is on the Brillouin zone boundary and $G_{\bar{k}}$ is nonsymmorphic, one has to form the factor group $G_{\bar{k}}/T_{\bar{k}}$ in order to form a projection matrix, based on the irreducible representations of this group (see for example reference [4]).

$T_{\bar{k}}$ is the subgroup of the translation group of the crystal lattice such that if $(E\,|\,\bar{m}) \in T_{\bar{k}}$ then $e^{-i\bar{k}\cdot\bar{m}} = 1$. The elements of $G_{\bar{k}}/T_{\bar{k}}$ are the cosets of $G_{\bar{k}}$ with the respect to $T_{\bar{k}}$. Two elements of $G_{\bar{k}}$, $(P_i\,|\,\bar{u}_i + \bar{n})$ and $(P_j\,|\,\bar{u}_j + \bar{m})$ belong to the same coset if $i = j$ and $e^{-i\bar{k}\cdot(\bar{u}_i+\bar{n})} = e^{-i\bar{k}\cdot(\bar{u}_j+\bar{m})}$. So the elements of $G_{\bar{k}}/T_{\bar{k}}$ can be characterized by the index of the point group operator and the value of the exponential term. We form the group $G_{\bar{k}}/T_{\bar{k}}$, starting with the elements $(\bar{P}_i\,|\,\bar{u}_i)$ with $P_i \in P_{\bar{k}}$ and the multiplication table of $P_{\bar{k}}$. These elements form the first **kgord** elements of $G_{\bar{k}}/T_{\bar{k}}$ (**kgord** is the order of $P_{\bar{k}}$). We form the multiplication table, using the multiplication rule

$$(P_i\,|\,\bar{u}_i)(P_j\,|\,\bar{u}_j) = (P_iP_j\,|\,P_i\bar{u}_j + \bar{u}_i) = (P_l\,|\,\bar{t}) = (P_l\,|\,\bar{u}_l + \bar{n})$$

The index l is determined from the multiplication table of $P_{\bar{k}}$.

If $e^{-i\bar{k}\cdot\bar{t}} \neq e^{-i\bar{k}\cdot\bar{u}_l}$ then $(P_l\,|\,\bar{t})$ does not belong to the coset of $(P_l\,|\,\bar{u}_l)$, but to a new coset with the same index for the point group operator but a different value for the exponential term. These two indices determine a new element of the group $G_{\bar{k}}/T_{\bar{k}}$.

k2gord is the order of $G_{\bar{k}}/T_{\bar{k}}$. At the beginning of the process **k2gord** = **kgord**. **listp(l)** is the index of the point group operator (in the enumeration of the elements of $P_{\bar{k}}$).

listp(1:kgord). **til(l,1:3)** are the vectors $\bar{t}$. For the first **kgord** elements these are equal to the $\bar{u}_i$ corresponding to the $P_i \in P_{\bar{k}}$. **s(I)** are the exponential values $e^{-i\bar{k}\cdot\bar{t}}$.

nopi(1:kgord) are the numbers of group elements of $G_{\bar{k}}/T_{\bar{k}}$ with the same point group operator index I. The Kth operator, which has index I for its point group operator, is the **nopli(I,K)**th element of $G_{\bar{k}}/T_{\bar{k}}$. Its exponential term is equal to **sil(I,K)**. We go through the multiplication table row by row, forming the elements in each row. If a new element is created, the multiplication table will obtain a new row and a new column. For all preceding rows this new element in the last column must be calculated, so the program returns to row 2, last column. The number of elements, already determined in row I is **nr(I)**. When a new element in row I has been calculated, **nr(I)** is increased by one. The point group index of a product is calculated by the statement **index1** = **mtab2(N1,N2)**.

$P_i\bar{u}_j + \bar{u}_i$ is calculated straightforwardly. Its exponential value, **sres**, is compared with the given exponential values for this point group operator, that is with **sil(index1,1:nopi(index1))**. If it is equal to **sil(index1,K1)**, the element in the multiplication table is set equal to **nopi(index1,K1)**.

If it is not equal to any of the given exponentials, we have found a new element: **k2gord** is increased by one, the multiplication table is enlarged with one row and column and the program sets the row count to 2 in order to calculate the element in the new column.

After a few such steps no new elements have to be added and the multiplication table is calculated as described above.

At the end of the process the order of $G_{\bar{k}}/T_{\bar{k}}$ is put into **kgord** and its multiplication table into **mtab2** in order to be used by function **Irrep**. The point group indices, as numerated in group $^2O_h/^2D_{6h}$ are put into **Ilistp**.

Section 5. Wave vectors in the same direction.

In the case that the new wave vector has the same symmetry as the preceding one, the diagonal elements of the irreducible representations of $P_{\bar{k}}$ are still stored. Go immediately to section 8.

Section 6. Calculation of the diagonal elements of the irreducible representations.

6.1 First print a few lines of output:
Print “Projection matrices for the wave vector **srk**”
Print “The point group of the wave vector consists of **kg** operators, indexed as nrs. **kkgel**.”
If $G_{\bar{k}}/T_{\bar{k}}$ had to be formed, print “The factor group $G_{\bar{k}}/T_{\bar{k}}$ consists of **I** point group operators **Ilistp(I)**, nonprimitive translations **til(I,K)**, exp = **s(I)**.

6.2 Function **Irrep** will calculate the irreducible representations of the group of order **kgord** with multiplication table **mtab2**. It writes the diagonal elements of the irreducible representatives.
Function **Irrep** and its subfunctions have been described as an independent program in reference [1]. The following changes have been made: The group order and multiplication table have not to be read as input by **Irrep**, since they already exist. In **Irrep**, the value **kgord** is called **G** and **mtab2** is called **multab.**
A few statements have been added at the end of function **repres**, in order to write the diagonal elements of the (allowable) irreducible representations.
An irreducible representation of $G_{\bar{k}}/T_{\bar{k}}$ is allowable if an element of the coset $(E\,|\,\bar{m})T_{\bar{k}}$ is represented by $e^{-i\bar{k}\cdot\bar{m}}$ times the unit matrix. The dimensions of the irreducible representations are stored in increasing order into **laj(J)**. If **steer(19)** = 'true', the irreducible character will be printed. Function **charac**, a function called by **Irrep**, uses the primes stored in **npri.**

6.3 The number of projection matrices is equal to the sum of the dimensions of the (allowable) irreducible representations. This number is stored into **nup**.
Each projection matrix is blockdiagonal for the index of the chemical element, which runs from 1 to **nel** and for the orbital quantum number l which runs from 0 to **lmax.**
So each projection matrix consists of **nel**·(**lmax** + 1) blocks along the main diagonal. This number is stored into **nblock**. Each block is itself a projection matrix.

Section 7. Formation of the summation sets $G(\bar{\mu},\bar{\nu})$.

Each block of a projection matrix, with fixed indices for the chemical element and orbital quantum number, which itself forms a projection matrix, can be divided further into subblocks labelled by the atomic positions $\bar{\mu},\bar{\nu}$ of the actual chemical element in the unit cell. In the calculation of such a subblock, the elements of $P_{\bar{k}}$ (or $G_{\bar{k}}/T_{\bar{k}}$) give only then a contribution, if they belong to the set $G(\bar{\mu},\bar{\nu})$ defined by

$$(P_i\,|\,\bar{u}_i) \in G(\bar{\mu},\bar{\nu}) \text{ if } (P_i\,|\,\bar{u}_i)^{-1}\,\bar{\mu} = \bar{\nu} + \bar{n}(i,\bar{\mu},\bar{\nu})$$

where $\bar{n}(i,\bar{\mu},\bar{\nu})$ is a lattice vector (see chapter 3.2).

In this section we form the sets $G(\bar{\mu},\bar{\nu})$ with the corresponding lattice vectors $\bar{n}(i,\bar{\mu},\bar{\nu})$ for all chemical elements. **np(K,I,J)** is the order of the set of space group operators F, for chemical element K, for which
F*(coordinate of atom I) – (coordinate of atom J) is a lattice vector.
npl(K,I,J,1:np(K,I,J)) are the indices of the space group operators F, which belong to this set. **nvec(K,I,J,L,1:3)** are the coordinates of the corresponding lattice vector.

The procedure is as follows: Initiate the index I1 for the chemical element, initiate the index of the first atom index I3 (corresponding to $\bar{\mu}$). Transform this atom position successively with all operators of $P_{\bar{k}}$ (or $G_{\bar{k}}/T_{\bar{k}}$) and check which new atom position, denoted by I9 (corresponding to $\bar{\nu}$), is obtained, up to a lattice vector.
If for chemical element I1, atom I3 is transformed to atom I9 by operator I4 of $P_{\bar{k}}$ (or $G_{\bar{k}}/T_{\bar{k}}$), up to a lattice vector **difi(1:3)**, then **np(I1,I3,I9)** is increased by one, **npl(I1,I3,I9,np(I1,I3,I9))** = **I4** and **nvec(I1,I3,I9,K,1:3)** = **difi(1:3)**. This process is repeated for all allowed values of I1 and I3. If it occurs that an atomic position is transformed to a position which is not a position of an atom of the same chemical element, this means that the input is in error. The space group given in the input does not describe the crystal symmetry of the input. In this case the program prints a message: “Wrong space group, I1, I3, I4, I5”. This informs the user that atom I3 of chemical element I1 is not transformed to an atom position of the same chemical element by operator I4 of $P_{\bar{k}}$ ($G_{\bar{k}}/T_{\bar{k}}$). I5 is the index of the point group operator corresponding to I4, in the enumeration of the point group of the space group. After this message the program goes to section 9. When all chemical elements have been treated, control goes to section 8.

Section 8. The formation of the projection matrices.

The projection matrices are formed according to the formula in chapter 3.2. All necessary information is now available.

8.1 Initiate the values for **J** (j), **JD** (d), the chemical element **ichem** (c), **L** (l) and the atom positions **mu1** $(\bar{\mu})$ and **mu2** $(\bar{\nu})$. If all values of these indices have been treated, go to section 1.20. **ncoset** is set equal to **np(ichem,mu1,mu2)**. This is the number of terms in the summation and is equal to the number of operators which transform atom **mu1** to atom **mu2**, up to a lattice vector.
If **ncoset** is zero, this means that the operations which transform atom **mu1** to atom **mu2** were not included in the input space group. This means that the actual symmetry is higher than that of the given space group. The following message is printed: “Space group is not maximal, **ichem**, **mu1**, **mu2**”. The case that the given space group is not the maximal space group, may also occur when **ncoset** is not zero, but has a lower value than it would have with the maximal space group. But such a case can not be detected by the program.

8.2 Initiate the values for **M1** (m) and **M2** (m’). Calculate the number **N3**, which determines the address of the elements $(-1)^{l\lambda}\,{}^{l}D_{mm'}$, and read them into **ldmm(1:144)**.

8.3 Calculate the terms and add them according to the formula in chapter 3.2. Perform 8.2 and 8.3 for all the allowed values of m and m’. The resulting sub-projection matrix is stored into **jdpk(K4,K5)**.

8.4 Test if the resulting matrix is idempotent and hermitian. If not, print a message.

8.5 Orthonormalise the columns of the sub-projection matrix and store the resulting rectangular **t**-matrix into **tmatri**. We make use of some properties of the projection matrix: If the diagonal terms of a column is equal to 0 or 1, all the other elements in that column vanish. We select first the columns with 1 in the diagonal terms; these are already orthonormal. Then we orthonormalise the remaining columns by the Schmidt procedure. The process is stopped as soon as we have got **ntr** columns, where **ntr** is the trace of the sub-projection matrix. If it is possible to select **ntr** orthonormal columns (for example by rounding off errors or because the program discarded some columns, since these had so small elements that errors would be large after normalisation) then the program prints an error message "Error, not enough orthonormal columns". Then the program goes to section 9. This error has never occurred when the authors ran the program. Therefore, one can perhaps discard the statements for the test and the error message.

8.6 Write the produced **t**-matrix. Go to section 8.1.

Section 9. End of the program.

Print "SYMPRJS has finished executing" and stop.

3.4 Program proj_symprjs.m

Program proj_symprjs.m reads the binary output file from SYMPRJS.m and orders the columns into a super projection matrix, where the submatrix blocks are ordered along the main diagonal as:

(chem_elem_1, atom_1, L = 0), (chem_elem_1, atom_1, L = 1),…,(chem_elem_1, atom_1, L = lmax(chem_elem_1), (chem_elem_1, atom_2, L = 0),…,(chem_elem_1, atom_2, L = lmax(chem_elem_1), (chem_elem_2, atom_1, L = 0),…, (chem_elem_2, atom_1, L = lmax(chem_elem_2),…

In proj_symprjs.m, the following statements have to be adapted to your naming the input/output files:

in_proj = fopen('xx', 'r'); where xx is your name for the SYMPRJS.m binary output file.
out_proj = fopen('yy', 'a'), where yy is your name for the proj_symprjs.m binary output file.

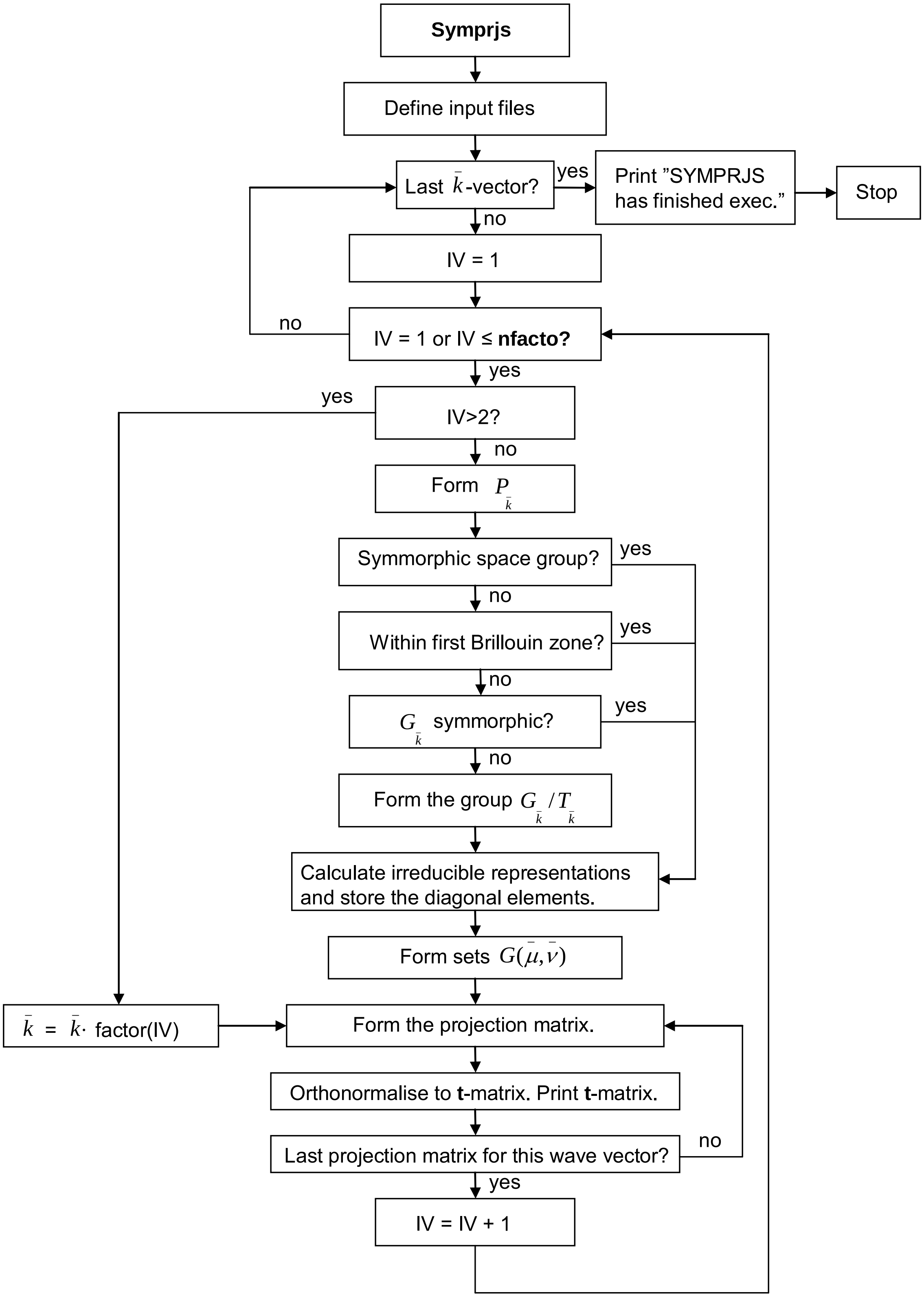
Symprjs
Define input files
Last $\bar{k}$-vector?
yes
Print "SYMPRJS has finished exec."
Stop
no
IV = 1
no
IV = 1 or IV ≤ **nfacto?**
yes
yes
IV>2?
no
Form $P_{\bar{k}}$
Symmorphic space group?
yes
no
Within first Brillouin zone?
yes
no
$G_{\bar{k}}$ symmorphic?
yes
no
Form the group $G_{\bar{k}}/T_{\bar{k}}$
Calculate irreducible representations and store the diagonal elements.
Form sets $G(\bar{\mu},\bar{\nu})$
$\bar{k} = \bar{k}\cdot$ factor(IV)
Form the projection matrix.
Orthonormalise to **t**-matrix. Print **t**-matrix.
Last projection matrix for this wave vector?
no
yes
IV = IV + 1

Figure 5: Flowing scheme of program Symprjs

4. Program SYMPWS

This is the main program, which calculates the coefficients to the plane waves, so that the linear combinations form irreducible basis functions for the given space group and wave vector.

Large parts of this program are identical with parts in program SYMPRJS. We shall refer for the description of those parts to the previous descriptions in chapter 3. This program uses some of the datasets prepared by program **prodats**.

4.1 Description of the input

The input of the program should consist of a sequence of data sets in the following order.

1 There shall be 20 logical 1's (true) and 0's (false). These logical parameters are read into an array **steer(1:20)** and the parameters steer the amount of output data that will be printed. See Table 4 for the meaning of **steer(I)** = 'true' or 'false' for each I. Additionally to what is stated in Table 4, we have that if **steer(17)** = 'true' then it is printed how the stable basis of plane waves transforms under the point group $P_{\bar{k}}$.

See chapter 4.3, section 3.

2 Three sets with three numbers each, defining the three rows of a matrix $\mathbf{A}$. The matrix $\mathbf{A}$ describes the primitive unit cell vectors $\bar{a}_1, \bar{a}_2, \bar{a}_3$ according to

$(\bar{a}_1, \bar{a}_2, \bar{a}_3) = (\bar{e}_x, \bar{e}_y, \bar{e}_z)\mathbf{A}$

where $(\bar{e}_x, \bar{e}_y, \bar{e}_z)$ are the unit vectors in the Cartesian coordinate system.

3 One set, giving **pgnr**. This is the index of the point group which belongs to the space group, as described in chapter 2. The indices are given in Table 3, column 1.

4 The sets of this point must be given only for a nonsymmorphic space group. **steer(20)** of point 1 in the input must be set equal to 'false' for a nonsymmorphic space group. If **steer(20)** = 'true' the program assumes that the space group is symmorphic and it will attempt to read the sets under input 5 immediately after those of input 3.

For nonsymmorphic space groups the input should consist of **order** sets, where **order** is the number of elements in the point group of the space group, the index of which (**pgnr**) was given as input under input 3.

Set no. n should contain the nonprimitive translation associated with the nth point group element, where the point group elements are given in order of Table 3, column 3.

For example, if **pgnr** = 9, the point group is ${}^2C_4$ and there should be 8 sets under point 6 if the space group is nonsymmorphic. Then the second set for example, should give the coordinates of the nonprimitive translation associated to the second point group operator, which according to Table 3 is h_4. Table 1 informs then that h_4 is the rotation through angle π about (0,0,1).

Even when there is no nonprimitive translation associated with a particular point group operator, it should be given as a zero vector, in order to obtain as many sets as there are point group operators and to have them in the required order. The integer **uco** in the first column of each set, informs the program about the coordinate system in which the nonprimitive translations are given. **uco** = 1 means Cartesian coordinates. **uco** = 0 means lattice coordinates.

5 This set contains the wave vectors $\bar{k}$ and the reciprocal lattice vectors $\bar{K}_n$ for which the calculations will be made. The set containing the wave vector is:

last, **wvco**, **rk(1)**, **rk(2)**, **rk(3)**, **nrec**, **krep**

If **last** = 1, the program stops assuming that this is the last set in the input.

rk(1:3) are the coordinates of $\bar{k}$. When **wvco** = 1 these coordinates should be in Cartesian coordinates, when **wvco** = 0 in reciprocal lattice coordinates. The input coordinates should be given in units of 2π, they are multiplied by 2π in the program.

nrec is the number of sets, each with one reciprocal lattice vector that follow after the present set.

For the first $\bar{k}$-vector it is necessary that **krep** = 0, this is explained below.

Then follow **nrec** sets. Each set contains the reciprocal lattice coordinates of one reciprocal lattice vector $\bar{K}_n$. These are the vectors $\bar{K}_n$, defining the plane waves $\psi_{\bar{k}_n} = e^{(i(\bar{k}+\bar{K}_n)\cdot\bar{r})}$ that should be included in the basis, which will be symmetrized.

One does not have to bother if the given $\bar{K}_n$-vectors define a stable basis, since the program itself extends the basis to form a stable basis.

Thereafter follows the next $\bar{k}$-vector in the input. If one wants to use the same reciprocal lattice vectors $\bar{K}_n$ for this $\bar{k}$-vector as for the preceding one, one sets **krep** = 1. Then the program uses the same set of $\bar{K}_n$. Then one can give immediately the next $\bar{k}$-vector etc. One ends the input with a set with **last** = 1. Other information in that set is ignored, the program stops after writing ‘SYMPWS has finished executing’.

You will have to update the following statements in SYMPWS.m:

read_inputdata = fopen('xx.dat','r'); where xx is the name of your ASCII input data file.

fid = fopen(‘yy.m','a'); where yy is the name of your .m output data file.

write_proj = fopen(‘zz','a'); where zz is the name of your binary output data file.

4.2 Formula, which has been programmed

We refer to references [4] and [5] for the theory which leads to the following formula for the projection matrix:

$$((^{j\bar{k}}S_{dd})_{mn})_{s's} = \frac{l_{j\bar{k}}}{g'_{\bar{k}}} \sum_{P_i \in P_{\bar{k}}} {}^{j\bar{k}}\mathbf{\Gamma}_A((P_i \mid \bar{u}_i))^*_{dd}\,\mathbf{\Gamma}((P_i \mid \bar{u}_i))_{mn}{}^{1/2} D(P_i)_{s's}$$

with $\mathbf{\Gamma}((P_i \mid \bar{u}_i))_{mn} = e^{(-i(\bar{k}+\bar{K}_m)\cdot\bar{u}_i)}$ if $P_i(\bar{k}+\bar{K}_n) = \bar{k}+\bar{K}_m$ and otherwise $\mathbf{\Gamma}((P_i \mid \bar{u}_i))_{mn} = 0$. All other symbols are the same as in chapter 3.2. The stable basis consists of functions $\psi_{\bar{k}_n} = e^{(i(\bar{k}+\bar{K}_n)\cdot\bar{r})}$.

The columns of the projection matrix are orthonormalized to each other, forming the matrix $(^{j\bar{k}}t_{dd})_{mn}$. These columns are the coefficients of the plane waves $\psi_{\bar{k}_m}$, forming linear combinations that are symmetry adapted to the dth row of the irreducible representation ${}^{j\bar{k}}\mathbf{\Gamma}_A$ of $G_{\bar{k}}$, the little group of $\bar{k}$ of the second kind. This gives immediately the symmetry adaptation for the whole space group.

4.3 Description of the program

The matrices ${}^{j\bar{k}}\mathbf{t}_{dd} = (^{j\bar{k}}t_{dd})_{mn}$ form the output of the program. They give the coefficients for the plane waves which form linear combinations that are adapted to the symmetry of the space group. This output is given for all wave vectors of the input and all irreducible representations ${}^{j\bar{k}}\Gamma$. The functions $\psi_{\bar{k}_n} = e^{(i(\bar{k}+\bar{K}_n)\cdot\bar{r})}$ are written as $K(n)$ in the output. A list of correspondence between $\bar{K}_n$ and $K(n)$ is given in the output under the heading ‘The following lattice vectors form a stable basis’.

The text in the output is explaining the details, it is not necessary to read through chapter 4.3 to understand the form of the output.

Here follows a detailed description of the functioning of the program and comments to possible input or program errors. Since whole sections of program SYMPWS are identical to sections of program SYMPRJS, we refer for those sections to the description in chapter 3.3.

The program Sympws can be divided into 9 sections.

Section 1. Input (see chapter 4.1 for the sequence of input).

1.1 Read input 1 (chapter 4.1, input 1).

1.2 Read input 2 (chapter 4.1, input 2).

1.3 Print input 2.

1.4 Calculate matrix **B** (chapter 3.3, section 1.4).

1.5 Print matrix **B.**

1.6 Read input 3 (chapter 4.1, input 3), the index **pgnr** of the point group.

1.7 Read **order** (chapter 3.3, section 1.11)

1.8 Read the elements of the point group (chapter 3.3, section 1.12).

1.9 For nonsymmorphic space groups (**steer(20)** = ‘false’), read the nonprimitive lattice translations (chapter 3.3, section 1.13).

1.10 If the point group is a subgroup of $^2D_{6h}$ (16 ≤ **pgnr** ≤ 31), go to section 1.12.

1.11 See chapter 3.3, section 1.15, go to section 1.13.

1.12 See chapter 3.3, section 1.16.

1.13 See chapter 3.3, section 1.17.

1.14 Print the element numbers **gel(1:order)**. This output follows immediately the printing under 1.11 or 1.12.

1.15 See chapter 3.3, section 1.19.

1.16 Transform the rotation matrices to the reciprocal lattice coordinates.

1.17 Read input 5 (chapter 4.1), transform if necessary the coordinates of $\bar{k}$ to the reciprocal lattice coordinates. Print the vectors $\bar{K}_n$ of the input. For **last** = 1, control goes to section 8.

Section 2. Formation of the point group of the wave vector $P_{\bar{k}}$.

2.1 Test $\bar{k}$ with respect to the Brillouin zone (see chapter 3.3 section 3.1).

2.2 See chapter 3.3, section 2.1-2.3. A difference with chapter 3.3, section 2 is that the transformations are made in reciprocal lattice coordinates instead of Cartesian coordinates.

Section 3. Formation of a stable basis.

The reciprocal lattice vectors $\bar{K}_n$ of the input are transformed by the operations of the point group $P_{\bar{k}}$ into each other or in new lattice vectors $\bar{K}_m$. The set is extended to be stable under $P_{\bar{k}}$. **kmat(I,J)** gives the index of the reciprocal lattice vector, to which the Jth reciprocal lattice vector is transformed by the Ith operator of $P_{\bar{k}}$. The stable basis consists of **nrec** vectors $\bar{K}_n$.They are printed under the heading “The following **nrec** lattice vectors form a stable basis”. If **steer(17)** = ‘true’ then also **kmat(I,J)** will be printed.

For symmorphic space group or $\bar{k}$ within Bz, go to section 6.

Section 4. Tests for the symmorphic group.

See chapter 3.3, section 3.2.

Section 5. Formation of the factor group $G_{\bar{k}}/T_{\bar{k}}$.

See chapter 3.3, section 4.

Section 6. Calculation of the diagonal elements of the irreducible representations.

6.1 See chapter 3.3, section 6.1.

6.2 See chapter 3.3, section 6.2.

6.3 The number of projection matrices is equal to the sum of the dimensions of the (allowable) irreducible representations. This number is stored into **nup**.

Section 7. The formation of the projection matrices.

The projection matrices are formed according to the formula in chapter 4.2. All necessary information is now available.

7.1 Initiate the values for **J** (j) and **JD** (d).

Read the diagonal elements ${}^{j\bar{k}}\Gamma_{dd}$ if **steer(18)** = ‘true’, these elements will also be printed. The dimension of the projection matrix, stored in **jdpk(I,K)** will be **nrec**, which is the dimension of $\Gamma((P_i \mid \bar{u}_i))_{mn}$ in chapter 4.2. This is the summation for the group elements. The factors to be summed are constructed a little differently, depending on if (1) $G_{\bar{k}}$ is symmorphic, (2) $G_{\bar{k}}$ is nonsymmorphic but $\bar{k}$ within the Bz, (3) Nonsymmorphic $G_{\bar{k}}$, $\bar{k}$ on the Bz-boundary.

7.2 Test if the resulting matrix is idempotent and hermitian. If not, print a message.

7.3 Orthonormalisation of the columns of the projection matrix, see chapter 3.3, section 8.5.

7.4 Write the produced **t**-matrix, go to section 7.1.

Section 8. End of the program.

Print “SYMPWS has finished executing” and stop.

4.4 Program proj_sympws.m

Program proj_sympws.m reads the binary output file from SYMPWS.m and orders the columns for each $\bar{k}$-vector into separate projection matrices.

In proj_sympws.m, the following statements have to be adapted to your naming the input/output files:

in_proj = fopen(‘xx’, ‘r’); where xx is your name for the SYMPWS.m binary output file.
out_proj = fopen(‘yy’, ‘a’), where yy is your name for the proj_sympws.m binary output file.

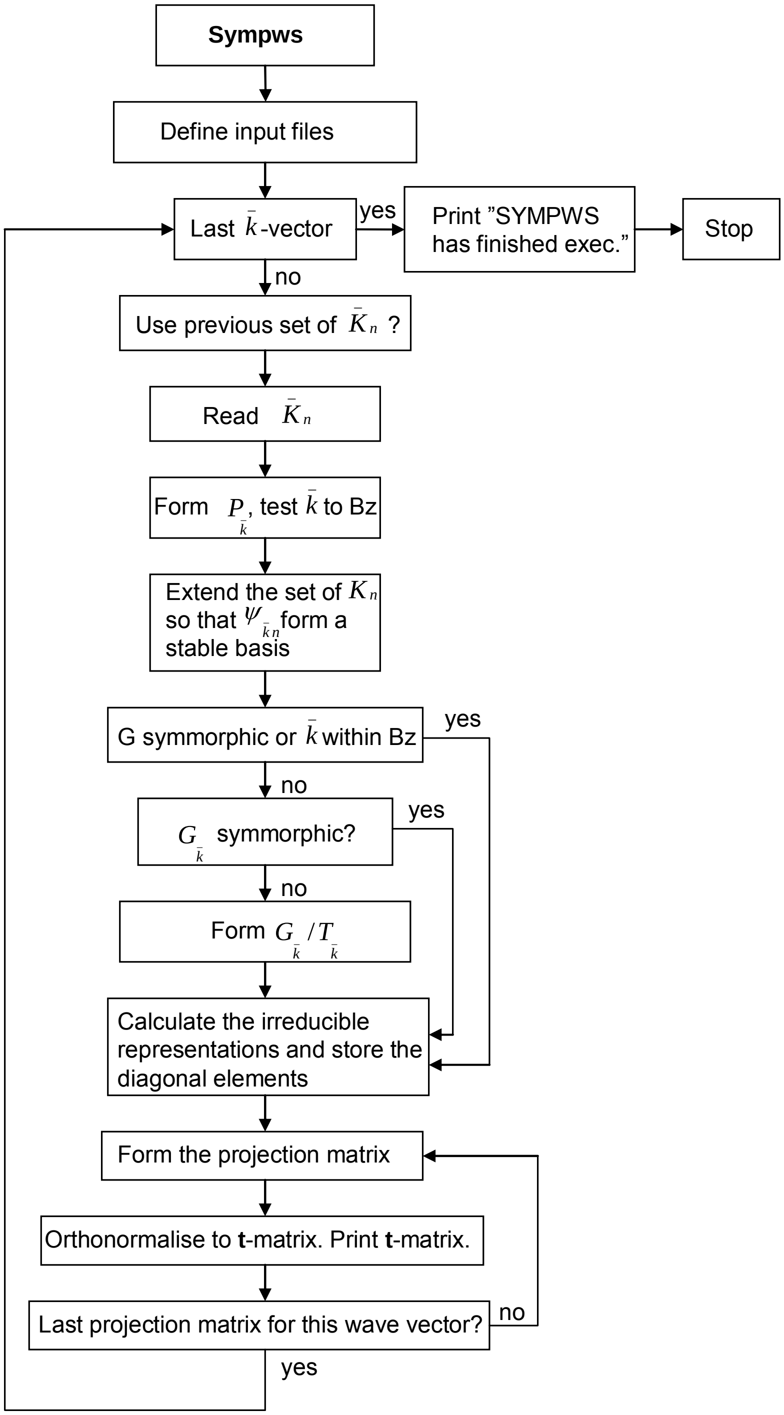
Sympws
Define input files
Last $\bar{k}$-vector
yes
Print "SYMPWS has finished exec."
Stop
no
Use previous set of $\bar{K}_n$?
Read $\bar{K}_n$
Form $P_{\bar{k}}$, test $\bar{k}$ to Bz
Extend the set of K_n so that $\psi_{\bar{k}n}$ form a stable basis
G symmorphic or $\bar{k}$ within Bz
yes
no
$G_{\bar{k}}$ symmorphic?
yes
no
Form $G_{\bar{k}}/T_{\bar{k}}$
Calculate the irreducible representations and store the diagonal elements
Form the projection matrix
Orthonormalise to **t**-matrix. Print **t**-matrix.
Last projection matrix for this wave vector?
no
yes

Figure 6: Flowing scheme of program SYMPWS

5. References

[1] The 2006 edition of Program Irrep. Submitted as QCPE program, replacing the earlier edition QCPE no. 163, 1970.

[2] O. V. Kovalev, Irreducible Representations of Space Groups: Irreducible Representations, Induced Representations and Corepresentations, Gordon and Breach, 1993.

[3] International tables of X-ray Crystallography, Vol. 1, Kynoch Press, Birmingham, England 1965.

[4] E. Blokker, Symmetry Projection of Crystal Wave Functions by Means of a Computer, Journal of Computational Physics, **12**, 471-490 (1973).

[5] P-O Jansson, Symmetry Adaptation of Crystal Spin-orbitals, Physica, **114A**, 482-484 (1982)