Tutorial to the program KPLOT

After starting the program KPLOT, a header is written on the screen indicating the release, followed by a ">" as a prompt. Now commands can be entered. All commands consist of 1 to 4 letters. Upper/lower case is ignored, but in the examples they are always given as capital letters. These commands may be followed by parameters. The following example illustrates the procedure:

First one enters a title:

```
>T 'NaCl, a simple example'
```

The command itself consists of only one letter 'T'. The text following 'T' is the actual title, which is treated by the program as a string. As a rule, every string should be enclosed in quotes. This requirement can only be waived, if the string does not contain blanks or special characters.

Next, a unit cell is defined:

```
>Z 5.6
```

This abbreviated input is sufficient, because NaCl is cubic. Using the defaults provided for the command 'Z', KPLOT completes the missing entries as if the entry would have been `Z 5.6 5.6 5.6 90 90 90`. Of course, one can always enter all six numbers explicitly

Next, we add the symmetries of the NaCl-structure. NaCl consists of two cubic face centered lattices of Na and Cl stacked within each other. For this simple example this information is already sufficient. First we need to provide the identity (symmetry). Since this symmetry is already prepared on the first place of the symmetry list, we just enter

```
>SE 1
```

Equivalently, we could have entered

```
>SY 'x,y,z'
```

of course. The rest of the symmetries are generated by the command

```
>GTY F
```

which introduces the F-centering.

If you want to see the symmetries now present in the symmetry list, enter

```
>OSY;
```

The symmetries will be shown as symbols.

Next the parameters of the atoms (coordinates, name, extension, etc.) are supposed to be entered. It is strongly advisable to have two dummy atoms at the beginning of the parameter list. In fact, at many places and defaults in the program it is assumed that these two atoms are present. Since only explicitly specified portions of the parameterlist will be plotted, these dummy atoms do not interfere with the proper drawing of structures. These two dummy atoms are prepared already and may be introduced into the parameter list by simply typing

```
>AE 2
```

The rest of the atoms have to be entered explicitly:

```
>ATOM  Na  1  0    0     0
>ATOM  Cl  1  0.5  0.5   0.5    0.4
```

Note that not all parameters have to be supplied explicitly. KPLOT sets default values at many places when input is omitted. E.g., in our case, the radius of Na 1 is automatically set to 0.3, while the radius for Cl is explicitly set to 0.4. The atom parameters can be shown on the screen by typing

```
>OA ;
```

Entering the semicolon prevents prompting for a range. If only "OA" is entered, KPLOT prompts "from,to?" expecting then two integer numbers indicating from what position to what position in the parameter list the parameters are to be displayed. In our case "1,4" would show the complete list. Note that the first integer defaults to "3" , since usually the dummy atoms are only of minor interest.

At this stage, the basic input is complete. Now the atoms which are to be plotted have to be compiled in a list. E. g., if we want to see one unit cell, we use the command

```
>ATB 1 3 4 0.5 0.5 0.5
```

The command "ATB" ("add triclinic box") results in a search for all atoms which are in a box with the center at atom "1" having walls with distances of 0.5 lattice units in positive and in negative directions from the center. This is exactly the content of the unit cell, because the (dummy) atom at no. 1 in the parameter list is in the centre of the unit cell. All the atoms found are automatically stored in a dedicated list (the so-called code list).

In order to achieve a good visual representation of the structure, it has to be rotated. This is done using the commands

```
>DK  2 18
>DK  1 18
```

Note that the atoms compiled for plotting in the code list are not plotted automatically. Explicit commands have to be entered defining which atoms are to be plotted and how they are to be connected.

```
>PK  3  4
>VBR  3  3  4  4  3
```

Still no plot is produced, since KPLOT only stores the above plot commands in a plot command list, but does not execute them. As a final preparatory step, a scaling factor has to be calculated to make the picture fit on the screen:

```
>BF
```

Now the command

```
>EPU
```

should generate the following plot:

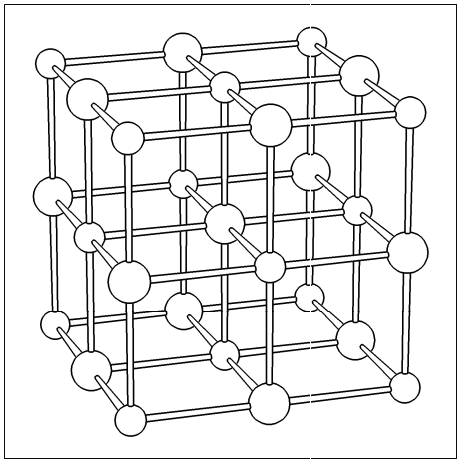

Fig. 1: NaCl

If one wants to generate a stereoscopic pair of pictures, two copies of the structure have to be plotted side by side, but each rotated (by a slightly different angle) somewhat around the y axis. For each plot only half of the screen is available now; thus the field has to bisected:

```
>HF
```

To produce a steroscopic plot, only two commands have to be entered: "calculate scaling factor" ("best fit") and "stereo plot":

```
>BF ; STPU
```

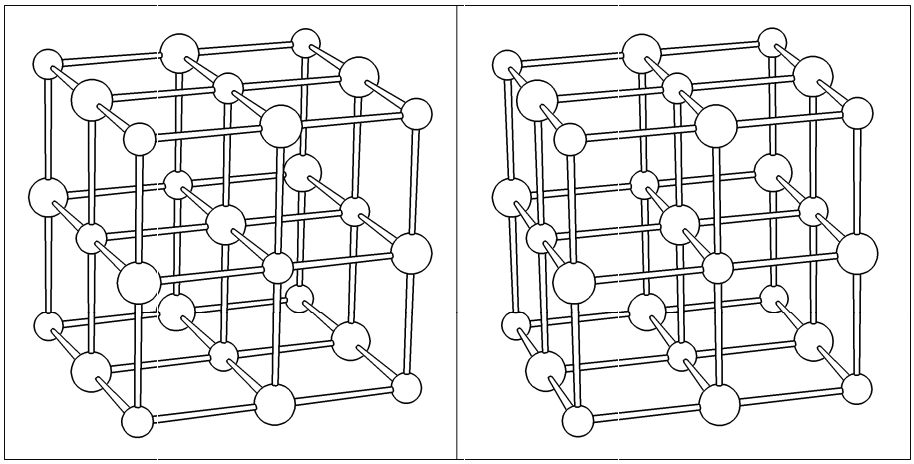

Fig. 2: Stereoscopic plot of NaCl

One notices that more than one command can be given within one command line. The individual commands have to be separated by a semicolon. This delimiter may always be entered, but is not necessary at the end of the line .

After start-up, KPLOT operates in dialogue mode. As a consequence, a "¿" appears when a command is expected. If a command is entered, for which parameters are expected, but no (or not sufficient, including defaults) parameters are provided, a prompt appears listing the expected input variable. This is quite useful; e.g. if one does not know what detailed input is expected for a certain command, one just enters the command alone. Example:

```
> OA
from,to? 3,4
```

Very often one wants to use the default values and suppress prompting. This can achieved easily by appending a slash, a semicolon or a comma. Example:

```
> OA,
```

Usually it is more convenient not to type in all commands of the basic input describing a structure (such as atomic parameters, cell parameters etc.). Instead, one can prepare a file using a text editor containing the basic data. If such a file is read in, prompting makes no sense, of course. Therefore, one can switch off the prompting by using 'NDLG'. Thus, a file which is going to be read by KPLOT, should look like:

```
NDLG
.....  (KPLOT commands)
CLSE / DLG
```

Such a file NACL.DAT containing KPLOT commands defining the basic input can be read in by

```
>GET  NACL.DAT
```

Analogously a file containing the current information available to KPLOT (cell parameters, parameter list, code list, symmetry list, etc.) can be created by

```
>PUTC  NACLNEW.DAT
```

The usage of 'GET' and 'PUTC' is more subtile than can be explained here. Note that the file NACL.DAT m u s t exist when using 'GET', while the file NACLNEW.DAT should not already exist when using 'PUTC', since it would be overwritten automatically, without any warning.

In the next example, we get to know a very powerful command. First we prepare an input file.

```
NDLG
T 'Cuban - example of a molecular structure'
Z 5.34 * * 72.26 72.26 72.26
HMS 'R3-R'  ! A specification HMS 'R3-' will not suffice, since  there
            ! exist more than one setting
AE 2
ATOM C 1 -.18711 .19519 .10706
ATOM C 2 .11546 .11546 .11546
ATOM H 1 -.32460 .34680 .18480 .2
ATOM H 2 .21 .21 .21 .2
CLSE;DLG
```

This file (assumed to have the name CUBANE.KPL) can then be read in by

```
>GET CUBANE.KPL
```

In order to obtain a plot, a very simple but powerful command is available:

```
>FM * H
```

FM means: find molecule. Here, KPLOT untertakes the following steps:

- All atoms belonging to the molecule(s) are searched for, according to a distance criterion. The specification of "H" in the example above is an option which causes hydrogen atoms to be also included in the search. The search for atoms is performed using an internal table where the most common interatomic distances are stored.
- Plot commands are generated for plotting atoms as polygons with 30 vertices and bonds as lines.
- An optimisation of the view is performed in order to minimize overlap of atoms in the plot.
- The scaling factor is determined.

Thus, the user only needs to enter

```
> EPU
```

in order to get the plot

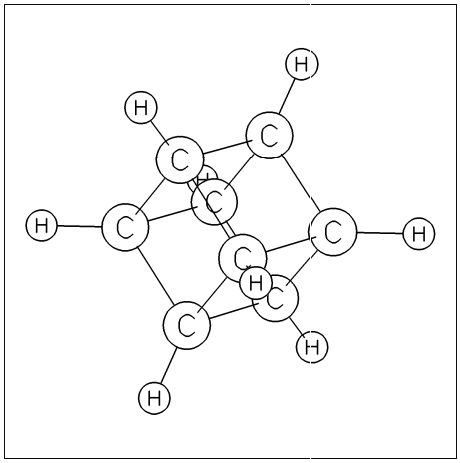

Fig. 3 Cubane (I)

In our case this optimisation has the consequence that the model looks rather oblique. This should be improved now:

```
>DKSV  3  355505
>BF
>EPU
```

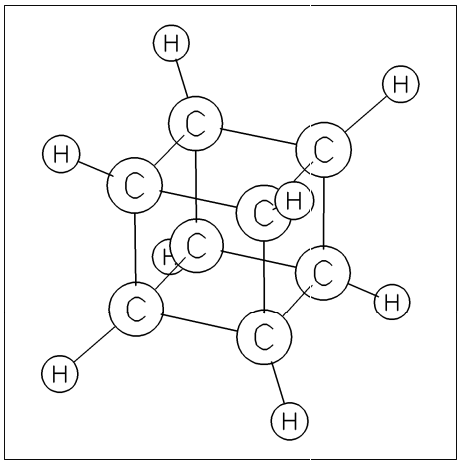

Fig. 4: Cubane (II)

The two numbers designate uniquely two atoms, and hence a vector from the first to the second atom. Using 'DKSV' the coordinate system is rotated in such a way that this vector becomes vertical (on the screen).

The drawing is still very rough, and we want to make it nicer. Therefore we remove the plot commands that had been generated automatically:

```
>LPK
```

Now we enter new ones:

```
>PK  3  6
>VBR  3  4  3  6  1.7
>EPU
```

A new plot is generated, where the bonds are now represented as sticks.

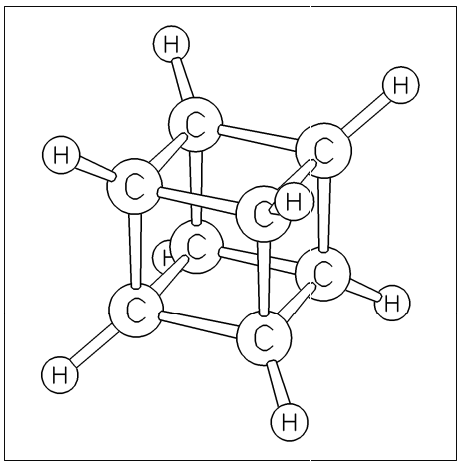

Fig. 5: Cubane (III)

If colors can be displayed, one may represent balls and sticks in different colors. To achieve this, control blocks have to be defined that are used to store information about colors, filling patterns etc.

```
>ATF  1    1  3   1  2 !  grey for C atoms
>ATF  2    1  3   2  1 !  red for H atoms
>FRBS  1  C            !  All C atoms point to block 1
>FRBS  2  H            !  All H atoms point to block 2
>ATF 11  1 3 3 1       !  control block for the bonds
>OPK                   !  show plot commands
>APK 2 ******* 11      !  set pointer to block 11
>EPU
```

Now the balls and sticks should be represented according to the comments above.

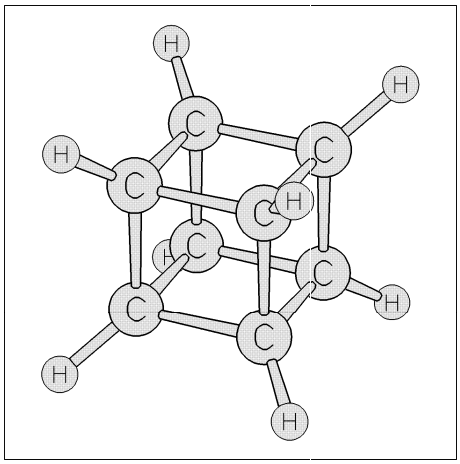

Fig. 6: Cubane (IV)

You have probably noticed that there are much more (16) atoms on the screen than entered in the parameter list (4). As mentioned above atoms in the parameter list are not plotted automatically. Atoms which are supposed to be plotted have to be selected and stored in a different list. The selection is done applying symmetries and translations along the lattice axes. In KPLOT a coding is used which may be treated as a rule how to obtain the coordinates of the atoms to be plotted from the atoms in the paramete r list, symmetries and translations. It is the so called atom designator code, most times shortly called code.

A code is a number with 6-9 digits with the following meaning:

	NR	TA	TB	TC	SYMNR
digit	9 8 7 6	5	4	3	2 1

NR is the position of an atom in the parameter list.

SYMNR is the position of a symmetry in the table of symmetries. Due to the fact that the symmetry list can store up to 192 symmetries, digit 2 may range from $0, \ldots, 9, \mathrm{A}, \ldots, \mathrm{J}\ (= 19)$

TA, TB and **TC** are translations increased by 5 in the directions of the base vectors of the unit cell, e.g. "5" means no translation, "6" a translation by +1 etc.

In order to obtain the coordinates of an atom given by a code, divide the code into the numbers NR, TA, TB, TC, and SYMNR, as shown above. Then take the coordinates of the atom NR, apply the symmetry SYMNR, and add to x TA-5, to y TB-5, and to z TC-5.

Example:

The code 345706 denotes an atom whose coordinates are found on the 3rd position in the parameter list. To these parameters the symmetry is applied, which is found on the 6th position in the list of symmetries. The result is finally shifted by -a and 2c. E.g., assume the coodinates $(.1, .2, .3)$ (in tricilinic coordinates) on the 3rd position of the parameter list and the symmetry $-x, 1/2 + y, 1/2 - z$ on the 6th position in the symmetry list, then the code 345706 gives an atom with the coordinates $(-1.1, 0.7, 2.2)$. Verify this!

Quite generally, atoms which are to be drawn have to be stored as codes in the codelist. This may be done directly using the commands 'ACI' or 'ACIM'; however, this is done only occasionally. Instead, leave the work to determine the codes of these atoms as much as possible to the program! Most times one wants to specify certain regions of the crystal that include the atoms that are to be plotted. Currently, three possibilities have been provided to specify such regions: spheres, boxes oriented with respect to the lattice, and boxes relative to a orthonormal system, the so-called free coordinate system.

Here, one of the most frequently used commands is 'ATB'. ATB means: add triclinic box. We have already used it with the NaCl example above. The general syntax is:

`ATB` $c_{or}, n_1, n_2, d_x, d_y, d_z$

The name triclinic should remind one that the orientation and dimension of the box refers to the unit cell, which is triclinic in general. In this sense, orthogonal cells are only special cases. c_{or} is a code defining the midpoint of the box, the origin. Walk from this point in the direction +a and -a by ad_x to meet the wall of the box parallel to b and c (a,b,c are the unit cell constants). Do the same with d_y and d_z to define the other walls. As in the example above c_{or} pointed to an atom with coordinates $(0.5, 0.5, 0.5)$ and the values $d_x = d_y = d_z = 0.5$ had been entered, all points could be reached that have triclinic coordinates between 0 and 1 in x, y, and z respectively. This region is searched for atoms having numbers between n_1 and n_2 (target interval), inclusively. n_1 and n_2 point to the parameter list. In some sense the complete codes of all atoms are searched having numbers between n_1 and n_2 and lying in the specified box.

In the example above, a "1" was specified for c_{or}. One would have expected "155501" because a code is required as input. But codes ending with "55501" (effectively referring to the original atom as stored in the parameter list) appear very often. Thus the "55501" may be omitted (as default KPLOT adds "55501" automatically). This must not be confused with the specifications for n_1 and n_2: no codes are allowed here.

Applying the 'ATB' command to our last example, one gets:

```
>CC
>ATB 1 3 6 .5 .5 .5
>BF
>EPU
```

At first glance the result is somewhat surprising: instead of the complete molecule only several scattered fragments are visible. If the outlines of the unit cell are added, it becomes

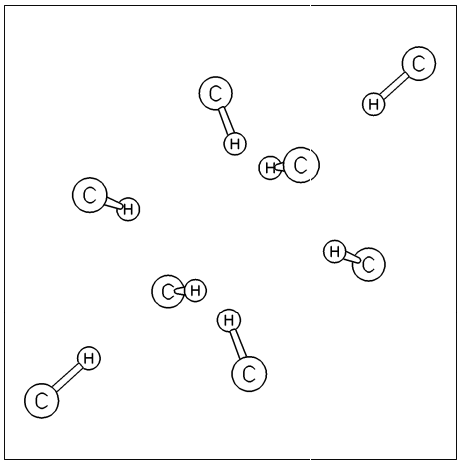

Fig. 7: Cubane (V)

obvious what has happened.

```
>GNZL ;
>BF
>EPU
```

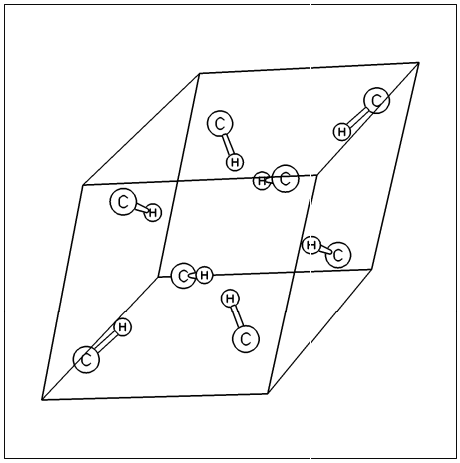

Fig. 8: Cubane (VI)

The description of the molecule i.e. the coordinates of the atoms do not agree with the unit cell. In each corner there is a fragment of the molecule.

We now want to complete these fragments to full molecules. We use a command which searches the surrounding of the atoms (in the code list) found so far and adds the new ones to the code list. Note that these new ones will also be taken into account during the search. This process stops if no new atoms are found.

```
>AUW 3 4 3 4 1.7  ! Complete carbon surroundings
```

The command 'AUW' must be handled with care, because the code list may overflow.

It remains to add the hydrogen atoms:

```
>AU 3 4 5 6 1.3
>DK 1 15  ! To minimize overlap
>BF
>STPU
```

As a result, the packing of the molecules of cubane is produced. But only the stereoscopic view allows to resolve the structure easily.

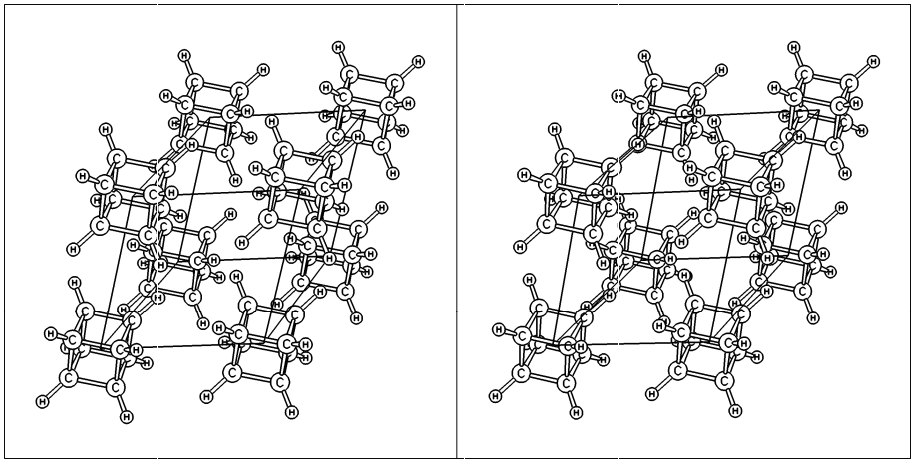

Fig. 9: Cubane (VII)

Facilities for (geometric) constructions

Sometimes the problem arises that positions of atoms have to be calculated from known geometric properties, but with respect to a (usually) triclinic cell. Let us consider the following example: The solution of the crystal structure of tertiary tin sulfide (from film data) gives the positions of the tin and the sulfur atoms, but no positions for the carbon atoms could be extracted from the Fourier map. Here are the data:

```
>NDLG
>T 'Alpha Di-tertiary butyl tin sulfide'
>Z 6.323 16.50 12.117 90 114.94
>RG 1402
>AE 2
>ATOM Sn 1 .05079 .06656 -.08874 .4 4
>ATOM S 1 -.2803 .0425 -.0392 .4 6
>CLSE;DLG
```

As one can see, this is a monoclinic system. Applying 'FM', a four-atoms ring is found consisting of tin and sulfur (fig. 10):

Unfortunately this four-ring lies oblique in an oblique cell. It is expected that the tertiary carbon atoms bonded to tin enclose approximately an angle of 109.5° C-Sn-C. The fastest strategy to locate the tertiary C-atoms is as follows: Turn on the cross-hair cursor by 'M' ("mouse on"), and click the (upper) tin atom and then both S atoms. The codes of the

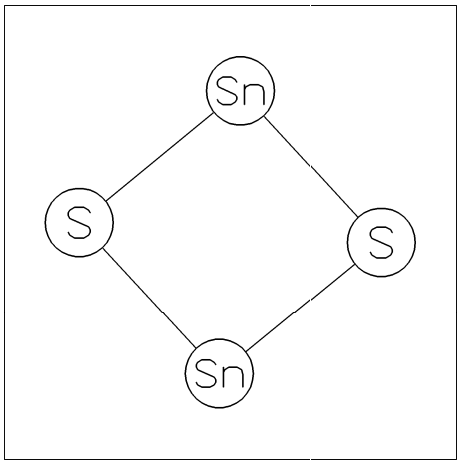

Fig. 10: t-butyl tin sulfide (I)

atoms clicked are stored in a special list, the so called group list of codes or mouse list. This list can be displayed by 'OGC' or 'OGCF' in short or detailed form, respectively. The command used to calculate the coordinates for the C-atoms is 'GZTU' ("generate two points in tetrahedral surrounding"). In our case:

```
>GZTU 0 0 C 3 0.3 2.2 * 0
```

Note that zeroes have been entered at three positions instead of codes to specify atoms. This technique may be used in general (except if stated otherwise): If a code is expected by the program and a zero is found, the code is taken from the mouse list. The result (after a new 'FM' and some rotation around the x axis) looks as follows:

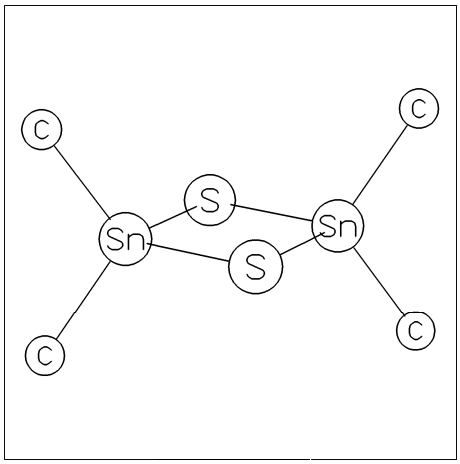

Fig. 11: t-butyl tin sulfide (II)

Now the primary C atoms have to be found. We click the upper right C atom and then the Sn atom bonded to the C atom. Due to the fact that we do not know anything about the orientation of the group, we "leave" it to the program to make a decision. We enter:

```
>GDTU 0 0 C 6 0.3 1.54  ! Generate two atoms in tetrah. surr.
```

This procedure is repeated using the C atom in the lower right corner. Again 'FM' is entered, producing the plot:

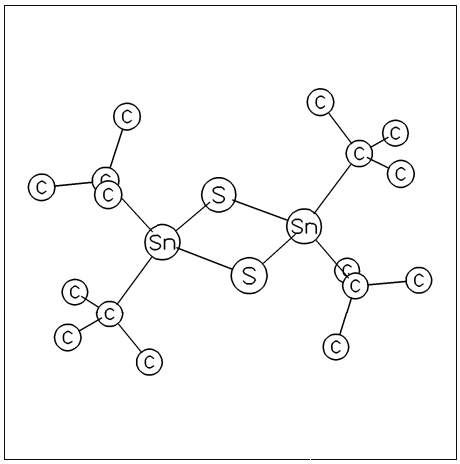

Fig. 12: t-butyl tin sulfide (III)

This model may be used for a refinement cycle in a structure solution program.

There are not always suitable commands for calculating positions in such an easy way. In those cases one has to use a set of more elementary commands, the so-called micro-commands. These commands manipulate a single point only, stored on the so-called point register. Using the example above, an alternative way how to obtain the locations of the C-atoms will be discussed.

First, we use the four-ring found to define an orthonormal coordinate system (the free coordinate system):

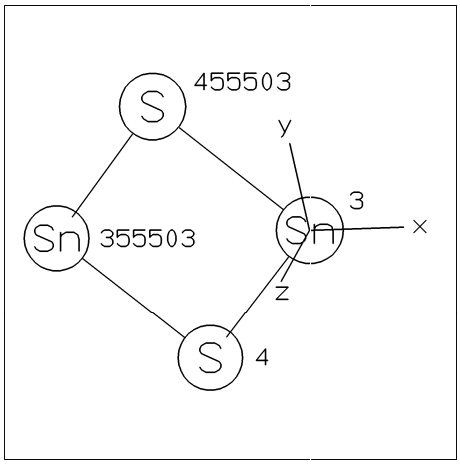

Fig. 13: t butyl tin sulfide (IV)

This is done by

```
>K  3  355503 3  455503 4
```

With respect to this coordinate system, we define the point on the point register having the desired distance but lying on the x-axis. Then this point is rotated around the y-axis by half the angle desired, stored as C-atom in the parameter list, rotated in the opposite direction by the full angle. This produces a new C atom having the calculated coordinates. The sequence looks like:

```
>P 2.2 0 0   ! Definition of a point on the x-axis
>DP 2 54.75  ! Rotation by 109.5/2
>AA C 1      ! Add as atom
>DP 2 -109.5 ! Rotate to the other direction
>AA C 2      ! and add as atom
```

The construction of the primary C-atoms is performed similarly. First, the coordinate system is defined in a suitable way:

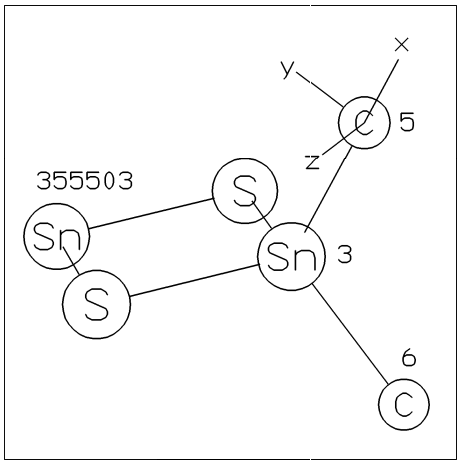

Fig. 14: t butyl tin sulfide (V)

```
>K  5  3 5  3 355503
```

Now the positions of the three C-atoms are calculated:

```
>P  -1.54 0 0
>DP  3 109.5; AA C 3
>DP 1 120; AA C 4
>DP 1 120; AA C 5
```

The atoms bonded to atom no. 6 may be constructed in the same way, but there exists a better method, because the second group is the mirror image of the first one. Note that the mirror plane is no crystallographic symmetry. Therefore we have to define a mirror plane ourselves. For this we define a coordinate system like we did in Fig. 13. Before we apply this mirror symmetry to the group, we remove atom no. 6.

```
>EPL 6
>K  3  355503 3  455503 4
>SPG  5 8  3
```

Entering 'FM' again, the complete molecule is found.

Using the knowledge gathered so far we are now going to attempt a more difficult task: We want to build the structure of the fullerene C_{60}:

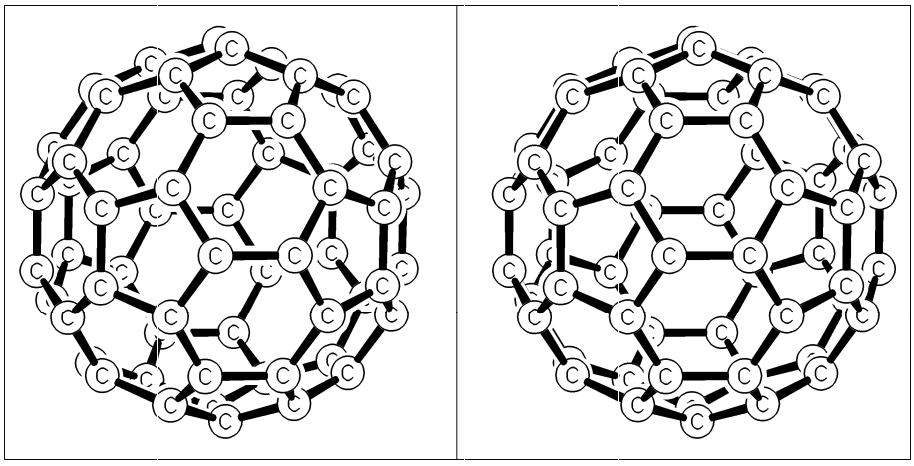

Fig. 15: C_{60}

We proceed as follows: First, we define an arbitrary cubic cell with a=10.

```
>T 'Construction of C60'
>Z 10
>AE 2
>SE 1
```

Now we generate a pentagon:

```
>P 1 0 0
>GP 5 3 C 1 ! Generate 5 points
```

The edge lengths of this pentagon are 1.17557; this is not the value desired. Thus, after setting the default value for group commands to 'R' (replace) the pentagon is shrunk to one with edge length equal 1.

```
>GOPT R    ! Use replace option
>L 3 4     ! Deposits 1.17557 as default value
>FRG 3 7 2 ! Shrink pentagon by this factor
```

Each pentagon is surrounded by 5 hexagons. One point of the hexagons is determined using 'PAW', because we do know the distance and angles:

```
>PAW 3 4 7 1 120 120 ! Generate one point of the hexagons
```

This point is now on the point register. We rotate the point around the z-axis generating 5 new points in the parameter list.

```
>GP 5 3 C 6 ! Generate five new points
```

The structure built so far looks as follows (fig. 16):

The atoms are labeled by their codes. This can be achieved using 'PSCD'. This may help to find one's way through the structure. Now we need the missing points of the hexagons. For this we use the command 'PZAA' and the knowledge that the length of a diagonal in a hexagon is twice the length of an edge.

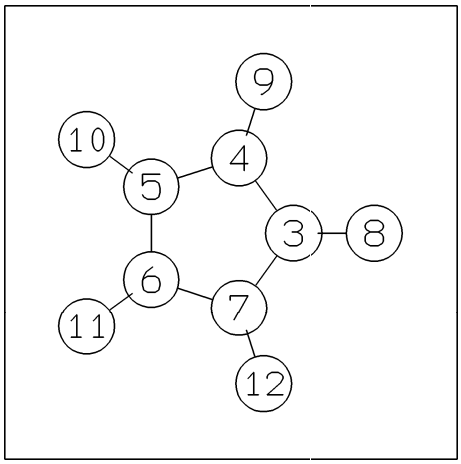

Fig. 16: Pentagon with neighbouring atoms

```
>PZAA 8 4 3  1 2  ! Point by two distances
>GP 5 3 C 11      ! Rotate as usual
>PZAA 9 3 4  1 2  ! The other point
>GP 5 3 C 16      ! times five
```

The structure built so far looks as follows:

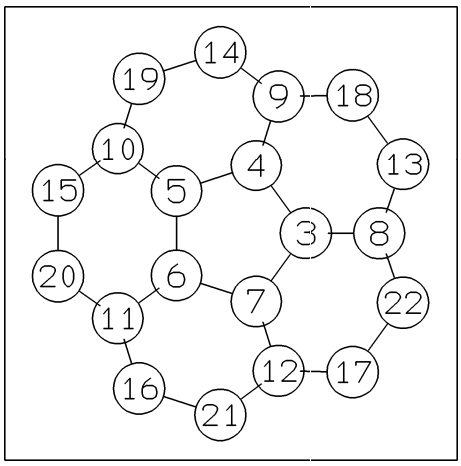

Fig. 17: Pentagons with next hexagons

Instead of continuing in this way, we now take the opportunity to learn an new technique. It is a powerful tool that is used frequently. Since we already have a pentagon (atoms 3-8), it would be clever to copy it and place it into the corner 8/13/22. This is done by 'KTG'. KTG allows, so to say, to pick up a group (as if with a fork-lift), transport it, and deposit it on a different location. In our case, this "fork-lift" is the free coordinate system.

```
>K 3 3 4 3 7      ! Starting position ...
>SK               ! ... save.
>K 8 8 22 8 13    ! New analogue position
>KTG 5 6 A        ! Atoms 5 and 6 are transported and added
```

Now we would like to repeat this operation for the other symmetry equivalent atoms. The simplest way to achieve this is by a fivefold rotation. For this we need to return to the original coordinate system.

```
>KS
>PL 23
>GP 5 3 C 21 * 2
>PL 24
>GP 5 3 C 25 * 2
```

The result looks like:

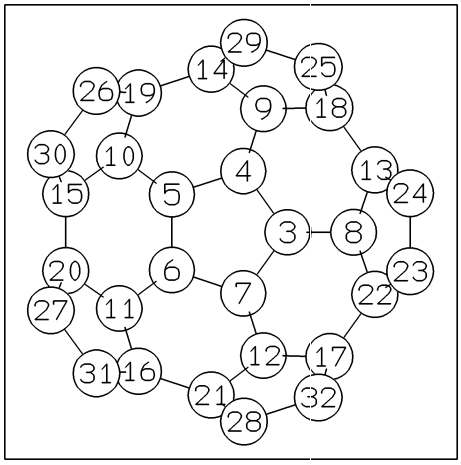

Fig. 18: Concatenated pentagons

The rest of the bucky ball is found very easily: Since there is a center of symmetry, the atoms found so far need only be mapped with respect to this center. But where exactly is this center located? Note that on the one hand it must lie on the c-axis, while on the other hand it must lie on the perpendicular bisector of e.g. the atoms 3 and 8. It is this vector we now have to construct.

```
>SP 3,,8            ! Center of mass between atoms 3 and 8
>AA SP              ! store atom (no. 33) and
>K 33 3 8 2 255601  ! define coordinate system.
```

The y-axis is the perpendicular bisector we searched. But where is the intersection point with the c-axis? To find the intersection, we construct an auxiliary point x:

```
>P 0 1 0
>AA X
```

and calculate the intersection point of two vectors:

```
>SVV 33 34 2 255602
```

The last two auxiliary point are no longer needed, so we can remove them and store the center of symmetry in the parameter list.

```
>EPL 33 34  ! Remove auxiliary points
>AA ZNRM    ! and add center (no. 33)
```

All atoms are now mapped using this point as a center of inversion:

```
>ZG 3 32 33 A  ! Generate centered group and add
```

The bucky ball is now complete, but it has not yet the edge length desired. It must be stretched:

```
>FDG 3 63 33 1.39
```

Now the (auxiliary) point no. 33 may be removed from the list:

```
>EPL 33
```

All the atoms belonging to the fullerene C_{60} are now in the parameter list. It remains to generate the plot commands. This is most easily done using 'FM'.

We have chosen an arbitrary cell in the example. If one wants to export the coordinates to a different cell, they have to be recalculated. In order to do so, the command 'ZTAK' is used. Via ZTAK a new cell is defined and the coordinates are transformed correspondingly. *Example:*

```
>ZTAK 27.8 27.8 27.8 90 90 90
```

Now the bucky ball is supposed to be brought to a position where the center of mass coincides with the origin (0,0,0) and the twofold axes coincide with the lattice axes. In order to achieve this, we need the center of mass again. In our case, the easiest way to generate the center of mass is to use 'SPC': The 60 atoms are loaded in the code list and the center of mass is directly calculated:

```
>CC
>ACI 3 62
>SPC ;
>AA SP 1  ! The point gets assigned no. 63
```

The twofold axes we search for cross the bonds, where two hexagons are concatenated. We select one (arbitrarily) and define the x-axis of the free coordinate system, e.g.

```
>K 63 4 9
```

Now we calculate the centers of mass (c.o.m.) between the designated atoms and define the coordinate system using the c.o.m.s:

```
>SP 4,,9
>AA SP 1
>SP 46,,51
>AA SP 2
>K 63 63 64 63 65
```

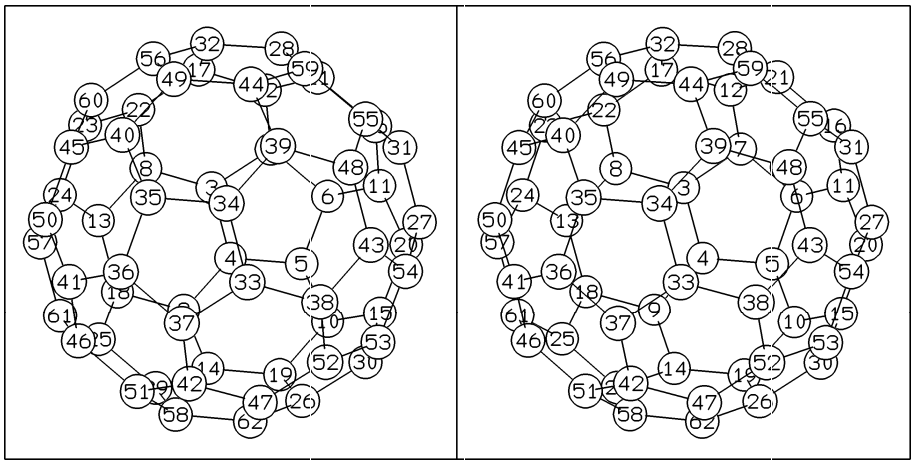

Fig. 19: Searching for twofold axes

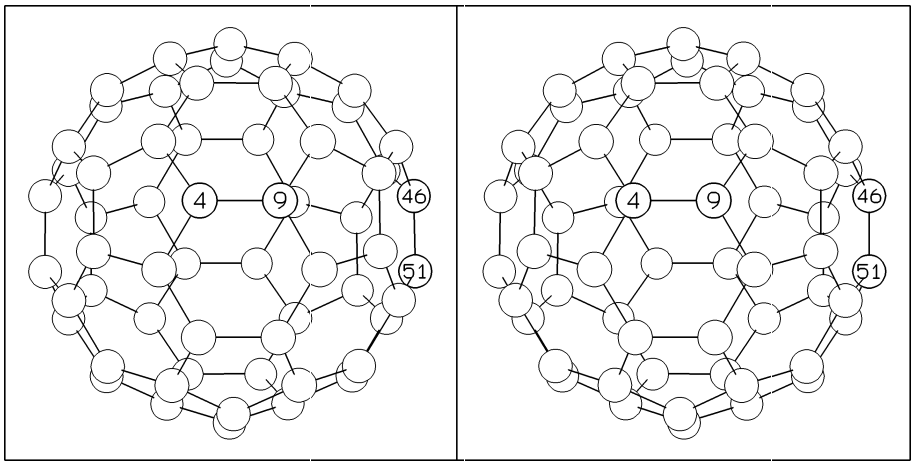

Fig. 20: Starting position for coordinate transformation

First, this configuration is saved for future reference. Then a new coordinate system can be defined:

```
>SK                        ! save coordinate system
>K 2 2 265501 2 256501     ! new coordinate system
>KTG 3 65                  ! transform also aux. points
```

If the coordinates of the auxiliary points are shown on the screen (OA 63 65), one sees that they are in the desired positions, and therefore the corresponding twofold axes point in the desired directions. Thus the auxiliary points may now be removed: `EPL 63 65` .

Indeed, the molecule possesses a high symmetry (the space group can be found using 'SFND' and the program RGS (Hundt, 1997)). We introduce the (correct) space group $Pm\bar{3}$ and reduce the parameters:

```
>HMS PM-3
>RPSY 3 62
```

Only 3 atoms remain from the 60 needed for the description without symmetry information. We can convince ourselves using 'FM' that the same picture is generated nevertheless.

Symmetries

Symmetries play an important role in crystal structures, but often it is impossible to determine all symmetries from the experimental data available. And even if a crystal structure has been solved, a problem may remain: has symmetry been overlooked? And if so, which space group has to be chosen and which transformations have to be performed to obtain the new description?

The problem of finding a space group is also of increasing interest for quite a different sort of structures - those occuring in computer simulations. The computing power availabe nowadays allows us to generate crystal structures in the computer. But these have to be classified and analyzed, of course, and here no experimental data are available such as reflections or the shape of the crystal.

KPLOT in combination with the program RGS offers a tool to answer those questions.

Let us first look at an easily understandable example: Which space group is to be assigned to the Rhenium trioxide type? Clearly the problem is scale independent, so we assume $a = b = c = 7.0$.

```
>Z 7
>SE 1
>AE 2
>ATOM Re 1 0 0 0
>ATOM O 1 .5 0 0
>ATOM O 1 0 .5 0
>ATOM O 1 0 0 .5
```

Note that all atoms in the cell are given, as if the space group were P1.

The first step is to find the symmetries. This is done by using the command 'SFND'. If default values are used for the tolerances, the following result is produced:

```
Tolg: 0.1000 Tols: 0.2500 Tolt: 0.2500
4-Axis [ 556 ]
4-Axis [ 565 ]
4-Axis [ 655 ]
3-Axis [ 644 ]
3-Axis [ 646 ]
3-Axis [ 664 ]
3-Axis [ 666 ]
2-Axis [ 546 ]
2-Axis [ 566 ]
2-Axis [ 645 ]
2-Axis [ 654 ]
2-Axis [ 656 ]
2-Axis [ 665 ]
A:  4 R: 556 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  4 R: 565 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  4 R: 655 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  3 R: 644 P:  0.0000  1.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  3 R: 646 P:  0.0000  1.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
```

```
A:  3 R: 664 P:  0.0000  0.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  3 R: 666 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 546 P:  0.0000  1.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 566 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 645 P:  0.0000  1.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 654 P:  0.0000  1.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 656 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 665 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
Z 1    0.00000   0.00000   0.00000 (0.000000)
```

First, KPLOT searches for "candidates" for symmetry directions using the unit cell supplied and a tolerance "Tolg", because the symmetries present must agree with the lattice. The direction specifications in square brackets are to be interpreted as follows: subtract 5 from each digit to obtain the true direction. The number 546 for example means: 0,-1,1. In the example above it is seen that there are four-, three-, and twofold axes. This should be expected when giving a cubic cell.

These candidates are then tested (using the tolerance "Tols" in Angstroms), i.e. we check whether they are really present in the structure. This is the case in the example chosen. In the output, each symmetry is given by the axis (A:), the direction (R:), a point on the symmetry element (P:), and a glide vector (G:). The numbers in brackets are the maximal deviations between atoms in Å after applying the symmetry and the corresponding equivalent atoms. This value is a qualitative criterion for the fit of the symmetry found only. Finally centres of symmetry (Tols) and translation symmetries (Tolt) are searched for. In the example above, a centre is found.

This result is passed to the program RGS. RGS generates a file called RGS.OUT readable by KPLOT containing the instructions how to transform the structure and which space group is to be selected. In our case, RGS.OUT looks like:

```
NDLG
MTRI;
TZUR   0.00000   0.00000   0.00000
TZ   1.0000  0.0000  0.0000  0.0000  1.0000  0.0000  0.0000  0.0000  1.0000
RG   221 (Following Int. Tables 3rd edition; HMS: PM-3M   )
RPSY * * 0.5 -1; NPZ * *
CLSE;DLG
```

Of course the execution of these commands reduces the amount of parameters:

```
 Atoms:
   No  Name          x         y         z         r    Clrpnt
    3  Re  1     0.000000  0.000000  0.000000   0.3000       0
    4  O   1     0.500000  0.000000  0.000000   0.3000       0
```

Question: What is the result, if (1) one O atom is replaced by an S atom, and (2) a second O atom replaced by Se? Answers: (1) P4/mmm and (2) Pmmm. Verify this statement!

The recipe for testing, whether a space group assumed for a structure is the correct one, is straightforeward under normal conditions:

```
SFND;;         (find symmetries, use defaults)
SYS RGS;       (start program RGS, in this form under DOS)
GET;           (program RGS reads the file FOR007.DAT)
ANAL;          (analyse input)
RGS;           (search space group)
RTS;           (if necessary result to file)
END;           (terminate RGS)
GET RGS.OUT;   (read file produced by RGS if written)
```

In version 8.4.0 and higher RGS has been integrated into KPLOT. So, the sequence above may be shortened to

```
SFND;;         (find symmetries, use defaults)
RGS;           (start subsystem RGS, do things automatically)
GET RGS.OUT;   (read file produced by RGS if written)
```

One has to make sure that only one type of atoms are on the same position if statistical occupation is present. If two or more atom types occupy the same position, one has to define an additional atom type (e.g. called "1"), that is placed on these positions.

The following structure was taken from the ICSD. Here it is rewritten as a KPLOT input:

```
NDLG
T 'COL= 15961 FORMEL:Y F3  Yttrium fluoride'
Z    5.64400
HMS 'P23     '
AE 2
ATOM (   3) 'Y   '     '1   '      0.500000  0.500000  0.000000
ATOM (   4) 'F   '     '1   '      0.500000  0.500000  0.500000
ATOM (   5) 'F   '     '2   '      0.250000  0.250000  0.250000
ATOM (   6) 'F   '     '3   '      0.750000  0.750000  0.750000
CLSE ; DLG
```

We note that space group P23 is used. If the check given above is performed, the space group $Pm\overline{3}m$ (no. 221) is found, however.

Certain problems commonly arise when structures are studied that are found in computer simulations. The reason is that the translational symmetry of the cell only reflects the periodic boundary conditions of the simulation and is usually triclinic, in most cases distorted, and occasionally contains additional translation symmetry (especially for large simulation cell unless an amorphous structure is present). In these cases the result of the symmetry analysis must be used in an intermediate step to idealize the cell.

The necessary sequence of steps is almost always the same, and can be executed by calling the built-in macro 'SSI'.

1. The structure is made triclinic if not already so.
2. It is checked for tranlational symmetry, and the cell is reduced accordingly, if such a symmetry is found. This step is repeated until the cell is primitive.

3. SFND is used to search for symmetries.

4. Using the symmetries found the cell is idealized.

5. Again the idealized structure is searched for symmetries, and type and number of symmetries are compared to the result of the last run. If more symmetry has been found, the procedure is continued at step 4. If nothing has changed, the task is finished. If symmetry has been lost, an error message is issued, and the search for symmetries is stopped.

Let us consider the example $CaCl_2$, where the following structure was the result of a computer simulation (given here in KPLOT notation):

```
Z   5.60068   5.30355   5.61138  90.09382  90.14352  89.92796
SE 1
AE 2
ATOM  Ca   2 0.7678569 0.5873027 0.5678615
ATOM  Ca  -1 0.2674217 0.5933311 0.0684757
ATOM  Cl   0 0.2693682 0.0929764 0.0684085
ATOM  Cl   0 0.2679240 0.5904401 0.5681831
ATOM  Cl   0 0.7683695 0.5905680 0.0685028
ATOM  Cl   0 0.7689828 0.0876594 0.5682762
```

If the command 'SSI' is given, one gets the following result (comments are given in brackets):

```
(Search for translation symmetry ...)
Tols: 0.2500 Tolt: 0.2500 Tolg: 0.1000 Ref: CA
T 1    0.50000   0.00000   0.50000 (0.032246) (... found)
Tols: 0.2500 Tolt: 0.2500 Tolg: 0.1000 Ref: CA (... try again.)
(Search for symmetries ...)
Tols: 0.2500 Tolt: 0.2500 Tolg: 0.1000 Ref: CA
4-Axis [ 556 ]
2-Axis [ 565 ]
2-Axis [ 655 ]
2-Axis [ 645 ]
2-Axis [ 665 ]
A:  4 R: 556 P:  0.3358 -0.1995  0.4097 G:  0.0000  0.0000  0.0000 ( 0.01748)
A:  2 R: 565 P:  0.8358 -0.1995  0.9097 G:  0.0000  0.0000  0.0000 ( 0.01840)
A:  2 R: 655 P:  0.3358  0.3005  1.4097 G:  0.0000  0.0000  0.0000 ( 0.01082)
A:  2 R: 645 P:  1.3358 -0.1995  0.9097 G:  0.0000  0.0000  0.0000 ( 0.01469)
A:  2 R: 665 P:  0.3358  0.8005  0.4097 G:  0.0000  0.0000  0.0000 ( 0.01984)
(... and idealize cell.)
Tolerances for distances: 0.100, angles:  4.000 and planes: 0.700 opt= 2.0
Idealisation according to type 4
Transformed old basis:
-1.0000 0.0000 0.0000   0.0000-1.0000 0.0000   0.0000 0.0000 1.0000
(Search again for symmetries.)
Tols: 0.2500 Tolt: 0.2500 Tolg: 0.1000 Ref: CA
4-Axis [ 556 ]
2-Axis [ 565 ]
```

```
2-Axis [ 655 ]
2-Axis [ 645 ]
2-Axis [ 665 ]
A:  4 R: 556 P:  0.6642  0.1995  0.4097 G:  0.0000  0.0000  0.0000 ( 0.01218)
A:  2 R: 565 P:  0.6642  0.1995  0.4097 G:  0.0000  0.0000  0.0000 ( 0.01356)
A:  2 R: 655 P: -0.3358  0.1995  0.4097 G:  0.0000  0.0000  0.0000 ( 0.01080)
A:  2 R: 645 P: -0.3358  0.1995  1.4097 G:  0.0000  0.0000  0.0000 ( 0.00198)
A:  2 R: 665 P: -0.3358  1.1995  0.9097 G:  0.0000  0.0000  0.0000 ( 0.01218)
Z 1    0.16419   0.19947   0.40968 (0.017334) (Nothing new found.)
```

RGS finds space group P4/mmm, and writes a file for KPLOT:

```
NDLG
MTRI;
TZUR   0.16419  -0.30053   0.40968
TZ    1.0000000   0.0000000   0.0000000 =
      0.0000000   1.0000000   0.0000000 =
      0.0000000   0.0000000   1.0000000
RG   123 (Following Int. Tables 3rd edition; HMS: P4/MMM  )
RPSY * * -2.0 -1; NPZ * *
CLSE;DLG
```

The result:

```
Cell constants:
3.9641 3.9641 5.3035 90.0000 90.0000 90.0000 Vol = 83.34
a* = 0.25227 b* = 0.25227 c* = 0.18855 al* = 90.00 be* = 90.00 ga* = 90.00
Symmetries (Spg.  No.  123 P4/MMM ):
1) X, Y, Z
2) -X, -Y, Z
...
15) -Y, -X, -Z
16) Y, X, -Z
Atoms:
No Name x y z r Clrpnt
3 Ca 2 0.500000 0.500000 0.000000 0.3000 0
4 Cl 0 0.500000 0.500000 0.500000 0.3000 0
5 Cl 0 0.000000 0.000000 0.000000 0.3000 0
```

Note that the new cell is smaller and has been re-arranged.

The next example is given as an exercise, and only the final result is given. A computer simulation of MgF_2 produced the following structure:

```
NDLG
Z 3.096 4.020 5.34 89.82 90.03 89.81
AE 2
SE 1
ATOM Mg 1 .3288806 .7613257 .9816873
ATOM Mg 2 .8304403 .7591122 .4817613
ATOM F  1 .3300893 .9868472 .6475843
```

```
ATOM F  2 .8282566 .9870328 .1482460
ATOM F  3 .3307128 .5330645 .3153760
ATOM F  4 .8298074 .5333360 .8154464
CLSE;DLG
```

The space group P$\bar{3}$m1 is found, and after performing the transformations suggested, the result is:

```
Cell constants:
    3.0863    3.0863    4.0200   90.0000   90.0000  120.0000   Vol =     33.16
 a* = 0.37414 b* = 0.37414 c* = 0.24876 al* =  90.00 be* =  90.00 ga* =  60.00
 Symmetries (Spg. No.   164  P3-M1   ):
  1)  X, Y, Z
  2)  -Y, X-Y, Z
  ...
 12)  X-Y, -Y, -Z
 Atoms:
   No  Name          x         y         z         r    Clrpnt
    3  Mg  1     0.000000  0.000000  0.500000   0.3000       0
    4  F   1     0.333333  0.666667  0.726870   0.3000       0
```

The two following examples are given to illustrate that it is sometimes advisable not to transform back to the original cell after idealisation because the transformed cell agrees better with the symmetries. But it is necessary then to add centering symmetries, in case the cell becomes bigger, or to remove redundant atoms and duplicate symmetries, in case the cell becomes smaller.

First, here is a very simple example, the primitive version of NaCl:

```
NDLG
Z 3.96 * * 60 60 60
SE 1
AE 2
ATOM Na 1 0 0 0
ATOM Cl 1 .5 .5 .5
CLSE;DLG
```

Searching for symmetries with SFND generates the following output:

```
Tolg: 0.1000 Tols: 0.2500 Tolt: 0.2500
4-Axis [ 644 ]
4-Axis [ 646 ]
...
2-Axis [ 654 ]
A:  4 R: 644 P:  1.0000  1.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  4 R: 646 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  4 R: 664 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  3 R: 666 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  3 R: 626 P:  1.0000  1.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  3 R: 662 P:  1.0000  1.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  3 R: 844 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
```

```
A:  2 R: 556 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 565 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 655 P:  0.0000  0.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 546 P:  0.0000  1.0000  0.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 645 P:  0.0000  1.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
A:  2 R: 654 P:  0.0000  1.0000  1.0000 G:  0.0000  0.0000  0.0000 ( 0.00000)
Z 1    0.00000   0.00000   0.00000 (0.000000)
Cell extremely distorted. A transformation is necessary.
```

The hint that the cell is distorted usually has to be taken seriously. Idealizing and setting opt=2 one gets:

```
>ZIDL * * * 2

Idealisation according to type 6
The new cell volume is   4.00000 times the old one.
Transformed old basis:
-0.5000-0.5000 0.0000  -0.5000 0.0000-0.5000   0.0000-0.5000-0.5000
```

The fourfold axes found are used to define a new cell. Because the cell has been enlarged, centering symmetries have to be introduced. These can be derived from the transformed old basis. But since we had specified opt=2, the program takes care of this automatically. The result:

```
Cell constants:
    5.6003    5.6003    5.6003   90.0000   90.0000   90.0000   Vol =    175.64
 a* = 0.17856 b* = 0.17856 c* = 0.17856 al* =  90.00 be* =  90.00 ga* =  90.00
 Symmetries:
  1)  X, Y, Z
  2)  1/2+X, 1/2+Y, Z
  3)  1/2+X, Y, 1/2+Z
  4)  X, 1/2+Y, 1/2+Z
 Atoms:
   No  Name          x         y         z         r    Clrpnt
    3  Na  1     0.000000  0.000000  0.000000   0.3000       0
    4  Cl  1    -0.500000 -0.500000 -0.500000   0.3000       0
```

The second example is again the result of a computer simulation. The packing of a AB_2 structure was studied and for technical reasons an atom E introduced (representing a pair of electrons). The following configuration was encountered:

```
NDLG
Z   4.92161   3.78057   9.67181  82.36315  94.18374  76.33431
SE 1
AE 2
ATOM   B   0  .1744231  .1195827  .7929490
ATOM   B   0  .0691554  .8584730  .8504569
ATOM   E   0  .1217892  .9890279  .8217030
ATOM   B   0  .0590812  .3621287  .3441971
ATOM   B   0  .1635934  .6309381  .2933850
```

```
ATOM   E   0  .1113373  .4965334  .3187910
ATOM   B   0  .5618897  .6118708  .0955053
ATOM   B   0  .6699524  .8768132  .0442606
ATOM   E   0  .6159210  .7443420  .0698830
ATOM   B   0  .5702492  .0828383  .5980743
ATOM   B   0  .6732230  .3530475  .5471899
ATOM   E   0  .6217361  .2179429  .5726321
ATOM   A   1  .6088738  .9846342  .3203271
ATOM   A   1  .6222734  .4815717  .8203067
ATOM   A   1  .1182584  .2514962  .0686944
ATOM   A   1  .1210187  .7278109  .5711045
CLSE;DLG
```

The analysis of symmetries produced the following output:

```
>SFND;;

Tolg: 0.1000 Tols: 0.2500 Tolt: 0.2500
Z 1    0.12179   0.48903   0.32170 (0.140502)
Z 2    0.11656   0.24278   0.07025 (0.215567)
Z 3    0.36886   0.36668   0.44579 (0.095926)
Z 4    0.36363   0.12044   0.19434 (0.188356)
T 1    0.00000   0.50000   0.50000 (0.130639)
T 2    0.50000   0.25000   0.75000 (0.142718)
T 3    0.50000   0.75000   0.25000 (0.142718)
```

Symmetry T1 is an A-centering while T2 is an unconventional centering. These centerings are processed using the command 'ITS', followed by another 'SFND':

```
>ITS;
Symmetries O.K.
Symmetries O.K.
>SFND;;
Tolg: 0.1000 Tols: 0.2500 Tolt: 0.2500
2-Axis [ 655 ]
2-Axis [ 546 ]
2-Axis [ 566 ]
A:  2 R: 655 P:  0.2592  0.5238  0.6923 G:  0.0000  0.0000  0.0000 ( 0.10034)
A:  2 R: 546 P:  0.2592  0.0238  0.6923 G:  0.0000  0.0000  0.0000 ( 0.05097)
A:  2 R: 566 P: -0.2408 -0.4762  1.1923 G:  0.0000  0.0000  0.0000 ( 0.04524)
Z 1    0.25920   0.02381   0.19229 (0.017376)
```

New translational symmetry has not been found. But now the cell is distorted with respect to the symmetries found. Therefore, it must be idealized.

```
>ZIDL***2

Idealisation according to type 3
The new cell volume is   2.00000 times the old one.
Transformed old basis:
 0.0000 1.0000 0.0000  -0.5000 0.0000 0.5000   0.5000 0.0000 0.5000
```

What now happens is the following: When transforming to the cell indicated by the symmetries found, the cell is enlarged. Consequently centering symmetries have to be introduced. The result so far is:

```
Cell constants:
    4.3972    3.1970    6.1287   90.0000   90.0000   90.0000   Vol =     86.16
 a* = 0.22742 b* = 0.31279 c* = 0.16317 al* =  90.00 be* =  90.00 ga* =  90.00
 Symmetries:
  1)  X, Y, Z
  2)  1/2+X, Y, 1/2+Z
 Atoms:
   No  Name          x         y         z         r    Clrpnt
    3  B   0    -1.166568  1.759190  0.252026   0.3000       0
    4  B   0    -1.664955  1.759211 -0.035932   0.3000       0
    5  E   0    -1.665761  1.759201 -0.141953   0.3000       0
    6  A   1    -0.664548  1.257822 -0.641938   0.3000       0
```

The command 'SFND' and passing the result to RGS results in:

```
NDLG
MTRI;
TZUR   0.25920  -0.16576   0.14195
TZ   0.0000  1.0000  0.0000  0.0000  0.0000  1.0000  1.0000  0.0000  0.0000
RG    65 (Following Int. Tables 3rd edition; HMS: CMMM    )
RPSY * * 0.5 -1; NPZ * *
CLSE;DLG
```

If the transformations are performed as suggested, one gets finally:

```
Cell constants:
    4.3972    6.1287    3.1970   90.0000   90.0000   90.0000   Vol =     86.16
 a* = 0.22742 b* = 0.16317 c* = 0.31279 al* =  90.00 be* =  90.00 ga* =  90.00
 Symmetries (Spg. No.    65  CMMM    ):
  1)  X, Y, Z
  2)  -X, -Y, Z
  ...
 16)  1/2+X, 1/2-Y, Z
 Atoms:
   No  Name          x         y         z         r    Clrpnt
    3  B   0     0.000000  0.606021  0.500000   0.3000       0
    4  E   0     0.500000  0.000000  0.500000   0.3000       0
    5  A   1     0.500000  0.500000  0.000000   0.3000       0
```

Since the steps taken to find the symmetries and to idealize the cell in most cases follow the same scheme, there exists a built-in macro 'SSI' (search symmetries and idealize) to do this job. But note that at many places in the macro-command 'SSI' default values are used in the individual commands, which cannot be changed.

Another application where RGS may be used should be mentioned. In the International Tables, with each space group the maximal non-isomorphic subgroups are listed. But this

often leads to a non-standard setting, and the question arises: How can we transform to the standard setting?

In the tables, with each subgroup a list of symmetries remaining in this subgroup is given. These are "input" into RGS, and a search for the space group is performed.

Example:

A subgroup of P$\bar{3}$m1 is e.g. C2/m if the symmetries 1,4,7, and 10 are kept.

```
>Z * * *  * * 120  ! Because we have a trigonal system
>SY 'Y,X,-Z'       ! Symmetry 4, 1 must be omitted!!
>ST -1             ! Symmetry 7, may also be entered as
                   ! sy '-x,-y,-z'. Symmetry 10 may be
                   ! omitted, because it is generated when
                   ! building the group.
>ANAL              ! May not be omitted.
>RGS               ! Seach the space group.
```

The commands are explained in detail in the RGS manual. The result is (finally printed to screen):

```
No origin shift necessary.
TZ  -1.0000  1.0000  0.0000 -1.0000 -1.0000  0.0000  1.0000 -1.0000  1.0000
It is space group C2/M         no  12
```

In version 8.3.0 and later, Kplot supports building of symmetry-trees. For all space groups (in their "standard setting") the maximal non-isomorphic space groups are tabulated, and one way (often there are several) to perform this transformation. The command is `TUG`, which will either list those subgroups or perform a transformation. *Example:*

```
>rg 225
>tug 0
Maximal subgroups of Fm-3m    (No.   225)
     Symbol     No. Index Block
( 1) I4/mmm     139     3   1
( 2) R-3m     16601     4   1
( 3) Fm-3       202     2   1
( 4) F432       209     2   1
( 5) F-43m      216     2   1
( 6) Pm-3m      221     4   2A
( 7) Pn-3m    22402     4   2A
```

If the transformation is supposed to be not only listed but also to be implemented, the space group number listed in the third column has to be given as parameter with `TUG` or the number of the first column as negative number.

Another useful command is `SPUG` (search path to subgroup). One enters the "starting" space group and the "target" space group, and the program tries to find sequences of subgroups that are as short as possible. *Examples:*

```
>spug 225 14
(  1)  Fm-3m      225  R-3m     16601  C2/m      1202  P21/c     1402

>spug 225 15
(  1)  Fm-3m      225  R-3m     16601  C2/m      1202  C2/c      1502
(  2)  Fm-3m      225  R-3m     16601  R-3c     16701  C2/c      1502
```

The compounds $Se_4[Mo_2O_2Cl_8]$ and $(Se_4)_2[HfCl_6][Mo_2O_2Cl_8]$ (M. Kellner, 2001) crystallize in the space group $P2_1/c$ (14). Both compounds are isotypic and may be derived from the NaCl type. How should one reduce the symmetry?

The path has been given above. However, one has to be careful because the transformations and the paths are not unique.

Starting point: $Fm\bar{3}m$, Na: 4a, Cl: 4b, a=5.6, V=175.6.

The first transformation to $R\bar{3}m$ is translationsgleich of index 4. However, Kplot transforms to the rhombohedral setting. If a transformation to the hexagonal setting is wanted, two commands must follow.

```
>TUG 16601  ! Transform to the first subgroup,
>RTHO       ! and then to the hexagonal setting.
>RG 16602   ! Load "correct" space group.
>tug,       ! Show maximal non-isomorphic subgroups.
Maximal subgroups of R-3m     (No. 16602)
     Symbol     No. Index Block
( 1) R32      15501     2   1
( 2) R-3      14801     2   1
( 3) R3m      16001     2   1
( 4) C2/m      1202     3   1
( 5) P-3m1      164     3   2B
( 6) R-3c     16701     2   2B
```

The result: $a = 3.9598$, $c = 9.6995$, $V = 131.7$, Na: 3a, Cl: 3b.

From the list above, one sees that the next step to C2/m is translationengleich of index 3.

```
TUG -4
```

The result: $a = 6.8586$, $b = 3.9598$, $c = 3.9598$, $\beta = 54.7356$, $V = 87.81$, Na: 2a, Cl: 2d.

The resulting cell has only half the desired volume, i.e. an isomorphic transfomation is necessary in order to obtain the "correct" cell. How does one find this transformation?

The following considerations lead to an efficient solution: The a- and the b-axes may not be altered except for the opposite directions; following the Int. Tab. only a transformation of block IIc with $c' = 2c$ can be used.

```
OTM 1 ! Reset the transformation matrix.
TIM 1 0 0  0 1 0  0 0 2
```

To find the standard setting one can leave KPLOT and use the program RGS:

```
SRGS temp.dat
```

and then in program RGS:

```
GET temp.dat
RGS
RTF RGS.OUT
```

back to Kplot:

```
IMP RGS.OUT  ! Import result from RGS
OTM 0        ! Show result.
```

Result: `TZ  -1 0 0   0 1 0   1 0 -2`
$a = 6.8586$, $b = 3.9598$, $c = 6.8586$, $\beta = 109.47$, $V = 175.6$ Na: 2a and 2d, Cl: 4i.

(A different way to find the last transformation may would to use the command `'SZ'`, which is preferable if the target cell is already known.)