FOCUS MANUAL

Table of Contents

Modified: 24-05-2014, Stef Smeets (stef.smeets@mat.ethz.ch)

1. Quick installation

Download the FOCUS package from https://www.crystal.mat.ethz.ch/Software/

Pre-compiled versions for OS X (Intel x64) and Linux (Ubuntu 12.04 x64) are available.

(Installation from source requires `svn` and `curl` to be installed on Linux systems, and Command Line Tools for Xcode and gfortran for systems running OS X)

Extract using:
```
        tar -xvzf focus-all.tar.gz
```
Install (and compile) in current directory by running:
```
        sh runme.sh
```

At this point, all the files necessary to run FOCUS are in the ./bin/, ./kriber_f/ and ./dls_f/ directories. Follow the instructions on the screen on how to set the correct path and environment variable to COSEQ_DB. The script has been tested to work correctly on recent versions of OS X and Ubuntu.

Note: The installation scripts set up the paths for the current directory. In case the installation is moved to another directory, be sure to update ./kriber_f/kriber.csh (line 56 and 57) and ./dls76/dls76.csh (line 5) with the correct path, as well as the system path and the $COSEQ_DB environment variable.

2. Full installation procedure

The FOCUS package comes with several useful tools for working with and analyzing FOCUS output. FOCUS and KRIBER are written in C and should therefore be compatible with any C compiler. Additionally, FOCUS requires Scons (2.2.0), fftw (3.3.2), libtbx (16772) and Python2.x (2.7) for compilation. The versions in brackets are the ones that have been tested.

DLS76 was written in FORTRAN and should be compiled using gfortran.

It is recommended that the installation scripts provided (see above) be used to compile FOCUS, or at least be used as a guide.

2.1.1. Overview of dependencies

FOCUS
- gcc
- Scons
- libtbx
- fftw

Kriber
- gcc

DLS76
- gfortran

install scripts
- curl
- svn

Curl and SVN are used in the install scripts to download the required packages, such as Scons (curl), libtbx (svn) and fftw (curl), from the internet.

This installation script will download and compile dependencies, and compile FOCUS and related programs in 4 steps by calling the following helper scripts:

1. `focus_build_script.sh`
   Downloads and compiles FFTW 3.3.2, libtbx, scons 2.2.0 and FOCUS (requires `curl` and `svn`)
2. `other_build_script.sh`
   Compiles and sets up the helper tools needed for FOCUS, including KRIBER
3. `dls_build_script.sh`
   Compiles and sets up DLS76
4. `install.sh`
   Moves all executables into ./bin/ and sets up paths.

2.1.2. OS X

For OS X, C-compilers are available via the Command Line Tools for Xcode https://developer.apple.com/downloads/index.action (a developer account is needed) or GCC http://hpc.sourceforge.net/ . Curl and Python come pre-installed, svn is not available in OS X 10.8+ and is a part of the Command Line Tools.

gfortran is available via http://hpc.sourceforge.net/ or fink/brew/macports.

The installation script was tested on OS X 10.7 and OS X 10.8 using the C-compiler from the Command Line Tools and gfortran from HPC and should work without any problems.

2.1.3. Linux/Unix

Most Linux distributions come with their own compiler suite, otherwise compilers are usually available via the package manager or http://gcc.gnu.org . Python usually also comes preinstalled, but is otherwise available from the packages manager or http://www.python.org

SVN and Curl did not come preinstalled on the system used for testing the installation procedure (Ubuntu 12.04), but both were available via the package manager.

The installation script was tested successfully on Ubuntu 12.04 LTS x64.

2.1.4. Compiling

In order to make installation as convenient as possible, the following command can be used to install FOCUS and all related tools to the current directory:

```
curl http://www.crystal.mat.ethz.ch/Software/focus_online_install.sh | sh
```

Everything is downloaded and self-contained in the current directory.

3. The FOCUS input file

A template for the input file can always be generated directly using:

```
    focus > template.inp
```

The template generated has sensible defaults for solving zeolite structures. Of course, some parameters are more sample specific than others. Some of the parameters in the template can be ignored or removed (e.g. all the profile related ones from `GenerateFWHM` to `NormalizeSurface`).

A sample input file that solves a structure readily has been provided in the directory ./manuals/example/

The parameters relevant for structure solution are discussed below.

Formatting:

#hash	denotes numerical values. #N is usually used for integers and #d for distances in Å.
$dollar	denotes an arbitrary string
x/y	denotes a choice of keywords
(parentheses)	denote optional input
[Square brackets]	indicate repeated input.

3.1.1. Title

Title $title

Set a title for the calculation. It is recorded in the output file.

3.1.2. SpaceGroup

SpaceGroup $spgr

Spacegroup for the supplied data.

3.1.3. UnitCell

UnitCell #a #b #c #al #be #ga

Dimensions of the unit cell in Å and angles in degrees. Parameters can be given in shortened form, i.e. by specifying only *a* for a cubic system.

3.1.4. AtomType

AtomType +/- node/nodebridge/* $label #N (#occ) (#U) (#scattering factor)

The `AtomType` parameter defines the cell contents of the structure to be solved, as determined by a chemical analysis or estimated by other means. One `AtomType` line for each type of atom needs to be specified. The first item after the keyword `AtomType` is either "+" or "-". All atoms specified with an `AtomType` line are used in the calculation of F000 (the Fourier magnitude at the origin of reciprocal space), but only atoms with the "+" marker are considered in the atom and/or the framework fragment recycling procedures. The next item is a "class label", `Node`, `NodeBridge`, or " ", which determines how FOCUS should treat this atom type. The latter is for non-framework atoms. Nodes are typically T atoms and `NodeBridge` denotes bridging (O) atoms. For the following item, an "atom label" and the number of atoms of this type per unit cell should be supplied. The rest is optional.

For a typical zeolite, such as ZSM-5, the `AtomType` parameters look like this:

```
AtomType  +  Node        Si      96
AtomType  -  NodeBridge  O      192
```

Also possible - but not used here - is the definition of the occupancy factor to be used in the recycling (default: 1.0), the isotropic temperature factor (default: 0.035), and a "scattering factor label" (default: derived from the preceding atom label).

For example, the line

```
AtomType  -  *  Ow  20  1.25  0.05  O
```

Describes an oxygen with an occupancy of 1.25 and an isotropic temperature factor of 0.05 Å^2, which is a commonly used approximation for water molecules in zeolite channels. The scattering factor used is that of oxygen, and 20 water per unit cell are expected. However, experience has shown that recycling extra framework atoms is not efficient, nor necessary, and for the calculation of F000 it would be sufficient to supply one `AtomType` line for oxygen and one for hydrogen using the default occupancy and temperature factors.

3.1.5. Chemistry

Chemistry MinDistance/MaxDistance $NodeType1 $label1 $NodeType2 $label2 #d

The `Chemistry` lines define the expected connectivity of the atom types specified before. This is related to the atom recycling procedure. The `Chemistry` lines define the individual minimum and/or maximum distances for each pair of atom types which are used in the atom recycling procedure. Following `Chemistry MinDistance/MaxDistance` are two pairs of "class label" and "atom label" as defined on `AtomType` lines, and the minimum distance for this pair of atom types in the same units as the lattice constants, usually Å. It should be noted that bonding is not considered in atom recycling mode. The minimum distances apply to all pairs of atoms, whether they are bonded or not.

For ZSM-5, this section could look like this:

```
Chemistry  MinDistance  Node        Si    Node        Si    2.6
Chemistry  MinDistance  NodeBridge  O     Node        Si    1.4
Chemistry  MinDistance  NodeBridge  O     NodeBridge  O     2.3
```

Specification of the `MaxDistance` is also possible in the same way. Note: if there is a "-" on the `AtomType` line as for

```
AtomType  -  NodeBridge  O      192
```

this atom type is not used in the atom recycling procedure. Therefore it would be sufficient to supply only the first `Chemistry` line for Si--Si.

3.1.6. MaxPotentiAlatoms

MaxPotentialAtoms #N

This parameter gives the maximum number of peaks which are considered in the assignment algorithm.

Example: `MaxPotentialAtoms 130`

With this value, if the algorithm tries to assign a silicon atom to one of the peaks in the asymmetric unit, but is not able to find a valid position among the peaks in the asymmetric unit which generate the 130 highest peaks in the unit cell, the silicon is not assigned at all.

3.1.7. MaxRecycledAtoms

MaxRecycledAtoms #N

`MaxRecycledAtoms` prescribes the maximum number of atoms in the unit cell that are actually assigned and is forced to be smaller or equal to `MaxPotentialAtoms`.

3.1.8. FwSearchMethod

FwSearchMethod FwTracking/AltFwTracking

The value can be either `FwTracking` or `AltFwTracking`, which are simple backtracking and "colored" backtracking, respectively. The latter is only useful in case of strict alternation of the T atoms, for example in case of a germanosilicate.

3.1.9. MaxPeaksFwSearch

MaxPeaksFwSearch #N

In the atom recylcing procedure, only complete frameworks are sought, and `MaxPeaksFwSearch` defines the maximum number of peaks in the unit cell that are used in the backtracking procedure.

3.1.10. MaxPeaksFwFragmentSearch

MaxPeaksFwFragmentSearch #N

In framework fragment recycling mode, `MaxPeaksFwFragmentSearch` determines the maximum number of peaks. Since the fragment search is significantly slower than the search for complete frameworks only, it is sometimes necessary to set `MaxPeaksFwFragmentSearch` to a smaller value than `MaxPeaksFwSearch` in order to retain reasonable computing times.

3.1.11. MinNodeDistance

MinNodeDistance #d

The `MinNodeDistance` and `MaxNodeDistance` parameters establish the lower and upper limits for the node-node distances which are used in the preparation of the lists of potential node-node bonds. In this case, a tolerance of 0.5 Å around the "ideal" distance of 3.1 Å is set:

```
MinNodeDistance 2.6
MaxNodeDistance 3.6
```

This is usually a good starting point for aluminosilicates. In case that no solutions are found, increasing the upper limit may help.

3.1.12. MaxNodeDistance

MaxNodeDistance #d

See `MinNodeDistance`.

3.1.13. MaxSymNodes

MaxSymNodes #N

`MinSymNodes` and `MaxSymNodes` set the lower and upper limits for the number of framework nodes per unit cell. While `MinSymNodes` just prevents frameworks with too low a density from being evaluated and printed and can usually be left at 0, `MaxSymNodes` cuts complete branches of the search tree. On the one hand, this can reduce the computing time for frameworks with a well-established low density, but on the other, one has to be careful not to prescribe a value that is too small. Normally, the choice for `MaxSymNodes` is based on the consideration that the number of T-sites per 1000 Å^3 in a zeolite must be less than 20.

3.1.14. MinSymNodes

MinSymNodes #N

See `MaxSymNodes`.

3.1.15. NodeType

NodeType #Nbonds #Natoms [$symaxis ...]

The `NodeType` keyword defines the number of bonds for a given node type, the maximum number of nodes of this type in the asymmetric unit and a list of the symmetry elements which can not be occupied by a node of this type.

For example, by default, this `NodeType` should always be specified:

```
        NodeType 4 * -6 -3 -1 4 6
```

In the example, only one node type with tetrahedral connectivity is defined. The asterisk "*" specifies that an unlimited number of nodes in the asymmetric unit can be of this type. The following numbers "`-6 -3 -1 4 6`" specify that this node type cannot be on a six- or threefold rotoinversion axis, an inversion center, or a four- or sixfold rotation axis.

Multiple NodeTypes can be specified, for example adding: NodeType 3 1 -6 -3 -1 4 6

tells FOCUS that there is a single Q^3 T atom in the asymmetric unit that is not on any of the specified axes of symmetry. If such information is known, it should be specified, as it can mean the difference between solving the structure and and finding no solutions at all.

However, adding NodeTypes means that the algorithm has to check every possible atom position for given connectivities, which will exponentially increase the time taken for the framework search.

3.1.16. MinLoopSize

MinLoopSize #N

Example: `MinLoopSize 4`

Supplying a value greater than three for `MinLoopSize` has two consequences: when atoms are recycled and only complete frameworks are sought, frameworks which have loops with less than `MinLoopSize` members are rejected (that just means they are not printed). In framework fragment recycling mode, the fragments which are candidates for the "largest fragment" for recycling are checked for `MinLoopSize`. Unfortunately, the present implementation of the loop size test is very time consuming. The time spent for the fragment search increases by roughly 40%. In this example, `MinLoopSize` was therefore kept at its default value of three, although four is perhaps more appropriate for high silica frameworks. (However, the structure of the high silica ZSM-18 (MEI) does contain 3-rings).

3.1.17. MaxLoopSize

MaxLoopSize #N

Example: `MaxLoopSize 24`

`MaxLoopSize` is less critical than `MinLoopSize` and just specifies the maximum loop size up to which the LC algorithm advances. The default value of 24 is sufficient for all known zeolite topologies. For loops with more than `MaxLoopSize` members, a "0" is printed. Cases where smaller values would result in a speed gain for the price of having some zeros in the LC are hardly imaginable.

3.1.18. EvenLoopSizesOnly

EvenLoopSizesOnly On/Off

If `EvenLoopSizesOnly` is turned Off, this means that all loop sizes greater than or equal to `MinLoopSize` are allowed. The `EvenLoopSizesOnly` option was introduced for the search for frameworks where a strict alternation of two atom types is expected. In these cases, only even loop sizes are possible. A special problem arises for aluminophosphates. Since the scattering powers of Al, and P are only slightly different, it is often not possible to determine the true space group from the powder profile. Only after the structure is known, can one introduce the strict Al-P alternation, which in many cases reduces the symmetry. For this situation, `EvenLoopSizesOnly` provides a more robust alternative to `AltFwTracking`.

It has to be noted that in framework fragment search mode the impact of `EvenLoopSizesOnly` on the computing time requirements is similar to setting `MinLoopSize` to a value greater than three. However, since loop sizes have to be computed only once per framework or framework fragment, `MinLoopSize` greater than three does not result in more time consumption if `EvenLoopSizesOnly` is switched On.

3.1.19. Checck3DimConnectifity

Check3DimConnectivity On/Off

If On, a filter procedure is called for each framework topology found. Only 3-dimensionally connected frameworks can pass this filter, layer or chain structures are rejected.

3.1.20. IdealT_NodeDistance

IdealT_NodeDistance #d

specifies the "ideal" node-node distance for four-connected nodes. This is the basic value for the geometrical tests and is normally set to 3.1 Å for aluminosilicates.

3.1.21. CheckTetrahedralGeometry

CheckTetrahedralGeometry Off/Normal/Hard

This parameter adds to the geometrical tests and checks for Tetrahedral connectivity of the framework. In most cases, this can be set to `Normal`. Turning this parameter `Off` tends to result in unrealistic frameworks. For high silica and Si-Al frameworks such as Dodecasil-1H, the `Hard` test is appropriate.

3.1.22. RandomInitialization

RandomInitialization Time/#seed

Defines the "seed" value for a portable pseudo random number generator, which is used to generate the starting phases. The special value Time tells FOCUS to use the machine time for the automatic determination of the seed value, which is then printed on the output file. This integer value - like any positive integer value - can be resupplied with `RandomInitialization` in order to rerun FOCUS with different output options or for testing or debugging purposes.

3.1.23. FeedBackCycles

FeedBackCycles [#Natom #Nfw ... ...]

Example: `FeedBackCycles 1 1 1 1 1 1 1 1 1 1`

This keyword is followed by an arbitrarily long sequence of nonnegative integers (including zero). The first integer specifies the number of times the atom recycling procedure is to be used in one trial, the second integer is for the number of framework fragment recycling loops, the third again for atom recycling, and so on. In the example, ten cycles with alternation of atom and framework fragment recycling are requested. Experience has shown that a simple alternation of atom recycling and framework fragment recycling, as in the example, is usually the most efficient approach.

3.1.24. FeedBackBreakIf

FeedBackBreakIf PhaseDiff < 5.00 % and DeltaR < 1.00 %

The recycling is prematurely terminated if both the phase set and the RF residual value have converged. Another special situation is, when no fragment which can be recycled is found. In this case, a trial continues with atom recycling (but the cycle is still counted as framework fragment recycling cycle).

3.1.25. Grid_xyz

Grid_xyz #Nx #Ny #Nz

Sets grid for the electron density map is defined such that a resolution of about 1/3 Å is achieved. One has also to take care that all symmetry elements pass through grid points. In its present form, FOCUS does not automatically generate an appropriate grid, but it does refuse to work with grid sizes that do not conform to this requirement. For example, in space group P-1 the grid sizes for all directions have to be a multiple of two, in order to have all inversion centers laying on a grid point. In the case of space group *P*6/*mmm*, the grid size in the *z*-direction has to be a multiple of two, and the grid sizes for the *x*- and *y*-direction have to be a multiple of six.

3.1.26. eDensityCutOff

eDensityCutOff #cutoff (%)

Default: `eDensityCutOff 1 %`

specifies the lower cut-off value for the peak search in the electron density maps. This specification can either be an absolute value, e.g. `eDensityCutOff 1.0`, or relative to the maximum value of the whole map, as is in the default. The overall maximum value of `MaxPotentialAtoms`, `MaxPeaksFwSearch`, and `MaxPeaksFwFragmentSearch` is the maximum number of peaks in the unit cell which are put on the peaklist by the peaksearch procedure. However, if there are less than this number of peaks with a maximum peak height above the value set by `eDensityCutOff`, the list will contain fewer peaks.

3.1.27. MinPfI

MinPfI #N

Default: `MinPfI 17`

`MinPfI` ("minimum number of points for interpolation") defines the minimum number of grid points with a positive electron density value surrounding a grid peak position. If the actual number is fewer than `MinPfI`, no interpolation for the peak position is carried out and the coordinates of the grid point are retained.

3.1.28. CatchDistance

CatchDistance #d

Default: `CatchDistance 0.5`

CatchDistance is the minimum distance a peak has to have to all of its symmetrically equivalent peaks (self-distance). For self-distances smaller than `CatchDistance`, a procedure is activated, which moves the peak onto the symmetry element which is responsible for the close contact.

3.1.29. eD_PeaksSortElement

eD_PeaksSortElement Grid_eD/Maximum/Integral.

Defines how the peaks are sorted.

3.1.30. Lambda

Lambda #d/$code

Sets the wavelength either in Å or one of the codes for the internally stored wavelengths.

```
CrA1: 2.28970        CrA2: 2.29361        Cr: 2.2909
FeA1: 1.93604        FeA2: 1.93998        Fe: 1.9373
CuA1: 1.54056        CuA2: 1.54439        Cu: 1.5418
MoA1: 0.70930        MoA2: 0.71359        Mo: 0.7107
AgA1: 0.55941        AgA2: 0.56380        Ag: 0.5608
```

3.1.31. FobsMin_d

FobsMin_d #d

Sets the minimum d-spacing for the reflections to be used in Å.

3.1.32. FobsScale

FobsScale #N

Defines the scale factor.

3.1.33. SigmaCutOff

SigmaCutOff #cutoff

This is normally set to 3. If standard deviations are available, reflections with an intensity smaller than `SigmaCutOff` times their standard deviation can be excluded.

3.1.34. OverlapFactor

OverlapFactor #of

Usually, a good value for the overlap factor is 0.3. Together with the individual FWHM for each reflection is used to determine the overlap groups, which are then processed according to `OverlapAction`.

3.1.35. OverlapAction

OverlapAction NoAction/EqualF2/EqualMF2

`NoAction` specifies that no action should be taken if reflections are in the same overlap groups. In case of `EqualF2`, overlapped reflections are equipartitioned to have equal structure factor amplitudes and in case of `EqualMF2`, the reflections are equipartitioned based on the structure factor amplitude times multiplicity. In case of electron diffraction, this value should be set to `NoAction`, while for powder diffraction data, `EqualMF2` is usually the most appropriate.

3.1.36. Reflection Usage

ReflectionUsage #N

This parameter specifies the number of reflections that are actually used. This can be absolute, for example `ReflectionUsage 80` will select the 80 highest reflections, or it can be relative, as in the example. In the latter case, reflections are selected in descending order of (equipartitioned) intensity times multiplicity (M.F) until the prescribed percentage of the total sum of M.F over all input reflections is accumulated. Usually, no more th an 400 reflections are necessary.

3.1.37. ScatteringFactorTable

ScatteringFactorTable xray/electron

This parameter specifies whether given structure factor amplitudes are from an electron or X-ray diffraction experiment.

3.1.38. ScatteringFactor

ScatteringFactor $label

Custom scattering factors can be specified by supplying a list of sin(θ)/λ with corresponding scattering factor values. The list should start with the keyword and the label for this particular scatterer, and end with &. The label can then be used the Chemistry and AtomType keywords for example. As an example using the conventional X-ray scattering factors for silicon:

```
ScatteringFactor sf_Si
  0.00   13.99892
  0.05   13.43532
  0.10   12.13342
  0.15   10.76817
  0.20    9.67481
  0.25    8.85950
  0.30    8.22972
  0.35    7.69734
  0.40    7.20320
  0.45    6.71934
  0.50    6.23962
  0.55    5.76816
  0.60    5.31198
  0.65    4.87773
  0.70    4.47063
  0.75    4.09426
  0.80    3.75063
  0.85    3.44035
  0.90    3.16292
  0.95    2.91693
  1.00    2.70033
&
```

3.1.39. Reflections

The last part of the input file is a listing of the extracted Fourier magnitudes. The data are given as reflection indices hkl, observed relative Fourier magnitude, the estimated standard deviation of the Fourier magnitude and the FWHM as derived from the refined profile parameters. If estimated standard deviations are not available, asterisks can be supplied instead. The reflection list should always end with the word "End". For example:

```
#  h   k   l       Fobs       Sigma     FWHM
   0   2   1       4.02           *   0.0535
   0   4   0       2.75           *   0.0534
   1   1   0       4.91           *   0.0534
End
```

4. KRIBER

Kriber can be used to analyze the results of the FOCUS output file. Kriber expects a strudat file to be present in the current directory. A strudat can be created from the FOCUS output file by using the `focus2strudat` script (See below).

The strudat file contains a listing of all the frameworks reported by FOCUS (i.e. the ones in the `fo2hist` output).

The main commands of interest are:

help	Prints out a summary of all commands.
reacs	This command is used to read in a framework from the strudat file. The user is then prompted for the framework to load, which is any framework in the strudat file (without the *).
dise	Show all entries in the strudat file.
addo	O atoms are added in the expected positions between T atoms.
wricif	Save the current framework to a cif file named 'structure.cif'.
dett	Calculate the coordination sequence of the current framework.
wriid	Write an input file for DLS named 'dlsinp' (See below).
exit	Exit the program.

5. DLS76

DLS can be used to perform distance least squares calculations on the framework. The easiest way to generate an input file for DLS is by running Kriber with the 'addo' command and then the 'wriid' command. Then, DLS can be run on the resulting '`dlsinp`' file.

6. EXTRAS

6.1.1. fo2hist

`Fo2hist` is essential for the analysis of the FOCUS output file. It will parse the FOCUS output file, count all identical frameworks, and report the histogram of solutions to the terminal. The idea is that the framework with the most occurences is the correct one.

All coordination sequences are referenced against a database of known frameworks (see ascii file ./kriber_f/coseq). `Fo2hist` will show a note when a match is found. In the output, the first column gives the framework name of the first occurence in the FOCUS output file, the second column the number of occurences of this framework and the third column the percentage.

Example: `fo2hist focus.out`

6.1.2. multifocal

`multifocal` is a python program written to make it easier to run several focus runs in parallel, and thus to speed up the structure solution process. The help file can be accessed with the `-h` command line parameter.

The program can run a single input file in parallel, or run several input files at the same time. The number of processes to run can be specified via the -p flag, and the number of trials per process via `-n`

Example: `multifocal focus.inp –p 8 –r 1000`

This will spawn 8 FOCUS processes, each performing 1000 trials.

There is a 1 second delay between the start of every process on the same input file, in order to avoid having the same seed for the random number generator, which is based on system time.

6.1.3. focus_check

This tool can be used to combine and analyze several output files simultaneously. In essence, the script will perform '`fo2hist`' on all the individual output files and once again on all output files combined.

Example: `focus_check zsm5_*.out`

In addition, a combined output file (all.out) will be created. In this file, the Fw name (i.e. Fw0001) has been renamed to reflect the original output file (i.e. Fw_zsm5_0_0001).

Specific framework names can then be retrieved with:

```
      grep '*Fw' all.out -m NUMBER -n | tail -n 1
```

Where number is the number of the Fw + 1. This will print a name corresponding to the framework name if a strudat file is created using these commands:

```
fo2hist all.out | cut -c2- > tf
section all.out Framework -take=tf > strudat
```

6.1.4. focus2cif

This script automatically converts all the frameworks in a focus output file to cif format. It uses `focus2strudat` and then `strudat2cif`.

6.1.5. strudat2cif

Convert all frameworks in a strudat file to cif format. This script calls Kriber for parsing and writing the cif file via the wricif command. Any command line argument specified will be passed on directly to Kriber, which can be useful to generate cif files of framework structures with oxygens added.

For example: `strudat2cif addo`

Will read every framework from the strudat file, add oxygens in the expected positions and write a cif file.

6.1.6. focus2strudat

Converts the given focus file to a strudat file, ready to be used with Kriber.

7. Example

This example shows the general workflow when using FOCUS using the example in the example directory.

Create a template input file:

```
focus > template.inp
```

Not all of the input parameters in the template are relevant for structure solution, but those that are have been discussed in the FOCUS input file chapter.

In this case, a working example file has been supplied for the DOH framework (doh.inp).

To start, run FOCUS on the input file:

```
focus doh.inp 100 > doh.out
```

The number 100 here specifies how many trials FOCUS should start. In the case of this simple structure, 100 trials is enough. This should take about 10-20 seconds. However, more complex problems can take several hours, or sometimes days to solve. The output file, doh.out, can be analyzed using the '`fo2hist`' program:

```
fo2hist doh.out
```

In this case, the framework is known and `fo2hist` will identify the correct framework as DOH:

```
F0000 : 229 0.70679 = DOH
F0015 : 58 0.179012
F0023 : 23 0.0709877
F0058 : 5 0.0154321
F0209 : 3 0.00925926
F0042 : 1 0.00308642
F0080 : 1 0.00308642
F0151 : 1 0.00308642
```

```
        F0252 : 1 0.00308642
        F0303 : 1 0.00308642
```

In case of an unknown structure, it is useful to look at all frameworks in more detail. To do so, a CIF file can be created by using `focus2cif`:

```
        focus2cif doh.out
```

This creates a cif file for every framework reported in the `fo2hist` output. In this case, Fw0000 is the correct framework. When it is decided to use this framework for refinement, the framework structure can be optimized using distance least-squares with the program DLS76. The easiest way to generate a DLS input file is by using Kriber. First, a strudat file should be created (in this case a strudat file is already available, as `focus2cif` will write one) with `focus2strudat`:

```
        focus2strudat doh.out
```

Then running Kriber:

```
        kriber
```

And issuing the following commands:

```
        reacs
        Fw0000
        addo
        wriid
        exit
```

'`reacs`' will tell the program to read a structure, in this case Fw0000, '`addo`' will add oxygens in the expected bridging positions and '`wriid`' will write a DLS76 input file called '`dlsinp`'.

Finally, run DLS76:

```
        dls76 dlsinp
```

This will generate 2 files, `dlsout` with the results of the least-squares refinement and `nfilea.inp` with the atomic parameters. The easiest way to use the new atomic parameters is to replace them in an existing CIF file, for example one written using the command '`wricif`' in Kriber.

8. Further reading

For further information, please consult the following literature:

Smeets, S; McCusker, L. B.; Baerlocher, C.; Mugnaioli, E. & Kolb, U.
Using FOCUS to solve zeolite structures from three-dimensional electron diffraction data
Journal of Applied Crystallography, 2013, 46, 1017-1023
http://dx.doi.org/10.1107/S0021889813014817

Grosse-Kunstleve, R. W.; McCusker, L. B. & Baerlocher, C.
Zeolite structure determination from powder diffraction data: applications of the it FOCUS method
Journal of Applied Crystallography, 1999, 32, 536-542
http://dx.doi.org/10.1107/S0021889899003453

Grosse-Kunstleve, R. W.; McCusker, L. B. & Baerlocher, C.
Powder Diffraction Data and Crystal Chemical Information Combined in an Automated Structure Determination Procedure for Zeolites
Journal of Applied Crystallography, 1997, 30, 985-995
http://dx.doi.org/10.1107/S0021889897005013

Grosse-Kunstleve, R. W.
Zeolite Structure Determination from Powder Data: Computer-based Incorporation of Crystal Chemical Information
PhD thesis, ETH Zurich, 1996
http://cci.lbl.gov/~rwgk/Dissertation/Dissertation.pdf