

refmac keywords (version 5.5.0026 and later)

Please note that most of these keywords were in the previous versions also. Some of the keywords were implemented as user requests. The numbers inside the brackets (if present) after the keyword indicates the version when this particular option became available.

Keywords that control Xray

Labin: labels from mtz
Twin refinement
Simultaneous experimental phasing and refinement
Map coefficients Weighting xray and geometry terms
Occupancy groups and refinement (5.6.0037)
Bulk solvent (some options are available after 5.6.0078)
Anisotropic Refinement
NCS constrained refinement (from 5.6.0087)

Geometry keywords

Using segment id
External distance, angle etc restraints
Harmonic restraints
Torsion angle restraints
VDW restraints (excluding antibumping between specified atoms - 5.6.0065)
NCS restraint (automatic ncs and local ncs restraints - 5.6.0043)
Harmonic distance and Bvalue restraints: Ridge type regression (5.6.0046)
Basepairs restraints (5.7.0000)
Bvalue restrain (Kullback-Liebler based restraints are active after 5.6.0050)

Controlling output pdb (from version 5.6.0112)

Copy remarks from input PDB

Ligand description

Various protocols

Links to web resources References

Description of some of the keywords

Xray keywords

Labin: labels from mtz file

`LABIN FP=<label> SIGFP=<label> IP=<label> SIGIP=<label> F+= <label>
SIGF+=<label> F-=<label> SIGF-=<label> HLA=<label> HLB=<label> HLC=<label>
HLD=<label> PHIB=<label> FOM=<label> FREE=<label>`

If only FP, SIGFP have been defined then a simple maximum likelihood refinement will be carried out

If F+, F-, SIGF+ and SIGF- have been defined then refinement using multivariate SAD function will be carried out.

If IP and SIGIP have been defined then refinement against intensities will be carried out. In the current version this option works only with *twin* keyword

If HLA, HLB, HLC, HLD have been defined then the external phase information will be used. These coefficients are usually generated by heavy atom refinement programs.

If PHIB and FOM have been defined then again external phase information will be used.

Note that HLA, HLB, HLC and HLD contain more information about phases than PHIB and FOM

Note: In the current version external phase information will not work with twin, SAD options. They will work together in future versions.

If no *labin* keyword is given then the program will try to find amplitudes of experimental intensities and corresponding sigmas as well as *FreeR_flag* labels and carry out simple refinement. If no *labin* keywords and *REFI SAD* keyword has been defined then the program will try to find labels for Friedel pairs, corresponding sigmas and will carry out SAD refinement

Examples of *labin*:

Simple refinement: *labin FP=FP SIGFP=SIGFP FREE=FreeR_flag*

Refinement with the external experimental phases: *labin FP=FP SIGFP=SIGFP HLA=HLA HLB=HLB HLC=HLC HLD=HLD FREE=FreeR_flag*

Refinement with the SAD function: *labin F+=F+ SIGF+=SIGF+ F-=F- SIGF-=SIGF- FREE=FreeR_flag*

Using intensities: *labin IP=I SIGIP=SIGI FREE=FreeR_flag twin*

Twin refinement

Current version takes only one keyword *twin*. All decisions are made automatically.

Note that in the current version of *refmac* (5.5.0031) twin refinement is not compatible with SAD or phased (using HLA, HLB, HLC and HLD) refinement. We are working on this

twin

This keyword gives a signal to the program. When *Refmac* sees this keyword it switches to twin refinement.

The program will find twin operators using tolerance level 0.001 and then using *Rmerge* values for each operator will make decision if the operator can be twin operator. After the first cycle of refinement it will remove all twin domains with fraction less than 5% making sure that the remaining operators together with the crystallographic ones form a group.

twin FilterLevel <value>

defines level at which small twin domains are removed. If twin fraction is small than the specified *value* then this domain is removed. After removing small domains the program makes sure that twin and crystal symmetry together form a group. Default value is 0.05

If *twin* keyword is defined then intensities can be used for refinement . See [labin](#) keyword above.

Other keywords

twin operator < operator > # It is not active yet
twin domain fraction < value > # It is not active yet
twin tolerance < value > # Active from the version 5.6.0051

This controls tolerance level that allowed symmetry constraints on cell parameters. Default value is 0.02. For merohedral cases this value is always 0, i.e. symmetry of lattice is higher. For pseudo merohedral cases higher tolerance may cause resolution stretching problem. In general if symmetry of lattice is very approximate then it is better to consider them as non-merohedral twinning cases. The reason is that in these cases in some directions overlap of reflections may overlap more than in other directions. Moreover at higher resolutions spots may become resolved.

twin FilterLevel < value > # Active from the version 5.6.040

Smallest allowed twin fraction. Default value is 0.05

twin rmergelevel <value> # Active from the version 5.6.0051

Larges Rmerge allowed. If Rmerge corresponding potential twin operator is larger than this value then this operator is discarded from further consideration. Default value is 0.5. If one wishes to check if reindexing is needed then this value could be increased to 0.6 or even 0.8. In these cases all potential twin operators will be accepted for further consideration and twin fractions will be calculated. If reindexing is needed then the first domain will have smaller estimated twin fraction than others.

Simultaneous SAD experimental phasing and refinement

The SAD target function performs refinement using the experimental phase information directly (using the SAD data and anomalous scatterers positions). To use SAD function appropriate labels (F+, F- and corresponding SIGF+, SIGF-) should be defined using [labin](#) keyword. Furthermore, at least one atom must have non-zero f". The anomalous formfactors can be defined by the keywords:

anomalous formfactor [Name] [f'] [f'']

It will modify form factor of the given atom

anomalous wavelength [wavelength]

If the wavelength is given then form factors (f' and f'' of all atoms) will be calculated using crossec. If for some element explicit formfactors are given then they will be used, for other atoms formfactors will be calculated. If wavelength is not given and mtz file has the wavelength then it will be used. If wavelength is not given and mtz does not have wavelength f'=0 and f''=0 will be used.

Refmac can also perform SAD phasing and refinement of substructure only. FB and PHIB output columns are generated for this case. No special input keywords are required, if Refmac sees substructure only in the pdb then it will switch. *refi substructure* keyword can be used to force the substructure phasing and refinement if needed from some reason.

Example:

labin F+=f_label+ SIGF+=sigf_label+ F-=f_label- SIGF-=sigf_label-
anom form SE -8.3 3.9

Occupancies of anomalous scatterers are refined by default if SAD target is used. Their refinement can be disabled by

refine orefine no

Simultaneous SIRAS experimental phasing and refinement (from 5.6.0059)

The SIRAS target function performs refinement using all native and derivative F⁺, F⁻ data simultaneously. The FN, F⁺ and F⁻ along with the SIGN, SIGF⁺ and SIGF⁻ labels need to be defined by *labin* in order to use the target. Furthermore, at least one atom must have non-zero f'' and heavy atom substructure of the derivative compound needs to be specified by XYZIN2. Isomorphism between the native and derivative is assumed at the current implementation. Different models of the native and derivative and their simultaneous refinement with restraints between them is planned for the future.

Example:

```
... XYZIN native.pdb XYZIN2 derivative_substructure.pdb ...
labin FN=f_native SIGFN=sigf_label+ F+=f_derivative+ SIGF+=sigf_derivative+ F-
=f_derivative- SIGF-=sigf_derivative-
anom form S -0.5 0.6
anom form SE -8.3 3.9
```

SIRAS phasing and refinement of the substructure is supported and performed automatically if only one substructure file (XYZIN) is inputted.

Weighting of Xray and geometry

weight auto | matrix [value]

If auto option has been given then the program will try to adjust weight parameter between X-ray and geometry. Current criterion is very simple: The program makes sure that rmsd bond from ideal values is between 0.015 and 0.025. If matrix value has been specified then it may be necessary to run the program several times and control rmsd for geometric parameters (e.g. bond lengths, angles)

Map calculation (mapc)

mapc free | coefs | shar

free : Subkeyword that controls behaviour of free reflections for map calculation Possible values are: *include* - free reflections are included, *exclude* - free reflections are excluded or *restore* - free reflections and missing reflections are estimated (see below [map coefficients](#)). Default value is *restore*

coef: Subkeyword for user defined map coefficient calculation. Values of the subkeyword are: *n,m* It will force the program to produce map coefficient nFo-mFc. mtz labels for these coefficients are F_{user}, PHI_{user}. Normal 2fo-fc and fo-fc type map coefficients are always calculated.

shar: subkeyword for map sharpening. Value of this subkeyword is a bvalue that is used for all map coefficients. Output coefficients are modified using: $F_{coef} * \exp(bvalue * |s|^{2/4})$

Map coefficients

When calculating map coefficients REFMAC by default tries to restore missing reflections. Statistical basis of this is that expected value of unknown structure factors for missing reflections are better approximated using D_{Fc} than with 0 values. Of course to restore missing reflections accurately one needs full integration over all unknown parameters with their appropriate probability distributions that is not feasible in the current version. However approximate integration gives the value D_{Fc} . Current approach is trade off between bias introduced by restoring and noise level introduced by using zero values for missing reflections. Note that since for restored structure factors D_{Fc} is used and D reflects error in parameters, the level of bias is reduced substantially. If one wants not to include these reflections in the map calculation it can be done as follows: 1) Do not generate list of all unmeasured reflections; 2) use the instruction:

mapcalculate free exclude or mapcalculate free include

Another way of not restoring unobserved reflections is to use sigmas when calculating map. I.e. use only those reflection for map calculation for which $\sigma > 0.0$. It can be done in *fft* or *ftbig* of *ccp4* suite but not guaranteed to work in other software. Unobserved reflections and their effect in map is a huge and underestimated problem that needs to be treated accurately.

Anomalous and difference anomalous maps can be generated. It is generated if SAD refinement is performed. If SAD refinement is performed then the following keyword must be used to generate (weighted) coefficients for these maps

anom maponly

Bulk solvent

There are two options for bulk solvent:

1) Babinet's bulk solvent. It is activated using
scale type bulk

keyword. The full keyword (or set of keywords) are:

scale type bulk
scale lsscale fixbulk bvalue <value> scale <scale>

The first keywords signals the program that Babinet's bulk solvent should be used. It has effect that the total calculated structure factors are multiplied by the factor $(1 - k_{Bulk} \exp(-B_{Bulk} |s|^{2/4}))$. k_{Bulk} and B_{Bulk} are in general adjustable parameters. The second keyword instructs the program to fix either k_{Bulk} or B_{Bulk} or both.

The second form of bulk solvent is mask based bulk solvent. It is activated (by default this form of bulk solvent is always calculated) using the following keyword:

solvent yes/no # Either use or not use mask based bulk solvent

solvent vdwprobe <value> ionprobe <value> rashrink <value>
solvent exclude DUM # Exclude atoms with name DUM from solvent mask calculation

These are parameters of the mask based bulk solvent. *vdwprobe* is probe radius around vdw type of atoms, *ionprobe* is probe radius around ion/polar/hydrogen bond capable atoms. *rshrink* is shrinkage radius after calculating mask with defined parameters. Default values are 1.2, 0.8, 0.8

Algorithm for mask based bulk solvent

- 1) Increase radius of vdw atoms by *vdwprobe* and ion atoms by *ionprobe*.
- 2) Put zero inside sphere's novym radiusom
- 3) Reduce protein mask by *rshrink*. If points inside the mask defined in the step 2) are closer than *rshrink* angstrom to outside then define this point as being outside.
- 4) calculate structure factors and add them to the protein structure factors $F_{\text{protein}} + k_{\text{mask}} \exp(-B_{\text{mask}} |s|^2/4)$. k_{mask} and B_{mask} are adjustable parameters.

solvent optimise # This option is available from 5.6.0078

Optimise solvent parameters using grid search.

Occupancy refinement (version 5.6.0037)

occupancy group id <number> chain <chain1> ... <chainn> residue <number> atom <name> alt <code>
or
occupancy group id <number> chain <chain1> ... <chainn> residue from <number> to <number> atom <name> alt <code>
occupancy group alts complete/incomplete <id1> ... <id2>
occupancy refine ncycle <number>

Where *id* defines occupancy group id. This may be referenced by the command *occupancy group alts <id>*.

Example:

<i>occupancy group id 1 chain A</i>	# chain A belongs to occupancy
<i>group 1</i>	
<i>occupancy group id 1 chain B residues from 200 to 500</i>	# all residues between residues
200 and 500 of chain A belong to the group with id 1	
<i>occupancy group id 2 chain C residue 250 alt A</i>	# all atoms of residue 250 of
chain C with alt code A belong to group 2	
<i>occupancy group id 2 chain D residue 250 atom OW</i>	# atom OW of the residue 250
of chain d belong to the group 3	
<i>occupancy group alts complete 1 2</i>	# occupancy group 1 and two
are mutually exclusive. Moreover sum of their occupancise should be equal to 1	
(subkeyword <i>complete</i>)	
<i>occupancy group alts incomplete 1 3</i>	# sum of occupancies of
occupancy group 1 and 3 must be less than 1 (and more than 0 obviously)	
<i>occupancy refine ncycle <number></i>	
<i>occupancy refine</i>	# refine occupancies.

Refinement of occupancy parameters will be carried out at every *ncycle*-th cycle. Default is 1, i.e. occupancy and restrained refinements are carried out one after another at every cycle.

If *occupancy refine* has not been defined then group definitions will only be used in making decision about non-bonded contacts. Atoms belonging to the alternative groups (e.g. group 1 and 2) do not see each other and therefore there is no non-bonding (vdw or other) interaction between them. Of course user can use [external restrain](#) keyword to enforce bonds between mutually exclusive atoms.

Anisotropic Refinement

refinement brefinement anisotropic | mixed | isotropic | overall

instruction activates B value refinement option. There are three options: overall, isotropic refinement (it is default), mixed and anisotropic refinement.

If mixed refinement is requested then by default the program will look at the input pdb file and will refine anisotropically only those atoms that have ANISOU card. Starting from the version 5.6.0082 the following instructions can also be used to define atoms or regions of atoms anisotropically:

brefine mixed anisou residues from <resnumber> <chain> to <resnumber> <chain> atoms <list of atoms>

All atoms that are in the list of atoms for residue range will be defined anisotropically. For example

brefine mixed anisou residues from 10 A to 100 A atoms CA # all CA atoms of residues between 10A and 100A will be anisotropic
brefine mixed anisou residues from 100 A to 150 A atoms C # all atoms with atom names starting with C for all residues between 100A and 150A will be anisotropic*
brefine mixed anisou residues from 100 A to 200 A # all atoms of the residues between 100A and 200A will be anisotropic

or

brefine mixed anisou residue <rsnumber> <chain> atoms <list of atoms>

Atoms with the names from the list of atoms of the given residue will be anisotropic. For example:

*brefine mixed aniso residue 100 A atoms FE S C**

Atoms FE, S and all atoms with the names starting C of the residue 100A will be anisotropic

or

brefine mixed anisou atoms <list of atoms>

All atoms with the names that coincide with one of the names from the list of atoms will be anisotropic. For example

brefine mixed anisou atoms FE CA CO

all atoms in all residues that have names FE or CA or CO will be anisotropic

For keywords controlling [restraints on Bvalues](#) see below.

NCS constrained refinement

ncsconstraints

It gives a signal to the program that ncs constraints should be used.

ncsconstraints matrix <value>

Where value is 12 numbers. First nine of them are elements of the matrix and the last three elements correspond to the translation.

Example:

```
ncscons matrix -  
-0.11118500    0.74538100    0.65730399    -  
-0.74538398   -0.50000000    0.44092199    -  
0.65730399   -0.44092101    0.61118501    -  
0.00000000    0.00000000    0.00000000
```

ncsconstraints euler <value>

where value consists of 12 number. First three are Euler angles - alpha, beta, gamma and the last three correspond to the translation

Example:

```
ncsconst euler 33.853855    52.324917    -146.14619    0.0000000    0.0000000  
0.0000000
```

ncsconstraints polar <value>

where value is a vector of 6 element first three of which are polar angles - psi, phi, chi and the last three are translation.

Example:

```
ncsconst polar 149.39453    179.99998    120.00000    0.0000000    0.0000000  
0.0000000
```

If no ncs operators are defined then those from the pdb header (MTRIX) are used. If at least one operator is defined then operators from the pdb header are ignored and the number of ncs operators in the list of keyword defines strict ncs operators. Current version does not allow refinement of ncs operators.

If ncs constraints operators are defined then the program will assume that all coordinates in the input pdb file must be multiplied by these operators to generate asymmetric unit.

Notes:

- 1) Strict NCS assumes that B values of equivalent atoms are identical.
- 2) If TLS is refined then it is assumed that equivalent rigid groups have corresponding TLS (after transformation by NCS symmetry)
- 3) Current version does not calculate anti-bumping restraints for strict NCS copies
- 4) Current version does not check if the ncs operators with some addition from the space group operators (that are the symmetry of the molecule) form a group. Future version will attempt to check this.
- 5) Mixture of different ncs constraints are not available yet.
- 6) Averaging must be done outside reftmac5, it does not do averaging yet.

Keywords that control geometry

Using segment id

Keyword
make segid yes

Effect on other instructions:

Chain names involved in all instruction will be interpreted as segment id. For example NCS restraints could be:

ncsr nchains 4 chains AAss BAss TOss Yass nspans 1 1 100 1

The program will interpret AAss, BAss, TOss, Yass as segment ids. This instruction also affects records in the pdb header

External and user defined restraints

External restraints

Keywords controlling general behaviour

external UndefinedAtoms ignore

If this keyword is defined then the program reports the list of restraints for which some atoms are absent in the input file of coordinates. These restraints will be ignored but the program will continue its work. This keyword acts only after it is defined. Influence of this keyword can be stopped using the following keyword:

external UndefinedAtoms STOP

Current version of the program allows several types of external.

Distance restraints

*external distance first chain [ch] residue [res] insertion [ins] - atom [n] [alcode [a]]
 second chain [ch] residue [res] insertion [ins]- atom [n] [alcode [a]] value [v] sigma [s]
 [symm y/n] type [value]*

external weight scale [value]
external weight gmwt [value]
external weight sgm n [value]
external weight sgm x [value]
external dmax [value]

This instruction will force to put restraints between defined atoms. Subkeywords insertion, altcode and symm are optional. If there is more than one restraint (including normal covalent bond restraint) then only the last one will be used.

Examples:

1) Restraint between atoms in the same asymmetric unit (without symmetry)

exte dist first chain A resi 2 atom CA second chain A resi 5 atom CA value 4.0 sigma 0.02

2) Restraint between atoms symmetry related atoms

exte dist first chain A resi 2 atom CA seco chain A resi 5 atom CA valu 4.0 sigm 0.02 symm Y

In this case all symmetry operators will be tried and that that brings these two atoms to the closest contact will be used for the restraint.

Weights on the distance restraints can be controlled using *external weight* subkweyords. Final sigmas are calculated using the following formula

type subkeywords defines the type of external restraints:

- 0 - bond type restraints, override existing distance restraint,
- 1 - use this restraints in addition to the existing bond restraint.
- 2 - external long(er) range distance restraints.

Weights on these restraints can be controlled using *external weight* subkeywords and final weights are calculated using Geman-Maclure type robust estimator functions.

$$\sigma_{final} = \max(sgm n, \min(sgmax, input_value)) / scale$$

Note that this keyword becomes active when it is defined and is applied only on type 2 external distance restraints (type 2 is default for external distance restraints).

gmwt subkeywords controls parameter of the function Geman-McLure.

extrnal dmax [value]

tells the program that ignore all distance restraints for which target value is larger than dmax. This instruction does not have backward effect.

For example if the following instructions are given then the first restraint will be used and the second restraint will be ignored

external distance first chain A residue 5 atom CA second chain A residue 100 atom CA value 4.5 sigma 0.02
external dmax 4.0

*external distance first chain A resi 2 atom CA second chain A resi 50 atom CA value 4.5
sigma 0.02*

Angle restraints

*external angle first chain [ch] residue [res] insertion [ins] -
atom [n] [altecode [a]] next chain [ch] residue [res] insertion [ins] atom [n] [altecode [a]]
] [symm y/n] next chain [ch] residue [res] insertion [ins]-
atom [n] [altecode [a]]] [symm y/n] value [v] sigma [s] [symm y/n]*

The three atoms are defined and the angle formed between these three atoms is restrained to the *value* defined by value with the sigma defined by *sigma* subkeyword.

Torsion angle restraints

*external torsion first chain [ch] residue [res] insertion [ins] atom [n] [altecode [a]] next
chain [ch] residue [res] insertion [ins] atom [n] [altecode [a]]] [symm y/n] next chain [ch]
residue [res] insertion [ins] atom [n] [altecode [a]]] next chain [ch] residue [res] insertion
[ins] atom [n] [altecode [a]]]
[symm y/n] value <v> sigma <s> period> <p>*

Example

*external torsion first chain A residue 220 atom C next chain A residue 220 atom CA next
chain A residue 220 atom C next chain A residue 221 atom N value -60 sigma 10 period 1*

Similar type of keywords could be used for planar and chiral volume restraints also. When chiral volume restraint is used care should be taken to define the sign of the volume correctly.

Interval restraints

*external interval first chain [ch] residue [res] insertion [ins] - atom [n] [altecode [a]]
second chain [ch] residue [res] insertion [ins]- atom [n] [altecode [a]]] dmin [v] dmax [v]
smin [s] smax [s] [symm y/n]*

This keyword defines interval restraints. If the distance between specified atoms is less than the value defined by *dmin* then quadratic antibumping restraints with sigma *smin* is used. If the calculated distance is more than *dmax* then quadratic attracting term is used with the sigma equal to the value defined by *smax*.

External restraints could be saved in a file and used in refinement as:

*refmac [all usual things] << eof
@file_external_restraints
all other instructions
eof*

Harmonic restraints

Under the pressure from various users I have added harmonic restraints. If you use these restraints then atoms will be restrained to their current position and movement from those positions will be slower than for other atoms. Keywords for harmonic restraints:

external harmonic chain [ch] residue [res] insertion [ins] atom [n] [altcode [a]] [sigma [value]]

or

external harmonic residues from [residue_number] [chain_name] to [residue_number] [chain_name] [atom <atname>] sigma [value] sigma 0.1

For example:

external harmonic chain A residue 225 atom CA

will put harmonic restraint on this atom

external harmonic residues from 225 A to 250 A sigma 0.02

will put harmonic restraints on all of the atoms of the residues between 225A to 250A. The weight will be calculated using $1.0/\sigma^2$

external harmonic residues from 225 A to 250 A atom CA sigma 0.02

will put harmonic restraints on CA atoms of the residues from 225A to 250A.

Harmonic distance restraints (Ridge regression)

Keywords:

ridge distance sigma <value> # Default 0.1
ridge distance dmax <value> # Default 4.2
ridge atoms <sigma>
ridge bvalue <sigma>

If *ridge distance sigma* (default 0.1A) instruction has been given then the program will add the following function to the target function:

$$\sum_{|d| < d_{max}} \frac{1}{\sigma^2} (|d| - |d_{current}|)^2$$

Where d is distance between atoms $d_{current}$ is current distance between atoms. The program updates at every cycle $d_{current}$ to the current distance between atoms. If this instruction is defined then the program will calculate the list of all pairs of atoms for which distance between is less than d_{max} . Default value is 4.2. Note that harmonic distance restraints will be applied together with non-bonded antibumping restraints.

If *ridge atoms* instructions is defined then the program adds the following function (it is same as [harmonic](#) restraints applied to all atoms)

$$\sum_{atoms} \frac{1}{\sigma^2} |\Delta_x|^2$$

Where Δ_x is the shifts to be applied to the atomic positions

If ridge bvalue instruction is defined then the program adds the following term:

$$\sum_{atoms} \frac{1}{\sigma^2} B_{iso}^2$$

Torsion angle restraints(from dictionary)

restr tors include | exclude

Include or exclude given torsion angle in the restraint calculation. Both subkeywords have the following syntax

resi | group | link [name] name [name] value [value] sigm [value] period [value]

For example

restr tors include resi VAL name chi1 value 60 sigma 2.0 period 3

This instruction will force chi1 torsion angle of all residues VAL to be restrained to 60 with period three.

Similarly this instruction can be applied to a group of residues (e.g. peptide, pyranose, DNA/RNA) or links between monomers (e.g. TRANS, ALPHA1-3 links). This restraint will be applied to all residues with name PHE.

An example how to exclude some torsion angles from restraints

restr torsion exclude residue PRO name chi1

If instruction is given up to the name of the monomer then all torsion angles in this monomer that have name starting with "var" will be restrained. For example:

restraint torsion include residue BLA

These instructions should be used with care. One should make sure that values used (in dictionary or defined by user in instructions) are valid and make chemical sense.

VDW restraints

There are several sets of subkeywords that control various behaviours of antibumping restraints

1)
vdwrestraints <weight>

Weight controls overall vdw repulsions (it includes ionic interactions also). Large weight means strong antibumping repulsion.

Or more specifically

2)
vdwrestraints overall <weights> sigma VDW | HBOND | METAL | TORS | DUMM <value>
increment TORSion | ADHB | AHHB | DUMM <value>

Exclude repulsions between specified chains:

The following subkeyword combinations allow user to define atoms or atom pairs to be removed from anti-bumping restraints.

3)
vdwrestraints exclude between chains <chains>

For example

vdwrestraints exclude between chains A B C D

Then antibumping restraints between all these chains will be removed.

4)
The following keyword tells the program to exclude antibumping restraints between two atoms

vdwrestraints exclude between atoms first atom <name> alt <alt> residue <residue> ins <ins> chain <chain> second atom <name> alt <alt> residue <residue> ins <ins> chain <chain>

For example

vdwrestraints exclude between atoms first atom O resiude 205 chain W second atom OE1 alt A residue 54 chain

6)
Excluding antibumping between residues

vdwrestraints exclude between residues first residue <residue> ins <ins> chain <chain> second residue <residue> ins <ins> chain <chain>

For example:

vdwrestraints exclude between residues first resiudue 205 chain W second residue 54 chain

7)
The following keyword is to exclude antibumping restraints for residues or atoms

vdwr exclude residue <res> ins <ins> chain <chain>

vdwr exclude residues from <res> ins <ins> chain <chain> to <res> ins <ins> chain <chain>

vdwr exclude atom atom <name> alt <alt> <res> ins <ins> chain <chain>

Basepair restraints and other pairwise restraints

Basepair restraints information is in \$CLIBD_MON/dnarna_basepairs.txt

Users can give an additional file with their basepair restraints. It can be done in two way:

- 1) By adding basepairs <basepair_file_name> in the command line
- 2) By using the followin keyword

restraints fbasepairs <file_name>

The following commands control basepair restraints

Automatic basepair restraints

restratints bp|pairs|basepairs auto

In this case basepairs will be found automatically and restrained.

or by defining range of residues where basepair restraints should be applied:

restraints bp|pairs|basepairs between residues from chain <chain> residue <residue> to chain <chain> residue <residue> and from chain <chain> residue <residue> to chain <chain> residue <residue> [antiparallel]

Examples:

restraints bp auto

restraints basepairs between residues from chain A residue 1 to chain A residue 12 and from chain B residue 24 to chain B residue 13 antiparallel

Note: basepair restraints are general. In principle any pairs of residues can restrained as soon as these pairs are defined in the file (dictionary)

Here is an extract from the "standard" basepair file:

```
#
#  nucleic acid base pairs. Some of the base pairs may not be standard. But they appear in
some structures
monomers A T label A:T
bond atom N6 A atom O4 T value 2.94 sigma 0.15
bond atom N1 A atom N3 T value 2.84 sigma 0.1
chiral atom N1 A atom C2 A atom C6 A atom N3 T value 0.0 sigma 0.5
chiral atom N3 T atom C2 T atom C4 T atom N1 A value 0.0 sigma 0.5
torsion atom C2 A atom N1 A atom N3 T atom C4 T value 180.0 sigma 10.0
```

Sugar puckers and other additional restraints

Bvalue Keywrods

Note that Bvalue restraints are applied to bonded as well as nonbonded atom pairs. Weights depend on the nature of atom pairs (bonded, angle related, torsion angle related or otherwise).

1)
bfactor set <value>

Set initial B values to this predefined value

2)
bfactor <scale> *kldivergence* <sigma1> <sigma2> <sigma3> <sigma4>

Overall scale and sigmas for Bvalue restraints.

sigmas are for bond, angle, torsion angle related atoms. sigma4 is for all other atom pairs. First three sigmas are constant. Sigma4 is used to design distance dependent sigmas

$\sigma = \sigma_4 \quad d \leq 3$
 $\sigma = 3 * \sigma_4 / d \text{ if } d > 3$

Where d is interatomic distance

Once sigma is available required weight is calculated using:

$w = (\text{scale}/\sigma)^2$

Restraints are based on Kullback-Liebler divergence and has a form form isotropic B values:

$w * (B_1 - B_2)^2 / (B_1 * B_2)$

Default values of the sigmas are: 0.1, 0.15, 0.2, 0.4. If one wants to change weights on B values it is recommended to change overall scale first. If it still does not give satisfactory results then other values could also be changed.

3)
sphere <sigma>

Restraints on sphericity. Smaller value means that atoms will be more spherical. Default value is 5.0

4)
rbond <sigma1> <sigma2> <sigma3> <sigma4>

Restraints on rigid bonds. Sigmas are for different type of atom pairs as described [above](#). Default values are 0.05, 0.07, 0.1, 0.2.

Exclude from refinement

refinement exclude all from [residue] [chain] to [residue] [chain]

All atoms between given residues will be excluded from refinement (restraints, structure factor and gradient calculations), but they will be used for mask calculation.

NCS restraints

Old instructions (for backward compatibility). One instruction per ncs group should be given:

ncsr nchains [nchains] chains [chain1] ... [chain_nchains] .. nspans [n1] [n11] [n12] .. [nn11] [nn12] [n4]

nchains - number of chains involved in this ncs chain - chains involved in this ncs n1 - number of spans n11,n12 - Start and end for the current ncs span n4 - weighting options

New instructions (a little bit more flexible and useful for complex molecules):

Definition of ncs groups:

ncsr group [id] nchain [chain] chains [chain1] ... [chain_nchain] residue [res1] [res2] ... residue [res1] [res2] sigx [value] sigb [value]

id - ncs id. It is used to group ncs related chains together. nchain - number of chains involved in this ncs. It defines the number of ncs matrices need to be calculated. residue - defines ncs restraint spans sigx - sigma on positional parameters sigb - sigma on atomic displacement parameters

Each ncsr id can have only one sigma (sigx) on positional and one on ADPS (sigb)

Example:

```
ncsr group 1 nchains 3 chains A B C residue_range 1 100 residue_range 201 300
residue_range 401 500
ncsr group 1 nchains 3 chains D E F residue_range 10 50
ncsr group 1 sigx 0.02
ncsr group 1 sigb 1.0
```

Auto NCS and local NCS restraints

ncsr

If the program sees *ncsr* are no chain definition then it will switch to automatic definition of ncs related molecules. To do this the program will do alignment and using the results of this alignment will find correspondence between atoms. Alignment will work for amino acid and DNA/RNA chains

ncsr local/global

If the *local* is defined then the program will restraint corresponding interatomic distances in two (or more) ncs related molecules. The formula for restraints is based on Grman-McLure

robust M-estimator functions

$$x^2/(1+w x^2)$$

Where $x = (d1-d2)/\sigma$, $d1$ and $d2$ are corresponding inter atomic distances, σ is the standard deviation and w is a parameter. Default value for this parameter is $1.0e-4$. It can be controlled (see below).

ncsr align level <value> iterate <Y/N> rmslevel <value>

These keywords control the result of alignment. *level* defines a alignment level. If the alignment score is more than this value then sequences are considered aligned. Score is calculated using the formula:

$$n_aligned / (\min(nalign_length1, nalign_length2))$$

where $n_aligned$ is the number of residues that have been aligned and they are identical $nalign_length1$ and $nalign_length2$ is $n_last_aligned - n_first_aligned$, difference between the first and last serial numbers of aligned residues for the first and second sequences respectively.

The keyword *iterate* indicates if iterative alignment is required. If the value is Y then the program will remove aligned residue pairs from alignment (more precisely the matrix elements corresponding to the aligned residues will be set to zero) and further alignment will be carried out. It is needed to be used if there are gene duplication, triplication etc. Example of such cases can be found in the pdb - 2vtu.

The keyword *rmslevel* controls level of acceptance of alignment below certain RMS value. RMS value is calculated as an average of local 5 residue rms of aligned residues. This ensures that if there are some conformational changes then the program do not reject alignment unnecessarily.

ncsr dmax <value>
ncsr diffmax <value>

If local ncs restraints are used then all atom pairs with interatomic distance less than *dmax* (default is 4.2A) are considered in generating ncs restraints. If difference between corresponding ncs related distances is less than *diffmax* (default is 1.0A) then this pair is included in restraints.

ncsr neighbours <include/exclude>

This keyword indicates if the neighbours of ncs related molecules should be included in ncs definitions.

ncsr gmparameter <value>

This keyword controls Geman-McLuire function. Default value is $1.0E-4$. A rule of thumb in defining this value: If one wants to halve the contribution to the gradient for pairs of distances for which difference is more than $a \cdot \sigma$ then one should define this value equal to $\sqrt{(\sqrt{2}-1)}/a$

Rigid body

If you do not define rigid groups then the program takes each chain (if available then segment) as a rigid group. I.e. if you want to refine each chain (segment) as a separate rigid group then the following keyword is sufficient. Note that even if you do not use segment but they are defined in the input PDB then these segments will be used as rigid groups.

mode rigid

TLS refinement

If you do not define TLS groups then the program takes each chain (if available then segment) will be as a TLS group. I.e. if you want to refine each chain (segment) as a separate TLS group then the following keyword is sufficient. Note that even if you do not use segment but they are defined in the input PDB then these segments will be used as rigid groups.

refi tlsc

If the keyword

tlsout addu

has been specified then the output file will contain ANISOU card for atoms involved in TLS group definitions. The values for ANISOU are contribution from TLS with added residual B values. In this case B value contain sum of residual and contribution from TLS .

NB: This keyword should be used for visualisation and analysis purposes only. The resultant output coordinate file should not be used as an input file for the next stage of refinement. In this case behaviour of refinement could be unpredictable.

tlsd waters add/exclude

By default current version of refmac (5.5 and further) includes waters close to chains to the same tls group as those chains. However this behaviour can be changed if one uses *tls waters exclude*. In this case waters will not be included in TLS definitions.

Controlling output PDB

Copy remarks from the input file:

pdbout copy remarkrs <number> <number> ..

gives signal to the program to copy remarks with specified number from the input file to the output file.

Note: Since REMARK 3 is refinement info it is not be copied.

Example: if the following instruction is specified

pdbout copy remarks 200

then the program will copy from the input file remark 200 (these remarks contain information about data collection)

LIBCHECK keywords

Dictionary from smile strings.

For full description of libcheck see [Alexei Vagin's libcheck page](#)

For smile string formats and syntax see: daylight site

To create a dictionary entry from SMILE string you need to have a file that contains SMILE for your ligand. One file should contain one ligand only. Then a dictionary entry can be created using libcheck:

libcheck file_smile [file] mon [give a reasonable name]

Then libcheck will create a dictionary entry. There should be one carriage return after the line libcheck and after the last instruction for the libcheck.

Dictionary from SYBIL MOL2 and SDF mol files

For mol2 see the [mol2 manual](#) and for sdf file see [sdf manual](#)

To create a dictionary entry from SYBIL MOL2 or SDF MOL file you need to have a file that contains ligand in one of these formats. Note that only 3D version of these formats can be used in this context. 2D version of these formats is equivalent to SMILE. That is why using SMILE strings in these cases seems to be more reasonable. Once you have file you can use libcheck to create a dictionary entry:

libcheck file_mol [MOL2 or SDF files] mon [mon name. It is optional]

The current version of libcheck creates dictionary from sdf v2000. V3000 has not been tested yet. If somebody wants to test please let me know.

Various protocols

In this sections protocols will be described using keywords. If you are using ccp4i then either there are appropriate options on the interface or you can create a file containing necessary keywords and then use "Developers option" to add this file. Then the keywords defined in this file will override the options defined in the ccp4i

External links

General information

[Main ccp4 wiki](#). A lot of useful info. It is dynamic and is becoming a powerful resource

To create a dictionary of ligands

[Dundee prodrq server](#) It can create dictionary for ligands for refinement in refmac.

[EBI MSD-CHEM server](#) You can search for ligand you are interested in. Then save the results in cif format. This file can be used in [libcheck](#) to create complete description of the ligand.

[Drugbank](#) is another server that can be used to get "ideal" structures.

REFERENCES

If you use REFMAC please cite to one of these papers!!!

1. "Refinement of Macromolecular Structures by the Maximum-Likelihood method" G.N. Murshudov, A.A.Vagin and E.J.Dodson, (1997) in Acta Cryst. **D53**, 240-255.
2. "Incorporation of Prior Phase Information Strengthen Maximum-Likelihood Structure Refinement" N.J.Pannu, G.N.Murshudov, E.J.Dodson and R.J.Read (1998) Acta Cryst. section **D54**, 1285-1294.
3. "Efficient anisotropic refinement of Macromolecular structures using FFT" G.N.Murshudov, A.Lebedev, A.A.Vagin, K.S.Wilson and E.J.Dodson (1999) Acta Cryst. section **D55**, 247-255.
4. "Use of TLS parameters to model anisotropic displacements in macromolecular refinement" M. Winn, M. Isupov and G.N.Murshudov (2000) Acta Cryst. 2001:D57 122-133
5. "Fisher's information matrix in maximum likelihood molecular refinement." Steiner R, Lebedev, A, Murshudov GN. Acta Cryst. 2003 D59: 2114-2124
6. "Macromolecular TLS refinement in REFMAC at moderate resolutions," Winn MD, Murshudov GN, Papiz MZ. Method in Enzymology, 2003:374 300-321
7. "Direct incorporation of experimental phase information in model refinement" Skubak P, Murshudov GN, Pannu NS. Acta Cryst. 2004 D60: 2196-2201
8. "REFMAC5 dictionary: organisation of prior chemical knowledge and guidelines for its use." Vagin, AA, Steiner, RS, Lebedev, AA, Potterton, L, McNicholas, S, Long, F and Murshudov, GN. Acta Cryst. 2004 D60: 2284-2295

[Garib Murshudov](#) Last modified: Sep 23 2010