

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.

Keywords that control Xray

[Labin: labels from mtz](#)

[Twin refinement](#)

[Simultaneous experimental phasing and refinement](#)

[Map coefficients](#)

[Weighting xray and geometry terms](#)

Geometry keywords

[Using segment id](#)

[External distance, angle etc restraints"](#)

[Harmonic restraints](#)

[Torsion angle restraints](#)

[VDW restraints](#)

[NCS restraints](#)

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 that are not active yet

twin operator < operator >

twin domain fraction < value >

twin tolerance < value >

twin FilterLevel < value >

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.

Occupancies of anomalous scatterers can be refined using

refine orefine anomalous

If this keyword has been specified then [labin](#) keyword must have F+ and F-

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

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

Current version allows only external distance restraints. These restraints could be written in a separate file and the used in the command line as psrestin.

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

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 in the same asymmetric unit (without symmetry)

*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. 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 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] sigma [value] sigma 0.1*

For example:

external harmonic chain A residue 225 atom CA will put harmominc restraint on this atom
external harmonic residues from 225 A to 250 A sigma 0.02

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

Torsion angle restraints

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 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.

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 nchain 3 chain A B C residue 1 100 residue 201 300 residue 401 500

ncsr group 1 nchain 3 chain D E F residue 10 50 ncsr group 1 sigx 0.02 ncsr group 1 sigb 1.0

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.

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 prodrgr server](#) It can create dictionary for ligands for refinement in reffmac.
[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.

REFERENCES

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

1. "Application of Maximum Likelihood Refinement" G. Murshudov, A.Vagin and E.Dodson, (1996) in the Refinement of Protein structures, Proceedings of Daresbury Study Weekend.
2. "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.
3. "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.
4. "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.
5. "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
6. "Fisher's information matrix in maximum likelihood molecular refinement." Steiner R, Lebedev, A, Murshudov GN. Acta Cryst. 2003 D59: 2114-2124
7. "Macromolecular TLS refinement in REFMAC at moderate resolutions," Winn MD, Murshudov GN, Papiz MZ. Method in Enzymology, 2003:374 300-321
8. "Direct incorporation of experimental phase information in model refinement" Skubak P, Murshudov GN, Pannu NS. Acta Cryst. 2004 D60: 2196-2201

9. "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: Mon Jun 23 21:20:24 BST 2008