



OpenEye
Scientific Software

AFITT-CL

Release 2.1.0

OpenEye Scientific Software, Inc.

July 08, 2011

CONTENTS

1	Front Matter	1
2	Introduction	3
2.1	Overview	3
3	Theory	5
3.1	Fragment Fitting	5
4	Installation and Platform Notes	7
4.1	Licenses	7
4.2	Installation	7
4.3	Uninstallation	9
5	FLYNN Usage	11
5.1	Types of input ligands	11
5.2	Stereochemistry Enumeration	12
5.3	Refinement Dictionaries	12
5.4	Note about REFMAC5 refinement dictionaries	12
5.5	Note on the MMFF94 versus MMFF94s forcefields	13
5.6	Note on FLYNN and writing CIF Files	14
5.7	Command Line Interface	14
5.8	MTZ File Parameters	18
5.9	Advanced Parameters	19
5.10	Fragment Fitting Parameters	20
5.11	3D Construction Parameters and Torsion Driving Parameters	20
5.12	Example Commands	20
5.13	Results on The Gold Test Set	22
6	WRITEDICT Usage	25
6.1	Command Line Interface	25
6.2	Example Commands	27
7	ALIGNGRID Usage	29
7.1	Command Line Interface	29
7.2	MTZ File Parameters	30
7.3	Example Commands	30
8	Integration with Coot	31
8.1	Using FLYNN	32
8.2	FLYNN Advanced Options	32

8.3	Using WriteDict	33
9	Release Notes	37
9.1	AFITT-CL 2.1.0	37
9.2	AFITT-CL 2.0.1	37
9.3	FLYNN 2.0.0	38
	Bibliography	39
	Index	41

FRONT MATTER

Copyright 1997-2011 OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific Software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of Accelrys, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrodinger, Inc.

Python is a trademark of the Python Software Foundation. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.

INTRODUCTION

FLYNN is a program to fit small molecules in electron (or other) density. As well as high-quality ligand fitting, the flynn package includes a facility (WRITEDICT) to write high-quality refinement dictionaries suitable for refinement with such programs as REFMAC5 [Murshudov-1997][Vagin-2004] as well as a tool that can rotate crystallographic maps from one protein's reference frame to another (ALIGNGRID).

2.1 Overview

Fitting small molecules to electron density is a challenging problem. Small molecules with few rotatable bonds are amenable to hand placement in density. Add a few rotatable bonds, however, and the problem becomes factorially more complicated. Furthermore, unlike protein modeling, many small molecules are relatively unique entities whose conformation cannot be predicted from previous homologies.

FLYNN uses a high-quality conformational analysis and a fast rigid overlay to position ligands in density in conjunction with a combined forcefield to further position these conformations while, at the same time, minimizing ligand strain. FLYNN can automatically locate ligand density and automatically fit fragment cocktails, or, failing that, FLYNN can use a supplied bounding box in which to place the ligands of interest.

Ligands are fit to density, however the placements are scored and ranked based on the real space correlation coefficient (RSCC) and optionally protein based docking scores: Piecewise-Linear-Potential (PLP) and Chemscore.

FLYNN reads common map formats such as CCP4, CNX/CNS (XPLOR) and MTZ as well as OpenEye's grid formats and a plethora of small molecule, protein and connection table formats.

THEORY

Current methods of ligand fitting that are based on either topological analysis of electron density [Menendez-2003] global optimization of position and conformation of a ligand in a density blob [Diller-1999] interatomic distance matrix [Koch-1974] [Cascarano-1991] or on varying torsion dihedral angles of shape-matched ligand conformations [Oldfield-2001] are unable to prevent creation of high energy, sometimes even chemically unrealistic, ligand models. As a result, there are a number of PDB ligands with unlikely, very high-energy structures [Perola-2004]. For example, the PDB structure of an inhibitor of RNA polymerase in *Inhu* has significant repulsion between the two methylene groups.

FLYNN is composed of two main components, location of ligand density and ligand fitting. Location of density generally works well with clean density but there are times when ligand density is unclear or not well resolved. In the latter cases, the user should supply a bounding box or simply just input the ligand density by itself and use the density as is.

By default, FLYNN samples bioactive conformations [Bostrom-2002] [Bostrom-2003] of the input ligand, however, there are times when it might be desirable to input the ligand and use it as is (see section *Required Parameters* for more details).

Once the location of the ligand in the map and the conformations have been selected, FLYNN then adapts the initial conformations to the ligand density using a modern force-field, MMFF94 [Halgren-I-1996] [Halgren-II-1996] [Halgren-III-1996] [Halgren-IV-1996] [Halgren-V-1996] [Halgren-VI-1999] [Halgren-VII-1999]

The potential function being used to adapt the ligand is

$$V = V_{ff} + \lambda V_{shape}$$

where V_{ff} represents the internal energy of the ligand and V_{shape} is the overlap between the ligand and the electron density. λ is a mixing parameter that represents the degree to which the shape of the density dominates the combined potential during the current optimization step [Wlodek-2006].

The strain placed on the ligand is bounded while the function is optimized producing high-quality fits with low-strain ligand conformations.

3.1 Fragment Fitting

Fragments are fit taking the input fragment cocktail and, one at a time, fitting each fragment against each region of detected density. Once a fragment has been placed, it is further analyzed to ensure that all possible orientations of the fragment have been sampled. In poor density, several orientations may fit equally well. To break ties, FLYNN scores each pose with the following scores:

- o RSCC (real space correlation coefficient) This is a measure of fit to electron density (higher is better). [Jones-1991]
- o PLP Piecewise-linear potential (lower is better). [Verkivker-2000]

o Chemscore (lower is better). [Eldridge-1997]

The docking scores are not used to fit the molecule, they are only used to rank the output. Unless highly symmetric molecules are being input, the real space correlation coefficient (RSCC) is the preferred method of ranking results to density.

To use fragment mode, please add the “-fragment” option to the command line. In future versions of FLYNN, this will most likely become the default setting.

The output of the fragment fitting process is a file for each density region that includes the fragments fit to the region sorted from best-fit to worst fit. For example:

```
prompt> flynn -in fragments.smi -out 2IKO_cocktail.sdf ...
```

would result in the files:

```
2IKO_cocktail_blob001.sdf
2IKO_cocktail_blob002.sdf
2IKO_cocktail_blob003.sdf
```

one for each blob found. By default, the first molecule in the file is the best fit to the density (RSCC):

$$RSCC = \frac{\sum |p_{obs} - \langle p_{obs} \rangle| \sum |p_{calc} - \langle p_{calc} \rangle|}{(\sum |p_{obs} - \langle p_{obs} \rangle|^2 \sum |p_{calc} - \langle p_{calc} \rangle|^2)^{\frac{1}{2}}} \text{ [Jones-1991]}$$

where obs refers to the experimental electron density and calc refers to the calculated electron density sampled from grid points around the residue being scored. The experimental density is usually the sigma weighted difference map when loading an MTZ map. The calculated density is generated by simulating scattering of the ligand conformation with bfactors set to 20.

To sort using another measure, for instance, PLP, use the sort flag:

```
prompt> flynn -sortBy plp -in fragments.smi -out 2IKO_cocktail.sdf ...
```

Note: The *-sortBy* flag can be used in non-fragment mode as well.

INSTALLATION AND PLATFORM NOTES

4.1 Licenses

To run AFITT-CL and the associated utilities you will need to obtain a license file for AFITT-CL from OpenEye Scientific Software (business@eyesopen.com). The license file should be in a file pointed to by the **OE_LICENSE** environment variable.

4.2 Installation

4.2.1 General Installation

By default, all OpenEye applications are installed into a single distribution directory tree on the specified machine. The default location for this tree is platform specific and will be detailed below.

The root of the tree (i.e. the `openeye` directory) contains the following subdirectories:

- admin** This directory is intended to contain any administrative scripts and tools associated with the installed applications. Currently, this directory is simply a placeholder on all platforms except for Microsoft Windows, where it contains the uninstaller executables.
- arch** This directory contains the collection of platform specific subdirectories. Each subdirectory contains the actual installed executables and support libraries for the associated platform. In the platform specific subdirectory there will be a subdirectory for each application. Within that will be another subdirectory for each version of that application.
- bin** This directory contains a startup script for each application that has been installed. This script determines, at run-time, what the current platform is and then calls the appropriate executable in the `arch`. This script enables the easy co-existence of multiple platforms and versions of any OpenEye application in the same distribution tree.
- data** This directory contains all of the associated data for the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.
- docs** This directory contains all of the documentation associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

examples This directory contains all of the examples associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

The startup script discussed in the section on the `bin` directory above will have the same name as the installed executable with which it is associated. When the script is called, it will attempt to determine the current platform and run the appropriate executable if installed. If an appropriate executable cannot be found, the script will report that information, as well as a list of the currently installed platforms. The auto-detection can be overridden by setting one of two environment variables:

- **OE_ARCH** can be used to specify a colon separated list of compatible distributions for the current platform such as:

```
redhat-RHEL5-x64:redhat-RHEL4-x64
```

Specification of this environment variable overrides the auto-detection process, if it is present. If none of the compatible distributions listed are found, the script will fall back to the auto-detection process.

- **APPNAME_OE_ARCH** can be used to specify a colon separated list of compatible distributions for a specific application (as specified by changing the **APPNAME** text in the environment variable name) just like **OE_ARCH** as detailed above.

Specification of this environment variable overrides the **OE_ARCH** environment variable as well as the auto-detection process. If none of the compatible distributions listed are found, the script will fall back to the **OE_ARCH** list first and then to the auto-detection process.

Specifying this variable provides a simple way to customize the behavior for individual applications on non-standard platforms.

The startup script also supports a few commandline arguments including:

- | | |
|---------------------|--|
| -path | Specifying this argument will output the full path of the executable to be run. The executable will not be started if this argument is present. |
| -print_arch | Specifying this argument will output the details of the current platform as detected by the script as well as which platform-version of the executable is being run. The executable will be started if this argument is present. |
| -use_version | Specifying this argument followed by a specific version number allows the user to control which released version of the executable to run. |

4.2.2 Linux/Unix

Linux/Unix distributions are provided as a gzipped tarball of the distribution tree described above. Installation is performed by untarring the file in the desired location. Multiple distributions can be installed in the same location without any challenge.

To ensure that the installed applications can be called from the command line, be sure to add the full path of the `openeye/bin` subdirectory to the **PATH** environment variable. For instance, if the distribution was installed into `/usr/local/openeye`, the **PATH** environment variable should contain: `/usr/local/openeye/bin`.

4.2.3 Windows

Windows distributions are provided as a standard EXE installer. By default, all OpenEye applications will install into the `C:\OpenEye` directory.

An OpenEye group with an application specific subgroup will be added to the *Start* menu. The application specific subgroup will contain links to the documentation, the uninstaller, as well as to a Windows command shell which has

the appropriate **PATH** settings already defined to allow the user to simply type the executable name at the prompt without concern for where the executable is actually installed.

For GUI applications, a link to the application will be created on the desktop as well as in the application specific subgroup of the *Start* menu.

4.2.4 Mac OS X

Mac OS X distributions are provided as a standard *pkg* installer delivered as a *dmg* disk image. By default, all OpenEye applications will install into the `/Applications/OpenEye` directory.

To ensure that the installed applications can be called from the command line in the *Terminal*, be sure to add `/Applications/OpenEye/bin` to the **PATH** environment variable.

For GUI applications, an application bundle which can be clicked on to start, will be present in the `/Applications/OpenEye` directory. This bundle cannot be moved independent of the `OpenEye` directory. For instance, the entire `OpenEye` directory can be moved as one piece, but moving the application bundle or the contents of any of the subdirectories in the `OpenEye` directory may cause the application to not start. However, the bundle can still be dragged into the Dock and run from there without any problem.

4.3 Uninstallation

4.3.1 Linux/Unix

To uninstall a single distribution of a product the relevant subdirectories for that product and version simply need to be deleted from within the following directories:

arch In the `openeye/arch` directory is a platform specific subdirectory. Within this are directories for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled. For example, to delete or uninstall v1.0.0 of a product, delete the folder “<product_name>/1.0.0”.

data In the `openeye/data` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

docs In the `openeye/docs` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

examples In the `openeye/examples` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

4.3.2 Windows

Installation of an OpenEye product on Windows causes an OpenEye group with an application specific subgroup to be added to the *Start* menu. One of the items in the application specific subgroup is a link to the uninstaller. Clicking on the uninstaller initiates a wizard which guides the user through uninstallation.

For GUI applications, uninstallation also removes the desktop link to the application as well as in the application specific subgroup of the *Start* menu.

4.3.3 Mac OS X

To uninstall a single distribution of a GUI application simply drag the application from /Applications/OpenEye/bin to the Trash can.

To uninstall a single distribution of a command line application you will need to delete the executable/folder from /Applications/OpenEye/arch/osx-10.6-x64/. For example, to delete or uninstall v1.0.0 of a product, delete the folder “1.0.0” located in /Applications/OpenEye/arch/osx-10.6-x64/<product_name>.

Associated documentation, data and example files for a single distribution can be uninstalled by deleting the subdirectory/folder from within the /Applications/OpenEye/data, /Applications/OpenEye/docs and /Applications/OpenEye/examples directories.

FLYNN USAGE

At the minimum, FLYNN requires a density file and a the connection table of the ligand - usually derived from a supplied molecule file e.g. .pdb, .sdf, .mol2, .smi. FLYNN automatically searches the density for volumes of unmodeled density that are similar in size to the input ligand. Because of this it is highly recommended that a protein is also input to FLYNN. This makes the job of locating the ligand density less prone to false positives.

Based on analyzing several hundred protein/ligand combinations, the best way to use FLYNN for ligand fitting is described as follows (of course your mileage may vary):

- **Always use a protein:**

The protein is used to identify modeled density where the ligand cannot be placed. Without this, FLYNN will identify many candidate locations for placing the ligand that will need to be analyzed.

- **Use difference maps when appropriate:**

The reflection data should be refined without a previous ligand conformation or generated without the ligand present. While in many cases the same result will be generated, if the ligand is fit to data refined with a previous ligand conformation, FLYNN may be biased towards the density modified by the pre-existing ligand conformation.

- **use fragment mode for fitting fragments**

If you have an input cocktail, use the *-fragment* flag. This option simulataneously fits all the cocktails against density and sorts them by best fit.

5.1 Types of input ligands

Not all ligands can be successfully fit by FLYNN. Such types include large polypeptides or proteins, very flexible molecules, or simply molecules that contain atoms that are not present in the MMFF94s force-field.

Perhaps the most important property is rotatable bond count. Although FLYNN may be able to generate conformers for molecules with more than 20 rotatable bonds, the results of such an exercise would be dubious at best and should be taken with a grain of salt.

FLYNN will automatically check for unhandled atom types, but will gladly try and fit large molecules or proteins that contain many rotatable bonds usually with disappointing results.

Note: Dangers of PDB format for ligand input Many crystallography packages do not enforce writing out connection table or bond order information when outputting pdb files, or worse, strip them out. If bond distances are not correct FLYNN's attempt to automatically assign bond orders and atom types can be suspect. To facilitate this, FLYNN includes a *-precheck* flag that can be used to verify that the input ligand is identified correctly and for validation of the automatic bond order assignments.

5.2 Stereochemistry Enumeration

Compounds that contain unspecified or ambiguous definitions of stereochemistry will be preprocessed before conformational sampling to explicitly enumerate stereochemistry. Input molecules that have three dimensional coordinates inherently have stereochemistry specified, but SMILES or two dimensional SD files may have atoms (R/S) or bonds (E/Z) for which the stereochemistry is unknown or unspecified. In the case where the stereochemistry supplied is suspect, FLYNN provides an option that will completely enumerate all stereochemical centers.

The resulting ligands will be sorted based on their best fit to density regardless of the stereochemistry. That is, the best solutions will percolate to the top of the output. See *Input Options* below for more details.

5.3 Refinement Dictionaries

Both FLYNN and its companion program WRITEDICT can generate refinement dictionaries. A refinement dictionary is needed to maintain proper ligand geometry when using external reciprocal space refinement programs such as REFMAC5 or CNS X/PLOR.

Refinement dictionaries are a list of geometrical constraints encoding used chemical bonding and molecular conformation information. They are used by various refinement packages to describe standard geometries and constraints that are used during the refinement process. The quality of the post refinement ligand conformations are directly related to the quality of the constraints [Vagin-2004].

FLYNN comes bundled with a refinement dictionary writer named WRITEDICT. WRITEDICT uses the MMFF94 forcefield to derive geometrical constraints for the input ligands or ligand-protein complexes. The output dictionaries enforce, as closely as possible, the input ligand's geometries while allowing the refinement programs to modify the geometry as needed.

WRITEDICT also automatically detects covalent bonds in pdb files and inserts the appropriate PDB LINK records and covalent bond entries in the output refinement dictionary.

5.4 Note about REFMAC5 refinement dictionaries

When outputting REFMAC5 refinement dictionaries, WRITEDICT writes out a **.cif** file and a **.pdb** file. The **.pdb** file is written out for two reasons:

- **Inconsistent Hydrogen Naming**

Some applications write out hydrogen names incorrectly in ways that cause REFMAC5 or visualization programs like coot or WinCoot to be unable to associate the hydrogens in the refinement dictionary with hydrogens in the **.pdb** file. In the worst case, REFMAC5 will crash entirely.

When necessary, *WRITEDICT* renames and renumbers hydrogens, if they exist, so that REFMAC5 won't crash and so the **.cif** file is consistent with the **.pdb** file.

- **Covalent Bonds**

WRITEDICT detects covalent bonds and outputs proper LINK records in the **.pdb** file and proper covalent constraints in the **.cif** file. Without using both outputs, REFMAC5 will not detect covalent bonds during refinement.

- **Known residues**

When WRITEDICT detects a known residue, it may remap the atom names to be canonical with the known residue (this prevents REFMAC5 from failing to refine). These new atom names are saved in the **.pdb** file that WRITEDICT outputs.

- **Input Ligands with no residue information**

When WRITEDICT analyzes a ligand with no residue information, it assigns the whole ligand to a single residue (by default UNL). If this is not the desired outcome, the ligand must be put into **PDB** or **MOL2** format and the residues must be manually assigned.

It is highly recommended to use WRITEDICT's generated **.pdb** file in conjunction with the **.cif**.

5.4.1 Looking up known residues

WRITEDICT has an internal dictionary of known residues. By default, known residues names are retained when the graph of the known residue is exactly the same as the input residue. When this occurs, writedict can replace REFMAC5's dictionary with the MMFF94 generated dictionary. If the graphs do not match then WRITEDICT relabels the residue with a new residue name since a different dictionary needs to be created.

Options available for residue matching is as follows:

Option	Meaning
exact	Known residue and input residue graphs must match exactly.
substructure	Input residue may be a substructure of the known residue. Cannot be used with the exact flag.
fuzzybonds	Bond orders do not need to match
atomname	Atom names must match

Options are set using the *-strict* flag and joined with commas (,).

The default is:

`-strict substructure,fuzzybonds`

To reject dictionaries that do not match exactly, use:

`-strict exact,fuzzybonds`

However, the more permissive default setting has been known to generate invalid dictionaries on occasion. It is always safe to force exact matches.

To have WRITEDICT attempt to see if the ligand's residue has already been deposited in RCSB simply add the *-lookup* switch. This forces WRITEDICT to compare (based on the current *-strict* settings) the input residue to all known residues. This is useful if the input is from a 2D connection table, such as smiles or does not contain residue information.

5.5 Note on the MMFF94 versus MMFF94s forcefields

The MMFF94s forcefield is a variant of MMFF94 that emulates time-averaged structures typically observed during crystallographic structure determination, mainly planar geometries at unstrained delocalized trigonal nitrogen centers. However, there are many theoretical studies that show nitrogen centers are puckered [Halgren-VI-1999].

That being said, due to the prevalence of crystallographic examples where time averaging has occurred, many chemists erroneously consider the time-averaged structure to be correct, hence this variant is available for use.

WRITEDICT approximates the MMFF94s forcefield by enforcing planar aniline nitrogen configurations using the out of plane atom types and parameters from the MMFF94s. See the *-planarAniline* parameter to specify planar constraints.

5.6 Note on FLYNN and writing CIF Files

When FLYNN is run, CIF dictionary files are always written as output. Because the dictionary files are not conformation independent, they are written as if the molecules were split into separate files whether or not the user specifies FLYNN's *-split* flag. (See the split flag in *Advanced Parameters* below for more details). These files are numbered, however, so that the file with the lowest number corresponds to the best scored fit ligand.

For example, consider using the *-sortBy plp* flag and getting the following output:

```
1nhu-poses_n001_b001_s01_c003.pdb      1nhu-poses_n002_b001_s01_c000.pdb      1nhu-
poses_n003_b001_s01_c001.pdb 1nhu-poses_n004_b001_s01_c002.pdb
```

This indicates that the best fit to density (c000) has the second best PLP score (n001). By default the files are sorted by RSCC.

FLYNN CIF output consists of the following files:

1. a CIF file
2. a PDB file (in case atom names were remapped) or links were added.
3. an OEB file that contains the original molecule with annotations

It is safest, however, to run WRITEDICT on the output ligand and protein complex. The CIF files generated by FLYNN are primarily useful for adjusting torsions and geometries in various ligand building programs, such as coot.

5.7 Command Line Interface

A description of the command line interface can be obtained by executing FLYNN with the *--help* option.

```
prompt> flynn --help
```

will generate the following output:

Help functions:

```
flynn --help simple      : Get a list of simple parameters
flynn --help all         : Get a complete list of parameters
flynn --help <parameter> : Get detailed help on a parameter
flynn --help html        : Create an html help file for this program
```

5.7.1 Required Parameters

-in <filename>

File containing a molecule to be fit to density.

File type	Extension
SMILES	.smi .ism .smi.gz .ism.gz
OEBinary	.oeb .oeb.gz
SDF	.sdf .mol .sdf.gz .mol.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz
Old OEBinary	.bin

-out <filename>

File containing resulting conformations exported in the following formats:

File type	Extension
OEBinary	.oeb .oeb.gz
SDF	.sdf .mol .sdf.gz .mol.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz
Old OEBinary	.bin

-map <filename>

Input density grid used to fit ligand. Note that **mtz** files have additional settings, see [MTZ File Parameters](#)

Grid File type	Extension
MTZ	.mtz
OpenEye	.grd
Grasp	.phi
CCP4	.map .ccp4
XPLOR	.xplor xplmap
ASCII Grid	.agd

5.7.2 Optional Parameters

Input Options

-prot <filename>

Optional protein used to mask density. This model is used to mask away density where the ligand should not be placed. While the protein is not required, it is highly recommended.

File type	Extension
OEBinary	.oeb .oeb.gz
SDF	.sdf .mol .sdf.gz .mol.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz
Old OEBinary	.bin

-fragment

Fragments are fit taking the input fragment cocktail and, one at a time, fitting each fragment against each region of detected density. Once a fragment has been placed, it is further analyzed to ensure that all possible orientations of the fragment have been sampled. In poor density, several orientations may fit equally well. To break ties, FLYNN scores each pose with the following scores:

- o RSCC (real space correlation coefficient) This is a measure of fit to electron density. [Jones-1991]
- o PLP Piecewise-linear potential. [Verkivker-2000]
- o Chemscore [Eldridge-1997]

The docking scores are not used to fit the molecule, they are only used to rank the output. Unless highly symmetric molecules are being input, the real space correlation coefficient (RSCC) is the preferred method of ranking results to density.

All scores are annotated to the output ligands using SD data when appropriate (.SD and .OEB output) or using the PDB remark field:

Example PDB output

```
REMARK OpenEye Flynn MMFF/Shape v 2.1.0
REMARK Stereo Variant:      1
REMARK Blob:                1
REMARK Conformer:          1
REMARK Tanimoto Shape:      0.2004
REMARK Tanimoto MMFF/Shape: 0.2034
REMARK Local Strain:        4.9316
REMARK RSCC: 0.529438
REMARK PLP: -51.395416
REMARK Chemscore: -16.288113
```

Example SDF output

```
> <Tanimoto MMFF/SHAPE>
0.2034

> <Tanimoto Shape>
0.2004

> <Overlap>
364.767659122018

> <Fit Overlap>
781.301114117834

> <Local Strain>
4.9316

> <Lambda>
1.600000023842

> <Ref Overlap>
1371.829467773438

> <Stereo>
1

> <Blob>
1

> <Conformer>
1

> <RSCC>
0.529438

> <PLP>
-51.395416

> <Chemscore>
-16.288113
```

To use fragment mode, please add the “-fragment” option to the command line. In future versions of FLYNN, this will most likely become the default setting.

The output of the fragment fitting process is a file for each density region that includes the fragments fit to the region sorted from best-fit to worst fit. For example:

```
prompt> flynn -in fragments.smi -out 2IKO_cocktail.sdf ...
```

would result in the files:

```
2IKO_cocktail_blob001.sdf
2IKO_cocktail_blob002.sdf
2IKO_cocktail_blob003.sdf
```

one for each blob found. The first molecule in the file is the best fit to the density.

To sort using another measure, for instance, PLP, use the sort flag:

```
prompt> flynn -sortBy plp -in fragments.smi -out 2IKO_cocktail.sdf ...
```

When oeb format is used, the blob is also stored in the OEB file. When using VIDA, simply expand the molecule to investigate the blobs density.

-param <paramfile>

Defines the control parameter file. This file can contain a collection of parameters which can be used instead of writing each parameter to the command-line. In addition, the parameter file written by any FLYNN run (see *-prefix* below), can be used with the *-param* flag in subsequent FLYNN executions. Any command given explicitly on the command line will supersede any command found in a file specified with the *-param* parameter.

-box <filename>

Use the input molecule to generate a box volume. All density inside this box will be used to place the ligand.

-boxpad <number>

Pad the box created with the *-box* flag by an amount in Ångströms.

[default=0.0]

-blobsThenBox

Search for blobs first, if now suitable blobs are found, use the supplied box to bound density.

[default=false]

-densityAsIs

Use the supplied density as it is; do not try to find suitable blobs inside the density. Note that the supplied density has to be pretty close to the actual ligand density for this to work

[default=false]

-distance <value>

Reject blobs whose average distance to the protein is greater than this value.

[default=4.0]

-ligandAsIs

Do not generate conformations for the ligand; use only the supplied conformations.

-mmff94s

Use the MMFF94s variant of the MMFF94 forcefield (this uses planar aniline nitrogens).

[default=false]

-reportfile <filename>

File location for writing the report file. This is a comma separated file containing specifics about the results.

```
Smiles,Blob,Stereo Variant,Conformer,Tanimoto Shape,Tanimoto MMFF/SHAPE,Local Strain
CC(=O)[C@H]1CC[C@@H]2[C@@]1(CC[C@H]3[C@H]2CCC4=CC(=O)CC[C@]34C)C,1,,0.4110,0.4296,1.342
CC(=O)[C@H]1CC[C@@H]2[C@@]1(CC[C@H]3[C@H]2CCC4=CC(=O)CC[C@]34C)C,1,,0.313,0.3704,4.553
CC(=O)[C@H]1CC[C@@H]2[C@@]1(CC[C@H]3[C@H]2CCC4=CC(=O)CC[C@]34C)C,1,,0.3265,0.3628,5.223
```

-resname <name>

Set the output residue name to **<name>**. **<name>** must be less than or equal to three characters in length. This forces the output residue to be named **<name>** even if the residues structure does not match a known deposited residue.

-chainid <chainid>

Forces the ligand to be placed in the chain **<chainid>**

-sortAllChiral <true/false>

If set to true, when multiple chiralities are enumerated, sort all chiralities from best to worst per blob, otherwise sort chiral structures independently. For example, if two blobs are found, then the output will have all chiral structures sorted from best fit to worst fit for the first blob, and then all chiral structures sorted from best fit to worst fit for the second blobs, etc Note that the second blob may contain better fits than the first. (Note: This may interleave chiral structures).

[default=true]

-verbose

This is a boolean flag that controls the level of detail written to the log file. By default FLYNN will only write minimal information to the console. Verbose logging will cause more information to be written to the log file in order to follow behavior during program execution.

[default=false]

5.8 MTZ File Parameters

-autoMTZ

Automatically try to open the mtz file using the DELWT and FDELWT columns from REFMAC5 mtz files.

[default=true]

-Fc <columnname>

Column to use for Fc. *Note, to load arbitrary columns, you can use the *-Fc** and -Phic columns and the Fc maptype.**

[default=FC]

-Fdelwt <columnname>

Column to use for Fdelwt or difference map amplitudes

[default=F]

-Fobs <columnname>

Column to use for FObs.

[default=F]

-Fwt <columnname>

Column to use for Fwt or regular map amplitudes.

[default=FWT]

- Phic <columnname>**
Column to use for Phic.
[default=PHIC]
- Phidelwt <columnname>**
Column to use for Phidelwt or difference map phases.
- Phiwt <columnname>**
Column to use for Phwt or regular map phases.
- mtype**
The map type to use for fitting. Fo-Fc, Fc, 2Fo-Fc, 3Fo-Fc, Fwt, Fdelwt...
[default=2Fo-Fc]

5.9 Advanced Parameters

- flipper**
Force enumeration of all stereochemical centers. Otherwise, only missing stereo chemistry will be enumerated.
[default=false]
- overlays**
Set the number of initial rigidly fit overlays to optimize into density.
[default=10]
- precheck**
Perform only a preliminary check of the data to see if fitting is possible. (This flag is meaningless without the -reporhtml flag).
[default=false]
- reporhtml**
Generate an html report of the fitting process. This is a useful first step to verify the data. It also includes a 2D image of the ligand that is being fit in order to verify bond orders. All output of FLYNN will be captured in the specified html file.
- residues**
Comma separated list of residues to use as distance constraints. If residues are specified, the *-distance* flag will use the residues to reject blobs that are too far away. Note that with a residue list, the distance computed is a minimum distance, not an average distance. Residues must be specified as <residue number><chain id>. For example: 120C,134C
- rms**
If the input ligand has 3D coordinates, this calculates the RMS distance to the ligand and records it in the molecule. If -split is on, it also records it in the file name.
[default=false]
- split**
Split the result into different files. Each filename is annotated with the resulting shape score so that when the output directory is listed alphabetically, the best scores will be displayed in order. In the case where the output format is pdb the ligand fitting results are written in REMARK statements at the top of the file.

The splits are labeled as follows outputname_n###_b###_s###_c### where n### is the number of the ordered results, b### is the blob for the result, s### is the stereovariant and c### is the number of the conformer fit as ordered by MMFF/Shape.

[default=false]

-suppressH

By default FLYNN outputs hydrogens of the resulting poses. Setting this to false will not output hydrogens.

[default=false]

5.10 Fragment Fitting Parameters

-blobTanimoto

Density is searched for each input ligand. This flag sets the minimum tanimoto overlap for which to consider two blobs the same. When pruning similar blobs, the larger is taken.

[default = 0.45]

-sortBy

Choose the fragment sorting function, tanimoto, rsc (real space correlation coefficient), plp (piecewise linear potential) or chemscore.

The fragments are fit in the normal fashion (tanimoto), but then re-scored and sorted using the chosen metric.

[default = rsc]

5.11 3D Construction Parameters and Torsion Driving Parameters

These are advanced conformer generation parameters. Normally, they do not need to be adjusted. Please consult the OMEGA manual for assistance:

5.12 Example Commands

The example commands in this section can be run with files found in the `examples/afitt-cl/2.0.1` directory under the top-level installation directory.

5.12.1 Basic Usage

```
Prompt> flynn -in lnhu.ism -map lnhu_prot.mtz -prot lnhu_prot.pdb  
          -out lnhu-poses.pdb
```

Use the ligand specified by **lnhu.ism** and fit it against the density in the mtz file **lnhu_prot.mtz**. Since this is an mtz file and **-autoMTZ** is defaulting to true, the Fdelwt map will automatically be generated and used as the density target. The protein atoms specified in `lnhu_prot.pdb` will be used to mask the density before the map is searched for appropriately sized ligand volumes.

5.12.2 Generating Reports

```
prompt> flynn -in lnhu.ism -map lnhu_prot.mtz -prot lnhu_prot.pdb  
          -out lnhu-poses.pdb -precheck -reporthtml lnhu.html
```

Performs a preliminary check of the data, ensuring that the ligand can be processed and that the mtz file can be opened. An html file is generated **1nhu.html** with a corresponding image file **1nhu.html.gif**. The html file can be opened with a web browser to view the ligand being fit and any corresponding issues.

The output will be written to 1nhu-poses.pdb in pdb format.

5.12.3 Choosing box regions of density

```
prompt> flynn -in 1nhu.ism -map 1nhu_sigma.mtz -prot 1nhu_prot.ent
          -out 1nhu_poses.pdb -box 1nhu.pdb -boxpad 2.0
```

This is the same as above except that the density is pruned down to the bounding box specified by **1nhu.pdb** padded by 2.0 Ångströms. Furthermore, if a suitable blob is not found, then the pruned density will be used as is. This is quite effective for regions of density with a high degrees of disorder. It is also useful to select this option when using the supplied interface to **coot**.

5.12.4 Choosing density close to residues

Use the *-residues* and *-distance* flags to indicate residues close near the desired density. Only blobs that are found within *-distance* to the selected residues will be selected. Residues are specified by <residuenumber><chain> separated by commas.

```
prompt> flynn -in 1nhu.ism -map 1nhu_prot.mtz -prot 1nhu_prot.pdb
          -out 1nhu_poses.pdb -residues 528B,477B
```

5.12.5 Choosing MTZ columns

```
prompt> flynn -in 1nhu.ism -map 1nhu_sigma.mtz -prot 1nhu_prot.ent
          -out 1nhu_poses.pdb -box 1nhu.pdb -boxpad 2.0
          -boxThenBlobs -autoMTZ false -Fobs Frefined1 -Fc FC
          -Phic PHIC -mtype 2Fo-Fc -autoMTZ false
```

This example shows how to choose a non-standard column from the MTZ file. the map is generated using **Frefined1** as the observed amplitude column, **FC** as the calculated amplitude column and **PHIC** as the calculated phase column. Furthermore, the regular map **2Fo-Fc** is used. Note that **-autoMTZ** must be set to false for these settings to take effect.

5.12.6 Fragment usage

The **-fragment** flag may be included with any of the above commands. There are three main differences when running in fragment mode:

1. Multiple ligands are fit simultaneously.
2. If the blobs detected for difference ligands overlap, the ligands are assigned to the same blob. (This is for reporting purposes)
3. Ligands are sorted according to the Real Space Correlated Coefficient.

```
prompt> flynn -fragment -in 1nhu.ism -map 1nhu_prot.mtz -prot 1nhu_prot.pdb
          -out 1nhu_poses.sdf
```

When in fragment mode, it is better to output to a format that supports SD style data (.sdf, .oeb). This makes it much easier to analyze the results in a molecular viewer that supports a sortable spreadsheet. (VIDA, MOE and so on)

To sort the results using another scoring function, use the **-sortBy** flag.

```
prompt> flynn -sortBy chemscore -in 1nhu.smi ...
```

5.13 Results on The Gold Test Set

Included in the FLYNN distribution is a copy of the protein+\$ligand complexes that have structure factors available from the *Electron Density Server* [EDS].

Along with this data set is a simple python script that can be used to fit ligands to the test data set. OpenEye has prepared the input test set as follows:

1. The Structure Factors and protein complexes were downloaded from **EDS**.
2. The ligands were separated from the proteins.
3. The protein was re-refined using the original mtz file to remove ligand bias.

A simple directory structure was created as follows:

- **pdbcode/protein.pdb** - the re-refined protein
- **pdbcode/ligand.pdb** - the original ligand
- **pdbcode/protein.mtz** - the re-refined difference map density.

The following python script was used to run FLYNN on all of the proteins. (This script is also included in the FLYNN distribution):

```
#####
## Copyright (C) OpenEye Scientific 2007
#####
## This script analyzes the Gold dataset shipped with the flynn distribution
import os, sys
command = "../..../bin/flynn -in %(CODE)s/ligand.pdb -map %(CODE)s/protein.mtz " \
          "-prot %(CODE)s/protein.pdb -out %(CODE)s_results.sdf " \
          "-verbose -rms -reportfile %(CODE)s.log %(REDIRECT)s %(CODE)s.out"
if sys.platform == "win32":
    REDIRECT = ">"
else:
    REDIRECT = "&>"
RMS = []
for file in os.listdir("."):
    print file
    if os.path.isdir(file) and len(file) == 4:
        code = file
        CMD = command % {"CODE":code, "REDIRECT": REDIRECT}
        print CMD
        os.system(CMD)
        index = None
        rmses = []
        for line in open("%s.log"%code):
            if index is None:
                index = line.strip().split(",").index("RMSD")
            else:
                rmses.append( float(line.strip().split(",")[index]) )
```

```
if rmses:  
    print "\t====>Lowest RMS", min(rmses)
```

The results of FLYNN are shown in Figure *Gold Test Set Results*. The mean RMS to the crystal structure is 0.676 angstroms.

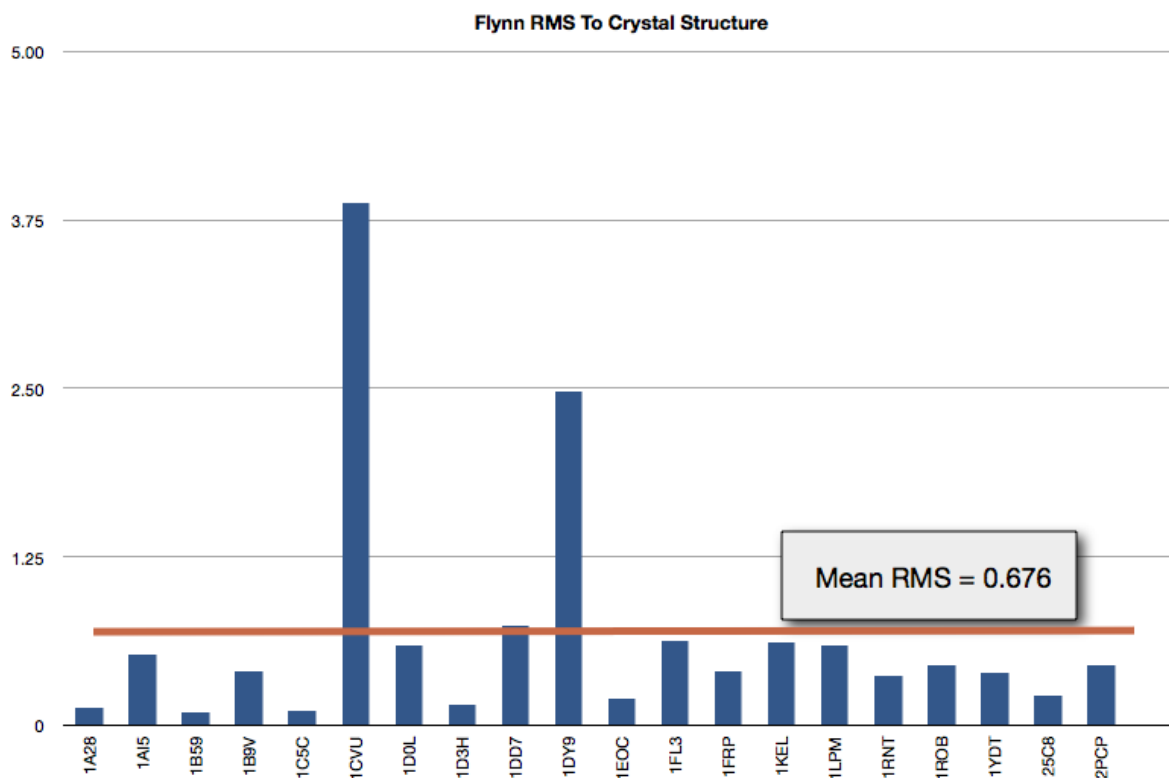


Figure 5.1: Gold Test Set Results

WRITEDICT USAGE

6.1 Command Line Interface

A description of the command line interface can be obtained by executing WRITEDICT with the `--help` option.

```
prompt> writedict --help
```

will generate the following output:

Help functions:

```
writedict --help simple      : Get a list of simple parameters
writedict --help all        : Get a complete list of parameters
writedict --help <parameter> : Get detailed help on a parameter
writedict --help html       : Create an html help file for this program
```

6.1.1 Required Parameters

-in <filename>

File containing a molecule to be fit to density.

File type	Extension
SMILES	.smi .ism .smi.gz .ism.gz
OEBinary	.oeb .oeb.gz
SDF	.sdf .mol .sdf.gz .mol.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz
Old OEBinary	.bin

-out <filename>

This specifies a prefix for outputting the refinement dictionary **.cif** or **.xplor** and the **pdb** file. The **pdb** file (and **pdb.oeb**) is always output and should be used for purposes of refinement since WRITEDICT sometimes corrects faulty hydrogen atom names and will include LINK records in the **pdb** if covalent bonds are detected.

Output Options

-assignCCP4

If true then CCP4 atom types will be assigned to the output **pdb** file.

[default=true]

-includeRotors

Set to true if the output should include rotor torsion terms. If this is set to false, refinement programs will not be able to modify any rotors.

[default=true]

-suppressH

If true, no hydrogens are output.

[default=false]

-type

set this to 0 for **REFMAC5** refinement dictionaries or 1 for **CNS/CNX (XPLOR)**. By default **REFMAC5** dictionaries are generated.

[default=0]

-planarAniline

Use the MMFF94s variant of the MMFF94 forcefield. This forcefield emulates time-averages structures observed in crystallographic and other structure determination methods.

The MMFF94s forcefield is a variant of MMFF94 that emulates time-averages structures typically observed during crystallographic structure determination, mainly planar geometries at unstrained delocalized trigonal nitrogen centers. However, there are many theoretical studies that show puckering at nitrogen centers. (see MMFF VI. in JCICS)

That being said, due to the prevalence of crystallographic studies, many chemist erroneously consider the time-averaged structure to be correct, hence this variant is available for use.

[default = false]

-strict

Change the known residue matching mode.

Options available for residue matching is as follows:

Option	Meaning
exact	Known residue and input residue graphs must match exactly.
substructure	Input residue may be a substructure of the known residue. Cannot be used with the exact flag.
fuzzybonds	Bond orders do not need to match
atomname	Atom names must match

Options are separated by commas. The default settings are set to limit the number of REFMAC5 issues. -strict exact,substructure will find more matches, but increases the risk of REFMAC5 rejecting the dictionary.

[default = exact,fuzzybonds]

-lookup

Attempts to lookup residues from the internal dictionary store using the current -strict status. This is useful to find out if a residue has already been deposited in the RCSB.

-nolookup

Force WRITEDICT to use the given residues, even if they do not match the deposited residues from the RCSB.

-writeFullDict

If true then all refinement dictionaries are created for all residues. If false only residues that are unknown or are in the set of replaceable residues or covalently bound to a ligand are output. Setting this to true can confuse REFMAC5.

{default=false}

-verbose

Output copious information of how writedict operates.

6.2 Example Commands

The basic usage of writedict is to take an input structure and generate the corresponding .pdb and .cif files for input into COOT or REFMAC5.

```
prompt> writedict -in lnhu.ism -out lnhu_out
```

This creates the files *Inhu_out.cif* and *Inhu_out.pdb*. The pdb is always created since the atom names may be remapped from the input. Since the input contains no residue information, the ligand is placed in residue "UNL".

To add your own residue names for unmapped residues:

```
prompt> writedict -in lnhu.ism -out lnhu_out -residues 153
```

Note that if the given residue name (153) does not map to the known residue 153 the next residue name in the unmapped residue list will be used. Since none are specified it will go to the standard defaults: "UNL,UN1,UN2..."

Note: When doing whole complexes, renaming residues gets quite complicated since you may not know which residues will map before you run WRITEDICT.

```
prompt> writedict -in lnhu.ism -out lnhu_out -residues LIG -nolookup
```

Force the residue to be named LIG. (It will also indicate an invalid dictionary)

```
Warning: LIG 1: Known Residue, but does not map to previous residue: LIG from: RCSB
Warning: LIG 1: previous smiles c1ccc2c(c1)CC3=C(NN=C23)c4ccncc4
Warning: LIG 1: current smiles c1ccc(cc1)CC(C(=O)[O-])N(Cc2cccc(c2)C(F)(F)F)C(=O)c3ccc(cc3Cl)Cl
Warning: LIG: Keeping residue name (dictionary may be invalid)
Warning: LIG: Adding new residue dictionary
Warning: LIG: already in dictionary from: RCSB but with different structure.
Warning: LIG: c1ccc2c(c1)CC3=C(NN=C23)c4ccncc4
Warning: LIG: c1ccc(cc1)CC(C(=O)[O-])N(Cc2cccc(c2)C(F)(F)F)C(=O)c3ccc(cc3Cl)Cl
Warning: LIG: Overwriting existing residue (dictionary may be invalid)
```

To ensure planarAniline constraints consistent with the MMFF94s forcefield, use the *-planarAniline* flag.

```
prompt> writedict -in lnhu.ism -out lnhu_out -planarAniline
```


ALIGNGRID USAGE

7.1 Command Line Interface

A description of the command line interface can be obtained by executing ALIGNGRID with the `--help` option.

```
prompt> aligngrid --help
```

will generate the following output:

```
Help functions:
  aligngrid --help simple      : Get a list of simple parameters
  aligngrid --help all        : Get a complete list of parameters
  aligngrid --help <parameter> : Get detailed help on a parameter
  aligngrid --help html       : Create an html help file for this program
```

7.1.1 Required Parameters

-grid

Input density grid. This is the grid that is to be rotated and translated into the new reference frame.

-out

The output grid in any writable format. These include **.ccp4**, **.map** and **.grd**

-protein

The query protein that corresponds to the input grid.

-target

The target protein that the grid and query protein will be aligned toward.

7.1.2 Advanced Options

-outprot

An optional file to output the query protein after it is translated and rotated into the new reference frame.

-padding

When writing out a **.grd** file, this indicates the added extents (in Ångströms around the protein when outputting the density).

-scale

The final map spacing in Ångströms. (The default is the minimum spacing in X, Y or Z)

-verbose

Triggers copious logging output.

7.2 MTZ File Parameters

-autoMTZ

Automatically try to open the mtz file using the DELWT and FDELWT columns from REFMAC5 mtz files.

[default=true]

-Fc <columnname>

Column to use for Fc. *Note, to load arbitrary columns, you can use the *-Fc** and -Phic columns and the Fc maptype.**

[default=FC]

-Fdelwt <columnname>

Column to use for Fdelwt or difference map amplitudes

[default=F]

-Fobs <columnname>

Column to use for FObs.

[default=F]

-Fwt <columnname>

Column to use for Fwt or regular map amplitudes.

[default=FWT]

-Phic <columnname>

Column to use for Phic.

[default=PHIC]

-Phidelwt <columnname>

Column to use for Phidelwt or difference map phases.

-Phiwt <columnname>

Column to use for Phwt or regular map phases.

-mtype

The map type to use for fitting. Fo-Fc, Fc, 2Fo-Fc, 3Fo-Fc, Fwt, Fdelwt...

[default=2Fo-Fc]

7.3 Example Commands

```
prompt> aligngrid -grid input.ccp4 -query protein1.pdb  
             -target protein2.pdb -out output.ccp4
```

Take the input grid in ccp4 format and protein. Align protein1 to protein2 and rotate and translate the input grid using the same alignment outputting the result to *output.ccp4*.

INTEGRATION WITH COOT

Due to its command line nature, AFITT-CL tools (such as FLYNN) can be integrated with other crystallographic applications such as Coot (and WinCoot). OpenEye supplies a simple script that allows users of Coot to use FLYNN directly from the Coot GUI.

This script adds a “Fit Ligand...” option and a “Make cif...” option to the extensions menu.

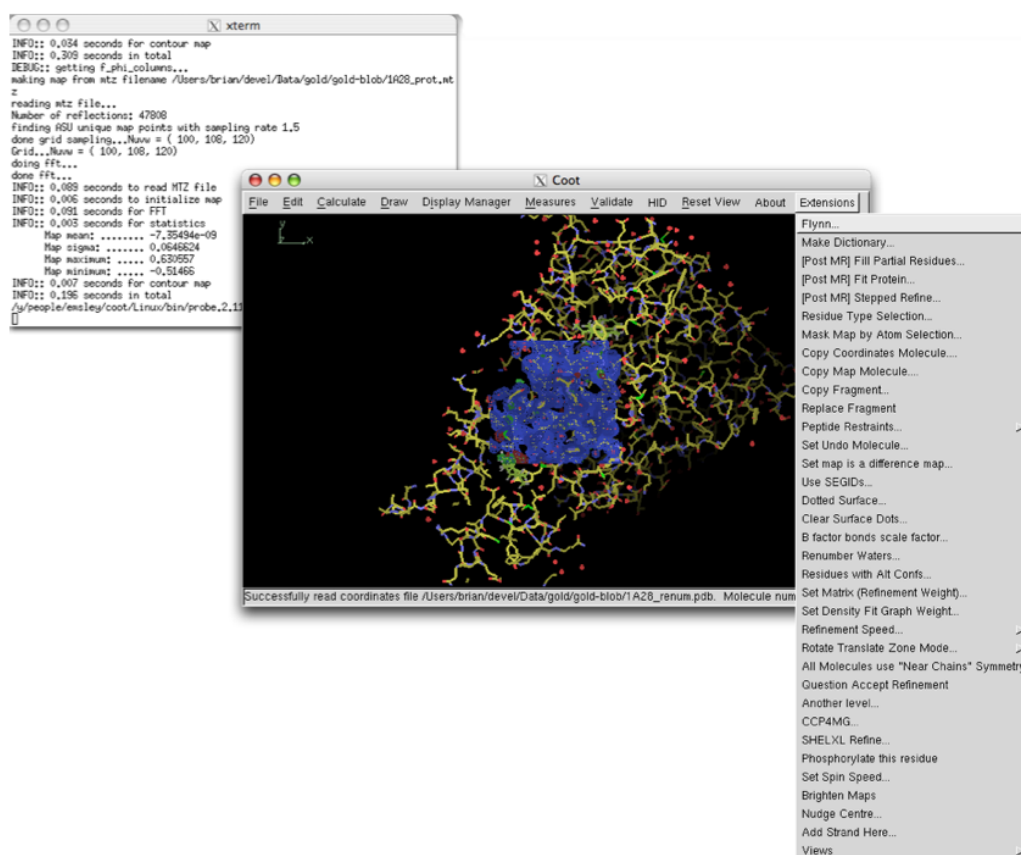


Figure 8.1: Coot Integration

8.1 Using FLYNN

Activating the “Fit Ligand...” menu option from the Extensions menu opens up the FLYNN dialog:

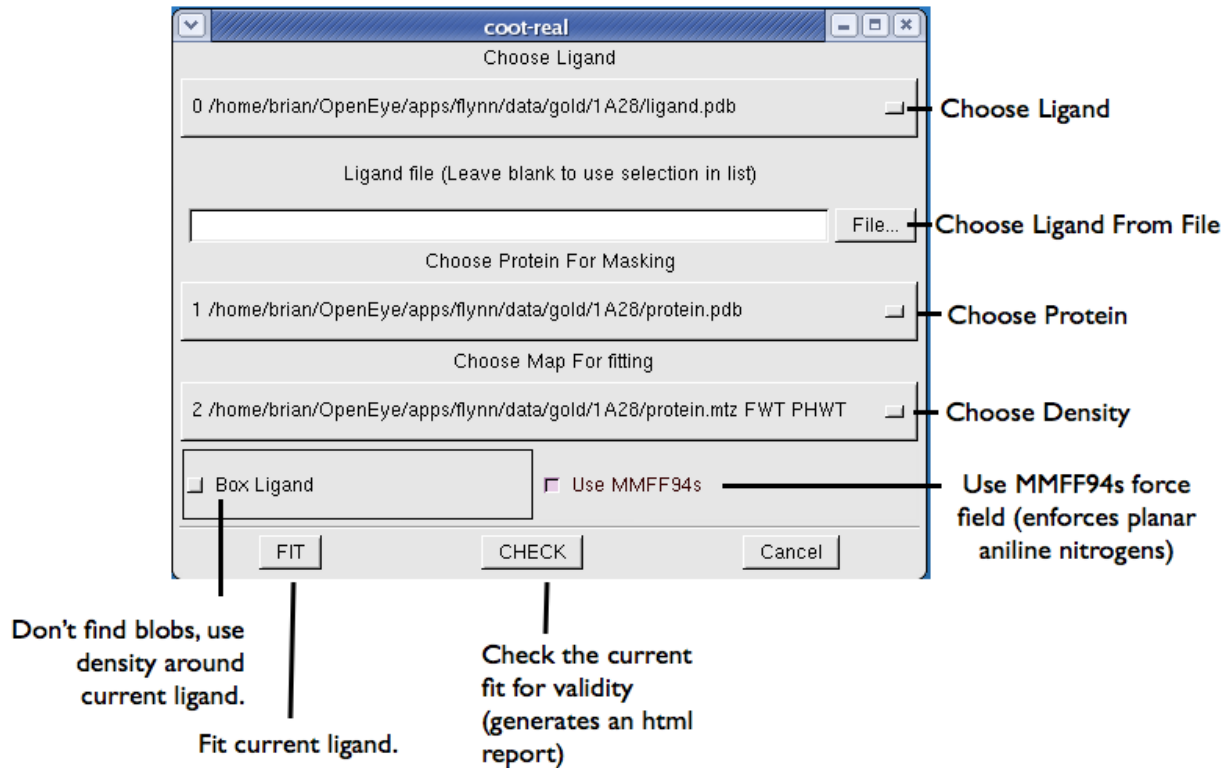


Figure 8.2: FLYNN Dialog to fit a ligand

To use the FLYNN dialog, select the Ligand to fit from the pull-down menu. Next select the Protein and then the desired map file.

If the map is a difference map, click on the “Difference Map” to optimize the analysis. Then click on Fit.

The output of FLYNN will appear in the console window used to start coot. When FLYNN is complete, coot will automatically load the resulting ligands.

8.2 FLYNN Advanced Options

FLYNN’s coot interface includes access to some of the advanced options contained in the application.

8.2.1 Box option

Unlike the AFITT GUI interface, coot does not have a method to select density blobs to be used for fitting. If the density is poor, flynn can use the input ligand as a bounding box to fit to the surrounding density.

8.2.2 Check option

Running check runs flynn without performing any fitting operation. When the check is completed, an html report is generated and opened in a webbrowser (if available). Since coot only display's single bonds, the report includes a depiction of the ligand being fit in order to verify that the ligand being analyzed is actually correct.

The report also indicates any problems or issues with the ligand or the map being fit against.

8.2.3 MMFF94s option

This option uses the MMFF94s variant of the MMFF94 forcefield to enforce planar aniline configurations. See the FLYNN *Advanced Parameters* command line flags for more details.

8.3 Using WriteDict

Activating the “Make Dictionary...” menu option opens up the WRITEDICT dialog:

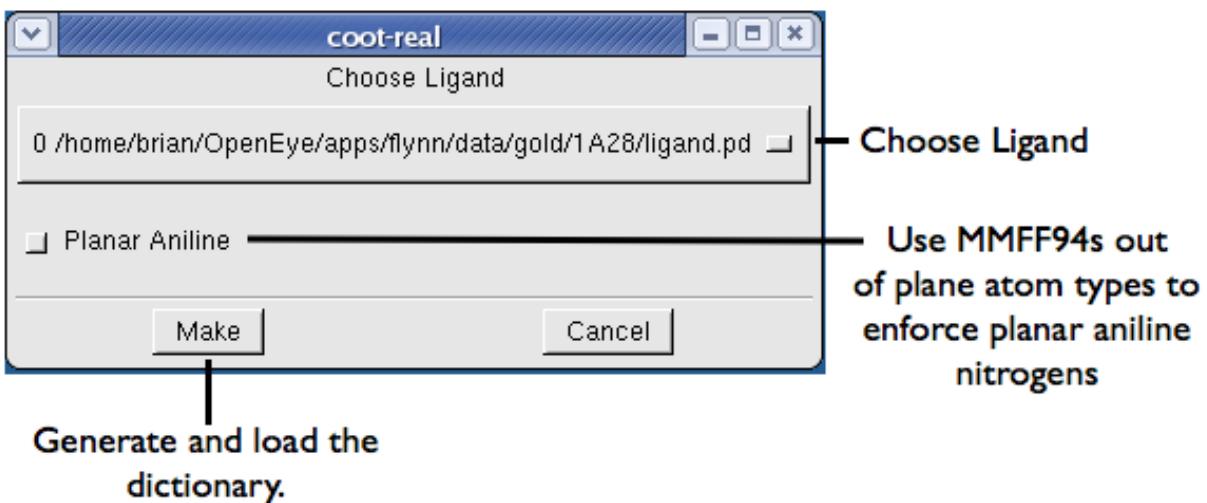


Figure 8.3: **Writedict Dialog**

To use the WRITEDICT dialog, simply select the protein or ligand from the pull-down menu and click “ok”. *Coot* will automatically open up the resulting dictionary and pdb files generated by WRITEDICT.

Note that new “.pdb” files are generated when using writedict, this is because hydrogen may have been relabeled and covalent *LINK* records may have been inserted into the *pdb* file.

8.3.1 WRITEDICT Advanced Options

FLYNN’s coot interface includes access to some of the advanced options contained in the application.

8.3.2 planarAniline option

Similar to FLYNN’s *MMFF94s* option, this enforces planar aniline nitrogen configurations. Note that since this is an approximation of the *MMFF94s* forcefield using planar constraints, it is not called *MMFF94s*. See the WRITEDICT

Output Options command line flags for more details.

8.3.3 Installing Coot Interface

The `coot` script file “`flynn.scm`” is included in your FLYNN distribution in the “`etc`” directory. The following steps are required to integrate FLYNN with `coot`.

1. Install `coot` (at least version 0.3.3)
2. Install `flynn` and `writedict` somewhere in your path. If you can't run FLYNN from the command line, neither can `Coot`.
3. Place `flynn.scm` to the `coot` extensions directory. This is the directory where the files `extensions.scm` and `coot.scm` are located.

This directory may be one of the following:

`/usr/local/coot/scheme /usr/local/xtal/coot64/share/coot/scheme/`

Typing **which coot** will help locate these directories.

4. Copy `flynn.scm` to the extension list in `coot.scm` which is in the same directory as `extensions.scm`

To accomplish this, change the following code in `coot.scm`

```
(define load-all-scheme
  (lambda (use-gui?)
    (let ((pre-list (list "filter.scm"
                        "coot-utils.scm"))
          (post-list (list "coot-lsq.scm"
                          "shelx.scm"))
```

to look like this:

```
(define load-all-scheme
  (lambda (use-gui?)
    (let ((pre-list (list "filter.scm"
                        "coot-utils.scm"))
          (post-list (list "coot-lsq."
                          "shelx.scm"
                          "flynn.scm"))
```

5. Finally, make the following changes to `extensions.scm`

For `coot` 0.3.3 to 0.5 near the top of the file, change:

```
(if (defined? 'coot-main-menubar)
    (let ((menu (coot-menubar-menu "Extensions"))))
```

to become:

```
(if (defined? 'coot-main-menubar)
    (let ((menu (coot-menubar-menu "Extensions"))
          (add-simple-coot-menu-menuitem menu "Flynn..."
            (lambda () (do-flynn-gui)))
          (add-simple-coot-menu-menuitem menu "Make Dictionary..."
            (lambda () (do-flynn-gendict-gui))))
```

For `coot` 0.6.x, near the top of the file, change:

```
(if (defined? 'coot-main-menubar)
  ;; -----
  ;;           extensions
  ;; -----
  ;;
  (let ((menu (coot-menubar-menu "Extensions")))
```

to become:

```
(if (defined? 'coot-main-menubar)
  ;; -----
  ;;           extensions
  ;; -----
  ;;
  (let ((menu (coot-menubar-menu "Extensions")))
    (add-simple-coot-menu-menuitem menu "Flynn..."
      (lambda () (do-flynn-gui)))
    (add-simple-coot-menu-menuitem menu "Make Dictionary..."
      (lambda () (do-flynn-gendict-gui))))
```

8.3.4 Installing WinCoot Interface

WinCoot does not support writing extensions in GUILE. Extensions may be written in scheme, however.

The WinCoot scheme script file *flynn.py* is included in your FLYNN distribution in the *utilities/flynn/wincoot* directory. The following steps are required to integrate FLYNN with *WinCoot*.

1. Install *WinCoot* (at least version 0.3.3)
2. Install *flynn* and *writedict* somewhere in your path. If you can't run FLYNN from the command line, neither can *WinCoot*.
3. Place this file *flynn.py* in the coot extensions directory where *extensions.py* is located.

Typically, this directory is *C:WinCootsharecootscheme*.

4. Add *flynn.py* to the extension list in *coot_load_modules+gui.py*. (same directory as *extensions.py*) *before extensions.py*

For example:

```
"coot-gui.py",
"flynn.py",
"extensions.py",
```

5. Add the following lines to the file named *extensions.scm* (this will be in your coot distribution)

The easiest place to put this right after the line

```
menu = coot-menubar-menu("Extensions")
```

here are the lines to add:

```
add-simple-coot-menu-menuitem(menu, "Fit Ligand...", do-flynn-gui)
add-simple-coot-menu-menuitem(menu, "Make CIF...", do-writedict-gui)
```

Flynn Takes Forever!

The most common case of FLYNN taking forever is accidentally choosing a protein in the ligand drop down menu. Trust me, it's been done. As mentioned before, FLYNN has no safe-guard for this.

RELEASE NOTES

9.1 AFITT-CL 2.1.0

July 2011

9.1.1 WRITEDICT

- Updated internal REFMAC and RCSB to current versions.
- When remapping dictionaries, WRITEDICT now indicates where the matching monomer was sourced.
- Moved “lookup” out of the -strict settings to a new flag “-lookup” adding this switch will search for a matching monomer from the internal libraries using the current -strict settings.
- -force options added. WRITEDICT will output the residue name even if it doesn’t match the internal list of known monomers.
- Various corrections to aniline nitrogen dictionaries.

9.1.2 FLYNN

- FLYNN now sorts the output fits based on real space correlation coefficient if applicable (RSCC)
- FLYNN now can automatically fit fragment cocktails. (use -fragment flag)
- FLYNN adds additional scores to all ligands besides the default MMFF Tanimoto/Shape, these are RSCC (Real space correlation coefficient), PLP (Piecewise-linear potential, requires protein) and Chemscore (requires protein).
- -rename now forces the ligand to use the given residue name even if it conflicts with the internal monomer library.
- -chainid Output the ligands with the given chain id.

9.2 AFITT-CL 2.0.1

September 2010

Package renamed from FLYNN to AFITT-CL. None of the internal programs have been renamed. AFITT and AFITT-CL will be released simultaneously with the same version numbers from this point forward.

9.2.1 WRITEDICT

- Added correct dictionary creation including connection table. This is required for newer versions of *coot*
- CIF dictionaries properly encode all LINK records.
- CIF dictionaries now encode proper planarity for guanine and no longer have duplicate planar atoms when using MMFF94s.
- CIF Dictionaries now look up known non-peptide residues. If the residue is not consistent with the known dictionary, the residue name is changed and a new MMFF94 derived dictionary is generated.
- Updated COOT integration to 0.6+

9.2.2 FLYNN

- Enhanced automatic density location. Blobs are now slightly larger than before making better targets for fitting.
- Added experimental fragment fitting. See the documentation for more information about this command line option. For version 2.0.1 this requires the fragment feature of the FLYNN license. This requirement will be removed when this feature is considered non-experimental. If you are interested in helping us test this feature, please contact business@eyesopen.com for more details.
- Refinement dictionaries are always written out for the ligand. Please note that these are written as if the `-split` flag is set to true. Refer the the documentation or type “`flynn -help`” at the command line for more details.
- Updated COOT integration to 0.6+

9.3 FLYNN 2.0.0

February 2010

- Internal Release used with collaborators. Used for testing new dictionary formats.

BIBLIOGRAPHY

- [Koch-1974] M.H. Koch, **Automatic interpretation of electron-density maps for organic structures**, *Acta Cryst A*, Vol. 30, pp. 67-70, **1974**
- [Cascarano-1991] G. Cascarano, C. Giacovazzo, M. Camalli, R. Spagna, and D. Watkin, **Automatic solution and refinement of crystal structures by means of the package UNIQUE**, *Acta Cryst. A*, Vol. 47, pp. 373-381, **1991**
- [Halgren-I-1996] T.A. Halgren, **Merck Molecular Force Field: I. Basis, Form, Scope, Parameterization and Performance of MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 490-519, **1996**
- [Halgren-II-1996] T.A. Halgren, **Merck Molecular Force Field: II. MMFF94 van der Waals and Electrostatic Parameters for Intermolecular Interactions**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 520-552, **1996**
- [Halgren-III-1996] T.A. Halgren, **Merck Molecular Force Field: III. Molecular Geometries and Vibrational Frequencies**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 553-586, **1996**
- [Halgren-IV-1996] T.A. Halgren and R.B. Nachbar, **Merck Molecular Force Field: IV. Conformational Energies and Geometries for MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 687-615, **1996**
- [Halgren-V-1996] T.A. Halgren, **Merck Molecular Force Field: V. Extension of MMFF94 using Experimental Data, Additional Computational Data and Empirical Rules**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 616-641, **1996**
- [Murshudov-1997] G.N. Murshudov, A.A. Vagin and E.J. Dodson, **Refinement of Macromolecular structures by Maximum likelihood method**, *Acta Cryst. D*, Vol. 53, pp. 240-255, **1997**
- [Halgren-VI-1999] T.A. Halgren, **MMFF VI. MMFF94s Option for Energy Minimization Studies**, *Journal of Computational Chemistry*, Vol. 20, pp. 720-729, **1999**
- [Halgren-VII-1999] T.A. Halgren, **MMFF VII. Characterization of MMFF94, MMFF94s and Other Widely Available Force Fields for Conformational Energies and for Intermolecular Interaction Energies and Geometries**, *Journal of Computational Chemistry*, Vol. 20, pp. 730-748, **1999**
- [Jones-1991] T.A. Jones, J.Y. Zou and S.W. Cowan **Improved Methods for Building Protein Models in Electron Density Maps and the Location of Errors in these Models**, *Acta Cryst.*, A47, 110-119
- [Verkivker-2000] Gennady M. Verkivker, Djamal Bouzida, Daniel K. Gehlaar, Paul A. Rejto, Sandra Arthurs, Anthony B. Colson, Stephan T. Freer, Veda Larson, Brock A. Luty, Tami Marrone and Peter W. Rose, **Deciphering common failures in molecular docking of ligand-protein complexes**, *Journal of Computer-Aided Molecular Design (JCAMD)*, Vol. 14, pp.731-751, 2000.
- [Eldridge-1997] Matthew D. Eldridge, Christopher W. Murray, Timothy R. Auton, Gaia V. Paolini and Roger P. Mee, **Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes**, * *Journal of Computer-Aided Molecular Design (JCAMD)**, Vol. 11, pp. 425-445, 1997.

- [Diller-1999] D.J. Diller and C. Verlinde, **A critical evaluation of several global optimization algorithms for the purpose of molecular docking**, *J. Comp. Chem.*, Vol. 20, 1740-1751, **1999**
- [Oldfield-2001] T. Oldfield, **Creating structure features by data mining the PDB to use as molecular-replacement models**, *Acta Cryst. D*, Vol 57, pp. 696-705, **2001**
- [Bostrom-2002] J. Boström, **Reproducing the Conformations of Protein-Bound Ligands: A Critical Evaluation of Several Popular Conformational Searching Tools**, *Journal of Computer-Aided Molecular Design (JCAMD)*, Vol. 15, pp. 1137, **2002**
- [Bostrom-2003] J. Boström, J.R. Greenwood, and J. Gottfries, **Assessing the Performance of OMEGA with Respect to Retrieving Bioactive Conformations**, *Journal of Molecular Graphics and Modeling*, Vol. 21, pp. 449-462, **2003**
- [Menendez-2003] A. Menéndez-Velazquez and S. García-Granda, **A procedure towards the automatic solution of crystal structures by means of topological analysis of Fourier maps**, *J. Appl. Cryst.*, Vol 36, pp. 193-205, **2003**
- [Vagin-2004] A.A. Vagin, R.A. Steiner, A.A. Lebedev, L. Potterton, S. McNicholas, F. Long and G.N. Murshudov, **REFMAC5 dictionary: organization of prior chemical knowledge and guidelines for its use**, *Acta Cryst. D.*, Vol. 60, pp. 2184-2195, **2004**
- [Perola-2004] E. Perola and P.S. Charifson, **Conformational analysis of drug-like molecules bound to proteins: an extensive study of ligand reorganization upon binding**, *J. Med. Chem.*, Vol. 47, pp. 2499-2510, **2004**
- [Wlodek-2006] S. Wlodek, A.G. Skillman and A. Nicholls, **Automated ligand placement and refinement with a combined force field and shape potential**, *Acta Cryst. D*, Vol. 62, pp. 741-749, **2006**
- [Sykes-2007] J.A. Grant, B.T. Pickup, M. Sykes, C.A. Kitchen and A. Nicholls, **A simple formula for dielectric polarisation energies: The Sheffield Solvation Model**, *Chem. Phys. Lett.*, Vol. 441, pp. 163-166, **2007**
- [EDS] (online: <http://eds.bmc.uu.se/eds/>)

INDEX

Symbols

- Fc <columnname>
 - aligngrid command line option, 30
 - flynn command line option, 18
- Fdelwt <columnname>
 - aligngrid command line option, 30
 - flynn command line option, 18
- Fobs <columnname>
 - aligngrid command line option, 30
 - flynn command line option, 18
- Fwt <columnname>
 - aligngrid command line option, 30
 - flynn command line option, 18
- Phic <columnname>
 - aligngrid command line option, 30
 - flynn command line option, 18
- Phidelwt <columnname>
 - aligngrid command line option, 30
 - flynn command line option, 19
- Phiwt <columnname>
 - aligngrid command line option, 30
 - flynn command line option, 19
- assignCCP4
 - writedict command line option, 25
- autoMTZ
 - aligngrid command line option, 30
 - flynn command line option, 18
- blobTanimoto
 - flynn command line option, 20
- blobsThenBox
 - flynn command line option, 17
- box <filename>
 - flynn command line option, 17
- boxpad <number>
 - flynn command line option, 17
- chainid <chainid>
 - flynn command line option, 18
- densityAsIs
 - flynn command line option, 17
- distance <value>
 - flynn command line option, 17
- flipper
 - flynn command line option, 19
- fragment
 - flynn command line option, 15
- grid
 - aligngrid command line option, 29
- in <filename>
 - flynn command line option, 14
 - writedict command line option, 25
- includeRotors
 - writedict command line option, 25
- ligandAsIs
 - flynn command line option, 17
- lookup
 - writedict command line option, 26
- map <filename>
 - flynn command line option, 15
- mmff94s
 - flynn command line option, 17
- mtype
 - aligngrid command line option, 30
 - flynn command line option, 19
- nolookup
 - writedict command line option, 26
- out
 - aligngrid command line option, 29
- out <filename>
 - flynn command line option, 14
 - writedict command line option, 25
- outprot
 - aligngrid command line option, 29
- overlays
 - flynn command line option, 19
- padding
 - aligngrid command line option, 29
- param <paramfile>
 - flynn command line option, 17
- planarAniline
 - writedict command line option, 26
- precheck
 - flynn command line option, 19
- prot <filename>

flynn command line option, 15

-protein
aligngrid command line option, 29

-reportfile <filename>
flynn command line option, 17

-reporthtml
flynn command line option, 19

-residues
flynn command line option, 19

-resname <name>
flynn command line option, 18

-rms
flynn command line option, 19

-scale
aligngrid command line option, 29

-sortAllChiral <true/false>
flynn command line option, 18

-sortBy
flynn command line option, 20

-split
flynn command line option, 19

-strict
writedict command line option, 26

-suppressH
flynn command line option, 20
writedict command line option, 26

-target
aligngrid command line option, 29

-type
writedict command line option, 26

-verbose
aligngrid command line option, 29
flynn command line option, 18
writedict command line option, 26

-writeFullDict
writedict command line option, 26

A

aligngrid command line option

- Fc <columnname>, 30
- Fdelwt <columnname>, 30
- Fobs <columnname>, 30
- Fwt <columnname>, 30
- Phic <columnname>, 30
- Phidelwt <columnname>, 30
- Phiwt <columnname>, 30
- autoMTZ, 30
- grid, 29
- mtype, 30
- out, 29
- outprot, 29
- padding, 29
- protein, 29
- scale, 29

-target, 29

-verbose, 29

APPNAME_OE_ARCH, 8

E

environment variable

- APPNAME_OE_ARCH, 8
- OE_ARCH, 8
- OE_LICENSE, 7
- PATH, 8, 9

F

flynn command line option

- Fc <columnname>, 18
- Fdelwt <columnname>, 18
- Fobs <columnname>, 18
- Fwt <columnname>, 18
- Phic <columnname>, 18
- Phidelwt <columnname>, 19
- Phiwt <columnname>, 19
- autoMTZ, 18
- blobTanimoto, 20
- blobsThenBox, 17
- box <filename>, 17
- boxpad <number>, 17
- chainid <chainid>, 18
- densityAsIs, 17
- distance <value>, 17
- flipper, 19
- fragment, 15
- in <filename>, 14
- ligandAsIs, 17
- map <filename>, 15
- mmff94s, 17
- mtype, 19
- out <filename>, 14
- overlays, 19
- param <paramfile>, 17
- precheck, 19
- prot <filename>, 15
- reportfile <filename>, 17
- reporthtml, 19
- residues, 19
- resname <name>, 18
- rms, 19
- sortAllChiral <true/false>, 18
- sortBy, 20
- split, 19
- suppressH, 20
- verbose, 18

O

- OE_ARCH, 8
- OE_LICENSE, 7

P

PATH, 8, 9

W

writedict command line option

- assignCCP4, 25
- in <filename>, 25
- includeRotors, 25
- lookup, 26
- nolookup, 26
- out <filename>, 25
- planarAniline, 26
- strict, 26
- suppressH, 26
- type, 26
- verbose, 26
- writeFullDict, 26