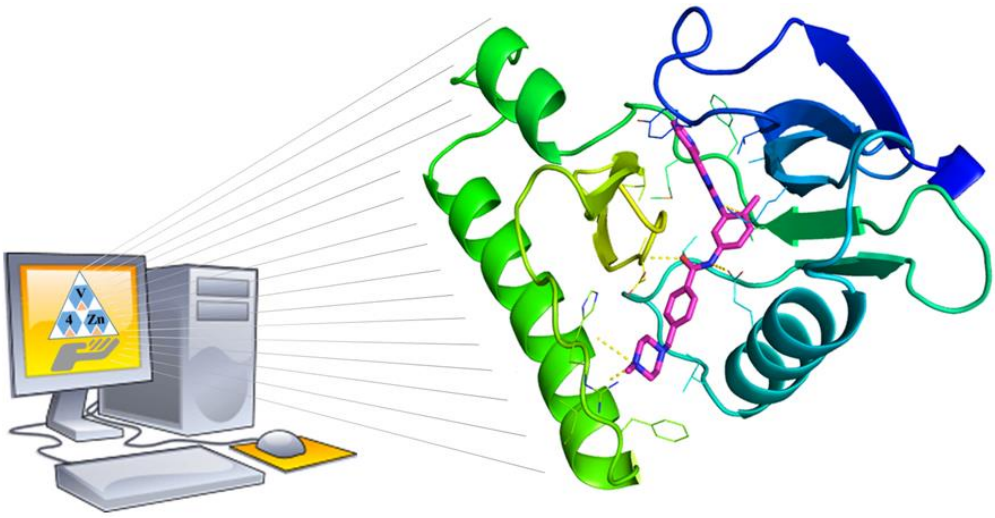
AMDock



Assisted Molecular Docking with AutoDock Vina and AutoDock4

USER MANUAL

Version 1.1.0

2019

This page intentionally blank

AMDock

USER MANUAL

Contributions from

Mario S. Valdés-Tresanco

Mario E. Valdés-Tresanco

Pedro A. Valiente

Ernesto Moreno

2019

TABLE OF CONTENTS

AMDock

Getting Started

INTRODUCTION

Hello! Here we introduce a manual exposing all functionalities of AMDock program. AMDock is a GUI program that assists docking simulations with AutoDock Vina and AutoDock4 (including AutoDock4$_{Zn}$ force field for zinc-containing metalloproteins). The program includes: preparation of protein and ligand input files, the definition of a proper search space and docking simulation itself. There are multiple options to perform these tasks. That way, the user will be free to choose how automatic the docking simulation will be. Several programs are used and their functionalities on AMDock context are explained across the manual. However, other applications of these programs are omitted here. All information about these programs can be reviewed in:

AutoDock Vina 1.2.1:

Manual: http://vina.scripps.edu/manual.html

Original Publication: Trott, O., Olson, A. J. (2010) AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem*, **31**: 455-461.

AutoDock 4.2.6:

Manual: http://autodock.scripps.edu/faqs-help/manual/autodock-4-2-user-guide

Original Publication: Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., Olson, A. J. (2009) Autodock4 and AutoDockTools4: automated docking with selective receptor flexibility. *J Comput Chem*, **30**: 2785-2791.

AutoDock4$_{Zn}$ Force Field:

Manual: http://mgldev.scripps.edu/AutoDockZN/AutoDockZN_userguide.pdf

Original Publication: Santos-Martins, D., Forli, S., João Ramos, M., Olson, A. J. (2014) AutoDock4$_{Zn}$: an improved AutoDock force field for small-molecule docking to zinc metalloproteins. *J Chem Info Model*, **54**: 2371-2379.

Open Babel 2.4.1:

Manual: http://openbabel.org/docs/dev/OpenBabel.pdf

Original Publication: O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., Hutchison, G. R. (2011) Open Babel: An open chemical toolbox. *J Cheminf*, **3**: 33.

PDB2PQR 2.1:

Manual: http://www.ics.uci.edu/~dock/pdb2pqr/userguide.html

Original Publication: Dolinsky, T. J., Nielsen, J. E., McCammon, J. A., Baker, N. A. (2004) PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucleic Acids Res*, **32**: W665-7.

AutoDockTools Module 1.5.7:

*Manual:*http://autodock.scripps.edu/faqs-help/tutorial/using-autodock-4-with-autodocktools

Original Publication: Sanner, M. F. (1999) Python: A Programming Language for Software Integration and Development. *J Mol Graphics Mod*, **17**: 57-61

AutoLigand:

Manual: http://autodock.scripps.edu/faqs-help/tutorial/using-autoligand-with-autodocktools

Original Publication: Harris, R., Olson, A., Goodsell, D. (2008) Automated prediction of ligand-binding sites in proteins. *Proteins*, **70**: 1505-1517

Optimal Box Size 1.1:

Manual: http://www.brylinski.org/eboxsize

Original Publication: Feinstein, W. P., Brylinski, M. (2015) Calculating an optimal box size for ligand docking and virtual screening against experimental and predicted binding pockets. *J Cheminf*, **7**: 18

PyMOL 1.8.5:

Manual: http://pymol.sourceforge.net/newman/userman.pdf

Original Publication: DeLano, W. L. (2002) The PyMOL Molecular Graphics System.

In any case, we have tried hard to be as clear and concise as possible. Comments and questions are welcome, please send them to AMDock mailing list.

How to cite AMDock?

When using AMDock program, please reference:

Coming soon!...

and the original publication of the corresponding module.

Thanks!

DOWNLOAD AND INSTALLATION

AMDock is distributed freely to the public. The current versions for Linux and Windows are available at https://github.com/Valdes-Tresanco-MS

Usually, the installation of a program becomes a “headache” to naive users. So, we have tried hard to make it as easy as possible. In this sense, the program has been written in Python and has been tested on Windows 10, Ubuntu 18.04 and Debian 8.0. It has not tested on MAC platform yet.

The Windows version of AMDock is distributed as a compressed file with all dependencies and functionalities. That means you can execute it quickly and without installing nothing more.

On the other hand, the UNIX version of AMDock is distributed as a compressed file. So, you must decompress this file, open the “README” file and follow the instructions.

AMDock

Basis

AMDock WORKFLOW

In order to understand how AMDock works and what kind of information is needed, it is necessary to know what programs are integrated into AMDock environment and which procedures are performed (Figure 1).

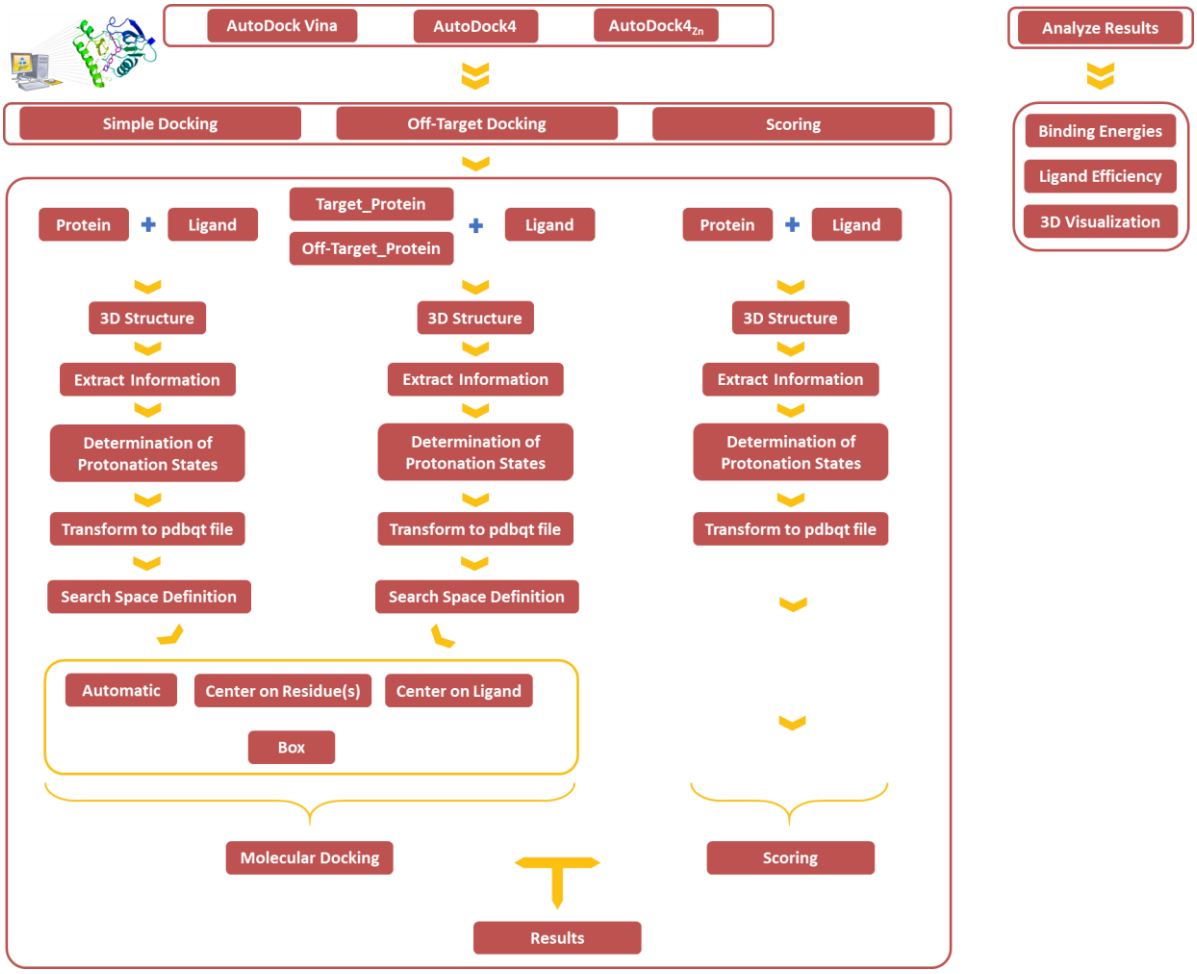


Figure 1. Workflow in AMDock

AMDock assists docking simulations with AutoDock Vina and AutoDock4 (including AutoDock4$_{Zn}$ force field for zinc-containing metalloproteins). Once the docking engine has been selected, you can perform a Simple Docking, Off-Target Docking or Scoring procedure.

AMDock proceeds by following 4 steps:

-Defining a Working Directory

First of all, a new directory is created (default name: Docking_Project) in a location specified by the user. This directory contains 2 folders: *i*) "input", where the protein (*.pdb, *.ent, *.pqr, *.pdbqt) and ligand (*.pdb, *.mol2, *.pdbqt) files are stored and *ii*) "results", where docking results are stored. Docking results will depend on the selected procedure (Simple Docking, Off-Target Docking or Scoring). Please, see the TUTORIAL section for more information. A *.amdock file is generated and placed in the working directory. This file contains the summarized data of the docking procedure.

-Preparing Input Files

For Protein: Cartesian coordinates of protein are needed. These can be taken from the protein file or a protein-ligand complex. Coordinates usually come from X-ray crystallography, NMR spectroscopy or model-building. AMDock deals with one of the three formats listed below:

*.pdbqt - This means you have prepared protein input file by yourself and AMDock will not perform any modification. This is recommended only if you know the minimum procedure to generate a proper *.pdbqt file.

*.pdb; *.ent - Files in these formats are processed by AMDock in order to generate a proper .pdbqt file (prepare_receptor4.py). AMDock will perform several tasks for you:

- ✓ remove heteroatoms. That include water, ions, crystallization reagents, ligands. *Important! Even when ligands atoms are removed from the protein-ligand file, ligand coordinates are stored and can be used to define a search space (see Tutorial II)*
- ✓ delete alternates (prepare_receptor4.py)
- ✓ determine protonation states (PDB2PQR v. 2.01)
- ✓ complete missing side chains (PDB2PQR v. 2.01) (*Important! This software is not able to model missing residues. These residues can be modeled with external software, i.e. MODELLER*)
- ✓ merge charges and remove non-polar hydrogens (prepare_receptor4.py)

For Ligand: Cartesian coordinates of the ligand are needed. These usually come from X-ray crystallography, NMR spectroscopy or model-building. AMDock deals with one of the three formats listed below:

*.pdbqt - This means that you have prepared protein input file by yourself and AMDock will not perform any modification. This is recommended only if you know the minimum procedure to generate a proper *.pdbqt file.

*.pdb; *.mol2 - Files in these formats are processed by AMDock in order to generate a proper .pdbqt file (prepare_ligand4.py). AMDock will perform several tasks for you:

- ✓ determines protonation states (OpenBabel v. 2.4.1)
- ✓ merge charges and remove non-polar hydrogens (prepare_ligand4.py)

As can be seen, AMDock should be able to handle successfully most protein and ligand files. Even so, is always recomandable checking the input files and remove unnecessary ions, solvent, cofactors; check for "connects" on ligand files.

-*Defining a Search Space*

A search space is defined when the coordinates of the geometrical center and the dimensions of the box have been defined. There are multiple options in AMDock for defining the search space (box's center and dimensions). These options are listed below and comprise from *automatic definition* to *user guide definition* of the search space.

Automatic: a potential ligand-binding site is identified and characterized by using AutoLigand tool. The AutoLigand code generates an object ("FILL") with the ligand's dimensions. Then, a box with optimal dimensions (see **Optimal Box Size 1.1**) is placed on the geometric center of this object. *This option is recommendable only when you have no idea where the binding site is*.

Center on Residue(s): a box with optimal dimensions (see **Optimal Box Size 1.1**) is placed on the geometric center of this/these residue(s). *This option is recommendable when you know which residue(s) is/are located at the binding site. This information usually comes from mutagenesis experiments, comparing proteins belonging to the same family, etc.*

Center on Ligand: A box with optimal dimensions (see **Optimal Box Size 1.1**) is placed on the geometric center of the ligand. This option will be available only if a protein-ligand complex was used as the protein input file. *This option is recommendable when you are interested in docking ligands belonging to the same family or ligands with the same binding mode.*

Box: Box's center and dimensions are defined by the user. *This option is usually preferred by expert users. Remember, the box's center and dimensions should be as proper as possible in order to ensure a suitable search space.*

-Run Docking

Running time will depend on several factors such as: number of poses, number of rotatable bonds of ligand, etc., although AutoDock Vina tends to be faster than AutoDock 4 by orders of magnitude.

After simulation ends, you will get automatically into "Results Analysis" tab. There, you will observe a summarizing table with Binding Energies, Estimated K*i* values, and Ligand Efficiencies. Besides, by pressing "Show Best Pose" button you will visualize in PyMOL the complex between the receptor and the ligand pose with the lowest energy. Likewise, by pressing "Show All Poses" button you will visualize in PyMOL the complex between the receptor and all ligand poses. You should keep in mind that the view is created automatically by a PyMOL script. Since it is automatic, the view cannot be in the most informative angle.

AMDock

Tutorials

TUTORIALS

BEFORE BEGINNING…

Here we describe how AMDock can be used to predict the protein-ligand complex structure from the coordinates of receptor and ligand separately. We are interested in showing only the AMDock procedure. Thus, we avoid discussions about the advantages and disadvantages of docking programs. We used one or another program for docking purposes at our convenience as well as structure files used in previous tutorials. All information about these programs can be reviewed in their respective manuals and original publications. Reading "AMDock WORKFLOW" section is encouraged before starting the tutorials.

SIMPLE DOCKING

Some applications and plugins have been designed for helping researchers in performing molecular docking simulations under a uniform and user-friendly graphical interface. These applications have been mainly focused on Autodock Vina, Autodock or both of them. To our knowledge, Autodock4$_{Zn}$ force field has not been incorporated in any of those applications. Therefore, we present a tutorial for Autodock4$_{Zn}$ in AMDock, although this procedure is valid for the rest of the docking engines included in AMDock.

This tutorial is based on a previous one available here. In the Autodock4$_{Zn}$ tutorial the authors selected as a study case the human farnesyltransferase (hFTase) complexed with the inhibitor L-778,123 (PDB ID: 1S63).

Receptor (hFTase.pdb) and ligand (L-778_123.pdb) files are available in ./Doc/Tutorials/I_Simple_Docking folder. The crystal structure of the complex (complex.pdb) is available as well for comparison purposes. As the first tutorial, we chose the easiest way to perform a docking simulation with AMDock. Let's begin!

1. Open AMDock program

2. Select Autodock4$_{Zn}$

   **hFTase is a Zn-metalloenzyme. Autodock4$_{Zn}$ includes a specialized potential describing the interactions for zinc-coordinating ligands.*

3. Set the Project Name (default Docking_Project)

4. Set the Location for Project

5. Check that “Simple Docking” checkbox is selected

6. Choose the receptor file (./Doc/Tutorials/ I_Simple_Docking /hFTase.pdbqt)

**In this case, we are using a .pdbqt file. That means AMDock will no perform any change in your file. A warning message will pop up because of the receptor file we are using contains a “ligand”. We will dismiss this warning message since we interested in keeping this ligand in the receptor structure. You can check later on what happens if we remove this cofactor from the structure.*

7. Choose the ligand file (./Doc/Tutorials/ I_Simple_Docking /L-778_123.pdb)

8. Press “Prepare Input” button

9. Then, for defining a search space, pick “Box” and set the box center (x = 18; y = 134; z = -1) and size (15; 11; 19)

Autodock4$_{Zn}$ is based on Autodock4. In AutoDock4, the search space size is specified by "grid points" (0.375 Angstrom). Thus, if you look into Autodock4$_{Zn}$ tutorial, the search space sizes (npts) are 40, 30, and 50. The AutoDock Vina search space sizes are given in Angstroms instead. Because both programs are included in the AMDock environment, we decided to standardize the space size definition.* ***AMDock defines the search space size in Angstroms**. Thus, a box with npts = 40, 30, 50 in Autodock4$_{Zn}$ tutorial, corresponds to a box with the following dimensions in Å units: 15Å x 11Å x 19Å. These dimensions have been calculated by multiplying npts x 0.375Å.*

Note: A warning message will pop up since AMDock estimates automatically an “ideal” size for the box based on ligand’s radius of gyration (see ***Optimal Box Size 1.1****). We can dismiss the message in this case since we know in advance the box dimensions.*

10. Press “Define Search Space” button

11. Press “Run Docking” button

12. When docking simulation ends, “Results Analysis” tab will appear automatically. There, you will observe a summarizing table with Binding Energies, Estimated Ki values, and Ligand Efficiencies.

13. Additionally, you can check the ligand pose with the lowest energy and complexes with all ligand poses.

14. Open the structure of the crystal structure (complex.pdb) and see that the docking engine we are using is able to reproduce the binding pose accurately.

15. If we perform the docking using only the receptor structure (without the ligand), we can’t reproduce the experimental pose since this ligand is essential for the inhibitor binding.

OFF-TARGET DOCKING

For this tutorial, we select as a study case the COX-1/2 system. These enzymes are blockaded by nonsteroidal anti-inflammatory drugs (NSAIDs) which are widely used as therapeutic agents for the treatment of pain and inflammation. The classical NSAIDs such as aspirin, ibuprofen, or flurbiprofen inhibit indifferently all the COXs isoforms. However, scientists are interested in the selective inhibition of COX-2 since the inhibition of COX-1 may lead to dangerous side effects. Structural analysis by Kurumbail *et al*. highlighted a selectivity pocket that is accessible in COX-2 but not in COX-1 because of the V523I substitution. Over the years, this V523I difference has been exploited to design inhibitors with exquisite selectivity for COX-2 relative to COX-1. Here, we were focused on one of the most potent and selective COX-2 inhibitors, SC-558 (IC_{50} = 9.3 nM).

Receptors (COX-1.pdb and COX-2.pdb) and ligand (SC-558.pdb) files are available in ./Doc/Tutorials/II_Off-Target_Docking folder. The structure of COX-2 complexed with SC-558 (complex.pdb) is available as well for comparison purposes.

1. Open AMDock program

2. Select Autodock Vina

3. Set the Project Name (default Docking_Project)

4. Set the Location for Project

5. Check that “Off-Target Docking” checkbox is selected

6. Choose the target receptor file (./Doc/Tutorials/II_Off-Target_Docking /COX-2.pdb)

**We use as COX-2 entry, a complex with celecoxib, an analog of SC-558 (Figure 2). Both ligands are quite similar, so, we presume they have a similar binding mode to COX-2.*

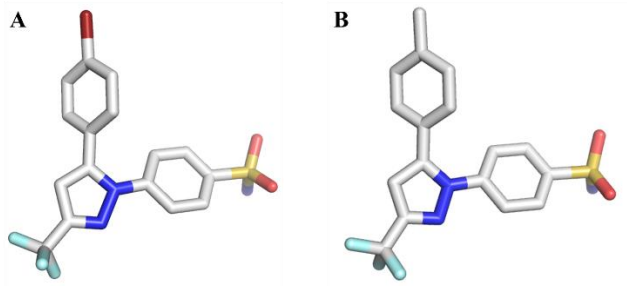


Figure 2. Chemical structure of COX-2 inhibitors. A) SC-558 B) Celecoxib

7. Choose the off-target receptor file (./Doc/Tutorials/II_Off-Target_Docking /COX-1.pdb)

**It is noteworthy to mention that both, the target and control protein, should be structurally aligned. This will be critical to define a similar search space in both proteins. AMDock performs this task automatically once both receptors have been defined. However, the method used to align the proteins is quite simple. So, it is mandatory to check the quality of the alignment. In general, both proteins should be similar, hence we should expect a low value of RMSD.*

8. Choose the ligand file (./Doc/Tutorials/II_Off-Target_Docking /SC-558.pdb)

9. Press "Prepare Input" button

10. Then, to define a search space for COX-2, press "Center on Ligand" and select "CEL".

**When a complex is used as entry, the ligand atoms are removed. However, the geometric center of the ligand is calculated and stored in memory. Thus, we can use it to define the box's center.*

11. Then, to define a search space for COX-1, press "Center on Ligand" and select any of those ligands. At this point, we can build the box in an approximate manner and then we can fix it later by checking the box center and size defined for the target protein, *i.e.* COX-2. Follow these steps:

11.1 Press "Show in PyMol" in the COX-2 column

11.2 Press "AMDock" button on PyMol's toolbar and then "AMDock Box Builder"

11.3 Automatically, you will see the box dimensions (20;20;20) and the center (x = 224.6; y = 22.5; z = -54.3)

11.4 Press "Reset" button in the COX-1 column and define manually the search space by picking "Box"option

12. Press "Define Search Space" button

12. Press "Run Docking" button

13. When docking simulation ends, "Results Analysis" tab will appear automatically. There, you will observe a summarizing table with Binding Energies, Estimated Ki values, and Ligand Efficiencies.

14. As predicted, similar binding poses were observed for both ligands, SC-558 and CEL with COX-2. In addition, a higher affinity was observed for SC-558 when binding COX-2 which is in agreement with experimental data.

SCORING

In this tutorial, we will study the effect of point mutations over ligand recognition by using the SCORING module of AMDock. We have selected as study case the human farnesyl-transferase (hFTase), the same protein we used in the first tutorial. However, we will focus now on substrate (farnesyl diphosphate, FPP) recognition. The hFTase is a prenyl-transferase. It transfers a 15-carbon isoprenoid (donated by FPP) to the cysteine of a C-terminal motif of substrate proteins. Prenyl-transferases are generally highly selective for their cognate isoprenoid diphosphate substrates. Thus, hFTase binds FPP while the human geranylgeranyl-transferase type I (hGGTase-I, another prenyl-transferase) binds geranylgeranyl diphosphate (GGPP). The 20-carbon GGPP binds to hFTase with nanomolar affinity (but as a competitive inhibitor), while the 15-carbon FPP is a poor substrate for hGGTase-I (Figure 3).

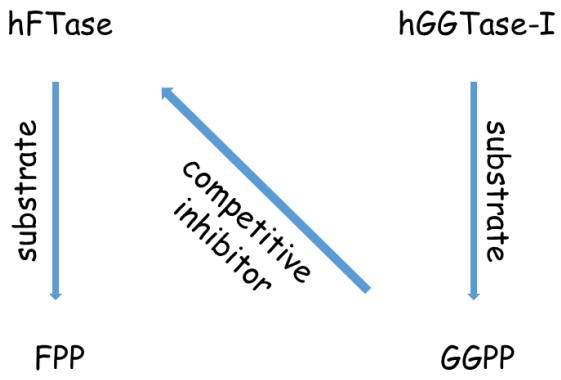


Figure 3. Diagram showing the relationship between enzymes and their cognates substrates and inhibitors.

The placement of GGPP-CaaX in the corresponding region of hFTase, leads to a steric clash with W102 and Y365, suggesting that one or both of these residues may be key determinants of isoprenoid specificity (Figure 4). In elegant work, Terry *et al.*,[1] re-engineered hFTase through site-directed mutagenesis to create an hFTase able to recognize GGPP substrate and geranylgeranylate its native CaaX substrates.

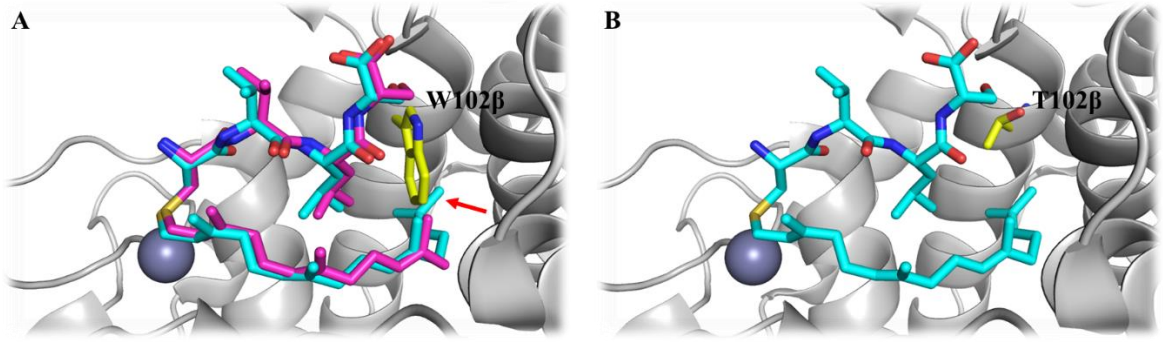


Figure 4. (A) Superposition of FPP-CVLS (magenta) and GGPP-CVLS (cyan), in wild-type hFTase active site. Red arrow indicates the steric clash between GGPP and W102β residue in the hFTase active site (B) GGPP-CVLS (cyan), in mutant hFTase W102βT active site. The mutation W102βT allows GGPP to fit into the active site of hFTase.

In this tutorial, we will try to predict the binding properties of both variants (the wild-type and the re-engineered) with the AMDock SCORING module.

> *Remember, the SCORING module will not generate a new complex from those of the ligand and receptor coordinates. Essentially, this module calculates the binding energy between two molecules belonging to a complex by using the scoring function of the selected docking engine. So, the ligands and receptors coordinates we will use as input were extracted from those of the complexes X-ray structures (PDB IDs: 2H6F and 2H6G).*

Let's begin with the wild-type first!

1. Open AMDock program

2. Select Autodock4$_{Zn}$

3. Set the Project Name (default Docking_Project)

4. Set the Location for Project

5. Check that "Scoring" checkbox is selected

6. Choose the receptor file (./Doc/Tutorials/III_Scoring/hFTase.pdb)

7. Choose the ligand file (./Doc/Tutorials/III_Scoring /FPP_CVLS.pdb)

8. Press “Prepare Input” button

9. As we said above, this module will not generate a new complex; hence, it is not necessary to define a search space. Press “Run Scoring” button then.

The predicted binding constant for the farnesylated CaaX peptide is 28nM.

Let’s continue with the mutant…

1. Open AMDock program

2. Select Autodock4$_{Zn}$

3. Set the Project Name (default Docking_Project)

4. Set the Location for Project

5. Check that “Scoring” checkbox is selected

6. Choose the receptor file (./Doc/Tutorials/III_Scoring /hFTase_W102T.pdb)

7. Choose the ligand file (./Doc/Tutorials/III_Scoring /GGPP_CVLS.pdb)

8. Press “Prepare Input” button

9. Press “Run Scoring” button.

Here, the predicted binding constant for the geranylgeranylated CaaX peptide is 9.9nM which is, as expected, very similar to that of farnesylated CaaX peptide (28nM).