Development, optimization, and integration of molecular fitting tools and models in UCSF Chimera

Desarrollo, optimización e integración de herramientas y modelos de ajuste molecular en Chimera

Pablo Solar Rodríguez
Universitary Master's Degree in BioInformatics and BioStatistics
BioInformatics – Structural and Computational Biology

Brian Jiménez García
Pablo Chacón – José Ramón López Blanco

24th May 2017

LICENCIA (LICENSE)

GNU Free Documentation License (GNU FDL)

FINAL WORK SHEET

Título del trabajo:	*Development, optimization and integration of molecular fitting tools and models in Chimera*
Nombre del autor:	*Pablo Solar Rodríguez*
Nombre del consultor/a:	*Brian Jiménez García*
Nombre del PRA:	*M. Jesús Marco Galindo*
Fecha de entrega (mm/aaaa):	05/2017
Titulación::	*Máster Universtario en BioInformática y BioEstadística*
Área del Trabajo Final:	*BioInformática – Biología Computacional y Estructural*
Idioma del trabajo:	*Inglés*
Palabras clave	*Chimera, fitting, computational biology, structural biology, plugin, ADP_EM, iMODFIT*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El TFM se encuadra en el campo de la bioinformática estructural y se centra en el desarrollo de herramientas computacionales para interpretar información 3D de proteínas y ácidos nucleicos proveniente de distintas fuentes experimentales como son la crio-microscopía electrónica (*EM*) y la cristalografía de rayos X.

El ajuste macromolecular o fitting es la forma estándar de interpretar la información contenida en un mapa de crio-microscopía electrónica de una determinada macromolécula con las estructuras atómicas disponibles de sus componentes. Se trata de un complicado puzzle donde se encajan estructuras a resolución atómica dentro de un mapa de densidad electrónica a menor resolución. Aunque existen diversas aproximaciones para llevar a cabo el ajuste molecular, son herramientas en desarrollo que es necesario perfeccionar y adaptar al usuario (López-Blanco/Chacón 2015).

El objetivo general de este TFM es la mejora y adaptación de los métodos de ajuste desarrollados por el grupo receptor del doctor P. Chacón, así como su integración dentro del programa UCSF Chimera a través de la realización de unos plugins. Concretamente, las herramientas *ADP_EM*, *Situs* e *iMODFIT*.

Al final, se obtendrán una serie de plugins para Chimera desarrollados en Python para las diferentes herramientas y métodos de ajuste molecular desarrollados por el grupo, que se harán públicos a la comunidad científica.

Abstract (in English, 250 words or less):

This project lies within the field of structural biology and will be focused on computational tools development to integrate 3D information of proteins and nucleic acids from different experimental sources such as cryo-electron microscopy (*EM*) or X-ray crystallography.

Fitting is the standard way of interpreting the information contained in cryo-electron-microscopy maps of macromolecular structures by means of the available atomic structural components. Multiresolution fitting is a complicated jigsaw puzzle in which the low-resolution 3D *EM* density map of a macromolecule complex acts as a fuzzy frame to guide the assemblage of interlocking atomic-resolution pieces. Although there are several approaches to perform fitting, those are tools in development phase that needs to be improved and better adapted to the users (López-Blanco, J. R. and Chacón, P. (2015)).

The main objective of this project is the improvement and adaptation of these fitting tools developed by P. Chacón's team, as well as its integration in UCSF Chimera program. Specifically, *ADP_EM*, *Situs* e *iMODFIT* tools.

At last, some Chimera plugins will be obtained based on the different fitting tools and models developed by the group.

Index

List of Figures

1. Overview

1.1 Project Context and Justification

Advances in current biology and medicine depend on understanding the actions and interactions of large molecular complexes. The characterization of these macromolecules can only be approached with the coordinated application of different complementary experimental techniques.

The hybrid methods allow combining computationally and in an automatic and reproducible way the structural information provided by the experimental techniques. It is a challenge in how computational methods would assist on characterizing, at atomic level of the different functional states of the macromolecules in solution and, therefore, to understand the molecular mechanisms of the main actors of the different biological functions.

In this context, one of the most interesting and fruitful fields of current structural bioinformatics focuses on the development of different methodologies for integrating structural information into different resolutions. The relevance and timeliness of this field has aroused my interest. I think it is an ideal framework to apply the knowledge acquired in the Master and it will guide my professional training towards the field of research in one of the most powerful research groups of this area at national and international level, like Pablo Chacón's group.

Thus, according to everything described above, this project lies within the field of structural biology and will be focused on computational tools development to integrate 3D information of proteins and nucleic acids from different experimental sources such as cryo-electron microscopy or X-ray crystallography.

Fitting is the standard way of interpreting the information contained in electron-microscopy (*EM*) maps of macromolecular

structures by means of the available atomic structural components. Multiresolution fitting is a complicated jigsaw puzzle in which the low-resolution *3D EM* density map of a macromolecule complex acts as a fuzzy frame to guide the assemblage of interlocking atomic-resolution pieces. Although there are several tools to perform fitting, they are in development phase that need to be improved and better adapted to the users (López-Blanco, J. R. and Chacón, P. (2015)).

The main objective of this project is the improvement and adaptation of these fitting tools developed by P. Chacón's receiving group, as well as its integration in UCSF Chimera program. Specifically, *ADP_EM*, *Situs* and *iMODFIT* tools.

At last, some Chimera plugins will be obtained based on the different fitting tools and models developed by the group and published to the scientific community (dissemination).

1.2 Project Objectives

1.2.1 General Objectives

Two main objectives will be defined in this project: training objective and fitting tools development objective. This section describes the objectives from a global and conceptual perspective.

The training objective will serve to get familiarized with the computational techniques of structural integration, specifically the fitting methods developed by the receiving team.

Once the brief training period has been completed, the main and most important objective would be the development, improvement, and adaptation of the molecular fitting methods developed by the receiving group (Lopéz-Blanco, J. R. and P. Chacón (2013), Chacón, P. and W. Wriggers (2002), Garzón, J. I., J. Kovacs, R. Abagyan and P. Chacón (2007)), like *ADP_EM*, *iMODFIT,* or *Situs*.

1.2.2 Specific Objectives

Training Objectives

1. Familiarization with theory and software of molecular fitting tools developed by the group, particularly with packages:

   - *Situs (http://situs.biomachina.org)*

   - *ADP_EM (http://chaconlab.org/methods/fitting/adpem)*

   - *iMODFIT (http://chaconlab.org/methods/fitting/imodfit)*

2. Familiarization with other bioinformatics fitting approaches like:

   - *mdff (http://www.ks.uiuc.edu/Research/mdff/)*

   - *gEMfitter (http://gem.loria.fr/gEMfitter/)*

   - *Integrative Modeling (https://integrativemodeling.org/)*

   - *Rosseta (https://www.rosettacommons.org/)*

Development Objectives

1. Development, improvement, optimization, and the integration of *ADP_EM*, *iMODFIT,* and *Situs* into UCSF Chimera. It is intended to improve de efficiency and usability of these tools trough code improvements *(C* and *C++*) and, specifically, develop Chimera **plugins** in *Python* for Chimera program, which is the most used visualization tool in the field of electron microscopy.

   o *ADP_EM* is a priority while *iMODFIT* and *Situs* plugins would be implemented as long as the project's timing and needs allow.

3

2. Depending on the project's timeline progress, integration of other molecular fitting programs such as *geMfitter* and *COLORES* (S*itus*) will be explored.

3. At last, the obtained Chimera plugins will be properly released to the scientific community.

1.3 Approach and Methods

The field of application is such a specific field that requires prior training. For this reason, in a first stage a strategy has been combined into a brief training to understand and get familiar with the problem of molecular fitting with the introduction to the approaches used in the field. This stage will also be useful to identify which tools could be improved and the way to do it.

In the second stage, the fitting tools will be developed and improved through computational knowledge based in *C/C++* and *Python*.

The technologies, software, and hardware required and used to perform both stages will be detailed in the following sections.

1.3.1 Software and Technology

- Products and plugins will be developed in Python mostly which is the base programming language where Chimera is built. Python version will be the latest and updated version supported by the development computers and frameworks. Syntactic analysis will be used to ensure the quality of the code created.

- It may be necessary the use of *C* and *C++* to create shared libraries and pipes that allow a bidirectional communication between Chimera plugins and the fitting processes like

ADP_EM or *iMODFIT*. This point is open due to variations in development requirements.

- *C* and *C++* compilers that will be used for the fitting tools processes will be *Intel Parallel Studio X*E which offers, apart from speed and management compilation, some useful *APIs* like Fourier transforms that are necessary in this project, as it will be seen in later chapters.

- Changes and versions of products during the development phase will be handled through *Git* repository in the internal intranet that the receiving group is actually using for their works.

- Chimera plugins development, the main objective of this project, use *Python* in order to be dynamically used by users. This is, that one user can load a plugin in Chimera from any folder location, i.e. independently of the administrative permissions.

- All products obtained will a simple, friendly user guide with detailed installation instructions.

- All products obtained will be published to the public trough the official media of the receiving group. In addition, they will be deployed in an online repository like *GitHub* so that they are use as much as possible.

  - It is also intended to contact the official developers of Chimera to include the products natively in the program, if they consider them fit and robust enough.

- In the first stage of development, plugins will be coded for Linux and Mac OS X, trying to adapt them to Windows platforms too, but this will depend on project deadlines, requirements and changes.

- Plugins developed should be compatible with previous and future versions of Chimera, as long as Chimera standards do not change.

- Since the plugins will handle several atomic structures in form of *PDBs* and *EM* density maps of biological macromolecules, a developed code quality will be necessary to ensure a good flow and memory management, as well as the own session of Chimera. It will also be necessary to take care of the computational cost and the complexity of the code developed in the products.

- A priori it is not possible to establish a usual testing strategy for the plugins developed since it is necessary to restart Chimera every time modifications are to check them. Thus, two phases of testing can be established:

  - Development phase testing: basic and simple tests about the correct functionality of different requirements that are demanded to the plugins.

  - Post-development phase testing: overload and stress testing with real data that will check the performance of the plugins in professional tasks.

- Products and plugins obtained should be developed with quality enough to be improved in a future with new functionalities in a simple, fast, secure and robust way.

- Support and development will be in constant supervision and change.

1.3.2 Hardware

This project will be carried out in a MacBook Pro 15" with the following characteristics:

Processor

CPU speed: 2.3 GHz
CPU Type: Quad Core i7
Cores: 4
Bus Speed: 5 GT/s DMI
Cache: 6 MB L3 cache
64-bit: Yes
Turbo Boost: Up to 3.3 GHz

Memory

Installed RAM: 8 GB (2x4GB)
Max. Amount: 8.0 GB
Nr. of Slots: 2
RAM Speed: 1600 MHz
RAM Type: DDR3, SDRAM

Storage and Media

Hard Drive: 500 GB, 5400 rpm.
Drive Brand: Hitachi or Toshiba
Drive Bus: Serial-ATA
Optical Drive: This unit has an 8x SuperDrive built in.
Optical Bus: Serial-ATA
Other Media: -

Graphics

Display Size: 15.4-inch
Graphics Card: NVIDIA GeForce GT 650M
Intel HD Graphics 4000
Card Memory: 1 GB (GT650)
Max. Resolution: 1440 x 900
BLU / Coating: LED TFT, Glossy.

Networking

AirPort: Built-in Airport Extreme (802.11 a/b/g/n).
Ethernet: 10/100/1000BASE-T (RJ-45)
Bluetooth: Built-in Bluetooth 4.0
Infrared: For use with Apple Remote only.
Modem: None

Operating System and Software

Original OS: 10.7.4 Lion
Maximum OS: Latest release of Mac OS X
Minimum OS: OS X 10.7.3 Build 11D2097

1.4 Project Planning

The planning has been subject to changes that were a consequence of variations or modifications suffered by the different products that were developed, either by internal or external requirements.

In this section, tasks and its deadlines variations, calendars, *Gantt* charts and project milestones will be exposed.

1.4.1 Initial Tasks

1. Training with molecular fitting theory and software developed by the receiving group:

   a. *Situs* → 2 days

   b. *ADP_EM* → 14 days

   c. *iMODFIT* → 3 days

2. Familiarization with other bioinformatics fitting approaches:

   a. *gEMfitter* → 1 day

   b. *mdff* → 1 day

   c. *Integrative Model* → 1 day

   d. *Rosetta* → 1 day

3. Improvement and adaptation of *ADP_EM* and *iMODFIT*. Chimera plugins development:

   a. *Chimera* plugins *definition* → 4 days

   b. *ADP_EM Chimera* plugin *GUI* → 5 days

   c. *ADP_EM fitting process in Chimera* → 8 days

   d. *ADP_EM* plugin *tests and results* → 6 days

4. Development of other approaches like *iMODTFIT* or *Situs:*

      a. *Subject to availability and changes* → 10 days

         i. if it not possible, *ADP_EM* plugin will be perfected

5. Project Memory creation and presentation → 10 days

      a. *Project Memory creation* → 7 days

      b. *Project presentation* → 5 days

1.4.2 Final Tasks

1. Training with molecular fitting theory and software developed by the receiving group:

      a. *Situs* → 1 day

      b. *ADP_EM* → 17 days

      c. *iMODFIT* → 3 days

2. Familiarization with other bioinformatics fitting approaches:

      a. *gEMfitter* → 0,5 day

      b. *mdff* → 0,5 day

      c. *Integrative Model* → 0,5 day

      d. *Rosetta* → 0,5 day

3. Improvement and adaptation of *ADP_EM* and *iMODFIT*. Chimera plugins development:

      a. *Chimera* plugins *definition* → 4 days

      b. *ADP_EM Chimera* plugin *GUI* → 8 days

      c. *ADP_EM fitting process in Chimera* → 12 days

      d. *ADP_EM* plugin *tests and results* → 9 days

4. Development of other approaches like *iMODTFIT* or *Situs* and project memory and presentation*:*

> a. *Subject to availability and changes* → 10 days
>
> > i. if it not possible, *ADP_EM* plugin will be perfected
> >
> > ii. *Project Memory creation* → 7 days
> >
> > iii. *Project presentation* → 5 days

1.4.3 Initial Tasks Timing

Stage	Task	Step	Step Days	Task Days	Stage Days	Total Days
Training	1	a	2	19	23	66
		b	14			
		c	3			
	2	a	1	4		
		b	1			
		c	1			
		d	1			
Chimera Development	3	a	4	23	33	
		b	5			
		c	8			
		d	6			
	4	-	10	10		
Memory and Presentation	5	a	7	10	10	
		b	5			

1.4.4 Final Tasks Timing

Stage	Task	Step	Step Days	Task Days	Stage Days	Total Days
Training	1	a	1	21	23	66
		b	17			
		c	3			
	2	a	0,5	2		
		b	0,5			
		c	0,5			
		d	0,5			
Chimera Development	3	a	4	33	33	
		b	8			
		c	12			
		d	9			
Other approaches, Memory and Presentation	4	a	10	10	10	

1.4.5 Project Calendar Comparison
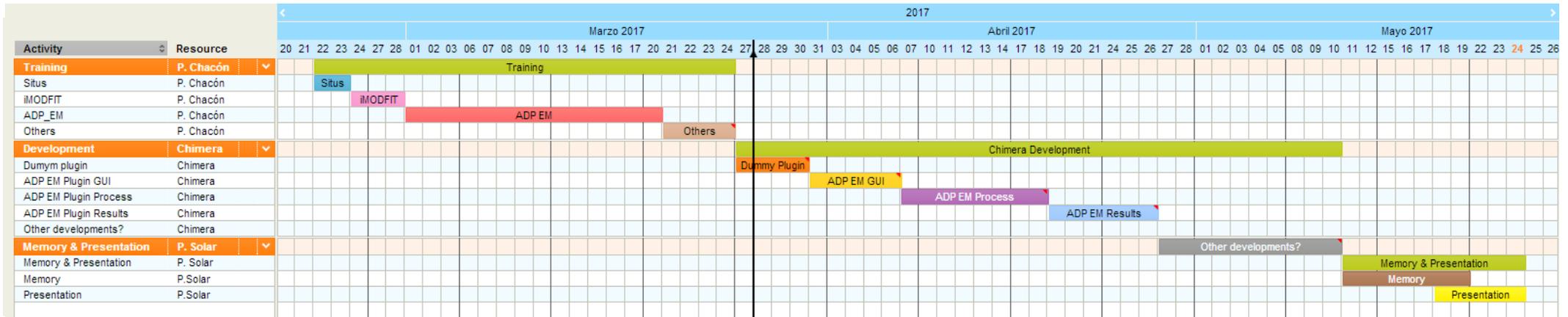
Figure 1: Initial Project Calendar.
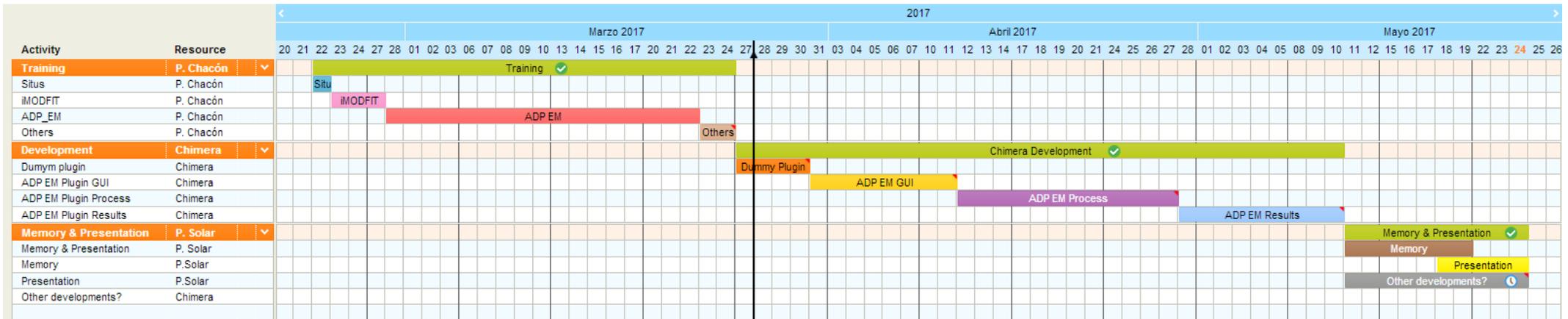
Figure 2: Final Project Calendar.

1.4.6 Project Milestones

Project Starting	**22.02.2017**
Training Stage Starting	**22.02.2017**
Situs	22.02.2017-22.02.2017
iMODFIT	23.02.2017-27.02.2017
UOC PEC1 Starting	*01.03.2017*
ADP_EM	28.02.2017-22.03.2017
UOC PEC2 Ending	*15.03.2016*
Other approaches	23.03.2017-24.03.2017
Training Stage Ending	**24.03.2017**
UOC PEC2 Starting	*16.03.2017*
Chimera Development Stage Starting	**27.03.2017**
Chimera plugins definition	27.03.2017-30.03.2017
ADP_EM Chimera plugin *GUI*	31.03.2017-11.04.2017
UOC PEC2 Ending	*05.04.2017*
UOC PEC3 Starting	*06.04.2017*
ADP_EM fitting process in Chimera	12.04.2017-27.04.2017
ADP_EM plugin tests and results	28.04.2017-10.05.2017
Chimera Development Stage Ending	**10.05.2017**
UOC PEC3 Ending	*10.05.2017*
Others, Memory and Presentation Stage Starting	**11.05.2017**
UOC PEC4 Starting	*11.05.2017*
Other approaches	11.05.2017-24.05.2017

Memory	11.05.2017-19.05.2017
Presentation	18.05.2017-24.05.2017

Others, Memory and Presentation Stage Ending   **24.05.2017**

UOC PEC4 Ending   *24.05.2017*

1.4.7 Risk Analysis

During the development phase, some risks had to be taken into account that could have affected the achievement of the project and its final outcome:

- Changes/modifications of the theoretical approaches of the processes involved: methods, fittings, etc. E.g., introducing a new set of theoretical basis into the fitting methods that were initially not taken into account.

- Changes/modifications of the practical approaches of the processes involved. E.g., introducing new GUI functionalities in the *ADP_EM* plugin that were initially not taken into account.

- Changes/modifications in the different tools used during the development of the project. E.g., a new version of Chimera that might have caused compatibility issues and forced to redo some parts of the project.

- Computational limitations of the different used resources. E.g., different tests during the testing phase might have had a high computational cost and caused a delay this stage.

- Limitations of Chimera that have entailed a rethinking and/or delay before not foreseen. For example, modifications made to different plugins required restarting Chimera every time to check them. This affected directly to the development stage.

Therefore, it is reiterated that the project, and specifically during its development phase, was subject to all kinds of changes, both practical and theoretical, that were resolved as progress was made in the achievement of the different products.

1.5 Brief Summary of the Products Obtained

Training Stage

- In-depth level knowledge, theoretical and practical, as well as different application situations of the *ADP_EM* molecular fitting model.

- Medium level knowledge and practical application of the *iMODFIT* molecular fitting model.

- General knowledge and practical application of the *Situs* molecular fitting model.

- Basic knowledge and practical application of the *gEMfitter* molecular fitting model.

- Basic knowledge and practical application of the *mdff* molecular fitting model.

- Basic knowledge and practical application of the *Rosetta* molecular fitting model.

Development Stage

- A complete *dummy* Chimera plugin without any kind of functionality, developed in *Python*, which can be loaded and integrated in Chimera. In addition, this product can be used as a template to create new plugins in Chimera.

- A complete Chimera plugin for the *ADP_EM* molecular fitting model and process with the following functionalities:

  o Ability to host the plugin in any user folder and allow its loading inside Chimera dynamically, regardless of the chosen directory.

    ▪ The plugin consists of a directory with the necessary *Python* files to be loaded in Chimera through a native loading dialog.

    ▪ An executable *C*-version of *ADP_EM* to run the process inside the plugin in Chimera.

  o Ability to load density maps and atomic structures from any location, allowing independent operation of the plugin and not forcing the user to have to host these resources within the plugin directory.

  o Graphic User Interface (*GUI*) with all the options needed to execute *the ADP_EM* process.

    ▪ This *GUI* is capable of interacting with other native modules and plugins already existing in Chimera, such as the *Volume Viewer* module that is related with density maps.

  o Validation of all the inputs introduced by the user in the *GUI*.

  o Calling to *ADP_EM* with all the inputs validated to perform exhaustive fitting calculations.

  o Generation of all the fitting solutions from the *ADP_EM* binary output (6-dimensions = 3 rotational + 3 translational for each solution).

  o Visualization of the different solutions in Chimera, directly accessible from the plugin's own *GUI*. In addition, the user

can interact with the solutions and save them for later analysis.

- An executable *C* version of the *ADP_EM* molecular fitting tool optimized and adapted to be used in Chimera. This version is included inside the own directory of the plugin.

  o Any changes that have to be made in the Chimera plugin will not affect the fitting tool itself and vice versa, any modifications that have to be made to the *ADP_EM* process will not affect the plugin.

  o This *ADP_EM* version is capable of communicating bidirectionally with Chimera in real time effectively.

- A complete Chimera plugin for the *iMODFIT* molecular fitting model and process.

  o Ability to host the plugin in any user folder and allow its loading inside Chimera dynamically, regardless of the chosen directory.

    ▪ The plugin consists of a directory with the necessary *Python* files to be loaded in Chimera through a native loading dialog.

    ▪ An executable *C*-version of *iMODFIT* to run the process inside the plugin in Chimera.

  o Ability to load density maps *EM* and atomic structures from any location, allowing independent operation of the plugin and not forcing the user to have to host these resources within the plugin directory.

  o Graphic User Interface (*GUI*) with all the options needed to execute *the iMODFIT* process.

    ▪ This *GUI* is capable of interacting with other native modules and plugins already existing in Chimera,

17

such as the *Volume Viewer* module that is related with maps of density or *MD Movie*.

- o Validation of all the inputs introduced by the user in the *GUI*.

- o Calling to *iMODFIT* with all the inputs validated to perform exhaustive fitting calculations.

- o Generation of all the fitting solutions from the *iMODFIT* process raw files.

- o Visualization of the different solutions in Chimera, directly accessible from the plugin's own *GUI*. In addition, the user can interact with the solutions and save them for later analysis.

- An executable *C* version of the *iMODFIT* molecular fitting tool, optimized and adapted to be used in Chimera. This version is included inside the own directory of the plugin.

  - o Any changes that have to be made in the Chimera plugin will not affect the fitting tool itself and vice versa, any modifications that have to be made to the *iMODFIT* process will not affect the plugin.

  - o This *iMODFIT* version is capable of communicating bidirectionally with Chimera in real time effectively.

1.6 Short description of memory chapters

Below are described the theoretical and practical chapters related to this work, which will be further detailed in later sections:

- **Macromolecular Fitting**: this chapter puts the molecular fitting into context, why it is used, and what major experimental techniques exist.

  - **Rigid Fitting**: it explains the rigid fitting method, why and when it is used, what the underlying idea is, and what mathematical theories it applies (such *as Fast Rotational Matching* [*FRM*]) and introduces *ADP_EM*.

  - **Flexible Fitting**: it explains the flexible fitting method, why and when it is used, what the underlying idea is, what techniques it applies (such as *Normal Mode Analysis* [*NMA*]) and introduces *iMODFIT*.

- ***ADP_EM* Rigid Fitting Tool**: this chapter introduces the *ADP_EM* method with the mathematical theory on which it is based. In particular, it explains how the search space is rotationally accelerated using *Fast Rotational Matching* (*FRM*) with spherical harmonics along with a small example. In addition, it outlines the process workflow scheme.

- ***iMODFIT* Flexible Fitting Tool**: it details the *iMODFIT* method and the mathematical theory on which it is based. It explains how the *Normal Mode Analysis* (*NMA*) in *Internal Coordinates* (*IC*) is used and how the different graining models are applied. In addition, it outlines the process workflow scheme.

Finally, the *Results* section is presented, which details the plugins developed for the methods presented in these chapters along with all the associated features.

2. Macromolecular Fitting Tools

2.1 Macromolecular Fitting

Density maps obtained by electron microscopy can be interpreted using available atomic structures. By fitting these structures inside low/medium resolution maps it is possible to obtain quasi-atomic information in order to unravel the functioning of macromolecular complexes (Wriggers, W. and Chacon, P. (2001)).

When available atomic structures and maps are in the same conformation, it is enough to find the correct orientation between them using a rigid fitting strategy. However, in multiple cases the conformations observed in maps are significantly different from those ones that are in crystals. In these situations, it is necessary to employ a flexible fitting strategy to take into account the different conformations.
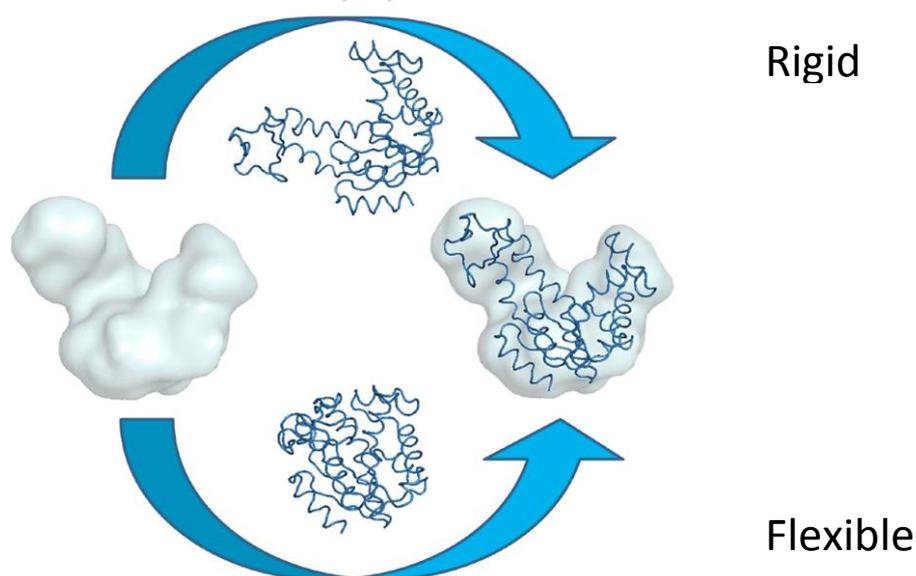

Rigid

Flexible

Figure 3: Rigid and flexible macromolecular fitting.

When the density map (left) and the atomic structure (above) have roughly the same conformation as the target map (right), the fit must be rigid. If the available atomic structure has a different conformation (below) to that of the target map, the setting must be flexible.

In this project, two macromolecular fittings are presented. The first one is rigid-based, *ADP_EM*, and the second one is flexible-based, *iMODFIT*.

2.1.1 Rigid Fitting

A number of high performance fitting programs have been developed over the last year to rigidly adjust atomic structures within density maps when the conformations of both are similar. To do this, there are several programs to carry out this task: *SITUS* (Wriggers, W., Milligan, R.A. y McCammon, J.A. (1999)), *EMFIT* (Rossmann, M.G. (2000)), *DOCKEM* (Roseman, A.M. (2000)), *FOLDHUNTER* (Jiang, W., Baker, M.L., Ludtke, S.J. y Chiu, W. (2001)), *COLORES* (Chacón, P. and W. Wriggers (2002)), *COAN* (Volkmann, N. y Hanein, D. (2003)), *3SOM* (Ceulemans, H. y Russell, R.B. (2004)) and *ADP_EM* (Garzon, J.I., Kovacs, J., Abagyan, R. y Chacon, P. (2007a)). In general, these tools perform an automated search of the all possible relative rotations and translations to maximize a scoring function. This score, typically a score correlation function, is calculated between the target experimental *EM* map and a simulated probe map of the atomic structure (Wriggers, W. and Chacon, P. (2001), Fabiola, F., Chapman, M.S. (2005)).

Despite its successful application, the exhaustive search performed by the majority of these docking tools is a very time-consuming process, and therefore they are not ready to support high-throughput fitting process. In this context, the set of possible positions between objects forms a vast search space. In the case of fitting only two objects, it is necessary to explore a 6-Dimensional space composed of a 3D translational space (spatial position of one object with respect to another defined by three Cartesian values) and a 3D rotational space (rotation of an object with respect to the another defined by three angular values). The simplest approach to this exploration (Figure 4), consisting of a systematic sampling of 6-Dimensional space, is impracticable if relatively small search intervals are used. Assuming a rigid fitting in which the rotational interval of 6°, for each translation it will be necessary to explore more than 10^5 rotations. If the maps also have dimensions of

100x100x100 cells or voxels, then the number of possible translations can be 10^6 (the center of one of the maps is superimposed on each voxel of the other). In summary, systematic sampling in this case may require the exploration of approximately one hundred billions of map positions relative to the other, with the consequent computational cost.
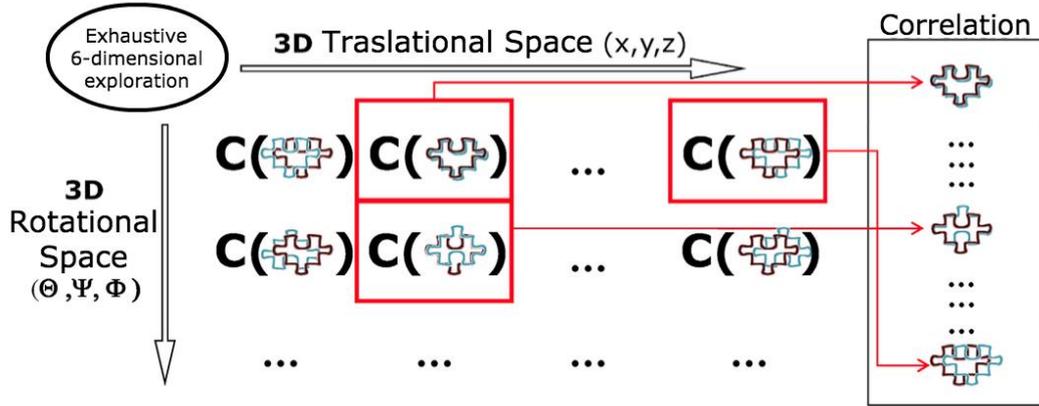

Figure 4: 6-Dimensional direct search.
The 6-Dimensional search discreetly explores the maximum number of possible positions and orientations of an object with respect to another. For each of these combinations the correlation between the two objects is calculated. In a final step, the possible solutions are ordered according to the correlation.

To increase efficiency, some methodologies have been developed to accelerate the search for some of the degrees of freedom that make up their space. The classical approach used to accelerate this search is to calculate the correlation in the frequency space. Through the combined use of the convolution theorem and *Fast Fourier Transform (FFT)* calculation techniques it is possible to accelerate the search in the translational space, requiring only a systematic exploration of the rotational space. Thus, for each rotation, the correlation in the translation space is calculated by:

$$C(T) = \int M_\lambda^A \bullet M_\lambda^B = F^{-1}\left[\int F\left[M_\lambda^A[T]\right] \bullet F\left[M_\lambda^B[T]\right]\right] \quad (1)$$

Where M_λ^A and M_λ^B are the electron density matrices representing three-dimensional maps. This type of fitting, called *Fast Translational Matching (FTM)* (Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A.A., Aflalo, C., and Vakser, I.A. (1992)), has already been successfully used in bioinformatics applications (Wriggers, W. and Chacon, P. (2001), (Eisenstein, M. and Katchalski-

Katzir, E. (2004)). Alternatively, it is also possible to accelerate the rotational space search by combining a suitable representation of the rotational space and the use of spherical harmonics (Ritchie, D.W. and Kemp, G.J. (2000), Kovacs, J.A. and Wriggers, W. (2002)). This methodology, called *Fast Rotational Matching (FRM)*, allows obtaining a search that is better adapted, and therefore faster, to the nature of the bioinformatics fittings described below. The *ADP_EM* algorithm combines *FTM* and *FRM* while *Situs* only uses *FFT*. The first one is described in later chapters.

2.1.2 Flexible Fitting

Flexible fitting methods are used to consider the conformational differences between the atomic structures and the density maps. Although the most commonly used method to study the flexibility of macromolecules are those based on *Molecular Dynamics* (*MD*) and *NMA* (*Normal Mode Analysis*), other alternative approaches have also been used successfully. The following section briefly discuss the *NMA* method in which flexible fitting tool used in this work is based (*iMODFIT*).

One of the most interesting alternatives to *MD* for studying the flexibility of macromolecules is the *NMA*. The *NMA* is an effective computational method for the study of large-scale and collective macromolecular motions despite its limitations (Kovacs, J.A., Chacon, P., Cong, Y., Metwally, E. and Wriggers, W. (2003)). The NMA can model with relative ease the collective and large amplitude movements of large macromolecular complexes. The main approximation of
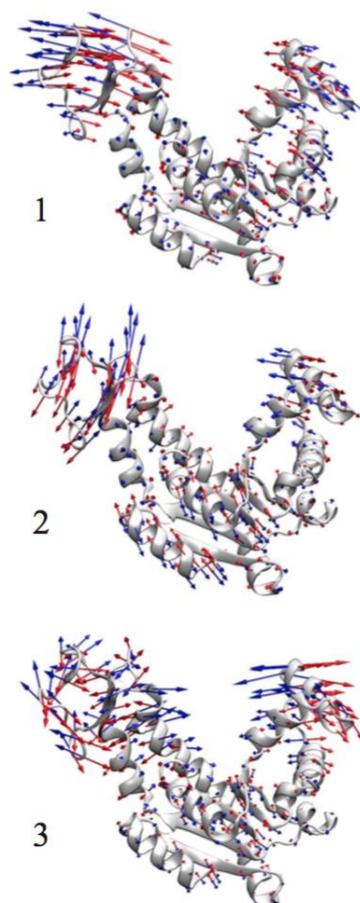
Figure 5: Normal vibration modes
The three lower energy modes of the protein structure of the adenylate kinase (4ake) protein have been shown with arrows.

23

this methodology is that the potential and kinetic energies vary quadratically around the minimum energy conformation of the system. From this assumption it is possible to decompose the macromolecular motion in a series of modes of deformation. These modes form an orthonormal basis of vectors which describes all possible shifts or deformations around the equilibrium conformation, that is, any movement can be expressed as a linear combination of these modes.

The first three modes of the adenylate kinase protein are shown in Figure 5. Each of them is associated with an energy (or frequency) so that it is possible to determine those movements that are more energy-efficient. Note that at higher frequency, higher energy, and vice versa.

It is not possible to identify which modes are functionally relevant without additional experimental data. However, in general, they will almost always be one or a few of the lowest frequency because they represent the conformational transitions with lower energy cost. In principle, it is possible to study the function of biomolecules by filtering out the less important, high-frequency motions, and focusing on the most dominant low frequency (lower energy) modes (Ma, J. (2005)).

2.2 *ADP_EM* Rigid Fitting Tool

ADP_EM is a rigid body fitting method used for interpreting the information contained in *EM* maps. The atomic structures are located inside the experimental map by maximizing their cross correlation. *ADP_EM* combines the *Fast Rotational Matching* method (*FRM*) (Kovacs, J.A. and Wriggers, W. (2002)) and translational scans using spherical harmonics and a convenient formulation of the three-dimensional rotation. Due to this, it is possible to improve the search efficiency and exhaustiveness of the rotational space.

The computational solution of the search problem can be reduced to finding the relative orientation and translation which maximizes the density cross-correlation of the structures/maps to be fitted. For a given rotation and translation the cross correlation is defined as the scalar product between the *EM* experimental map ρ_{low}, and a low-pass filtered version of the atomic structure, ρ_{high}:

$$C(T, R) = \int_{\mathbb{R}^3} \rho_{low} \times \Omega_T \Lambda_R \, \rho_{high} \qquad (1)$$

where Ω_T and Λ_R denote the translational and rotational operators, respectively. To find the highest correlation values, some previous approaches perform a systematic rotational scan of an atomic structure (ρ_{high}) relative to a fixed map (ρ_{low}), combined with a *Fast Fourier Transform* (*FFT*)-accelerated translational search based on the convolution theorem.

This well-known exhaustive search protocol is borrowed from the protein-protein docking field (Gabb, H.A., Jackson, R.M. and Sternberg, M.J. (1997), Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A.A., Aflalo, C. and Vakser, I.A. (1992), Vakser, I.A., Matar, O.G. and Lam, C.F. (1999)). In contrast, *ADP_EM* also accelerates the rotational search providing higher efficiency. This method, named *Fast Rotational Matching* (*FRM*), uses a spherical harmonics parameterization of the three-dimensional rotation to efficiently compute the correlation of all rotations for each position (Fig.6).

A detailed description of the theory underlying the *FRM* method was given elsewhere (Chacón, P. and W. Wriggers (2002), Kovacs, J.A., Chacon, P., Cong, Y., Metwally, E. and Wriggers, W. (2003). Briefly, density functions to be fitted are first approximated by expansions in a basis of spherical harmonic (*SH*) functions. To this end, the *EM* map is partitioned into concentric spherical layers each of which is represented by finite sums as:

$$\rho_{low}(r, \beta, \lambda) = \sum_{l=0}^{B-1} \sum_{m=-l}^{l} C_{lm}^{low}(r) Y_{lm}(\beta, \lambda)$$

$$\rho_{high}(r, \beta, \lambda) = \sum_{l=0}^{B-1} \sum_{m=-l}^{l} C_{lm}^{high}(r) Y_{lm}(\beta, \lambda)$$

$$(2)$$

where:

- $C_{lm}^{(r)}$ are coefficients associated with a specific, complex-valued spherical harmonic function $Y_{lm}(\beta, \lambda)$ defined on the unit sphere;

- $l \geq 0$ and $-l \leq m \leq l$ are the *SH* degree and order, and β and λ are the co-latitude and longitude, respectively.

- According to the sampling theorem, the number of sampling points (in each β and λ) used is equal to twice of the bandwidth B.

Instead of recasting the exhaustive search into a formulation involving five angles and just one translational parameter (Kovacs, J.A., Chacon, P., Cong, Y., Metwally, E. and Wriggers, W. (2003)), here the three rotational degrees of freedom are accelerated, while the three translational ones are simply scanned. Considering only the rotational part, the fitting function can now be expressed in terms of an inverse *Fourier* transform of the *SH* transforms (eq. 2) of the density maps (Kovacs, J.A. and Wriggers, W. (2002)):

$$C(R) = FT_{m,h,m'}^{-1} \left[\sum_{l} d_{mh}^{l} \, d_{hm'}^{l} \int_{0}^{\infty} C_{lm}^{low}(r) \, \overline{C_{lm'}^{high}(r)} \, r^2 dr \right] \qquad (3)$$

where the d_{mn}^{l} are real coefficients that define the matrix elements of the irreducible representations of the three-dimensional rotation group. This expression can be computed very efficiently by precomputing the coefficients d_{mn}^{l} and by using as upper limit of integration the maximum shell radius for which the density has non-zero values. In this way, eq. 3 allows, for a given translation, a very fast calculation of the correlation function for all rotations, which

will be sampled at twice the bandwidth *B* used in the harmonic transformation of the maps (eq. 2).

For example, *B=16* corresponds to scanning 16,000 rotations with a sampling step of 11.25°. If the rotational sampling step is set to 5.6° (*B=32*), more than 130,000 rotations will be explored. Thus, this method offers an efficient and customizable rotational screening (extracted with the permission of the receiving group (Garzón, J. I., J. Kovacs, R. Abagyan and P. Chacón (2007))).
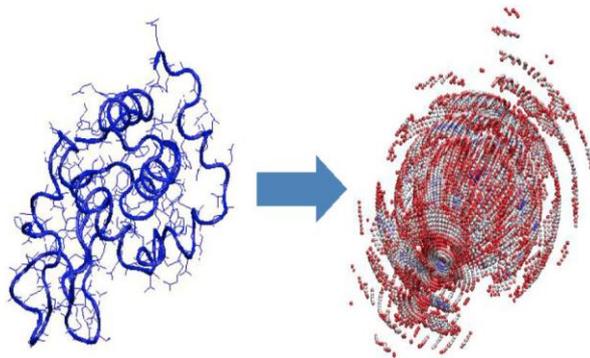
Figure 6: Radial and spherical sampling. The figure on the right represents the radial sampling of the electronic density of the *1lza* protein (left). Points with the highest electronic density are shown in red, with lower density in blue. Each spherical layer of this radial sampling is then used to perform the expansion on the basis of spherical harmonics.

The exhaustive search is then performed by applying this *FRM* rotational scan on a list of translational points that uniformly cover all the search space. Moreover, the translational space is limited to

Figure 7: Modelling the atomic structure using *ADP_EM*.

ADP_EM adjusts the atomic structure of the component to the map areas at low resolution where the correlation of electronic densities is greater. In the example shown, since the complex is a trimer (made up of three identical structures) there are three correlation maxima.

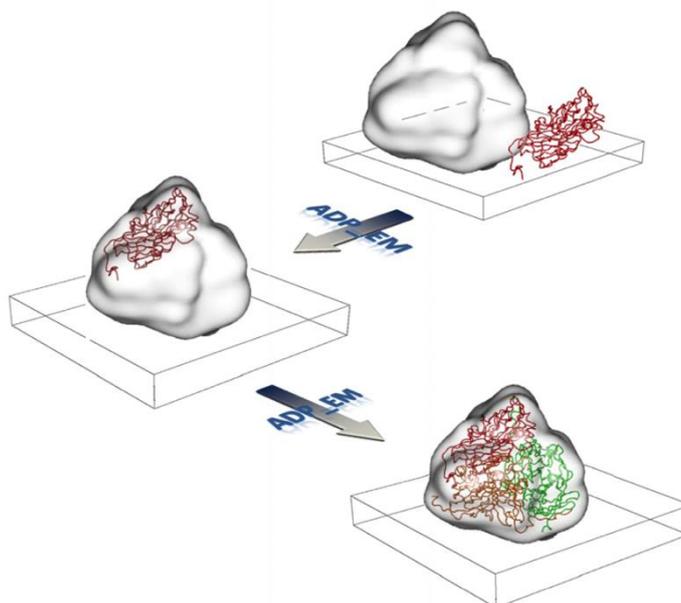positions on which the dimension of the atomic structure roughly fits inside the experimental *EM* map to prevent scanning nonsense points. There are other translational strategies, such as radial search (useful for structures with holes) or center-based search (practical for fitting structures with similar dimensions) (Kovacs, J.A., Chacon, P., Cong, Y., Metwally, E. (2003). These sampling schemes take advantage of geometry but their application range is not universal as the uniform sampling scheme using a mask. Therefore, the masking strategy is set as default in this project.
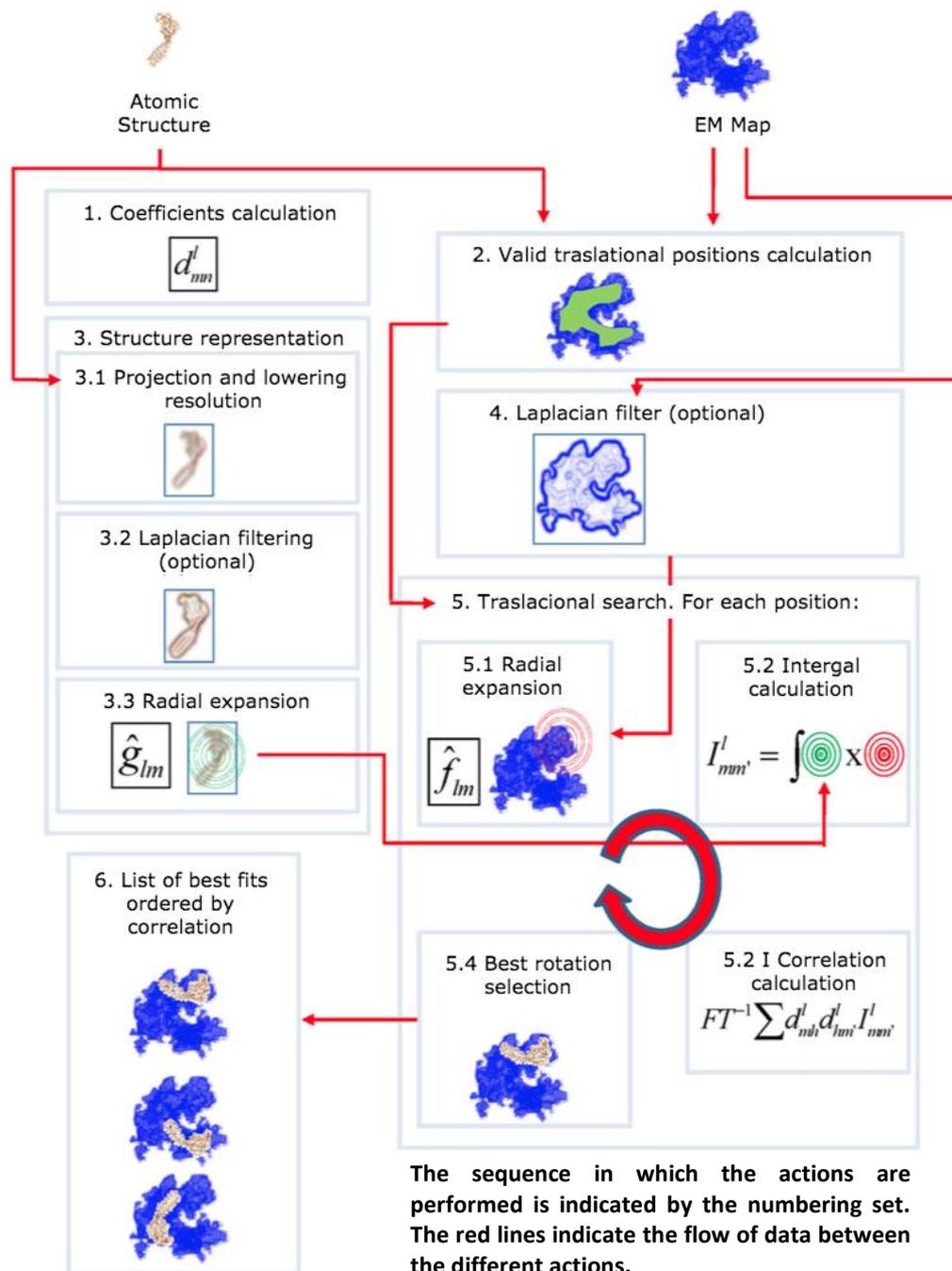

The sequence in which the actions are performed is indicated by the numbering set. The red lines indicate the flow of data between the different actions.

Figure 8: *ADP_EM* **Workflow.**

Although the density cross-correlation works reasonably well, in particular cases may lead to ambiguous false positives. This can be critical when the resolutions are low, typically less than 15 Å, and small components are to be placed in a large density map. Some alternatives can be taken into account to improve the fitting contrast. For example, the fitting can be performed by a local correlation criterion (Roseman, A.M. (2000), Rath, B.K., Hegerl, R., Leith, A., Shaikh, T.R., Wagenknecht, T. and Frank, J. (2003)), or the maps can be pre-filtered with a *Laplacian* kernel (Chacón, P. and W. Wriggers (2002)). Since its implementation does not need any change in the registration scheme, here the fitting is performed with *Laplacian*-filtered maps instead of the original density maps. The strategy of convolving the maps with a *Laplacian* kernel improves the numerical contrast among potential solutions, by including both density and contour overlap.

A new version of *ADP_EM* and a plugin were developed and integrated in Chimera. This will be described in later chapters.

2.3 *iMODFIT* Flexible Fitting Tool

iMODFIT is an approach to obtain a flexible atomic model from a low-resolution experimental map and an initial atomic structure in different conformations.

Basically, *NMA* in *IC* reduces the conformational search space to physically realistic collective motions and implicitly maintains the covalent structures, thus preventing distortions. Because low-frequency modes computed in *IC* provide a reasonable and inexpensive direct view of the relevant conformational space, it is possible to use the most probable deformation directions encoded in this essential space to flex the atomic structure while maximizing the density overlap with the target experimental map.

The *NMA* decomposes motion into a set of collective deformation modes. This reduces dramatically the number of variables and

improves efficiency (Bray, J.K., Weiss, D.R., Levitt, M., 2011, Kovacs, J.A., Cavasotto, C.N., Abagyan, R., 2005, López-Blanco, J.R., Garzon, J.I., Chacon, P., 2011, Lu, M., Poon, B., Ma, J., 2006, Mendez, R., Bastolla, U., 2010).

The *iMOD NMA* engine is used to compute low-frequency modes in *IC* (Lopez-Blanco, J.R., Garzon, J.I., Chacon, P., 2011). In *NMA,* the macromolecule is modelled as a series of pseudo-atoms connected by harmonic springs. The modes are calculated by solving a general eigenvalue problem that diagonalizes the second derivative matrices of potential and kinetic energies (Lopéz-Blanco, J. R. and P. Chacón (2013)):

$$HU = \lambda_k TU \quad \text{where} \quad U = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N), \tag{1}$$

Here, $\boldsymbol{u}_k$ is the kth deformation vector with its associated λ_k eigenvalue, and H and T are the kinetic energy matrixes, respectively. The equation of the potential energy is (Lopéz-Blanco, J. R. and P. Chacón (2013)):

$$V = \sum_{i<j} F_{ij}(r_{ij} - r_{ij}^0)^2 + s \sum_{\alpha} (\theta_\alpha - \theta_\alpha^0)^2 \quad \text{where} \quad F_{ij} = 1 / \left(1 + \left(r_{ij}^0 / 3.8\right)^6\right) \tag{2}$$

One of the most interesting advantages of *iMODFIT* is its versatility because it can handle different types of complexes and different graining levels to represent structures:

- Heavy-atoms (*HA*): considers all heavy atoms for proteins and nucleic acids (next-hydrogen).

- Five pseudo-atoms (*C5*): uses *NH*, *Cα*, and *CO*, including a *Cβ* and virtual mass located at the mass center.

- *Cα*: select a single *Cα* atom per amino acid for proteins.

Indeed, these different representations had to be handled in the Chimera plugin developed that will be seen in later chapters.

iMODFIT workflow is summarized in the Figure 9. In essence, the tool interactively explores the lowest frequency modes to improve

the cross correlation with a target map. The atomic structure must be approximately placed in the correct position inside the map. To this end, *ADP_EM (*Garzón, J. I., J. Kovacs, R. Abagyan and P. Chacón (2007)) or COLORES (Chacón, P. and W. Wriggers (2002)) would be a good choice and, indeed, it is the way of working in this project.

First, it starts by calculating the 5% lowest-frequency modes in *IC* from the initial model (step 1). Then, randomly merges the 10% of them into a single deformation vector to generate a trial structure (Steps 2 and 3). Then, the new trial model is low-pass-filtered to produce a simulated density map. Finally, the fitting score is calculated using the normalized cross-correlation between the *EM* experimental target map, p_{exp}, and this simulated map, p_{trial}, (step 4):

$$C = \sum_{i}^{voxels} \frac{\left[\rho_{\exp}^{i} - \left\langle \rho_{\exp} \right\rangle \right]\left[\rho_{trial}^{i} - \left\langle \rho_{trial} \right\rangle \right]}{\sigma_{\exp}\sigma_{trial}} \qquad (3)$$

This new flexed conformation is accepted only if the cross-correlation improves; otherwise, the tool goes back (step 1) to generate new trial deformation and so on. This is repeated until convergence is reached except when the flexed conformation deviates more than 0.1 Å *RMSD* from the previous one used for *NMA*. In this situation, normal modes are refreshed using the last accepted conformation. In addition, a local rigid-body optimization is performed every 200 iterations to realign the flexed conformation in order to compensate the small changes in the center of mass and orientation required for optimal fitting.

Summarizing, combining rigid fitting as *ADP_EM*, to localize an atomic structure into a target density map, with flexible fitting as *iMODFIT*, to observe conformational changes in that complex, it is possible to obtain a great quality fitting with efficiency and reliability. Moreover, it is very easy to use because it does not require elaborated pre-processing steps (just a previous approximate rigid body fitting). In addition, it is highly customizable and the user can control all the fitting parameters. Furthermore, the computational cost is low, only a few minutes are needed to perform the fitting.
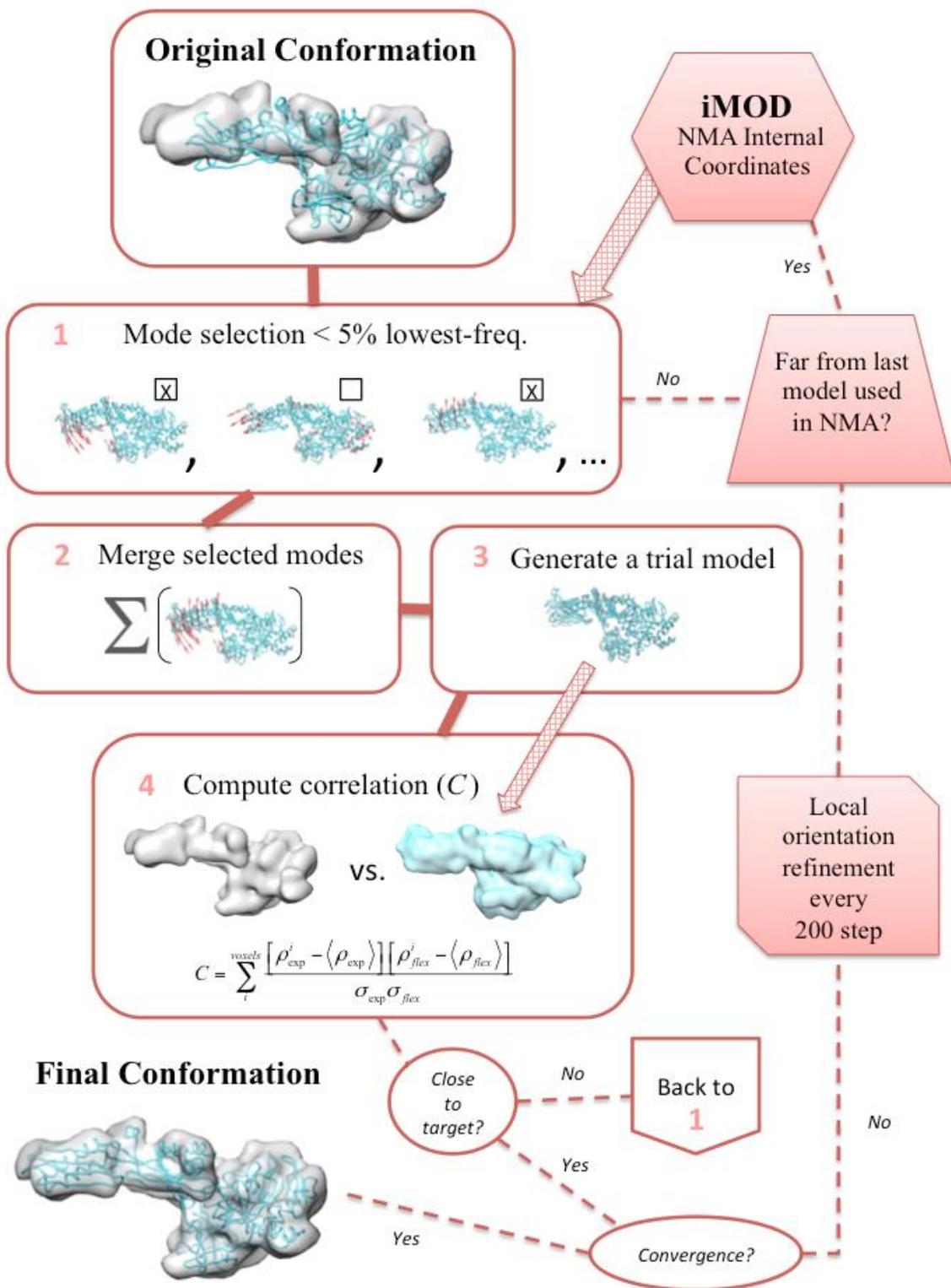
Figure 9: *iMODFIT* Workflow.

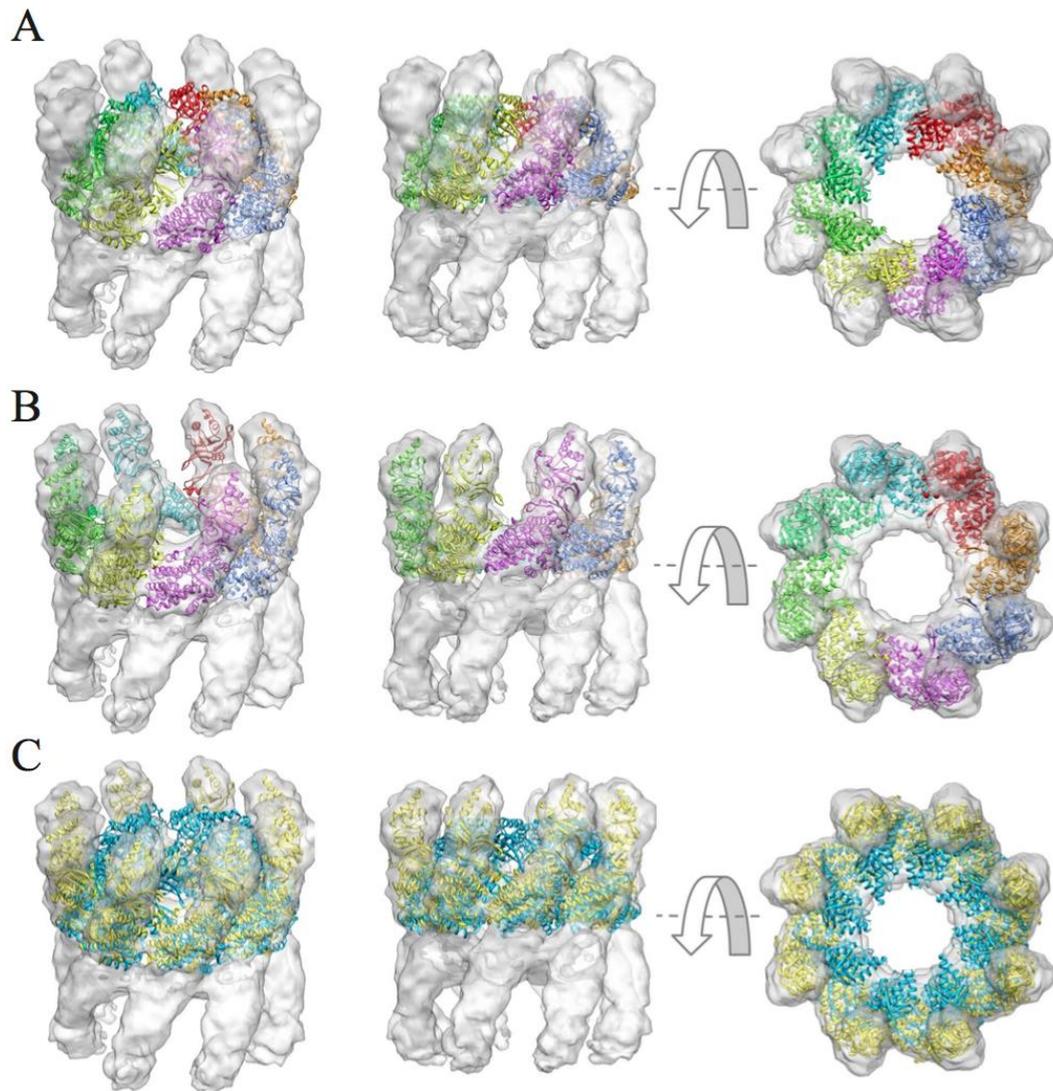Following, some flexible fittings with *iMODFIT* in combination with *ADP_EM* are presented:

Figure 10: Flexible fitting of the thermosome into an experimental *EM* map at 10Å. Panel A shows the initial orientation obtained with *ADP_EM*, in B, the model adjusted with *iMODFIT*, and in C, the overlap of the initial structure (blue) on the model (yellow). The map is displayed in transparency.
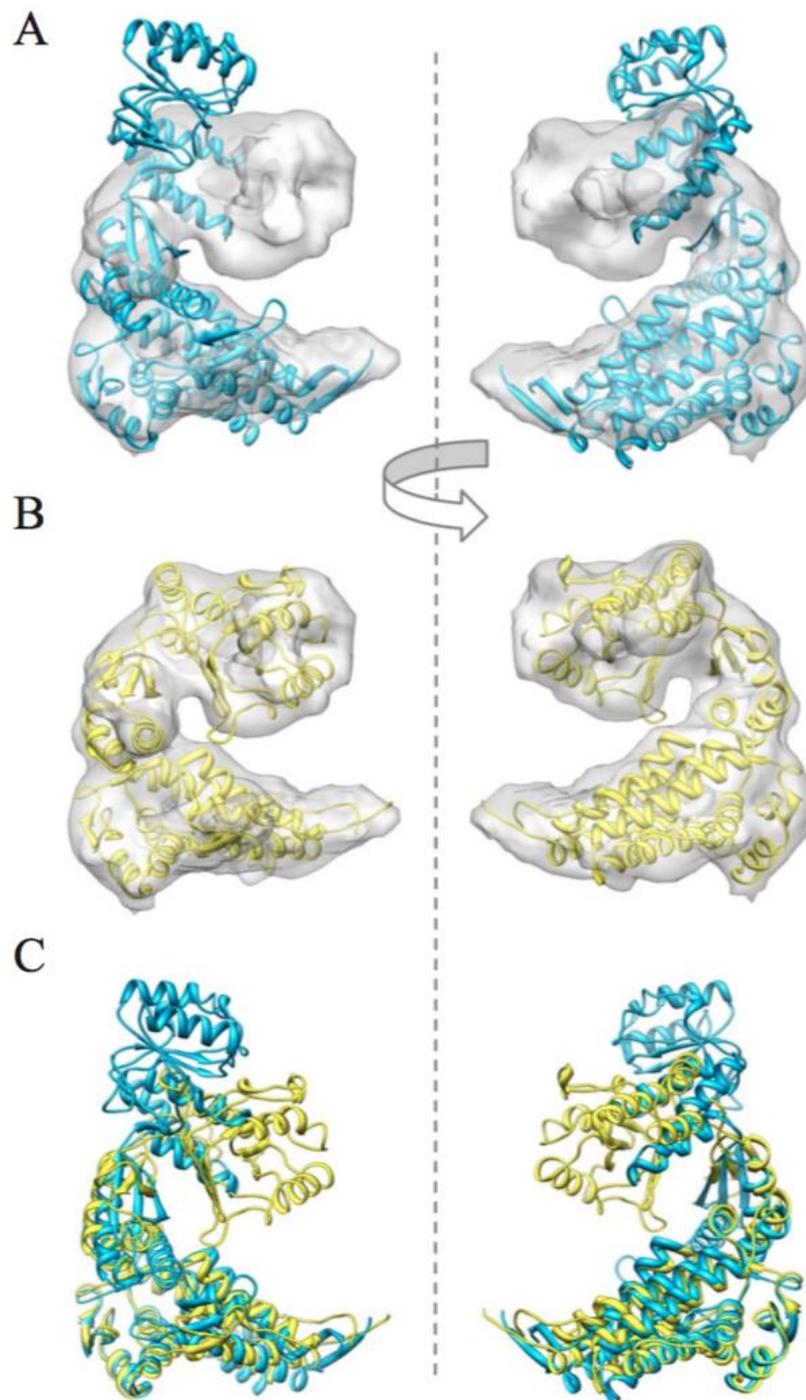
Figure 11: Flexible fitting of the *GroEL* into an experimental *EM* map at 10 Å
In panels A and B two side views of the initial structures (blue) and adjusted (yellow) are shown, respectively. In C the initial structure aligned on the final structure with Chimera is shown.

As previously mentioned, it is frequent to first perform a rigid body fitting (*ADP_EM*) and then adjust the conformations by applying a flexible fitting (*iMODFIT*) for greater accuracy.

The following is an image illustrating this case:
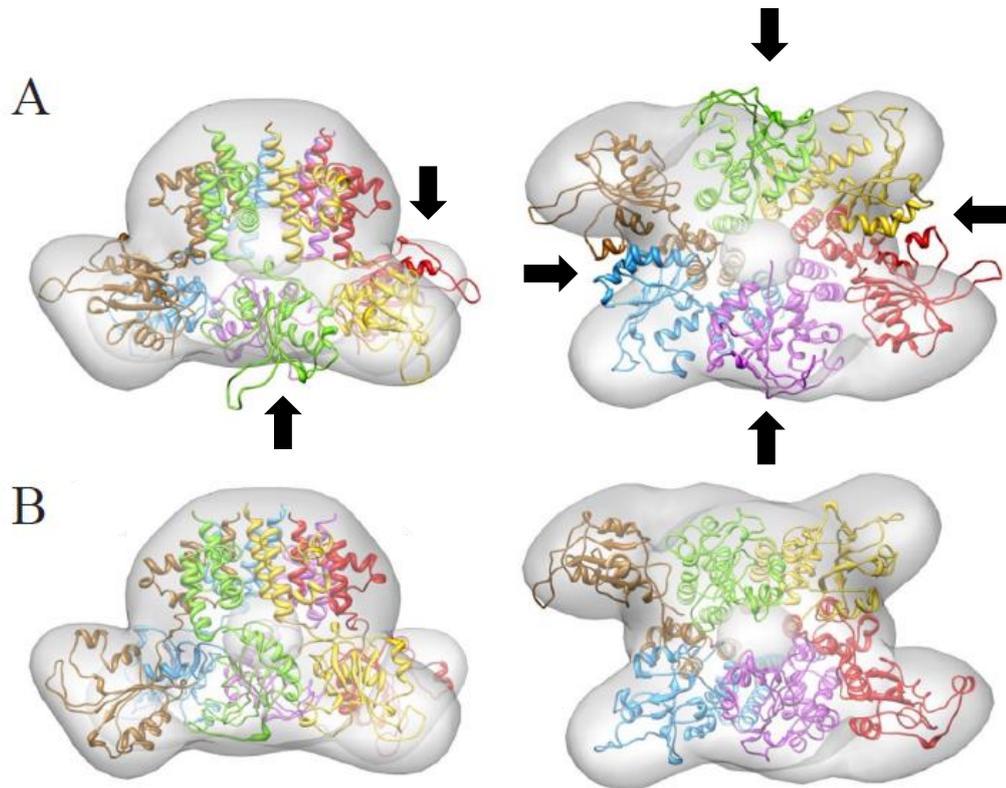
Figure 12: Flexible fitting of the *RepB* in the presence of *DNA* using *iMODFIT*.
The orientation of the initial structure (panel A) has been determined with *ADP_EM*. Panel
B shows the final model of the flexibly fitted obtained with the default parameters. Arrows
indicate the regions that require flexible fitting.

3. Results

3.1 Chimera

UCSF Chimera is a 3D visualization program for *EM* density maps and atomic structures. This program includes a suite of tools for interactive analyses of sequences and structures. In particular, it offers interaction with molecular structures and related data, including supramolecular assemblies, molecular dynamics trajectories, and multiple sequence alignments. Moreover, it enhances researcher workflow with novel extension features and the creation of HD images and animations for publication and presentation purposes.
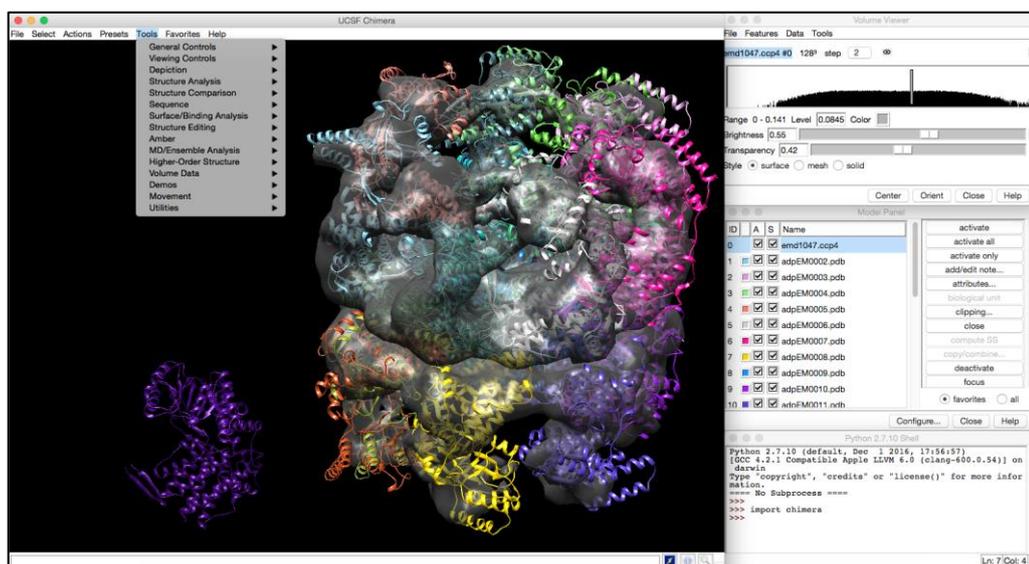

Figure 13: A snapshot of the UCSF Chimera program.

Besides supporting 3D visualization, other native features include:

- Multiscale Models to visualize large-scale molecular assemblies.

- ViewDock to screen docked ligand orientations.

- Volume Viewer to visualize density maps.

- Multalign Viewer to display sequence alignments, with crosstalk to any associated structure.

Chimera is distributed with full documentation and a number of tutorials. It can be downloaded free of charge for academic, government, non-profit, and personal use. It is available for several platforms, including Windows, MacOS X, and Linux.

Chimera is developed and supported by the Resource for Biocomputing, Visualization, and Informatics, and it is funded by the NIH National Center for Research Resources at the University of California, San Francisco.

The software is specifically designed for extensibility, to allow outside developers to incorporate new desirable functions.

The principal objective of this work has been the development, optimization, and adaptation of the *ADP_EM* and *iMODFIT* tools to be integrated as plugins in Chimera.

3.2 *ADP_EM* Plugin for Chimera

The main objective of this project is the development of two plugins for Chimera to perform the two types of fitting (rigid and flexible), this is, *ADP_EM* and *iMODFIT*. Both of them conserve the same parameters and features as the original methods but are easier to use. Furthermore, the user can interactively visualize the different solutions as soon as they are computed. In this section, the *ADP_EM* plugin for Chimera is presented.

The *ADP_EM* plugin developed for Chimera is located in a newly created *EM Fitting* section in the *Tools* menu that Chimera offers to the users (Figure 14).
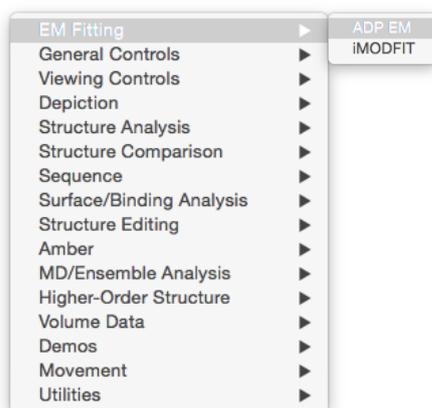
Figure 14: *ADP_EM* **entry menu in Chimera.**

When *ADP_EM* is loaded, the basic window or *GUI* that is presented to the user is shown (Figure 15):
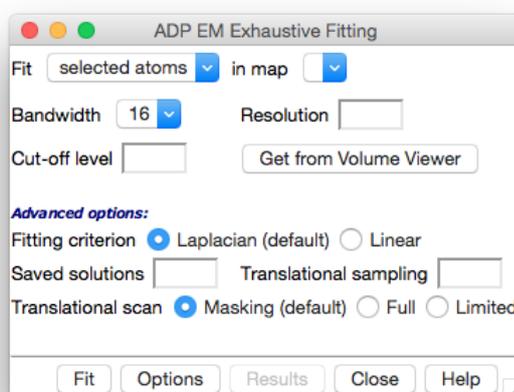
Figure 15: *ADP_EM* **basic** *GUI* **in Chimera.**

As it can be seen, the most important parameters of the original methods are readily available in the basic window. The first field selects the atomic structure and the second is the map. The rest of them are described below:

- **Bandwidth**: it corresponds to the bandwidth in the harmonic transformation. Its values are set to 16, 24, 32, 48 or 64. They correspond to increasing angular sampling values.

- **Resolution**: the nominal resolution of the projection map [Å].

- **Cut-off**: is the density threshold value for the experimental map. All density levels below this value will be not considered. User can get this value from *Volume Viewer* dialog through the *Get from Volume Viewer* button available.

These parameters are the minimum required to perform the fitting. The next ones are advanced features that should be used carefully by the user:

- **Fitting criterion**: this option sets the fitting criterion:

  - *Standard cross-correlation* → the scalar product between the density map of the low resolution map and the low-pass filtered atomic structure. Recommended for resolutions < 15 Å, specially the atomic model accounts all the density of the map.

  - *Laplacian filter* → is applied by default to maximize the fitting contrast. Recommended for resolutions ≥ 15 Å, specially when the atomic model only accounts a part of the density of the experimental map.

- **Saved solutions**: number of the saved solutions that *ADP_EM* will compute (50 by default).

- **Translational sampling**: in [Å], by default twice of voxel size of the density map. Values > 6 Å should not be used.

- **Translational scan**: translational scan strategy:

  - Full search → all the translational points inside the target *EM* map will be explored.

  - Limited → radial search starting from the center of mass.

  - Masking search → by default.

Apart from the button that displays the options for expert users, there are four more:

- **Fit**: performs the fitting after validating properly the parameters.

- **Results**: displays the solutions of the fitting in a new panel. It is disabled while fitting is being performed.

- **Close**: close the *GUI* but keeps all the values associated.

- **Help**: opens a help guide in the browser.

Once the user has loaded the atomic structure and the map and inserted coherent values for the basic parameters, the fitting can be performed by clicking the *Fit* button (Figure 16).

When the *ADP_EM* is executed, a new window is shown to the user with the process log and a progress bar to check the status (Figure 17).
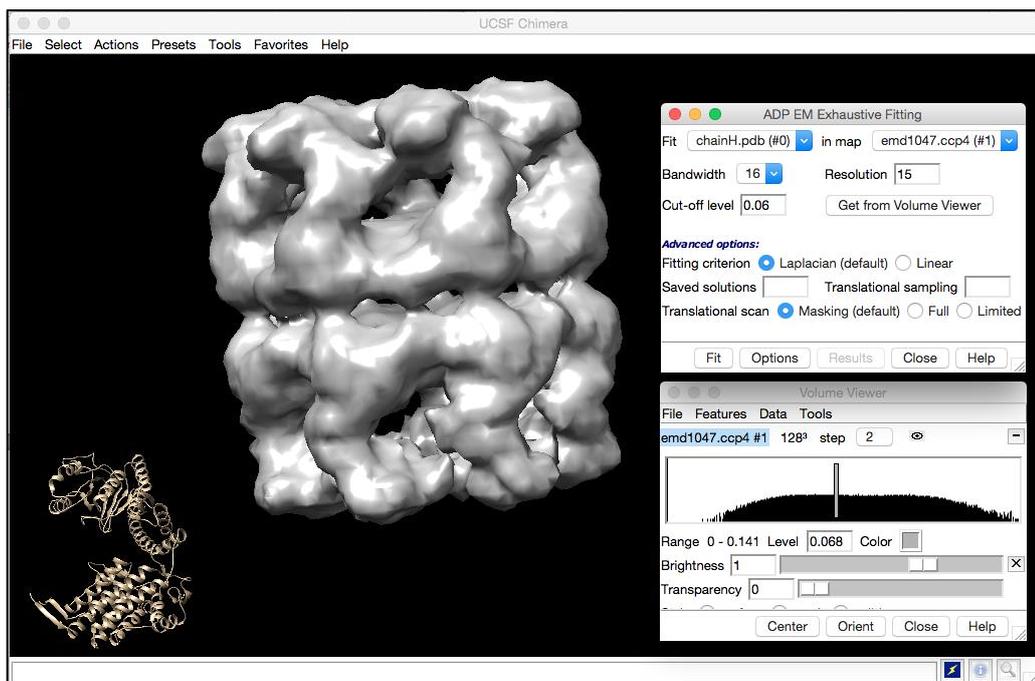
Figure 16: ADP_EM GUI status just before perform fitting. *EM* **map and atomic structure are loaded and the minimum required parameters are set.**
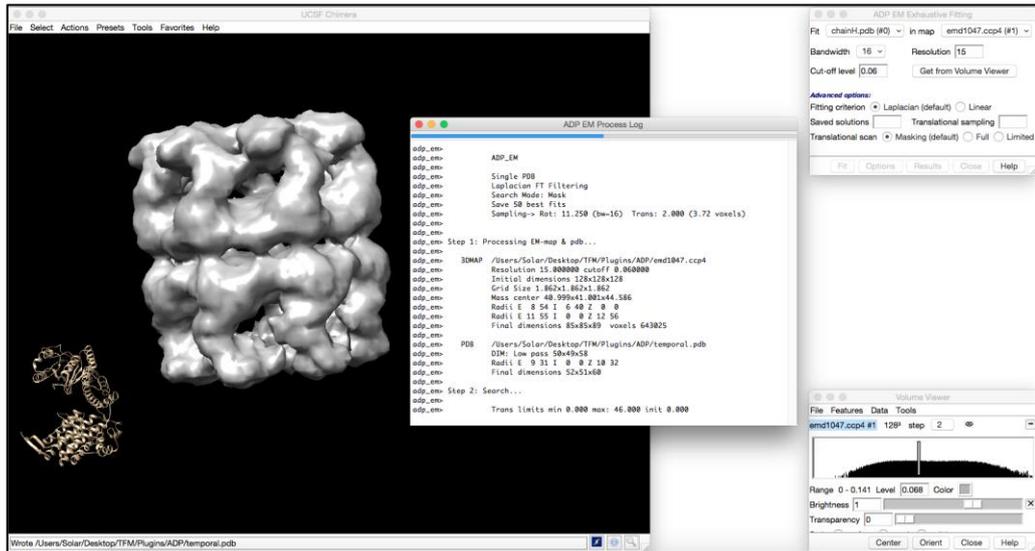
Figure 17: *ADP_EM* **fitting process shown by the log window. Notice how all the buttons are disabled while the process is running.**

In addition to these parameters, some extra features had been provided to expert *ADP_EM* users (Figure 18). These parameters can be found in the *Options* button, that are displayed in a new panel:

- **Number peaks explored per docking**: default 30.

- **Number peaks stored per iteration**: default 20.

- **Number peaks stored in the search**: default 100.

- **Number peaks stored in the multi-docking search**: default 500.

- **Translational threshold in grid units**: default 2.0.

- **Rotational threshold in degrees**: default 360/bandwidth.

- **Width between spherical layers**: default 1.0.

- **Density cut-off of the simulated map**: default 0.0
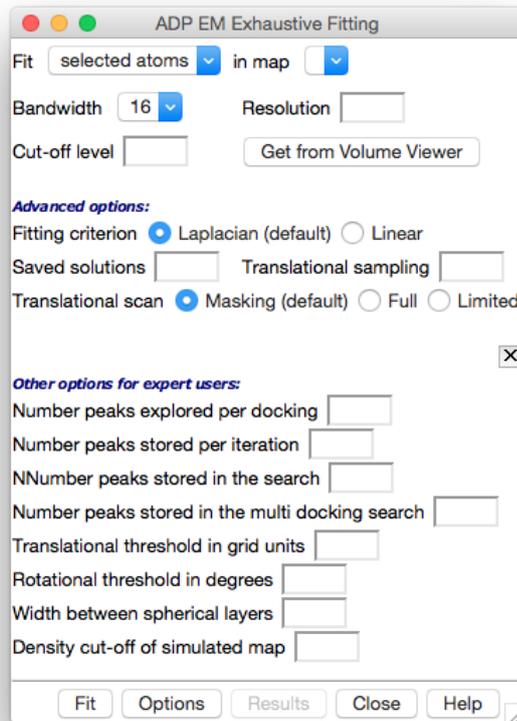
Figure 18: *ADP_EM* **expert GUI in Chimera.**

Mainly, the execution time varies depending on the *Bandwidth* value. The finer the angular sampling (bandwidth), the more it takes.
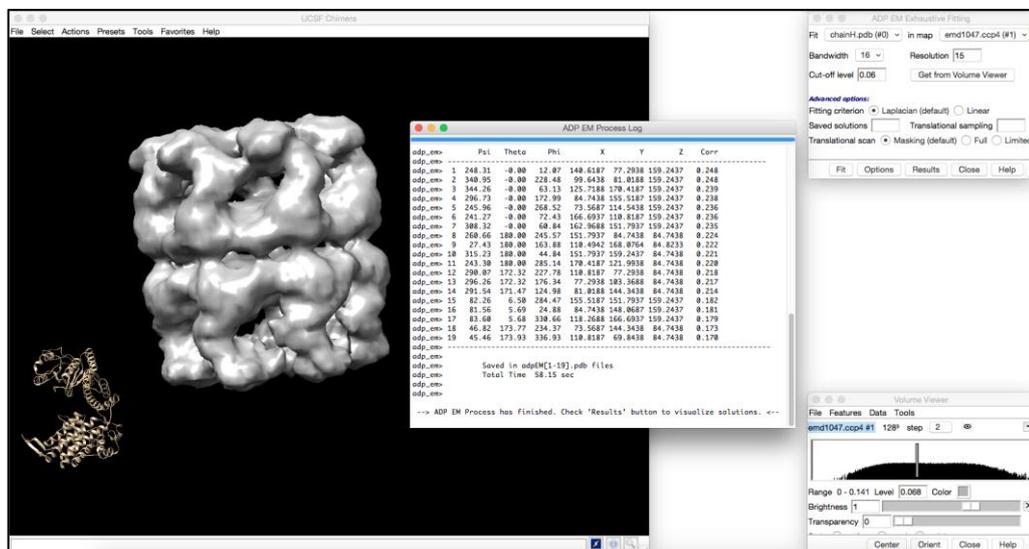
Figure 19: *ADP_EM* **process finished.Notice how all the buttons are now enabled.**

Once the process is finished, the log informs that the user can check the solutions (Figure 19), and the user can visualize the solutions in the *Results* panel which can be seen in Figure 20:
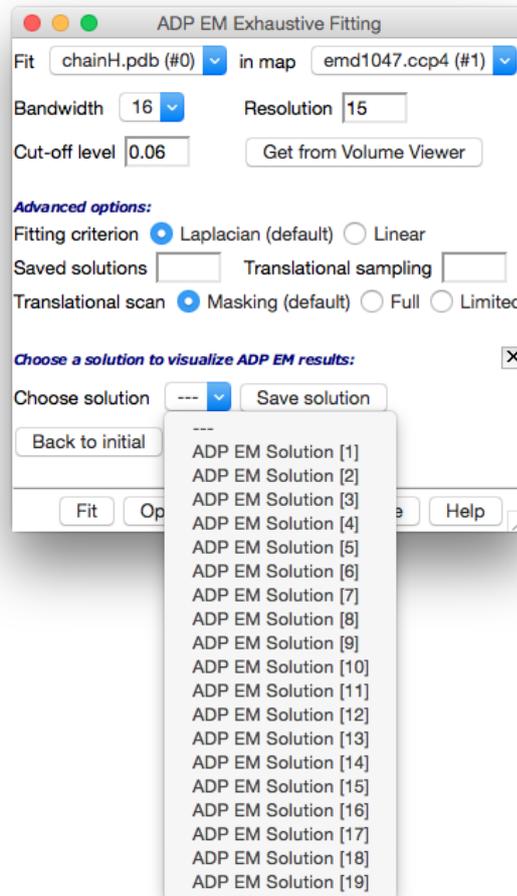
Figure 20: *ADP_EM Results* panel with all the calculated solutions dropped down.

As it can be seen, the main options keep visible to let the user perform another fitting if it is necessary. Apart of this, there is a drop-down menu to choose one of the solutions computed by *ADP_EM*.

When one of them is chosen, the opened atomic structure updates its coordinates and moves into the map to the fitted position calculated by *ADP_EM* (Figures 21, 22 and 23).
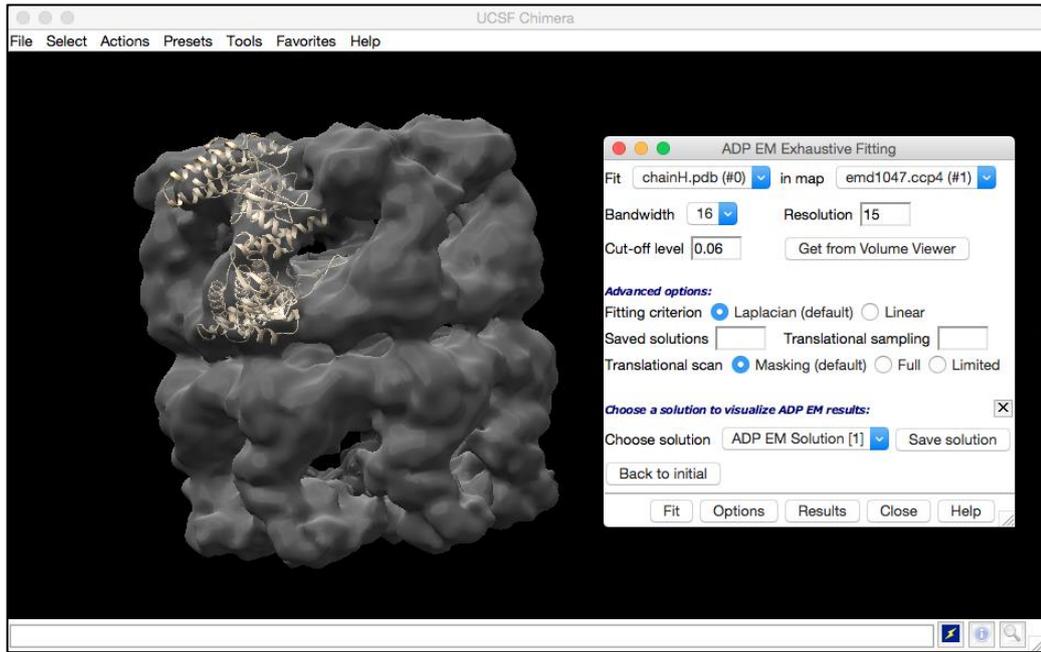
Figure 21: *ADP_EM* Solution 1 fitted into the map.
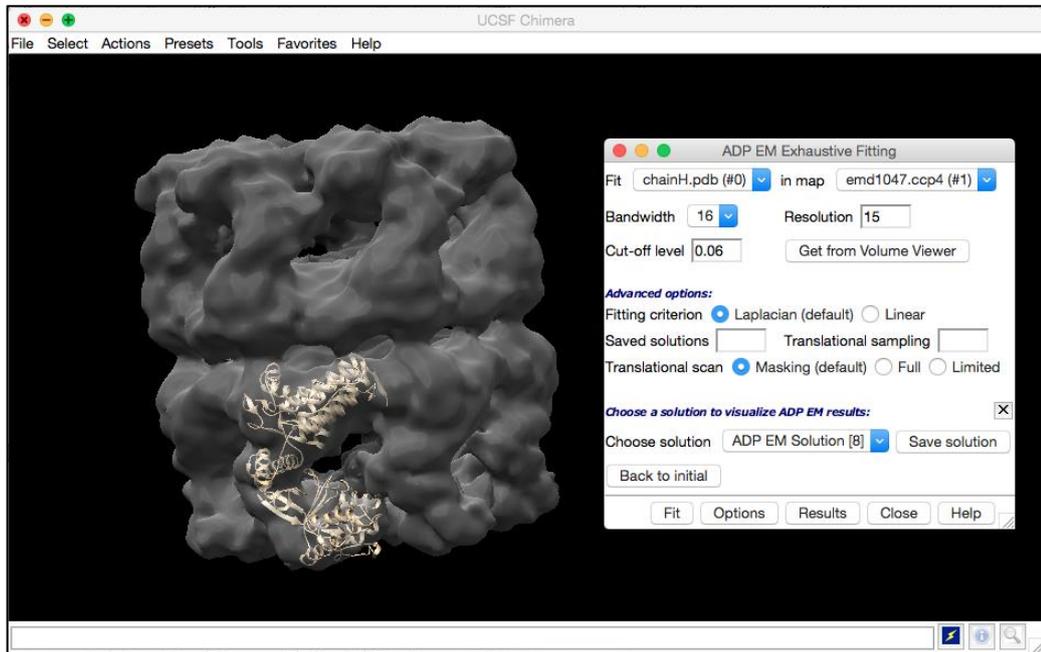
Figure 22: *ADP_EM* Solution 8 fitted into the map.

44
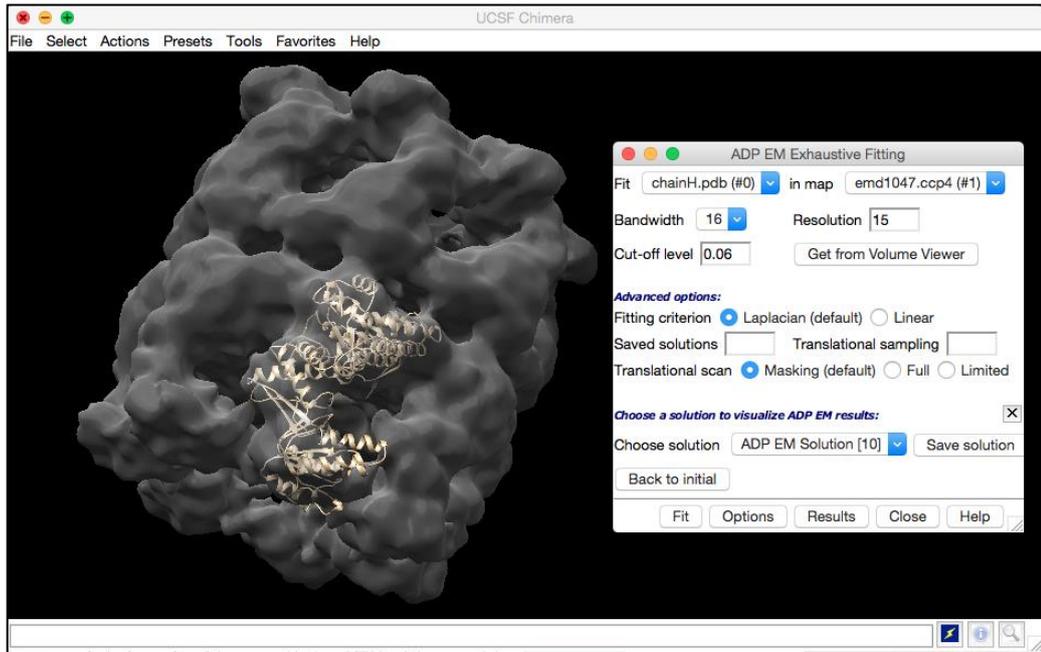
Figure 23: *ADP_EM* **Solution 10 fitted into the map.**

Additionally, a button called *Save Solution* was implemented to make a copy, with a representative name, to the *Model Panel* of the chosen solution. *Model Panel* is a very important dialog of Chimera that shows to the user all the models opened in the program (Figure 24).
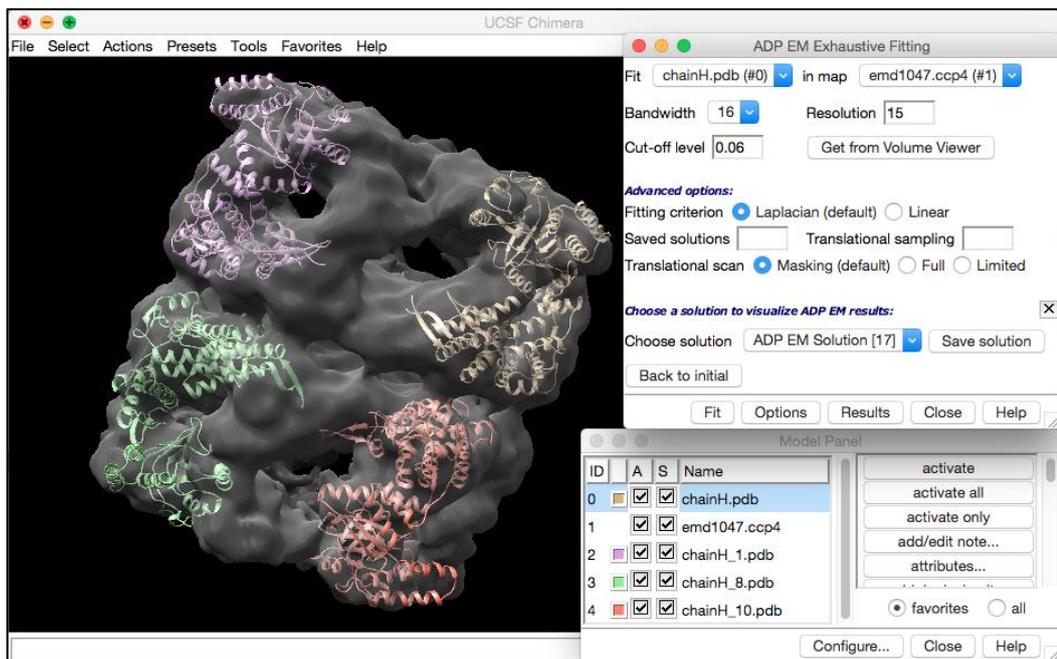
Figure 24: Copies of solutions 1, 8 and 10 and molecule positioned in solution 17.

45

The last button in the *Results* panel, *Back to initial*, simply resets the coordinates of the molecule and moves it to the original position it had before the fitting was performed.

Briefly, the *Help* button opens a user guide in the browser with instructions to load correctly the plugin in Chimera and a description of the parameters involved to use them properly.
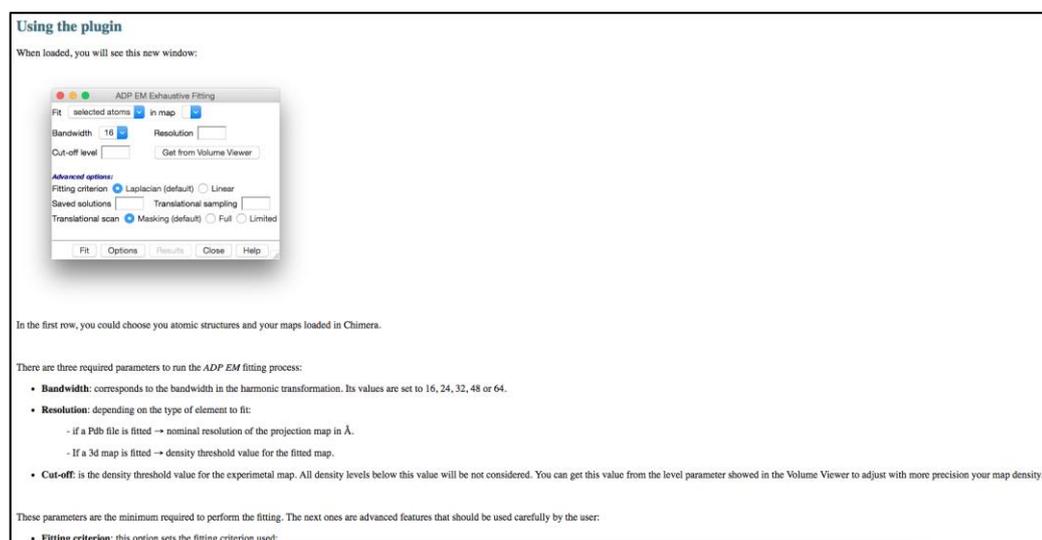
Figure 25: A snapshot of the user guide.

It is very common that users, once they loaded the atomic structure and the map, move the molecule for some analytic reasons. This situation had to be considered during the plugin development because it involves camera motions related to the objects as well as changes in their internal coordinates.
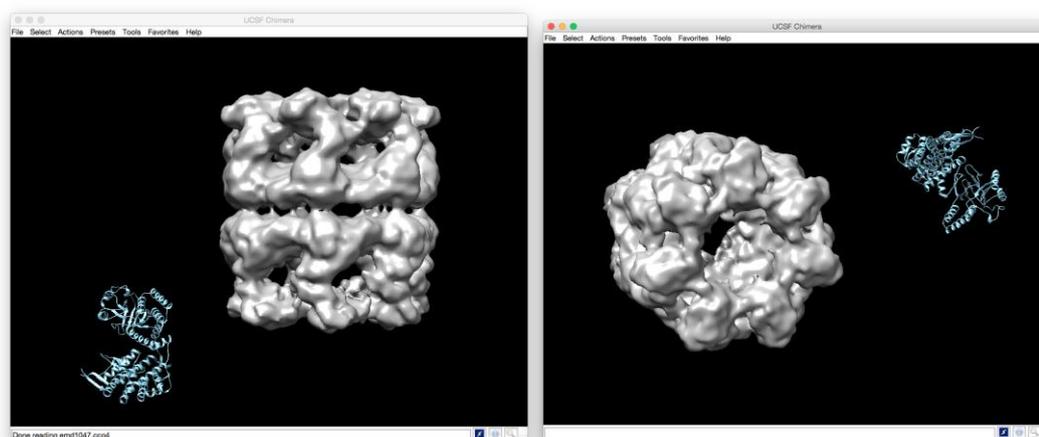
Figure 26: Objects loaded (A) and objects moved (B).

This had added considerable difficulty in achieving the plugin but it maintains its efficiency and maintainability.
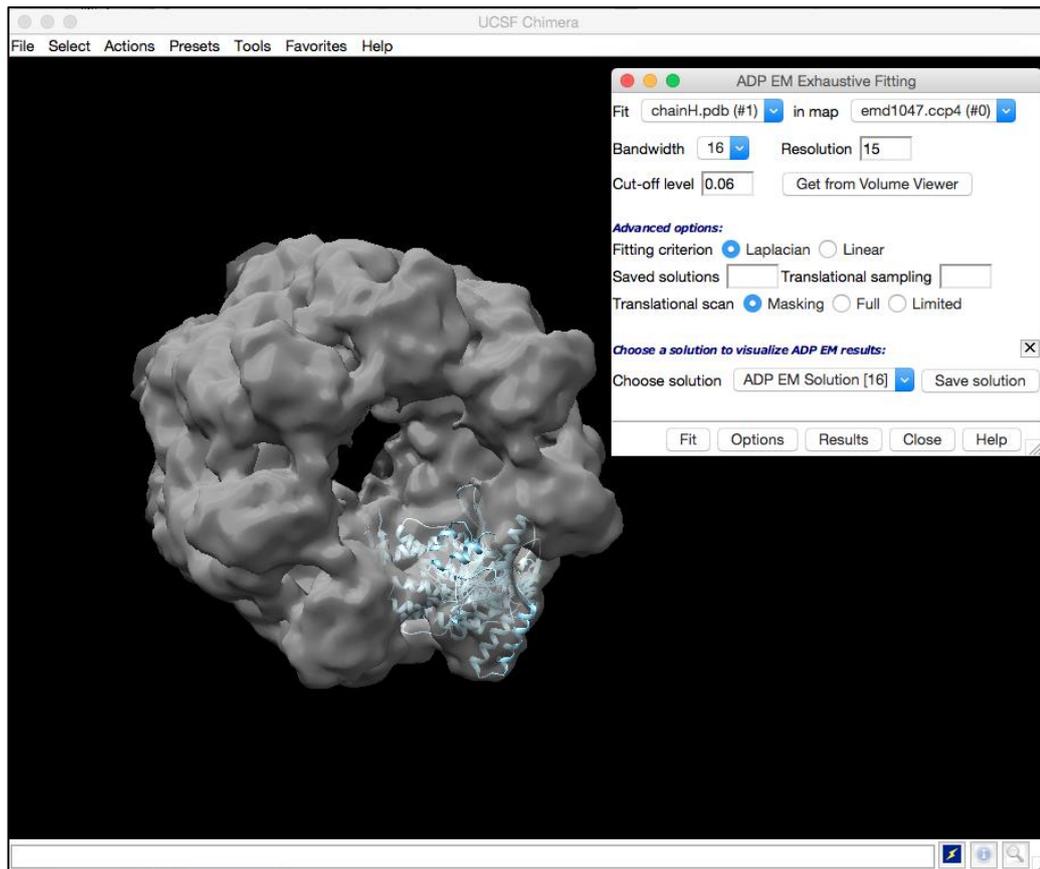
Figure 27: Solution 16 of the moved molecule from Figure 25b.

In later chapters the dissemination of this plugin will be described as well as the different guides and considerations for its correct usage.

3.3 *iMODFIT* Plugin for Chimera

The *iMODFIT* plugin is located (with *ADP_EM*) in the newly created *EM Fitting* section of the Chimera *Tools* panel (Figure 28). It can be used (and should be) with some of the solutions provided from *ADP_EM* in a previous execution.
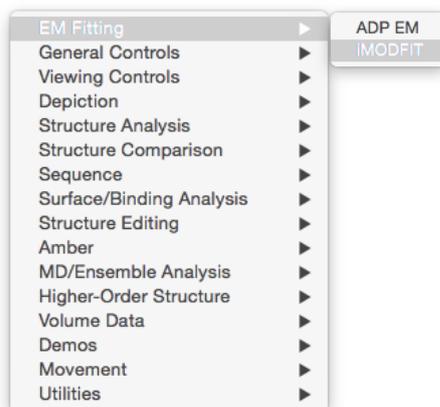
Figure 28: *iMODFIT* **entry menu in Chimera.**

When *iMODFIT* is loaded, the basic window or *GUI* that is presented to the user is shown below (Figure 29):
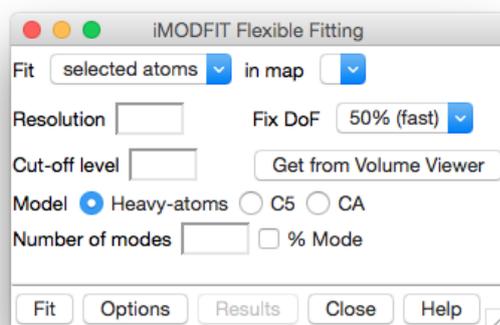
Figure 29: *iMODFIT* **simplest** *GUI* **in Chimera.**

As can be seen, there are many options for the user to perform the flexible fitting that match the original parameters of the method. The first field selects the atomic structure to fit in the map, the second one. The rest of them are described below:

- **Resolution**: the resolution criterion follows *EMAN* package procedures. It is the nominal resolution of the projection map in [Å].

- **Fix DoF**: randomly fixed ratio of dihedral coords. Example: 0.7 = 70% of dihedrals will be randomly fixed. User can choose between *None, 50% (fast), 70% (faster)* or *90% (fastest)*.

- **Cut-off**: is the density threshold value for the experimental map. All density levels below this value will be not considered. User can get this value from *Volume Viewer* dialog when the map is loaded through the button available.

- **Model***: represents the coarse-grained model. User can choose between *Heavy-atoms, C5* or *Cα* (as described in 2.3)

- **Number of modes and % Mode**: Used modes range, either number [1,N] <integer>, or ratio [0,1) <float> (default=0.05).

Apart from the button that displays the options for expert users, there are four more:

- **Fit**: performs the fitting after validating properly the parameters.

- **Results**: displays the solutions of the fitting in a new panel. It is disabled while fitting is being performed.

- **Close**: close the *GUI* but keeps all the values associated.

- **Help**: opens a help guide in the browser.

Once the user has loaded the atomic structure and the map and inserted coherent values for the minimum parameters, the fitting can be performed by clicking de *Fit* button (Figure 30).

When the *iMODFIT* is executed, a new window is shown to the user with the process log (Figure 31).
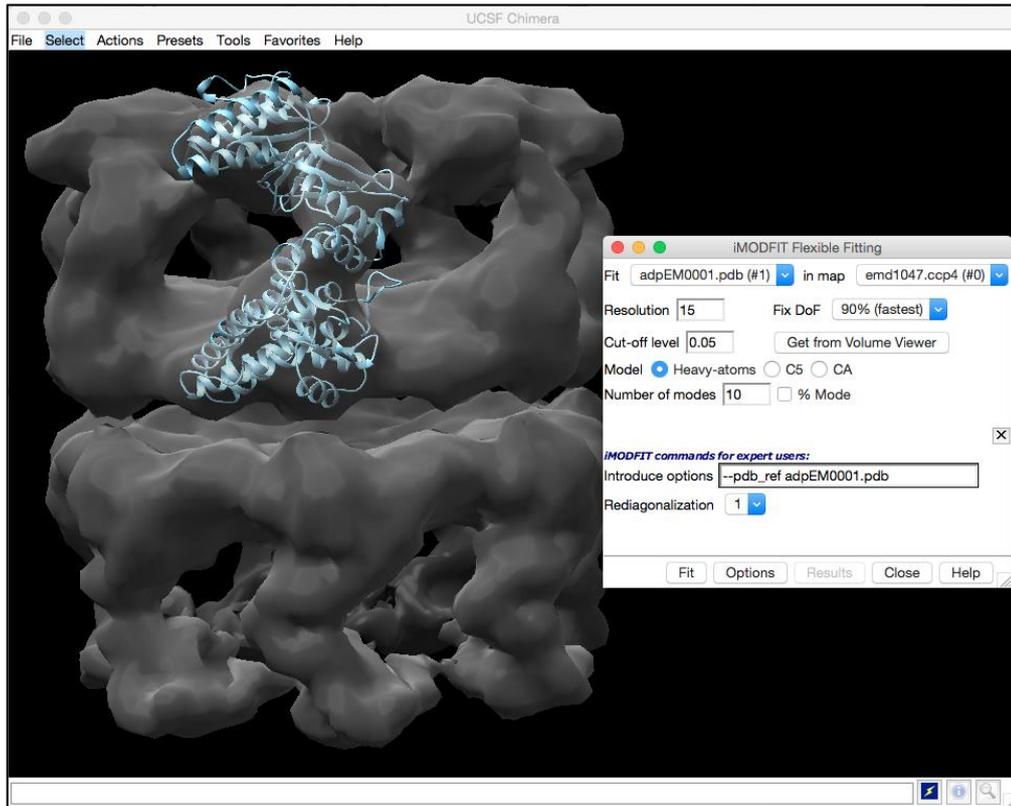
Figure 30: *iMODFIT GUI* status just before start fitting. *EM* map and the first solution of *ADP_EM* are loaded and the minimum required parameters are set.
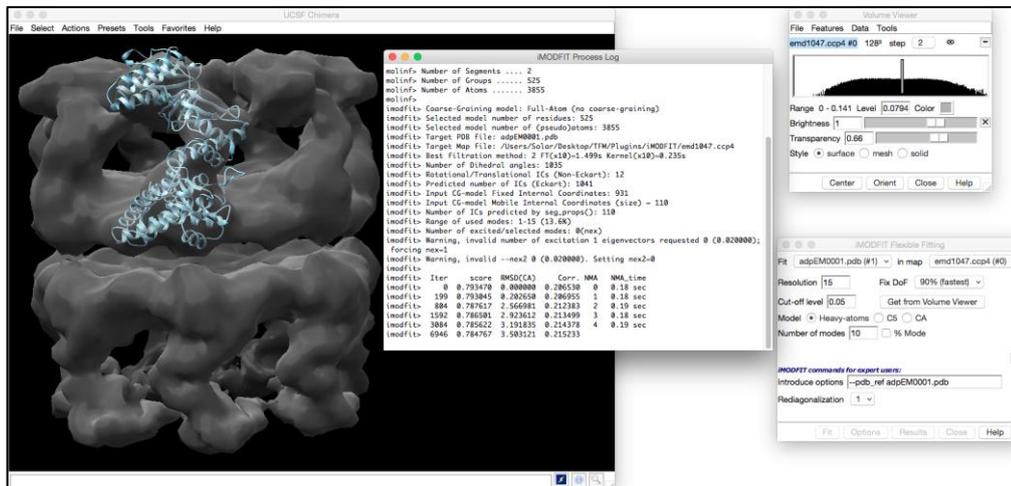
Figure 31: *iMODFIT* fitting process shown by the log window. Notice how all the buttons are disabled while the process is running.

In addition to these parameters, some extra features had been provided to expert users. These parameters can be found in the

Options button, which displays a new panel to insert them (Figure 32):
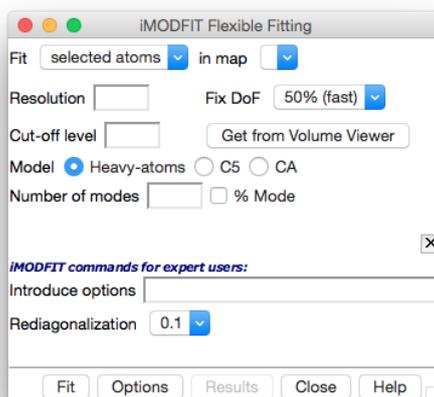
Figure 32: *iMODFIT* expert *GUI* in Chimera.

- **Introduce options**: this field allows user introduce advanced commands for *iMODFIT*. Example: *--addnevs 0.2 –pdb_ref chainX.pdb*

- **Rediagonalization**: *RMSD* ratio to trigger diagonalization. Default 0. User can choose between *None, 0.1, 0.5* or *1.*

Mainly, the execution time varies depending on the values of the parameters*.* Usually, *iMODFIT* takes several minutes to perform the flexible fitting.
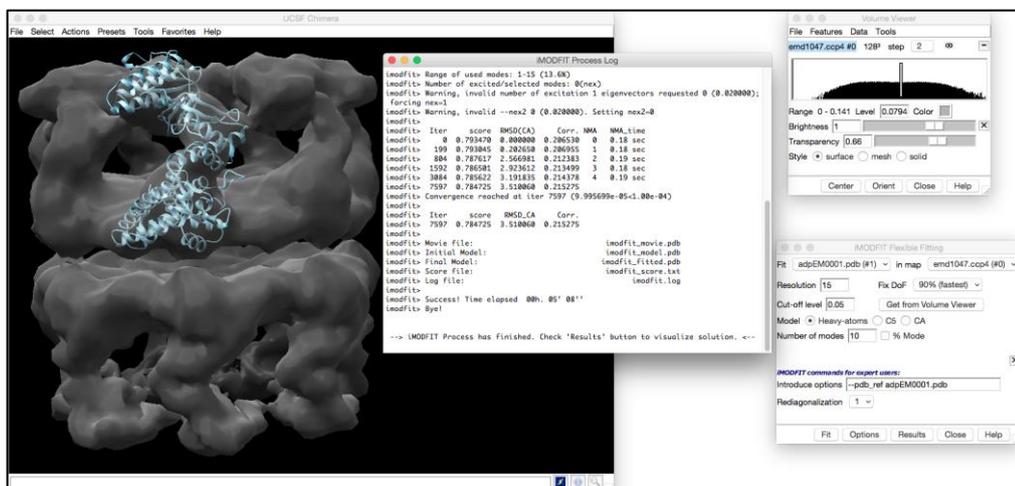
Figure 33: *iMODFIT* process finished. Notice how all the buttons are now enabled.

51

Once the process is finished, the log informs that the user can check the solutions (Figure 33), and the user can visualize the solutions in the *Results* panel which can be seen in Figure 34:
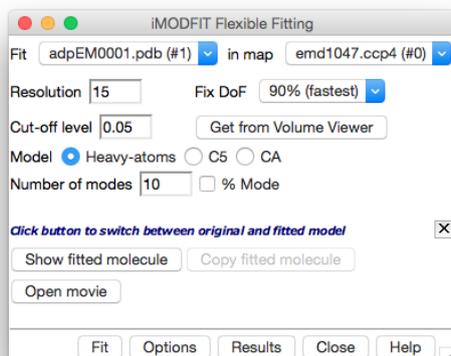
Figure 34: *iMODFIT Results* panel.

Additionally, *iMODFIT* generates five files in the working directory:

- **imodfit_fitted.pdb**: fitted atomic structure

- **imodfit_movie.pdb**: multi-pdb trajectory movie

- **imodfit_score.pdb**: score file to check for convergence

- **imodfit_model.pdb**: original atomic structure

- **imodfit.log**: used command log

As can be seen, the main options keep visible to let the user perform another flexible fitting if it is necessary. The button *Show fitted molecule* switches between the original molecule and the fitted one generated (Figure 35 and 36) and enables, when showing the fitted molecule, the possibility of copying it (Figure 37 and 38).
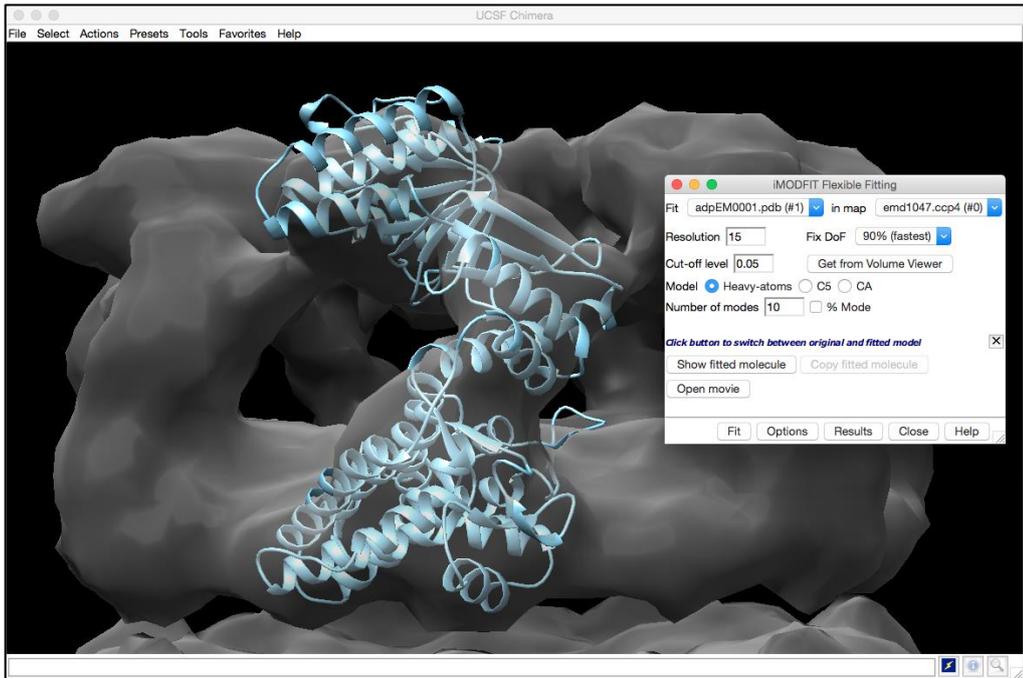
Figure 35: *iMODFIT* showing the original molecule.
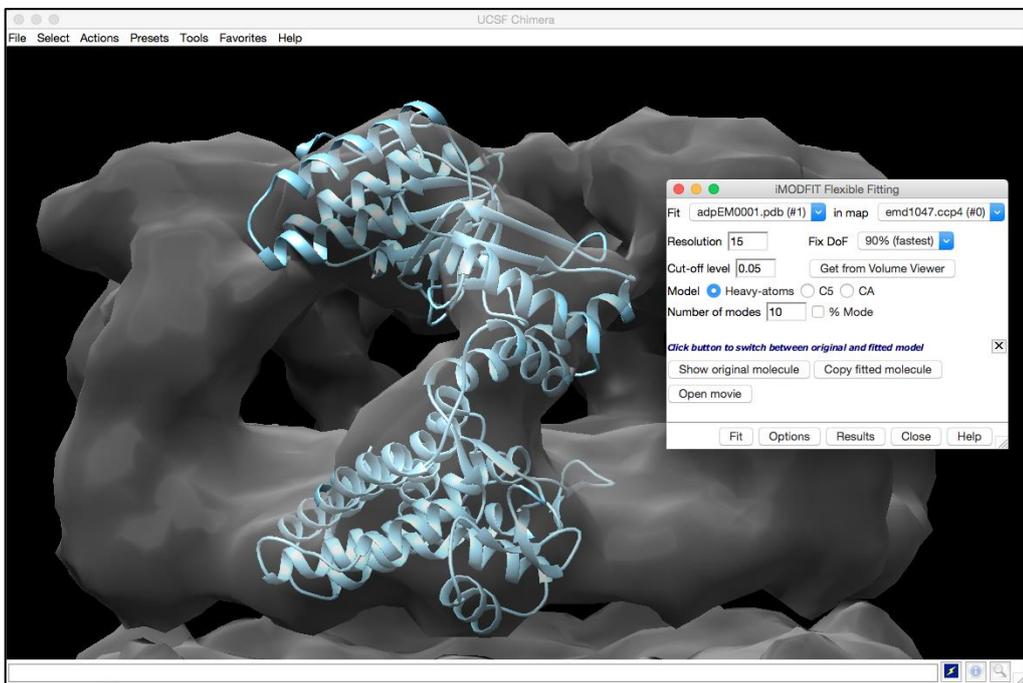
Figure 36: *iMODFIT* showing the fitted molecule.
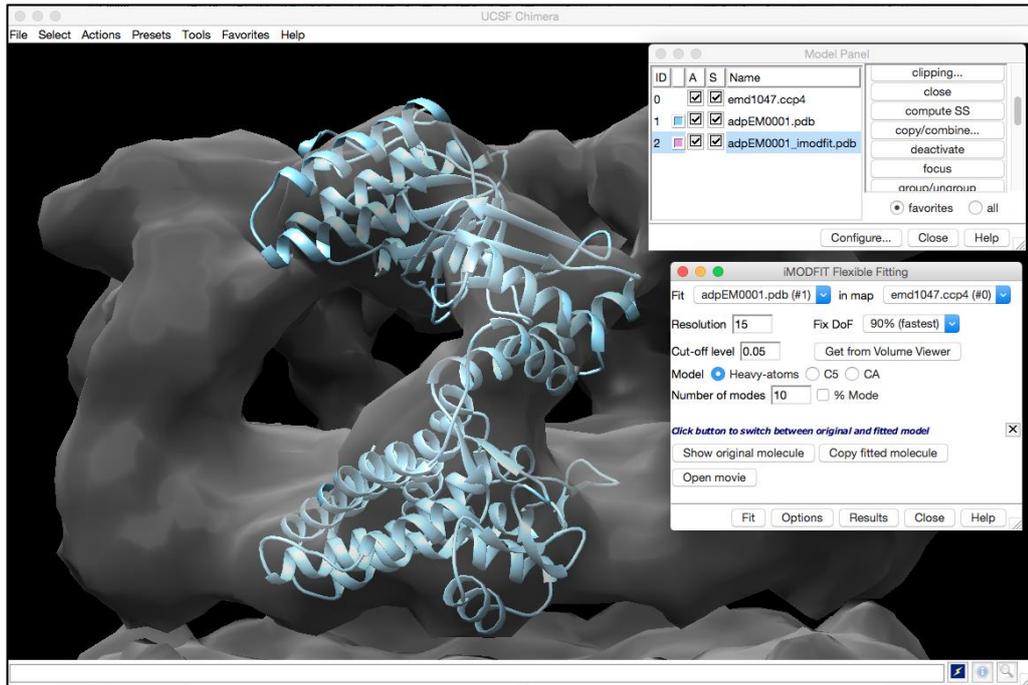
Figure 37: *iMODFIT* showing the fitted molecule and its copy in the Model Panel.
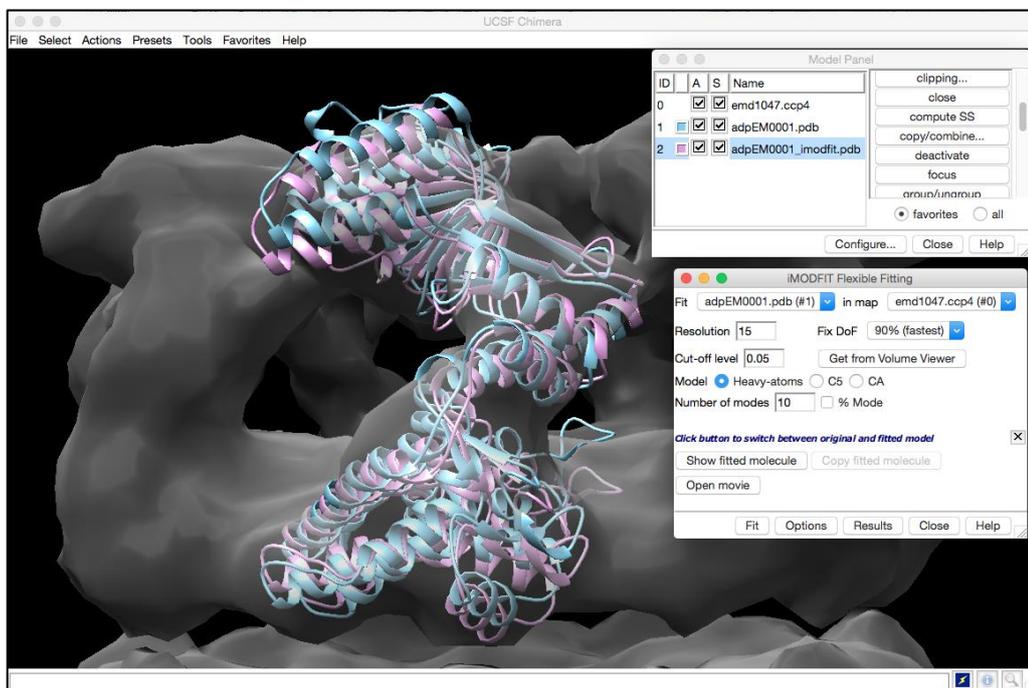
Figure 38: *iMODFIT* showing the original molecule and the fitted copy.

As *iMODFIT* generates a multi-pdb trajectory movie, there has been included another option, *Open movie*, that allows the user to open this file and check all the different trajectories that were generated in the flexible fitting (Figure 39 and 40).
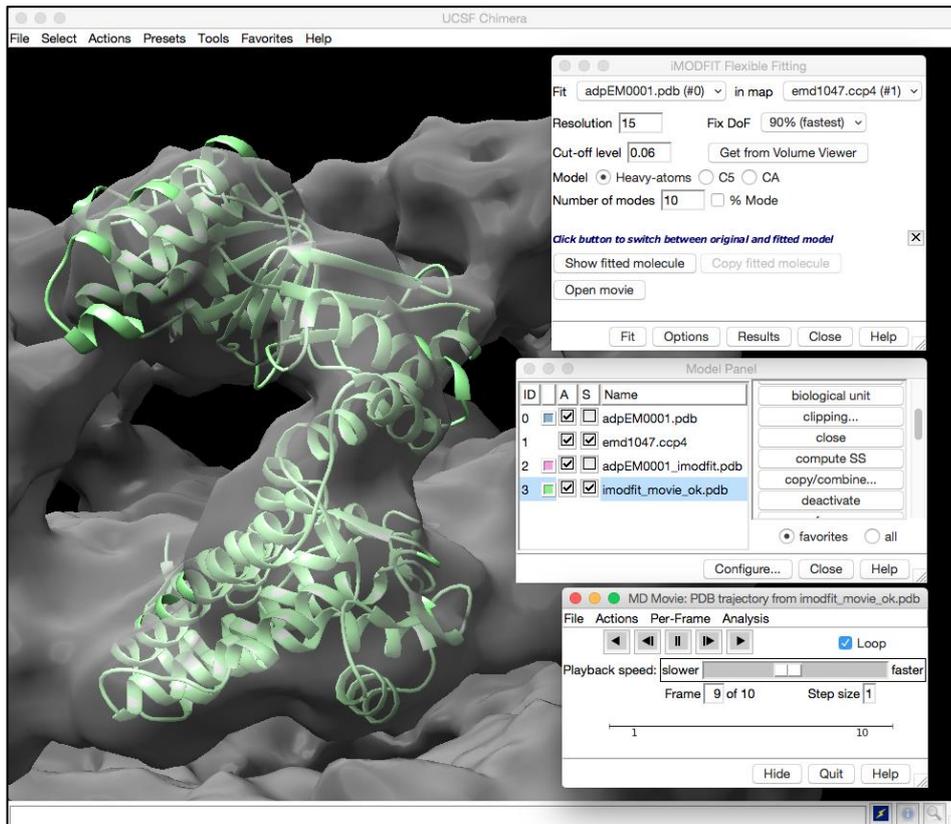
Figure 39: Frame 9 of the trajectory movie generated by *iMODFIT*.
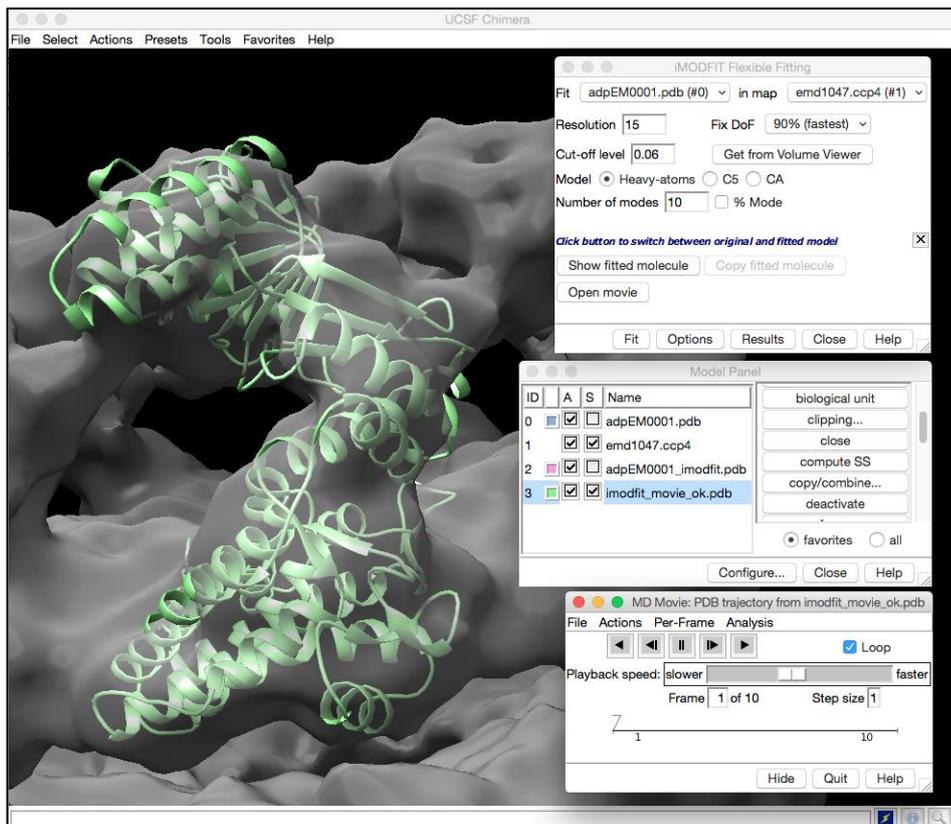
Figure 40: Frame 1 of the trajectory movie generated by *iMODFIT*.

Briefly, the *Help* button opens a user guide in the browser with instructions to load correctly the plugin in Chimera and a description of the parameters involved to use them properly.
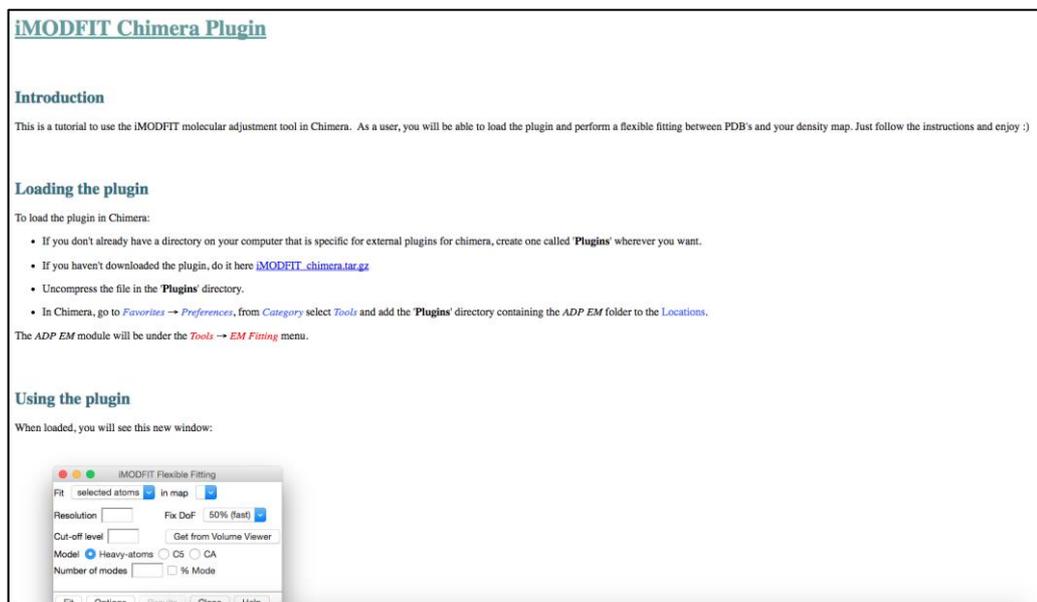
Figure 41: A snapshot of the user guide.

In later chapters the dissemination of this plugin will be described as well as the different guides and considerations for its correct usage.

3.4 Plugins Testing

In this section the different tests made for checking the correct functionality of the *ADP_EM* and *iMODFIT* plugins will be described. As described before, Chimera requires a restart every time a change its made in a plugin, so there is no way to create an isolated test to check some feature. In this context, each test is a result of making a modification in the plugin code, restarting Chimera, and testing such a change.

1.	Open Chimera and load the *ADP_EM* plugin with no data.	✓

2.	Open Chimera and load the *ADP_EM* plugin with some molecule and *EM* map previously loaded. Check if the objects appear in the plugin to be selected.	✓
3.	Open Chimera, load the *ADP_EM* plugin and display the *Options* panel.	✓
4.	Open Chimera, load the *ADP_EM* plugin and open the help guide in the browser with the *Help* button.	✓
5.	Open Chimera, load the *ADP_EM* plugin and close it with *Close* button correctly.	✓
6.	Open Chimera, load the *ADP_EM* plugin with no data and try to perfom a fitting. The message *Choose model and map* is shown.	✓
7.	Open Chimera, load the *ADP_EM* plugin with some molecule and *EM* map previously loaded and try to perfom a fitting. The message *Cutoff must be defined* is shown.	✓
8.	Open Chimera, load the *ADP_EM* plugin with some molecule and *EM* map previously loaded and get the cutoff level from the *Volume Viewer* dialog.	✓
9.	Open Chimera, load the *ADP_EM* plugin with some molecule and *EM* map previously loaded, cutoff defined and try to perfom a fitting. The message *Resolution must be defined* is shown.	✓
10.	Open Chimera, load the *ADP_EM* plugin with some molecule and *EM* map previously loaded, cutoff defined, resolution defined to 70 or -2 and try to perfom a fitting. The message *Resolution must be between 0 and 59* is shown.	✓
11.	Open Chimera, load the *ADP_EM* plugin with some molecule and *EM* map previously loaded, cutoff defined, resolution defined and try to perfom a fitting.	✓
12.	Open Chimera, load the *ADP_EM* plugin with all the existing parameters introduced and check that all the values are properly set into the plugin.	✓
13.	Open Chimera, load the *ADP_EM* plugin with the minimum required parameters and some advanced options introduced and try to perform a fitting.	✓
14.	Check that the process log window opens correctly and shows the *ADP_EM* process log when performing a fitting. Also check the progress bar status works fine.	✓
15.	Check that the *Fit, Options, Results* and *Close* buttons	✓

	are disabled when performing a fitting.	
16.	Check that the *Fit, Options, Results* and *Close* buttons are enabled when fitting process is finished.	✓
17.	Display the *Results* panel when the fitting process is finished.	✓
18.	Check that all the *ADP_EM* solutions are set to the drop-down menu in the *Results* panel.	✓
19.	Check that the molecule moves correctly to the position of the first chosen *ADP_EM* solution.	✓
20.	Check that the molecule switches correctly its position between different chosen *ADP_EM* solutions.	✓
21.	Check that a copy of the chosen *ADP_EM* solution is created in the *Model Panel* when *Save solution* button is pressed.	✓
22.	Check that the copy of the chosen *ADP_EM* solution is the same and is in the same position as it.	✓
23.	Check that *ADP_EM* plugin removes all temporal files generated during the fitting process.	✓
24.	Check that *ADP_EM* plugin does not create *PDBs* with the solutions and store all in memory.	✓
25.	Check that *ADP_EM* plugin creates pipes between Chimera and the fitting process correctly.	✓
26.	Check that *ADP_EM* plugin communicates bidirectionally between Chimera and the fitting process.	✓
27.	Check that *ADP_EM* can load files from different locations and perform the fitting correctly.	✓
28.	Check that *ADP_EM* communicates properly with Chimera native dialogs.	✓
29.	Check that *ADP_EM* is able to make another fitting when a previous one is finished and the *Results* panel is being displayed.	✓
30.	Check that data stored in memory from a previous fitting is properly cleaned to ensure a correct functioning when performing a new one.	✓
31.	Check that *ADP_EM* prevents Chimera blocking when performing a fitting.	✓
32.	Open Chimera and load the *iMODFIT* plugin with no data.	✓

33.	Open Chimera and load the *iMODFIT* plugin with some molecule and *EM* map previously loaded. Check if the objects appear in the plugin to be selected.	✓
34.	Open Chimera, load the *iMODFIT* plugin and display the *Options* panel.	✓
35.	Open Chimera, load the *iMODFIT* plugin and open the help guide in the browser with the *Help* button.	✓
36.	Open Chimera, load the *iMODFIT* plugin and close it with *Close* button correctly.	✓
37.	Open Chimera, load the *iMODFIT* plugin with no data and try to perfom a fitting. The message *Choose model and map* is shown.	✓
38.	Open Chimera, load the *iMODFIT* plugin with some molecule and *EM* map previously loaded and try to perfom a fitting. The message *Cutoff must be defined* is shown.	✓
39.	Open Chimera, load the *iMODFIT* plugin with some molecule and *EM* map previously loaded and get the cutoff level from the *Volume Viewer* dialog.	✓
40.	Open Chimera, load the *iMODFIT* plugin with some molecule and *EM* map previously loaded, cutoff defined and try to perfom a fitting. The message *Resolution must be defined* is shown.	✓
41.	Open Chimera, load the *iMODFIT* plugin with some molecule and *EM* map previously loaded, cutoff defined, resolution defined to 70 or -2 and try to perfom a fitting. The message *Resolution must be between 0 and 59* is shown.	✓
42.	Open Chimera, load the *iMODFIT* plugin with some molecule and *EM* map previously loaded, cutoff defined, resolution defined and try to perfom a fitting.	✓
43.	Open Chimera, load the *iMODFIT* plugin with all the existing parameters introduced and check that all the values are properly set into the plugin.	✓
44.	Open Chimera, load the *iMODFIT* plugin with the minimum required parameters and some advanced options introduced and try to perform a fitting.	✓
45.	Check that the process log window opens correctly and shows the *iMODFIT* process log when performing a fitting.	✓
46.	Check that the *Fit, Options, Results* and *Close* buttons	✓

	are disabled when performing a fitting.	
47.	Check that the *Fit, Options, Results* and *Close* buttons are enabled when fitting process is finished.	✓
48.	Display the *Results* panel when the fitting process is finished.	✓
49.	Check that the *imodfit_fitted.pdb* file is generated when the *iMODFIT* fitting process is finished.	✓
50.	Check that the *imodfit_model.pdb* file is generated when the *iMODFIT* fitting process is finished.	✓
51.	Check that the *imodfit.log* file is generated when the *iMODFIT* fitting process is finished.	✓
52.	Check that the *imodfit_score.txt* file is generated when the *iMODFIT* fitting process is finished.	✓
53.	Check that the *imodfit_movie.pdb* file is generated when the *iMODFIT* fitting process is finished.	✓
54.	Check that the molecule updates correctly its position to the fitted one when the *Show fitted molecule* button is pressed. Also check that the name button is changed to *Show original molecule*.	✓
55.	Check that the *Copy fitted molecule* button is enabled *Show fitted molecule* button is pressed.	✓
56.	Check that a copy of the *iMODFIT* fitted molecule is created in the *Model Panel* when *Copy fitted molecule* button is pressed.	✓
57.	Check that the fitted molecule updates correctly its position to the original one when the *Show original molecule* button is pressed. Also check that the name button is changed to *Show fitted molecule*.	✓
58.	Open the trajectory movie generated by *iMODFIT* when the *Open movie* button is pressed.	✓
59.	Check that the trajectory movie generated by *iMODFIT* is shown in the *Model Panel*.	✓
60.	Check that the trajectories are properly set in the movie and all the frames were generated correctly. movie generated by *iMODFIT* is shown in the *Model Panel*.	✓
61.	Check that *iMODFIT* plugin removes all temporal files generated during the fitting process.	✓
62.	Check that *iMODFIT* plugin creates pipes between Chimera and the fitting process correctly.	✓

63.	Check that *iMODFIT* plugin communicates bidirectionally between Chimera and the fitting process.	✓
64.	Check that *iMODFIT* can load files from different locations and perform the fitting correctly.	✓
65.	Check that *iMODFIT* communicates properly with Chimera native dialogs.	✓
66.	Check that *iMODFIT* is able to make another fitting when a previous one is finished and the *Results* panel is being displayed.	✓
67.	Check that data stored in memory from a previous fitting is properly cleaned to ensure a correct functioning when performing a new one.	✓
68.	Check that *iMODFIT* prevents Chimera blocking when performing a fitting.	✓

3.5 Dissemination of the Plugins

In order to make available to the scientific community the developed bioinformatics tools, a long ago was created a dedicated website by the receiving group, http://chaconlab.org/, where the user can find all the necessary documentation organized in the following sections:

- **User guide**: it explains the basic and advanced options of the different tools.

- **Tutorial**: it comments practical examples with usage tips and instructions to quickly and easily perform the most common tasks of each of the tools.

- **Frequently Asked Questions (FAQ)**: this section contains answers to the most common questions on how to use and customize the different tools.

- **Other resources**: this includes links to programs and external servers that may be useful for, for example, repairing atomic models, obtaining atomic structures and density maps, or evaluating the quality of structures.

- **Installation**: it explains how to install the different bioinformatics tools.

Now, a new **Chimera** section has been included and the user can freely download the *ADP_EM* and the *iMODFIT* plugins for UCSF Chimera:

ADP_EM

http://chaconlab.org/hybrid4em/adp-em/adpem-chimera
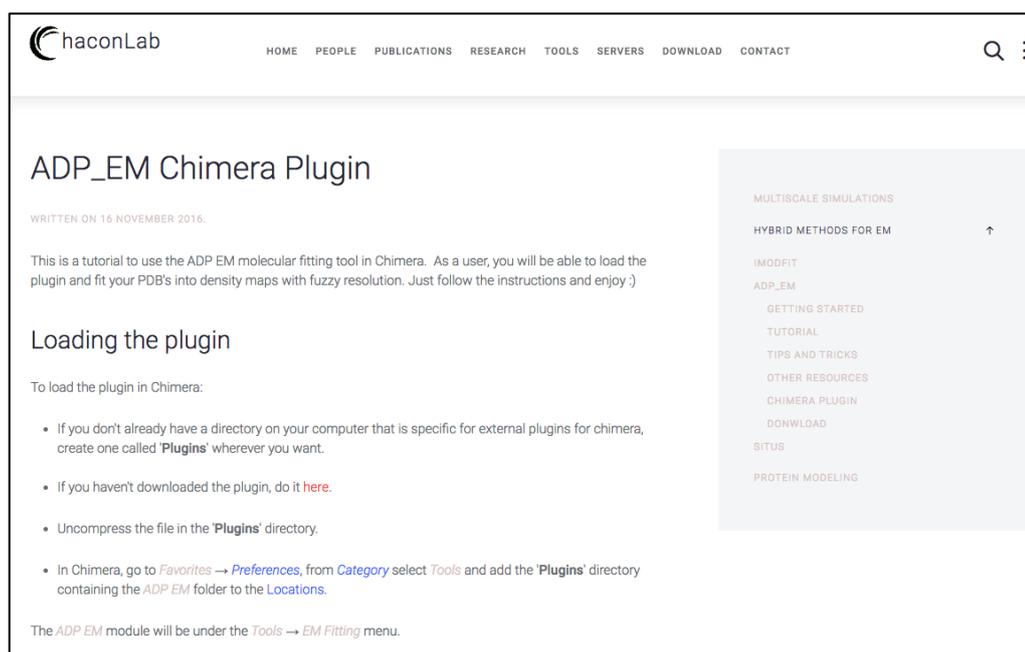
Figure 42: Snapshot of the *ADP_EM* Chimera plugin the receiving group web.

iMODFIT

http://chaconlab.org/hybrid4em/imodfit/imod-chimera
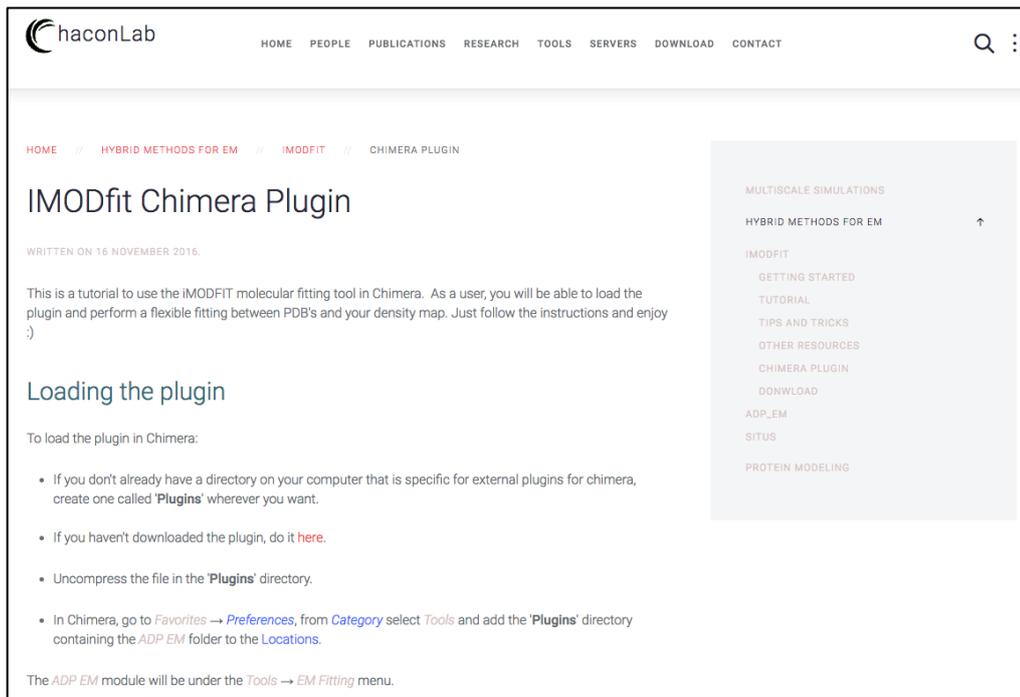
Figure 43: : Snapshot of the *iMODFIT* Chimera plugin the receiving group web.

The complete code of the developed plugins is also freely available to users in a *GitHub* repository from which they can be downloaded and/or consulted:

- *ADP_EM* → https://github.com/pablosolar/adp_em

- *iMODFIT* → https://github.com/pablosolar/imodfit

3.6 Conclusions

Training Stage

During the implementation of the project, many aspects of structural biology have been learned, such as biophysical techniques for obtaining data or the main methods of fitting, how and when they are applied, what tools are currently used and which are expected to be developed in one future.

63

In particular, I got used to the state of the art of the fitting software used in the main field, including *ADP_EM*, *iMODFIT*, *Situs* (developed in the group), *mdff*, *gEMfitter*, *Integrative Modelling,* and *Rosetta*). In addition, familiarization with the tools available for structural analysis, such as UCSF Chimera, has been essential in achieving the success of the project.

Development Stage

A set of plugins for making rigid (*ADP_EM*) and flexible (*iMODFIT*) fittings have been designed and implemented for Chimera, which are:

- Applicable to large macromolecules (proteins, nucleic acids, multiple chains, and small rigid ligands).

- Capable of using different reductionist approaches.

- Efficient and easy to configure.

- Accessible and documented via the web.

- Compatible with the original versions of the methods.

- Intuitive to users, dynamic and light running.

- Robust and optimized for integration into Chimera.

It has been systematically validated that the solutions generated by the plugins and the execution times are exactly the same as those generated by the original methods original methods.

The methodology used and the planning scheduled to carry out the project has been satisfactory. The only significant deviation is the pending development of the plugin for *COLORES*.

Looking to the future, the integration of more fitting tools as Chimera plugins together with further development of current

ADP_EM and *iMODFIT* versions, **would be worth publishing in a specialized high impact journal**.

From a global and global perspective, the project has proved to be a success and will have direct applications and real uses from the moment it is disseminated.

3.7 Acknowledgements

4. Glossary

3D: Tridimensional.

ADP_EM: Another Docking Platform - Electron Microscopy.

B: Bandwidth.

C++: general-purpose programming language (Object Oriented).

CC: Cartesian Coordinates.

COLORES: COrrelation based LOw RESolution.

CSIC: Spanish National Research Council (*Consejo Superior de Investigaciones Científicas)*.

DoF: degrees of freedom.

EM: Electron Microscopy.

EMAN: software package for reconstructing 3d models from a set of randomly oriented particle images.

FFT: Fast Fourier Transform.

FRM: Fast Rotational Matching.

FTM: Fast Translational Matching.

GUI: Graphic User Interface

HA: Heavy Atoms.

IC: Internal Coordinates.

MD: Molecular Dynamics.

NMA: Normal Mode Analysis.

PDB: Protein Data Bank.

RMSD: Root Mean Square Deviation.

5. Bibliography

1.   Bahar, I., Lezon, T.R., Yang, L.W. y Eyal, E. (2010b) Global dynamics of proteins: bridging between structure and function, Annu Rev Biophys, 39, 23-42.

2.   Bray, J.K., Weiss, D.R., Levitt, M., 2011. Optimized torsion-angle normal modes reproduce conformational changes more accurately than cartesian modes. Biophys J 101, 2966-2969.

3.   Ceulemans, H. y Russell, R.B. (2004) Fast fitting of atomic structures to low-resolution electron density maps by surface overlap maximization, J Mol Biol, 338, 783-793.

4.   Chacón, P. and W. Wriggers (2002). "Multi-resolution contour-based fitting of macromolecular structures." Journal of molecular biology 317(3): 375-384.

5.   Eisenstein, M. and Katchalski-Katzir, E.(2004) On proteins, grids, correlations, and docking. C. R. Biologies. 327(5): p. 409-420.

6.   Fabiola, F., Chapman, M.S. (2005) Fitting of high-resolution structures into electron microscopy reconstruction images, Structure, 13, 389-400.

7.   Gabb, H.A., Jackson, R.M. and Sternberg, M.J. (1997) Modelling protein docking using shape complementarity, electrostatics and biochemical information, J Mol Biol, 272, 106-120.

8.   Garzón, J. I., J. Kovacs, R. Abagyan and P. Chacón (2007). "ADP_EM: fast exhaustive multiresolution docking for high-throughput coverage." Bioinformatics 23(4): 427-433.

9.   Jiang, W., Baker, M.L., Ludtke, S.J. y Chiu, W. (2001) Bridging the information gap: computational tools for intermediate resolution structure interpretation, J Mol Biol, 308, 1033-1044.

10. Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A.A., Aflalo, C., and Vakser, I.A.(1992) Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. Proc. Natl. Acad. Sci. USA. 89(6): p. 2195-2199.

11. Kovacs, J.A. and Wriggers, W.(2002) Fast rotational matching. Acta Crystallogr. D. Biol. Crystallogr. 58(8): p. 1282-1286.

12. Kovacs, J.A., Chacon, P., Cong, Y., Metwally, E. and Wriggers, W. (2003) Fast rotational matching of rigid bodies by fast Fourier transform acceleration of five degrees of freedom, Acta Crystallogr D Biol Crystallogr, 59, 1371-1376.

13.   Kovacs, J.A., Cavasotto, C.N., Abagyan, R., 2005. Conformational sampling of protein flexibility in generalized coordinates: Application to ligand docking. J Comput Theor Nanosci 2, 354-361.

14.   López-Blanco, J. R. and Chacón, P. (2015), Structural modeling from electron microscopy data. WIREs Comput Mol Sci, 5: 62–81. doi:10.1002/wcms.1199

15.   Lopéz-Blanco, J. R. and P. Chacón (2013). "iMODFIT: efficient and robust flexible fitting based on vibrational analysis in internal coordinates." Journal of Structural Biology 184(2): 261–270.

16.   Lopez-Blanco, J.R., Garzon, J.I., Chacon, P., 2011. iMod: multipurpose normal mode analysis in internal coordinates. Bioinformatics 27, 2843-2850.

17.   Lu, M., Poon, B., Ma, J., 2006. A New Method for Coarse-Grained Elastic Normal-Mode Analysis. J Chem Theory Comput 2, 464-471.

18.   Ma, J. (2005) Usefulness and limitations of normal mode analysis in modeling dynamics of biomolecular complexes, Structure, 13, 373-380.

19.   Mendez, R., Bastolla, U., 2010. Torsional network model: normal modes in torsion angle space better correlate with conformation changes in proteins. Phys Rev Lett 104, 228103-228107.

20.   Rath, B.K., Hegerl, R., Leith, A., Shaikh, T.R., Wagenknecht, T. and Frank, J. (2003) Fast 3D motif search of EM density maps using a locally normalized cross-correlation function, J Struct Biol, 144, 95-103.

21.   Ritchie, D.W. and Kemp, G.J.(2000) Protein docking using spherical polar Fourier correlations. Proteins. 39(2): p. 178-194.

22.   Roseman, A.M. (2000) Docking structures of domains into maps from cryo-electron microscopy using local correlation, Acta Crystallogr D Biol Crystallogr, 56, 1332-1340.

23.   Rossmann, M.G. (2000) Fitting atomic models into electron-microscopy maps, Acta Crystallogr D Biol Crystallogr, 56, 1341-1349.

24.   Vakser, I.A., Matar, O.G. and Lam, C.F. (1999) A systematic study of low-resolution recognition in protein--protein complexes, Proc Natl Acad Sci U S A, 96, 8477-8482.

25.   Volkmann, N. y Hanein, D. (2003) Docking of atomic models into reconstructions from electron microscopy, Methods Enzymol, 374, 204-225.

26.   Wriggers, W. and Chacon, P. (2001) Modeling tricks and fitting techniques for multiresolution structures, Structure (Camb), 9, 779-788.

27.   Wriggers, W., Milligan, R.A. y McCammon, J.A. (1999) Situs: A package for docking crystal structures into low-resolution maps from electron microscopy, J Struct Biol, 125, 185-195.

6. Appendants

6.1 *ADP_EM* Relevant Code

Although in section 3.5 corresponding to the dissemination of the plugins a link has been cited to the complete code developed for each one of them, below is the most relevant code, commented appropriately, starting with the ADP.

```python
# -----------------------------------------------------------------
# Class to register the ADP EM plugin in Chimera.
# The plugin will be located in EM Fitting/Volume Data menu
#
import chimera.extension


# -----------------------------------------------------------------
#
class ADPEM_EMO(chimera.extension.EMO):

    def name(self):
        return 'ADP EM'
    def description(self):
        return 'Fitting density map in an exhaustive way through ADP_EM Algorithm'
    def categories(self):
        return ['EM Fitting']
    def icon(self):
        return None
    def activate(self):
        self.module('adpgui').show_adp_em_dialog()
        return None

chimera.extension.manager.registerExtension(ADPEM_EMO(__file__))
```

> *In iMODFIT, this is similar*

```python
# -----------------------------------------------------------------
# Dialog to perform rigid fitting through ADP EM algorithm.
#

import chimera
from chimera.baseDialog import ModelessDialog
import ADP


# -----------------------------------------------------------------
# ADP EM Plugin Dialog
#
class ADP_EM_Dialog(ModelessDialog):
```

```python
# Title of APD EM plugin
title = 'ADP EM Exhaustive Fitting'
# Name of ADP EM plugin
name = 'ADP EM'
# Buttons of ADP EM GUI
buttons = ('Fit', 'Options', 'Results', 'Close')
# Path of help guide of ADP EM plugin
help = ('adp_em.html', ADP)
# Name of the folder where ADP EM plugin is located
plugin_folder = 'ADP/'
# Path of the process of ADP EM
adp_em = plugin_folder + 'adp_em'
# Steps for the process progress bar
steps = ["Step 1", "Step 2", "Step 3"]
# Steps marks for the process progress bar
marks = ["ADP_EM", "Trans limit", "Interpolation", "Total Time"]
# Variable to represent the dropdown menu that will show all the ADP EM Solutions
mb = None
# Array to store the solutions generated by ADP EM
solutions_chimera = []
# Variables to handle a possible movement of the molecule by the user
# They will store an Xform object (gives the rotation and traslation for a model)
# relative to the pdb, the map, its inverse or the Xform of the last solution chosen
# It works as a movements traceback
bos = None
xf = None
xfC = None
xform_last_solution = None


#----------------------
# ADP EM Chimera Commands

# ADP in Chimera indicator
adp_em_chimera_opt = "--chimera"

# No solutions
adp_em_chimera_no_save = "--no_save"

# Fitting criterion
adp_em_chimera_laplacian = "-f"
adp_em_chimera_laplacian_val = "1"

# Saved solutions
adp_em_chimera_saved_solutions = "-n"
adp_em_chimera_saved_solutions_val = "50"

# Translational sampling
adp_em_chimera_sampling = "-t"
adp_em_chimera_sampling_val = "2"

# Translational scan
adp_em_chimera_scan = "-s"
adp_em_chimera_scan_val = "2"

# Peaks explored per docking
adp_em_chimera_peaks_explored = "--ne"
adp_em_chimera_peaks_explored_val = "30"
```

```python
# Peaks stored per iteration
adp_em_chimera_peaks_iteration = "--np"
adp_em_chimera_peaks_iteration_val = "20"

# Peaks stored per search
adp_em_chimera_peaks_search = "--nr"
adp_em_chimera_peaks_search_val = "100"

# Peaks stored per multidocking search
adp_em_chimera_peaks_msearch = "--nrm"
adp_em_chimera_peaks_msearch_val = "500"

# Translational threshold
adp_em_chimera_translational_threshold = "--rt"
adp_em_chimera_translational_threshold_val = "2.0"

# Rotational threshold
adp_em_chimera_rotational_threshold = "--rc"
adp_em_chimera_rotational_threshold_val = None

# Width between spherical layers
adp_em_chimera_width_layers = "--lw"
adp_em_chimera_width_layers_val = "1.0"

# Density cutoff of simulated map
adp_em_chimera_cutoff_simulated = "--cutoff2"
adp_em_chimera_cutoff_simulated_val = "0.0"
```

```python
# -------------------------------------------------------------------------
# Performs the ADP EM process
#
def Fit(self):

    # If a fitting is performed when Results panel is active, close it
    for widget in self.mmf.winfo_children():
        widget.destroy()
        # and clean the array which store the solutions (previous fitting)
        self.solutions_chimera = []
    self.results_panel.set(False)

    # Validation of the parameters introduced by the user
    if self.check_models() is False:
        return

    # Disable Fit, Options and Close buttons when ADP EM process is performed
    self.disable_process_buttons()

    # Retrieve plugin path
    self.plugin_path = __file__[:__file__.index(self.plugin_folder)]
```

```python
#-----------------------
# Calling ADPEM process
#-----------------------
from subprocess import STDOUT, PIPE, Popen
import os, sys

# Get the full path of ADP EM process
command = self.plugin_path + self.adp_em
# Set the workspace
cwd = self.plugin_path + self.plugin_folder
# Set the file name that will be generated by ADP EM with all the solutions parameters:
# center of mass, Euler Angles and traslations of the PDB solutions
self.filename_chimera = cwd + "chimera.bin"

# PDB selected in the menu
pdbSelected = self.object_menu.getvalue()
# Map selected in the menu
mapSelected = self.map_menu.volume()
# Temporal pdb that will be wrote when performing the process.
# It is necessary because if the user moves the molecule, the camera changes, and the
# coordinates and the internal states of the pdb and the map remain inconsistent
# So saving the pdb ensures that the ADP EM process is going to be done with consistent
#  values relative to the map
pdb_path = cwd + "temporal_" + pdbSelected.name

# Save the pdb xform just before performing the fitting
self.saved_pdb = pdbSelected.openState.xform
# Save the pdb xform premultiplied by the map inverse xform
# This is necessary to be able to move the molecule to the origin regardless the ADP EM process
self.xf = pdbSelected.openState.xform
self.xf.premultiply(mapSelected.openState.xform.inverse())

# Back to origin (Ensure that pdb and map have the same internal state when performing ADP EM
process
pdbSelected.openState.xform = mapSelected.openState.xform

# Save pdb relative to the map
from Midas import write
write(pdbSelected, mapSelected, pdb_path)

# Update user view
pdbSelected.openState.xform = self.saved_pdb

# Record position state
self.record_position_state()

# Get options values
self.get_options_chimera()

# Variable to move to the center of mass when moving the pdb to the first ADP EM solution
self.fitting_center_mass = True
```

```python
# Retrieve the full command to perform the fitting: ap_em + arguments
cmd = [command, mapSelected.openedAs[0], pdb_path,
    self.bandwidth.variable.get(), self.cutoff.get(), self.resolution.get(),
    self.adp_em_chimera_laplacian, self.adp_em_chimera_laplacian_val,
    self.adp_em_chimera_saved_solutions, self.adp_em_chimera_saved_solutions_val,
    self.adp_em_chimera_sampling, self.adp_em_chimera_sampling_val,
    self.adp_em_chimera_scan, self.adp_em_chimera_scan_val,
    self.adp_em_chimera_peaks_explored, self.adp_em_chimera_peaks_explored_val,
    self.adp_em_chimera_peaks_iteration, self.adp_em_chimera_peaks_iteration_val,
    self.adp_em_chimera_peaks_search, self.adp_em_chimera_peaks_search_val,
    self.adp_em_chimera_peaks_msearch, self.adp_em_chimera_peaks_msearch_val,
    self.adp_em_chimera_translational_threshold,
self.adp_em_chimera_translational_threshold_val,
    self.adp_em_chimera_rotational_threshold, self.adp_em_chimera_rotational_threshold_val,
    self.adp_em_chimera_width_layers, self.adp_em_chimera_width_layers_val,
    self.adp_em_chimera_cutoff_simulated, self.adp_em_chimera_cutoff_simulated_val,
    self.adp_em_chimera_no_save, self.adp_em_chimera_opt]

# Execute the command with the respective arguments creating pipes between the process and
Chimera
# Pipes will be associated to the standard output and standard error required to show the process log
# in the window
adp_em_process = Popen(cmd, stdout=PIPE, stderr=PIPE, cwd=cwd, universal_newlines=True)

# Text widget for process log that will showthe standard output of the process
from Tkinter import *
root = Tk()
root.wm_title("ADP EM Process Log")
import ttk
self.var_det = IntVar(root)
self.pbar_det = ttk.Progressbar(root, orient="horizontal", length=400, mode="determinate",
variable=self.var_det, maximum=100)
self.pbar_det.pack(side=TOP, fill=X)
S = Scrollbar(root)
T = Text(root, height=30, width=85)
S.pack(side=RIGHT, fill=Y)
T.pack(side=LEFT, fill=Y)
S.config(command=T.yview)
T.config(yscrollcommand=S.set)

# Read first line
line = adp_em_process.stdout.readline()
# Variable to check the process status and show its output in a friendly format to the user
process_progress = False
# Continue reading the standard output until ADP EM is finished
# If the current line is an iteration for a model, replace in the widget the last showed
# If it is a new model or is part of the ADP EM process, inserts the line at the end of the widget
while line:
  if process_progress is False:
    index_before_last_print = T.index(END)
    T.insert(END, line)
  else:
    T.delete(index_before_last_print + "-1c linestart", index_before_last_print)
    T.insert(END, line)
```

```python
      T.update()
      T.yview(END)
      self.check_steps_marks(line)
      line = adp_em_process.stdout.readline()
      if '[' in line and '%' in line:
        process_progress = True
      elif '100% Finish!':
        process_progress = False
      sys.stdout.flush()
    T.insert(END, "\n --> ADP EM Process has finished. Check 'Results' button to visualize solutions. <--
\n")
    T.update()
    T.yview(END)
    # When ADP EM process is finished, the results are set into the Results panel...
    self.fill_results()
    # and the plugin buttons are enabled again
    self.enable_process_buttons()

    # Remove the temporal pdb
    os.remove(pdb_path)
```

```python
  # -----------------------------------------------------------------------
  # Records position state between pdb and map
  #
  def record_position_state(self):

    self.bos = self.map_menu.volume().openState
    bxfinv = self.bos.xform.inverse()
    self.xf = self.object_menu.getvalue().openState.xform
    self.xf.premultiply(bxfinv)

  # -----------------------------------------------------------------------
  # Moves the molecule to the center of mas
  #
  def move_center_mass(self):

    # Get the molecule
    m = self.object_menu.getvalue()
    # Import adphandle to handle pdb transformations
    import adphandle as adph

    # To move the center of mass, Euler Angles should be (0, 0, 0)
    ea = map(float, "0 0 0".split())
    # The position (traslation) to the pdb center of mass is given by ADP EM
    t = map(float, self.com)

    # First apply rotation, then traslation and finally get the related Xform
    self.xf_center_mass = adph.euler_xform_adp(ea, t)
    # Updates the coordinates of the molecule with the xform of the center of mass
    adph.transform_atom_coordinates_adp(m.atoms, self.xf_center_mass)
```

```python
# -----------------------------------------------------------------------------
# Shows a solution generated by ADP EM from the drop-down in the Results Panel
#
def show_adp_solution (self):

  # If an ADP EM solution is chosen...
  if not '---' in self.mb.variable.get():

    # Import adphandle to handle pdb transformations
    import adphandle as adph

    # Get the molecule
    m = self.object_menu.getvalue()

    # Roll back previous state. Set the molecule to the previous state
    self.back_current_state_adp()

    # Go back to origin...
    if self.xform_last_solution is None:
      # if a solution is chosen fot the first time, it is necessary to move the molecule to the center of
mass
      self.move_center_mass()
    else:
      # if not, move the molecule to the previous position to the one that was in the last solution chosen
      adph.transform_atom_coordinates_adp(m.atoms, self.xform_last_solution.inverse())

    # -----------------------
    # Get selected solution
    # -----------------------
    # Solution_X  = [Euler Angles(Phi, Theta, Psi), Traslation(X, Y, Z)]
    self.adp_chosed_solution = self.get_adp_chosed_solution(self.mb.variable.get())
    so = self.solutions_chimera[self.adp_chosed_solution - 1]

    # Get the Euler Angles associated with the solution
    ea = map(float, so[len(so) / 2:])

    # Get the traslation associated with the solution
    t = map(float, so[:len(so) / 2])

    # -----------------------
    # Xform of the solution
    # -----------------------
    # First apply rotation, then traslation and finally get the related Xform
    xform_solution = adph.euler_xform_adp(ea, t)

    # Updates the coordinates of the molecule with the xform of the chosen solution
    # This moves it into the position of the chosen ADP EM solution
    adph.transform_atom_coordinates_adp(m.atoms, xform_solution)

    # Save current state because user may move the solution
    self.save_current_state_adp()
    self.xform_last_solution = xform_solution
```

```python
# -------------------------------------------------------------------------
# Saves the current state of the molecule relative to the map
# It is necessary because user may move the solution
#
def save_current_state_adp(self, event = None):

 m = self.object_menu.getvalue()
 v = self.map_menu.volume()
 if m is None or v is None:
  return

 # Applies the inverse of the map xform to the molecule xform to save the current
 # state and movement made
 bxfinvC = v.openState.xform.inverse()
 self.xfC = m.openState.xform
 self.xfC.premultiply(bxfinvC)

# -------------------------------------------------------------------------
# Roll back previous state. Set the molecule to the previous state
#
def back_current_state_adp(self, event = None):

 m = self.object_menu.getvalue()
 v = self.map_menu.volume()
 if m is None or v is None:
  return

 # This only applies to the first movement after process is finished
 if self.xfC is None:
  # Back to origin
  m.openState.xform = v.openState.xform
  return

 # Update the molecule xform with xform that stores the last state and movement made
 oxfC = v.openState.xform
 oxfC.multiply(self.xfC)
 m.openState.xform = oxfC
```

```python
# -------------------------------------------------------------------------
# Apply a rotation and translation to atoms.
#
def transform_atom_coordinates_adp(atoms, xform):

  for a in atoms:
     a.setCoord(xform.apply(a.coord()))

# -------------------------------------------------------------------------
# Rotation applied first, then translation.
#
def euler_xform_adp(euler_angles, translation):

  xf = euler_rotation_adp(*euler_angles)
  from chimera import Xform
  xf.premultiply(Xform.translation(*translation))
  return xf
```

```
# ---------------------------------------------------------------------------
# Convert Euler angles to an equivalent Chimera transformation matrix.
#
# Angles must be in degrees, not radians.
#
# Uses the most common Euler angle convention z-x-z (the chi-convention)
# described at
#
#   http://mathworld.wolfram.com/EulerAngles.html
#
def euler_rotation_adp(phi, theta, psi):

    from chimera import Xform, Vector
    xf1 = Xform.rotation(Vector(0,0,1), phi)    # Rotate about z-axis
    xp = xf1.apply(Vector(1,0,0))          # New x-axis
    xf2 = Xform.rotation(xp, theta)         # Rotate about new x-axis
    zp = xf2.apply(Vector(0,0,1))          # New z-axis
    xf3 = Xform.rotation(zp, psi)          # Rotate about new z-axis

    xf = Xform()
    xf.premultiply(xf1)
    xf.premultiply(xf2)
    xf.premultiply(xf3)

    return xf
```

6.2 *iMODFIT* Relevant Code

```
# -------------------------------------------------------------------------------
# Dialog to perform flexible fitting through iMODFIT algorithm.
#

import chimera
from chimera.baseDialog import ModelessDialog
import iMODFIT

# -------------------------------------------------------------------------------
# iMODFIT Plugin Dialog
#
class iMODFIT_Dialog(ModelessDialog):

    # Title of iMODFIT plugin
    title = 'iMODFIT Flexible Fitting'
    # Name of iMODFIT plugin
    name = 'iMODFIT'
```

```python
# Buttons of iMODFIT GUI
buttons = ('Fit', 'Options', 'Results', 'Close')
# Path of help guide of iMODFIT plugin
help = ('imodfit.html', iMODFIT)
# Name of the folder where iMODFIT plugin is located
plugin_folder = 'iMODFIT/'
# Path of the process of iMODFIT
imodfit = plugin_folder + 'imodfit'
# Variable to keep the workspace
cwd = None
# Name of the fitted pdb generated after iMODFIT process
fitted_molecule = "imodfit_fitted.pdb"
# Name of the trajectory movie
imovie = plugin_folder + "imodfit_movie.pdb"


#--------------------------
# iMODFIT Chimera Commands

# iMODFIT in Chimera indicator
imodfit_chimera_opt = "--chimera"

# More PDBs
imodfit_chimera_morepdbs = "--morepdbs"

# PDB Reference
imodfit_chimera_pdb_ref = "--pdb_ref"
imodfit_chimera_pdb_ref_val = None

# Trajectory (movie)
imodfit_chimera_t = "-t"

# Fixing diagonalization
imodfit_chimera_r = "-r"
imodfit_chimera_r_val = "0"

# Rediagonalization
imodfit_chimera_re = "--rediag"
imodfit_chimera_re_val = "0"

# Coarse-grained model
imodfit_chimera_m = "-m"
imodfit_chimera_m_val = "2"

# Modes range
imodfit_chimera_n = "-n"
imodfit_chimera_n_val = "0.05"

# Advanced commands
imodfit_chimera_adv_commands = []
```

```python
# ----------------------------------------------------------------------
# Performs the iMODFIT process
#
def Fit(self):

  # If a fitting is performed when Results panel is active, close it
  for widget in self.mmf.winfo_children():
    widget.destroy()
  self.results_panel.set(False)

  # Validation of the parameters introduced by the user
  if self.check_models() is False:
    return

  # Disable Fit, Options and Close buttons when iMODFIT process is performed
  self.disable_process_buttons()

  # Retrieve the full plugin path
  self.plugin_path = __file__[:__file__.index(self.plugin_folder)]

  #----------------------
  # Calling iMODFIT process
  #----------------------
  from subprocess import STDOUT, PIPE, Popen
  import os, sys

  # Get the full path of iMODFIT process
  command = self.plugin_path + self.imodfit
  # Set the workspace
  self.cwd = self.plugin_path + self.plugin_folder

  # PDB selected in the menu
  pdbSelected = self.object_menu.getvalue()
  # Map selected in the menu
  mapSelected = self.map_menu.volume()

  # Get options values
  self.get_options_chimera()

  # Retrieve the full command to perform the fitting: imodfit + arguments
  cmd = [command, pdbSelected.openedAs[0], mapSelected.openedAs[0], self.resolution.get(),
self.cutoff.get(),
      self.imodfit_chimera_m, self.imodfit_chimera_m_val,
      self.imodfit_chimera_t,
      self.imodfit_chimera_r, self.imodfit_chimera_r_val,
      self.imodfit_chimera_re, self.imodfit_chimera_re_val,
      self.imodfit_chimera_morepdbs,
      self.imodfit_chimera_n, self.imodfit_chimera_n_val] + self.imodfit_chimera_adv_commands

# Execute the command with the respective arguments creating pipes between the process and
Chimera
# Pipes will be associated to the standard output and standard error required to show the process log
# in the window
imodfit_process = Popen(cmd, stdout=PIPE, stderr=PIPE, cwd=self.cwd, universal_newlines=True)
```

```python
# Text widget for process log that will showthe standard output of the process
from Tkinter import *
root = Tk()
root.wm_title("iMODFIT Process Log")
S = Scrollbar(root)
T = Text(root, height=30, width=85)
S.pack(side=RIGHT, fill=Y)
T.pack(side=LEFT, fill=Y)
S.config(command=T.yview)
T.config(yscrollcommand=S.set)

# Read first line
line = imodfit_process.stdout.readline()
# Variables to check the process status and show its output in a friendly format to the user
iter = False
model_iter = False
index_before_last_print = None
first_ite_sec = True

# Continue reading the standard output until iMODFIT is finished
# If the current line is an iteration for a model, replace in the widget the last showed
# If it is a new model or is part of the iMODFIT process, inserts the line at the end of the widget
while line:
    if len(line.strip()) == 0:
        line = imodfit_process.stdout.readline()
        continue
    if iter is False or (iter is True and 'NMA' in line):
        T.insert(END, line)
        model_iter = True
    elif iter is True and 'sec' in line:
        if first_ite_sec is True:
            T.insert(END, line)
            model_iter = True
        else:
            T.delete(index_before_last_print + "-1c linestart", index_before_last_print)
            T.insert(END, line)
            first_ite_sec = False
    elif model_iter is True:
        index_before_last_print = T.index(END)
        T.insert(END, line)
        model_iter = False
    elif iter is True and 'sec' not in line:
        T.delete(index_before_last_print + "-1c linestart", index_before_last_print)
        T.insert(END, line)
    T.update()
    T.yview(END)
    line = imodfit_process.stdout.readline()
    if ('NMA_time' in line and 'Score' in line):
        iter = True
    if 'sec' in line and iter is True:
        first_ite_sec = True
        model_iter = True
```

```python
    if index_before_last_print is not None:
      T.delete(index_before_last_print + "-1c linestart", index_before_last_print)
    index_before_last_print = T.index(END)
  if 'Convergence' in line:
    iter = False


T.insert(END, "\n\n --> iMODFIT Process has finished. Check 'Results' button to visualize solution.
<--\n")
T.update()
T.yview(END)


# When iMODFIT process is finished, the results are set into the Results panel...
self.fill_results()
# and the plugin buttons are enabled again
self.enable_process_buttons()
```

```python
# ------------------------------------------------------------------------
# Reads the coordinates of the fitted molecule and updates the original
# molecule position to show the fitting made by iMODFIT
#
def show_fitted_molecule(self, fitted):

 # Get opened molecule (the one selected in the menu)
 p = self.object_menu.getvalue()
 # Get some atom
 a0 = p.atoms[0]
 # Coordinates array
 cs = []
 # Names array
 ns = []

 # Depends on the button state, chose the original molecule
 # or the fitted one to update the position
 fitted_mol = None
 if fitted is True:
   fitted_mol = self.cwd + self.fitted_molecule
 else:
   fitted_mol = self.object_menu.getvalue().openedAs[0]

 # Read thhe coordinates of the fitted molecule and fill
 # coordinates and names arrays
 with open(fitted_mol) as pdbfile:
  for line in pdbfile:
   if line[:4] == 'ATOM' or line[:6] == "HETATM":
    # Split the line according to PDB format
    n = line[12:16]
    # Construct a copy of coordinates
    c = a0.coord()
    c[0] = x = float(line[30:38])
    c[1] = y = float(line[38:46])
    c[2] = z = float(line[46:54])
    # Store all atomic coordinates
    cs.append(c)
    # store names (for self-consistency checking)
    ns.append(n.strip())
```

```python
# Update coordinates of the molecule
# coordinate array index
i = 0
for a in p.atoms:
  # Check if both atoms are the same
  # if atom names don't match
  if a.name != ns[i]:
    print 'Warning! ', ns[i], ' and ', a.name, ' mismatch at residue ', a.residue
    print 'Searching ', ns[i], ' in residue ', a.residue
    isfound = False
    for b in a.residue.atoms:
      # Atom found
      if b.name == ns[i]:
        print 'Atom (', ns[i], ') found. Updating (', b.name, ') coordinates'
        # Set new coordinates into "b" atom
        b.setCoord(cs[i])
        isfound = True
        break
    if not isfound:
      print 'Error! Atom ', ns[i], ' not found in residue ', a.residue
  # if atom names match
  else:
    # Set new coordinates into "a" atom
    a.setCoord(cs[i])

  # Update index
  i = i + 1
```

```python
# -------------------------------------------------------------------------
# Opens the MD movie created by iMODTFIT with the model trajectories
#
def open_movie(self):

  # Import the native dialog MovieDialog from Chimera
  from Movie.gui import MovieDialog
  # import the loadEnsemble native functionality from Chimera to load the movie
  from Trajectory.formats.Pdb import loadEnsemble

  # Load the movie created by iMODFIT
  movie = self.plugin_path + self.imovie
  loadEnsemble(("single", movie), None, None, MovieDialog)
```

The full code can be found at:

- *ADP_EM* → https://github.com/pablosolar/adp_em

- *iMODFIT* → https://github.com/pablosolar/imodfit