*n*MOLDYN User's Guide
Version 3.4

Eric Pellegrini[a], Paolo Calligari[b], Vania Calandrini[c]

Konrad Hinsen[c,d], Gerald R. Kneller[c,d,†]

October 6, 2009

Description

The *n*MOLDYN User's Guide describes how to use the various features of the **M**olecular **D**ynamics (MD) analysis program *n*MOLDYN. This guide includes a description of the capabilities of the program, how to use these capabilities, the necessary input files and formats, and a comprehensive list of the analysis available in the program and how to set up an run these analysis.

[a]Institut Laue-Langevin 6, rue Jules Horowitz BP 156 - 38042 Grenoble Cedex 9, France
[b]Ecole Normale Supérieure 24, rue Lhomond - 75231 Paris CEDEX 5, France
[c]Centre de Biophysique Moleculaire, CNRS Rue Charles Sadron, 45071 Orleans, Cedex 02, France
[d]Synchrotron Soleil - Division Experiences, Saint Aubin - BP 48 91192 Gif sur Yvette Cedex, France
[†]Corresponding author. Electronic mail:`kneller@cnrs-orleans.fr`

Abstract

*n*MOLDYN is a modular program package for the analysis of **M**olecular **D**ynamics (MD) trajectories, especially designed for the computation and decomposition of neutron scattering spectra.

The current release 3.0 of *n*MOLDYN is an upgrade of the version *n*MOLDYN 2.0 [1], which extends the functionality of the original version *n*MOLDYN 1.0 [2]. It provides an improved user interface (both graphical/interactive and batch), and can be used as a tool set for implementing new analysis modules.

*n*MOLDYN allows one to calculate several dynamics quantities such as the mean-square displacement, the velocity autocorrelation function as well as its Fourier Transform (the density of states) and its memory functions, the angular velocity autocorrelation function and its Fourier transform, the reorientational correlation function ... Moreover it can compute several quantities related to neutron scattering such as the coherent and incoherent intermediate scattering functions with their Fourier transforms and their memory functions, the elastic incoherent structure factor, the static coherent structure factor or the radial distribution function. Additionally, the *n*MOLDYN package allows one to construct modified trajectories from an input trajectory; rigid-body trajectories, in which the internal motions of the molecules (or parts thereof) are eliminated, frequency-filtered trajectories, from which motions outside a specified frequency interval are eliminated, global motion filtered trajectory where the global rotation and translation are removed from the input trajectory ...

All *n*MOLDYN calculations can be applied to a whole system or to arbitrary subsets. The most common subsets can be selected in the graphical interface, less common selections can be specified by Python code via the command-line interface.

As *n*MOLDYN 2.0, *n*MOLDYN 3.0 uses **M**olecular **M**odelling **T**ool**K**it (MMTK)[3, 4], an Open Source program library for molecular simulation applications, and expects trajectories to be in MMTK format (**N**etwork **C**ommon **D**ata **F**orm (NetCDF) [5, 6]). However, many trajectory converters are included in the distribution of *n*MOLDYN. *n*MOLDYN 3.0, was written by Eric Pellegrini.

Contents

List of Figures

List of Tables

List of Acronyms

Angular Correlation (*AC*) , 67–69

Angular Density Of States (*ADOS*) , 105–108

Auto-Regressive (*AR*) , 86, 88, 90, 116, 126

Auto-Regressive Analysis (*ARA*) , 86, 91–93, 173

Angular Velocity AutoCorrelation Function (*AVACF*) , 102, 104, 105, 173

network Common Data Language (CDL) , 32, 34, 61, 64, 66, 68, 72, 75, 93, 101, 104, 107, 115, 120, 125, 130, 134, 139, 143, 146, 150, 153, 158, 159, 163, 165

Chemistry at HARvard Macromolecular Mechanics (CHARMM) , 10, 22, 23, 29

Coordination Number (*CN*) , 10, 150, 151, 153–155

Center Of Mass Trajectory (*COMT*) , 10, 85, 86

Dynamic Coherent Structure Factor (*DCSF*) , 112, 115, 173

Dynamic Coherent Structure Factor using an Auto-Regressive model (*DCSFAR*) , 117, 120

Dynamic Incoherent Structure Factor (*DISF*) , 122, 125, 139, 173

Dynamic Incoherent Structure Factor using an Auto-Regressive model (*DISFAR*) , 127, 130

Dynamic Incoherent Structure Factor using an Gaussian approximation (*DISFG*) , 132, 134, 173

Density Of States (*DOS*) , 70, 73, 75, 76, 90, 93, 173

Elastic Incoherent Structure Factor (*EISF*) , 134–136, 139, 173

Fast Correlation Algorithm (FCA) , 59, 111, 122, 184

Global Motion Filtered Trajectory (*GMFT*) , 11, 78–80

Graphical User Interface (*GUI*) , 10, 11, 16–18, 56, 175, 176, 179

Molecular Dynamics (MD) , 1, 9, 42, 58, 78, 80, 86, 94, 95, 97, 111, 121, 135, 160, 184

1. Introduction

Although **M**olecular **D**ynamics (MD) simulation techniques are widely used in physics, chemistry and biology, their possibilities are often not fully exploited because of the lack of easy-to-use analysis tools. This is especially true for very complex systems which force most computational scientists to use standard program packages containing only very limited functionality to analyze MD trajectories.

In this NOTE we present the new release 3.0 of the modular program package *n*MOLDYN, which replaces the original version *n*MOLDYN 2.0, for the analysis of MD trajectories.

The program *n*MOLDYN was developed mainly for use in connection with neutron scattering experiment, although many of the quantities are also used in other contexts. The combination of neutron scattering experiments and MD simulations is a powerful tool to study the structure and dynamics of complex molecular systems. Neutron scattering is sensitive to time and space correlations of atomic positions on the *ns* time scale and the Å length scale [7, 8]. These are exactly the time and space domains covered by classical MD simulations. On the length scale under consideration the neutron-target interaction can be modelled by pseudopotentials with zero range which are centered on the atomic nuclei of the targets. The coupling between neutron and target is described by so-called scattering lengths describing the strength of the neutron-nucleus interaction [7]. The differential scattering cross section can be expressed in terms of quantum time correlation functions of the spatially Fourier transformed particle density. The corresponding *classical* time correlation function can be easily obtained from MD simulations. This enables a direct comparison between simulated and measured neutron scattering intensities for classical systems if recoil effects in the scattering process are not dominant [9]. The experimental data can be used to test the quality of the MD force field which is the central input for the simulations [10, 11, 12, 13]. Conversely, the simulated intensities allow a detailed analysis of the dynamical and structural behaviour of the system under consideration [14, 15]. The latter is particularly important for complex systems for which an interpretation of the measured intensities in terms of simple analytical models is difficult, if not impossible.

The program package *n*MOLDYN allows neutron scattering intensities to be efficiently calculated from MD simulations. The calculation of various space and time correlation functions permits a detailed analysis of the structure and dynamics of the system under consideration. *n*MOLDYN contains modules for the calculation of dynamics-related, scattering-related and structure-related properties. In addition rigid body trajectories of subunits of the system can be extracted from molecular dynamics trajectory files. These subunits can be arbitrarily defined, their size can range from a few atoms to a whole domain in a macromolecule. From the rigid body trajectories angular correlation functions and reorientational correlation functions can be obtained.

The third generation *n*MOLDYN presented here, offers an interactive graphical user interface for standard calculations, highly flexible script-based processing for non-standard applications and a machine-independent compact binary file format. These improvements were made possible by the use of

1. **python**, a high-level object oriented langage [16, 17];
2. **NumPy**, a python package needed for scientific computing [18];
3. **NetCDF**, portable binary file format and its corresponding library [5, 6];

4. **Scientific Python**, an additional scientific computing library [20, 21];

5. **MMTK**, a molecular simulation library [3, 4].

All of these packages are developped and distributed following the Open Source principles [22]; anyone can use and improve them without being hindered by licensing restricitons. All the time-consuming algorithms use efficient implementations in C or in Pyrex, a language that allows to write code that mixes Python and C data types, and compiles it into a C extension for Python [23].

This NOTE is organized as follows:

Section 1 gives an overview of *n*MOLDYN.
Section 2 gives the instruction to install *n*MOLDYN properly in an existing python distribution.
Section 3 describes the different *n*MOLDYN file formats.
Section 4 describes how to set up and run an analysis in *n*MOLDYN from the **G**raphical **U**ser **I**nterface (*GUI*).
Section 5 describes how to set up and run an analysis in *n*MOLDYN from the command-line interface.

1.1 New features in version 3.0

- Installers for MacOS X, Win32 and linux
- A new Graphical interface A brand new graphical interface is implemented with Tk library [24].
- A new plotting engine A brand bew plotting engine is implemented using matplotlib python library [25]. This allows to produce high quality plots directly from *n*MOLDYN.
- Trajectory converters Some trajectory converters are implemented and made directly accessible from *n*MOLDYN *GUI*. Those converters allows the conversion of trajectory coming from Amber9 [26], **C**hemistry at **HAR**vard **M**acromolecular **M**echanics (CHARMM) [27], DL_POLY [28], MaterialsStudio [29], **NA**noscale **M**olecular **D**ynamics (NAMD) [32], **V**ienna **A**b-initio **S**imulation **P**ackage (VASP) [34] and X-PLOR [35] to *n*MOLDYN **M**olecular **M**odelling **T**ool**K**it (MMTK) trajectory input formats.
- Converter from NetCDF to ASCII A converter from **N**etwork **C**ommon **D**ata **F**orm (NetCDF) to ASCII that wraps ncdump [36] program is implemented.
- Converter from ASCII to NetCDF A converter from ASCII to NetCDF that wraps ncgen [37] program is implemented.
- Pair Distribution Function The **P**air-**D**istribution **F**unction (*PDF*) analysis is implemented using Pyrex code for a faster analysis.
- Coordination number The **C**oordination **N**umber (*CN*) analysis is implemented using Pyrex code for a faster analysis.
- Spatial Density The **S**patial **D**ensity (*SD*) analysis is implemented using Pyrex code for a faster analysis.
- Radius Of Gyration The **R**adius **O**f **G**yration (*ROG*) analysis is implemented.
- Root Mean Square Displacement The **R**oot **M**ean-**S**quare **D**eviation (*RMSD*) analysis is implemented.

- Center Of Mass Trajectory The **C**enter **O**f **M**ass **T**rajectory (*COMT*) analysis is implemented. It determines the trajectory of the center of mass of one or several group of atoms.
- Global Motion Filter The **G**lobal **M**otion **F**iltered **T**rajectory (*GMFT*) analysis is implemented. It removes the global translation and rotation degrees of freedom from a trajectory.
- Quasi Harmonic Analysis The **Q**uasi-**H**armonic **A**nalysis (*QHA*) analysis is implemented. It allows for the decomposition of a system into its different modes of vibration.
- Static Coherent Structure Factor The **S**tatic **C**oherent **S**tructure **F**actor (*SCSF*) analysis is implemented. It allows for the direct computation of the static coherent structure factor.
- Smoothed Static Coherent Structure Factor The **S**moothed **S**tatic **C**oherent **S**tructure **F**actor (*SSCSF*) analysis is implemented. It allows for the direct computation of the static coherent structure factor withtout any q-vectorsrandom generation. It is implemented using Pyrex code for a faster analysis.
- Order parameter The **O**rder **P**arameter (*OP*) analysis is implemented.
- Order parameter using the contact model The **O**rder **P**arameter using **C**ontact **M**odel (*OPCM*) analysis developped by Zhang *et al.* [73] is implemented.
- ScrewFit analysis The ScrewFit 2D methode developped by Paolo Calligari and Gerald Kneller [38] is implemented.
- Partial terms The partials term for all scattering-related properties are now available.
- A new atom selection engine A brand new and much more powerful atom selection engine is implemented. When setting up an anlysis, it allows for the selection from the *GUI* of almost any kind of subset.

1.2 User feedback

If you have problems installing or running *n*MOLDYN after reading this document, please send a complete description of the problem by email to pellegrini@ill.fr. If you discover and fix a problem not described in this manual we would appreciate if you would tell us about this as well, so we can alert other users and incorporate the fix into the public distribution. Your suggestions are welcome at pellegrini@ill.fr.

2. Installing *n*MOLDYN

When downloading *n*MOLDYN 3 from its website [39, 40], you will get the full Python source code which is covered by the CeCILL License [41]. The current stable version of *n*MOLDYN is 3.0.4. One of the major goal of the new *n*MOLDYN version targeted an easier installation on the most current platforms respectively Win32 and Unix (Linux and MacOS). This has been done with the development of plat-form specific installers.

*n*MOLDYN needs the following modules/libraries in order to work properly:

1. Tcl/Tk version $\geq$ 8.0 [78],
2. Python version $\geq$ 2.4 [17],
3. Numpy version $\geq$ 1.2 [18],
4. matplotlib version $\geq$ 0.98 [25],
5. Pyro version $\geq$ 3.9 [19],
6. Scientific Python version $\geq$ 2.8 [20, 21],
7. **M**olecular **M**odelling **T**ool**K**it (MMTK) version $\geq$ 2.6.0 [3, 4],
8. pywin32 version $\geq$ 210 (for a win32 installation only) [42].

The version number is important (and even compulsory for python) and in case where you would decide to use older versions of those packages, we can not guarantee that *n*MOLDYN will work properly or even work at all.

2.1 Unix users

The tar/zip source files of *n*MOLDYN can be downloaded from the site:

http://sourcesup.cru.fr/projects/nmoldyn.

The instructions to install *n*MOLDYN from this file are:

1. `tar xzf nMOLDYN-3.x.y.tar.gz (or unzip nMOLDYN-3.x.y.zip)`
2. `cd nMOLDYN-3.x.y`
3. `python setup.py build`
4. `python setup.py install`

The last command may require administrator privileges.

2.2 Windows users

Installing *n*MOLDYN on Windows is straightforward. The *n*MOLDYN windows installer can be downloaded from the site:

http://sourcesup.cru.fr/projects/nmoldyn.

Once downloaded, to launch the *n*MOLDYN windows installer, just double-click on it. A dialog will open asking for a confirmation about the place where to install *n*MOLDYN. By default, this should be the path for your Python installation and that value should not be changed unless you have several Python installation on your machine (not recommended).

Currently, the only *n*MOLDYN Windows installer available was build with Python 2.5. If you want to build a version of *n*MOLDYN compatible with python 2.4, the strategy to adopt is a little bit different. This time, you have to

1. download the archive *nMOLDYN-3.x.y.zip* or *nMOLDYN-3.x.y.tar.gz*;
2. uncompress it in the directory *dir* of your choice;
3. open a DOS shell;
4. go directory *dir*;
5. enter python.exe setup.py bdist_wininst, this will create a *dist* directory containing the *n*MOLDYN Windows installer, *nMOLDYN-3.x.y.win32-py2.4.exe*;
6. double-click on the newly created *nMOLDYN-3.x.y.win32-py2.4.exe* file in the *dist* directory. This will install *n*MOLDYN

This strategy should also work with Python 2.6.

3. Input and output files

Almost, if not all, functionnalities provided by *n*MOLDYN are based on **N**etwork **C**ommon **D**ata **F**orm (NetCDF) input file. However, in certain circumstances *n*MOLDYN can use or produce another type of files. We will start this section by explaining in details the NetCDF file format format introducing next the other file formats used by *n*MOLDYN.

3.1 NetCDF file format

NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. The project homepage is hosted by the Unidata program at the **U**niversity **C**orporation for **A**tmospheric **R**esearch (*UCAR*) [43]. They are also the chief source of NetCDF -based software, standards development, updates ... [44]. The format is an open standard.

The data format is self-describing. This means that there is a header which describes the layout of the rest of the file, in particular the data arrays, as well as arbitrary file metadata in the form of name/value attributes. The format is platform independent, with issues such as endianness being addressed in the software libraries. The data arrays are rectangular, not ragged, and stored in a simple and regular fashion that allows efficient subsetting.

*n*MOLDYN expects trajectories to be in NetCDF format and follow the conventions of **M**olecular **M**odelling **T**ool**K**it (MMTK). Trajectories that have not been produced with MMTK or MMTK-based programs must be converted to MMTK format before they can be analyzed with *n*MOLDYN. This conversion is necessary because no other common trajectory format permits efficient access both to conformations at a given time and to one-atom trajectories for all times. In addition to providing such an access, the NetCDF format has several advantages that make it particularly suitable for archiving trajectories:

- compact files (binary storage);
- machine-independent format;
- fully self-contained, complete information about the system is stored in the trajectory file.

The conversion of the trajectories from different formats to the MMTK format can be made directly via the *n*MOLDYN graphical user interface (see Section 4.1.2).

3.2 PDB file format

The **P**rotein **D**ata **B**ank (*PDB*) file format is used to store coordinate or velocity data. This is the standard format for coordinate data for many bioinformatic programs. A full description of this file format can be obtained from the *PDB* web site [45]. You can generate *PDB* files in *n*MOLDYN by extracting one or several frames from a MMTK trajectory file (see Section 4.1.3). *PDB* files are also used in *n*MOLDYNwhen performing subset, deuteration or group selection from a *n*MOLDYN selection file (see Sections 4.2.2.1, 4.2.2.2 and 4.2.2.3).

3.3 *n*MOLDYN preference file

This file will be used by *n*MOLDYN to setup the preferences (e.g. documenation style, log file path ...). It is based on the **ConfigParser** module of the Python Standard Library [46] and as such it must respect the corresponding format. Section 4.1.6 will explain how to build, load or save such a file from *n*MOLDYN.

3.4 *n*MOLDYN selection files

These files allows to perform some subset selection from a *PDB* file when running an analysis. Their format will be explained in Sections 4.2.2.1, 4.2.2.2 and 4.2.2.3.

3.5 *n*MOLDYN autostart files

These files are python scripts that allows to run a given analysis just as a classical python script. See Section 5.1 for explanations about the format of these files.

3.6 *n*MOLDYN input files

These files are python scripts that allows to run a given analysis from the *n*MOLDYN command-line interface. For those familiar with *n*MOLDYN 2, these files correspond to the input scripts that was generated by **xMOLDYN** and that had to be run with **pMOLDYN**. See Section 5.2 for explanations about the format of these files.

4. Using *n*MOLDYNfrom the Graphical User Interface

Through the *n*MOLDYN graphical user interface, you will usually open a trajectory, then specify the parameters for the analysis you wish to perform and finally start the calculation itself. But you can also perform some other actions such as plotting the results of an analysis, performing some file conversions ... The graphical interface gives access to most of the functionalities of *n*MOLDYN. Moreover, from the graphical user interface it is possible to create an input file for the command-line interface or an autostart analysis python script. Both kind of files provide a convenient starting point to set up and run new analysis direclty from the command-line interface (see Section 5).
To run *n*MOLDYN from the *GUI*, type the following command line:

python-dir`/bin/nMOLDYNStart.py` on unix

or

python-dir`\Scripts\nMOLDYNStart.py` on Windows

python-dir being the prefix of your python distribution. The command line accepts some arguments that can change how *n*MOLDYN will start. The following arguments are currently supported by *n*MOLDYN:

- `-h/--help`
  Format: not an editable argument
  Value: *None*
  Description: displays the details of the command line arguments. **Will not start *n*MOLDYN in *GUI* mode.**

- `--version`
  Format: not an editable argument
  Value: *None*
  Description: displays the version of *n*MOLDYN. **Will not start *n*MOLDYN in *GUI* mode.**

- `-n/--netcdf=`
  Format: string
  Value: *filename*
  Description: starts *n*MOLDYN from the graphical user interface loading directly the *filename* NetCDF file.

- `-i/--input=`
  Format: string
  Value: *filename*
  Description: runs an analysis with the command-line interface using *filename* *n*MOLDYN input file (see Section 5.2 for details). **Will not start *n*MOLDYN in *GUI* mode.**

- `-c/--contents=`
  Format: string
  Value: *filename molname selkwd*
  Description: if *file* provided, displays the contents of the trajectory file *file*. If *file* and *molname* provided displays the selection keywords associated to *molname*. If *file* and *molname* and *selkwd* provided displays the selection values associated to selection keyword *selkwd*. **Will not start *n*MOLDYN in *GUI* mode.**

- `-t/--test=`
  Format: string
  Value: *testname*
  Description: runs one or several analysis benchmarks **Will not start the *GUI*.** The format for *testname* string is

  test1,test2,...

  where *test1*, *test2* ... are the name of benchmark 1, 2 ... (see Section 4.4.4 for a comprehensive list of the available benchmarks).

- `--acroread_path=`
  Format: string
  Value: *filename*
  Description: sets the path for Acrobat Reader executable to *filename*. It will overide the preferences settings.

- `--vmd_path=`
  Format: string
  Value: *filename*
  Description: sets the path for VMD executable to *filename*. It will overide the preferences settings.

- `--documentation_style=`
  Format: string
  Value: *html* or *pdf*
  Description: sets the format for the online documentation either to HTML (*html*) either to pdf (*pdf*). It will overide the preferences settings.

- `--ncdump_path=`
  Format: string
  Value: *filename*
  Description: sets the path for **ncdump** executable to *filename*. It will overide the preferences settings.

- `--ncgen_path=`
  Format: string
  Value: *filename*
  Description: sets the path for **ncgen** executable to *filename*. It will overide the preferences settings.

- `--outputfile_path=`
  Format: string
  Value: *dirname*

Description: sets the directory where all the output files will be written to *dirname*. It will overide the preferences settings.

- `--trajfile_path=`
  Format: string
  Value: *dirname*
  Description: sets the default MMTK trajectory input files directory search path to *dirname*. It will overide the preferences settings.

- `--warning_acroread=`
  Format: string
  Value: *yes* or *no*
  Description: if set to *yes*, will warn you if acrobat reader is not found on your system. It will overide the preferences settings.

- `--warning_ncdump=`
  Format: string
  Value: *yes* or *no*
  Description: if set to *yes*, will warn you if **ncdump** program is not found on your system. It will overide the preferences settings.

- `--warning_ncgen=`
  Format: string
  Value: *yes* or *no*
  Description: if set to *yes*, will warn you if **ncgen** program is not found on your system. It will overide the preferences settings.

- `--warning_vmd=`
  Format: string
  Value: *yes* or *no*
  Description: if set to *yes*, will warn you if **VMD** program is not found on your system. It will overide the preferences settings.

- `--progress_rate=`
  Format: integer in [0,100]
  Value: *prog_rate*
  Description: displays the progress of the analysis every *prog_rate* percents. It will overide the preferences settings.

When starting *n*MOLDYN from the *GUI*, the start-up window shown in figure 4.1 will pop up. This window contains four drop down menu buttons,

- **File**
- **Analysis**
- **View**
- **Help**

and a text window.

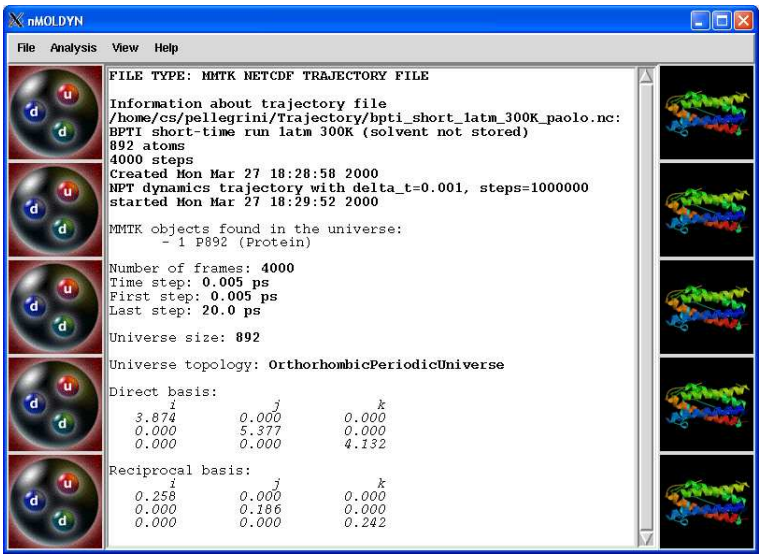

Figure 4.1: The *n*MOLDYN main window.

Everything concerning IO manipulations are done via the drop down menu button **File**. The drop down menu buttons **Analysis** allows to set up and start the analysis you are interested in. The **View** and **Help** buttons allow one to inspect input/output data and ask for help respectively. All of these menus will be described below.Finally, the text window will display all the information concerning the loaded file and the main actions performed on it.

4.1 The File menu

Pressing the **File** menubutton brings up a menu from which it is possible to choose the following options:

- Load NetCDF file
- Trajectory conversion
- Frame snapshot
- Convert NetCDF to ASCII
- Convert ASCII to NetCDF
- Preferences
- Quit

that will be described in the forthcoming sections.

4.1.1 Load NetCDF file

The **Load NetCDF file** option allows one to select the **NetCDF** file of the trajectory which will be used to perform the analysis. Clicking on it, a file browser like the one showed in figure 4.2 pops up.

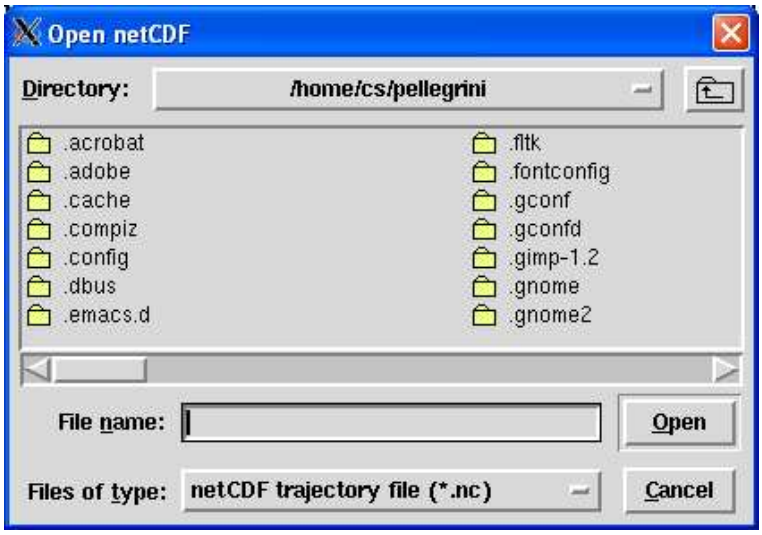

Figure 4.2: File-Directory selection window.

To select a directory, double click on it. By doing this, the complete pathname of the selected directory appears on the **Directory** selection button at the top of the window. To select the parent directory left-click the directory selection button. When a file in the list is highlighted (clicked) its name is shown in the **File name** field. The selection is confirmed by pressing the button **Open** of the browser. By doing this, the file selection window closes.

Two kind of files can be loaded in *n*MOLDYN, the MMTK NetCDF files and the MMTK trajectory set files (usually with respective **.nc** and **.ncs** extensions), the latter being just an ASCII file where each line is a link to a MMTK NetCDF file. The figure 4.3 shows an example of a trajectory set file.

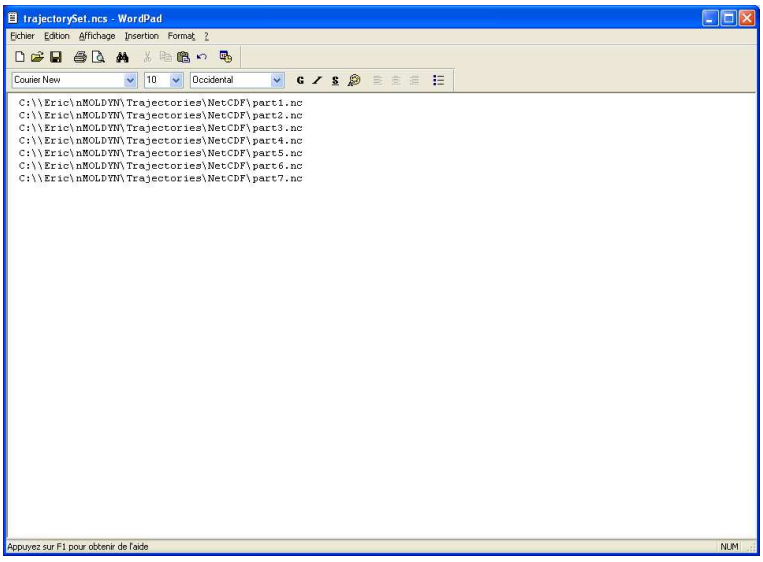

Figure 4.3: Example of a MMTK trajectory set file.

For more details about trajectory set files, please refer to the MMTK documentation [47]. Please note again that the MMTK NetCDF file format is central to *n*MOLDYN as it is the

only trajectory format that *n*MOLDYN can handle directly. Trajectories that have not been produced with MMTK or MMTK-based programs must be converted to MMTK format before they can be analyzed with *n*MOLDYN.

One important step that is performed by *n*MOLDYN when loading the trajectory is the determination of the chemical contents of the MMTK universe corresponding to the loaded trajectory. *n*MOLDYN attributes to each MMTK chemical object found in the universe a name that is either its MMTK database name if this object is referenced in the MMTK internal database otherwise its name will be its number of atoms appended to its chemical object type (respectively A, AC, M, NC, PC, P for Atom, AtomCluster, Molecule, NucleotideChain, PeptideChain, Protein chemical types.). In the seldom case where the system contains some isomers, *n*MOLDYN will distinguish them by appending to their name the suffix **_iso***i*' where *i* is the number of the isomer. In order to retrieve to which isomer corresponds the *n*MOLDYN name, a PDB file whose name is the name of the isomer is created in the output file directory.

4.1.2 Trajectory conversion

The **Trajectory conversion** option allows to convert a trajectory derived with a non MMTK-based program to the NetCDF MMTK trajectory format. Pressing the button **Trajectory conversion** brings up a menu from which it is possible to choose the following trajectory converters:

- Amber NetCDF to MMTK
- CHARMM/X-PLOR to MMTK
- DL_POLY to MMTK
- MaterialsStudio
- NAMD to MMTK
- VASP to MMTK

Pressing the **MaterialsStudio** menubutton brings up an additional menu from which it is possible to choose the following Materials Studio converters:

- Discover module to MMTK
- Forcite module to MMTK

4.1.2.1 Amber to MMTK

This converter allows the conversion from a NetCDF trajectory generated with Amber 9 or 10 [26] to a MMTK NetCDF trajectory (unfortunately, both NetCDF files do not follow the same convention). For version of Amber lower than 9, Amber provides some tools for the conversion to Amber NetCDF trajectories. Pressing the **Amber NetCDF to MMTK** menubutton, the dialog shown in figure 4.4 will pop up.

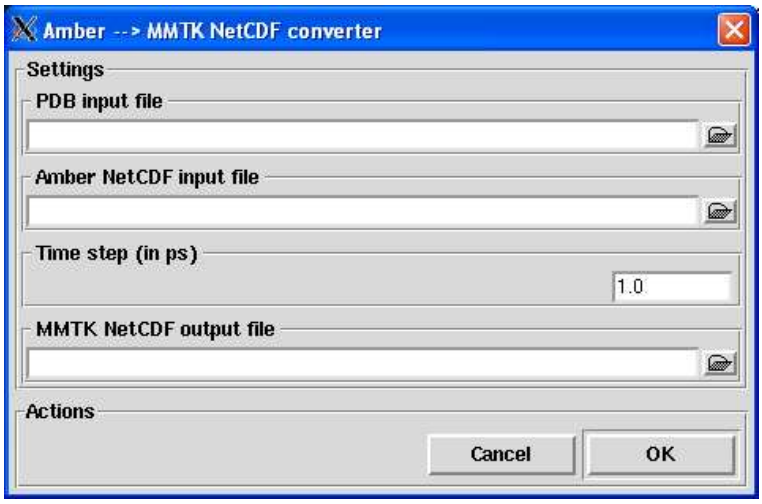

Figure 4.4: The Amber to MMTK converter dialog.

To perform the conversion, the following input fields must be filled:

- **PDB input file**
  Format: string
  Default: *None*
  Description: a PDB file of the system must be provided for the conversion. This file is necessary to build up the MMTK universe related to the MMTK trajectory.

- **Amber NetCDF input file**
  Format: string
  Default: *None*
  Description: the Amber 9 or 10 NetCDF trajectory file that contains all the trajectory frames.

- **Time step (in ps)**
  Format: strictly positive float
  Default: *1.0*
  Description: the time step in *ps* between two consecutive frames of the Amber NetCDF trajectory. You have to provide this information because it is not contained in the Amber NetCDF trajectory file.

- **MMTK NetCDF output file**
  Format: string
  Default: *None*
  Description: the name of the MMTK NetCDF trajectory that will be written. Once, an Amber NetCDF file has been loaded, a default name for the MMTK NetCDF output file will be proposed. This default name will be *file_mmtk.nc* if *file.nc* is the Amber NetCDF trajectory file name.

4.1.2.2 CHARMM/X-PLOR to MMTK

This converter allows the conversion from a trajectory generated with **C**hemistry at **HAR**vard **M**acromolecular **M**echanics (CHARMM) or X-PLOR [27, 35] to a MMTK NetCDF trajectory. Pressing the **CHARMM/X-PLOR to MMTK** menubutton, the dialog shown in figure 4.5 will pop up.

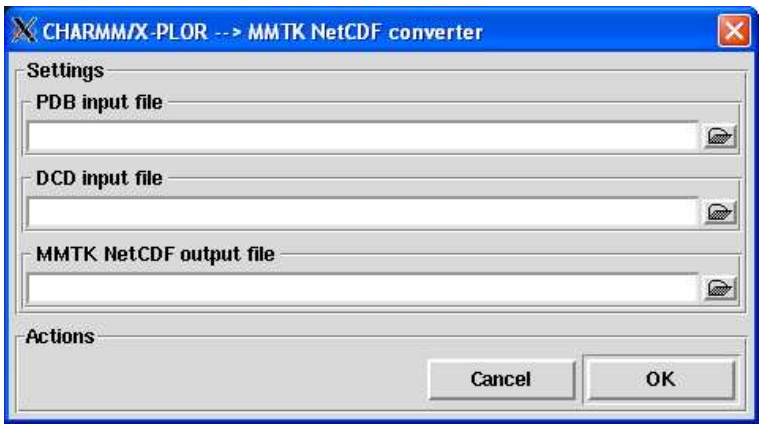

Figure 4.5: The CHARMM to MMTK converter dialog.

To perform the conversion, the following input fields must be filled:

- **PDB input file**
  Format: string
  Default: *None*
  Description: a PDB file of the system must be provided for the conversion. This file is necessary to build up the MMTK universe related to the MMTK trajectory.

- **DCD input file**
  Format: string
  Default: *None*
  Description: the CHARMM DCD trajectory file that stores the trajectory frames.

- **MMTK NetCDF output file**
  Format: string
  Default: *None*
  Description: the name of the MMTK NetCDF trajectory that will be written. Once, a DCD file has been loaded, a default name for the MMTK NetCDF output file will be proposed. This default name will be *file.nc* if *file.dcd* is the DCD trajectory file name.

4.1.2.3 DL_POLY to MMTK

This converter allows the conversion from a trajectory generated with DL_POLY [28] to a MMTK NetCDF trajectory. Pressing the **DL_POLY to MMTK** menubutton, the dialog shown in figure 4.6 will pop up.

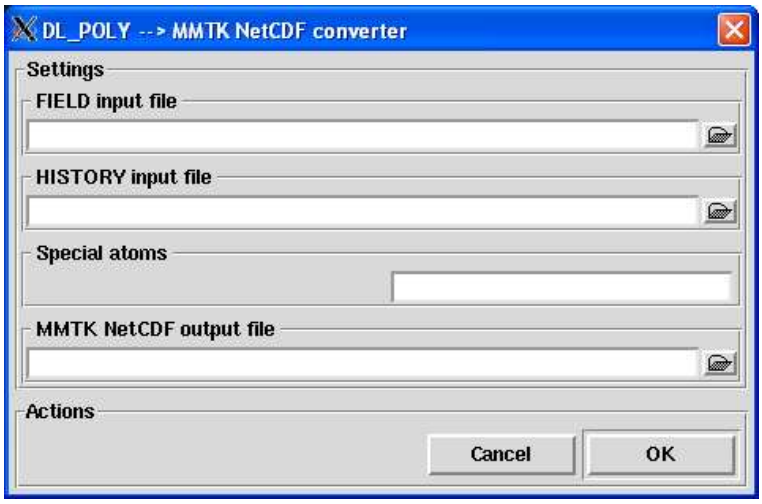

Figure 4.6: The DL_POLY to MMTK converter dialog.

To perform the conversion, the following input fields must be filled:

- **FIELD input file**
  Format: string
  Default: *None*
  Description: the DL_POLY FIELD file that stores the informations about the system. This file is necessary to build up the MMTK universe related to the MMTK trajectory.

- **HISTORY input file**
  Format: string
  Default: *None*
  Description: the DL_POLY HISTORY file that stores the trajectory frames.

- **Special atoms**
  Format: string
  Default: *None*
  Description: *n*MOLDYN will create the MMTK universe with the atom names specified in the FIELD file. By default, *n*MOLDYN will interpret these names directly as if they were a chemical symbol. If this fails, *n*MOLDYN will remove the last character until it corresponds to a known chemical symbol. For example, an atom defined in the FIELD file as CB, will first be interpreted as an atom of chemical symbol CB. As it does not exist, *n*MOLDYN will interpret it as an atom of chemical symbol C, namely a carbon atom. Usig htis procedure, it can happen that some atom names can be misunderstood or event not understood at all by MMTKÀs an example, the figure 4.7 shows the example of a FIELD file where one carbon atom specification (the one in red in the figure) will be misinterpreted as a cesium atom.

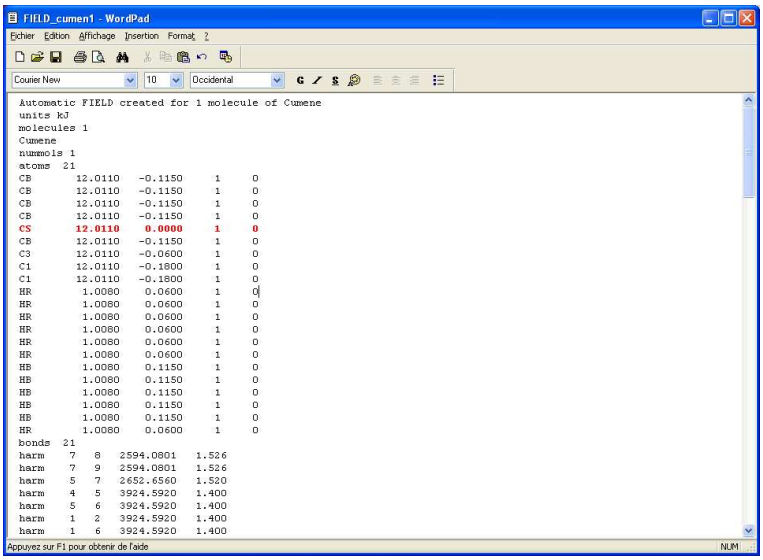

Figure 4.7: Example of a DL_POLY/FIELD file for which the **Special atoms** field must be filled because one carbon atom name, CS, will be interpreted as a cesium atom.

The aim of the **Special atoms** field is precisely to avoid such problems. The format for the **Special atoms** field is

atom name1:element1 sep atom name2:element2 ... where *sep* can be a white space, a comma or a semicolon.

In the example showed in figure 4.7, the string CS:C should be entered in the **Special atoms** field. Interestingly, the **Special atoms** field can also be used to specify united atoms. The syntax is exactly the same but, in that case, the element name must be replaced by the MMTK united atom code (e.g. CH3, CH2, CH, NH, NH2, NH3, OH, SH ...).

- **MMTK NetCDF output file**
  Format: string
  Default: *None*
  Description: the name of the MMTK NetCDF trajectory that will be written.

4.1.2.4 Discover to MMTK

This converter allows the conversion from a trajectory generated with MaterialsStudio Discover module [30] to a MMTK NetCDF trajectory. Pressing the **Discover to MMTK** menubutton, the dialog shown in figure 4.8 will pop up.

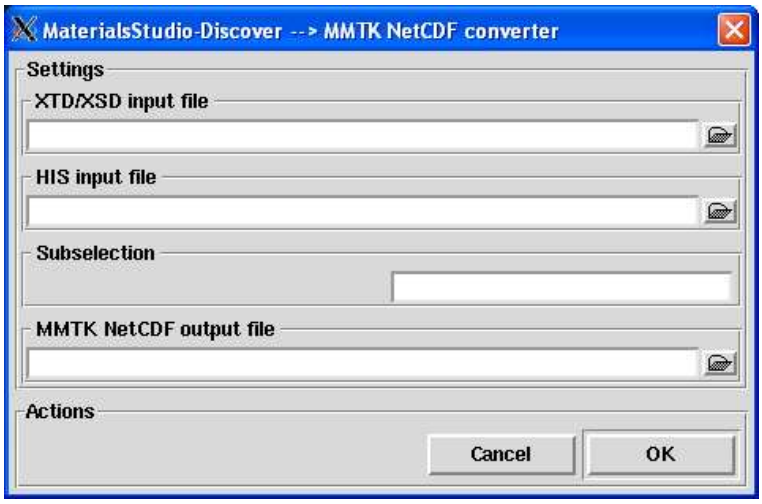

Figure 4.8: The Discover to MMTK converter dialog.

To perform the conversion, the following input fields must be filled:

- **XTD/XSD input file**
  Format: string
  Default: *None*
  Description: a XTD or XSD file of the system must be provided for the conversion. This file is necessary to build up the MMTK universe related to the MMTK trajectory. XTD or XSD files are automatically generated during a simulation performed with MaterialsStudio.

- **HIS input file**
  Format: string
  Default: *None*
  Description: the Discover HIS file file that stores the trajectory frames.

- **Subselection**:
  Format: integer or Python expression or string
  Default: *None*
  Description: it can happen that the molecular hierarchy is not correctly set in the XTD/XSD file or that you do not want to include all the atoms of the Discover trajectory into the MMTK trajectory. In that case, it is possible to specify how the system should be organized.

  The format for the **Subselection** field can be

  - ⋆ a single integer that specifies the index of the atom to select in the XSD/XTD file,
  - ⋆ a valid python expression that will generate nested lists of integers where each list will generate a distinct MMTK AtomCluster made of the atoms whose indexes in the XSD/XTD match the integers of the list. As an example, entering [[1,2,3,5],[10,20,21,23],[90,93]] will consider only atoms 1, 2, 3, 5, 10, 20, 21, 23, 90 and 93 of the XSD/XTD file when creating the MMTK trajectory. Moreover, those

atoms will be gathered such as atoms 1, 2, 3 and 5 will be in a fist AtomCluster, atoms 10, 20, 21 and 23 will be in a second AtomCluster and atoms 90 and 93 will be gathered in a third AtomCluster,

⋆ a string with the following format:
min_1:max_1:skip_1 sep ... sep min_N:max_N:skip_N
where *sep* can be a white space, a comma or a semicolon and each block *mini:maxi:skipi* will specify a distinct MMTK AtomCluster made of atoms whose indexes in the XSD/XTD file ranges from *min_i* to *max_i* by jumps of *skip_i* atoms. For example, entering 2:5:1;20:30:3 will generate a first AtomCluster made of atoms 2,3,4,5 and a second AtomCluster made of atoms 20,23,26 and 29.

- **MMTK NetCDF output file**
  Format: string
  Default: *None*
  Description: the name of the MMTK NetCDF trajectory that will be written. Once, a HIS has been loaded, a default name for the MMTK NetCDF output file will be proposed. This default name will be *file.nc* if *file.his* is the HIS trajectory file name.

4.1.2.5 Forcite to MMTK

This converter allows the conversion from a trajectory generated with Materials Studio Forcite module [31] to a MMTK NetCDF trajectory file. Pressing the **Forcite to MMTK** menubutton, the dialog shown in figure 4.9 will pop up.

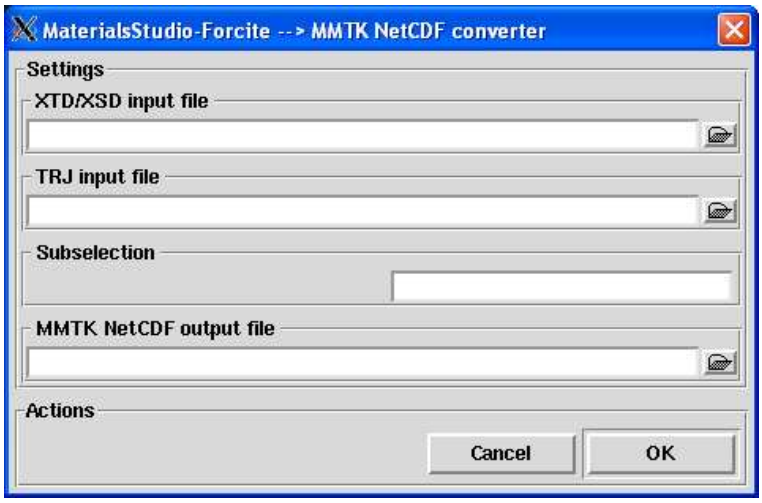

Figure 4.9: The Forcite to MMTK converter dialog.

To perform the conversion, the following input fields must be filled:

- **XTD/XSD input file**
  Format: string
  Default: *None*
  Description: a XTD or XSD file of the system must be provided for the conversion. This file is necessary to build up the MMTK universe related to the MMTK trajectory.

XTD or XSD files are automatically generated during a simulation performed with MaterialsStudio.

- **TRJ input file**
  Format: string
  Default: *None*
  Description: the Forcite TRJ file that stores the trajectory frames.

- **Subselection**:
  Format: integer or Python expression or string
  Default: *None*
  Description: it can happen that the molecular hierarchy is not correctly set in the XTD/XSD file or that you do not want to include all the atoms of the Discover trajectory into the MMTK trajectory. In that case, it is possible to specify how the system should be organized.

  The format for the **Subselection** field can be

  - ⋆ a single integer that specifies the index of the atom to select in the XSD/XTD file,
  - ⋆ a valid python expression that will generate nesteds list of integers where each list will generate a distinct MMTK AtomCluster made of the atoms whose indexes in the XSD/XTD match the integers of the list. As an example, entering [[1,2,3,5],[10,20,21,23],[90,93]] will consider only atoms 1, 2, 3, 5, 10, 20, 21, 23, 90 and 93 of the XSD/XTD file when creating the MMTK trajectory. Moreover, those atoms will be gathered such as atoms 1, 2, 3 and 5 will be in a fist AtomCluster, atoms 10, 20, 21 and 23 will be in a second AtomCluster and atoms 90 and 93 will be gathered in a third AtomCluster,
  - ⋆ a string with the following format:
    min_1:max_1:skip_1 sep ... sep min_n:max_n:skip_n
    where *sep* can be a white space, a comma or a semicolon and each block *min_i:max_i:skip_i* will specify a distinct MMTK AtomCluster made of atoms whose indexes in the XSD/XTD file ranges from *min_i* to *max_i* by jumps of *skip_i* atoms. For example, entering 2:5:1;20:30:3 will generate a first AtomCluster made of atoms 2,3,4,5 and a second AtomCluster made of atoms 20,23,26 and 29.

- **MMTK NetCDF output file**
  Format: string
  Default: *None*
  Description: the name of the MMTK NetCDF trajectory that will be written. Once, a TRJ file has been loaded, a default name for the MMTK NetCDF output file will be proposed. This default name will be *file.nc* if *file.trj* is the TRJ trajectory file name.

4.1.2.6 NAMD to MMTK

This converter allows the conversion from a trajectory generated with **NA**noscale **M**olecular **D**ynamics (NAMD) [32] to a MMTK NetCDF trajectory. Pressing the **NAMD to MMTK** menubutton, the dialog shown in figure 4.10 will pop up.

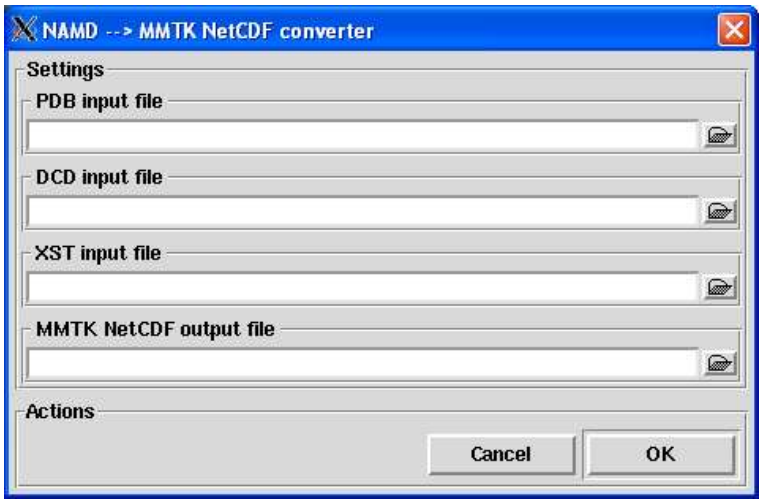

Figure 4.10: The NAMD to MMTK converter dialog.

To perform the conversion, the following input fields must be filled:

- **PDB input file**
  Format: string
  Default: *None*
  Description: a PDB file of the system must be provided for the conversion. This file is necessary to build up the MMTK universe related to the MMTK trajectory.

- **DCD input file**
  Format: string
  Default: *None*
  Description: the CHARMM DCD trajectory file that stores the trajectory frames.

- **XST input file**:
  Format: string
  Default: *None*
  Description: The NAMD e**X**tended **S**ystem **T**rajectory (XST) file has to be provided to the converter.

- **MMTK NetCDF output file**
  Format: string
  Default: *None*
  Description: the name of the MMTK NetCDF trajectory that will be written. Once, a DCD file has been loaded, a default name for the MMTK NetCDF output file will be proposed. This default name will be *file.nc* if *file.dcd* is the DCD trajectory file name.

4.1.2.7 VASP to MMTK

This converter allows the conversion from a trajectory generated with **V**ienna **A**b-initio **S**imulation **P**ackage (VASP) [34] to a MMTK NetCDF trajectory. Pressing the **VASP to MMTK** menubutton, the dialog shown in figure 4.11 will pop up.

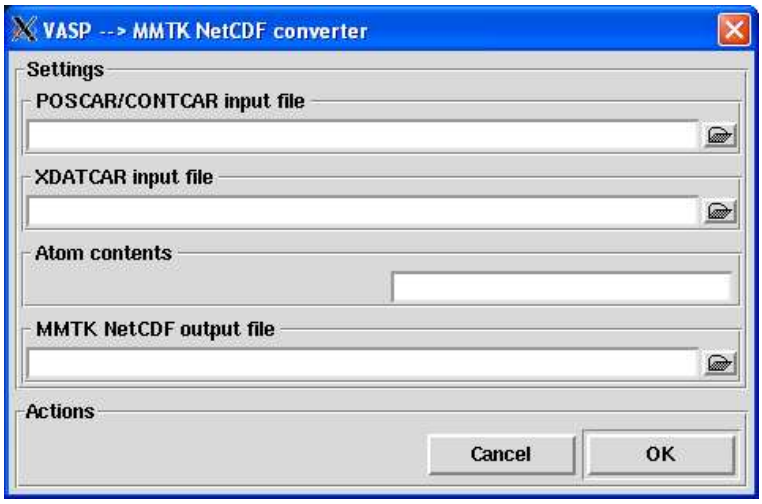

Figure 4.11: The VASP to MMTK converter dialog.

To perform the conversion, the following input fields must be filled:

- **CONTCAR/POSCAR input file**
  Format: string
  Default: *None*
  Description: the VASP CONTCAR or POSCAR file that stores the informations about the system. This file is necessary to build up the MMTK universe related to the MMTK trajectory.

- **XDATCAR input file**
  Format: string
  Default: *None*
  Description: the VASP XDATCAR file that stores the trajectory frames.

- **Atom contents**
  Format: string
  Default: *None*
  Description: the CONTCAR file contains the number of atoms of each element in the system but does not tell the elements the system is made of. This information has to provided using the **Atom contents** field.

  The format for the **Atom contents** field is

  element1 sep element2 ... where *sep* can be a white space, a comma or a semicolon and *element1, elemnt2 ...* are the chemical symbols of element 1, 2 ...

  Interestingly, the **Atom contents** field can also be used to specify united atoms. The syntax is exactly the same but, in that case, the element name must be replaced by the MMTK united atom code (e.g. CH3, CH2, CH, NH, NH2, NH3, OH, S ...).

- **MMTK NetCDF output file**
  Format: string
  Default: *None*
  Description: the name of the MMTK NetCDF trajectory that will be written.

4.1.3 Frame snapshot

The **Frame snapshot** option allows the extraction of one or several frames in *PDB* format from a MMTK NetCDF trajectory. Clicking on it, the dialog shown in figure 4.12 will pop up.

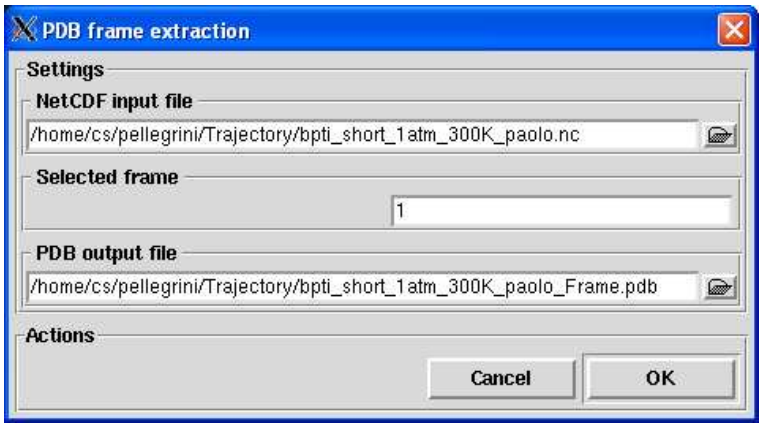

Figure 4.12: The dialog used to export trajectory frames to a PDB file.

To perform the frame extraction, the following input fields must be filled:

- **NetCDF input file**
  Format: string
  Default: *None*
  Description: a MMTK NetCDF trajectory file of the system must be provided for the extraction. If a trajectory is currently loaded, it will be proposed by default for the frame extraction.

- **Selected frames**
  Format: integer or Python expression or string
  Default: *1*
  Description: this field will store the frames selected for extraction. The format for the **Selected frames** field can be:

  ⋆ an integer specifying the index of a single frame to extract

  ⋆ a valid Python expression that will generate a list of integers where each integer specify the index of a frame to extract.

  ⋆ a string with the following format:
  min 1:max 1:skip 1 sep . . . sep min N:max N:skip N
  where *sep* can be a white space, a comma or a semicolon and each block *min i:max i:skip i* will specify a range of frames including frame *min i* to frame *max i* by jump of *skip i* frames.

- **PDB output file**
  Format: string
  Default: *None*
  Description: this field will store the name of the *PDB* output file that will contain the extracted frames. The line

  REMARK Frame i

  will be written before each written frame *i*.

4.1.4 Convert NetCDF to ASCII

The **Convert NetCDF to ASCII** option allows the conversion of any kind of NetCDF file to a network **C**ommon **D**ata **L**anguage (CDL) file [77]. Clicking on it, the dialog shown in figure 4.13 will pop up.

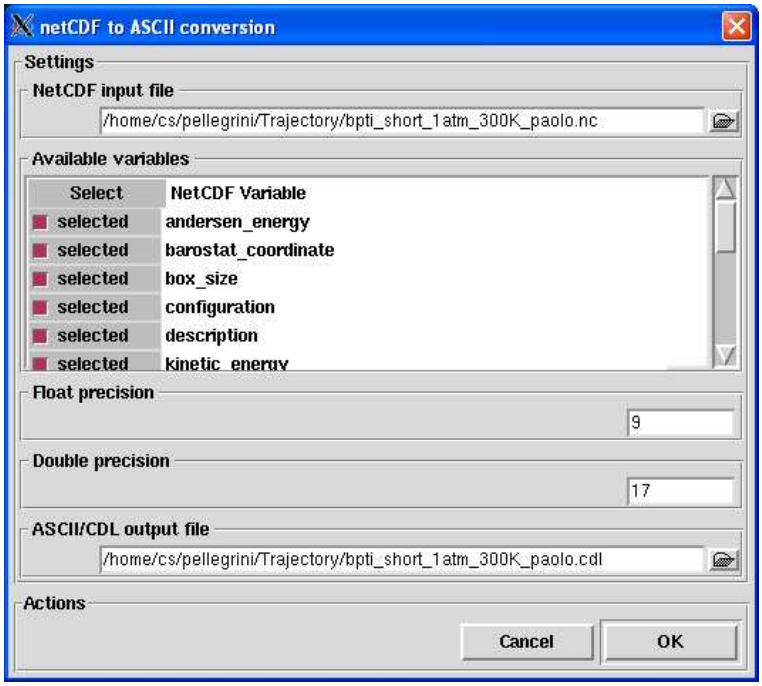

Figure 4.13: The dialog used to convert files in NetCDF format to CDL format.

To use this functionnality, the **ncdump** program [36] provided with the NetCDF library must be installed and the path to the **ncdump** executable must be defined in the *n*MOLDYN preferences. If **ncdump** is not installed, this functionnality will be disabled.

To perform the conversion, the following input fields must be filled:

- **NetCDF file**
  Format: string
  Default: *None*
  Description: a NetCDF file must be provided for the conversion. If a NetCDF file is currently loaded (a MMTK trajectory or other kind of NetCDF file), it will be proposed

by default for conversion. Once the NetCDF file is loaded all the variables found in the NetCDF file will be displayed in the **Available variables** field.

- **Available variables**
  Format: Not an editable field
  Default: *None*
  Description: this field displays all the variables contained in the NetCDF file. It contains two columns: the first one displays the checkbuttons that will allow to unselect/select which NetCDF variable (displayed on the right column) should be considered for conversion.

- **Float precision**
  Format: integer
  Default: *9*
  Description: this field stores the precision at which floating numbers will be written in the ASCII/CDL output file.

- **Double precision**
  Format: integer
  Default: *17*
  Description: this field stores the precision at which double numbers will be written in the ASCII/CDL output file.

- **ASCII/CDL output file**
  Format: string
  Default: *None*
  Description: this field stores the name of the ASCII/CDL output file.

4.1.5 Convert ASCII to NetCDF

The **Convert ASCII to NetCDF** option allows the conversion of from an ASCII file to a NetCDF file. Clicking on it, the dialog shown in figure 4.1.5 will pop up.

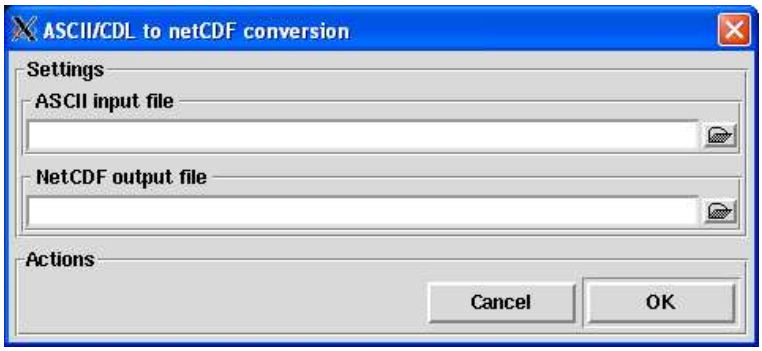

Figure 4.14: The dialog used to convert files in ASCII/CDL format to NetCDF format.

To use this functionnality, the **ncgen** program [37] provided with the NetCDF library must be installed and the path to the **ncgen** executable must be defined in the *n*MOLDYN preferences. If **ncgen** is not installed, this functionnality will be disabled.

To perform the conversion, the following input fields must be filled:

- **ASCII input file**
  Format: string
  Default: *None*
  Description: this field stores the name of the ASCII input file that will be used for the conversion. Two possible ASCII file formats are compatible for the conversion. The first one is the CDL file format [77] and the second one is a simpler but less general format where only numeric data can be converted. The latter format must contain white spaces separated columns where each column will be interpreted as a NetCDF unidimensional array of double. The file can contain header lines that must start with '#' character. The rest of the line will be stored in the NetCDF global attribute 'comment'. Some NetCDF global variables can also be specified for the conversion. They must be declared one after the other inside the header using the following format;

  # global name = value

  Finally, a name and some additional NetCDF variable attributes (units ...) can be given to the columns that will be converted. Column names must be declared one after the other inside the header using the following format:

  # variable name = value ; attribute1 = value1 ; attribute2 = value2 ...

  In such a case the number of variable declaration must be exactly the same than the number of columns. Otherwise the NetCDF variable name will be Column$i$ where i is the column index.

  The figure 4.15 shows an example of such file.

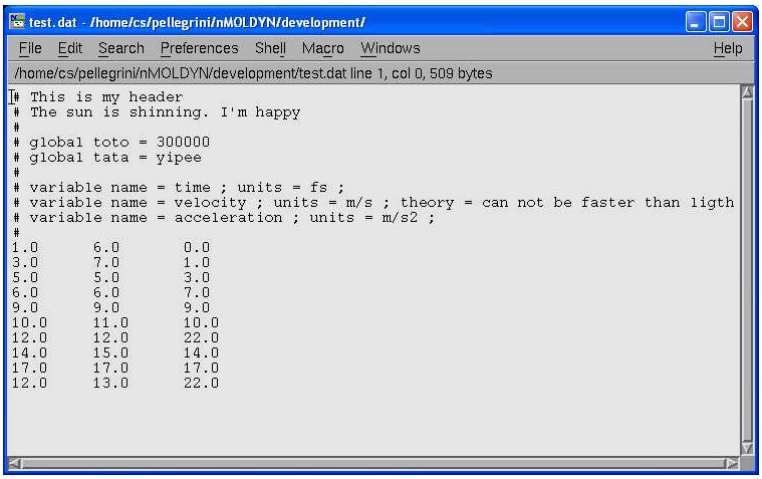

Figure 4.15: Example of an ASCII file that can be converted to a NetCDF file.

The output NetCDF file resulting from its conversion will contain:

- ⋆ a NetCDF dimension 'nvalues' whose value corresponds to the length of the columns to convert (in that case 10),
- ⋆ a global attribute 'comment' whose value is ' The sun is shinning. I'm happy',
- ⋆ a global attribute 'toto' whose value is '300000',
- ⋆ a global attribute 'tata' whose value is 'yipee',
- ⋆ a NetCDF variable 'time' of dimension 'nvalues' with an additional attribute 'units' whose value is 'fs',
- ⋆ a NetCDF variable 'velocity' of dimension 'nvalues' with additional attributes 'units' and 'theory' whose values are respectively 'm/s' and 'can not be faster than light',
- ⋆ a NetCDF variable 'acceleration' of dimension 'nvalues' with an additional attribute 'units' whose value is 'm/s2',

- **NetCDF output file**
  Format: string
  Default: *None*
  Description: this field stores the name of the NetCDF output file.

4.1.6 Preferences

The **Preferences** option allows to set the *n*MOLDYN preferences using the **ConfigParser** Python-module mechanism [46]. In *n*MOLDYN the preferences are classified in the three following sections:

- File handling: contains the preferences variables related to the file handled by *n*MOLDYN (log file, output file ...),
- External programs: contains the preferences variables related to the actions performed by *n*MOLDYN that require an external program (e.g. displaying the documentation, animating a trajectory, converting NetCDF to ASCII or ASCII to NetCDF ...),

- Miscellaneous: contains the other preferences variables that could not be classified elsse-where.

Pressing the **Preferences** menubutton will pop up the dialog shown in figure 4.16.

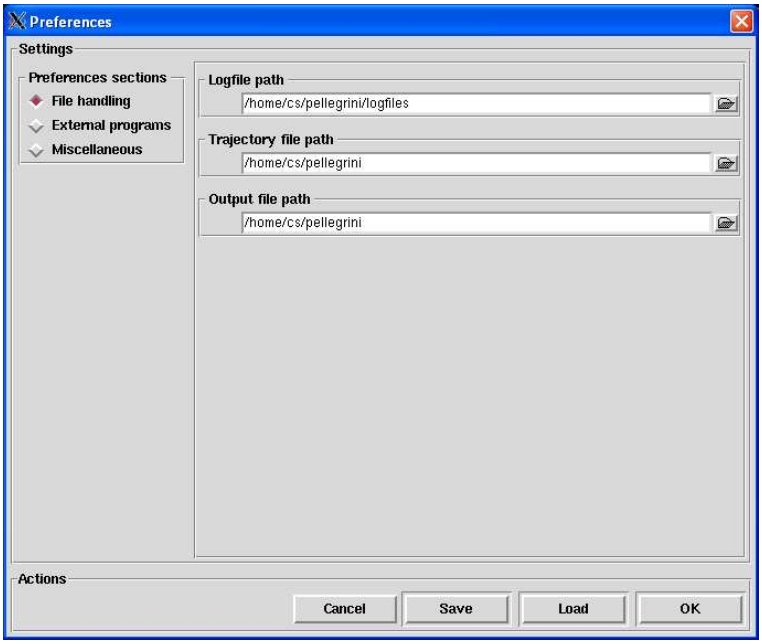

Figure 4.16: The dialog used to set up *n*MOLDYN preferences.

By default, the dialog for **File handling** section is displayed. Clicking on the **File handling**, **External programs** or **Miscellaneous** radiobutton will display the dialog corresponding to the selected section (see Figure 4.17).

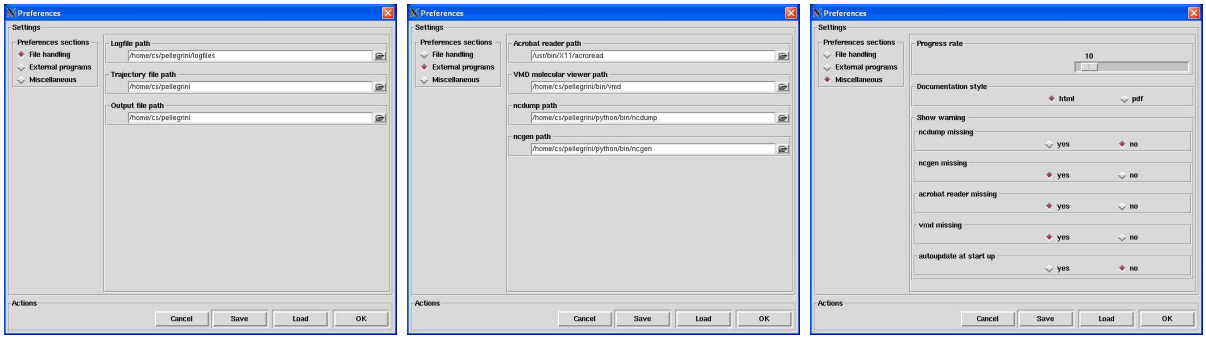

Figure 4.17: The three preferences sections dialogs available in *n*MOLDYN. On the left, the **file handling** section, on the middle the **external programs** section and on the right, the **miscellaneous** section.

Each preference section dialog contains an entry for each of its associated preferences variable initialized with a default value. Here is the list of the *n*MOLDYN preferences variables:

- **Logfile path**
  Section: File handling
  Preferences variable name: logfile_path
  Format: string
  Default: The user $HOME directory
  Description: The path for the directory where the *n*MOLDYN log files will be written.

- **Trajectory file path**
  Section: File handling
  Preferences variable name: trajfile_path
  Format: string
  Default: The user $HOME directory
  Description: The path for the directory where the MMTK NetCDF trajectories will be loaded by default.

- **Output file path**
  Section: File handling
  Preferences variable name: outputfile_path
  Format: string
  Default: The user $HOME directory
  Description: The path for the directory where the *n*MOLDYN output files will be written.

- **Acrobat reader path**
  Section: External programs
  Preferences variable name: acroread_path
  Format: string
  Default: *None*
  Description: The path for the acrobat reader executable. If this path is not set, it will not be possible to display the documentation in pdf format.

- **VMD molecular viewer path**
  Section: External programs
  Preferences variable name: vmd_path
  Format: string
  Default: *None*
  Description: The path for the **VMD** molecular viewer executable [33]. If this path is not set, the **Animate** and **Effective mode** options of the **View** menu will be disabled.

- **ncdump path**
  Section: External programs
  Preferences variable name: ncdump_path
  Format: string
  Default: *None*
  Description: The path for the **ncdump** executable. If this path is not set, the **Convert NetCDF to ASCII** option of the **File** menu will be disabled.

- **ncgen path**
  Section: External programs
  Preferences variable name: ncgen_path
  Format: string
  Default: *None*
  Description: The path for the **ncgen** executable. If this path is not set, the **Convert ASCII to NetCDF** option of the **File** menu will be disabled.

- **Progress rate**
  Section: Miscellaneous
  Preferences variable name: progress_rate
  Format: not an editable entry
  Default: *10*
  Description: The rate at which the progress of a *n*MOLDYN analysis will be displayed on the console and written in the *n*MOLDYN log file.

- **Documentation style**
  Section: Miscellaneous
  Preferences variable name: documentation_style
  Format: not an editable entry
  Default: *html*
  Description: The format for the *n*MOLDYN users guide when clicking on *Help –> Help* item and for the online help. HTML if **html** is selected, PDF if **pdf** is selected.

- **ncdump missing**
  Section: Miscellaneous
  Preferences variable name: warning_ncdump
  Format: no an editable entry
  Default: *yes*
  Description: If set to *yes*, you will be informed if **ncdump** was not found at each *n*MOLDYN start and each time the **Preferences** dialog is closed.

- **ncgen missing**
  Section: Miscellaneous
  Preferences variable name: warning_ncgen

Format: no an editable entry
Default: *yes*
Description: If set to *yes*, you will be informed if **ncgen** was not found at each *n*MOLDYN start and each time the **Preferences** dialog is closed.

- **VMD missing**
  Section: Miscellaneous
  Preferences variable name: warning_vmd
  Format: no an editable entry
  Default: *yes*
  Description: If set to *yes*, you will be informed if **VMD** was not found at each *n*MOLDYN start and each time the **Preferences** dialog is closed.

- **acrobat reader missing**
  Section: Miscellaneous
  Preferences variable name: warning_acroread
  Format: no an editable entry
  Default: *yes*
  Description: If set to *yes*, you will be informed if **acrobat reader** was not found at each *n*MOLDYN start and each time the **Preferences** dialog is closed.

The **Actions** frame contains four buttons which are respectively:

- **Cancel**
- **Save**
- **Load**
- **OK**

Pressing the **Cancel** button will cancel the preferences settings and close the preferences dialog leaving the preferences in the state they were when opening the **Preferences** dialog. Pressing the **Save** button will pop up a file browser from which you will select a location to save the preferences. By default, the preferences file name is:

`$USERPROFILE\Application Data\nMOLDYN\nMOLDYN.ini` on Windows,

`$HOME/.nMOLDYN` on Unix and,

`$HOME/Library/Preferences/nMOLDYN.pref` on MacOS

If those paths does not exist, they will be created. These default paths will be the ones that will be searched when *n*MOLDYN is started. Pressing the **Load** button will load a preferences file through a dialog. **OK** will use the settings for the running session of *n*MOLDYN but will not save them.

The figure 4.18 shows an example of a *n*MOLDYN preferences file built under a linux workstation. Please note the format that must be strictly respected.

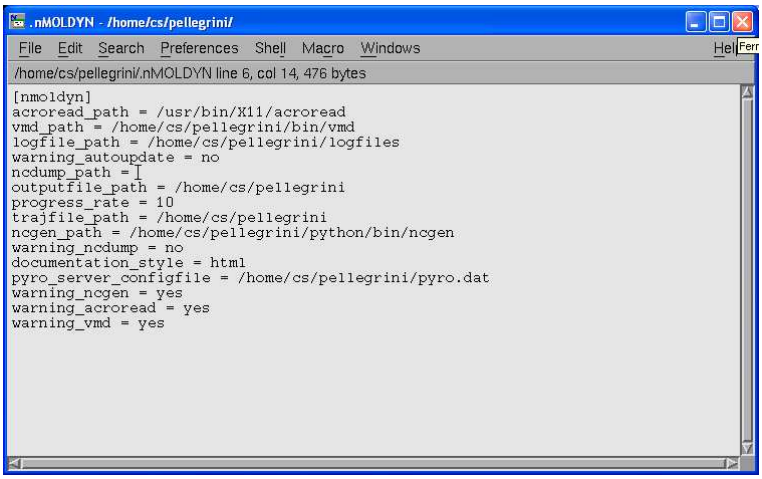

Figure 4.18: Example of a *n*MOLDYN preferences file.

As can be seen from that figure, the file must start with the line '[nmoldyn]' followed by th declaration of each preferences variables. For the vairables that may be missing in that file or with an empty value (e.g. ncdump_path in figure 4.18), the default value will be used.

4.1.7 Quit

Pressing the **Quit** menubutton will close *n*MOLDYN.

4.2 The Analysis menu

Pressing the **Analysis** menubutton brings up a menu from which it is possible to choose the following menus:

- Dynamics
- Scattering
- Structure
- NMR

In this section we will review each of these menus. However, we will first introduce three concepts that are common to almost all the analysis available in *n*MOLDYN which are:

- weighting scheme
- atom selection
- *n*MOLDYN running modes

4.2.1 Weighting scheme

In quantities that are averages over all atoms, nMOLDYN gives the possibility to choose between differents atomic weighting schemes. Presently, nMOLDYN implements the following schemes:

- Equal weighting:
$$\omega_\alpha = \frac{1}{N_{atoms}} \tag{4.1}$$
- Mass weighting:
$$\omega_\alpha = \frac{m_\alpha}{\sum_{\alpha=1}^{N_{atoms}} m_\alpha} \tag{4.2}$$
- Atomic number weighting:
$$\omega_\alpha = \frac{Z_\alpha}{\sum_{\alpha=1}^{N_{atoms}} Z_\alpha} \tag{4.3}$$
- Incoherent neutron scattering:
$$\omega_\alpha = \frac{b^2_{\alpha,\mathrm{inc}}}{\sum_{\alpha=1}^{N_{atoms}} b^2_{\alpha,\mathrm{inc}}} \tag{4.4}$$
- Coherent neutron scattering:
$$\sqrt{\omega_\alpha} = \frac{b_{\alpha,\mathrm{coh}}}{\sqrt{\sum_{\alpha=1}^{N_{atoms}} b^2_{\alpha,\mathrm{coh}}}} \tag{4.5}$$

where ω_α is the weight for atom α, N_{atoms} is the number of (selected) atoms in the system (or in the subsystem) for which the analysis is performed and m_α, Z_α, $b_{\alpha,\mathrm{inc}}$, and $b_{\alpha,\mathrm{coh}}$ are respectively the mass, the atomic number, the incoherent scattering length and the coherent scattering length of atom α where

$$b_{\alpha,\mathrm{coh}} = \overline{b_\alpha} \tag{4.6}$$
$$b_{\alpha,\mathrm{inc}} = \sqrt{\overline{b_\alpha^2} - \overline{b_\alpha}^2} \tag{4.7}$$

the average being done over isotopes and relative spin orientations of neutrons and nucleus. Using such a definition, we have $\sum_{\alpha=1}^{N} \omega_\alpha = 1$.

If we now group atoms into their different species A, B ... (e.g. oxygens, hydrogens ...) such that:

$$N = \sum_{I=1}^{N_{species}} n_I \tag{4.8}$$

where $N_{species}$ is the total number of selected species and n_I is the number of atoms of specie I. Then, we can define the weight for a given atomic specie I as:

- Equal weighting (per specie):
$$\mathcal{W}_\mathcal{I} = \frac{n_I}{N_{atoms}} \tag{4.9}$$
- Mass weighting (per specie):
$$\mathcal{W}_\mathcal{I} = \frac{n_I m_I}{\sum_{I=1}^{N_{species}} n_I m_I} \tag{4.10}$$

- Atomic number weighting:

$$\mathcal{W}_{\mathcal{I}} = \frac{n_I Z_I}{\sum_{I=1}^{N_{species}} n_I Z_I} \tag{4.11}$$

- Incoherent neutron scattering:

$$\mathcal{W}_{\mathcal{I}} = \frac{n_I b_{I,\text{inc}}^2}{\sum_{I=1}^{N_{species}} b_{I,\text{inc}}^2} \tag{4.12}$$

- Coherent neutron scattering:

$$\sqrt{\mathcal{W}_{\mathcal{I}}} = \frac{n_I b_{I,\text{coh}}}{\sqrt{\sum_{I=1}^{N_{species}} b_{I,\text{coh}}^2}} \tag{4.13}$$

and we have $\sum_{I=1}^{N_{species}} \mathcal{W}_{\mathcal{I}} = \sum_{I=1}^{N_{species}} \sum_{\alpha=1}^{n_I} \omega_{\alpha,I} = 1$ where $\omega_{\alpha,I}$ is the atomic weight of atom α of specie I defined in equations 4.1 to 4.2. The weigthing scheme based on specie will be useful when dealing with analysis for which partial terms can be defined.

4.2.2 Atom selection

It is sometimes necessary to define a subset of atoms on which a given action or analysis will be performed. *n*MOLDYN provided this possibility by a general mechanism that can be applied to several selection types such as:

- Subset selection: selection of a subset of atoms on which an analysis will be performed,
- Deuteration selection: definition of a subset of hydrogen atoms whose parameters will be the ones of deuterium. This allows to account for isotopic deuteration replacement performed in neutron experiments,
- Group selection: definition of one or several groups of atoms on which a given action will be performed collectively.

Albeit different in their nature, we will see in the following sections, that all these selections use the same syntax what is quite convenient from a user point of view.

4.2.2.1 Subset selection

This kind of selection is used when one wants to narrow an analysis on a given subset of atoms of the system. For instance, assuming that you performed a MD of a protein in a water box and that you are interested in calculating the diffusion constant of the protein via a **M**ean-**S**quare **D**isplacement (*MSD*) analysis. In that case, it will be necessary to perform the analysis only on the atoms of the protein.

By default, *n*MOLDYN consider all the atoms for an analysis. The dialog from which a subset selection is performed is displayed in figure 4.19.

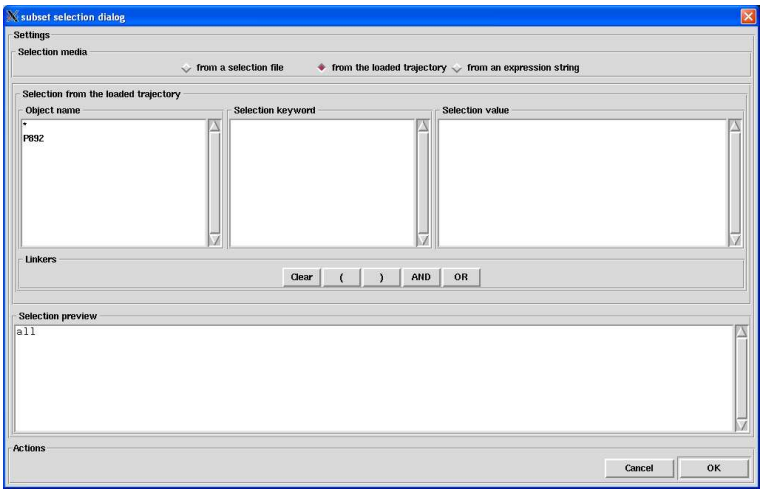

Figure 4.19: The dialog from where a subset selection is performed.

At the bottom of the dialog, the **Actions** frame contains the **Cancel** button to cancel the selection and the **OK** button to validate the selecton.

On the top of the dialog, three radiobuttons allows to select from which media the selection will be performed. This can be:

- **from a selection file**: this will perform the selection from a *n*MOLDYN subset selection file,
- **from the loaded trajectory**: this will perform the selection directly from the contents of the universe contained in the loaded trajectory,
- **from an expression string**: this will perform the selection from a valid python expression declaring a list of atoms to include in the selection.

When clicking on one of these radiobutton, a media-specific dialog will be displayed in the underneath frame.

selection from a selection file

To perform a subset selection from a selection file, you have to click on the **from a selection file** radiobutton. A dialog will be displayed in the underneath frame from which you will be asked for a subset selection file (see Fig. 4.20).

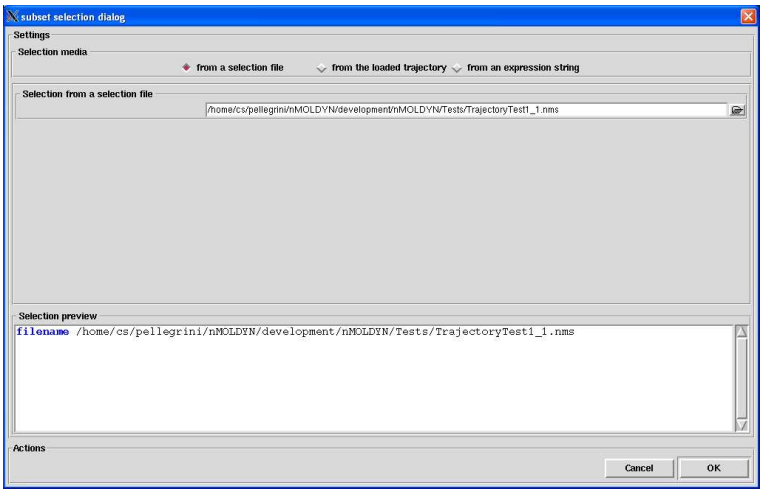

Figure 4.20: The subset selection dialog for a selection from a selection file.

The format of a *n*MOLDYN selection file is quite simple. It is an ASCII file with the **.nms** extension whose contents is a python script made of two lines. The first line must set the variable *pdb* to the path of a *PDB* file of the first frame of the trajectory being processed (it can be obtained using the frame extractor tool described in Section 4.1.3 for example).The second line must set the variable *subset* to a list of integers where each integers represents the *PDB* serial number of the atoms to select. An example of a subset selection file is shown in figure 4.21.

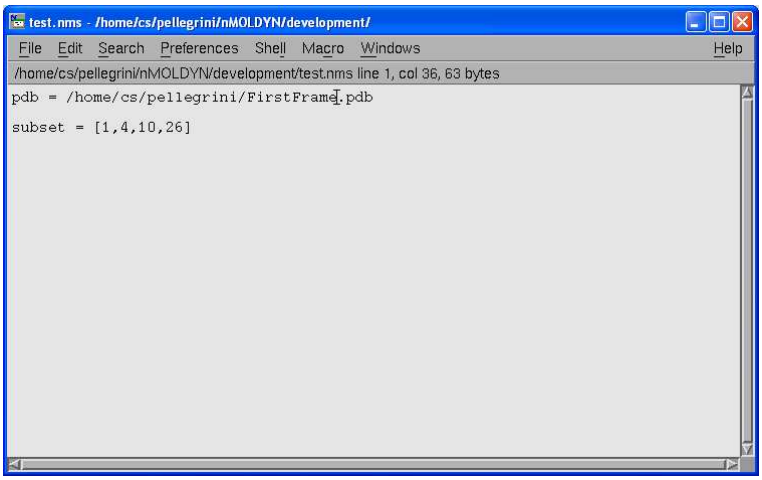

Figure 4.21: Example of a subset file.

Once a selection file has been loaded, the constructed selection string that will be used by *n*MOLDYN for this kind of selection is displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

selection from the loaded trajectory

To perform a subset selection from the loaded trajectory, you have to click on the **from the loaded trajectory** radiobutton. A dialog will be displayed in the underneath frame from

which you will construct your selection directly from the contents of the universe related to the loaded trajectory (see Fig. 4.22).

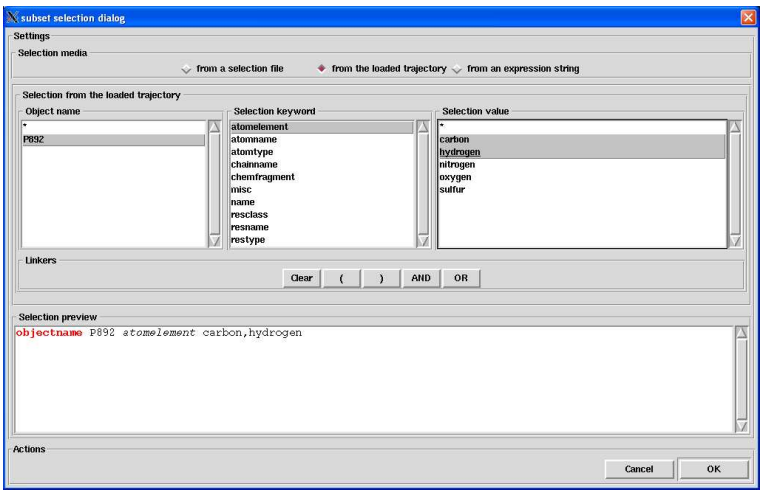

Figure 4.22: The subset selection dialog for a selection from the loaded trajectory.

To construct the selection, it is compulsory to proceed in the following order:

1. select an object name among the ones displayed in the **Object name** listbox. This will display in the **Selection keywords** listbox the selection keywords associated to the selected object.

2. select a selection keyword among the ones displayed in **Selection keywords** listbox. This will display in the **Selection value** listbox the values associated to the selected keyword.

3. unselect/select one or several values among the ones displayed in the **Selection value** listbox.

By doing so, you will construct a selection string with the following format:

objectname name keyword value1,value2, ...

where *name* is the selected object name (step 1), *keyword* is the selected keyword (step 2) and *value1,value2,...* are the selected values (step 3). This constructed selection string, that will be used by *n*MOLDYN, is displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

You can associate several selection keywords to a given selected object by repeating steps 2,3. In that case, the constructed selection string will have the following format:

objectname name keyword1 value1,value2,... OR keyword2, value1,value2,value3...

where the keyword *OR* will be interpreted by *n*MOLDYN as an union operator in the sense that it will take the union between the set of atoms generated by *keyword1 value1,value2*, *keyword2 value1,value2,value3* ...

You can also include several objects in the selection by repeating steps 1,2,3. In that case, the constructed selection string will have the following format:

objectname name1 keyword1 value1,... OR keyword2, value1,value2, ... OR objectname name2 keyword1 value1 ...

Finally, using the buttons within the **Linkers** frame each time a step 3 is completed allows to construct more complex selection strings using the **(**, **)**, **AND**, **OR** linkers, the **AND** linker acting as an intersection operator while the **OR** link, described above, acts as an union operator. The button **Clear** clears up the selection string under construction.

The table 4.1 lists the selection keywords and values depending on the MMTK type of the object being processed.

Object MMTK type	Selection keyword	Selection value
Atom	name	the object MMTK name
AtomCluster	atomelement	the element name (e.g. hydrogen)
	atomname	the atom MMTK name
	name	the object MMTK name
Molecule	chemfrag	the chemical fragment name. One of amine, hydroxy, methyl or thiol
	atomelement	the element name (e.g. hydrogen)
	atomname	the atom MMTK name
NucleotideChain	atomelement	the element name (e.g. hydrogen)
	atomname	the atom MMTK name
	atomtype	the atom MMTK type
	misc	one of backbone or bases
	name	the object MMTK name
	nuclname	the nucleotide MMTK name.
	nucltype	the nucleotide MMTK type
PeptideChain	chemfrag	the chemical fragment name. One of amine, c_alphas, hydroxy, methyl or thiol
	atomelement	the element name (e.g. hydrogen)
	atomname	the atom MMTK name
	atomtype	the atom MMTK type
	misc	one of backbone or sidechains
	name	the object MMTK name
	resclass	the residue class. One of acidic, aliphatic, aromatic, basic, charged, hydrophobic, polar or small
	resname	the residue MMTK name
	restype	the residue MMTK type
Protein	chemfrag	one of amine, c_alphas, hydroxy, methyl or thiol
	atomelement	the element name (e.g. hydrogen)
	atomname	the atom MMTK name
	atomtype	the atom MMTK type
	chainname	the chain MMTK name
	misc	one of backbone or sidechains
	name	the object MMTK name
	resclass	the residue class. One of acidic, aliphatic, aromatic, basic, charged, hydrophobic, polar or small
	resname	the residue MMTK name
	restype	the residue MMTK type

Table 4.1: List of selection keywords and their associated values according to the MMTK type of the selected object.

Here are some examples of subset selection strings constructed from a protein whose name is P892:

- *objectname P892 atomelement carbon*: will select only the carbon atoms,
- *objectname P892 atomelement carbon,hydrogen*: will select only the carbon and hydrogen atoms,
- *objectname P892 atomelement oxygen and restype Ala,Arg*: will select only the oxygen atoms of Alanine and Arginine residues,
- *objectname P892 atomelement carbon OR resclass acidic*: will select all the carbon atoms plus the all the atoms of the acidic residues.

selection from an expression string

To perform a subset selection from an expression string, you have to click on the **from an expression string** radiobutton. A dialog will be displayed in the underneath frame from which you will enter a valid Python expression that must declare the *selection* variable as a list of atoms of the loaded universe. The variable *self.universe* will be used as a reference for that universe (see Fig. 4.23).

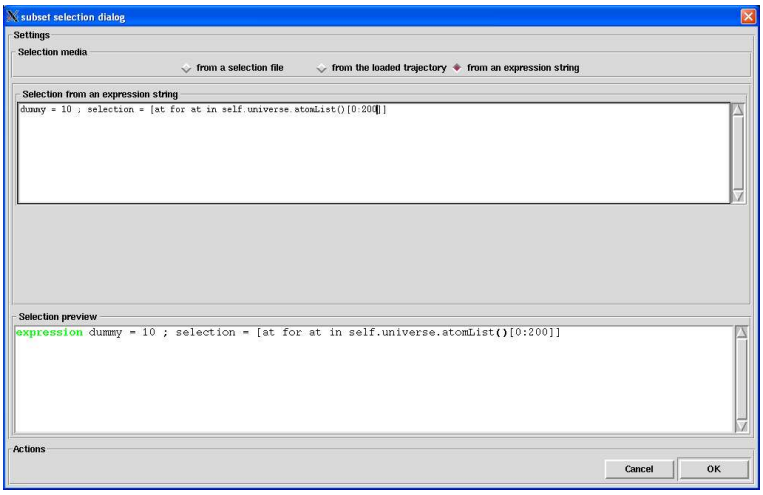

Figure 4.23: The subset selection dialog for a selection from an expression string.

Once an expression string has been entered press **Return** to register it. The constructed selection string that will be used by *n*MOLDYN for this kind of selection will be displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

Here are some examples of valid expression strings that can be entered:

- *selection = self.universe.atomList()[0:10]*: will select the ten first atoms of the universe,
- *selection = [at for at in self.universe if at._mass ≥ 10]*: will select only the atoms of the universe whose mass is greater than 10 *amu*.

4.2.2.2 Deuteration selection

This kind of selection is useful for analysis if you want to change the parameters (e.g. mass, scattering lengths) of some hydrogens atoms to the ones of deuterium. This allows to simulate the system in a fully or partially deuterated state.

By default, *n*MOLDYN does not select any hydrogen atom for deuteration for an analysis. The dialog from which a deuteration selection is performed is displayed in figure 4.24.

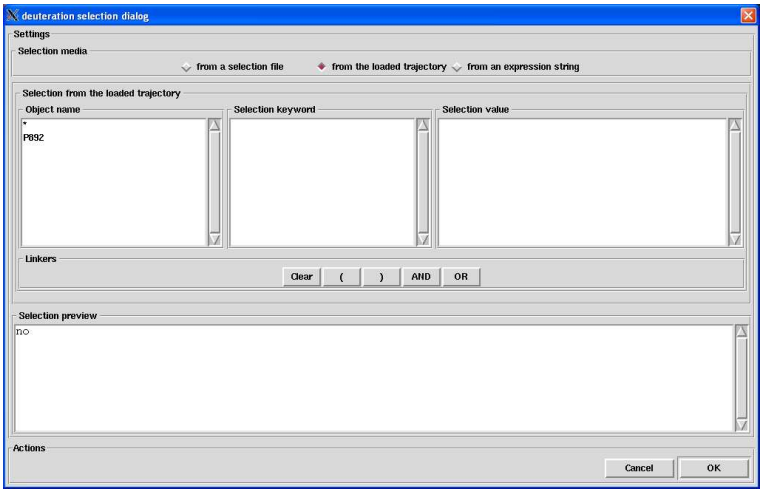

Figure 4.24: The dialog from where a deuteration selection is performed.

As can be seen from that figure, the deuteration dialog is exactly the same that the subset selection dialog. At the bottom of the dialog, the **Actions** frame contains the **Cancel** button to cancel the selection and the **OK** button to validate the selecton.

On the top of the dialog, three radiobuttons allows to select from which media the selection will be performed. This can be:

- **from a selection file**: this will perform the selection from a *n*MOLDYN deuteration selection file,
- **from the loaded trajectory**: this will perform the selection directly from the contents of the universe contained in the loaded trajectory,
- **from an expression string**: this will perform the selection from a valid python expression declaring a list of atoms to include in the selection.

When clicking on one of these radiobutton, a media-specific dialog will be displayed in the underneath frame.

selection from a selection file

To perform a deuteration selection from a selection file, you have to click on the **from a selection file** radiobutton. A dialog will be displayed in the underneath frame from which you will be asked for a deuteration selection file (see Fig. 4.25).

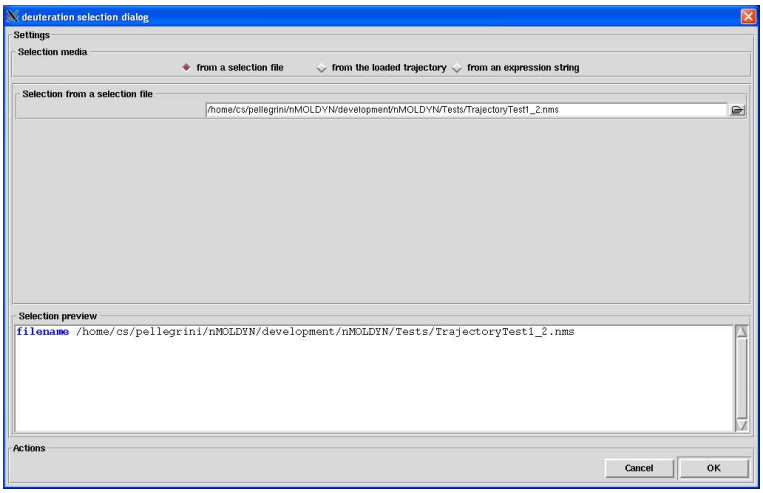

Figure 4.25: The deuteration selection dialog for a selection from a selection file.

The format of a *n*MOLDYN deuteration selection file is quite simple. It is an ASCII file with the **.nms** extension whose contents is a python script made of two lines. The first line must set the variable *pdb* to the path of a *PDB* file of the first frame of the trajectory being processed (it can be obtained using the frame extractor tool described in Section 4.1.3 for example).The second line must set the variable *deuteration* to a list of integers where each integers represents the *PDB* serial number of the atoms to select. An example of a deuteration selection file is shown in figure 4.26.

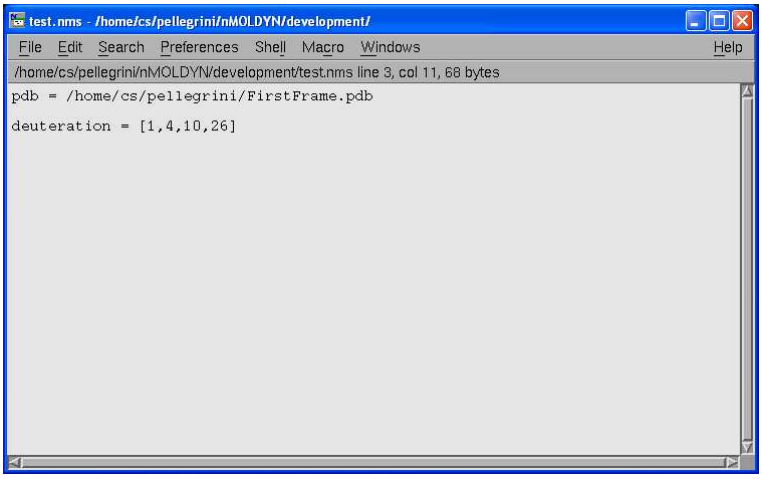

Figure 4.26: Example of a deuteration file.

Once a selection file has been loaded, the constructed selection string that will be used by *n*MOLDYN for this kind of selection is displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

selection from the loaded trajectory

To perform a deuteration selection from the loaded trajectory, you have to click on the **from the loaded trajectory** radiobutton. A dialog will be displayed in the underneath frame from which you will construct your selection directly from the contents of the universe related to the loaded trajectory (see Fig. 4.27).

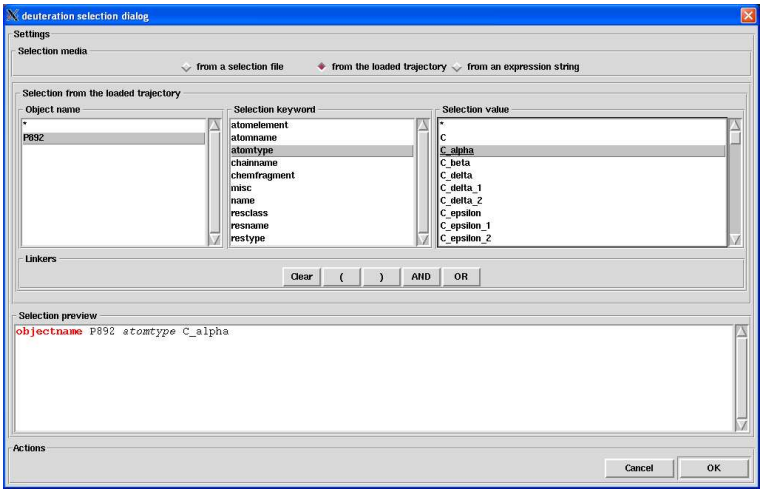

Figure 4.27: The deuteration selection dialog for a selection from the loaded trajectory.

To construct the selection, it is compulsory to proceed in the following order:

1. select an object name among the ones displayed in the **Object name** listbox. This will display in the **Selection keywords** listbox the selection keywords associated to the selected object.

2. select a selection keyword among the ones displayed in **Selection keywords** listbox. This will display in the **Selection value** listbox the values associated to the selected keyword.

3. unselect/select one or several values among the ones displayed in the **Selection value** listbox.

By doing so, you will construct a selection string with the following format:

objectname name keyword value1,value2, ...

where *name* is the selected object name (step 1), *keyword* is the selected keyword (step 2) and *value1,value2,...* are the selected values (step 3). This constructed selection string, that will be used by *n*MOLDYN, is displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

You can associate several selection keywords to a given selected object by repeating steps 2,3. In that case, the constructed selection string will have the following format:

objectname name keyword1 value1,value2,... OR keyword2, value1,value2,value3...

where the keyword *OR* will be interpreted by *n*MOLDYN as an union operator in the sense that it will take the union between the set of atoms generated by *keyword1 value1,value2*, *keyword2 value1,value2,value3* ...

You can also include several objects in the selection by repeating steps 1,2,3. In that case, the constructed selection string will have the following format:

objectname name1 keyword1 value1,... OR keyword2, value1,value2, ... OR objectname name2 keyword1 value1 ...

Finally, using the buttons within the **Linkers** frame each time a step 3 is completed allows to construct more complex selection strings using the **(**, **)**, **AND**, **OR** linkers, the **AND** linker acting as an intersection operator while the **OR** link, described above, acts as an union operator. The button **Clear** clears up the selection string under construction.

The table 4.1 lists the selection keywords and values depending on the MMTK type of the object being processed.

selection from an expression string

To perform a deuteration selection from an expression string, you have to click on the **from an expression string** radiobutton. A dialog will be displayed in the underneath frame from which you will enter a valid Python expression that must declare the *selection* variable as a list of atoms of the loaded universe. The variable *self.universe* will be used as a reference for that universe (see Fig. 4.28).

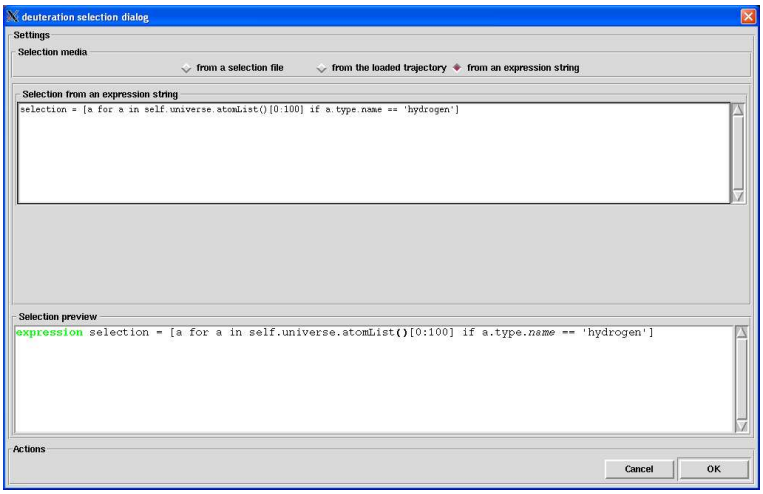

Figure 4.28: The deuteration selection dialog for a selection from an expression string.

Once an expression string has been entered press **Return** to register it. The constructed selection string that will be used by *n*MOLDYN for this kind of selection will be displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

Whatever the selection media used, in case where non-hydrogens atoms were accidently introduced in the deuteration selection, *n*MOLDYN will filtered them out from the resulting selection.

4.2.2.3 Group selection

This kind of selection is especially desgned for analysis based on group of atoms on which a given operation is performed collectively. By collectively, we mean that the selected group of atoms will be treated as it was one object. For example, in rigid-body based analysis, selecting a group of atoms will mean that this group will be considered as a single rigid-body that will be used to derve the rigid-body trajectory. Another example, the study where one needs to define the center of mass of a group of atoms in order to derive a property such as coordination number or spatial density (see Sections 4.2.6.2 and 4.2.6.3). In that case, a center of mass will be defined for each selected group of atoms.

By default, *n*MOLDYN consider all the atoms for a group selection. The dialog from which a group selection is performed is displayed in figure 4.29.

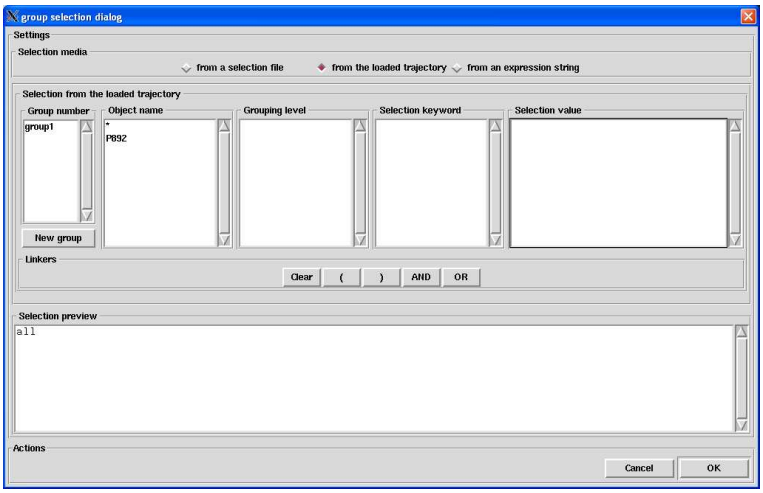

Figure 4.29: The dialog from where a group selection is performed.

At the bottom of the dialog, the **Actions** frame contains the **Cancel** button to cancel the selection and the **OK** button to validate the selecton.

On the top of the dialog, three radiobuttons allows to select from which media the selection will be performed. This can be:

- **from a selection file**: this will perform the selection from a *n*MOLDYN group selection file,
- **from the loaded trajectory**: this will perform the selection directly from the contents of the universe contained in the loaded trajectory,
- **from an expression string**: this will perform the selection from a valid python expression declaring a list of atoms to include in the selection.

When clicking on one of these radiobutton, a media-specific dialog will be displayed in the underneath frame.

selection from a selection file

To perform a group selection from a selection file, you have to click on the **from a selection file** radiobutton. A dialog will be displayed in the underneath frame from which you will be asked for a group selection file (see Fig. 4.30).

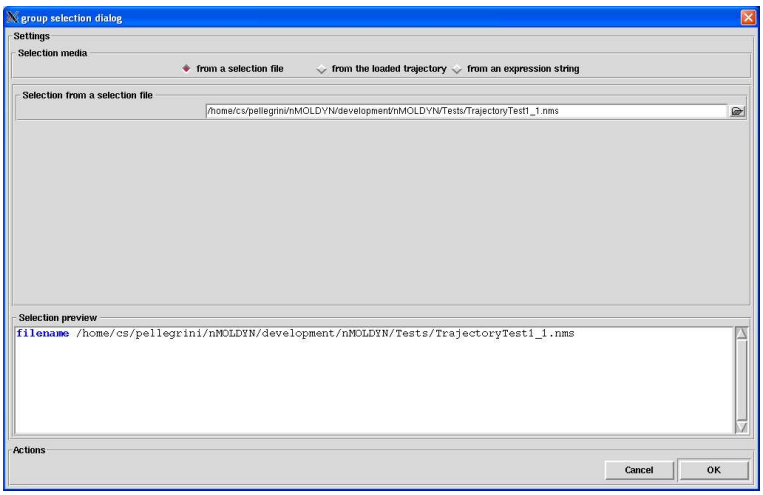

Figure 4.30: The group selection dialog for a selection from a selection file.

The format of a *n*MOLDYN group selection file is quite simple. It is an ASCII file with the **nms** extension whose contents is a python script made of two lines. The first line must set the variable *pdb* to the path of a *PDB* file of the first frame of the trajectory being processed (it can be obtained using the frame extractor tool described in Section 4.1.3 for example).The second line must set the variable *deuteration* to nested lists of integers where each nested list will generate a group of atoms whose *PDB* serial number match the integer of the nested list. An example of a deuteration selection file is shown in figure 4.31.

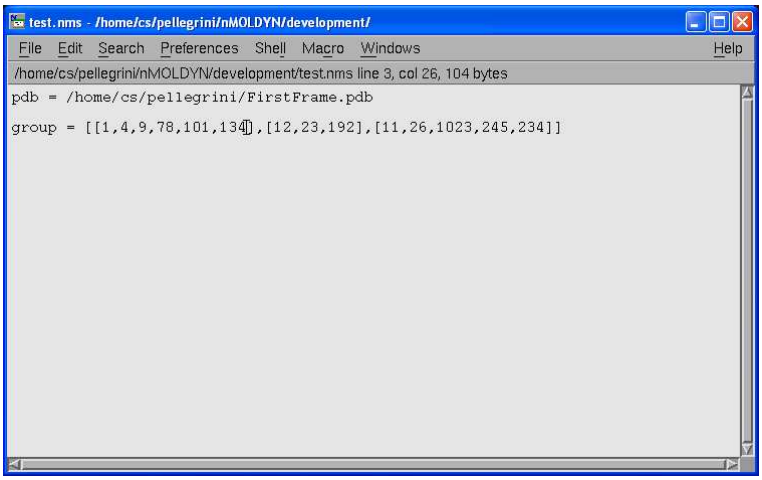

Figure 4.31: Example of a group file.

This example will create three groups. The first one made of atoms 1, 4, 9, 78, 101 and 134 of the *PDB* file, the second one made of the atoms 12, 23 and 192 and the third one made of the atoms 11, 26, 1023, 245 and 234.

Once a selection file has been loaded, the constructed selection string that will be used by nMOLDYN for this kind of selection is displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

selection from the loaded trajectory

To perform a group selection from the loaded trajectory, you have to click on the **from the loaded trajectory** radiobutton. A dialog will be displayed in the underneath frame from which you will construct your selection directly from the contents of the universe related to the loaded trajectory (see Fig. 4.32).

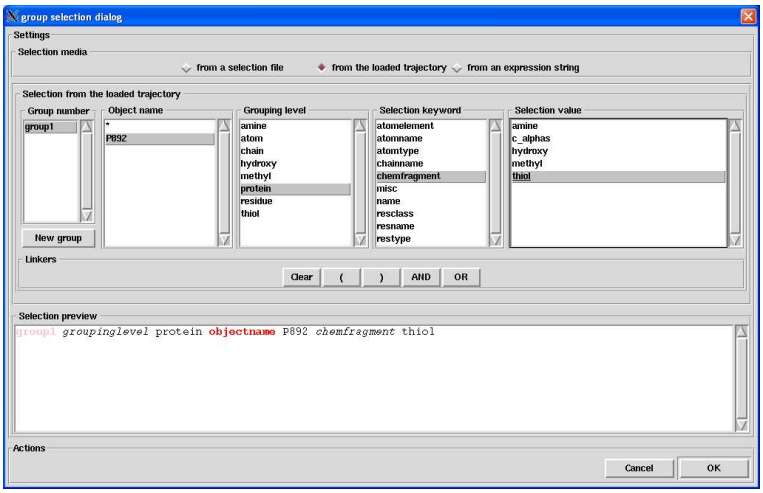

Figure 4.32: The group selection dialog for a selection from the loaded trajectory.

To construct the selection, it is compulsory to proceed in the following order:

1. create a new type of group by clicking on **New group** button or overwrite an existing one by selecting it in the **Group number** listbox,

2. select an object name among the ones displayed in the **Object name** listbox. This will display the grouping levels associated to the selected object in the **Grouping level** listbox and the selection keywords associated to the selected object in the **Selection keywords** listbox,

3. select a grouping level among the ones displayed in the **Grouping level** listbox. This is specific to group selection as when selecting a group of atoms, you must specify at which level the selected atoms will be grouped. For example, if you perform a group selection on a protein. If you set the grouping level to *protein*, then the group will be the whole protein. But, if you set the grouping level to *residue*, then the group selection will results on a set of $N_{residues}$ group where $N_{residues}$ is the number of residues of the protein.

4. select a selection keyword among the ones displayed in **Selection keywords** listbox. This will display in the **Selection value** listbox the values associated to the selected keyword.

5. unselect/select one or several values among the ones displayed in the **Selection value** listbox.

By doing so, you will construct a selection string with the following format:

group1 groupinglevel level objectname name keyword value1,value2, ...

where *group1* is the name of the group, *level* is the selected grouping level, *name* is the selected object name (step 1), *keyword* is the selected keyword (step 2) and *value1,value2,...* are the selected values (step 3). This constructed selection string is displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

You can associate several selection keywords to a given selected object by repeating steps 2,3. In that case, the constructed selection string will have the following format:

group1 groupinglevel level objectname name keyword1 value1,value2,... OR keyword2, value1,value2,value3...

where the keyword *OR* will be interpreted by *n*MOLDYN as an union operator in the sense that it will take the union between the set of atoms generated by *keyword1 value1,value2*, *keyword2 value1,value2,value3* ...

You can also include several objects in the selection by repeating steps 1,2,3. In that case, the constructed selection string will have the following format:

group1 groupinglevel level objectname name1 keyword1 value1,... OR keyword2, value1,value2, ... OR objectname name2 keyword1 value1 ...

Finally, using the buttons within the **Linkers** frame each time a step 3 is completed allows to construct more complex selection strings using the **(**, **)**, **AND**, **OR** linkers, the **AND** linker acting as an intersection operator while the **OR** link, described above, acts as an union operator. The button **Clear** clears up the selection string under construction.

The table 4.1 lists the selection keywords and values depending on the MMTK type of the object being processed.

Here are some examples of group selection strings constructed from a protein whose name is P892 and whose number of residues is 58:

- *group1 groupinglevel residue objectname P892 atomelement carbon*: will create 58 groups made of the carbon atoms of each residue,
- *group1 groupinglevel protein objectname P892 atomelement carbon,hydrogen*: will create 1 group made of the carbon and hydrogens of the whole protein,
- *group1 groupinglevel amine objectname P892 atomelement nitrogen group2 groupinglevel hydroxy objectname P892 atomelement oxygen*: will create two families of groups. The first one contains groups made of the nitrogen atom of the each amine group of the protein. The second one contains groups made of the oxygen atom of the each hydroxy group of the protein.

selection from an expression string

To perform a group selection from an expression string, you have to click on the **from an expression string** radiobutton. A dialog will be displayed in the underneath frame from which you will enter a valid Python expression that must declare the *selection* variable as nested lists of atoms of the loaded universe. The variable *self.universe* will be used as a reference for that universe (see Fig. 4.33).

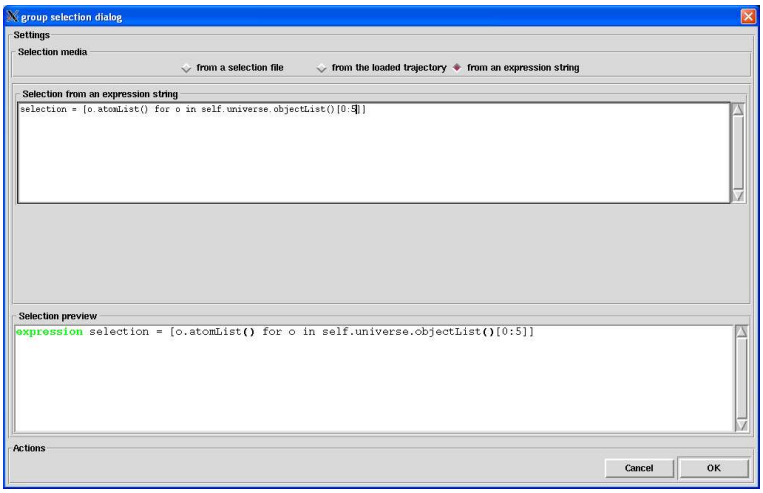

Figure 4.33: The group selection dialog for a selection from an expression string.

Here are some examples of valid expression strings that can be entered:

- *selection = self.universe.objectList[0].atomList()*: will pack all the atoms of the first object of the universe in one group,
- *selection = [o.atomList()[0:10] for o in self.universe.objectList()]*: will create a distinct group for each object of the universe with its ten first atoms.

Once an expression string has been entered press **Return** to register it. The constructed selection string that will be used by *n*MOLDYN for this kind of selection will be displayed in the **Selection preview** entry at the bottom of the dialog with highlighted keywords.

4.2.3 Running modes

All the analysis dialogs in *n*MOLDYN contains a frame called **Actions** located at the bottom of the dialog. That frame contains the buttons

- Cancel,
- Estimate,
- Save,
- Run,
- Save And Run.

The **Cancel** button cancel the analysis and close the dialog. The **Estimate** button performs just a single step of the analysis and pops up a message box with the estimated time for the full analysis. Not all the analysis can be estimated. In that case, the popped up message will inform you that the analysis is not estimable. The **Save** button pops up a dialogs from where you can save the analysis settings either to a *n*MOLDYN autostart file either to a *n*MOLDYN input file. The **Run** button runs the analysis directly from the *GUI* and finally the **Save & and Run** combinates the actions of **Save** and **Run** buttons.

4.2.4 The Dynamics menu

Pressing the button **Dynamics** brings up a menu from which it is possible to choose the following analysis:

- Mean-Square Displacement
- Root Mean-Square Displacement
- Radius Of Gyration
- Velocity AutoCorrelation Function
- Density Of States
- Pass-Band Filtered Trajectory
- Global Motion Trajectory
- Center Of Mass Trajectory
- Rigid-Body Trajectory
- Center Of Mass Trajectory
- Auto-Regressive Analysis
- Quasi-Harmonic Analysis
- Reorientational Correlation Function
- Angular Velocity AutoCorrelation Function
- Angular Density Of States

4.2.4.1 Mean-Square Displacement

Theory and implementation

Molecules in liquids and gases do not stay in the same place, but move constantly. This process is called diffusion and it happens quite naturally in fluids at equilibrium. During this process, the motion of an individual molecule does not follow a simple path [48]. As it travels, the molecule undergoes some collisions with other molecules which prevent it from following a straight line. If the path is examined in close detail, it will be seen to be a good approximation to a random walk. Mathematically, a random walk is a series of steps where each step is taken in a completely random direction from the one before. This kind of path was famously analysed by Albert Einstein in a study of Brownian motion. He showed that the *MSD* of a particle following a random walk is proportional to the time elapsed. This relationship can be written as

$$< r^2 >= 6Dt + C \tag{4.14}$$

where $< r^2 >$ is the *MSD* and t is the time. D and C are constants. The constant D defines the so-called diffusion coefficient.

The figure 4.34 shows an example of a *MSD* analysis performed on a waterbox of 768 water molecules.

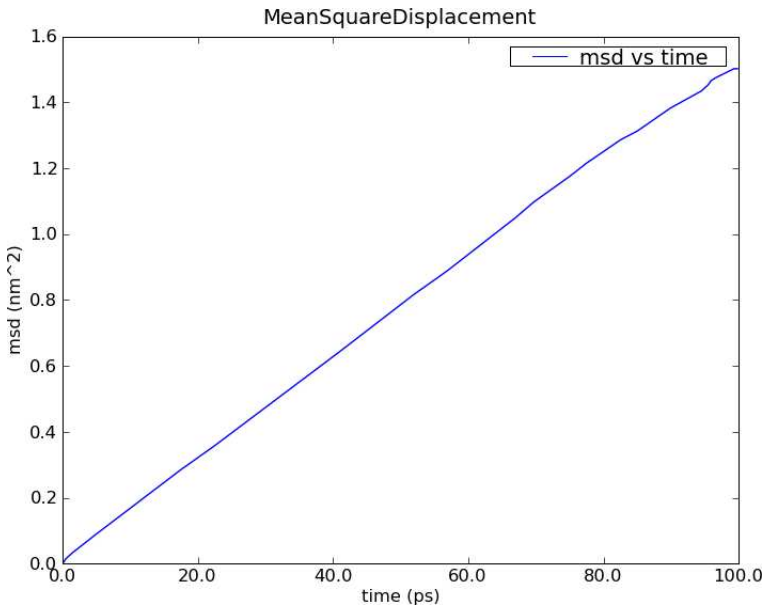

Figure 4.34: *MSD* calculated for a 100 ps MD simulation of 256 water molecules using NPT condition at 1 bar and 300 K.

To get the diffusion coefficient out of this plot, the slope of the linear part of the plot should be calculated.

Defining,

$$\mathbf{d}_\alpha(t, t_0) \doteq \mathbf{R}_\alpha(t_0 + t) - \mathbf{R}_\alpha(t_0). \tag{4.15}$$

the *MSD* of particle α can be defined as:

$$\Delta^2_\alpha(t) = \left\langle \mathbf{d}^2_\alpha(t, t_0) \right\rangle_{t_0} \tag{4.16}$$

where $\mathbf{R}_\alpha(t_0)$ and $\mathbf{R}_\alpha(t_0 + t)$ are respectively the position of particle α at times t_0 and $t_0 + t$. One can introduce a *MSD* with respect to a given axis $\mathbf{n}$:

$$\Delta^2_\alpha(t, t_0; \mathbf{n}) \doteq \left\langle d^2_\alpha(t, \tau; \mathbf{n}) \right\rangle_{t_0} \tag{4.17}$$

with

$$d_\alpha(t, t_0; \mathbf{n}) \doteq \mathbf{n} \cdot \mathbf{d}_\alpha(t, t_0). \tag{4.18}$$

The calculation of *MSD* is the standard way to obtain diffusion coefficients from MD simulations. Assuming Einstein-diffusion in the long time limit one has for isotropic systems

$$D_\alpha = \lim_{t \to \infty} \frac{1}{6t} \Delta^2_\alpha(t). \tag{4.19}$$

There exists also a well-known relation between the *MSD* and the velocity autocorrelation function. Writing $\mathbf{d}_\alpha(t) = \int_0^t d\tau\, \mathbf{v}_\alpha(\tau)$ in Eq. (4.16) one can show (see e.g. [50]) that

$$\Delta^2_\alpha(t) = 6 \int_0^t d\tau\, (t - \tau) C_{vv;\alpha\alpha}(\tau). \tag{4.20}$$

Using now the definition (4.19) of the diffusion coefficient one obtains the relation

$$D_\alpha = \int_0^t d\tau\, C_{vv;\alpha\alpha}(\tau). \tag{4.21}$$

With Eq. (4.41) this can also be written as

$$D_\alpha = \pi \tilde{C}_{vv;\alpha\alpha}(0). \tag{4.22}$$

Computationally, the *MSD* is calculated using the **F**ast **C**orrelation **A**lgorithm (FCA) [51]. In this framework, in the discrete case, the mean-square displacement of a particle is given by

$$\Delta^2(m) = \frac{1}{N_t - m} \sum_{k=0}^{N_t-m-1} [\mathbf{r}(k+m) - \mathbf{r}(k)]^2, \qquad m = 0 \ldots N_t - 1, \tag{4.23}$$

where $\mathbf{r}(k)$ is the particle trajectory and N_t is the the number of frames of the trajectory. We define now the auxiliary function

$$S(m) \doteq \sum_{k=0}^{N_t-m-1} [\mathbf{r}(k+m) - \mathbf{r}(k)]^2, \qquad m = 0 \ldots N_t - 1, \tag{4.24}$$

which is splitted as follow:

$$\begin{aligned} S(m) &= S_{AA+BB}(m) - 2S_{AB}(m), & (4.25)\\ S_{AA+BB}(m) &= \sum_{k=0}^{N_t-m-1} [\mathbf{r}^2(k+m) + \mathbf{r}^2(k)], & (4.26)\\ S_{AB}(m) &= \sum_{k=0}^{N_t-m-1} \mathbf{r}(k) \cdot \mathbf{r}(k+m). & (4.27) \end{aligned}$$

The function $S_{AB}(m)$ can be computed using the FCA method described in Section A. For $S_{AA+BB}(m)$ the following recursion relation holds:

$$\begin{aligned} S_{AA+BB}(m) &= S_{AA+BB}(m-1) - \mathbf{r}^2(m-1) - \mathbf{r}^2(N_t - m), & (4.28)\\ S_{AA+BB}(0) &= \sum_{k=0}^{N_t-1} \mathbf{r}^2(k). & (4.29) \end{aligned}$$

This allows one to construct the following efficient scheme for the computation of the *MSD*:

1. Compute $DSQ(k) = \mathbf{r}^2(k), \qquad k = 0 \ldots N_t - 1; \qquad DSQ(-1) = DSQ(N_t) = 0.$
2. Compute $SUMSQ = 2 \cdot \sum_{k=0}^{N_t-1} DSQ(k)$.
3. Compute $S_{AB}(m)$ using the FFT method.
4. Compute *MSD(m)* in the following loop:

$$\begin{aligned} SUMSQ &\leftarrow SUMSQ - DSQ(m-1) - DSQ(N_t - m)\\ MSD(m) &\leftarrow (SUMSQ - 2 \cdot S_{AB}(m)/(N_t - m)\\ m &\text{ running from 0 to } N_t - 1 \end{aligned}$$

It should be noted that the efficiency of this algorithm is the same as for the FCA computation of time correlation functions since the number of operations in step (1), (2), and (4) grows linearly with N_t.

Parameters

Pressing the **Mean-Square Displacement** button will pop up the dialog shown on figure 4.35

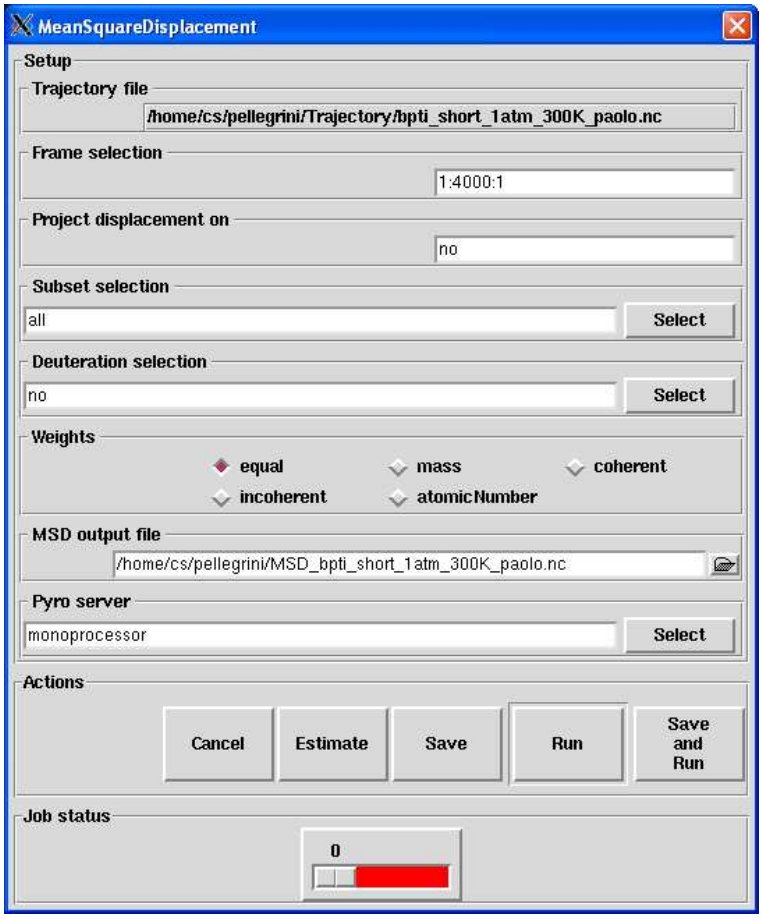

Figure 4.35: The dialog from where the *MSD* analysis will be set up and run.

The following input fields controls the parameters for the *MSD* analysis:

- **Trajectory file**
  Format: Not an editable entry
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Project displacement on**
  Format: string
  Default: *no*
  Description: this widget allows to specify a vector along which the *MSD* will be computed. This vector does not need to be normalized as *n*MOLDYN will perform the normalization when processing it. The entered value must have the following format:

  vx:vy:vz

  where *vx*, *vy* and *vz* are floats that represent respectively the x, y and z coordinates of the vector.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *equal*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *MSD*. See Section 4.2.1 for more details.

- **MSD output file**
  Format: string
  Default: *MSD_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *MSD* analysis. A CDL version of the NetCDF output file is also automatically created with *MSD_traj_file.cdl* name.

Output

The results of a *MSD* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which the *MSD* was evaluated,
- msd: the corresponding *MSD* in nm^2.

4.2.4.2 Root Mean-Square Deviation

Theory and implementation

The **R**oot **M**ean-**S**quare **D**eviation (*RMSD*) is maybe the most popular estimator of structural similarity. It is a numerical measure of the difference between two structures that can be defined as:

$$RMSD(t) = \sqrt{\frac{\sum_{\alpha=1}^{N_\alpha}(\mathbf{r}_\alpha(t) - \mathbf{r}_\alpha(t_{ref}))}{N_\alpha}} \tag{4.30}$$

where N_α is the number of atoms of the system, and $\mathbf{r}_\alpha(t)$ and $\mathbf{r}_\alpha(t_{ref})$ are respectively the position of atom α at time t and t_{ref} where t_{ref} is a reference time usually choosen as the first step of the simulation. Typically, *RMSD* is used to quantify the structural evolution of the system during the simulation. It can provide precious information about the system especially if it reached equilibrium or conversely if major structural changes occured during the simulation.

In *n*MOLDYN, *RMSD* is computed using the discretized version of equation 4.30:

$$RMSD(n \cdot \Delta t) = \sqrt{\frac{\sum_{\alpha=1}^{N_\alpha}(\mathbf{r}_\alpha(t) - \mathbf{r}_{ref}(t))}{N_\alpha}}, \qquad n = 0 \ldots N_t - 1. \tag{4.31}$$

where N_t is the number of frames and Δt is the time step.

Parameters

Pressing the **Root Mean-Square Displacement** button will pop up the dialog shown on figure 4.36

The following input fields controls the parameters for the *RMSD* analysis:

- **Trajectory file**
  Format: Not an editable entry
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

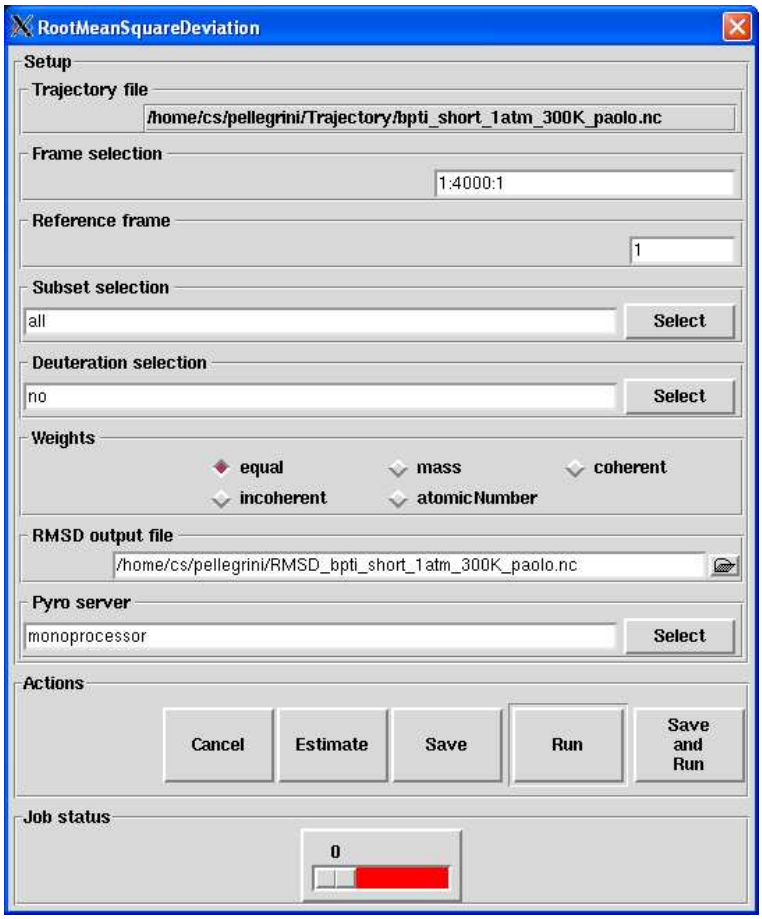

Figure 4.36: The dialog from where the *RMSD* analysis will be set up and run.

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Reference frame**
  Format: integer in [1,*traj_length*] where *traj_length* is the number of frames of the input trajectory
  Default: *1*
  Description: this widget allows to specify which frame should be the reference for the *RMSD* analysis. The value entered should be an integer ranging from 1 to *traj_length* where *traj_length* is the number of rames of the input trajectory.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *equal*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *RMSD*. See Section 4.2.1 for more details.

- **RMSD output file**
  Format: string
  Default: *RMSD_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *RMSD* analysis. A CDL version of the NetCDF output file is also automatically created with *RMSD_traj_file.cdl* name.

Output

The results of a *RMSD* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which the *RMSD* was evaluated,
- rmsd: the corresponding *RMSD* in *nm*.

4.2.4.3 Radius of gyration

Theory and implementation

Radius **O**f **G**yration (*ROG*) is the name of several related measures of the size of an object, a surface, or an ensemble of points. It is calculated as the Root Mean Square Distance between the system and a reference that can be either the center of gravity of the system either a given axis. In *n*MOLDYN, the reference is choosen to be the center of gravity of the system under study. Mathematically, it can be defined as:

$$ROG(t) = \sqrt{\frac{\sum_{\alpha=1}^{N_\alpha}(\mathbf{r}_\alpha(t) - \mathbf{r}_{cms}(t))}{N_\alpha}} \tag{4.32}$$

where N_α is the number of atoms of the system, and $\mathbf{r}_\alpha(t)$ and $\mathbf{r}_{cms}(t)$ are respectively the position of atom α and the center of mass of the system at time t.

ROG describes the overall spread of the molecule and as such is a good measure for the molecule compactness. For example, it can be useful when monitoring folding process.

In *n*MOLDYN, *ROG* is computed using the discretized version of equation 4.32:

$$ROG(n \cdot \Delta t) = \sqrt{\frac{\sum_{\alpha=1}^{N_\alpha}(\mathbf{r}_\alpha(t) - \mathbf{r}_{cms}(t))}{N_\alpha}}, \qquad n = 0 \ldots N_t - 1. \tag{4.33}$$

where N_t is the number of frames and Δt is the time step.

Parameters

Pressing the **Radius Of Gyration** button will pop up the dialog shown on figure 4.37

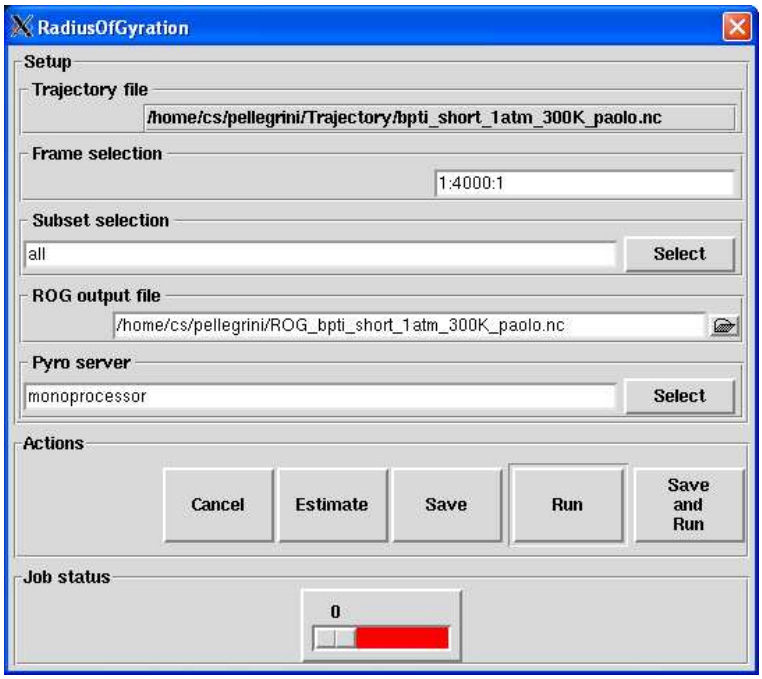

Figure 4.37: The dialog from where the *ROG* analysis will be set up and run.

The following input fields controls the parameters for the *ROG* analysis:

- **Trajectory file**
  Format: Not an editable entry
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  ⋆ *2:10:3* will select the frames 2, 5 and 8.

  ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *equal*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *ROG*. See Section 4.2.1 for more details.

- **ROG output file**
  Format: string
  Default: *ROG_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *ROG* analysis. A CDL version of the NetCDF output file is also automatically created with *ROG_traj_file.cdl* name.

Output

The results of a *ROG* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which the *ROG* was evaluated,
- rog: the corresponding *ROG* in *nm*.

4.2.4.4 Angular Correlation

Theory and implementation

The angular correlation analysis computes the autocorrelation of a set of vectors describing the extent of a molecule in three orthogonal directions. This kind of analysis can be useful when trying to highlight the fact that a molecule is constrainted in a given direction.

For a given triplet of non-colinear atoms g=(**a1**,**a2**,**a3**), one can derive an orthonormal set of three vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ using the following scheme:

- $\mathbf{v}_1 = \frac{\mathbf{n}_1+\mathbf{n}_2}{||\mathbf{n}_1+\mathbf{n}_2||}$ where $\mathbf{n}_1$ and $\mathbf{n}_2$ are respectively the normalized vectors along (**a1**,**a2**) and (**a1**,**a3**) directions.
- $\mathbf{v}_2$ is defined as the clockwise normal vector orthogonal to $\mathbf{v}_1$ that belongs to the plane defined by **a1**, **a2** and **a3** atoms

- $\vec{v_3} = \vec{v_1} \times \vec{v_2}$

Thus, one can define the following autocorrelation functions for the vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ defined on triplet t:

$$AC_{g,i}(t) = \langle \mathbf{v}_{t,i}(0) \cdot \mathbf{v}_{t,i}(t) \rangle, \qquad i = 1, 2, 3 \tag{4.34}$$

And the angular correlation averaged over all triplets is:

$$AC_i(t) = \sum_{g=1}^{N_{triplets}} AC_{g,i}(t), \qquad i = 1, 2, 3 \tag{4.35}$$

where $N_{triplets}$ is the number of selected triplets.

Parameters

Pressing the **Angular Correlation** button will pop up the dialog shown on figure 4.38

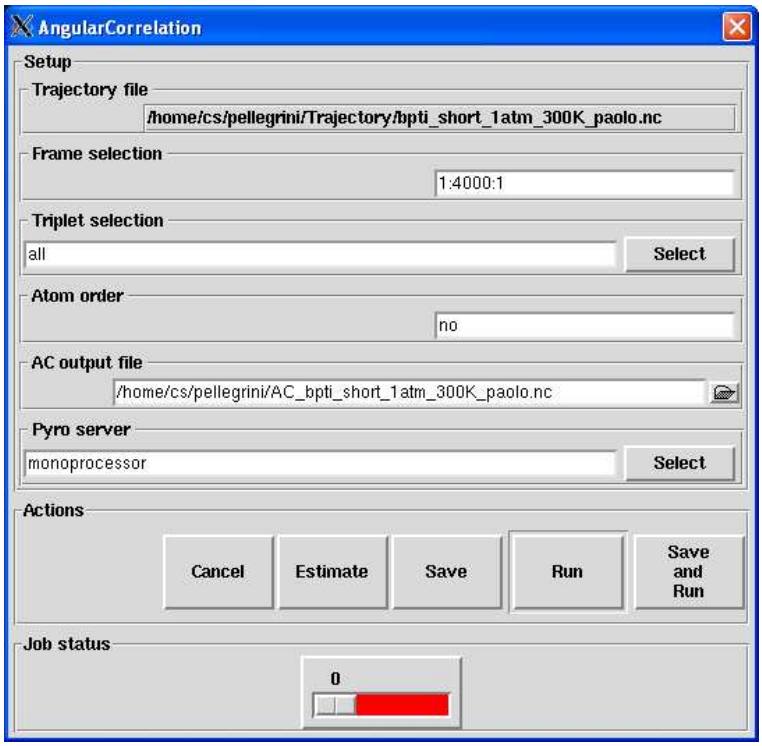

Figure 4.38: The dialog from where the AC analysis will be set up and run.

The following input fields controls the parameters for the **A**ngular **C**orrelation (AC) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - *2:10:3* will select the frames 2, 5 and 8.
  - *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Triplet selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the triplets of atoms from which the analysis will be performed. See Section 4.2.2.3 for more details. Any selection that does not contain exactly three atoms will be discarded.

- **Atom order**
  Format: string
  Default: *no*
  Description: this widget allows to specify the order in which the atoms **a1**, **a2** and **a3** should be ordered. By default, the order will be defined by *n*MOLDYN by ranking for the atoms of each triplet by their MMTK name. Otherwise, the entered value must have the following specific format:

  MMTK name for ***a1****,MMTK name for* ***a2****,MMTK name for* ***a3***

- **AC output file**
  Format: string
  Default: *AC_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *AC* analysis. A CDL version of the NetCDF output file is also automatically created with *AC_traj_file.cdl* name.

Output

The results of a *AC* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which the *AC* was evaluated,
- triplet: the index for each triplet considered in the analysis,
- ac_1_by_triplet: the angular correlation of $\mathbf{v}_1$ for each triplet according Eq.4.34,
- ac_2_by_triplet: the angular correlation of $\mathbf{v}_2$ for each triplet according Eq.4.34,
- ac_3_by_triplet: the angular correlation of $\mathbf{v}_3$ for each triplet according Eq.4.34,
- ac_1: the group-averaged angular correlation for $\mathbf{v}_1$ according Eq.4.35,
- ac_2: the group-averaged angular correlation for $\mathbf{v}_2$ according Eq.4.35,
- ac_3: the group-averaged angular correlation for $\mathbf{v}_3$ according Eq.4.35.

4.2.4.5 Velocity Autocorrelation Function

Theory and implementation

The **V**elocity **A**uto**C**orrelation **F**unction (*VACF*) is another interesting property describing the dynamics of a molecular system. Indeed, it reveals the underlying nature of the forces acting on the system.

In a molecular system that would be made of non interacting particles, the velocities would be constant at any time triggering the *VACF* to be a constant value. Now, if we think about a system with small interactions such as in a gas-phase, the magnitude and direction of the velocity of a particle will change gradually over time due to its collision with the other particles of the molecular system. In such a system, the *VACF* will be represented by a decaying exponential.

In the case of solid phase, the interaction are much stronger and, as a results, the atoms are bound to a given position from which they will move backwards and forwards oscillating between positive and negative values of their velocity. The oscillations will not be of equal magnitude however, but will decay in time, because there are still perturbative forces acting on the atoms to disrupt the perfection of their oscillatory motion. So, in that case the *VACF* will look like a damped harmonic motion.

Finally, in the case of liquid phase, the atoms have more freedom than in solid phase and because of the diffusion process, the oscillatory motion seen in solid phase will be cancelled quite rapidly depending on the density of the system. So, the *VACF* will just have one very damped oscillation before decaying to zero. This decaying time can be considered as the average time for a collision between two atoms to occur before they diffuse away.

Mathematically, the *VACF* of atom α in an atomic or molecular system is usually defined as

$$C_{vv;\alpha\alpha}(t) \doteq \frac{1}{3}\langle \mathbf{v}_\alpha(t_0) \cdot \mathbf{v}_\alpha(t_0 + t)\rangle_{t_0}. \tag{4.36}$$

In some cases, e.g. for non-isotropic systems, it is useful to define *VACF* along a given axis,

$$C_{vv;\alpha\alpha}(t;\mathbf{n}) \doteq \langle v_\alpha(t_0;\mathbf{n}) v_\alpha(t_0 + t;\mathbf{n})\rangle_{t_0}, \tag{4.37}$$

where $v_\alpha(t;\mathbf{n})$ is given by

$$v_\alpha(t;\mathbf{n}) \doteq \mathbf{n} \cdot \mathbf{v}_\alpha(t). \tag{4.38}$$

The vector $\boldsymbol{n}$ is a unit vector defining a space-fixed axis.

The *VACF* of the particles in a many body system can be related to the incoherent dynamic structure factor by the relation:

$$lim_{q\to 0}\frac{\omega^2}{q^2}\mathcal{S}(\mathbf{q},\omega) = G(\omega), \tag{4.39}$$

where $G(\omega)$ is the **D**ensity **O**f **S**tates (*DOS*). For an isotropic system it reads

$$G(\omega) = \sum_{\alpha} b^2_{\alpha,inc}\tilde{C}_{vv;\alpha\alpha}(\omega), \tag{4.40}$$

$$\tilde{C}_{vv;\alpha\alpha}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} dt\ \exp[-i\omega t]C_{vv;\alpha\alpha}(t). \tag{4.41}$$

For non-isotropic systems relation (4.39) holds if the *DOS* is computed from the atomic velocity autocorrelation functions $C_{vv;\alpha\alpha}(t;\mathbf{n}_q)$, where $\mathbf{n}_q$ is the unit vector in the direction of $\boldsymbol{q}$.

Parameters

Pressing the **Velocity Autocorrelation Function** button will pop up the dialog shown on figure 4.39

The following input fields controls the parameters for the *VACF* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Differentiation order**
  Format: integer in [0,5]
  Default: 0 if velocities are stored in the trajectory file, 1 otherwise
  Description: this widget allows to specify the order of the derivation scheme used to get

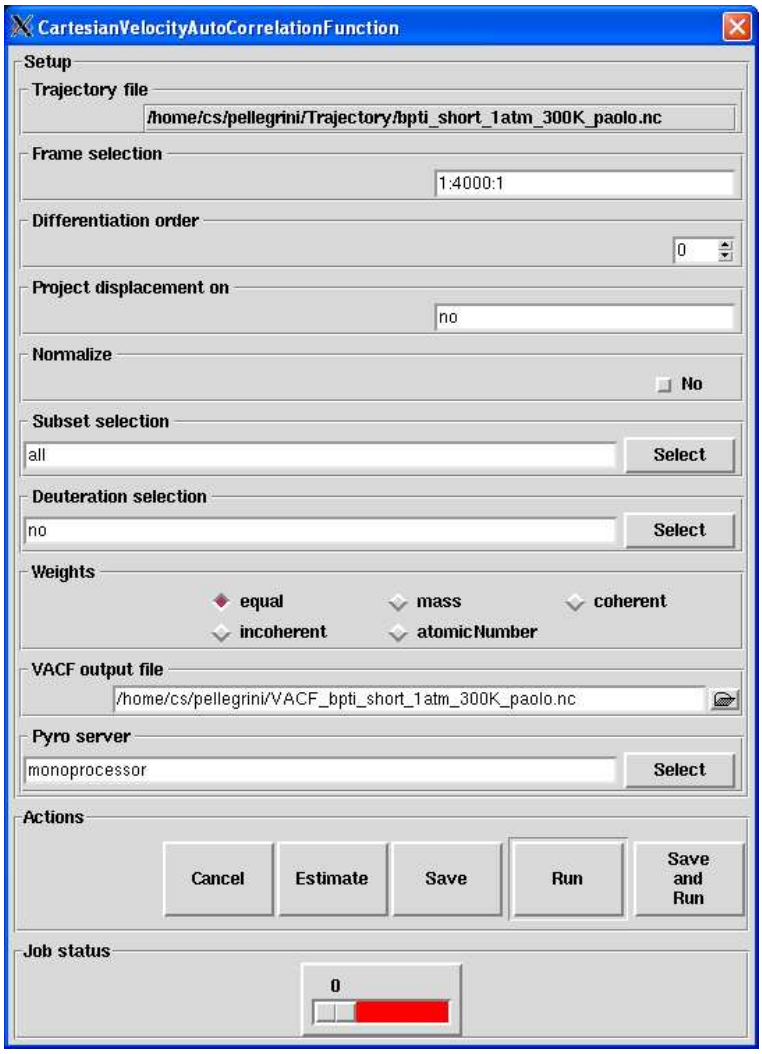

Figure 4.39: The dialog from where the *VACF* analysis will be set up and run.

the velocities out of the coordinates. If your trajectory NetCDF file already contains the velocities then just select *0*. However, you can still decide to get the velocities out of the coordinates. In that case, *n*MOLDYN performs a numerical differentiation of the input data. To do so, *n*MOLDYN can perform numerical differentiation from order *1* to order *5*. Using order *1*, the first time derivative of each point $r(t_i)$ is calculated as

$$\dot{r}(t_i) = \frac{r(t_{i+1}) - r(t_i)}{\Delta t}, \tag{4.42}$$

where Δt is the time step. Choosing order N with N=*2,...,5*, *n*MOLDYN calculates the first time-derivative of each point $r(t_i)$ ($r = x, y, z$) using the N-order polynomial interpolating the N+*1* points across $r(t_i)$, where $r(t_i)$ belongs to this set [52].

- **Project displacement on**
  Format: string
  Default: *no*
  Description: this widget allows to specify a vector along which the *VACF* will be computed. This vector does not need to be normalized as *n*MOLDYN will perform the normalization when processing it. The entered value must have the following format:

  vx:vy:vz

  where *vx*, *vy* and *vz* are floats that represent respectively the x, y and z coordinates of the vector.

- **Normalize**
  Format: string equal to *yes* or *no*
  Default: *no*
  Description: if set to *yes* normalize to 1 the $VACF(t = 0)$.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *equal*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *VACF*. See Section 4.2.1 for more details.

- **VACF output file**
  Format: string
  Default: *VACF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *VACF* analysis. A CDL version of the NetCDF output file is also automatically created with *VACF_traj_file.cdl* name.

Output

The results of a *VACF* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which the *VACF* was evaluated,
- vacf: the corresponding *VACF* in nm^2s^{-2}.

4.2.4.6 Density Of States

Theory and implementation

*n*MOLDYN calculates the power spectrum of the *VACF*, which in case of the mass-weighted *VACF* defines the phonon discrete *DOS*, (see Section 4.2.4.5) defined as:

$$DOS(n \cdot \Delta\nu) \doteq \sum_{\alpha} \omega_{\alpha} \tilde{C}_{vv;\alpha\alpha}(n \cdot \Delta\nu), \qquad n = 0 \ldots N_t - 1. \tag{4.43}$$

N_t is the total number of time steps and $\Delta\nu = 1/(2N_t\Delta t)$ is the frequency step. $DOS(n \cdot \Delta\nu)$ can be computed either for the isotropic case or with respect to a user-defined axis. The spectrum $DOS(n \cdot \Delta\nu)$ is computed from the *unnormalized* *VACF*, such that *DOS(0)* gives an approximate value for the diffusion constant $D = \sum_{\alpha} D_{\alpha}$ (see Eqs. 4.21 and 4.22). $DOS(n \cdot \Delta\nu)$ is smoothed by applying a Gaussian window in the time domain [76] (see Section A). Its width in the time domain is $\sigma_t = \alpha/T$, where T is the length of the simulation. We remark that the diffusion constant obtained from *DOS* is biased due to the spectral smoothing procedure since the *VACF* is weighted by this window Gaussian function. *n*MOLDYN computes the density of states starting from both atomic velocities and atomic coordinates. In this case the velocities are computed by numerical differentiation of the coordinate trajectories correcting first for possible jumps due to periodic boundary conditions.

Parameters

Pressing the **Density Of States** button will pop up the dialog shown on figure 4.40
The following input fields controls the parameters for the *DOS* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

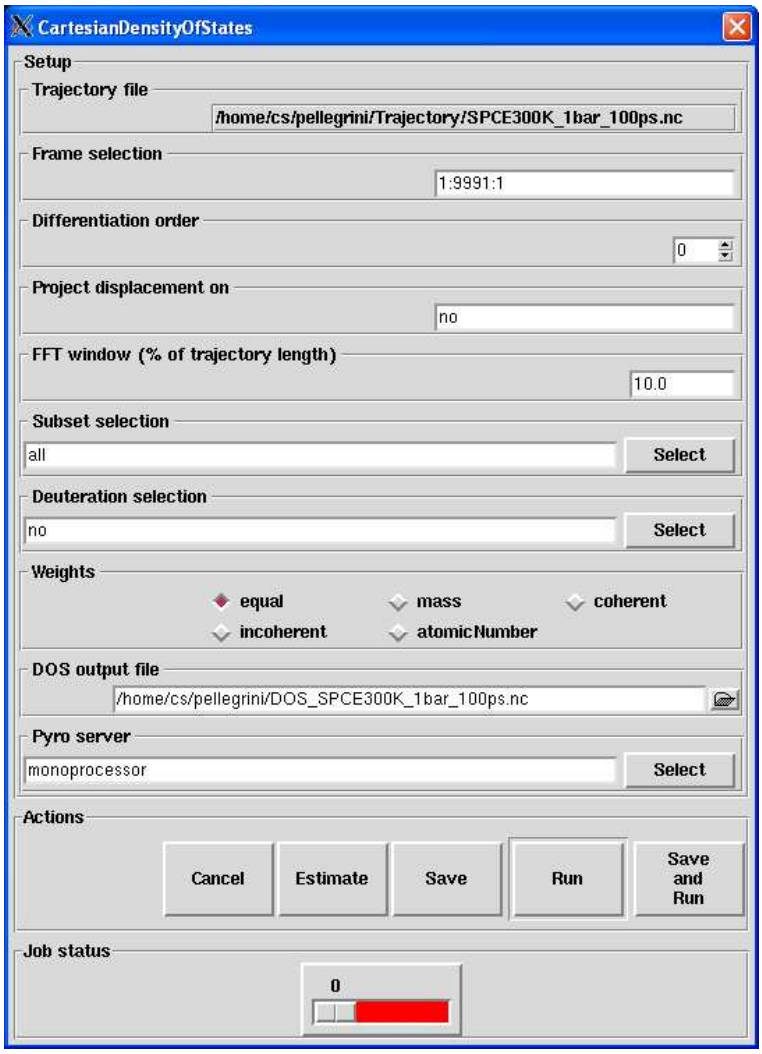

Figure 4.40: The dialog from where the *DOS* analysis will be set up and run.

⋆ *2:10:3* will select the frames 2, 5 and 8.

⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Differentiation order**
  Format: integer in [0,5]
  Default: 0 if velocities are stored in the trajectory file, 1 otherwise
  Description: this widget allows to specify the order of the derivation scheme used to get the velocities out of the coordinates. If your trajectory NetCDF file already contains the velocities then just select *0*. However, you can still decide to get the velocities out of the coordinates. In that case, *n*MOLDYN performs a numerical differentiation of the input data. To do so, *n*MOLDYN can perform numerical differentiation from order *1* to order *5*. Using order *1*, the first time derivative of each point $r(t_i)$ is calculated as

$$\dot{r}(t_i) = \frac{r(t_{i+1}) - r(t_i)}{\Delta t}, \tag{4.44}$$

where Δt is the time step. Choosing order N with N=*2,...,5*, nMOLDYN calculates the first time-derivative of each point $r(t_i)$ ($r = x, y, z$) using the N-order polynomial interpolating the N+*1* points across $r(t_i)$, where $r(t_i)$ belongs to this set [52].

- **Project displacement on**
  Format: string
  Default: *no*
  Description: this widget allows to specify a vector along which the *DOS* will be computed. This vector does not need to be normalized as nMOLDYN will perform the normalization when processing it. The entered value must have the following format:

  vx:vy:vz

  where *vx*, *vy* and *vz* are floats that represent respectively the x, y and z coordinates of the vector.

- **FFT window**
  Format: float in [0.0,100.0]
  Default: *10.0*
  Description: this widget allows to define the width in percentage of the trajectory length of the Gaussian function to be used in the smoothing procedure for the calculation of the *DOS*. See Appendix A for more details.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *equal*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *DOS*. See Section 4.2.1 for more details.

- **DOS Output file**
  Format: string

Default: *DOS_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
Description: this widget allows to enter the name of the NetCDF output file of the *DOS* analysis. A CDL version of the NetCDF output file is also automatically created with *DOS_traj_file.cdl* name.

Output
The results of a *DOS* analysis are stored in a NetCDF file whose main variables are namely:

- frequency: the frequencies in *THz* at which the *DOS* was evaluated,
- dos: the corresponding *DOS*.

4.2.4.7 Pass-Band Filtered Trajectory

Theory and implementation
It is often of interest to restrict attention to motions in a specific frequency interval, both for quantitative analysis and for visualization by animated display. This is particularly useful to study low-frequency motions without being distracted by the high frequency "noise". *n*MOLDYN can create **P**ass-**B**and **F**iltered **T**rajectory (*PBFT*) by applying a frequency pass-band filter to the atomic trajectories, either on the whole system or on an user-defined subset. The result is stored in a new trajectory file that contains only motions in the chosen interval. Frequency filtering uses a straightforward algorithm:

- take the discrete Fourier transform of each particle trajectory,
- set the Fourier coefficients outside the filtering interval to zero,
- and transform back to the time domain.

This corresponds to applying a rectangular window in the frequency domain.

Parameters
Pressing the **Pass-Band Filtered Trajectory** button will pop up the dialog shown on figure 4.41
The following input fields controls the parameters for the *PBFT* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

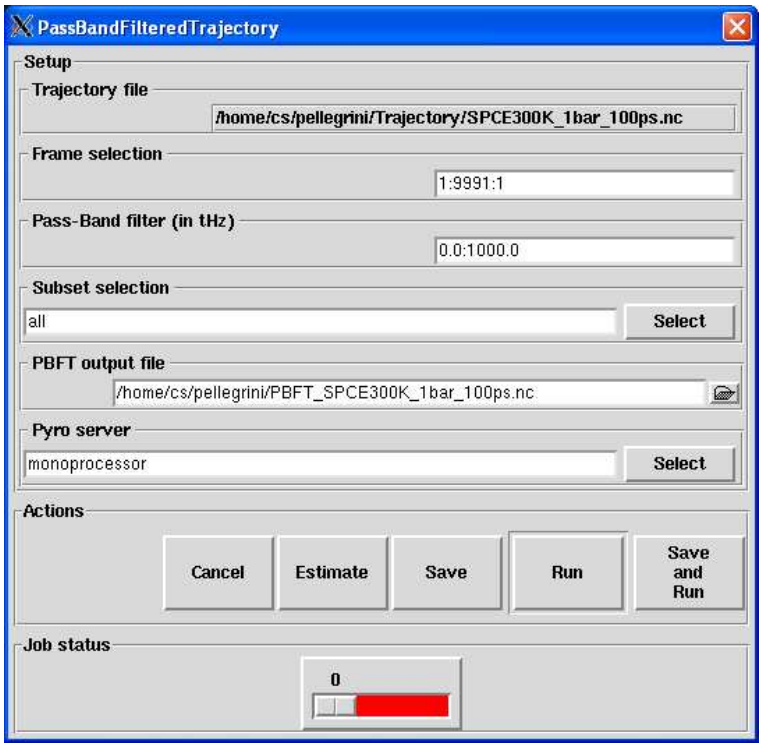

Figure 4.41: The dialog from where the *PBFT* analysis will be set up and run.

where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Pass-band filter**
  Format: string
  Default: *0.0:1000.0*
  Description: this widget allows to specify respectively the minimum and maximum frequencies for the pass-band filter. The entered value must have the following format:

  fmin:fmax

  where *fmin* and *fmax* are respectively the minimum and maximum frequencies in *THz* of the pass-band filter.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **PBFT Output file**
  Format: string
  Default: *PBFT_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *PBFT* analysis.

Output

The results of a *PBFT* analysis is a MMTK NetCDF trajectory file containing the pass-band filtered trajectory.

4.2.4.8 Global Motion Filtered Trajectory

Theory and implementation

It is often of interest to separate global motion from internal motion, both for quantitative analysis and for visualization by animated display. Obviously, this can be done under the hypothesis that global and internal motions are decoupled within the length and timescales of the analysis. *n*MOLDYN can create **G**lobal **M**otion **F**iltered **T**rajectory (*GMFT*) by filtering out global motions (made of the three translational and rotational degrees of freedom), either on the whole system or on an user-defined subset, by fitting it to a reference structure (usually the first frame of the MD). Global motion filtering uses a straightforward algorithm:

- for the first frame, find the linear transformation such that the coordinate origin becomes the center of mass of the system and its principal axes of inertia are parallel to the three coordinates axes (also called principal axes transformation),
- this provides a reference configuration $\mathcal{C}_{ref}$,
- for any other frames f, finds and applies the linear transformation that minimizes the RMS distance between frame f and $\mathcal{C}_{ref}$.

The result is stored in a new trajectory file that contains only internal motions. This analysis can be useful in case where diffusive motions are not of interest or simply not accessible to the experiment (time resolution, powder analysis ...).

Parameters

Pressing the **Global Motion Filtered Trajectory** button will pop up the dialog shown on figure 4.42

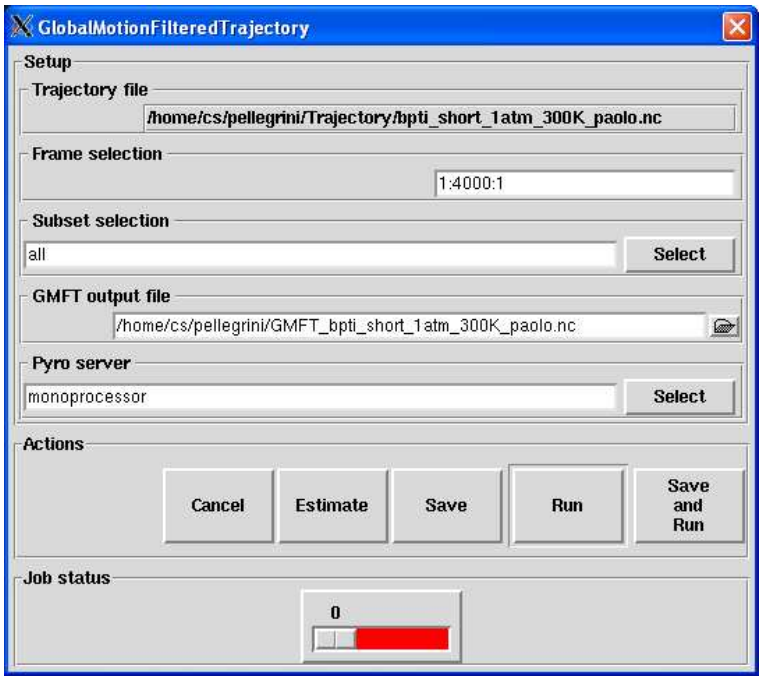

Figure 4.42: The dialog from where the *GMFT* analysis will be set up and run.

The following input fields controls the parameters for the *GMFT* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **GMFT Output file**
  Format: string
  Default: *GMFT_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *GMFT* analysis.

Output

The results of a *GMFT* analysis is a MMTK NetCDF trajectory file containing the trajectory with the global motion filtered out.

4.2.4.9 Rigid-Body Trajectory

Theory and implementation

To analyze the dynamics of complex molecular systems it is often desirable to consider the overall motion of molecules or molecular subunits. We will call this motion rigid-body motion in the following. Rigid-body motions are fully determined by the dynamics of the centroid, which may be the center-of-mass, and the dynamics of the angular coordinates describing the orientation of the rigid body. The angular coordinates are the appropriate variables to compute angular correlation functions of molecular systems in space and time. In most cases, however, these variables are not directly available from MD simulations since MD algorithms typically work in cartesian coordinates. Molecules are either treated as flexible, or, if they are treated as rigid, constraints are taken into account in the framework of cartesian coordinates [54]. In *n*MOLDYN, **R**igid-**B**ody **T**rajectory (*RBT*) can be defined from a MD trajectory by fitting rigid reference structures, defining a (sub)molecule, to the corresponding structure in each time frame of the trajectory. Here 'fit' means the optimal superposition of the structures in a least-squares sense. We will describe now how rigid body motions, i.e. global translations and rotations of molecules or subunits of complex molecules, can be extracted from a MD trajectory. A more detailed presentation is given in [55]. We define an optimal rigid-body trajectory in the following way: for each time frame of the trajectory the atomic positions of a rigid reference structure, defined by the three cartesian components of its centroid (e.g. the center of mass) and three angles, are as close as possible to the atomic positions of the corresponding structure in the MD configuration. Here 'as close as possible' means as close as possible in a least-squares sense.

Optimal superposition. We consider a given time frame in which the atomic positions of a (sub)molecule are given by $\mathbf{x}_\alpha$, $\alpha = 1 \ldots N$. The corresponding positions in the reference structure are denoted as $\mathbf{x}_\alpha^{(0)}$, $\alpha = 1 \ldots N$. For both the given structure and the reference structure we introduce the yet undetermined centroids $\mathbf{X}$ and $\mathbf{X}^{(0)}$, respectively, and define the deviation

$$\mathbf{\Delta}_\alpha \doteq \mathbf{D}(\mathbf{q}) \left[\mathbf{x}_\alpha^{(0)} - \mathbf{X}^{(0)}\right] - [\mathbf{x}_\alpha - \mathbf{X}]\,. \tag{4.45}$$

Here $\mathbf{D}(\mathbf{q})$ is a rotation matrix which depends on also yet undetermined angular coordinates which we chose to be *quaternion parameters*, abbreviated as vector $\mathbf{q} = (q_0, q_1, q_2, q_3)$. The

quaternion parameters fulfill the normalization condition $\mathbf{q} \cdot \mathbf{q} = 1$ [56]. The target function to be minimized is now defined as

$$m(\mathbf{q}; \mathbf{X}, \mathbf{X}^{(0)}) = \sum_{\alpha} \omega_{\alpha} |\mathbf{\Delta}|_{\alpha}^{2}. \tag{4.46}$$

where ω_{α} are atomic weights (see Section 4.2.1). The minimization with respect to the centroids is decoupled from the minimization with respect to the quaternion parameters and yields

$$\mathbf{X} = \sum_{\alpha} \omega_{\alpha} \mathbf{x}_{\alpha}, \tag{4.47}$$

$$\mathbf{X}^{(0)} = \sum_{\alpha} \omega_{\alpha} \mathbf{x}_{\alpha}^{(0)}. \tag{4.48}$$

We are now left with a minimization problem for the rotational part which can be written as

$$m(\mathbf{q}) = \sum_{\alpha} \omega_{\alpha} \left[\mathbf{D}(\mathbf{q}) \mathbf{r}_{\alpha}^{(0)} - \mathbf{r}_{\alpha} \right]^{2} \stackrel{!}{=} Min. \tag{4.49}$$

The relative position vectors

$$\mathbf{r}_{\alpha} = \mathbf{x}_{\alpha} - \mathbf{X}, \tag{4.50}$$

$$\mathbf{r}_{\alpha}^{(0)} = \mathbf{x}_{\alpha}^{(0)} - \mathbf{X}^{(0)}, \tag{4.51}$$

are fixed and the rotation matrix reads [56]

$$\mathbf{D}(\mathbf{q}) = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(-q_0 q_3 + q_1 q_2) & 2(q_0 q_2 + q_1 q_3) \\ 2(q_0 q_3 + q_1 q_2) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(-q_0 q_1 + q_2 q_3) \\ 2(-q_0 q_2 + q_1 q_3) & 2(q_0 q_1 + q_2 q_3) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{pmatrix}. \tag{4.52}$$

Quaternions and rotations. The rotational minimization problem can be elegantly solved by using quaternion algebra. Quaternions are so-called hypercomplex numbers, having a real unit, **1**, and three imaginary units, **I**, **J**, and **K**. Since $\mathbf{IJ} = \mathbf{K}$ (cyclic), quaternion multiplication is not commutative. A possible matrix representation of an arbitrary quaternion,

$$\mathbf{A} = a_0 \cdot \mathbf{1} + a_1 \cdot \mathbf{I} + a_2 \cdot \mathbf{J} + a_3 \cdot \mathbf{K}, \tag{4.53}$$

reads

$$\mathbf{A} = \begin{pmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{pmatrix}. \tag{4.54}$$

The components a_{ν} are real numbers. Similarly as normal complex numbers allow one to represent rotations in a plane, quaternions allow one to represent rotations in space. Consider the quaternion representation of a vector $\mathbf{r}$, which is given by

$$\mathbf{R} = x \cdot \mathbf{I} + y \cdot \mathbf{J} + z \cdot \mathbf{K}, \tag{4.55}$$

and perform the operation

$$\mathbf{R}' = \mathbf{Q}\mathbf{R}\mathbf{Q}^{T}, \tag{4.56}$$

where $\mathbf{Q}$ is a normalized quaternion,

$$\|\mathbf{Q}\|^2 \doteq q_0^2 + q_1^2 + q_2^2 + q_3^2 = \frac{1}{4} tr\{\mathbf{Q}^T \mathbf{Q}\} = 1. \tag{4.57}$$

The symbol *tr* stands for 'trace'. We note that a normalized quaternion is represented by an *orthogonal* 4×4 matrix. $\mathbf{R}'$ may then be written as

$$\mathbf{R}' = x' \cdot \mathbf{I} + y' \cdot \mathbf{J} + z' \cdot \mathbf{K}, \tag{4.58}$$

where the components x', y', z', abbreviated as $\mathbf{r}'$, are given by

$$\mathbf{r}' = \mathbf{D}(\mathbf{q})\mathbf{r}. \tag{4.59}$$

The matrix $\mathbf{D}(\mathbf{q})$ is the rotation matrix defined in (4.52).

Solution of the minimization problem. In quaternion algebra, the rotational minimization problem may now be phrased as follows:

$$m(\mathbf{q}) = \sum_\alpha \omega_\alpha \|\mathbf{Q}\mathbf{R}_\alpha^{(0)}\mathbf{Q}^T - \mathbf{R}_\alpha\|^2 \stackrel{!}{=} Min. \tag{4.60}$$

Since the matrix $\mathbf{Q}$ representing a normalized quaternion is orthogonal this may also be written as

$$m(\mathbf{q}) = \sum_\alpha \omega_\alpha \|\mathbf{Q}\mathbf{R}_\alpha^{(0)} - \mathbf{R}_\alpha\mathbf{Q}\|^2. \stackrel{!}{=} Min. \tag{4.61}$$

This follows from the simple fact that $\|\mathbf{A}\| = \|\mathbf{A}\mathbf{Q}\|$, if $\mathbf{Q}$ is normalized. Eq. (4.61) shows that the target function to be minimized can be written as a simple quadratic form in the quaternion parameters [55],

$$\begin{aligned} m(\mathbf{q}) &= \mathbf{q} \cdot \mathbf{M}\mathbf{q}, & (4.62)\\ \mathbf{M} &= \sum_\alpha \omega_\alpha \mathbf{M}_\alpha. & (4.63) \end{aligned}$$

The matrices $\mathbf{M}_\alpha$ are positive semi-definite matrices depending on the positions $\mathbf{r}_\alpha$ and $\mathbf{r}_\alpha^{(0)}$:

$$\left.\begin{aligned} M_{\alpha,11} &= x_\alpha^2 + y_\alpha^2 + z_\alpha^2 + x_{0\alpha}^2 + y_{0\alpha}^2 + z_{0\alpha}^2 - 2x_\alpha x_{0\alpha} - 2y_\alpha y_{0\alpha} - 2z_\alpha z_{0\alpha}\\ M_{\alpha,12} &= 2(y_\alpha z_{0\alpha} - z_\alpha y_{0\alpha})\\ M_{\alpha,13} &= 2(-x_\alpha z_{0\alpha} + z_\alpha x_{0\alpha})\\ M_{\alpha,14} &= 2(x_\alpha y_{0\alpha} - y_\alpha x_{0\alpha})\\ M_{\alpha,22} &= x_\alpha^2 + y_\alpha^2 + z_\alpha^2 + x_{0\alpha}^2 + y_{0\alpha}^2 + z_{0\alpha}^2 - 2x_\alpha x_{0\alpha} + 2y_\alpha y_{0\alpha} + 2z_\alpha z_{0\alpha}\\ M_{\alpha,23} &= -2(x_\alpha y_{0\alpha} + y_\alpha x_{0\alpha})\\ M_{\alpha,24} &= -2(x_\alpha z_{0\alpha} + z_\alpha x_{0\alpha})\\ M_{\alpha,33} &= x_\alpha^2 + y_\alpha^2 + z_\alpha^2 + x_{0\alpha}^2 + y_{0\alpha}^2 + z_{0\alpha}^2 + 2x_\alpha x_{0\alpha} - 2y_\alpha y_{0\alpha} + 2z_\alpha z_{0\alpha}\\ M_{\alpha,44} &= -2(y_\alpha z_{0\alpha} + z_\alpha y_{0\alpha})\\ M_{\alpha,44} &= x_\alpha^2 + y_\alpha^2 + z_\alpha^2 + x_{0\alpha}^2 + y_{0\alpha}^2 + z_{0\alpha}^2 + 2x_\alpha x_{0\alpha} + 2y_\alpha y_{0\alpha} - 2z_\alpha z_{0\alpha} \end{aligned}\right\} \tag{4.64}$$

The rotational fit is now reduced to the problem of finding the minimum of a quadratic form with the constraint that the quaternion to be determined must be normalized. Using the method of Lagrange multipliers to account for the normalization constraint we have

$$m'(\mathbf{q}, \lambda) = \mathbf{q} \cdot \mathbf{M}\mathbf{q} - \lambda(\mathbf{q} \cdot \mathbf{q} - 1) \stackrel{!}{=} Min. \tag{4.65}$$

This leads immediately to the eigenvalue problem

$$\begin{aligned} \mathbf{M}\mathbf{q} &= \lambda\mathbf{q}, & (4.66)\\ \mathbf{q} \cdot \mathbf{q} &= 1. & (4.67) \end{aligned}$$

Now any normalized eigenvector $\mathbf{q}$ fulfills the relation $\lambda = \mathbf{q} \cdot \mathbf{M}\mathbf{q} \equiv m(\mathbf{q})$. Therefore the eigenvector belonging to the smallest eigenvalue, λ_{min}, is the desired solution. At the same time λ_{min} gives the average error per atom.

The result of *RBT* analysis is stored in a new trajectory file that contains only *RBT* motions.

Parameters

Pressing the **Rigid-Body Trajectory** button will pop up the dialog shown on figure 4.43

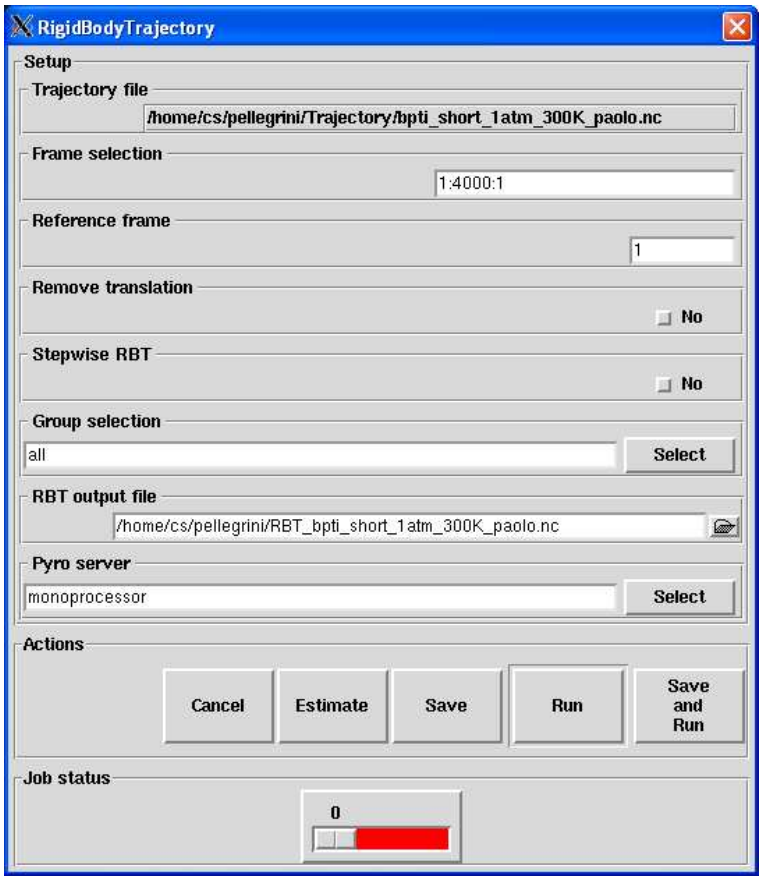

Figure 4.43: The dialog from where the *RBT* analysis will be set up and run.

The following input fields controls the parameters for the *RBT* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an inte-

ger specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Reference frame**
  Format: integer in [1,*traj_length*] where *traj_length* is the number of frames of the input trajectory
  Default: *1*
  Description: this widget allows to specify which frame should be the reference for the *RBT* analysis. The value entered should be an integer ranging from 1 to *traj_length* where *traj_length* is the number of rames of the input trajectory.

- **Remove translation**
  Format: string equal to *yes* or *no*
  Default: *no*
  Description: if set to *yes* the translation motion will be removed from the *RBT*.

- **Stepwise RBT**
  Format: string equal to *yes* or *no*
  Default: *no*
  Description: if set to *yes*, each frame *f* will serve as the reference for the frame *f+1* when defining the *RBT* canceling the value entred in **Reference frame** entry.

- **Group selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the groups of atoms that will be defined as rigid-bodies when performing the *RBT*. See Section 4.2.2.3 for more details.

- **RBT output file**
  Format: string
  Default: *RBT_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *RBT* analysis.

Output

The results of a *RBT* analysis is a MMTK NetCDF trajectory file containing the rigid-body trajectory. Beside the trajectory, other NetCDF variables are stored in this file namely:

- groupi, i = 1, 2, ... N_{groups}: the MMTK indexes of each atom of groups 1, 2, ... N_{groups} where N_{groups} is the number of groups selected for the analysis,

- quaternion: a matrix of dimension (N_{Frames}, N_{groups}, 4) where N_{Frames} is the number of selected frames for the *RBT* analysis. This matrix contains the quaternions produced by the *RBT* analysis at each frame and for each rigid-body,
- cms: a matrix of dimension (N_{Frames}, N_{groups}, 3) that contains the coordinates of the center of mass of each rigid-body for each frame,
- fit: a matrix of dimension (N_{Frames}, N_{groups}) that contains the results of the fit produced by the *RBT* fitting procedure at each frame and for each rigid-body.

4.2.4.10 Center Of Mass Trajectory

Theory and implementation

The **C**enter **O**f **M**ass **T**rajectory (*COMT*) analysis consists in deriving the trajectory of the respective centers of mass of a set of groups of atoms. In order to produce a visualizable trajectory, *n*MOLDYN assigns the centers of mass to pseudo-hydrogen atoms whose mass is equal to the mass of their associated group. Thus, the produced trajectory can be reused for other analysis. In that sense, *COMT* analysis is a practical way to reduce noticeably the dimensionality of a system.

Parameters

Pressing the **Center Of Mass Trajectory** button will pop up the dialog shown on figure 4.44

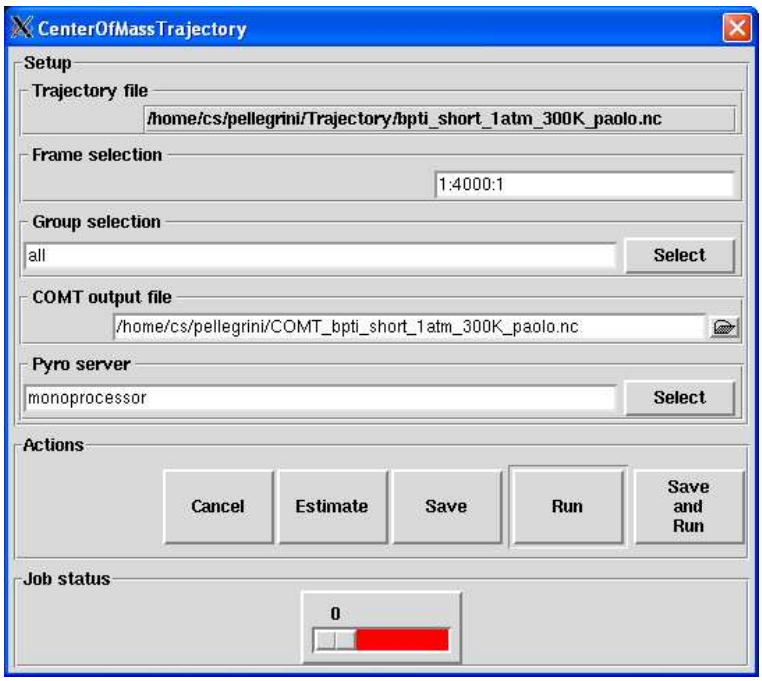

Figure 4.44: The dialog from where the *COMT* analysis will be set up and run.

The following input fields controls the parameters for the *COMT* analysis:

- **Trajectory file**
  Format: string

Default: *traj_file* where *traj_file* is the name of the loaded trajectory
Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - *2:10:3* will select the frames 2, 5 and 8.
  - *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Group selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the groups of atoms whose trajectories of their centers of mass will define the *COMT*. See Section 4.2.2.3 for more details.

- **COMT output file**
  Format: string
  Default: *COMT_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *COMT* analysis.

Output

The results of a *COMT* analysis is a MMTK NetCDF trajectory file containing the trajectory of the centers of mass of each selected group of atoms.

4.2.4.11 Auto-Regressive Analysis

Theory and implementation

The concept of **A**uto-**R**egressive **A**nalysis (*ARA*) analysis is intimitely related to the one of memory function. Memory functions have been used for a long time in theoretical statistical physics to describe the time dependence of autocorrelation functions. Nevertheless, the use of memory functions in the context of MD simulations has been hindered by the lack of a suitable numerical algorithm for their calculation. Such an algorithm has been published and is now implemented in *n*MOLDYN [57]. The key point is that a reliable estimates for memory

functions can be obtained by assuming an **A**uto-**R**egressive (*AR*) model for the underlying stochastic process and not for the memory function itself.

To compute the memory function $\xi(t)$ from a discrete time serie $x(n) \equiv x(n\Delta t)$ the latter is modelled by an *autoregressive stochastic process* of order P [58, 59],

$$x(t) = \sum_{n=1}^{P} a_n^{(P)} x(t - n\Delta t) + \epsilon_P(t). \tag{4.68}$$

Here $\epsilon_P(t)$ is *white noise* with zero mean and amplitude σ_P. The coeffients $\{a_n^{(P)}\}$ are fitted to the discrete time serie using Burg's algorithm [60, 61], and σ_P is given by

$$\sigma_P^2 = r(0) - \sum_{n=1}^{P} a_n^{(P)}(n\Delta t), \tag{4.69}$$

where $r(t)$ is the autocorrelation function of $x(t)$

$$r(t) := \langle x(t)x(0)\rangle. \tag{4.70}$$

In all following calculations nMOLDYN works with a set of coefficients $\{a_n\}$ which has been averaged over all selected atoms and the three Cartesian coordinates.

VACF within the AR model The autocorrelation function $r(t)$ introduced in the previous Section is here the *normalized VACF*

$$VACF(t) := \frac{\langle v(t)v(0)\rangle}{\langle v^2(0)\rangle} \tag{4.71}$$

hence $r(0) = VACF(0) = 1$. Here $v(t)$ is the x-, y-, or z-component of the velocity of a 'tagged' atom. The memory function $\xi(t)$ of $\psi(t)$ is defined by the relation

$$\frac{d}{dt} VACF(t) = -\int_0^t d\tau\, \xi(t-\tau)\, VACF(\tau). \tag{4.72}$$

Eq. (4.72) is called the *memory function equation*.

Within the AR-model the z-transform of the *VACF* has the form

$$\mathrm{VACF}^{(AR)}(z) = \frac{1}{a_P^{(P)}} \frac{-z^P \sigma_P^2}{\prod_{k=1}^{P}(z - z_k)\, \prod_{l=1}^{P}(z - z_l^{-1})}. \tag{4.73}$$

Here the $\{z_k\}$ are the zeros of

$$p(z) = z^P - \sum_{k=1}^{P} a_k^{(P)} z^{P-k}. \tag{4.74}$$

We recall that the z-transform of an arbitrary discrete function $f(n)$ is given by $F(z) = \sum_{n=-\infty}^{+\infty} f(n) z^{-n}$, and the inverse transform by $f(n) = \frac{1}{2\pi i} \oint_C dz\, z^{n-1} F(z)$. Applying the inverse z-transform to (4.73) yields

$$VACF^{(AR)}(n) = \sum_{j=1}^{P} \beta_j z_j^{|n|}, \tag{4.75}$$

where the coefficients β_j are given by

$$\beta_j = \frac{1}{a_P} \frac{-z_j^{P-1}\sigma_P^2}{\prod_{k=1,k\neq j}^{P}(z_j - z_k)\,\prod_{l=1}^{P}(z_j - z_l^{-1})}. \tag{4.76}$$

Note that $VACF^{(AR)}(n)$ has a multiexponential form, and that the stability criterion

$$|z_j| < 1, \quad j = 1, \ldots, P, \tag{4.77}$$

must be fulfilled. This is guaranteed by the Burg-algorithm [60, 61].

Density of states within the AR model Evaluating $\mathrm{VACF}^{(AR)}(z)$ as given by (4.73) at $z = \exp(i\omega\Delta t)$ yields the density of states within the AR model:

$$DOS^{(AR)}(\omega) = \frac{\Delta t}{2}\frac{k_B T}{M}\mathrm{VACF}^{(AR)}\left(\exp[i\omega\Delta t]\right). \tag{4.78}$$

Here M is the mass of the tagged atom, k_B is the Boltzmann constant, and T the temperature. Note that the *VACF* and the density of states within the *AR* model are entirely determined by the coefficients $a_n^{(P)}$.

Discrete memory function of the VACF within the AR model Starting from the memory function equation of the *VACF* (Eq. 4.72), the first step towards a numerical computation of the memory function consists in discretizing Eq. 4.72

$$\frac{VACF(n+1) - VACF(n)}{\Delta t} = -\sum_{k=0}^{n-1} \Delta t\, \xi(n-k)\psi(k), \tag{4.79}$$

Eq. (4.79) is now subjected to a *one-sided* z-transform. Using that

$$Z_>\left\{f(n+1) - f(n)\right\} = zF_>(z) - zf(0) \tag{4.80}$$

for any discrete function $f(n)$ whose one-sided z-transform exists, one obtains from (4.79)

$$\Xi_>(z) = \frac{1}{\Delta t^2}\left(\frac{z}{\mathrm{VACF}_>(z)} + 1 - z\right), \tag{4.81}$$

using that $VACF(0) = 1$. The one-sided z-transform of an arbitrary discrete function $f(n)$ is defined as $F_>(z) = \sum_{n=0}^{\infty} f(n) z^{-n}$. Here it has been used that the one-sided z-transform of the discrete convolution integral is just the product $\Xi_>(z)\mathrm{VACF}_>(z)$. Inserting the definition of the one-sided z-transform for $\Xi_>(z)$ and $\mathrm{VACF}_>(z)$, this equation can be rewritten as

$$\sum_{j=0}^{\infty} \xi(j)\, z^{-j} = \frac{1}{\Delta t^2} \frac{\sum_{j=0}^{\infty}\left[VACF(j) - VACF(j+1)\right] z^{-j}}{\sum_{j=0}^{\infty} VACF(j)\, z^{-j}}. \tag{4.82}$$

Note that the term proportional to z cancels out. The time dependent memory function is, in principle, obtained by comparing the coefficients of the series on the *lhs* and the *rhs* of Eq. (4.82). To construct a numerical method, we replace the series by polynomials of order N, where $t = N\Delta t$ defines the time window for the memory function to be computed. After this first step a polynomial division is performed on the *rhs* of Eq. (4.82), and after a subsequent

multiplication of both sides with z^{-N} one obtains the time dependent memory function, $\xi(j)$, by comparison of coefficients,

$$\begin{aligned} \frac{z^{-N}}{\Delta t^2} \frac{\sum_{j=0}^{N} \left[VACF(j) - VACF(j+1)\right] z^{N-j}}{\sum_{j=0}^{N} VACF(j)\, z^{-j}} &= z^{-N} \left(\sum_{j=0}^{N} c_j\, z^{N-j} + R \right) \\ &= \sum_{j=0}^{N} \xi(j)\, z^{-j}. \end{aligned} \tag{4.83}$$

Within nMOLDYN $VACF(n)$ is replaced by the autocorrelation function calculated in the framework of the autoregressive model, $VACF^{(AR)}(n)$, as in Eqs. 4.75 and 4.76. The coefficients c_j are obtained by polynomial division and R is a rest which does not contain information on the memory function within the time interval $t \in [0, N\Delta t]$. The discrete memory function is therefore given by $\xi(j) = c_{N-j}$.

A remark concerning the discretization scheme (4.79) is in place here. The discrete convolution sum is effectively a first order approximation of the convolution integral. More sophisticated approximations could be used, but they would lead to less convenient expressions upon z-transformation. Correspondingly, we have chosen a first order approximation for the differentiation on the left-hand side of (4.72). In this way the first order (integro-)differential equation (4.72) is transformed into the first order difference equation (4.79).

However, this simple discretization scheme together with the use of the one-sided z-transform leads to a significant error in $\xi(0)$. It is clear from Eq. (4.72) that due to the symmetry of the autocorrelation function ($\psi(t) = \psi(-t)$), the derivative $d\psi/dt$ should vanish at $t = 0$. However, in the discretized version it is approximated by a forward difference that is always negative. A higher-order calculation shows that the estimate for $\xi(0)$ that results from the procedure described above should be doubled.

MSD within the AR model The relation between the *VACF* and the *MSD* reads

$$MSD(t) = \langle [x(t) - x(0)]^2 \rangle = 2 \int_0^t d\tau\, (t - \tau) C_{vv}(t) \tag{4.84}$$

By discretizing the above equation one obtains

$$MSD(n) = 2 \sum_{k=0}^{n} \Delta t^2 (n - k) C_{vv}(k). \tag{4.85}$$

Using

$$f_1(n) = \Theta(n) \cdot n f_2(n) = \Theta(n) \cdot C_{vv}(n) \tag{4.86}$$

into Eq. 4.85, gives

$$MSD(n) = 2 \sum_{k=-\infty}^{+\infty} \Delta t^2 f_1(n - k) f_2(k) \tag{4.87}$$

Making use of the one-side z-transform (equivalent to the Laplace transform for discrete functions), we obtain

$$MSD^{>}(z) = 2 F_1^{>} F_2^{>} \Delta t^2, \tag{4.88}$$

where

$$F_1^{>} = \frac{z}{(z-1)^2}. \tag{4.89}$$

Introducing Eq. 4.89 into Eq. 4.88 yields

$$MSD^{>}(z) = 2\frac{z\Delta t^2}{(z-1)^2} C^{>}_{vv}(z) \tag{4.90}$$

and its inverse z-transform reads

$$MSD(n) = 2\Delta t^2 \cdot \frac{1}{2\pi i} \oint dz z^{n-1} \cdot \frac{z}{(z-1)^2} \cdot C^{>}_{vv}(z). \tag{4.91}$$

Using the expression of the non-normalized $C^{\rangle}_{vv}(z)$ obtained in the framework of the *AR* model

$$C^{>}_{vv}(z) = \sum_{j=1}^{P} \beta_j \frac{z}{z - z_j} \cdot \langle v^2 \rangle \tag{4.92}$$

one finds the expression of *MSD* within the same *AR* model

$$MSD^{AR}(n) = 2\Delta t^2 \langle v^2 \rangle \sum_{j=1}^{P} \beta_j \frac{1}{2\pi i} \oint dz \frac{z^{n+1}}{(z-1)^2} \cdot \frac{1}{z - z_j} \tag{4.93}$$

$$= 2\Delta t^2 \langle v^2 \rangle \sum_{j=1}^{P} \beta_j \{ \frac{n}{1 - z_j} - \frac{z_j}{(1-z_j)^2} + \frac{z_j^{n+1}}{(1-z_j)^2} \}. \tag{4.94}$$

$$MSD^{AR}(n) \overset{n \to +\infty}{\simeq} 2Dn\Delta t \; ; \; D = \Delta t \langle v^2 \rangle \sum_{j=1}^{P} \frac{\beta_j}{1 - z_j} = \frac{\langle v^2 \rangle}{\gamma} \tag{4.95}$$

which allows one to compute the *MSD* within the *AR* model from the poles and the β_j coefficients of the non-normalized *VACF*.

Friction coefficient within the AR model The friction coefficient is defined as the integral over the memory function. In the discrete case we write

$$\xi_0 := \sum_{n=0}^{\infty} \Delta t \, \xi(n) = \Delta t \, \Xi_{>}(1). \tag{4.96}$$

As shown in [57], the *AR* model allows us to express $\Psi_{>}(z)$ as

$$\mathrm{VACF}^{(AR)}_{>}(z) = \sum_{n=0}^{\infty} VACF^{(AR)}(n) z^{-n} = \sum_{j=1}^{P} \beta_j \frac{z_j}{z - z_j}, \tag{4.97}$$

where the coefficients β_j are given by eq. (4.76), and the roots z_j must fulfill the stability criterion (4.77). Inserting (4.97) into (4.81) yields $\Xi^{(AR)}_{>}(z)$, the z-transform of the discrete memory function within the *AR* model,

$$\Xi^{(AR)}_{>}(z) = \frac{1}{\Delta t^2} \left(\frac{z}{\mathrm{VACF}^{(AR)}_{>}(z)} + 1 - z \right). \tag{4.98}$$

Using (4.98) we obtain thus within the *AR* model

$$\xi_0^{(AR)} = \frac{1}{\Delta t} \frac{1}{\sum_{j=1}^{P} \beta_j \frac{1}{1 - z_j}}. \tag{4.99}$$

This shows that ξ_0 can be obtained from the zeros z_j of the characteristic polynomial $p(z)$, defined in (4.74).

In the framework of the autoregressive model, *n*MOLDYN allows one to calculate the *VACF*, the *VACF* memory function, the *DOS*, the *MSD*, and the *AR* coefficients $a_n^{(P)}$ of the velocity trajectory, averaged over all selected atoms and three Cartesian coordinates.

Parameters

Pressing the **Auto-Regressive Analysis** button will pop up the dialog shown on figure 4.45

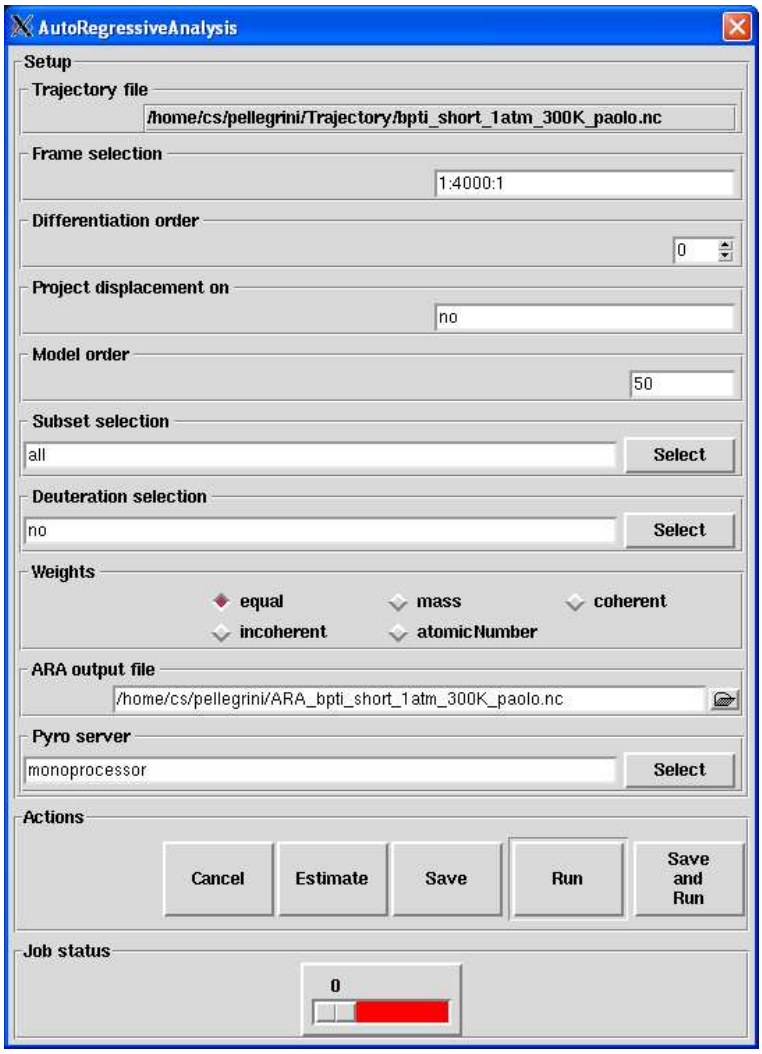

Figure 4.45: The dialog from where the *ARA* analysis will be set up and run.

The following input fields controls the parameters for the *ARA* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.

Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

first:last:step

where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Differentiation order**
  Format: integer in [0,5]
  Default: 0 if velocities are stored in the trajectory file, 1 otherwise
  Description: this widget allows to specify the order of the derivation scheme used to get the velocities out of the coordinates. If your trajectory NetCDF file already contains the velocities then just select *0*. However, you can still decide to get the velocities out of the coordinates. In that case, *n*MOLDYN performs a numerical differentiation of the input data. To do so, *n*MOLDYN can perform numerical differentiation from order *1* to order *5*. Using order *1*, the first time derivative of each point $r(t_i)$ is calculated as

$$\dot{r}(t_i) = \frac{r(t_{i+1}) - r(t_i)}{\Delta t}, \tag{4.100}$$

  where Δt is the time step. Choosing order N with *N=2,...,5*, *n*MOLDYN calculates the first time-derivative of each point $r(t_i)$ ($r = x, y, z$) using the N-order polynomial interpolating the *N+1* points across $r(t_i)$, where $r(t_i)$ belongs to this set [52].

- **Project displacement on**
  Format: string
  Default: *no*
  Description: this widget allows to specify a vector along which the *ARA* will be computed. This vector does not need to be normalized as *n*MOLDYN will perform the normalization when processing it. The entered value must have the following format:

  vx:vy:vz

  where *vx*, *vy* and *vz* are floats that represent respectively the x, y and z coordinates of the vector.

- **Model order**
  Format: integer in [1,N_{frames}[where N_{frames} is the number of selected frames for the analysis
  Default: *50*

Description: this widget allows to specify P, the order (= poles number) of the autoregressive model. *A priori* the autocorrelation function and its power spectrum can be approximated to almost arbitrary precision by increasing the order of the autoregressive model. In practice it has been proven that reliably computation can be carried out up to P of the order of 1000 poles.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *equal*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the various properties computed through *ARA* analysis (*MSD*, *VACF*, *DOS*,). See Section 4.2.1 for more details.

- **ARA output file**
  Format: string
  Default: *ARA_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *ARA* analysis. A CDL version of the NetCDF output file is also automatically created with *ARA_traj_file.cdl* name.

Output

The results of an *ARA* analysis are stored in a NetCDF file whose main variables are namely:

- time_vacf: the times in *ps* at which the autoregressive *VACF* was evaluated,
- vacf: the corresponding autoregressive *VACF* in nm^2s^{-2},
- frequency: the frequencies in *THz* at which the autoregressive *DOS* was evaluated,
- dos: the corresponding autoregressive *DOS*,
- time_msd: the times in *ps* at which the autoregressive *MSD* was evaluated,
- msd: the corresponding autoregressive *MSD* in nm^2,

- time_memory: the times in *ps* at which the autoregressive *VACF* memory function was evaluated,
- memory_function: the corresponding autoregressive *VACF* memory function,
- n: the index for the autoregressive coefficients $a_n^{(P)}$,
- ar_coefficients: the autoregressive coefficients $a_n^{(P)}$.

4.2.4.12 Quasi Harmonic Analysis

Theory and implementation

Quasi-**H**armonic **A**nalysis (*QHA*) is a method for obtaining effective modes of vibration from fluctuations calculated by a MD simulation. The underlying principle is that from atomic fluctuations, an effective force field can be calculated relative to the average dynamic structure that yields the same fluctuation matrix as that obtained from a normal mode calculation. Since the fluctuation matrix is inversely proportional to the effective force constant matrix, they have common eigenvectors which correspond to the quasiharmonic modes of vibration. Quasiharmonic modes can be analyzed in the same way as normal modes, and comparison of the results with those from harmonic approximation calculations for the same system is straightforward.

The way to perform a *QHA* in *n*MOLDYN is based on the diagonalization of the fluctuation matrix that can be easily retrieved from a MD simulation. Indeed, from a MD simulation, coordinates which define the position of all atoms as a function of time are saved at each step of the MD. From these coordinates, both the average position $\langle \mathbf{x} \rangle$ and the covariance matrix of fluctuations about the average position σ can be calculated the latter being defined as:

$$\sigma_{ij} = \langle (x_i - \langle x \rangle)(x_j - \langle x \rangle) \rangle \tag{4.101}$$

with variances as the diagonal elements and with covariances as the off-diagonal elements.

Once the covariance matrix of fluctuations is obtained, the Quasi-Harmonic modes of vibration Δx and their corresponding frequencies ω are calculated by solving (diagonalizing) the equation:

$$(\sigma' - \lambda' \mathbf{I})\,\eta = 0 \tag{4.102}$$

where $\mathbf{I}$ is the identity matrix and

$$\sigma' = \mathbf{M^{1/2}} \sigma \mathbf{M^{1/2}} \tag{4.103}$$

$\mathbf{M}$ being the diagonal mass matrix.

The solutions of 4.102 yielding:

$$\omega = (k_B T / \lambda')^{1/2} \tag{4.104}$$

and

$$\mathbf{\Delta x} = \mathbf{M}^{-1/2} \eta \tag{4.105}$$

Once the normal modes have been obtained, a great variety of analysis can be performed. *n*MOLDYN proposes some of them namely:

- Local and global character indicator

  Normal modes of vibration can involve all the atoms of the system, as in the case of low-frequency global deformation motions, or be localized in one particular part of the

molecule. It is useful to have indicators which define the degree of global character or local character for particular modes of vibration. Since the eigen vectors form an orthonormal basis $\sum_{j=1}^{3N} \Delta x_{ji}^2 = 1$ where N is the number of selected atoms, a local character indicator is given by:

$$lci = \sum_{j=1}^{3N} \Delta x_{ji}^4 \tag{4.106}$$

which is large for modes i with significant local character. The global character indicator is given by:

$$gci = \sum_{j=1}^{3N} \frac{|\Delta x_{ji}|}{\sqrt{3N}}^{-4} \tag{4.107}$$

which is large for modes i without significant global character. These indicators are not invariant with the orientation of the system and are only qualitative indicators of character because only the sum of the squares is invariant to rotation. If the motion is dominated by a single component of a single-atom, then Δx_i will be close to unity for that element and zero for all others. This results in the maximum possible value for the local characteer indicator of 1 and a maximum for the lack of global character indicator of $9N^2$. The other extreme is represented by a net translation of all atoms in the *(1,1,1)* direction, where all elements are the same with a value of $\sqrt{3N}$. In this case the local character indicator has a minimum value of $3N^{-1}$ and the lack of global character indicator will hav a minimum of 1. Modes with significant mixing of global and local character may have a large local indicator and a small lack of global character indicator; thus two indicators are needed to evaluate the character of the motion.

- Projection of MD trajectories onto normal modes

  It is useful to compare normal modes of vibration (or other type of motion) with the MD simulation. This allows examination of the amplitudes of this type of motion and the time scales. This is a straightforward procedure in which the effects of solvent damping may be explored for the * case in which the MD simulation is carried out with explicit solvent even if the normal mode calculation is carried out in vaccuum. The procedure is to generate a time series of the projection of the difference of trajectory position and the average position into a mode of interest given by:

$$A_a(t) = \sum_{i=1}^{3N} \Delta x_{ai} M_i^{\frac{1}{2}} \left(x_i(t) - \langle x_i \rangle \right) \tag{4.108}$$

  This time series can be evaluated with standard procedures to determine correlation functions and spectra. It is also useful in analyzing the results of a *QHA* analysis to determine which modes are predominantly vibrational in character and which arise mainly from jumping between energy substates.

Beside the quantitative analysis described above, one of the best way to understand a normal mode of vibration is through a visual examination. To do this, a trajectory must be created which depicts the molecular system as a function of time from a given starting configuration. For a set of normal modes to view, this trajectory is given by:

$$\mathbf{x}(t) = \langle \mathbf{x} \rangle + \sum_{i=1}^{N_{modes}} \alpha_i M^{-\frac{1}{2}} \Delta x_i cos(\omega_i t) \tag{4.109}$$

where α_i and ω_i are respectively the amplitude and frequency of the motion associated to normal mode i and N_{modes} is the number of selected normal modes to visualize. nMOLDYN allows the visualization of normal modes through a specific viewer (see Section 4.3.3).

For more details about QHA and related techniques please refer to Ref. [53]

Parameters

Pressing the **Quasi-Harmonic analysis** button will pop up the dialog shown on figure 4.46

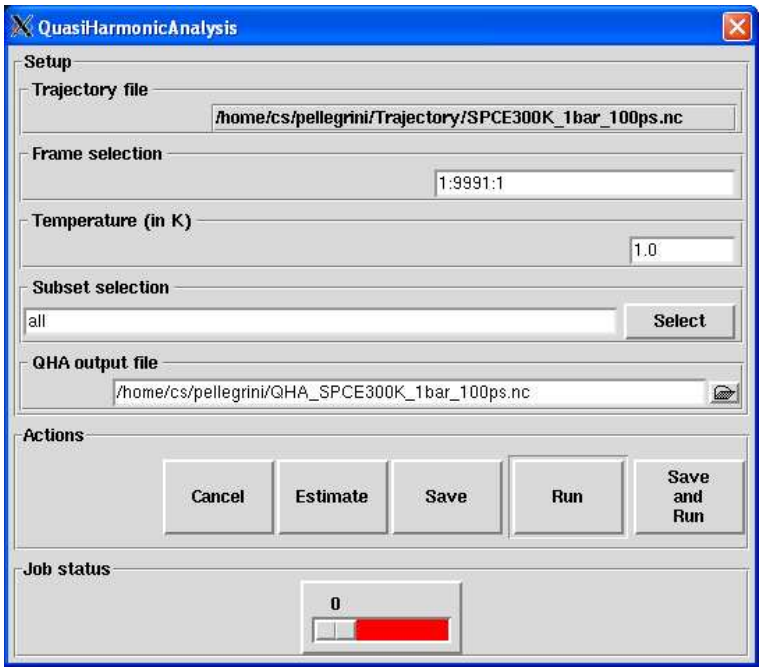

Figure 4.46: The dialog from where the *QHA* analysis will be set up and run.

The following input fields controls the parameters for the *QHA* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Temperature (in K)**
  Format: strictly positive float
  Default: *1.0*
  Description: this widget allows to define the temperature factor defined in equation 4.104.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **QHA Output file**
  Format: string
  Default: *QHA_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *QHA* analysis.

Output

The results of a *QHA* analysis are stored in a NetCDF file whose main variables are namely:

- time_msd: the times in *ps* at which the *QHA* was evaluated,
- omega: the *3N* eigen values ω defined in equation 4.104,
- dx: a *(3N,3N)* matrix giving the eigen vectors, Δx defined in equation 4.105
- mode: the mode index ranging from 1 to *3N*,
- lci: the local character indicator defined in equation 4.106,
- gci: the global character indicator defined in equation 4.107,
- at: a $(N_{frames}, 3N)$ matrix where N_{frames} is the number of selected frames for the analysis. This matrix stores the projection of MD trajectory onto normal modes as defined in equation 4.108,
- avgstruct: a *(N,3)* matrix storing the averaged structure $\langle x \rangle$.

4.2.4.13 Reorientational Correlation Function

Theory and implementation

The molecular reorientational correlation function is defined as the conditional probability to find a molecule with orientation $\mathbf{\Omega}_1$ at time t_1, given it had the orientation $\mathbf{\Omega}_0$ at time t_0. In the following this probability will be denoted by $p(\mathbf{\Omega}_1, t_1|\mathbf{\Omega}_0, t_0)$. Here $\mathbf{\Omega}$ denotes a set of angular coordinates, as Euler angles or quaternion parameters. The joint probability $p(\mathbf{\Omega}_1, t_1; \mathbf{\Omega}_0, t_0)$ which gives the probablity to find a molecule with orientation $\mathbf{\Omega}_0$ at time t_0 and with orientation $\mathbf{\Omega}_1$ at time t_1, can be expressed as $p(\mathbf{\Omega}_1, t_1; \mathbf{\Omega}_0, t_0) = p(\mathbf{\Omega}_1, t_1|\mathbf{\Omega}_0, t_0) \cdot p(\mathbf{\Omega}_0, t_0)$. Here $p(\mathbf{\Omega}_0, t_0)$ is the probability to find a molecule with orientation $\mathbf{\Omega}_0$ at time t_0. If we consider an isotropic system in thermal equilibrium the reorientational correlation function depends only on the time *difference*, $t = t_1 - t_0$, and the *change* in orientation, $\mathbf{\Omega}$, i.e. $p(\mathbf{\Omega}_1, t_1|\mathbf{\Omega}_0, t_0) = p(\mathbf{\Omega}, t|\mathbf{0}, 0)$. In addition we have $p(\mathbf{\Omega}_0, t_0) = 1/(8\pi^2)$, where $8\pi^2$ is the volume of the angular space.

The reorientational correlation function may now be expanded in Wigner rotation matrices [62] which form a complete set of basis functions in $\mathbf{\Omega}$ [63, 64]:

$$p(\mathbf{\Omega}, t|\mathbf{0}, 0) = \sum_{jmn} \frac{2j+1}{8\pi^2} p^j_{mn}(t) D^{*j}_{mn}(\mathbf{\Omega}). \tag{4.110}$$

In the following the coefficients $p^j_{mn}(t)$ are called p-coefficients. Using the orthogonality of the Wigner functions,

$$\int d\Omega\, D^{*j}_{mn}(\mathbf{\Omega}) D^{j'}_{m'n'}(\mathbf{\Omega}) = \frac{8\pi^2}{2j+1} \delta_{jj'} \delta_{mm'} \delta_{nn'}, \tag{4.111}$$

Eq. (4.110) can be inverted to give:

$$p^j_{mn}(t) = \int d\Omega\, p(\mathbf{\Omega}, t|\mathbf{0}, 0) D^j_{mn}(\mathbf{\Omega}). \tag{4.112}$$

Writing the reorientational correlation function as

$$p(\mathbf{\Omega}, t|\mathbf{0}, 0) = \frac{1}{N} \sum_\alpha \langle \delta[\mathbf{\Omega} - \mathbf{\Omega}_\alpha(t)] \rangle, \tag{4.113}$$

where $\mathbf{\Omega}_\alpha(t)$ is the orientation of molecule α with respect to its initial orientation and $\langle \ldots \rangle$ is a thermal average, relation (4.112) can be written as

$$p^j_{mn}(t) = \frac{1}{N} \sum_\alpha \langle D^j_{mn}(\mathbf{\Omega}_\alpha(t)) \rangle. \tag{4.114}$$

The p-coefficients can also be expressed as time correlation functions of irreducible tensor components. This is convenient for numerical purposes since time correlation functions of discrete and finite time series can be very efficiently computed by Fast Fourier Transform techniques (see Section A). Consider the general form of the time correlation function

$$\langle T^j_m(t_1) T^{*j}_n(t_0) \rangle = \int\int d\Omega_1 d\Omega_0\, p(\mathbf{\Omega}_1, t_1; \mathbf{\Omega}_0, t_0) T^j_m(\mathbf{\Omega}_1) T^{*j}_n(\mathbf{\Omega}_0), \tag{4.115}$$

where T^j_m are the components of an *irreducible tensor* [63, 64]. From the transformation properties of irreducible tensors it follows that

$$T^j_m(\mathbf{\Omega}_1) = \sum_k D^j_{mk}(\mathbf{\Omega}) T^j_k(\mathbf{\Omega}_0). \tag{4.116}$$

For an isotropic system in thermal equilibrium we may now write

$$p(\mathbf{\Omega}_1, t_1; \mathbf{\Omega}_0, t_0) = p(\mathbf{\Omega}, t|\mathbf{0}, 0) \cdot \frac{1}{8\pi^2}. \tag{4.117}$$

Inserting this in (4.115), performing a change in the integration variables from $(\mathbf{\Omega}_1, \mathbf{\Omega}_0)$ to $(\mathbf{\Omega}, \mathbf{\Omega}_0)$, and using the orthogonality of the Wigner functions one can show that

$$\langle T_m^j(t) T_n^{*\,j}(0)\rangle = p_{mn}^j(t) \cdot \frac{1}{2j+1} \sum_l |\hat{T}_l^j|^2, \tag{4.118}$$

where the components $\hat{T}_l^j$ are referred to a convenient reference frame. In practice only tensors with integer j are relevant. In this case, the well known spherical harmonics [63, 64] may be used to define irreducible tensors. They are related to the Wigner functions by

$$Y_m^j(\alpha, \beta) = \sqrt{\frac{2j+1}{4\pi}} D_{m0}^j(\alpha, \beta, \gamma), \tag{4.119}$$

where α, β, γ are Euler angles. Following Rose [65] the Wigner functions can be expressed as complex polynomials in the quaternion parameters:

$$\begin{aligned} D_{mn}^j(\mathbf{q}) = \textstyle\sum_p (-1)^p \frac{[(j+m)!(j-m)!(j+n)!(j-n)!]^{1/2}}{(j+m-p)!(j-n-p)!p!(p+n-m)!} \\ \times (q_0 + iq_3)^{j+m-p} (q_0 - iq_3)^{j-n-p} (q_2 + iq_1)^{p+n-m} (q_2 - iq_1)^p. \end{aligned} \tag{4.120}$$

Here the quaternion parameters describe the rotation of the space-fixed coordinate system into the body-fixed coordinate system. The corresponding rotation matrix is given in Eq. (4.52). According to Eq. (4.119) the spherical harmonics are just special cases of the Wigner functions,

$$\begin{aligned} Y_m^j(\mathbf{q}) = \sqrt{\tfrac{2j+1}{4\pi}} \textstyle\sum_p (-1)^p \frac{[(j+m)!(j-m)!]^{1/2} j!}{(j+m-p)!(j-p)!p!(j-m)!} \\ \times (q_0 + iq_3)^{j+m-p} (q_0 - iq_3)^{j-p} (q_2 + iq_1)^{p-m} (q_2 - iq_1)^p. \end{aligned} \tag{4.121}$$

Using the normalization of the spherical harmonics and Eq. (4.118) one arrives at the following expression for the p-coefficients

$$p_{mn}^j(t) = 4\pi \langle Y_m^j[\mathbf{q}(t)] Y_n^{*j}[\mathbf{q}(0)]\rangle. \tag{4.122}$$

The following relations for the p-coefficients hold:

$$p_{mn}^j(0) = \delta_{mn}^j, \tag{4.123}$$

$$p_{mn}^{*j}(t) = p_{-m-n}^j(t) = p_{nm}^j(-t). \tag{4.124}$$

The coefficients δ_{mn}^j are the components of the $(2j+1)\times(2j+1)$ unit matrix. The initial value of the the p-coefficients is an immediate consequence of definition (4.112) and $p(\mathbf{\Omega}, 0|\mathbf{0}, 0) = \delta(\mathbf{\Omega})$. Eq. (4.124) follows from the symmetry of the Wigner functions and the symmetry of classical time correlation functions.

Since measurable quantities must be real it follows from (4.124) that only p-coefficients with $m = n = 0$ can be directly measured. $p_{00}^1(t)$ is measured by infrared spectroscopy (dipole-dipole correlation function) and $p_{00}^2(t)$ by relaxation NMR experiments. Here one measures in most cases the integral over $p_{00}^2(t)$.

Parameters

Pressing the **Reorientational Correlation Function** button will pop up the dialog shown on figure 4.47

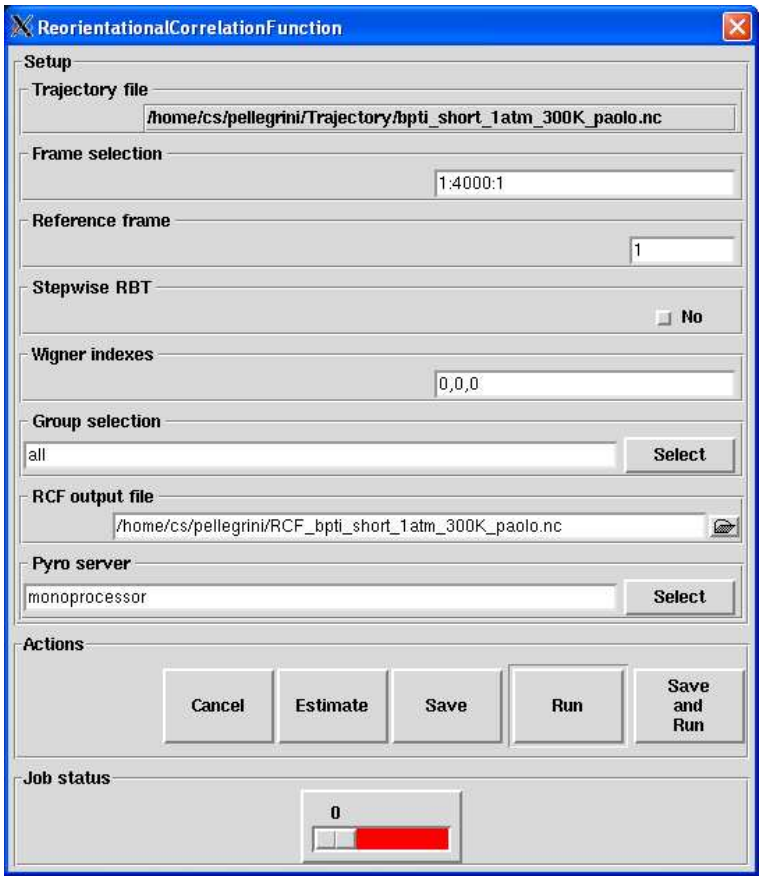

Figure 4.47: The dialog from where the *RCF* analysis will be set up and run.

The following input fields controls the parameters for the **R**eorientational **C**orrelation **F**unction (*RCF*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Reference frame**
  Format: integer in [1,*traj_length*] where *traj_length* is the number of frames of the input trajectory
  Default: *1*
  Description: this widget allows to specify which frame should be the reference for the *RCF* analysis. The value entered should be an integer ranging from 1 to *traj_length* where *traj_length* is the number of rames of the input trajectory.

- **Stepwise RBT**
  Format: string equal to *yes* or *no*
  Default: *no*
  Description: if set to *yes*, each frame *f* will serve as the reference for the frame *f+1* when defining the *RBT* canceling the value entred in **Reference frame** entry.

- **Wigner indexes**
  Format: string
  Default: *0,0,0*
  Description: this widget allows to specify which Wigner triplet *(j,m,n)* to select for the *RCF* analysis. The entered value must have the following specific format:

  j:m:n

  where *j*, *m*,*n* are positive integers that represent respectively the *j*, *m*,*n* Wigner indexes.

- **Group selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the groups of atoms that will be defined as rigid-bodies when performing the *RCF*. See Section 4.2.2.3 for more details.

- **RCF output file**
  Format: string
  Default: *RCF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *RCF* analysis. A CDL version of the NetCDF output file is also automatically created with *RCF_traj_file.cdl* name.

Output

The results of a *RCF* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which the *RCF* was evaluated,
- rcf: the corresponding *RCF*.

4.2.4.14 Angular Velocity AutoCorrelation Function

Theory and implementation

Similarly to the translational velocity autocorrelation functions introduced in Section 4.2.4.5 one can define angular velocity autocorrelation functions to characterize the angular motion of molecules. In general the angular velocity is referred to an orthonormal *body-fixed* coordinate system. Usually this is the principal axis system in which the tensor of inertia is diagonal. Depending on its geometry, a molecule will behave differently with respect to rotational motion about different body-fixed axes. The autocorrelation function for the angular velocity components ω_i' is defined as

$$C_{\omega\omega}(t;i) \doteq \langle \omega_i'(0)\omega_i'(t)\rangle. \tag{4.125}$$

The prime indicates a body fixed coordinate system. The components ω_i' are related to the quaternion parameters describing the orientation of the molecule and their time derivatives [14, 66]:

$$\begin{pmatrix} 0 \\ \omega_x' \\ \omega_y' \\ \omega_z' \end{pmatrix} = 2 \cdot \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{pmatrix} \begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix}. \tag{4.126}$$

Here the quaternion parameters describe the rotation of the space-fixed coordinate system into the body-fixed coordinate system. The corresponding rotation matrix is explicitly given in Eq. (4.52).

The components of the angular velocity may be used to define rotation angles describing rotations about the body-fixed axes [14]:

$$\Phi_i(t) = \int_0^t d\tau\, \omega_i'(\tau). \tag{4.127}$$

Parameters

Pressing the **Angular Velocity Autocorrelation Function** button will pop up the dialog shown on figure 4.48

The following input fields controls the parameters for the **A**ngular **V**elocity **A**uto**C**orrelation **F**unction (*AVACF*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string

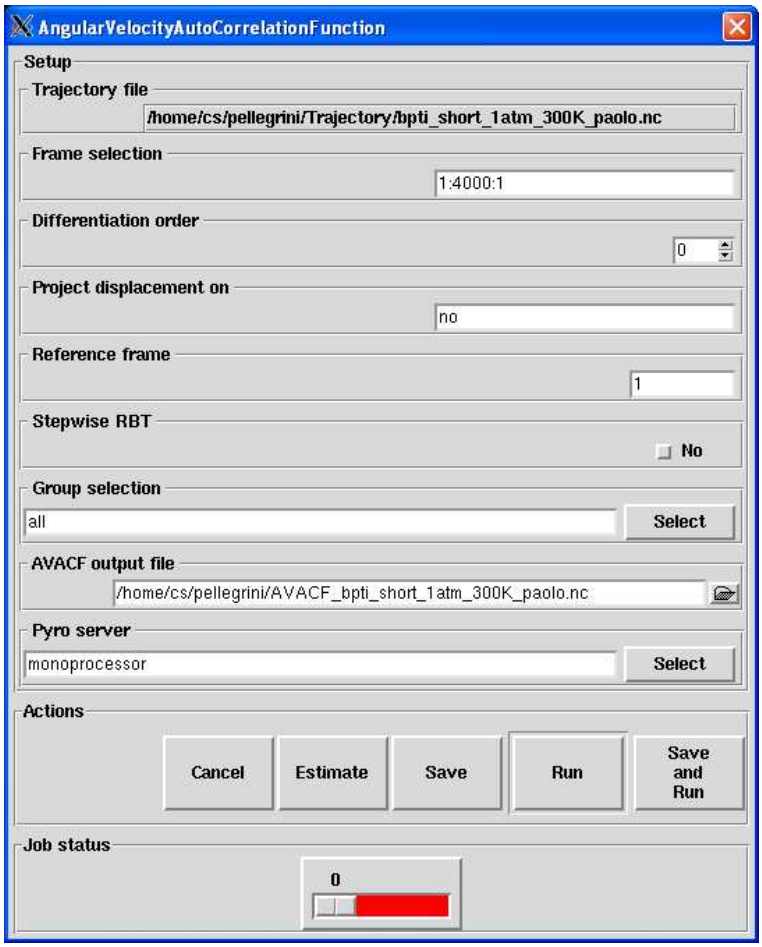

Figure 4.48: The dialog from where the *AVACF* analysis will be set up and run.

Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

first:last:step

where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Differentiation order**
  Format: integer in [0,5]
  Default: 0 if velocities are stored in the trajectory file, 1 otherwise
  Description: this widget allows to specify the order of the derivation scheme used to get

the velocities out of the coordinates. If your trajectory NetCDF file already contains the velocities then just select *0*. However, you can still decide to get the velocities out of the coordinates. In that case, *n*MOLDYN performs a numerical differentiation of the input data. To do so, *n*MOLDYN can perform numerical differentiation from order *1* to order *5*. Using order *1*, the first time derivative of each point $r(t_i)$ is calculated as

$$\dot{r}(t_i) = \frac{r(t_{i+1}) - r(t_i)}{\Delta t}, \tag{4.128}$$

where Δt is the time step. Choosing order N with *N=2,...,5*, *n*MOLDYN calculates the first time-derivative of each point $r(t_i)$ ($r = x, y, z$) using the N-order polynomial interpolating the *N+1* points across $r(t_i)$, where $r(t_i)$ belongs to this set [52].

- **Project displacement on**
  Format: string
  Default: *no*
  Description: this widget allows to specify a vector along which the *AVACF* will be computed. This vector does not need to be normalized as *n*MOLDYN will perform the normalization when processing it. The entered value must have the following format:

  vx:vy:vz

  where *vx*, *vy* and *vz* are floats that represent respectively the x, y and z coordinates of the vector.

- **Reference frame**
  Format: integer in [1,*traj_length*] where *traj_length* is the number of frames of the input trajectory
  Default: *1*
  Description: this widget allows to specify which frame should be the reference for the *AVACF* analysis. The value entered should be an integer ranging from 1 to *traj_length* where *traj_length* is the number of rames of the input trajectory.

- **Stepwise RBT**
  Format: string equal to *yes* or *no*
  Default: *no*
  Description: if set to *yes*, each frame f will serve as the reference for the frame *f+1* when defining the *RBT* canceling the value entred in **Reference frame** entry.

- **Group selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the groups of atoms that will be defined as rigid-bodies when performing the *AVACF*. See Section 4.2.2.3 for more details.

- **AVACF output file**
  Format: string
  Default: *AVACF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *AVACF* analysis. A CDL version of the NetCDF output file is also automatically created with *AVACF_traj_file.cdl* name.

Output
The results of a *AVACF* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which the *AVACF* was evaluated,
- avacf: the corresponding *AVACF*.

4.2.4.15 Angular Density Of States

Theory and implementation
In *n*MOLDYN, the **A**ngular **D**ensity **O**f **S**tates (*ADOS*) is simply defined as the Fourier transform of the *AVACF*.

Parameters
Pressing the **Angular Density Of States** button will pop up the dialog shown on figure 4.49
The following input fields controls the parameters for the *ADOS* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

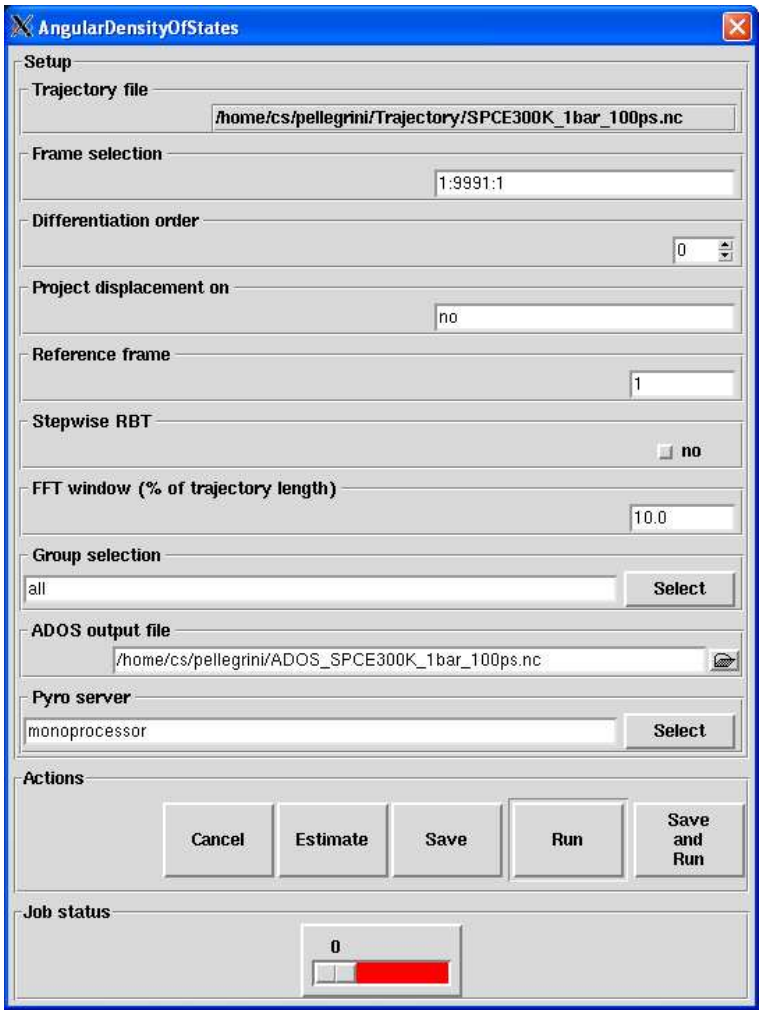

Figure 4.49: The dialog from where the *ADOS* analysis will be set up and run.

- **Differentiation order**
  Format: integer in [0,5]
  Default: 0 if velocities are stored in the trajectory file, 1 otherwise
  Description: this widget allows to specify the order of the derivation scheme used to get the velocities out of the coordinates. If your trajectory NetCDF file already contains the velocities then just select *0*. However, you can still decide to get the velocities out of the coordinates. In that case, *n*MOLDYN performs a numerical differentiation of the input data. To do so, *n*MOLDYN can perform numerical differentiation from order *1* to order *5*. Using order *1*, the first time derivative of each point $r(t_i)$ is calculated as

$$\dot{r}(t_i) = \frac{r(t_{i+1}) - r(t_i)}{\Delta t}, \tag{4.129}$$

  where Δt is the time step. Choosing order N with *N=2,...,5*, *n*MOLDYN calculates the first time-derivative of each point $r(t_i)$ ($r = x, y, z$) using the N-order polynomial interpolating the *N+1* points across $r(t_i)$, where $r(t_i)$ belongs to this set [52].

- **Project displacement on**

Format: string
Default: *no*
Description: this widget allows to specify a vector along which the *ADOS* will be computed. This vector does not need to be normalized as *n*MOLDYN will perform the normalization when processing it. The entered value must have the following format:

vx:vy:vz

where *vx*, *vy* and *vz* are floats that represent respectively the *x*, *y* and *z* coordinates of the vector.

- **Reference frame**
  Format: integer in [1,*traj_length*] where *traj_length* is the number of frames of the input trajectory
  Default: *1*
  Description: this widget allows to specify which frame should be the reference for the *ADOS* analysis. The value entered should be an integer ranging from 1 to *traj_length* where *traj_length* is the number of rames of the input trajectory.

- **Stepwise RBT**
  Format: string equal to *yes* or *no*
  Default: *no*
  Description: if set to *yes*, each frame *f* will serve as the reference for the frame *f+1* when defining the *RBT* canceling the value entred in **Reference frame** entry.

- **FFT window**
  Format: float in [0.0,100.0]
  Default: *10.0*
  Description: this widget allows to define the width in percentage of the trajectory length of the Gaussian function to be used in the smoothing procedure for the calculation of the *ADOS*. See Appendix A for more details.

- **Group selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the groups of atoms that will be defined as rigid-bodies when performing the *ADOS*. See Section 4.2.2.3 for more details.

- **ADOS output file**
  Format: string
  Default: *ADOS_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *ADOS* analysis. A CDL version of the NetCDF output file is also automatically created with *ADOS_traj_file.cdl* name.

Output

The results of a *ADOS* analysis are stored in a NetCDF file whose main variables are namely:

- frequency: the frequencies in *THz* at which the *ADOS* was evaluated,
- ados: the corresponding *ADOS*.

4.2.5 The Scattering menu

Pressing the button **Scattering** brings up a menu from which it is possible to choose the following analysis:

- Dynamic Coherent Structure Factor
- Dynamic Coherent Structure Factor (AR Model)
- Dynamic Incoherent Structure Factor
- Dynamic Incoherent Structure Factor (AR Model)
- Dynamic Incoherent Structure Factor (Gaussian Approximation)
- Elastic Incoherent Structure Factor
- Static Coherent Structure Factor
- Smoothed Static Coherent Structure Factor

Before introducing each of these analysis, a brief introduction about the scattering theory within the classical framework will be given.

4.2.5.1 Introduction

The quantity of interest in neutron scattering experiments with thermal neutrons is the *dynamic structure factor*, $\mathcal{S}(\mathbf{q},\omega)$, which is closely related to the double differential cross-section [7], $d^2\sigma/d\Omega dE$. The double differential cross section is defined as the number of neutrons which are scattered per unit time into the solid angle interval $[\Omega, \Omega + d\Omega]$ and into the energy interval *[E,E+dE]*. It is normalized to $d\Omega$, dE, and the flux of the incoming neutrons,

$$\frac{d^2\sigma}{d\Omega dE} = N \cdot \frac{k}{k_0}\mathcal{S}(\mathbf{q},\omega). \tag{4.130}$$

Here N is the number of atoms, and $k \equiv |\mathbf{k}|$ and $k_0 \equiv |\mathbf{k}_0|$ are the wave numbers of scattered and incident neutrons, respectively. They are related to the corresponding neutron energies by $E = \hbar^2 k^2/2m$ and $E_0 = \hbar^2 k_0^2/2m$, where m is the neutron mass. The arguments of the dynamic structure factor, $\boldsymbol{q}$ and ω, are the momentum and energy transfer in units of $\hbar$, respectively:

$$\mathbf{q} = \frac{\mathbf{k}_0 - \mathbf{k}}{\hbar}, \tag{4.131}$$

$$\omega = \frac{E_0 - E}{\hbar}. \tag{4.132}$$

The modulus of the momentum transfer can be expressed in the scattering angle θ, the energy transfer, and the energy of the incident neutrons:

$$q = \sqrt{2 - \frac{\hbar\omega}{E_0} - 2\cos\theta\sqrt{2 - \frac{\hbar\omega}{E_0}}}. \tag{4.133}$$

The dynamic structure factor contains information about the structure and dynamics of the scattering system [67]. It can be written as

$$\mathcal{S}(\mathbf{q},\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} dt\, \exp[-i\omega t]\mathcal{F}(\mathbf{q},t). \tag{4.134}$$

$\mathcal{F}(\mathbf{q},t)$ is called the *intermediate scattering function* and is defined as

$$\mathcal{F}(\mathbf{q},t) = \sum_{\alpha,\beta}\Gamma_{\alpha\beta}\langle\exp[-i\mathbf{q}\cdot\hat{\mathbf{R}}_\alpha(0)]\exp[i\mathbf{q}\cdot\hat{\mathbf{R}}_\beta(t)]\rangle, \tag{4.135}$$

$$\Gamma_{\alpha\beta} = \frac{1}{N}\left[\overline{b_\alpha}\,\overline{b_\beta} + \delta_{\alpha\beta}(\overline{b_\alpha^2} - \overline{b_\alpha}^2)\right]. \tag{4.136}$$

The operators $\hat{\mathbf{R}}_\alpha(t)$ in Eq. (4.135) are the position operators of the nuclei in the sample. The brackets $\langle\ldots\rangle$ denote a quantum thermal average and the time dependence of the position operators is defined by the Heisenberg picture. The quantities b_α are the scattering lengths of the nuclei which depend on the isotope and the relative orientation of the spin of the neutron and the spin of the scattering nucleus. If the spins of the nuclei and the neutron are not prepared in a special orientation one can assume a random relative orientation and that spin and position of the nuclei are uncorrelated. The symbol $\overline{\ldots}$ appearing in $\Gamma_{\alpha\beta}$ denotes an average over isotopes and relative spin orientations of neutron and nucleus.

Usually one splits the intermediate scattering function and the dynamic structure factor into their *coherent* and *incoherent* parts which describe collective and single particle motions, respectively. Defining

$$b_{\alpha,coh} \doteq \overline{b_\alpha}, \tag{4.137}$$

$$b_{\alpha,inc} \doteq \sqrt{\overline{b_\alpha^2} - \overline{b_\alpha}^2}, \tag{4.138}$$

the coherent and incoherent intermediate scattering functions can be cast in the form

$$\mathcal{F}_{coh}(\mathbf{q},t) = \frac{1}{N}\sum_{\alpha,\beta} b_{\alpha,coh}\, b_{\beta,coh}\langle\exp[-i\mathbf{q}\cdot\hat{\mathbf{R}}_\alpha(0)]\exp[i\mathbf{q}\cdot\hat{\mathbf{R}}_\beta(t)]\rangle, \tag{4.139}$$

$$\mathcal{F}_{inc}(\mathbf{q},t) = \frac{1}{N}\sum_{\alpha} b_{\alpha,inc}^2\langle\exp[-i\mathbf{q}\cdot\hat{\mathbf{R}}_\alpha(0)]\exp[i\mathbf{q}\cdot\hat{\mathbf{R}}_\alpha(t)]\rangle. \tag{4.140}$$

Rewriting these formulas, nMOLDYN introduces the partial terms as:

$$\mathcal{F}_{\mathrm{coh}}(\mathbf{q},t) = \sum_{I,J\geq I}^{N_{species}} \sqrt{n_I n_J \omega_{I,\mathrm{coh}}\omega_{J,\mathrm{coh}}}\mathcal{F}_{IJ,\mathrm{coh}}(\mathbf{q},t), \tag{4.141}$$

$$\mathcal{F}_{\mathrm{inc}}(\mathbf{q},t) = \sum_{I=1}^{N_{species}} n_I\omega_{I,\mathrm{inc}}\mathcal{F}_{I,\mathrm{inc}}(\mathbf{q},t) \tag{4.142}$$

where:

$$\mathcal{F}_{IJ,\mathrm{coh}}(\mathbf{q},t) = \frac{1}{\sqrt{n_I n_J}}\sum_{\alpha}^{n_I}\sum_{\beta}^{n_J}\langle\exp[-i\mathbf{q}\cdot\hat{\mathbf{R}}_\alpha(t_0)]\exp[i\mathbf{q}\cdot\hat{\mathbf{R}}_\beta(t_0+t)]\rangle_{t_0}, \tag{4.143}$$

$$\mathcal{F}_{I,\mathrm{inc}}(\mathbf{q},t) = \frac{1}{n_I}\sum_{\alpha=1}^{n_I}\langle\exp[-i\mathbf{q}\cdot\hat{\mathbf{R}}_\alpha(t_0)]\exp[i\mathbf{q}\cdot\hat{\mathbf{R}}_\alpha(t_0+t)]\rangle_{t_0}. \tag{4.144}$$

where n_I, n_J, $N_{species}$, $\omega_{I,\mathrm{coh,inc}}$ and $\omega_{J,\mathrm{coh,inc}}$ are defined in Section 4.2.1.

The corresponding dynamic structure factors are obtained by performing the Fourier transformation defined in Eq. 4.134.

An important quantity describing *structural* properties of liquids is the *static structure factor*, which is defined as

$$\mathcal{S}(\mathbf{q}) \doteq \int_{-\infty}^{+\infty} d\omega\, \mathcal{S}_{coh}(q,\omega) = \mathcal{F}_{coh}(\mathbf{q},0). \tag{4.145}$$

In the classical framework the intermediate scattering functions are interpreted as classical time correlation functions. The position operators are replaced by time-dependent vector functions and quantum thermal averages are replaced by classical *ensemble averages*. It is well known that this procedure leads to a loss of the universal detailed balance relation,

$$\mathcal{S}(\mathbf{q},\omega) = \exp[\beta\hbar\omega]\mathcal{S}(-\mathbf{q},-\omega), \tag{4.146}$$

and also to a loss of all odd moments

$$\langle\omega^{2n+1}\rangle \doteq \int_{-\infty}^{+\infty} d\omega\, \omega^{2n+1}\mathcal{S}(\mathbf{q},\omega), \qquad n = 1,2,\ldots. \tag{4.147}$$

The odd moments vanish since the classical dynamic structure factor is even in ω, assuming invariance of the scattering process with respect to reflections in space. The first moment is also universal. For an atomic liquid, containing only one sort of atoms, it reads

$$\langle\omega\rangle = \frac{\hbar q^2}{2M}\,, \tag{4.148}$$

where M is the mass of the atoms. Formula (4.148) shows that the first moment is given by the average kinetic energy (in units of $\hbar$) of a particle which receives a momentum transfer $\hbar\mathbf{q}$. Therefore $\langle\omega\rangle$ is called the *recoil moment*. A number of 'recipes' has been suggested to correct classical dynamic structure factors for detailed balance and to describe recoil effects in an approximate way. The most popular one has been suggested by Schofield [68]

$$\mathcal{S}(\mathbf{q},\omega) \approx \exp[\frac{\beta\hbar\omega}{2}]\mathcal{S}_{cl}(\mathbf{q},\omega). \tag{4.149}$$

One can easily verify that the resulting dynamic structure factor fulfills the relation of detailed balance. Formally, the correction (4.149) is correct to first order in $\hbar$. Therefore it cannot be used for large q-values which correspond to large momentum transfers $\hbar q$. This is actually true for all correction methods which have suggested so far. For more details we refer to Ref. [9].

4.2.5.2 Dynamic Coherent Structure Factor

Theory and implementation

Please refer to Section 4.2.5.1 for more details about the theoretical background related to the dynamic coherent structure factor. In this analysis, *n*MOLDYN proceeds in two steps. First, it computes the partial and total intermediate coherent scattering function using equation 4.141. Then, the partial and total dynamic coherent structure factors are obtained by performing the Fourier Transformation, defined in Eq. 4.134, respectively on the total and partial intermediate coherent scattering functions.

*n*MOLDYN computes the coherent intermediate scattering function on a rectangular grid of equidistantly spaced points along the time-and the q-axis, repectively:

$$\mathcal{F}_{\mathrm{coh}}(q_m, k{\cdot}\Delta t) \doteq \sum_{I=1,J\geq I}^{N_{species}} \sqrt{n_I n_J \omega_{I,\mathrm{com}}\omega_{I,\mathrm{com}}}\overline{\langle\rho_I(-\mathbf{q},0)\rho_J(\mathbf{q},k\cdot\Delta t)\rangle}^{q}, \qquad k = 0\ldots N_t{-}1,\ m = 0\ldots N_q{-}1. \tag{4.150}$$

where N_t is the number of time steps in the coordinate time series, N_q is a user-defined number of q-shells, $N_{species}$ is the number of selected species, n_I the number of atoms of species I, ω_I the weight for specie I (see Section 4.2.1 for more details) and $\rho_I(\mathbf{q}, k \cdot \Delta t)$ is the Fourier transformed particle density for specie I defined as,

$$\rho_I(\mathbf{q}, k \cdot \Delta t) = \sum_{\alpha}^{n_I} \exp[i\mathbf{q} \cdot \mathbf{R}_\alpha(k \cdot \Delta t)]. \tag{4.151}$$

The symbol $\overline{\ldots}^q$ in (4.150) denotes an average over q-vectors having *approximately* the same modulus $q_m = q_{min} + m \cdot \Delta q$. The particle density must not change if jumps in the particle trajectories due to periodic boundary conditions occcur. In addition the *average* particle density, N/V, must not change. This can be achieved by choosing q-vectors on a lattice which is reciprocal to the lattice defined by the MD box. Let $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ be the basis vectors which span the MD cell. Any position vector in the MD cell can be written as

$$\mathbf{R} = x'\mathbf{b}_1 + y'\mathbf{b}_2 + z'\mathbf{b}_3, \tag{4.152}$$

with x', y', z' having values between 0 and 1. The primes indicate that the coordinates are box coordinates. A jump due to periodic bounday conditions causes x', y', z' to jump by ± 1. The set of dual basis vectors $\mathbf{b}^1, \mathbf{b}^2, \mathbf{b}^3$ is defined by the relation

$$\mathbf{b}_i \mathbf{b}^j = \delta_i^j. \tag{4.153}$$

If the q-vectors are now chosen as

$$\mathbf{q} = 2\pi \left(k\mathbf{b}^1 + l\mathbf{b}^2 + m\mathbf{b}^3 \right), \tag{4.154}$$

where k,l,m are integer numbers, jumps in the particle trajectories produce phase changes of multiples of 2π in the Fourier transformed particle density, i.e. leave it unchanged. One can define a grid of q-shells or a grid of q-vectors along a given direction or on a given plane, giving in addition a *tolerance* for q. nMOLDYN looks then for q-vectors of the form (4.163) whose moduli deviate within the prescribed tolerance from the equidistant q-grid. From these q-vectors only a maximum number per grid-point (called generically q-shell also in the anisotropic case) is kept.

The q-vectors can be generated isotropically, anisotropically or along user-defined directions.

The $\sqrt{\omega_I}$ may be negative if they represent normalized coherent scattering lenghts, i.e.

$$\sqrt{\omega_I} = \frac{b_{I,\mathrm{coh}}}{\sqrt{\sum_{I=1}^{N_{species}} n_I b_{I,\mathrm{coh}}^2}}. \tag{4.155}$$

Negative coherent scattering lengths occur in hydrogenous materials since $b_{coh,H}$ is negative [7]. The density-density correlation is computed via the FCA technique described in Section A.

Parameters

Pressing the **Dynamic Coherent Structure Factor** button will pop up the dialog shown on figure 4.50

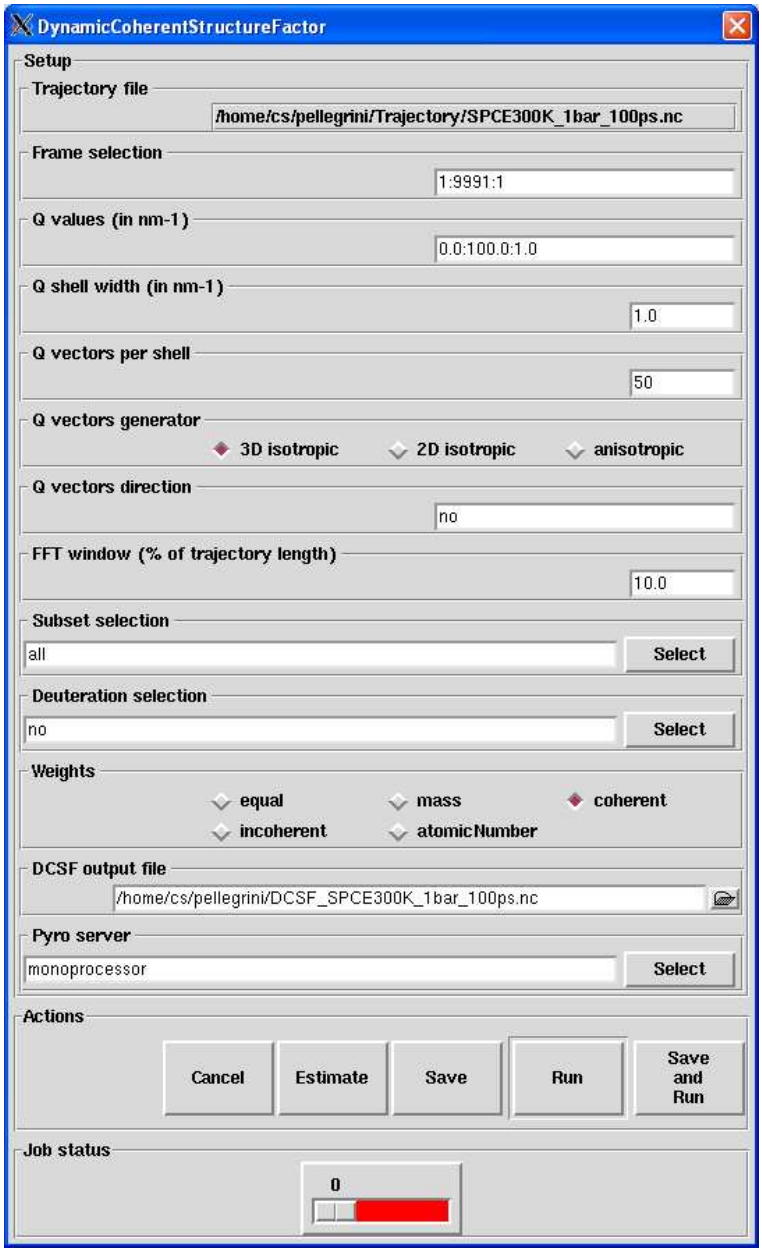

Figure 4.50: The dialog from where the *DCSF* analysis will be set up and run.

The following input fields controls the parameters for the **D**ynamic **C**oherent **S**tructure **F**actor (*DCSF*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information

purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Q values (in nm-1)**
  Format: string
  Default: *0:100:1.*
  Description: this widget allows to select the modulii of the q-vectors. This must be a string of the form:

  $q_{min} : q_{max} : q_{step}$

  In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max} is the radius of the biggest q-shell.

  For example,

  - ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
  - ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **Q shell width (in nm-1)**
  Format: strictly positive float
  Default: *1.0*
  Description: this widget allows to define the tolerance dq on the q-modulii. So, for each q-shell of modulus q, nMOLDYN will accept a q-vector to belong to that shell if its modulus falls in the range *[q-dq/2,q+dq/2]*. This parameter fix the q-resolution.

- **Q vectors per shell**
  Format: strictly positive integer
  Default: *50*
  Description: this widget allows to specify the number of q-vectors , N_q to generate for each q-shell. Indeed, when generating q-vectors, nMOLDYN will try to generate N_q q-vectors for each q-shell in order to carry out the averages of Eq. 4.151. For a given q-shell, if nMOLDYN could generate less q-vectors than N_q, it will only use the number of generated q-vectors instead of N_q and if nMOLDYN could generate more q-vectors than N_q, it will pick up randomly N_q q-vectors among the generated q-vectors. The higher this parameter is the smoother will be the computed intermediate scattering function but at the cost of a slower analysis.

- **Q vectors generator**
  Format: string equal to *3D isotropic*, *2D isotropic* or *anisotropic*
  Default: *3D isotropic*
  Description: this option allows to specify how the q-vectors should be generated:

  ⋆ 3D isotropic the q-vectors are generated randomly on concentric spheres.

  ⋆ 2D isotropic the q-vectors are generated randomly on concentric rings in a given plane.

  ⋆ anisotropic the q-vectors are generated randomly on one or several defined directions.

- **Q vectors direction**
  Format: string
  Default: *no*
  Description: this widget allows to specify one or several preferential directions along which the q-vectors have to be generated. Depening on the q-vectors generation type, the entered value will take different values:

  ⋆ 3D isotropic the default value *no* must be used.

  ⋆ 2D isotropic a string of the form

  $$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z$$

  where $q1_x, q1_y, q1_z$ and $q2_x, q2_y, q2_z$ are respectively the *x,y,z* components of q-vector ***q1*** and ***q2***, ***(q1,q2)*** defining the plane on which the q-vectors will be generated.

  ⋆ anisotropic a string of the form

  $$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z; \ldots$$

  where $q1_x, q1_y, q1_z$, $q2_x, q2_y, q2_z$ ... are respectively the *x,y,z* components of q-vector ***q1***, ***q2*** ..., the q-vector generation being performed along each defined direction.

- **FFT window**
  Format: float in [0.0,100.0]

Default: *10.0*
Description: this widget allows to define the width in percentage of the trajectory length of the Gaussian function to be used in the smoothing procedure for the calculation of the coherent structure factor out of the intermediate scattering function. See Appendix A for more details.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *coherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *DCSF*. See Section 4.2.1 for more details.

- **DCSF output file**
  Format: string
  Default: *DCSF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *DCSF* analysis. A CDL version of the NetCDF output file is also automatically created with *DCSF_traj_file.cdl* name.

Output

The results of a *DCSF* analysis are stored in a NetCDF file whose main variables are namely:

- octan: an array storing the codes for space octan. For example, X+Y+Z+ for the space octan corresponding to positive X, Y and Z.
- qvectors_statistics: array storing the number of q-vectors generated per space octan,
- q: the q-shells radii in nm^{-1},
- time: the times in *ps* at which the intermediate coherent scattering function is evaluated,
- Fqt-total: the total intermediate coherent scattering function,
- Fqt-XY: the partial intermediate coherent scattering function for species X and Y,
- frequency: the frequencies in *THz* at which the coherent structure factor is evaluated,

- Sqw-total: the total dynamic coherent structure factor,
- Sqw-XY: the partial dynamic coherent structure factor for species X and Y.

4.2.5.3 Dynamic Coherent Structure Factor (AR Model)

Theory and implementation

Another memory function that can be calculated by *n*MOLDYN is the memory function related to the coherent intermediate scattering function. It is defined through the corresponding memory function equation

$$\partial_t \mathcal{F}_{\mathrm{coh}}(\mathbf{q},t) = -\int_0^t d\tau\, \xi(\mathbf{q},t-\tau)\mathcal{F}_{\mathrm{coh}}(\mathbf{q},\tau). \tag{4.156}$$

The memory function $\xi(\mathbf{q},t)$, which depends on time as well as on q, permits the analysis of memory effects on different length scales. From a numerical point of view the calculation of the memory function equation relevant to the coherent intermediate scattering function is completely analogous to the case of the *VACF* memory function, the discrete time signal being here

$$\sum_{\alpha=1}^{N} b_{\alpha,\mathrm{coh}} \exp[-i\mathbf{q}\cdot\mathbf{R}_\alpha(t)]. \tag{4.157}$$

See Section 4.2.4.11 for more details about auto-regressive process.

In the framework of the Autoregressive model, *n*MOLDYN allows the intermediate coherent scattering function, its Fourier spectrum (the coherent dynamical structure factor) and its memory function to be computed on a rectangular grid of equidistantly spaced points along the time- and the q-axis, repectively. The user is referred to Section 4.2.4.11 for more theoretical details. The dynamical variable of the correlation function under consideration,

$$\sum_{I}^{N_{species}} \sqrt{n_I \omega_I} \sum_{\alpha=1}^{n_I} \exp[-i\mathbf{q}\cdot\mathbf{R}_\alpha(n\Delta t)] \tag{4.158}$$

is considered as a discrete "signal", which is modeled by an autoregressive stochastic process of order P. For each q-values the program calculates the set of the relevant P complex coefficients $\{a_n\}$ of the stochastic process, averaging aver all atoms of the system and over all cartesian components. The correlation functions and their Fourier spectra are then computed according to the algorithm described in Section 4.2.4.11. Starting from the discretized memory function equation, which relates the time evolution of the correlation function to its memory function, and using the correlation function calculated by the *AR* model, the program computes for each q-value the discretized memory function (see Section 4.2.4.11). The program performs the above calculations isotropically.

Parameters

Pressing the **Dynamic Coherent Structure Factor (AR Model)** button will pop up the dialog shown on figure 4.51

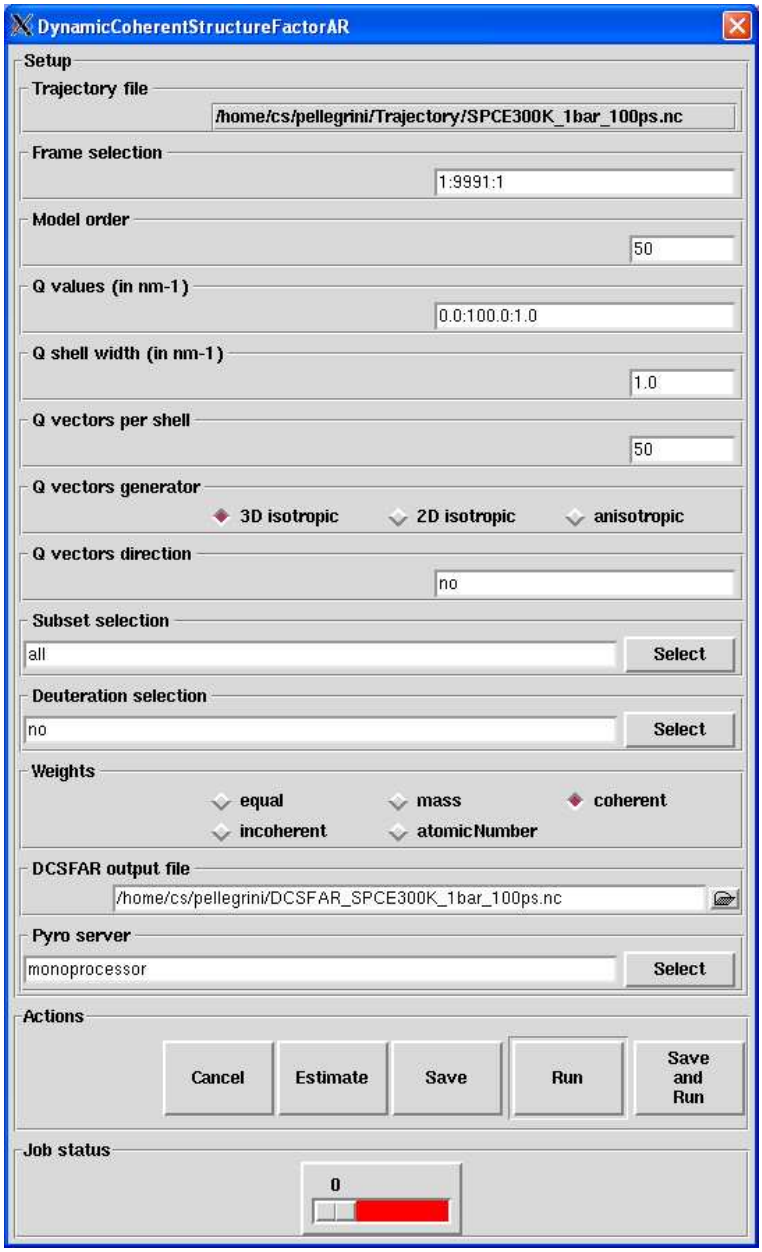

Figure 4.51: The dialog from where the *DCSFAR* analysis will be set up and run.

The following input fields controls the parameters for the **D**ynamic **C**oherent **S**tructure **F**actor using an **A**uto-**R**egressive model (*DCSFAR*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information

purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Model order**
  Format: integer in [1,N_{frames}[where N_{frames} is the number of selected frames for the analysis
  Default: *50*
  Description: this widget allows to specify P, the order (= poles number) of the autoregressive model. *A priori* the autocorrelation function and its power spectrum can be approximated to almost arbitrary precision by increasing the order of the autoregressive model. In practice it has been proven that reliably computation can be carried out up to P of the order of 1000 poles.

- **Q values (in nm-1)**
  Format: string
  Default: *0:100:1.*
  Description: this widget allows to select the modulii of the q-vectors. This must be a string of the form:

  $q_{min} : q_{max} : q_{step}$

  In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max} is the radius of the biggest q-shell.

  For example,

  - ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
  - ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **Q shell width (in nm-1)**
  Format: strictly positive float
  Default: *1.0*
  Description: this widget allows to define the tolerance dq on the q-modulii. So, for each q-shell of modulus q, nMOLDYN will accept a q-vector to belong to that shell if its modulus falls in the range *[q-dq/2,q+dq/2]*. This parameter fix the q-resolution.

- **Q vectors per shell**
  Format: strictly positive integer
  Default: *50*
  Description: this widget allows to specify the number of q-vectors , N_q to generate for each q-shell. Indeed, when generating q-vectors, nMOLDYN will try to generate N_q q-vectors for each q-shell in order to carry out the averages of Eq. 4.151. For a given q-shell, if nMOLDYN could generate less q-vectors than N_q, it will only use the number of generated q-vectors instead of N_q and if nMOLDYN could generate more q-vectors than N_q, it will pick up randomly N_q q-vectors among the generated q-vectors. The higher this parameter is the smoother will be the computed intermediate scattering function but at the cost of a slower analysis.

- **Q vectors generator**
  Format: string equal to *3D isotropic*, *2D isotropic* or *anisotropic*
  Default: *3D isotropic*
  Description: this option allows to specify how the q-vectors should be generated:

  - ⋆ 3D isotropic the q-vectors are generated randomly on concentric spheres.
  - ⋆ 2D isotropic the q-vectors are generated randomly on concentric rings in a given plane.
  - ⋆ anisotropic the q-vectors are generated randomly on one or several defined directions.

- **Q vectors direction**
  Format: string
  Default: *no*
  Description: this widget allows to specify one or several preferential directions along which the q-vectors have to be generated. Depening on the q-vectors generation type, the entered value will take different values:

  - ⋆ 3D isotropic the default value *no* must be used.
  - ⋆ 2D isotropic a string of the form

    $q1_x, q1_y, q1_z; q2_x, q2_y, q2_z$

    where $q1_x, q1_y, q1_z$ and $q2_x, q2_y, q2_z$ are respectively the x,y,z components of q-vector ***q1*** and ***q2***, ***(q1,q2)*** defining the plane on which the q-vectors will be generated.

⋆ anisotropic a string of the form

$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z; \ldots$

where $q1_x, q1_y, q1_z$, $q2_x, q2_y, q2_z$ ... are respectively the x,y,z components of q-vector ***q1***, ***q2*** ..., the q-vector generation being performed along each defined direction.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *coherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *DCSFAR*. See Section 4.2.1 for more details.

- **DCSFAR output file**
  Format: string
  Default: *DCSFAR_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *DCSFAR* analysis. A CDL version of the NetCDF output file is also automatically created with *DCSFAR_traj_file.cdl* name.

Output

The results of a *DCSFAR* analysis are stored in a NetCDF file whose main variables are namely:

- octan: an array storing the codes for space octan. For example, X+Y+Z+ for the space octan corresponding to positive X, Y and Z,
- qvectors_statistics: array storing the number of q-vectors generated per space octan,
- q: the q-shells radii in nm^{-1},
- time: the times in *ps* at which the intermediate coherent scattering function is evaluated,
- Fqt: the total intermediate coherent scattering function,

- frequency: the frequencies in *THz* at which the coherent structure factor is evaluated,
- Fqt_memory_function: the corresponding intermediate coherent scattering autoregressive memory function,
- Sqw: the total dynamic coherent structure factor,
- n: the index for the autoregressive coefficients $a_n^{(P)}$,
- ar_coefficients_real: the real part of the autoregressive coefficients $a_n^{(P)}$,
- ar_coefficients_imag: the imaginary part of the autoregressive coefficients $a_n^{(P)}$.

4.2.5.4 Dynamic Incoherent Structure Factor

Theory and implementation

Please refer to Section 4.2.5.1 for more details about the theoretical background related to the dynamic incoherent structure factor. In this analysis, *n*MOLDYN proceeds in two steps. First, it computes the partial and total intermediate incoherent scattering function $\mathcal{F}_{\text{inc}}(\mathbf{q}, t)$ using equation 4.142. Then, the partial and total dynamic incoherent structure factors are obtained by performing the Fourier Transformation, defined in Eq.4.134, respectively on the total and partial intermediate incoherent scattering function.

*n*MOLDYN computes the incoherent intermediate scattering function on a rectangular grid of equidistantly spaced points along the time-and the q-axis, repectively:

$$\mathcal{F}_{\text{inc}}(q_m, k{\cdot}\Delta t) \doteq \sum_{I=1}^{N_{species}} n_I \omega_{I,\text{inc}} F_{I,\text{inc}}(q_m, k{\cdot}\Delta t), \qquad k = 0 \ldots N_t{-}1,\ m = 0 \ldots N_q{-}1. \quad (4.159)$$

where N_t is the number of time steps in the coordinate time series, N_q is a user-defined number of q-shells, $N_{species}$ is the number of selected species, n_I the number of atoms of species I, $\omega_{I,\text{inc}}$ the weight for specie I (see Section 4.2.1 for more details) and $F_{I,\text{inc}}(q_m, k \cdot \Delta t)$ is defined as:

$$F_{I,inc,\alpha}(q_m, k \cdot \Delta t) = \sum_{\alpha=1}^{n_I} \overline{\langle \exp[-i\mathbf{q} \cdot \mathbf{R}_\alpha(0)] \exp[i\mathbf{q} \cdot \mathbf{R}_\alpha(t)] \rangle}^q. \quad (4.160)$$

The symbol $\overline{\ldots}^q$ in (4.160) denotes an average over q-vectors having *approximately* the same modulus $q_m = q_{min} + m \cdot \Delta q$. The particle density must not change if jumps in the particle trajectories due to periodic boundary conditions occcur. In addition the *average* particle density, N/V, must not change. This can be achieved by choosing q-vectors on a lattice which is reciprocal to the lattice defined by the MD box. Let $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ be the basis vectors which span the MD cell. Any position vector in the MD cell can be written as

$$\mathbf{R} = x'\mathbf{b}_1 + y'\mathbf{b}_2 + z'\mathbf{b}_3, \quad (4.161)$$

with x', y', z' having values between 0 and 1. The primes indicate that the coordinates are box coordinates. A jump due to periodic bounday conditions causes x', y', z' to jump by ± 1. The set of dual basis vectors $\mathbf{b}^1, \mathbf{b}^2, \mathbf{b}^3$ is defined by the relation

$$\mathbf{b}_i \mathbf{b}^j = \delta_i^j. \quad (4.162)$$

If the q-vectors are now chosen as

$$\mathbf{q} = 2\pi \left(k\mathbf{b}^1 + l\mathbf{b}^2 + m\mathbf{b}^3 \right), \quad (4.163)$$

where k,l,m are integer numbers, jumps in the particle trajectories produce phase changes of multiples of 2π in the Fourier transformed particle density, i.e. leave it unchanged. One can define a grid of q-shells or a grid of q-vectors along a given direction or on a given plane, giving in addition a *tolerance* for q. nMOLDYN looks then for q-vectors of the form (4.163) whose moduli deviate within the prescribed tolerance from the equidistant q-grid. From these q-vectors only a maximum number per grid-point (called generically q-shell also in the anisotropic case) is kept.

The q-vectors can be generated isotropically, anisotropically or along user-defined directions.

The correlation functions defined in 4.160 are computed via the FCA technique described in Section A. Although the efficient FCA technique is used to compute the atomic time correlation functions, the program may consume a considerable amount of CPU-time since the number of time correlation functions to be computed equals the number of atoms times the total number of q-vectors. This analysis is actually one of the most time-consuming among all the analysis available in nMOLDYN.

Parameters

Pressing the **Dynamic Incoherent Structure Factor** button will pop up the dialog shown on figure 4.52

The following input fields controls the parameters for the **D**ynamic **I**ncoherent **S**tructure **F**actor (*DISF*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Q values (in nm-1)**
  Format: string
  Default: *0:100:1.*
  Description: this widget allows to select the modulii of the q-vectors. This must be a

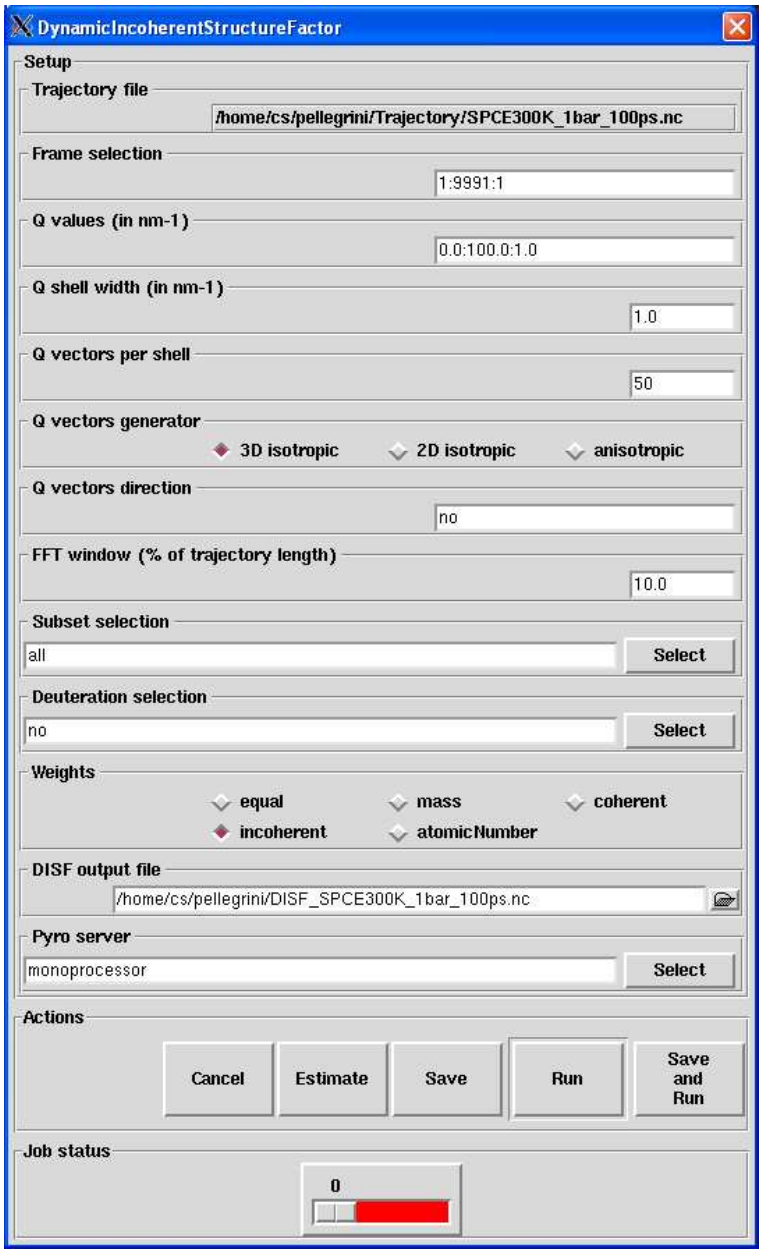

Figure 4.52: The dialog from where the *DISF* analysis will be set up and run.

string of the form:

$q_{min} : q_{max} : q_{step}$

In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max} is the radius of the biggest q-shell.

For example,

- ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **Q shell width (in nm-1)**
  Format: strictly positive float
  Default: *1.0*
  Description: this widget allows to define the tolerance dq on the q-modulii. So, for each q-shell of modulus q, nMOLDYN will accept a q-vector to belong to that shell if its modulus falls in the range *[q-dq/2,q+dq/2]*. This parameter fix the q-resolution.

- **Q vectors per shell**
  Format: strictly positive integer
  Default: *50*
  Description: this widget allows to specify the number of q-vectors , N_q to generate for each q-shell. Indeed, when generating q-vectors, nMOLDYN will try to generate N_q q-vectors for each q-shell in order to carry out the averages of Eq. 4.151. For a given q-shell, if nMOLDYN could generate less q-vectors than N_q, it will only use the number of generated q-vectors instead of N_q and if nMOLDYN could generate more q-vectors than N_q, it will pick up randomly N_q q-vectors among the generated q-vectors. The higher this parameter is the smoother will be the computed intermediate scattering function but at the cost of a slower analysis.

- **Q vectors generator**
  Format: string equal to *3D isotropic*, *2D isotropic* or *anisotropic*
  Default: *3D isotropic*
  Description: this option allows to specify how the q-vectors should be generated:
  - ⋆ 3D isotropic the q-vectors are generated randomly on concentric spheres.
  - ⋆ 2D isotropic the q-vectors are generated randomly on concentric rings in a given plane.
  - ⋆ anisotropic the q-vectors are generated randomly on one or several defined directions.

- **Q vectors direction**
  Format: string
  Default: *no*
  Description: this widget allows to specify one or several preferential directions along which the q-vectors have to be generated. Depening on the q-vectors generation type, the entered value will take different values:
  - ⋆ 3D isotropic the default value *no* must be used.
  - ⋆ 2D isotropic a string of the form

    $q1_x, q1_y, q1_z; q2_x, q2_y, q2_z$

where $q1_x, q1_y, q1_z$ and $q2_x, q2_y, q2_z$ are respectively the *x,y,z* components of *q*-vector ***q1*** and ***q2***, ***(q1,q2)*** defining the plane on which the *q*-vectors will be generated.

⋆ anisotropic a string of the form

$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z; \ldots$

where $q1_x, q1_y, q1_z$, $q2_x, q2_y, q2_z$ ... are respectively the *x,y,z* components of *q*-vector ***q1***, ***q2*** ..., the *q*-vector generation being performed along each defined direction.

- **FFT window**
  Format: float in [0.0,100.0]
  Default: *10.0*
  Description: this widget allows to define the width in percentage of the trajectory length of the Gaussian function to be used in the smoothing procedure for the calculation of the coherent structure factor out of the intermediate scattering function. See Appendix A for more details.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *incoherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *DISF*. See Section 4.2.1 for more details.

- **DISF output file**
  Format: string
  Default: *DISF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *DISF* analysis. A CDL version of the NetCDF output file is also automatically created with *DISF_traj_file.cdl* name.

Output

The results of a *DISF* analysis are stored in a NetCDF file whose main variables are namely:

- octan: an array storing the codes for space octan. For example, X+Y+Z+ for the space octan corresponding to positive X, Y and Z,
- qvectors_statistics: array storing the number of q-vectors generated per space octan,
- q: the q-shells radii in nm^{-1},
- time: the times in ps at which the intermediate incoherent scattering function is evaluated,
- Fqt-total: the total intermediate incoherent scattering function,
- Fqt-X: the partial intermediate incoherent scattering function for specie X,
- frequency: the frequencies in THz at which the in coherent structure factor is evaluated,
- Sqw-total: the total dynamic incoherent structure factor,
- Sqw-X: the partial dynamic incoherent structure factor for specie X.

4.2.5.5 Dynamic Incoherent Structure Factor (AR Model)

Theory and implementation

nMOLDYN allows one to calculate the memory function related to the incoherent intermediate scattering function as well. It is defined through the corresponding memory function equation

$$\partial_t \mathcal{F}_{inc}(\mathbf{q},t) = -\int_0^t d\tau\, \xi(\mathbf{q},t-\tau)\mathcal{F}_{inc}(\mathbf{q},\tau). \tag{4.164}$$

The memory function $\xi(\mathbf{q},\mathbf{t})$, which depends on q as well as on time, permits the analysis of memory effects on different length scales. As in the previuos cases, the numerical calculation of the memory function equation relevant to the incoherent intermediate scattering function is based on the Autoregressive model, the discrete time signal being here

$$\sum_{\alpha=1}^{N} b_{\alpha,\mathrm{inc}} \exp[-i\mathbf{q}\cdot\mathbf{R}_\alpha(t)]. \tag{4.165}$$

See Section 4.2.4.11 for more details about auto-regressive process.

In the framework of the Autoregressive model nMOLDYN allows the intermediate coherent scattering function, its Fourier spectrum (the incoherent dynamical structure factor) and its memory function to be computed on a rectangular grid of equidistantly spaced points along the time- and the q-axis, repectively. The user is referred to Section 4.2.4.11 for more theoretical details. The dynamical variable of the correlation function under consideration

$$\sum_{I=1}^{N_{species}} n_I\omega_I \sum_{\alpha=1}^{nI} \exp[-i\mathbf{q}\cdot\mathbf{R}_\alpha(n\Delta t)] \tag{4.166}$$

is considered as a discrete "signal", which is modeled by an autoregressive stochastic process of order P. For each q-values the program calculates a set of P complex coefficients a_n for the *AR* model averaging aver all atoms of the system and over all cartesian components. The correlation functions and their Fourier spectra are then computed according to the algorithm described in Section 4.2.4.11. Starting from the discretized memory function equation, whitch relates the time evolution of the correlation function to its memory function (see Section 4.2.4.11), and using the correlation function calculated by the *AR* model, the program computes for each q-value the discretized memory function. The program performs the above calculations isotropically.

Parameters

Pressing the **Dynamic Incoherent Structure Factor (AR Model)** button will pop up the dialog shown on figure 4.53

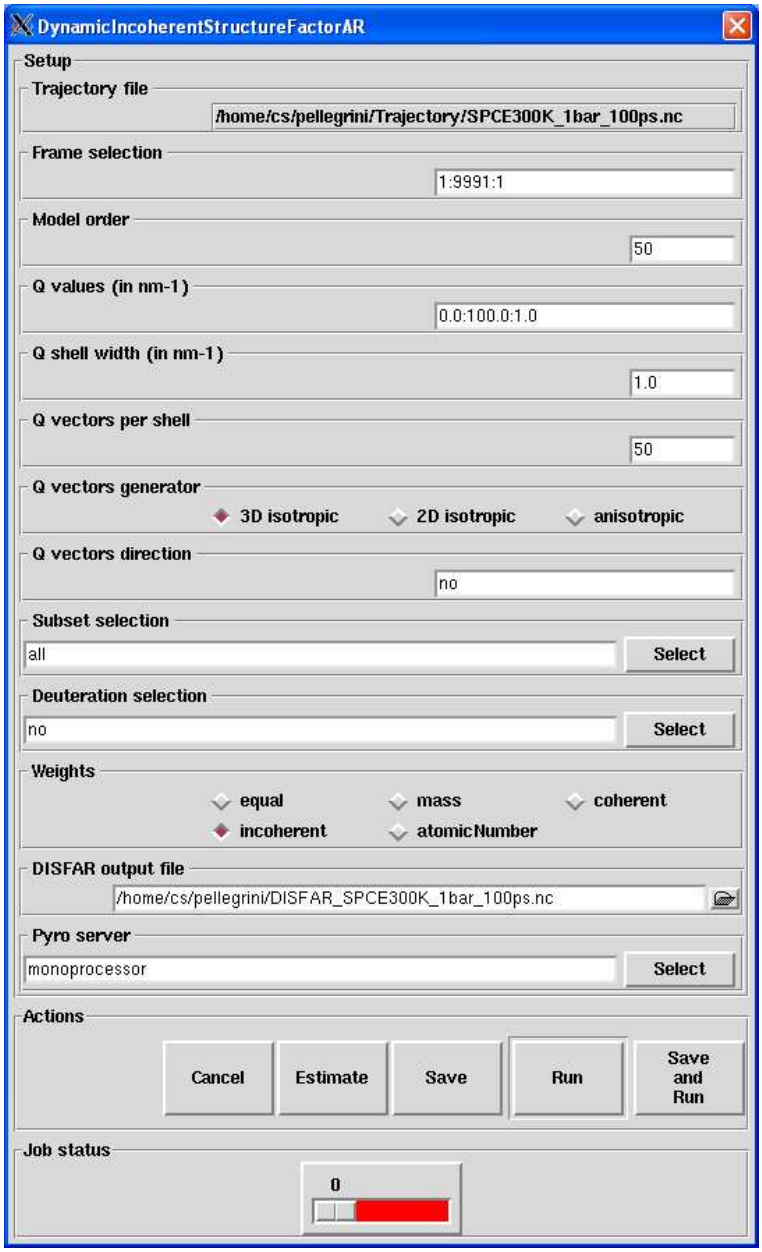

Figure 4.53: The dialog from where the *DISFAR* analysis will be set up and run.

The following input fields controls the parameters for the **D**ynamic **I**ncoherent **S**tructure **F**actor using an **A**uto-**R**egressive model (*DISFAR*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information

purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Model order**
  Format: integer in [1,N_{frames}[where N_{frames} is the number of selected frames for the analysis
  Default: *50*
  Description: this widget allows to specify P, the order (= poles number) of the autoregressive model. *A priori* the autocorrelation function and its power spectrum can be approximated to almost arbitrary precision by increasing the order of the autoregressive model. In practice it has been proven that reliably computation can be carried out up to P of the order of 1000 poles.

- **Q values (in nm-1)**
  Format: string
  Default: *0:100:1.*
  Description: this widget allows to select the modulii of the q-vectors. This must be a string of the form:

  $q_{min} : q_{max} : q_{step}$

  In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max} is the radius of the biggest q-shell.

  For example,

  - ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
  - ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **Q shell width (in nm-1)**
  Format: strictly positive float
  Default: *1.0*
  Description: this widget allows to define the tolerance dq on the q-modulii. So, for each q-shell of modulus q, nMOLDYN will accept a q-vector to belong to that shell if its modulus falls in the range *[q-dq/2,q+dq/2]*. This parameter fix the q-resolution.

- **Q vectors per shell**
  Format: strictly positive integer
  Default: *50*
  Description: this widget allows to specify the number of q-vectors , N_q to generate for each q-shell. Indeed, when generating q-vectors, nMOLDYN will try to generate N_q q-vectors for each q-shell in order to carry out the averages of Eq. 4.151. For a given q-shell, if nMOLDYN could generate less q-vectors than N_q, it will only use the number of generated q-vectors instead of N_q and if nMOLDYN could generate more q-vectors than N_q, it will pick up randomly N_q q-vectors among the generated q-vectors. The higher this parameter is the smoother will be the computed intermediate scattering function but at the cost of a slower analysis.

- **Q vectors generator**
  Format: string equal to *3D isotropic*, *2D isotropic* or *anisotropic*
  Default: *3D isotropic*
  Description: this option allows to specify how the q-vectors should be generated:

  ⋆ 3D isotropic the q-vectors are generated randomly on concentric spheres.

  ⋆ 2D isotropic the q-vectors are generated randomly on concentric rings in a given plane.

  ⋆ anisotropic the q-vectors are generated randomly on one or several defined directions.

- **Q vectors direction**
  Format: string
  Default: *no*
  Description: this widget allows to specify one or several preferential directions along which the q-vectors have to be generated. Depening on the q-vectors generation type, the entered value will take different values:

  ⋆ 3D isotropic the default value *no* must be used.

  ⋆ 2D isotropic a string of the form

  $$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z$$

  where $q1_x, q1_y, q1_z$ and $q2_x, q2_y, q2_z$ are respectively the x,y,z components of q-vector ***q1*** and ***q2***, ***(q1,q2)*** defining the plane on which the q-vectors will be generated.

⋆ anisotropic a string of the form

$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z; \ldots$

where $q1_x, q1_y, q1_z$, $q2_x, q2_y, q2_z$ ... are respectively the x,y,z components of q-vector ***q1***, ***q2*** ..., the q-vector generation being performed along each defined direction.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *incoherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *DISFAR*. See Section 4.2.1 for more details.

- **DISFAR output file**
  Format: string
  Default: *DISFAR_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *DISFAR* analysis. A CDL version of the NetCDF output file is also automatically created with *DISFAR_traj_file.cdl* name.

Output

The results of a *DISFAR* analysis are stored in a NetCDF file whose main variables are namely:

- octan: an array storing the codes for space octan. For example, X+Y+Z+ for the space octan corresponding to positive X, Y and Z,
- qvectors_statistics: array storing the number of q-vectors generated per space octan,
- q: the q-shells radii in nm^{-1},
- time: the times in *ps* at which the intermediate coherent scattering function is evaluated,
- Fqt: the total intermediate coherent scattering function,

- frequency: the frequencies in *THz* at which the coherent structure factor is evaluated,
- Fqt_memory_function: the corresponding intermediate coherent scattering autoregressive memory function,
- Sqw: the total dynamic coherent structure factor,
- n: the index for the autoregressive coefficients $a_n^{(P)}$,
- ar_coefficients_real: the real part of the autoregressive coefficients $a_n^{(P)}$,
- ar_coefficients_imag: the imaginary part of the autoregressive coefficients $a_n^{(P)}$.

4.2.5.6 Dynamic Incoherent Structure Factor (Gaussian Approximation)

Theory and implementation

The *MSD* can be related to the incoherent intermediate scattering function via the cumulant expansion [49, 50]

$$\mathcal{F}^g_{\rm inc}(\mathbf{q},t) = \sum_{I=1}^{N_{species}} n_I \omega_{I,\rm inc} \mathcal{F}^g_{I,\rm inc}(\mathbf{q},t) \tag{4.167}$$

where $N_{species}$ is the number of selected species, n_I the number of atoms of species I, $\omega_{I,\rm inc}$ the weight for specie I (see Section 4.2.1 for more details) and

$$\mathcal{F}^g_{I,\rm inc}(\mathbf{q},t) = \frac{1}{n_I}\sum_{\alpha}^{n_I} \exp[-q^2\rho_{\alpha,1}(t) + q^4\rho_{\alpha,2}(t) \mp \ldots]. \tag{4.168}$$

The cumulants $\rho_{\alpha,k}(t)$ are defined as

$$\rho_{\alpha,1}(t) = \frac{1}{2!}\langle d^2_\alpha(t;\mathbf{n}_q)\rangle \tag{4.169}$$

$$\rho_{\alpha,2}(t) = \frac{1}{4!}\left[\langle d^4_\alpha(t;\mathbf{n}_q)\rangle - 3\langle d^2_\alpha(t;\mathbf{n}_q)\rangle^2\right] \tag{4.170}$$

$$\vdots$$

The vector $\mathbf{n}_q$ is the unit vector in the direction of $\mathbf{q}$. In the Gaussian approximation the above expansion is truncated after the q^2-term. For certain model systems like the ideal gas, the harmonic oscillator, and a particle undergoing Einstein diffusion, this is exact. For these systems the incoherent intermediate scattering function is completely determined by the *MSD*.

*n*MOLDYN allows one to compute the total and partial incoherent intermediate scattering function in the *Gaussian approximation* by discretizing equation 4.167:

$$\mathcal{F}^g_{\rm inc}(q_m, k\cdot\Delta t) \doteq \sum_{I=1}^{N_{species}} n_I \omega_{I,\rm inc} F^g_{I,\rm inc}(q_m, k\cdot\Delta t), \qquad k = 0\ldots N_t-1,\ m = 0\ldots N_q-1. \tag{4.171}$$

with for each specie the following expression for the intermediate scattering function:

$$F^g_{I,\alpha,\rm inc}(q_m, k\cdot\Delta t) = \frac{1}{n_I}\sum_{\alpha}^{n_I} \exp\left[-\frac{(q_m)^2}{6}\Delta^2_\alpha(k\cdot\Delta t)\right] \qquad \text{isotropic system,} \tag{4.172}$$

$$F^g_{I,\alpha,\rm inc}(q_m, k\cdot\Delta t) = \frac{1}{n_I}\sum_{\alpha}^{n_I} \exp\left[-\frac{(q_m)^2}{2}\Delta^2_\alpha(k\cdot\Delta t;\mathbf{n})\right] \qquad \text{non-isotropic system} \tag{4.173}$$

N_t is the total number of time steps in the coordinate time series and N_q is a user-defined number of q-shells. The (q,t)-grid is the same as for the calculation of the intermediate incoherent scatering function (see Section 4.2.5.4). The quantities $\Delta^2_\alpha(t)$ and $\Delta^2_\alpha(t;\mathbf{n})$ are the mean-square displacements, defined in Equations (4.16) and (4.17), respectively. They are computed by using the algorithm described in Section 4.2.4.1. nMOLDYN corrects the atomic input trajectories for jumps due to periodic boundary conditions. It should be noted that the computation of the intermediate scattering function in the Gaussian approximation is much 'cheaper' than the computation of the full intermediate scattering function, $\mathcal{F}_{\text{inc}}(q,t)$, since no averaging over different q-vectors needs to be performed. It is sufficient to compute a single mean-square displacement per atom.

Parameters

Pressing the **Dynamic Incoherent Structure Factor (Gaussian Approximation)** button will pop up the dialog shown on figure 4.54

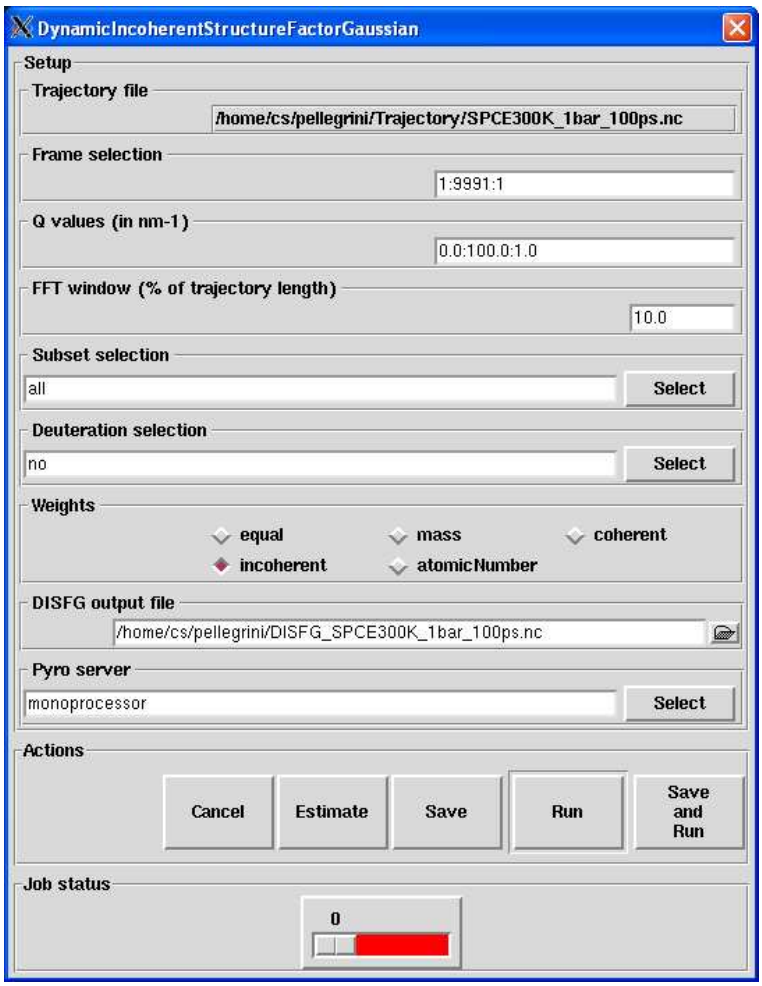

Figure 4.54: The dialog from where the *DISFG* analysis will be set up and run.

The following input fields controls the parameters for the **D**ynamic **I**ncoherent **S**tructure **F**actor using an **G**aussian approximation (*DISFG*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Q values (in nm-1)**
  Format: string
  Default: *0:100:1.*
  Description: this widget allows to select the modulii of the q-vectors. This must be a string of the form:

  $q_{min} : q_{max} : q_{step}$

  In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max} is the radius of the biggest q-shell.

  For example,

  - ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
  - ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **FFT window**
  Format: float in [0.0,100.0]
  Default: *10.0*
  Description: this widget allows to define the width in percentage of the trajectory length of the Gaussian function to be used in the smoothing procedure for the calculation of the

coherent structure factor out of the intermediate scattering function. See Appendix A for more details.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *incoherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *DISFG*. See Section 4.2.1 for more details.

- **DISFG output file**
  Format: string
  Default: *DISFG_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *DISFG* analysis. A CDL version of the NetCDF output file is also automatically created with *DISFG_traj_file.cdl* name.

Output

The results of a *DISFG* analysis are stored in a NetCDF file whose main variables are namely:

- q: the q-shells radii in nm^{-1},
- time: the times in ps at which the intermediate incoherent scattering function is evaluated,
- Fqt-total: the total intermediate incoherent scattering function,
- Fqt-X: the partial intermediate incoherent scattering function for specie X,
- frequency: the frequencies in THz at which the in coherent structure factor is evaluated,
- Sqw-total: the total dynamic incoherent structure factor,
- Sqw-X: the partial dynamic incoherent structure factor for specie X.

4.2.5.7 Elastic Incoherent Structure Factor

Theory and implementation

The **E**lastic **I**ncoherent **S**tructure **F**actor (*EISF*) is defined as the limit of the incoherent intermediate scattering function for infinite time,

$$EISF(\mathbf{q}) \doteq \lim_{t\to\infty} \mathcal{F}_{\text{inc}}(\mathbf{q},t). \tag{4.174}$$

Using the above definition of the EISF one can decompose the incoherent intermediate scattering function as follows:

$$\mathcal{F}_{\text{inc}}(\mathbf{q},t) = EISF(\mathbf{q}) + \mathcal{F}'_{\text{inc}}(\mathbf{q},t), \tag{4.175}$$

where $\mathcal{F}'_{\text{inc}}(\mathbf{q},t)$ decays to zero for infinite time. Taking now the Fourier transform it follows immediately that

$$\mathcal{S}_{\text{inc}}(\mathbf{q},\omega) = EISF(\mathbf{q})\delta(\omega) + \mathcal{S}'_{\text{inc}}(\mathbf{q},\omega). \tag{4.176}$$

The *EISF* appears as the amplitude of the *elastic* line in the neutron scattering spectrum. Elastic scattering is only present for sytems in which the atomic motion is confined in space, as for solids. To understand which information is contained in the *EISF* we consider for simplicity a system where only one sort of atoms is visible to the neutrons. To a very good approximation this is the case for all systems containing a large amount of hydrogen atoms, as biological systems. Incoherent scattering from hydrogen dominates by far all other contributions. Using the definition of the van Hove self-correlation function $G_s(\mathbf{r},t)$ [7],

$$b^2_{\text{inc}} G_s(\mathbf{r},t) \doteq \frac{1}{2\pi^3}\int d^3q\, \exp[-i\mathbf{q}\cdot\mathbf{r}]\mathcal{F}_{inc}(\mathbf{q},t), \tag{4.177}$$

which can be interpreted as the conditional probability to find a tagged particle at the position $\mathbf{r}$ at time t, given it started at $\mathbf{r}=\mathbf{0}$, one can write:

$$EISF(\mathbf{q}) = b^2_{\text{inc}}\int d^3r\, \exp[i\mathbf{q}\cdot\mathbf{r}]\, G_s(\mathbf{r},t=\infty). \tag{4.178}$$

The *EISF* gives the sampling distribution of the points in space in the limit of infinite time. In a real experiment this means times longer than the time which is observable with a given instrument. The *EISF* vanishes for all systems in which the particles can access an infinite volume since $G_s(\mathbf{r},t)$ approaches $1/V$ for large times. This is the case for molecules in liquids and gases.

For computational purposes it is convenient to use the following representation of the *EISF* [14]:

$$EISF(\mathbf{q}) = \sum_{I=1}^{N_{species}} n_I \omega_{I,\text{inc}} EISF_I(q) \tag{4.179}$$

where $N_{species}$ is the number of selected species, n_I the number of atoms of species I, $\omega_{I,\text{inc}}$ the weight for specie I (see Section 4.2.1 for more details) and for each specie the following expression for the elastic incoherent scattering function is

$$EISF_I(\mathbf{q}) = \frac{1}{n_I}\sum_{\alpha}^{n_I}\langle|\exp[i\mathbf{q}\cdot\mathbf{R}_\alpha]|^2\rangle. \tag{4.180}$$

This expression is derived from definition (4.174) of the *EISF* and expression (4.142) for the intermediate scattering function, using that for infinite time the relation

$$\langle\exp[-i\mathbf{q}\cdot\mathbf{R}_\alpha(0)]\exp[i\mathbf{q}\cdot\mathbf{R}_\alpha(t)]\rangle = \langle|\exp[i\mathbf{q}\cdot\mathbf{R}_\alpha]|^2\rangle \tag{4.181}$$

holds. In this way the computation of the *EISF* is reduced to the computation of a static thermal average. We remark at this point that the length of the MD trajectory from which the *EISF* is computed should be long enough to allow for a representative sampling of the conformational space.

*n*MOLDYN allows one to compute the elastic incoherent structure factor on a grid of equidistantly spaced points along the q-axis:

$$EISF(q_m) \doteq \sum_{I=1}^{N_{species}} n_I \omega_I EISF_I(q_m), m = 0 \ldots N_q - 1. \tag{4.182}$$

where N_q is a user-defined number of q-shells, the values for q_m are defined as $q_m = q_{min} + m \cdot \Delta q$, and for each specie the following expression for the elastic incoherent scattering function is:

$$EISF_I(q_m) = \frac{1}{n_I} \sum_{\alpha}^{n_I} \overline{\langle |\exp[i\mathbf{q} \cdot \mathbf{R}_\alpha]|^2 \rangle}^q. \tag{4.183}$$

Here the symbol $\overline{\ldots}^q$ denotes an average over the q-vectors having the same modulus q_m. The program corrects the atomic input trajectories for jumps due to periodic boundary conditions.

Parameters

Pressing the **Elastic Incoherent Structure Factor** button will pop up the dialog shown on figure 4.55

The following input fields controls the parameters for the *EISF* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

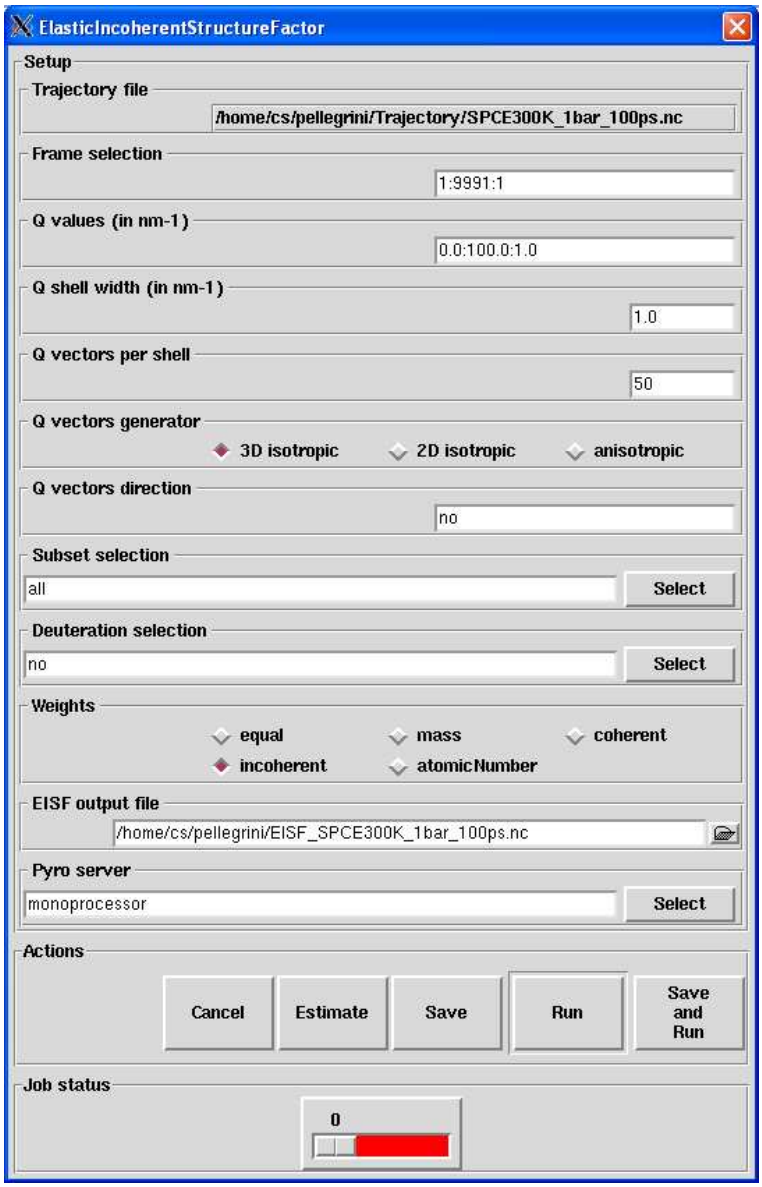

Figure 4.55: The dialog from where the *EISF* analysis will be set up and run.

- **Q values (in nm-1)**
  Format: string
  Default: *0:100:1.*
  Description: this widget allows to select the modulii of the q-vectors. This must be a string of the form:

  $q_{min} : q_{max} : q_{step}$

  In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max}

is the radius of the biggest q-shell.

For example,

- ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **Q shell width (in nm-1)**
  Format: strictly positive float
  Default: *1.0*
  Description: this widget allows to define the tolerance dq on the q-modulii. So, for each q-shell of modulus q, nMOLDYN will accept a q-vector to belong to that shell if its modulus falls in the range *[q-dq/2,q+dq/2]*. This parameter fix the q-resolution.

- **Q vectors per shell**
  Format: strictly positive integer
  Default: *50*
  Description: this widget allows to specify the number of q-vectors , N_q to generate for each q-shell. Indeed, when generating q-vectors, nMOLDYN will try to generate N_q q-vectors for each q-shell in order to carry out the averages of Eq. 4.151. For a given q-shell, if nMOLDYN could generate less q-vectors than N_q, it will only use the number of generated q-vectors instead of N_q and if nMOLDYN could generate more q-vectors than N_q, it will pick up randomly N_q q-vectors among the generated q-vectors. The higher this parameter is the smoother will be the computed intermediate scattering function but at the cost of a slower analysis.

- **Q vectors generator**
  Format: string equal to *3D isotropic*, *2D isotropic* or *anisotropic*
  Default: *3D isotropic*
  Description: this option allows to specify how the q-vectors should be generated:
  - ⋆ 3D isotropic the q-vectors are generated randomly on concentric spheres.
  - ⋆ 2D isotropic the q-vectors are generated randomly on concentric rings in a given plane.
  - ⋆ anisotropic the q-vectors are generated randomly on one or several defined directions.

- **Q vectors direction**
  Format: string
  Default: *no*
  Description: this widget allows to specify one or several preferential directions along which the q-vectors have to be generated. Depening on the q-vectors generation type, the entered value will take different values:
  - ⋆ 3D isotropic the default value *no* must be used.

⋆ 2D isotropic a string of the form

$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z$

where $q1_x, q1_y, q1_z$ and $q2_x, q2_y, q2_z$ are respectively the x,y,z components of q-vector ***q1*** and ***q2***, ***(q1,q2)*** defining the plane on which the q-vectors will be generated.

⋆ anisotropic a string of the form

$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z; \ldots$

where $q1_x, q1_y, q1_z$, $q2_x, q2_y, q2_z$ ... are respectively the x,y,z components of q-vector ***q1***, ***q2*** ..., the q-vector generation being performed along each defined direction.

- **FFT window**
  Format: float in [0.0,100.0]
  Default: *10.0*
  Description: this widget allows to define the width in percentage of the trajectory length of the Gaussian function to be used in the smoothing procedure for the calculation of the coherent structure factor out of the intermediate scattering function. See Appendix A for more details.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *incoherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *DISF*. See Section 4.2.1 for more details.

- **EISF output file**
  Format: string
  Default: *EISF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *EISF* analysis. A CDL version of the NetCDF output file is also automatically created with *EISF_traj_file.cdl* name.

Output

The results of a *EISF* analysis are stored in a NetCDF file whose main variables are namely:

- octan: an array storing the codes for space octan. For example, X+Y+Z+ for the space octan corresponding to positive X, Y and Z,
- qvectors_statistics: array storing the number of q-vectors generated per space octan,
- q: the q-shells radii in nm^{-1},
- eisf-total: the total dynamic incoherent structure factor,
- eisf-X: the partial dynamic incoherent structure factor for specie X.

4.2.5.8 Static Coherent Structure Factor

Theory and implementation

This analysis is a shortcut to obtain the static coherent structure factor defined as $S(q) = \mathcal{F}_{\text{coh}}(q, t = 0)$. It uses exactly the same procedure as the one defined in Section 4.2.5.2.

Parameters

Pressing the **Static Coherent Structure Factor** button will pop up the dialog shown on figure 4.56

The following input fields controls the parameters for the **S**tatic **C**oherent **S**tructure **F**actor (*SCSF*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

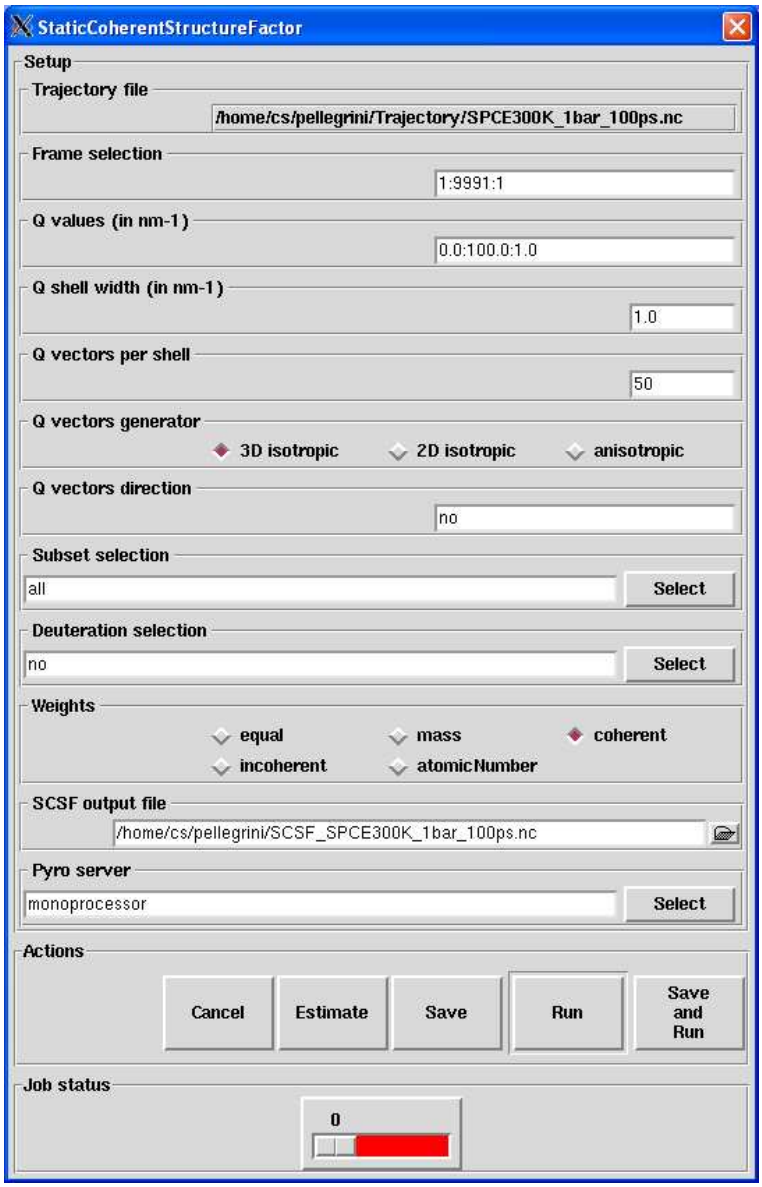

Figure 4.56: The dialog from where the *SCSF* analysis will be set up and run.

- **Q values (in nm-1)**
  Format: string
  Default: *0:100:1.*
  Description: this widget allows to select the modulii of the q-vectors. This must be a string of the form:

  $q_{min} : q_{max} : q_{step}$

  In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max}

is the radius of the biggest q-shell.

For example,

- ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **Q shell width (in nm-1)**
  Format: strictly positive float
  Default: *1.0*
  Description: this widget allows to define the tolerance dq on the q-modulii. So, for each q-shell of modulus q, nMOLDYN will accept a q-vector to belong to that shell if its modulus falls in the range *[q-dq/2,q+dq/2]*. This parameter fix the q-resolution.

- **Q vectors per shell**
  Format: strictly positive integer
  Default: *50*
  Description: this widget allows to specify the number of q-vectors , N_q to generate for each q-shell. Indeed, when generating q-vectors, nMOLDYN will try to generate N_q q-vectors for each q-shell in order to carry out the averages of Eq. 4.151. For a given q-shell, if nMOLDYN could generate less q-vectors than N_q, it will only use the number of generated q-vectors instead of N_q and if nMOLDYN could generate more q-vectors than N_q, it will pick up randomly N_q q-vectors among the generated q-vectors. The higher this parameter is the smoother will be the computed intermediate scattering function but at the cost of a slower analysis.

- **Q vectors generator**
  Format: string equal to *3D isotropic*, *2D isotropic* or *anisotropic*
  Default: *3D isotropic*
  Description: this option allows to specify how the q-vectors should be generated:
  - ⋆ 3D isotropic the q-vectors are generated randomly on concentric spheres.
  - ⋆ 2D isotropic the q-vectors are generated randomly on concentric rings in a given plane.
  - ⋆ anisotropic the q-vectors are generated randomly on one or several defined directions.

- **Q vectors direction**
  Format: string
  Default: *no*
  Description: this widget allows to specify one or several preferential directions along which the q-vectors have to be generated. Depening on the q-vectors generation type, the entered value will take different values:
  - ⋆ 3D isotropic the default value *no* must be used.

⋆ 2D isotropic a string of the form

$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z$

where $q1_x, q1_y, q1_z$ and $q2_x, q2_y, q2_z$ are respectively the x,y,z components of q-vector ***q1*** and ***q2***, *(**q1**,**q2**)* defining the plane on which the q-vectors will be generated.

⋆ anisotropic a string of the form

$q1_x, q1_y, q1_z; q2_x, q2_y, q2_z; \ldots$

where $q1_x, q1_y, q1_z$, $q2_x, q2_y, q2_z$ ... are respectively the x,y,z components of q-vector ***q1***, ***q2*** ..., the q-vector generation being performed along each defined direction.

- **FFT window**
  Format: float in [0.0,100.0]
  Default: *10.0*
  Description: this widget allows to define the width in percentage of the trajectory length of the Gaussian function to be used in the smoothing procedure for the calculation of the coherent structure factor out of the intermediate scattering function. See Appendix A for more details.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *coherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *SCSF*. See Section 4.2.1 for more details.

- **SCSF output file**
  Format: string
  Default: *SCSF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *SCSF* analysis. A CDL version of the NetCDF output file is also automatically created with *SCSF_traj_file.cdl* name.

Output

The results of a *SCSF* analysis are stored in a NetCDF file whose main variables are namely:

- octan: an array storing the codes for space octan. For example, X+Y+Z+ for the space octan corresponding to positive X, Y and Z.
- qvectors_statistics: array storing the number of q-vectors generated per space octan,
- q: the q-shells radii in nm^{-1},
- time: the times in ps at which the intermediate coherent scattering function is evaluated,
- frequency: the frequencies in THz at which the coherent structure factor is evaluated,
- Sqw-total: the total dynamic coherent structure factor,
- Sqw-XY: the partial dynamic coherent structure factor for species X and Y.

4.2.5.9 Smoothed Static Coherent Structure Factor

Theory and implementation

This analysis differs from most of the other scattering-related analysis available in nMOLDYN in the sense that it does not use a discrete q-vectors generation but results from a integration over all the q-vectors for a given q-shell. In that context, the static coherent structure factor is defined as:

$$S_{\text{coh}}(q) = \sum_{I,J \geq I}^{N_{species}} \sqrt{n_I n_J \omega_{I,\text{coh}} \omega_{J,\text{coh}}} S_{IJ}(q) \tag{4.184}$$

where $N_{species}$ is the number of selected species, n_I, n_J are respectively the number of atoms of species I and J, $\omega_{I,\text{coh}}$ and $\omega_{J,\text{coh}}$ are respectively the weights for species I and J (see Section 4.2.1 for more details) and:

$$S_{IJ,\text{coh}} = \delta_{IJ} + \frac{1}{\sqrt{n_I n_I}} \left\langle \sum_{\alpha,\beta \neq \alpha}^{n_I,n_J} \frac{sin(q r_{\alpha\beta})}{q r_{\alpha\beta}} \right\rangle \tag{4.185}$$

where q is the radius of the q-shell under process, and $r_{\alpha\beta}$ is the distance between atoms α and β. For more detials about **S**moothed **S**tatic **C**oherent **S**tructure **F**actor (*SSCSF*) analysis please refer to Ref. [69]

Parameters

Pressing the **Smoothed Static Coherent Structure Factor** button will pop up the dialog shown on figure 4.57

The following input fields controls the parameters for the *SSCSF* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

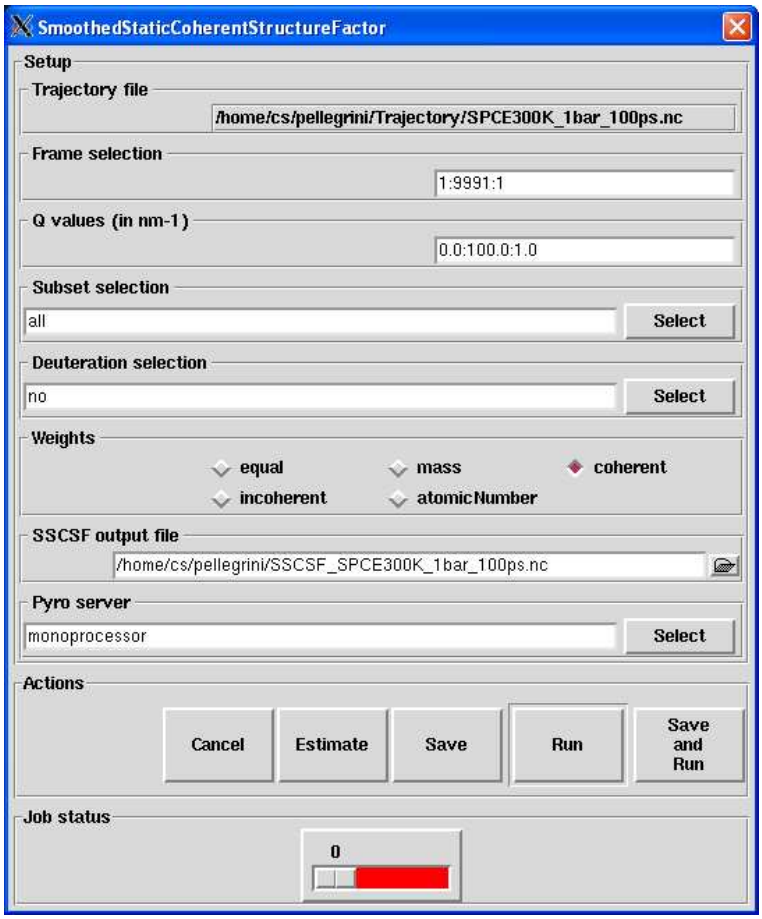

Figure 4.57: The dialog from where the *SSCSF* analysis will be set up and run.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Q values (in nm-1)**
  Format: string

Default: *0:100:1.*
Description: this widget allows to select the modulii of the q-vectors. This must be a string of the form:

$q_{min} : q_{max} : q_{step}$

In this way, the intermediate scattering function will be calculated for discrete q defined as $q_m = q_{min} + m \cdot q_{step}$ where q_{min} is the radius of the smallest q-shell, q_{step} is the distance between two consecutive q-shells and with m running from 0 to N_{shell} where N_{shell} is the number of selected q-shells defined as $N_{shell} = E(\frac{q_{max}-q_{min}}{q_{step}}) + 1$ where q_{max} is the radius of the biggest q-shell.

For example,

- ⋆ *0:10:1* will generate q-shells of radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- ⋆ *3:12:2* will generate q-shells of radii 3, 5, 7, 9, 11.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *coherent*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *SSCSF*. See Section 4.2.1 for more details.

- **SSCSF output file**
  Format: string
  Default: *SSCSF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *SSCSF* analysis. A CDL version of the NetCDF output file is also automatically created with *SSCSF_traj_file.cdl* name.

Output
The results of a *SSCSF* analysis are stored in a NetCDF file whose main variables are namely:

- q: the q-shells radii in nm^{-1}

- Sq-total: the total dynamic coherent structure factor
- Sq-XY: the partial dynamic coherent structure factor for species X and Y

4.2.6 The Structure menu

Pressing the button **Structure** brings up a menu from which it is possible to choose the following analysis:

- Pair Distribution Function
- Coordination Number
- Spatial Density
- ScrewFit Analysis

4.2.6.1 Pair Distribution Function

Theory and implementation

The **P**air-**D**istribution **F**unction (*PDF*) is an example of a pair correlation function, which describes how, on average, the atoms in a system are radially packed around each other. This proves to be a particularly effective way of describing the average structure of disordered molecular systems such as liquids. Also in systems like liquids, where there is continual movement of the atoms and a single snapshot of the system shows only the instantaneous disorder, it is extremely useful to be able to deal with the average structure.

The *PDF* is useful in other ways. For example, it is something that can be deduced experimentally from x-ray or neutron diffraction studies, thus providing a direct comparison between experiment and simulation. It can also be used in conjunction with the interatomic pair potential function to calculate the internal energy of the system, usually quite accurately.

Mathematically, the *PDF* can be computed using the following formula:

$$PDF(r) = \sum_{I=1,J\geq I}^{N_{species}} n_I n_J \omega_I \omega_J g_{IJ}(r) \tag{4.186}$$

where $N_{species}$ is the number of selected species, n_I and n_J are respectively the numbers of atoms of species I and J, ω_I and ω_J respectively the weights for species I and J (see Section 4.2.1 for more details) and $PDF_{\alpha\beta}(r)$ is the partial *PDF* for I and J species that can be defined as:

$$PDF_{IJ}(r) = \frac{\langle \sum_{\alpha=1}^{n_I} n_{\alpha J}(r) \rangle}{n_I \rho_J 4\pi r^2 dr} \tag{4.187}$$

where ρ_J is the density of atom of specie J and $n_{\alpha J}(r)$ is the mean number of atoms of specie J in a shell of width dr at distance r of the atom α of specie I.

From the computation of *PDF*, two related quantities are computed in nMOLDYN, the **R**adial-**D**istribution **F**unction (*RDF*) defined as:

$$RDF(r) = 4\pi r^2 \rho_0 PDF(r) \tag{4.188}$$

and the **T**otal-**C**orrelation **F**unction (*TCF*) defined as:

$$TCF(r) = 4\pi r \rho_0 (PDF(r) - 1.0) \tag{4.189}$$

where ρ_0 is the average atomic density defined as:

$$\rho_0 = \frac{N}{V} \tag{4.190}$$

where N is the total number of atoms of the system and V the volume of the simulation box.

In nMOLDYN, the *PDF*, the *RDF* and the *TCF* are further splitted into an intra-and inter-molecular parts which added together give respectively the total *PDF*, *RDF* and *TCF*.

Parameters

Pressing the **Pair Distribution Function** button will pop up the dialog shown on figure 4.58

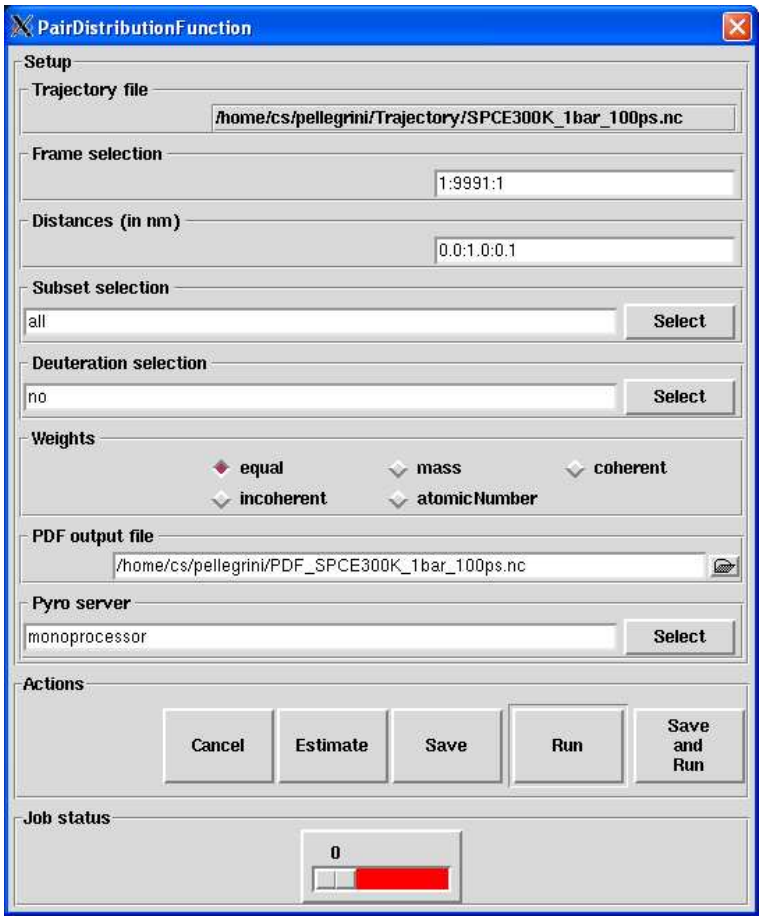

Figure 4.58: The dialog from where the *PDF* analysis will be set up and run.

The following input fields controls the parameters for the *PDF* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Distances (in nm)**
  Format: string
  Default: *0.0:1.0:0.1.*
  Description: this widget allows to select distances in *nm* at which the *PDF* will be computed. This must be a string of the form:

  $r_{min} : r_{max} : r_{step}$

  In this way, the *PDF*, the *RDF* and the *TCF* will be calculated for discrete r defined as $r_m = r_{min} + m \cdot r_{step}$ where r_{min} is the smallest r , r_{step} is the distance between two consecutive r values and with m running from 0 to $N_{rvalues}$ where $N_{rvalues}$ is the number of selected r values defined as $N_{rvalues} = E(\frac{r_{max}-r_{min}}{r_{step}}) + 1$ where r_{max} is the radius of the biggest r value .

  For example,

  - ⋆ *0:10:1* will compute *PDF* for $r = 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\ nm$.
  - ⋆ *3:7:1.2* will compute *PDF* for $r = 3,\ 4.2,\ 5.4,\ 6.6\ nm$.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. See Section 4.2.2.1 for more details.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset hydrogen atoms that will take the atomic parameters of deuterium. See Section 4.2.2.2 for more details.

- **Weights**
  Format: string equal to *equal*, *mass*, *coherent*, *incoherent* or *atomicNumber*
  Default: *equal*
  Description: this widget allows the selection of the weighting scheme to apply on each atomic contribution to the *PDF*. See Section 4.2.1 for more details.

- **PDF output file**
  Format: string
  Default: *PDF_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *PDF* analysis. A CDL version of the NetCDF output file is also automatically created with *PDF_traj_file.cdl* name.

Output

The results of a *PDF* analysis are stored in a NetCDF file whose main variables are namely:

- r: the distances in *nm* at which the *PDF*, the *RDF* and the *TCF*are computed,
- pdf-XY-intra: the intramolecular partial *PDF* for species X and Y,
- pdf-XY-inter: the intermolecular partial *PDF* for species X and Y,
- pdf-XY: the partial *PDF* for species X and Y (intramolecular and intermolecular),
- rdf-XY-intra: the intramolecular partial *RDF* for species X and Y,
- rdf-XY-inter: the intermolecular partial *RDF* for species X and Y,
- rdf-XY: the partial *RDF* for species X and Y (intramolecular and intermolecular),
- tcf-XY-intra: the intramolecular partial *TCF* for species X and Y,
- tcf-XY-inter: the intermolecular partial *TCF* for species X and Y,
- tcf-XY: the partial *TCF* for species X and Y (intramolecular and intermolecular).

4.2.6.2 Coordination number

Theory and implementation

In chemistry, the **C**oordination **N**umber (*CN*) is the total number of neighbours of a central atom in a molecule or ion. The definition used in *n*MOLDYN is somewhat different and can be seen as an extension of as the former definition. Indeed, in *n*MOLDYN, the *CN* is not defined over one defined central atom but around the centers of gravity of a set of group of atoms. So, if only one group made of only atom is selected for the analysis, then, the definition is the same as the original definition. In that context, the *CN* is defined as:

$$n(r, r+dr) = \frac{1}{N_G} \sum_{g=1}^{N_G} \sum_{I=1}^{N_{species}} n_{gI}(r, r+dr) \tag{4.191}$$

where N_G is the number of groups of atoms, $N_{species}$ is the number of species found in the system and $n_{gI}(r)$ is the *CN* defined for specie I defined as the number of atoms of species I found in a shell of width dr at a distance r of the center of gravity of the group of atom g.

nMOLDYN allows one to compute the *CN* on a set of equidistantly spaced distances at different times:

$$CN(r_m) \doteq \frac{1}{N_{frames}} \frac{1}{N_G} \sum_{f=1}^{N_{frames}} \sum_{g=1}^{N_G} \sum_{I=1}^{N_{species}} CN_{gI}(r_m, t_f), \qquad m = 0 \ldots N_r - 1,\ n = 0 \ldots N_{frames} - 1. \tag{4.192}$$

where N_r and N_{frames} are respectively the number of distances and times at which the *CN* is evaluated and

$$CN_{gI}(r_m, t_f) = n_{gI}(r_m, t_f), \tag{4.193}$$

is the number of atoms of specie I found within $[r_m, r_m + dr]$ at frame f from the center of gravity of group g.

From these expression, several remarks can be done. Firstly, the Eqs 4.192 and 4.193 can be restricted to intramolecular and intermolecular distances only. Secondly, these equations can be averaged over the selected frames providing a time averaged intra and intermolecular *CN*. Finally, the same equations (time-dependent and time-averaged) can be integrated over r to provide a cumulative *CN*. nMOLDYN computes all these variations.

The concept of *CN* is useful for structure-related analysis. It can reveal for instance some packing effects that may have occured during the simulation.

Parameters

Pressing the **Coordination Number** button will pop up the dialog shown on figure 4.59

The following input fields controls the parameters for the *CN* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

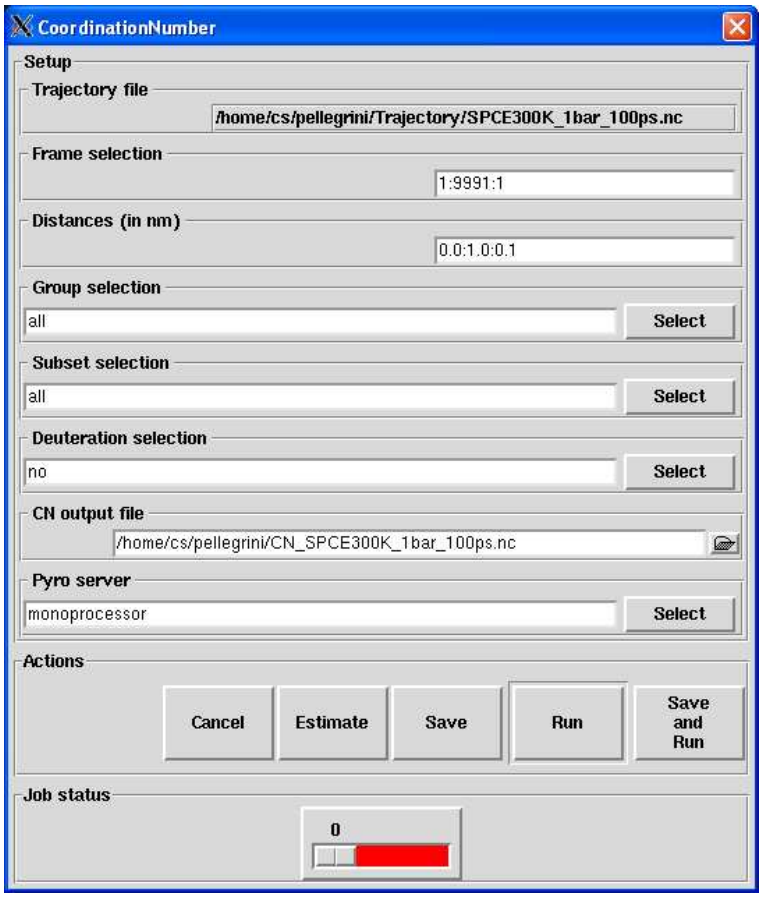

Figure 4.59: The dialog from where the *CN* analysis will be set up and run.

- **Distances (in nm)**
  Format: string
  Default: *0.0:1.0:0.1.*
  Description: this widget allows to select distances in *nm* at which the *CN* will be computed. This must be a string of the form:

  $r_{min} : r_{max} : r_{step}$

  In this way, the *CN* will be calculated for discrete r defined as $r_m = r_{min} + m \cdot r_{step}$ where r_{min} is the smallest r , r_{step} is the distance between two consecutive r values and with m running from 0 to $N_{rvalues}$ where $N_{rvalues}$ is the number of selected r values defined as $N_{rvalues} = E(\frac{r_{max}-r_{min}}{r_{step}}) + 1$ where r_{max} is the radius of the biggest r value .

  For example,

  - ⋆ *0:10:1* will compute *PDF* for $r = 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10$ *nm.*
  - ⋆ *3:7:1.2* will compute *PDF* for $r = 3,\ 4.2,\ 5.4,\ 6.6$ *nm.*

- **Group selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the groups of atoms whose centers of gravity will be used to compute the *CN* (see Section 4.2.6.2). See Section 4.2.2.3 for more details about group selection.

- **Subset selection**
  Format: subset selection string
  Default: *all*
  Description: this widget allows the selection of a subset of the system for the analysis. Only the atoms of this subset will be considered when looking around the centers of gravity of the selected groups of atoms. See Section 4.2.2.1 for more details about subset selection.

- **Deuteration selection**
  Format: deuteration selection string
  Default: *no*
  Description: this widget allows the selection of a subset of hydrogen atoms that will be considered as deuterium atoms. See Section 4.2.2.2 for more details about deuteration selection.

- **CN output file**
  Format: string
  Default: *CN_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *CN* analysis. A CDL version of the NetCDF output file is also automatically created with *CN_traj_file.cdl* name.

Output

The results of a *CN* analysis are stored in a NetCDF file whose main variables are namely:

- r: the distances in *nm* at which all the *CN* variants are computed,
- time: the times in *ps* at which all the time-dependant *CN* variants are computed,
- cn-X-intra: the intramolecular time-dependent partial *CN* for specie X,
- cn-X-inter: the intermolecular time-dependent partial *CN* for specie X,
- cn-X: the intramolecular plus intermolecular time-dependent partial *CN* for specie X,
- cn-cumul-X-intra: the intramolecular time-dependent partial cumulative *CN* for specie X,
- cn-cumul-X-inter: the intermolecular time-dependent partial cumulative *CN* for specie X,

- cn-cumul-X: the intramolecular plus intermolecular time-dependent partial cumulative *CN* for specie X,
- cn-timeavg-X-intra: the intramolecular time-avergaed partial *CN* for specie X,
- cn-timeavg-X-inter: the intermolecular time-averaged partial *CN* for specie X,
- cn-timeavg-X: the intramolecular plus intermolecular time-averaged partial *CN* for specie X,
- cn-timeavg-cumul-X-intra: the intramolecular time-averaged partial cumulative *CN* for specie X,
- cn-timeavg-cumul-X-inter: the intermolecular time-averaged partial cumulative *CN* for specie X,
- cn-timeavg-cumul-X: the intramolecular plus intermolecular time-averaged partial cumulative *CN* for specie X,
- cn-timeavg: the time-averaged total *CN*,
- cn-timeavg-cumul: the time-averaged total cumulative *CN*.

4.2.6.3 Spatial Density

Theory and implementation

The **S**patial **D**ensity (*SD*) can be seen as an generalization of the pair distribution function. Indeed, pair distribution functions are defined as orientionally averaged distribution functions. are in the sense that, Altough these correlation functions reflects many key features of the short-range order in molecular systems, it should be realized that an average spatial assembly of non-spherical particles can not be uniquely characterized from these one-dimensionals functions. So, structural models postulated for the molecular ordering in nonsimple systems based only on one-dimensional *PDF* will always be somewhat ambiguous. The goal of *SD* analysis is to provide greater clarity in the structual analysis of molecular systems by utilizing distribution function which span both the radial and angular coordinates of the separation vector. This can provide useful information about the average local structure in a complex system.

*n*MOLDYN allows one to compute the *SD* in spherical coordinates on a set of concentrics shells surrounding the centers of mass of selected triplets of atoms using the formula:

$$SD(r_l,\theta_m,\phi_n) \doteq \frac{1}{N_{triplets}N_{groups}} \sum_{t=1}^{N_{triplets}} \sum_{g=1}^{N_{groups}} \langle n_{tg}(r_l,\theta_m,\phi_n)\rangle\,, \qquad l=0\ldots N_r-1, m=0\ldots N_\theta-1, n=0\ldots N \tag{4.194}$$

where $N_{triplets}$ and N_{groups} are respectively the number of triplets and groups, r_l, θ_m and ϕ_n are the spherical coordinates at which the *SD* is evaluated, N_r, N_θ and N_ϕ are respectively the number of discrete r, θ and ϕ values and $n_{tg}(r_l,\theta_m,\phi_n)$ is the number of group of atoms of type g whose centers of mass is found to be in the volume element defined by $[r, r+dr]$, $[\theta,\theta+d\theta]$ and $[\phi,\phi+d\phi]$ in the spherical coordinates basis centered on the center of mass of triplet t.

So technically, *n*MOLDYN proceeds more or less on the following way:

- defines the center of mass $c_i^t \qquad i=1,2\ldots N_{triplets}$ for each triplet of atoms,
- defines the center of mass $c_i^g \qquad i=1,2\ldots N_{groups}$ for each group of atoms,

- constructs an oriented orthonormal basis $\mathcal{R}_i^t i = 1, 2 \ldots N_{triplets}$ centered on each c^t, this basis is defined from the three vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$,
  - ⋆ $\mathbf{v}_1 = \frac{\mathbf{n}_1+\mathbf{n}_2}{||\mathbf{n}_1+\mathbf{n}_2||}$ where $\mathbf{n}_1$ and $\mathbf{n}_2$ are respectively the normalized vectors in (**a1**,**a2**) and (**a1**,**a3**) directions where (**a1**,**a2**,**a3**) are the three atoms of the triplet t,
  - ⋆ $\mathbf{v}_2$ is defined as the clockwise normal vector orthogonal to $\mathbf{v}_1$ that belongs to the plane defined by **a1**, **a2** and **a3** atoms,
  - ⋆ $\vec{v_3} = \vec{v_1} \times \vec{v_2}$
- expresses the cartesian coordinates of each c^g in each $\mathcal{R}^t$,
- transforms these coordinates in spherical coordinates,
- discretizes the spherical coordinates in r_l, θ_m and ϕ_n,
- does $n_{tg}(r_l, \theta_m, \phi_n) = n_{tg}(r_l, \theta_m, \phi_n) + 1$.

Parameters

Pressing the **Spatial Density** button will pop up the dialog shown on figure 4.60
The following input fields controls the parameters for the *SD* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,
  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Distances (in nm)**
  Format: string
  Default: *0.0:1.0:0.1.*

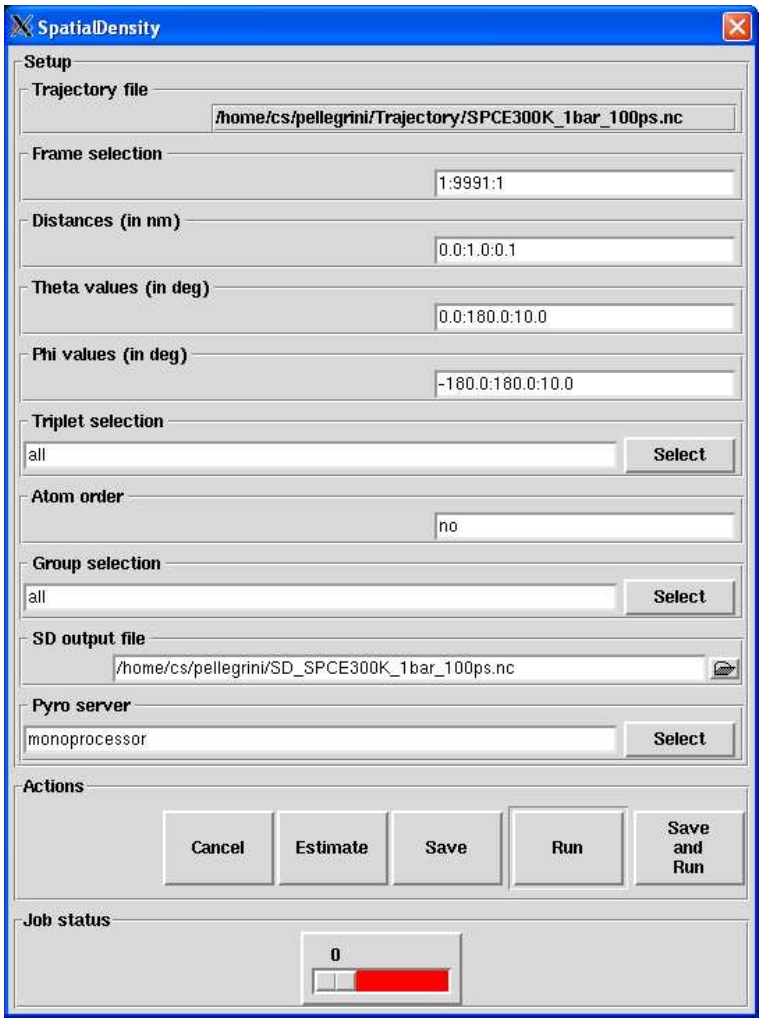

Figure 4.60: The dialog from where the *SD* analysis will be set up and run.

Description: this widget allows to select distances in *nm* at which the *CN* will be computed. This must be a string of the form:

$r_{min} : r_{max} : r_{step}$

In this way, the *CN* will be calculated for discrete r defined as $r_m = r_{min} + m \cdot r_{step}$ where r_{min} is the smallest r , r_{step} is the distance between two consecutive r values and with m running from 0 to $N_{rvalues}$ where $N_{rvalues}$ is the number of selected r values defined as $N_{rvalues} = E(\frac{r_{max} - r_{min}}{r_{step}}) + 1$ where r_{max} is the radius of the biggest r value .

For example,

- ⋆ *0:10:1* will compute *PDF* for $r = 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10$ *nm*.
- ⋆ *3:7:1.2* will compute *PDF* for $r = 3,\ 4.2,\ 5.4,\ 6.6$ *nm*.

- **Theta values (in deg)**
  Format: string
  Default: *0.0:180.0:10.0.*
  Description: this widget allows to select the spherical θ angle in deg at which the *SD* will be computed. This must be a string of the form:

  $\theta_{min} : \theta_{max} : \theta_{step}$

  In this way, the *SD* will be calculated for discrete θ values defined as $\theta_m = \theta_{min} + m \cdot \theta_{step}$ where θ_{min} is the minimum θ value, θ_{step} is the step between two consecutive θ values and with m running from 0 to N_θ where N_θ is the number of selected θ values defined as $N_\theta = E(\frac{\theta_{max} - \theta_{min}}{\theta_{step}}) + 1$ where θ_{max} is the maximum θ value.

- **Phi values (in deg)**
  Format: string
  Default: *-180.0:180.0:10.0.*
  Description: this widget allows to select the spherical ϕ angle in deg at which the *SD* will be computed. This must be a string of the form:

  $\phi_{min} : \phi_{max} : \phi_{step}$

  In this way, the *SD* will be calculated for discrete ϕ values defined as $\phi_m = \phi_{min} + m \cdot \phi_{step}$ where ϕ_{min} is the minimum ϕ value, ϕ_{step} is the step between two consecutive ϕ values and with m running from 0 to N_ϕ where N_ϕ is the number of selected ϕ values defined as $N_\phi = E(\frac{\phi_{max} - \phi_{min}}{\phi_{step}}) + 1$ where ϕ_{max} is the maximum ϕ value.

- **Triplet selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the triplets of atoms from which the analysis will be performed. See Section 4.2.2.3 for more details about this kind of selection. Any selection that does not contain exactly three atoms will be discarded.

- **Atom order**
  Format: string
  Default: *no*
  Description: this widget allows to specify the order in which the atoms **a1**, **a2** and **a3** should be ordered. By default, the order will be defined by *n*MOLDYN by ranking for the atoms of each triplet by their MMTK name. Otherwise, the entered value must have the following specific format:

  MMTK name for ***a1****,MMTK name for* ***a2****,MMTK name for* ***a3***

- **Group selection**
  Format: group selection string
  Default: *all*
  Description: this widget allows the selection of the group of atoms to consider in the analysis (see Section 4.2.6.3). See Section 4.2.2.3 for more details about group selection.

- **SD output file**
  Format: string
  Default: *SD_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *SD* analysis. A CDL version of the NetCDF output file is also automatically created with *SD_traj_file.cdl* name.

Output

The results of a *SD* analysis are stored in a NetCDF file whose main variables are namely:

- theta: the spherical θ angles in deg at which all the *SD* is computed,
- phi: the spherical ϕ angles in deg at which all the *SD* is computed,
- sd_r*val*nm: the *SD* defined for the shell of radius *r=val*.

4.2.6.4 ScrewFit analysis

Theory and implementation

The **S**crew**F**it **A**nalysis (*SFA*) is based on the superposition of molecular structures, defined by atomic coordinates, combining quaternionic representation of rotation matrices and Chasles' theorem on rigid-body displacements. When applied to subsequent peptide planes in protein structures, *SFA* gives local helical parameters of the protein backbone winding.

SFA takes into consideration three structural parameters: the orientational distance, the helix radius of screw motion, and the straightness. The orientational distance, Δ, refers to the relative orientation of two subsequent peptide planes. The radius of screw motion, ρ, is related to the distances between the C-atoms, in the reference and target peptide planes, and the local axis of screwmotion. The radius ρ is a measure of the local curling of the backbone conformation. In a flat backbone conformation, like an extended β-strand, ρ is close to zero ; when the local backbone conformation is curled, as in the case of α-helices and β-turns, ρ increases. The maximum observed values usually do not exceed 0.3. Straightness, σ, is the scalar product between the unit vectors of the axis of screwmotion, relative to four consecutive peptide planes, it gives information about local curvatures or kinks of a protein backbone tract. For each pair of consecutive peptide planes represented by their atoms C, O, N, ScrewFit defines these three parameters indicating their relative orientation and distance from a common axis of rotation (the axis of screw motion)

Given the definition of *SFA* your system must contain at least one protein to perform this analysis.

For more detailed and technical information about this analysis please refers to the original paper [38].

Parameters

Pressing the **ScrewFit Analysis** button will pop up the dialog shown on figure 4.61

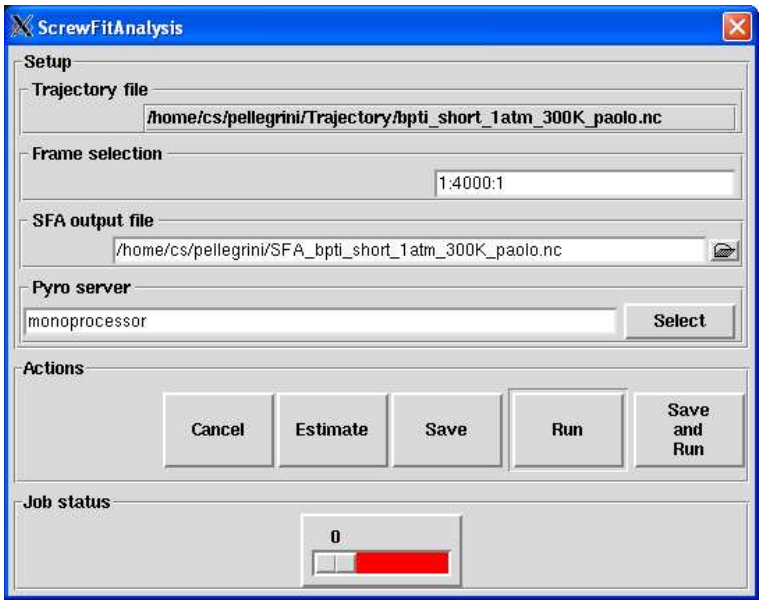

Figure 4.61: The dialog from where the *SFA* analysis will be set up and run.

The following input fields controls the parameters for the *SFA* analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **SFA output file**
  Format: string
  Default: *SFA_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *SFA* analysis. A CDL version of the NetCDF output file is also automatically created with *SFA_traj_file.cdl* name.

Output

The results of a *SD* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which all the time-dependant *SFA* is computed,

and for each protein chain *name* found in the protein

- *name*_straightness_y: variables equals to the length of chain *name* - 4,
- *name*_helixradius_y: variables equals to the length of chain *name* - 3,
- *name*_orientdist_y: variables equals to the length of chain *name* - 1,
- *name*_straightness: the straightness,
- *name*_helixradius: the helix radius,
- *name*_orientdist: the orientional distance.

4.2.7 The NMR menu

Pressing the button **NMR** brings up a menu from which it is possible to choose the following analysis:

- Order Parameter
- Order Parameter (Contact Model)

4.2.7.1 Order Parameter

Theory and implementation

Adequate and accurate cross comparison of the NMR and MD simulation data is of crucial importance in versatile studies conformational dynamics of proteins. NMR relaxation spectroscopy has proven to be a unique approach for a site-specific investigation of both global tumbling and internal motions of proteins. The molecular motions modulate the magnetic interactions between the nuclear spins and lead for each nuclear spin to a relaxation behavior which reflects its environment. Since its first applications to the study of protein dynamics, a wide variety of experiments has been proposed to investigate backbone as well as side chain dynamics. Among them, the heteronuclear relaxation measurement of amide backbone ^{15}N nuclei is one of the most widespread techniques. The relationship between microscopic motions and measured spin relaxation rates is given by Redfield's theory [70]. Under the hypothesis that ^{15}N relaxation occurs through dipole-dipole interactions with the directly bonded ^{1}H atom and chemical shift anisotropy (CSA), and assuming that the tensor describing the CSA is axially symmetric with its axis parallel to the N-H bond, the relaxation rates of the ^{15}N nuclei are determined by a time correlation function,

$$C_{ii}(t) = \langle P_2(\mu_i(0) \cdot \mu_i(t)) \rangle \tag{4.195}$$

which describes the dynamics of a unit vector $\mu_i(t)$ pointing along the ^{15}N-^{1}H bond of the residue i in the laboratory frame. Here $P_2(.)$ is the second order Legendre polynomial. The Redfield theory shows that relaxation measurements probe the relaxation dynamics of a selected nuclear spin only at a few frequencies. Moreover, only a limited number of independent observables are accessible. Hence, to relate relaxation data to protein dynamics one has to postulate either a dynamical model for molecular motions or a functional form for $C_{ii}(t)$, yet depending on a limited number of adjustable parameters. Usually, the tumbling motion of proteins in solution is assumed isotropic and uncorrelated with the internal motions, such that:

$$C_{ii}(t) = C^G(t) \cdot C^I_{ii}(t) \tag{4.196}$$

where $C^G(t)$ and $C^I_{ii}(t)$ denote the global and the internal time correlation function, respectively. Within the so-called model free approach [71, 72] the internal correlation function is modeled by an exponential,

$$C^I_{ii}(t) = S_i^2 + (1 - S_i^2) exp\left(-\frac{t}{\tau_{eff,i}}\right) \tag{4.197}$$

Here the asymptotic value $S_i^2 = C_{ii}(+\infty)$ is the so-called generalized order parameter, which indicates the degree of spatial restriction of the internal motions of a bond vector, while the characteristic time $\tau_{eff,i}$ is an effective correlation time, setting the time scale of the internal relaxation processes. S_i^2 can adopt values ranging from 0 (completely disordered) to 1 (fully ordered). So, S_i^2 is the appropriate indicator of protein backbone motions in computationally feasible timescales as it describes the spatial aspects of the reorientational motion of N-H peptidic bonds vector.

When performing Order Parameter analysis, nMOLDYN computes for each residue i both $C_{ii}(t)$ and S_i^2. It also computes a correlation function averaged over all the selected bondsdefined as:

$$C^I(t) = \sum_{i=1}^{N_{bonds}} C^I_{ii}(t) \tag{4.198}$$

where N_{bonds} is the number of selected bonds for the analysis.

Parameters

Pressing the **Order Parameter** button will pop up the dialog shown on figure 4.62

The following input fields controls the parameters for the **O**rder **P**arameter (*OP*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

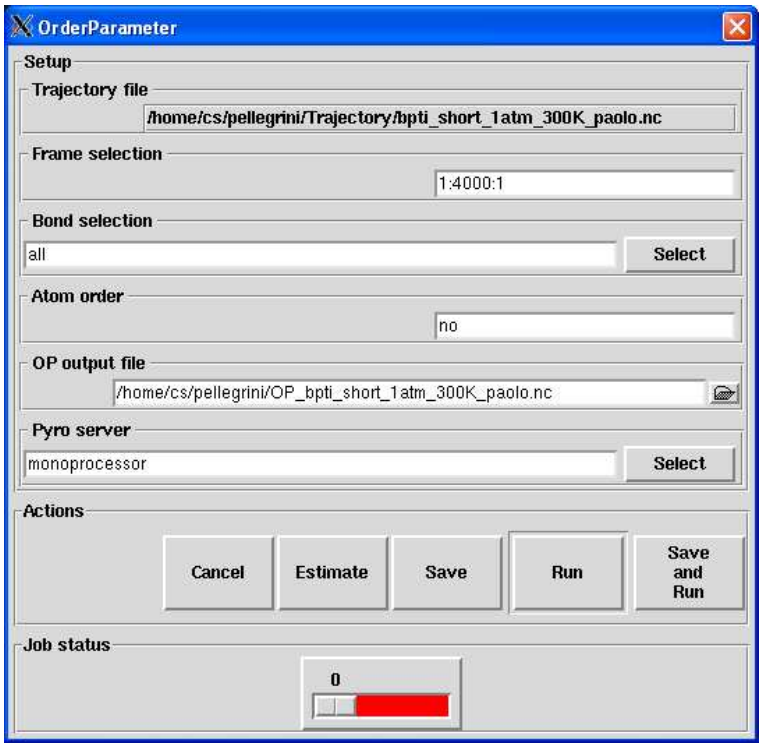

Figure 4.62: The dialog from where the *OP* analysis will be set up and run.

where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

For example,

- ⋆ *2:10:3* will select the frames 2, 5 and 8.
- ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **Bond selection**
  Format: string
  Default: *all*
  Description: this widget allows the selection of the pairs of atoms from which Eq. 4.195 will be computed. See Section 4.2.2.3 for more details. Any selection that does not contain exactly two atoms will be discarded.

- **Atom order**
  Format: string
  Default: *no*
  Description: this widget allows to specify the order in which the atoms **a1**, **a2** should be ordered. By default, the order will be defined by *n*MOLDYN by ranking for the atoms of each bond by their MMTK name. Otherwise, the entered value must have the following

specific format:

MMTK name for ***a1****,MMTK name for* ***a2***

- **OP output file**
  Format: string
  Default: *OP_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *OP* analysis. A CDL version of the NetCDF output file is also automatically created with *OP_traj_file.cdl* name.

Output

The results of an *OP* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which all the time-dependent *OP* is computed,
- bond: the index of the bonds selected for the analysis. In case where the system under study is the a protein or a peptide chain, this index is simply the sequence number of the first atom forming the bond, otherwise it is just an integer ranging from 1 to N_{bonds} where N_{bonds} is the number of selected bonds,
- p2: the order parameter P2 defined in Eq. 4.195,
- p2-bondavg: the bond-averaged P2 defined in Eq. 4.198,
- s2: the generalized order parameter defined above in Section 4.2.7.1.

4.2.7.2 Order Parameter (Contact Model)

Theory and implementation

This analysis is based on an analytical relationship for the estimation of the generalized order parameter S_i^2 (see Section 4.2.7.1) of N-H vectors of the protein backbone. It related S_i^2 of the N-H vector of residue i to close contact experienced by the H atom and the carbonyl oxygen of the preceeding residue *i-1* with heavy atoms k using the formula:

$$S_i^2(t) = tanh\left(2.656\sum_k\left((exp(-r^O_{i-1,k}(t)))) + 0.8exp(-r^H_{i,k}(t))\right)\right) + b \tag{4.199}$$

where $r^O_{i-1,k}$ is the distance between the carbonyl oxygen of residue *i-1* and heavy atom k and $r^H_{i,k}$ is the distance between the amide proton of residue i and heavy atom k. The parameter b is set to -0.1 which takes into account that order parameters of rigid protein regions typically lie around 0.9. The sum ranges over all heavy atoms k that do not belong to amino acids i and *i-1*. For more details about this method, please refer to Ref. [73]

Beside the time-dependent $S_i^2(t)$ defined in Eq; 4.199, nMOLDYN also provide a time-averaged S_i^2 defined as:

$$S_i^2 = \frac{1}{N_{frames}}\sum_{i=1}^{N_{frames}} S_i^2(t_i) \tag{4.200}$$

where N_{frames} is the number of selected frames for the analysis.

Parameters

Pressing the **Order Parameter (Contact Model)** button will pop up the dialog shown on figure 4.63

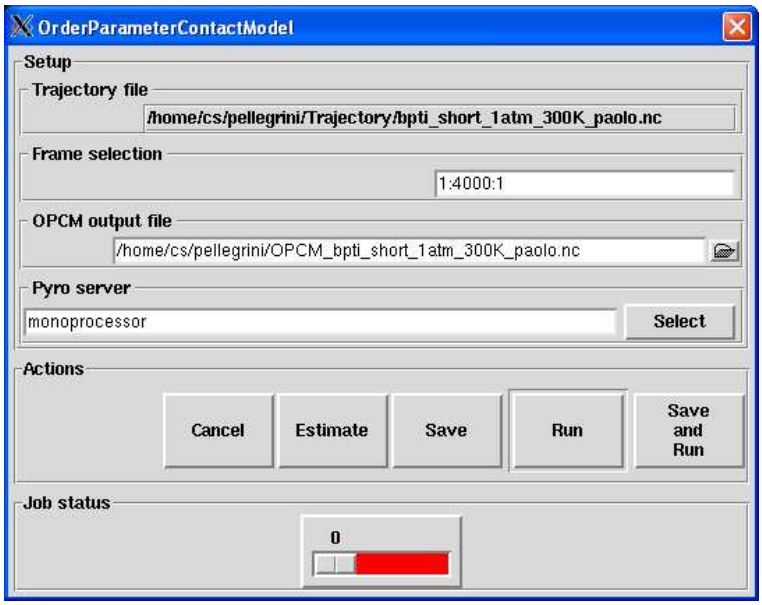

Figure 4.63: The dialog from where the *OPCM* analysis will be set up and run.

The following input fields controls the parameters for the **O**rder **P**arameter using **C**ontact **M**odel (*OPCM*) analysis:

- **Trajectory file**
  Format: string
  Default: *traj_file* where *traj_file* is the name of the loaded trajectory
  Description: the value of this widget can not be changed. It just recalls for information purpose the name of the trajectory file loaded for the analysis.

- **Frame selection**
  Format: string
  Default: *1:traj_length:1* where *traj_length* is the number of frames of the trajectory.
  Description: this widget allows to select the trajectory frames that will be used for the analysis. This must be a string of the form:

  first:last:step

  where *first* is an integer specifying the first frame number to consider, *last* is an integer specifying the last frame number to consider and *step* is an integer specifying the step number between two frames.

  For example,

  - ⋆ *2:10:3* will select the frames 2, 5 and 8.
  - ⋆ *1:5:1* will select the frames 1, 2, 3, 4 and 5.

- **OPCM output file**
  Format: string
  Default: *OPCM_traj_file.nc* where *traj_file.nc* is the name of the input trajectory
  Description: this widget allows to enter the name of the NetCDF output file of the *OPCM* analysis. A CDL version of the NetCDF output file is also automatically created with *OPCM_traj_file.cdl* name.

Output

The results of an *OPCM* analysis are stored in a NetCDF file whose main variables are namely:

- time: the times in *ps* at which all the time-dependent *OPCM* defined in Eq. 4.199 is computed,

and for each protein chain *name* found in the system

- *name*_sequence: the residue numbers of the chain *name*,
- *name*_s2: the time-dependent generalized order parameter defined for each residue of chain *name* as defined in Eq. 4.199,
- *name*_s2_timeavg: the time-averaged generalized order parameter defined for each residue of chain *name* as defined in Eq. 4.200.

4.3 The View menu

Pressing the button **View** brings up a menu from which it is possible to choose the following options:

- Plot
- Animation
- Effective Mode

All of these functions will be descrived below.

4.3.1 Plot

This option allows one to plot the results of almost any *n*MOLDYN analysis.

Pressing the **Plot** button will pop up the dialog shown on figure 4.64

The following input fields controls the parameters used to plot variables coming from a NetCDF file:

- **NetCDF input file**
  Format: string
  Default: the loaded NetCDF filename if one is already currently loaded. Nothing otherwise
  Description: this widget allows to specify the name of the NetCDF file whose contents should be plotted.

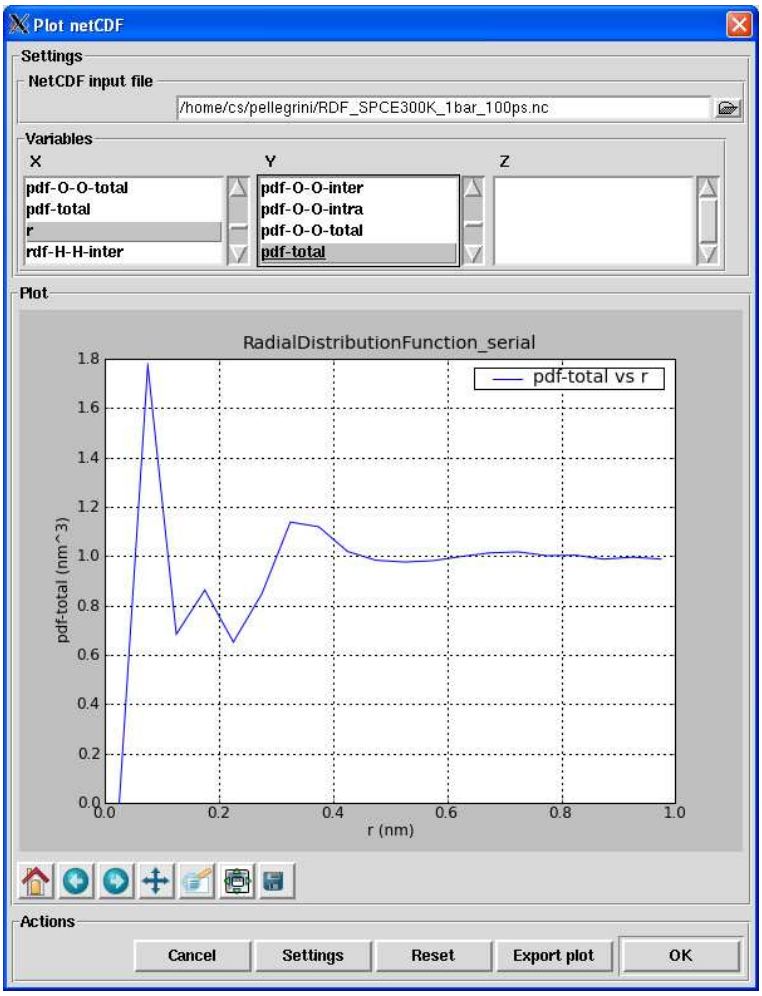

Figure 4.64: The dialog from where the results of *n*MOLDYN analysis can be plotted.

- **Variables**
  Format: not an editable etry
  Default: no variable selected
  Description: this widget allows to specify the NetCDF variables that should be plotted. When loading the NetCDF file, *n*MOLDYN distinguish between 1D and 2D arrays. The 1D arrays are stored as X and Y variables and displayed into their corresponding **X** and **Y** variables listboxes. The 2D arrays are stored as Z variables and displayed into their corresponding **Z** variables listbox. In order to select for plotting a set of NetCDF variables, click first on the X variable, then on the Y variable and eventually on the Z variable you want to plot.

- **Plot**
  Format: not an editable etry
  Default: no plot displayed
  Description: this widget contain the space allocated to display the plots. The plots are generated using matplotlib python graphical library. At the botton of the widget, there are seven buttons that allows to perform various actions on the displayed plot. These

buttons are the following:

-  takes you to the first view.
-  the `Back` is akin to the web browser `Back` button. It is used to navigate back between previously defined views. It has no meaning unless, the `Pan/Zoom` and `Zoom to rect mode` modes, defined below, have been used. This is analogous to trying to click `Back` on your web browser before visiting a new page. Nothng happens.
-  the `Forward` is akin to the web browser `Forward` button. It is used to navigate forward between previously defined views. It has no meaning unless, the `Pan/Zoom` and `Zoom to rect mode` modes, defined below, have been used. This is analogous to trying to click `Forward` on your web browser before visiting a new page. Nothing happens.
-  activates the `Pan/Zoom` mode. The `Pan/Zoom` button has two modes: pan and zoom. Then put your mouse somewhere over an axes.
  - Pan mode: press the left mouse button and hold it, dragging it to a new position. When you release it, the data under the point where you pressed will be moved to the point where you released. If you press 'x' or 'y' while panning, the motion will be constrained to the x or y axis, respectively.
  - Zoom mode: press the right mouse button, dragging it to a new position. The x axis will be zoomed in proportionate to the rightward movement and zoomed out proportionate to the leftward movement. Ditto for the y axis and up/down motions. The point under your mouse when you begin the zoom should remain in place, allowing you to zoom to an arbitrary point in the figure. You can use the modifier keys 'x' and 'y' or 'CONTROL' to constrain the zoom to the x axis, y axis, or aspect ratio preserve, respectively.
-  activates the `Zoom to rect` mode. Put your mouse somewhere over the plot and press the left mouse button. Drag the mouse while holding the button to a new location and release. The plots limits will be zoomed to the rectangle you have defined. There is also a 'zoom out to rectangle' in this mode with the right button, which will place your entire plot in the region defined by the rectangle you have defined.
-  pops up a dialog from which you can adjust some basic positional plot parameters.
-  pops up a file browser from which you can save the plot to a file. You can save your plot to the following format:
  - Portable Networks Graphics (.png)
  - Enhance Metafile (.emf)
  - Encapsulated Postscript (.eps)

* Portable Document Format (.pdf)
* Postscript (.ps)
* Raw RGBA bitmap (.raw,.rgba)
* Scalable Vector Graphics (.svg)

Pressing **Return** or clicking on the **OK** button will display the plot for the selected NetCDF variables. Once a plot is displayed, you can change some plot settings by clicking on **Settings** button that will pop up the dialog shown in figure 4.65.

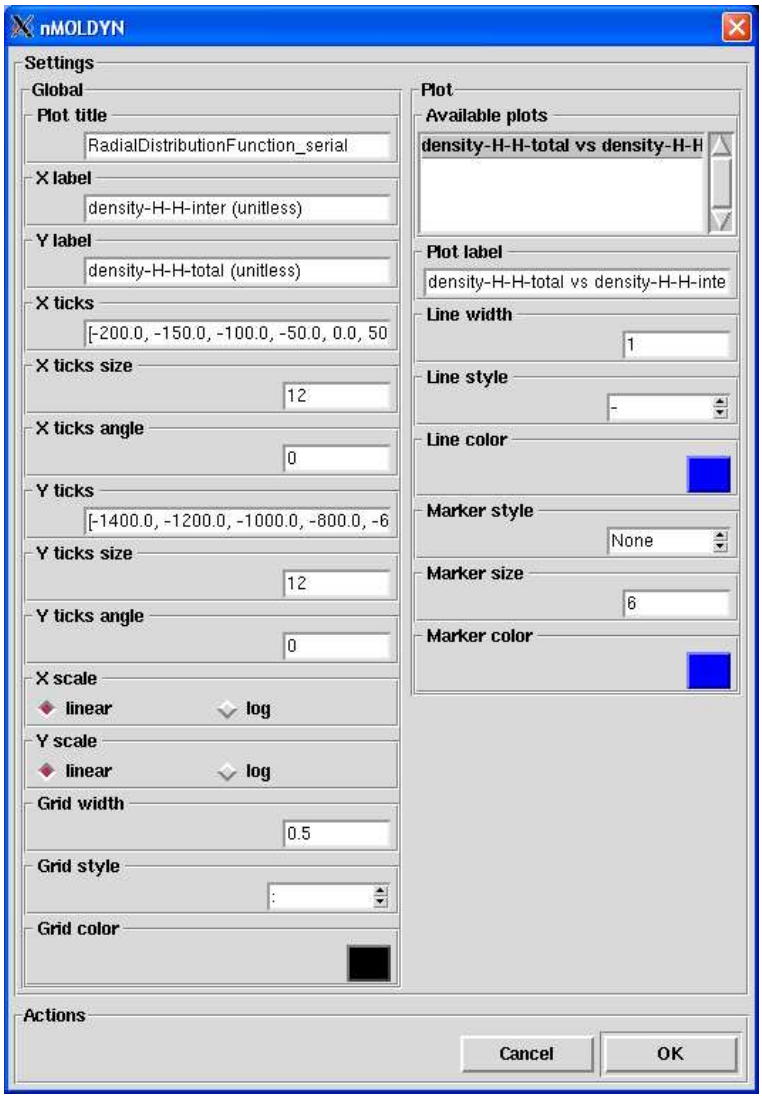

Figure 4.65: The dialog from which the plot settings are defined.

From this dialog you can change many plot parameters such as the plot title, the x and y labels, the x and y ticks size, the x and y scales, the line colors, the line width ...

You can also export the curent plot to an ASCII file by clicking on **Export plot**. This will pop up the dialog shown in figure 4.66.

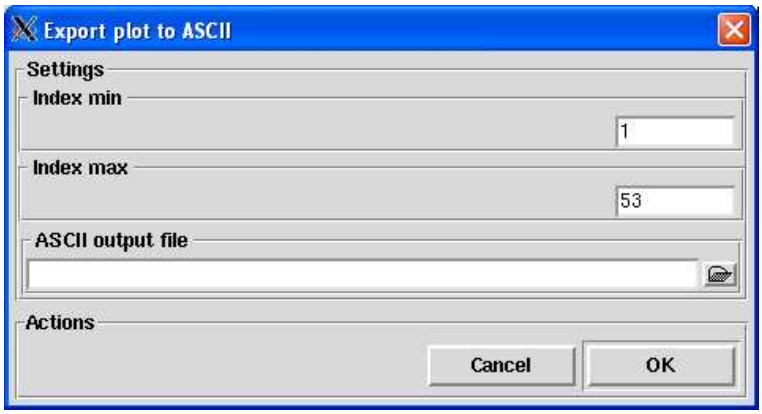

Figure 4.66: The dialog from which the current plot can be exported to an ASCII file.

From this dialog, you will be asked for the range of values to export and a file name for the output ASCII file.

Finally, pressing the **Cancel** button will close the plot dialog.

4.3.2 Animation

This option allows one to view an animation of a trajectory using **V**isual **M**olecular **D**ynamics (VMD). To do so, VMD must be installed and the preferences variables **vmd_path** storing the path for VMD reader executable must be correctly set.

Pressing the **Animation** button will pop up the dialog shown on figure 4.67

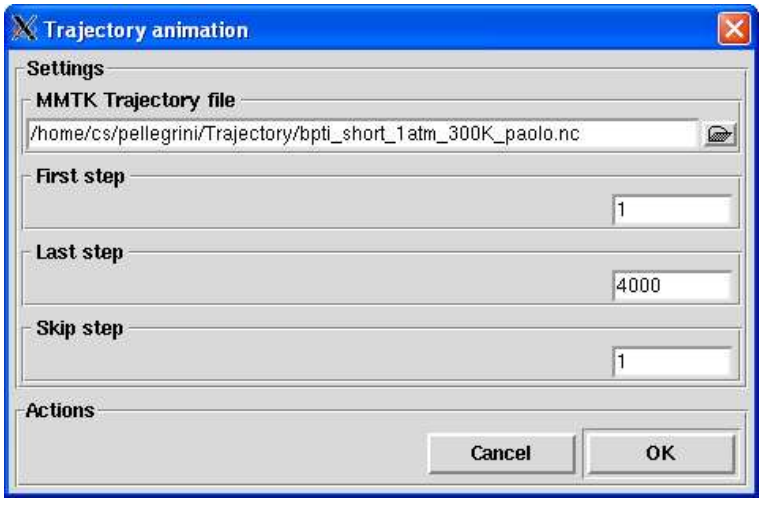

Figure 4.67: The dialog from where a trajectory can be animated using VMD

The following input fields controls the parameters to visualize an animation of a trajectory through VMD:

- **MMTK Trajectory file**
  Format: string
  Default: the loaded trajectory filename if one trajectory is currently loaded. Nothing otherwise
  Description: this widget allows to specify the name of the trajectory animation to visualize.

- **First step**
  Format: integer in [1,*traj_length*] where *traj_length* is the number of frames of the input trajectory
  Default: *1* if one trajectory is currently loaded.or nothing otherwise.
  Description: this widget allows to specify the first step of the trajectory to visualize.

- **Last step**
  Format: integer in [1,*traj_length*] where *traj_length* is the number of frames of the input trajectory
  Default: the trajectory last step if one trajectory is currently loaded or nothing otherwise.
  Description: this widget allows to specify the last step of the trajectory to visualize. Depending on the trajectory, calling VMD from *n*MOLDYN is quite time-consuming, the entered value should not be too high.

- **Skip step**
  Format: integer in [1,*traj_length*[where *traj_length* is the number of frames of the input trajectory
  Default: *1* if one trajectory is currently loaded.or nothing otherwise.
  Description: this widget allows to specify the number of trajectory steps to skip between each step to visualize.

Pressing the **OK** button will run the animation, pressing the **Cancel** button will close the dialog.

4.3.3 Effective mode

This option allows one to visualize the results of a *QHA* analysis (see Section 4.2.4.12) as an animation of a selected effective mode coming out the analysis using VMD. To do so, VMD must be installed and the preferences variables **vmd_path** storing the path for VMD reader executable must be correctly set.

Pressing the **Effective mode viewer** button will pop up the dialog shown on figure 4.68

The following input fields controls the parameters necessary to visualize the animation through VMD:

- **QHA input file**
  Format: string
  Default: the name of the *QHA* analysis output file if one such file is already currently loaded. Nothing otherwise
  Description: this widget allows to select the name of the *QHA* analysis output file to visualize.

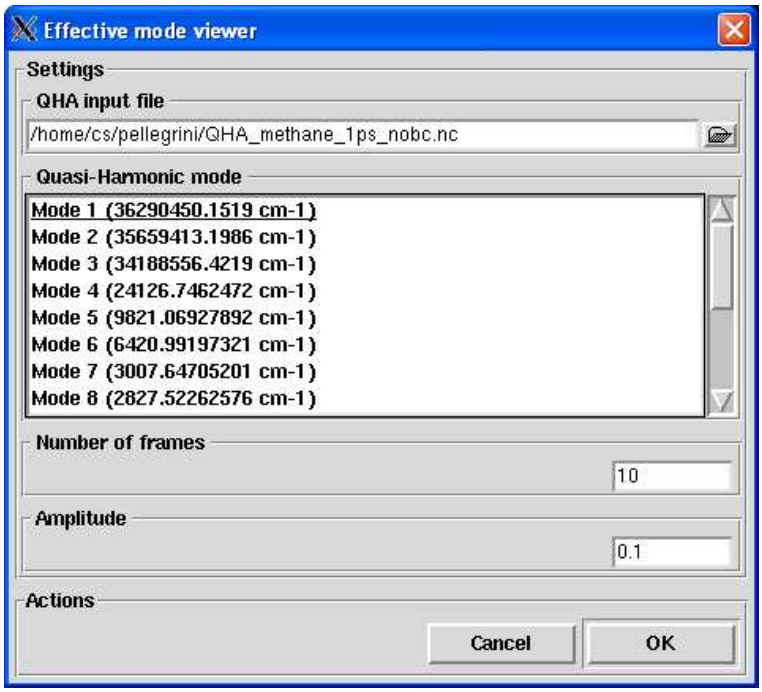

Figure 4.68: The dialog from where a the pseudo-trajectory associated to a selected effective can be animated using VMD

- **Quasi-Harmonic mode**
  Format: not an editable entry
  Default: no effective mode selected
  Description: this widget allows to select which combination of effective modes to visualize.

- **Number of frames**
  Format: strictly positive integer
  Default: *10* if one *QHA* file is currently loaded. Nothing otherwise
  Description: this widget allows to specify the number of frames the pseudo-trajectory should contain. Calling VMD from *n*MOLDYN is quite time-consuming, the entered value should not be too high.

- **Amplitude**
  Format: strictly positive float
  Default: *0.1* if one *QHA* file is currently loaded. Nothing otherwise
  Description: this widget allows to specify the amplitude in *nm* of the motion associated to the selected effective mode (see Eq. 4.109).

Pressing the **OK** button will run the animation, pressing the **Cancel** button will close the dialog.

4.4 The Help menu

Pressing the button **Help** brings up a menu from which it is possible to choose the following options:

- Documentation
- Mailing List
- API
- Analysis benchmark
- About nMOLDYN

All of these functions will be descrived below.

4.4.1 Documentation

Pressing the **Documentation** button will pop up the *n*MOLDYN users guide. Depending on the value of the preferences variable **documentation_style** (see Section 4.1.6), the users guide will be displayed either in HTML format on a webbrowser selected by default by Python either in PDF format on acrobat reader. In the later case, the preferences variables **acroread_path** storing the path for acrobat reader executable must also be correctly set.

4.4.2 Mailing List

Pressing the **Mailing List** button will open the *n*MOLDYN mailing list webpage

http://sourcesup.cru.fr/forum/?group_id=194

in a web browser selected by default by python.

4.4.3 API

Pressing the **API** button will open the *n*MOLDYN API in a web browser selected by default by python.

4.4.4 Analysis benchmark

The purpose of this option is to test the stability of most of the analysis provided by the running version of *n*MOLDYN versus the results produced by the intensively tested version 2.2.5. As *n*MOLDYN is an open source program, this option can be interesting for experienced users that may have brought some changes in a given analysis and that would like to insure that its scientific contents is still valid.

Pressing the **Analysis benchmark** button will pop up the dialog shown on figure 4.69

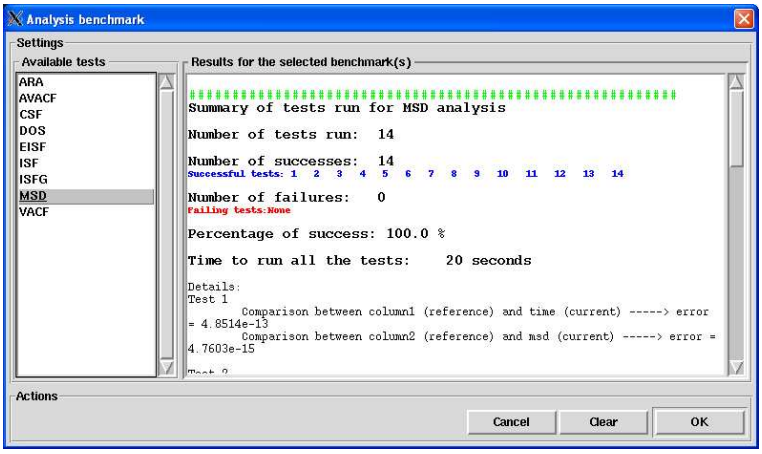

Figure 4.69: The dialog from where the analysis benchmarks can be run.

From this dialog, one can select on the left panel one or many analysis benchmarks. If you want more information about the contents of a given benchmark, right-click on one of the benchmark. This will pop up a small informative dialog. Pressing the **OK** button will run the benchmark. Depending on the selected analysis, you may wait a little bit before the results will be displayed on the right panel with a detailed statistics about the tests that have been run and the results obtained (e.g. number of tests run, percentage of success, time taken for the benchmark ...). For the tests that failed, clicking on the test number will pop up a window where the settings for the current version and the reference version are displayed for comparison.

The currently available benchmarks are:

- **ARA** for *ARA* analysis,
- **AVACF** for *AVACF* analysis,
- **DCSF** for *DCSF* analysis,
- **DOS** for *DOS* analysis,
- **EISF** for *EISF* analysis,
- **DISF** for *DISF*analysis,
- **DISFG** for *DISFG* analysis,
- **MSD** for *MSD* analysis,
- **VACF** for *VACF* analysis,

4.4.5 About nMOLDYN

Pressing the **About nMOLDYN** button will just pop up the dialog showed in figure 4.70 from which general information about *n*MOLDYN are displayed (e.g. authors, *n*MOLDYN and MMTK webpages ...).

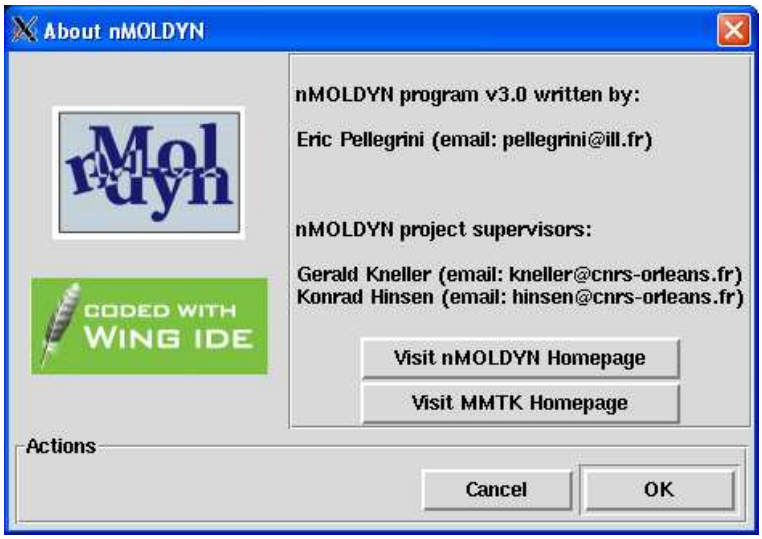

Figure 4.70: The dialog from which general information about *n*MOLDYN are displayed.

5. Using nMOLDYN from the command-line interface

In some situations a graphical interface cannot be used for technical reasons (e.g., text-mode connection to remote machines or Tk not available) or it is not the most convenient solution. This occurs typically when one needs to perform a large number of similar calculations or when the subset, deuteration or group selections that are to be used for a given analysis requires more flexibility than the *n*MOLDYN *GUI* selection dialogs can offer. For these situations, *n*MOLDYN provides a command-line interface that reads all input information from a single input file. There is two formats for *n*MOLDYN input files: *n*MOLDYN autostart files and *n*MOLDYN input files.

5.1 *n*MOLDYN autostart files

The *n*MOLDYN autostart files (extension **.py**) are Python scripts that can be direclty run without an explicit call to *n*MOLDYN. To run such a file, just type:

`./file.py` on unix

or

`file.py` on Windows

where `file.py` is the name of the *n*MOLDYN autostart file.
A *n*MOLDYN autostart file must have the following format:

1. `#!`*`path_to_your_python_executable`*

2. `from nMOLDYN.Analysis.Template import` *`analysis_name`*

3. `parameters = {}`

4. `parameters["version"] = "`*`nmoldyn_version`*`"`
5. `parameters["estimate"] = "no"`

6. `parameters["`*`name1`*`"] =` *`value1`*

⋮

7. `parameters["`*`nameN`*`"] =` *`valueN`*

8. `analysis =` *`analysis_name`*`(parameters)`
9. `analysis.runAnalysis()`

some explanations are necessary about the structure of this file:

- Line 1.: *path_to_your_python_executable* is the path to the Python executable of the Python distribution where *n*MOLDYN is installed. This line is optional on Windows.
- Line 2.: *analysis_name* is the *n*MOLDYN internal name of the analysis to run. A comprehensive list of the *n*MOLDYN analysis internal names associated to each analysis is showed in Tab. 5.1.
- Line 3.: this line initalizes the dictionnary that will stores the analysis input parameters.
- Line 4.: *nmoldyn_version* is the version number of *n*MOLDYN. This line is optional.
- Line 5.: if set to "yes" instead of "no". the analysis will be run in estimate mode (see Section 4.2.3 for details). If "no" the full analysis will be run. If this parameter is omitted, the full analysis will be run.
- Line 6.: *name1* is the name in lower case of the first parameters of the analysis with value equal to *value1*. Idem for Line 7.. The order in which the parameters appear is not relevant. A comprehensive list of the parameters names and expected values associated to each analysis can be found using the right column of Tab. 5.1.
- Line 8.: *analysis_name* is the same that in Line 2..
- Line 9.: this line runs the analysis.

The figure 5.1 shows an example of an autostart file for the *MSD* analysis. It was direclty generated from *n*MOLDYN *GUI*. Saving an autostart file from the *GUI* (see Section 4.2.3) is often the easiest way to get such file. This file providing then a convenient starting point for customization.

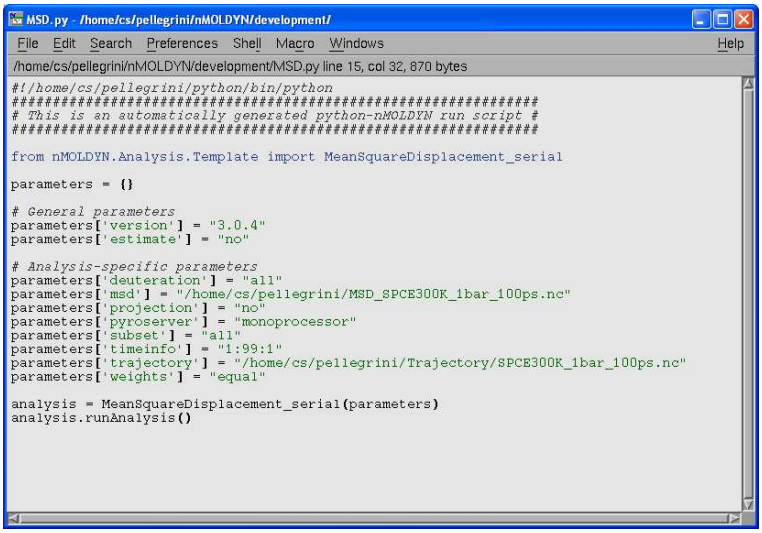

Figure 5.1: Example of a *n*MOLDYN autostart file derived for a *MSD* analysis.

Analysis	nMOLDYN internal name	Section
Mean-Square Displacement	MeanSquareDisplacement_serial	4.2.4.1
Root Mean-Square Deviation	RootMeanSquareDeviation_serial	4.2.4.2
Radius of gyration	RadiusOfGyration_serial	4.2.4.3
Angular Correlation	AngularCorrelation_serial	4.2.4.4
Velocity Autocorrelation Function	CartesianVelocityAutoCorrelationFunction_serial	4.2.4.5
Density Of States	CartesianDensityOfStates_serial	4.2.4.6
Pass-Band Filtered Trajectory	PassBandFilteredTrajectory_serial	4.2.4.7
Global Motion Filtered Trajectory	GlobalMotionFilteredTrajectory_serial	4.2.4.8
Rigid-Body Trajectory	RigidBodyTrajectory_serial	4.2.4.9
Center Of Mass Trajectory	CenterOfMassTrajectory_serial	4.2.4.10
Auto-Regressive Analysis	AutoRegressiveAnalysis_serial	4.2.4.11
Quasi Harmonic Analysis	QuasiHarmonicAnalysis_serial	4.2.4.12
Reorientational Correlation Function	ReorientationalCorrelationFunction_serial	4.2.4.13
Angular Velocity AutoCorrelation Function	AngularVelocityAutoCorrelationFunction_serial	4.2.4.14
Angular Density Of States	AngularDensityOfStates_serial	4.2.4.15
Dynamic Coherent Structure Factor	DynamicCoherentStructureFactor_serial	4.2.5.2
Dynamic Coherent Structure Factor (AR Model)	DynamicCoherentStructureFactorAR_serial	4.2.5.3
Dynamic Incoherent Structure Factor	DynamicIncoherentStructureFactor_serial	4.2.5.4
Dynamic Incoherent Structure Factor (AR Model)	DynamicIncoherentStructureFactorAR_serial	4.2.5.5
Dynamic Incoherent Structure Factor (Gaussian Approximation)	DynamicIncoherentStructureFactorGaussian_serial	4.2.5.6
Elastic Incoherent Structure Factor	ElasticIncoherentStructureFactor_serial	4.2.5.7
Static Coherent Structure Factor	StaticCoherentStructureFactor_serial	4.2.5.8
Smoothed Static Coherent Structure Factor	SmoothedStaticCoherentStructureFactor_serial	4.2.5.9
Pair Distribution Function	PairDistributionFunction_serial	4.2.6.1
Coordination number	CoordinationNumber_serial	4.2.6.2
Spatial Density	SpatialDensity_serial	4.2.6.3
ScrewFit analysis	ScrewFitAnalysis_serial	4.2.6.4
Order Parameter	OrderParameter_serial	4.2.7.1
Order Parameter (Contact Model)	OrderParameterContactModel_serial	4.2.7.2

Table 5.1: List of the nMOLDYN analysis internal names.

5.2 *n*MOLDYN input files

The *n*MOLDYN input files (extension **.nmi**) are ASCII files that must be run through *n*MOLDYN. To run such a file, just type:

`./nMOLDYNStart.py -i` *`file.py`* on unix

or

`nMOLDYNStart.py` *`file.py`* on Windows

where *file.py* is the name of the *n*MOLDYN input file.
An *n*MOLDYN input file must have the following format:

1. `analysis =` *`analysis_name`*

2. `version = "`*`nmoldyn_version`*`"`
3. `estimate = "no"`

4. *`name1 = value1`*

⋮

5. *`nameN = valueN`*

some explanations are necessary about the structure of this file:

- Line 1.: *analysis_name* is the *n*MOLDYN internal name of the analysis to run. A comprehensive list of the *n*MOLDYN analysis internal names associated to each analysis is showed in Tab. 5.1.
- Line 2.: *nmoldyn_version* is the version number of *n*MOLDYN. This line is optional.
- Line 3.: if set to "yes" instead of "no". the analysis will be run in estimate mode (see Section 4.2.3 for details). If "no" the full analysis will be run. If this parameter is omitted, the full analysis will be run.
- Line 4.: *name1* is the name in lower case of the first parameters of the analysis with value equal to *value1*. Idem for Line 5.. The order in which the parameters appear is not relevant. A comprehensive list of the parameters names and expected values associated to each analysis can be found using the right column of Tab. 5.1.

The figure 5.2 shows an example of a *n*MOLDYN input file for the *MSD* analysis. It was direclty generated from *n*MOLDYN *GUI*. Saving an input file from the *GUI* (see Section 4.2.3) is often the easiest way to get such file. This file providing then a convenient starting point for customization.

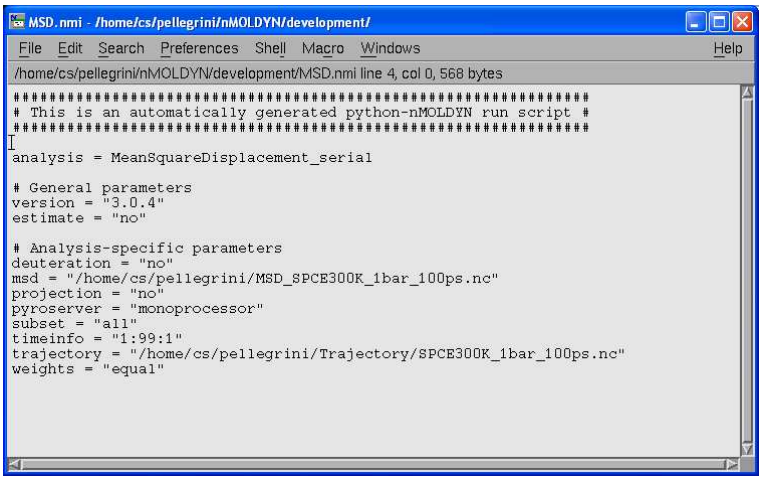

Figure 5.2: Example of a *n*MOLDYN input file derived for a *MSD* analysis.

For those familiar with *n*MOLDYN 2, the *n*MOLDYN input files are the equivalent of the input file through *p*MOLDYN. They have been kept in *n*MOLDYN 3 for historical reasons even if the authors think that the *n*MOLDYN autostart are a little bit more convenient.

Bibliography

[1] Rog, T.; Murzin, K.; Hinsen, K.; Kneller, G.R. *J. Comp. Chem.* **2002**, *24*, 657-667.

[2] Kneller, G.R.; Keiner, V.; Kneller, M.; Schiller, M. *Comp. Phys. Comm.* **1995**, *91*, 191-214.

[3] Hinsen, K. *J.Comp.Chem* **2000**, *21*, 79-85.

[4] http://sourcesup.cru.fr/projects/mmtk/

[5] http://www.unidata.ucar.edu/software/netcdf/

[6] Rew, R.; Davis, G.; Emmerson, S.; Davies, H. NetcdfUser'sGuide for c, an interface for Data Access, version 3. http://www.unidata.ucar.edu/packages/netcdf/guidec, 1997.

[7] Lovesey, S.W. *Theory of Neutron Scattering from Condensed Matter, Vol. 1*; Clarendon Press, Oxford, 1984.

[8] Bee, M. *Quasielastic Neutron Scattering: Principles and Applications in Solid State Chemistry, Biology, and Materials Science*; Adam Hilger, Bristol, 1988.

[9] Kneller, G.R. *Molecular Physics* **1994**, *83*(1), 63-87.

[10] Kneller, G.R.; Geiger, A. *Molecular Physics* **1989**, *68*(2), 487-498.

[11] Kneller, G.R.; Geiger, A. *Molecular Physics* **1990**, *70*(3), 465-483.

[12] Boehm H.J.; Ahlrichs, R. *Mol. Phys.* **1985**, *54*(6), 1261-1274.

[13] Anderson, J.; Ullo, J.J.; Yip, S. *J. Chem. Phys.* **1987**, *86*(7), 4078-4089.

[14] Kneller, G.R; Doster, W.; Settles, M.; Cusack, S.; Smith, J.C. *J. Chem. Phys.*, **1992**, *97*(12), 8864-8879.

[15] Dianoux, A.J.; Kneller, G.R.; Sauvageol, J.L.; Smith, J.C. *J. Chem. Phys.*, **1993**, *99*(7), 5586-5596.

[16] van Rossum, G. The Python web site, http://www.python.org/.

[17] http://www.python.org/download/

[18] http://numpy.scipy.org/

[19] http://pyro.sourceforge.net/

[20] Hinsen, K. ScientificPython. http://dirac.cnrs-orleans.fr

[21] http://sourcesup.cru.fr/projects/scientific-py/

[22] OpenSource web site. http://www.opensource.org, 1997.

[23] http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/

[24] http://wiki.python.org/moin/TkInter

[25] http://matplotlib.sourceforge.net/

[26] http://www.ambermd.org

[27] http://www.charmm.org/

[28] http://www.cse.scitech.ac.uk/ccg/software/DL_POLY

[29] http://www.accelrys.com/products/materials-studio

[30] http://www.accelrys.com/products/materials-studio/modules/discover.html

[31] http://www.accelrys.com/products/materials-studio/modules/forcite.html

[32] http://www.ks.uiuc.edu/Research/namd

[33] http://www.ks.uiuc.edu/Research/vmd

[34] http://cms.mpi.univie.ac.at/vasp

[35] http://cns-online.org/v1.21

[36] http://www.unidata.ucar.edu/software/netcdf/ncdump-man-1.html

[37] http://www.unidata.ucar.edu/software/netcdf/ncgen-man-1.html

[38] Kneller, G.R.; Calligari, P. *Acta Crystallographica* **2006**, *D62* 302-311.

[39] http://sourcesup.cru.fr/projects/nmoldyn/

[40] http://dirac.cnrs-orleans.fr/plone/software/nmoldyn

[41] http://www.cecill.info/

[42] http://sourceforge.net/projects/pywin32/

[43] http://www.ucar.edu/

[44] http://my.unidata.ucar.edu/content/software/netcdf/software.html

[45] http://www.rcsb.org/pdb/

[46] http://docs.python.org/library/configparser.html

[47] http://dirac.cnrs-orleans.fr/Manuals/MMTK/MMTK_37.html

[48] http://www.compsoc.man.ac.uk/ lucky/Democritus/

[49] Rahman, A.; Singwi, K.S.; Sjölander, A. *Phys. Rev.* **1962**, *126*, 986-996.

[50] Boon, J.P.; Yip, S. *Molecular Hydrodynamics*; McGraw-Hill, New York, 1980.

[51] Kneller, G.R. Technical Report Jül 2215, Forschungszentrum Jülich (ISSN 0366-0885), ZB Forschungszentrum Jülich, D-52425 Jülich, Germany.

[52] Abramowitz, M.; Stegun, I.A. *Handbook of Mathematical Functions*; Dover, New York, 1972.

[53] Brooks, B.R.; Janezic, D.; Karplus, M. *J. Comp. Chem.* **1995**, *16*(12), 1522-1542.

[54] Ryckaert, J.P.; Ciccotti, G.; Berendsen, H.J.C *J. Comput. Phys.* **1977**, *23*, 327-341.

[55] Kneller, G.R. *Molecular Simulation* **1991**, *7*, 113-119.

[56] Altmann, S.L. *Rotations, Quaternions, and Double Groups*; Clarendon Press, Oxford, 1986.

[57] Kneller, G.R.; Hinsen, K. *J. Chem. Phys.* **2001**, *115*, 11097-11105.

[58] Papoulis, A. *Probability, random variables, and Stochastic Processe;3rd ed*; McGraw-Hill, New York, 1991.

[59] Makhoul, J. *J. Proc. IEEE* **1975**, *63* 561-580.

[60] Burg, J. *Maximum Entropy Spectral Analysis*; PhD thesis, Stanford University, Stanford, CA, 1975.

[61] Makhoul, J. *J. IEEE Transactions on Acoustics, Speech, and Signal Processing* **1977**, *25*, 423-428.

[62] Lynden-Bell, R.M.; Stone, A.J. *Molecular Simulation* **1989**, *3*, 271-281.

[63] Edmonds, A.R. *Angular momentum in Quantum Mechanics*; Princeton University Press, Princeton, New Jersey, 1957.

[64] Brink, D.M.; Satchler, G.R. *Angular Momentum*; Clarendon Press, Oxford, 1968.

[65] Rose, M.E. *Elementary Theory of Angular Momentum*; John Wiley, New York, 1957.

[66] Allen, M.P.; Tildesley, D.J. *Computer Simulation of Liquids*; Oxford University Press, Oxford, 1987

[67] van Hove, L. *Phys. Rev.* **1954**, *95*(1), 249-262.

[68] Schofield, P. *Phys. Rev. Letters* **1960**, *4*(5), 239-240.

[69] Fischer, H.E.; Barnes, A.C.; Salmon, P.S. *Rep. Prog. Phys.* **2006**, *69*(1), 233-299.

[70] Refield, A. *IBM J. Res. & Dev.* **1957**, *1*, 19-31.

[71] Lipari, G.; Szabo, A. *J. Am. Chem. Soc.* **1982**, *104*, 4546-4559.

[72] Lipari, G.; Szabo, A. *J. Am. Chem. Soc.* **1982**, *104*, 4559-4570.

[73] F. Zhang and R. Bruschweiler. *J. Am. Chem. Soc.* **2002**, *124*, 12654-12655.

[74] Brigham, E.O. *The Fast Fourier Transfrom*; Prentice Hall, Englewood Cliffs (NJ) USA, 1974.

[75] Papoulis, A. *Signal Analysis*, McGraw-Hill, Singapore, 1984.

[76] Harris, F.J. *Proc. IEEE* **1978**, *66*(1), 51-83.

[77] http://www.unidata.ucar.edu/software/netcdf/guide_12.html

[78] http://www.tcl.tk/software/tcltk/

Appendices

A. The FCA algorithm

Most of the quantities which can be extracted from MD simulations are time correlation functions. Correlation functions of discrete time series can be efficiently calculated by using the Fast Fourier Transform (FFT) [74]. The FCA allows the number of multiplications (complexity) to be reduced from $\propto N_t^2$ to $\propto N_t \log_2(N_t)$. In *n*MOLDYN all time correlation functions are computed using the FCA method which will be outlined in the following. We will also briefly comment on spectral smoothing of Fourier transformed correlation functions.

We consider two time series

$$a(k \cdot \Delta t), \qquad b(k \cdot \Delta t), \qquad k = 0 \ldots N_t - 1, \tag{A.1}$$

of length $T = (N_t - 1) \cdot \Delta t$ which are to be correlated. In the following the shorthands $a(k)$ and $b(k)$ will be used. The discrete correlation function of $a(k)$ and $b(k)$ is defined as

$$c_{ab}(m) \doteq \begin{cases} \frac{1}{N_t - m} \sum_{k=0}^{N_t - m - 1} a^*(k) b(k+m), & m = 0 \ldots N_t - 1, \\ \\ \frac{1}{N_t - |m|} \sum_{k=|m|}^{N_t - 1} a^*(k) b(k - |m|), & m = -(N_t - 1) \ldots -1. \end{cases} \tag{A.2}$$

The prefactors in front of the sums ensure the proper normalization of the individual channels, $m = -(N_t - 1) \ldots N_t - 1$. The asterisk denotes a complex conjugate. According to (A.2), $c_{ab}(m)$ has $2N_t - 1$ data points and obeys the symmetry relation

$$c_{ab}(m) = c_{ba}^*(-m). \tag{A.3}$$

In case that $a(k)$ and $b(k)$ are identical, the corresponding correlation function $c_{aa}(m)$ is called an *auto*correlation function. We define now the extended, periodic time series

$$A(k) = \begin{cases} a(k) & k = 0 \ldots N_t - 1 \\ 0 & k = N_t \ldots 2N_t - 1 \end{cases}, \tag{A.4}$$

$$B(k) = \begin{cases} b(k) & k = 0 \ldots N_t - 1 \\ 0 & k = N_t \ldots 2N_t - 1 \end{cases}, \tag{A.5}$$

which have the period $2N_t$,

$$A(k) = A(k + m \cdot 2N_t), \qquad B(k) = B(k + m \cdot 2N_t), \qquad m = 0, \pm 1, \pm 2, \ldots . \tag{A.6}$$

The discrete, cyclic correlation of $A(k)$ and $B(k)$ is defined as

$$S_{AB}(m) = \sum_{k=0}^{2N_t - 1} A^*(k) B(k+m). \tag{A.7}$$

It is easy to see that

$$c_{ab}(m) = \frac{1}{N_t - |m|} S_{AB}(m), \qquad -(N_t - 1) \leq m \leq N_t - 1. \tag{A.8}$$

Using the correlation theorem of discrete periodic functions [74], $S_{AB}(m)$ can be written as

$$S_{AB}(m) = \frac{1}{2N_t} \sum_{n=0}^{2N_t - 1} \exp\left[2\pi i \left(\frac{mn}{2N_t}\right)\right] \tilde{A}^*\left(\frac{n}{2N_t}\right) \tilde{B}\left(\frac{n}{2N_t}\right) \tag{A.9}$$

where $\tilde{A}\left(\frac{n}{2N_t}\right)$ and $\tilde{B}\left(\frac{n}{2N_t}\right)$ are the discrete Fourier transforms of $A(k)$ and $B(k)$, respectively:

$$\tilde{A}\left(\frac{n}{2N_t}\right) = \sum_{k=0}^{2N_t-1} \exp\left[-2\pi i\left(\frac{nk}{2N_t}\right)\right] A(k), \tag{A.10}$$

$$\tilde{B}\left(\frac{n}{2N_t}\right) = \sum_{k=0}^{2N_t-1} \exp\left[-2\pi i\left(\frac{nk}{2N_t}\right)\right] B(k). \tag{A.11}$$

If the Fourier transforms of the signals $A(k)$ and $B(k)$ as well as the inverse transform in (A.9) are computed by FFT, $S_{AB}(m)$ can be computed by $\propto N_t \log_2(N_t)$ instead of $\propto N_t^2$ multiplications. It is sometimes said that the FFT method induces spurious correlations. We emphasize that this is only the case if the time series $a(k)$ and $b(k)$ are not properly extended, as indicated in Eqs. (A.4) and (A.5). The FFT method and the direct scheme (A.2) give, apart from round-off errors, *identical results*.

In many cases not only the computation of a correlation function is required, but also the computation of its Fourier spectrum. In principle one could use the product $\tilde{A}^*\left(\frac{n}{2N_t}\right)\tilde{B}\left(\frac{n}{2N_t}\right)$ which is already available as an intermediate step in the computation of $S_{AB}(m)$ according to (A.9). This would, however, not be a good estimate for the spectrum of $c_{ab}(m)$ [75]. In nMOLDYN all spectra are smoothed by applying a window in the time domain [75]:

$$P_{ab}\left(\frac{n}{2N_t}\right) = \Delta t \cdot \sum_{m=-(N_t-1)}^{N_t-1} \exp\left[-2\pi i\left(\frac{nm}{2N_t}\right)\right] W(m)\,\frac{1}{N-|m|} S_{AB}(m). \tag{A.12}$$

The time step Δt in front of the sum yields the proper normalization of the spectrum. In nMOLDYN a Gaussian window [76] is used:

$$W(m) = \exp\left[-\frac{1}{2}\left(\alpha\frac{|m|}{N_t-1}\right)^2\right], \qquad m = -(N_t-1)\ldots N_t-1. \tag{A.13}$$

Its widths in the time and frequency domain are $\sigma_t = \alpha/T$ and $\sigma_\nu = 1/(2\pi\sigma_t)$, respectively. We recall that $T = (N_t-1)\cdot\Delta t$ is the length of the simulation. σ_ν corresponds to the width of the resolution function of the Fourier spectrum.