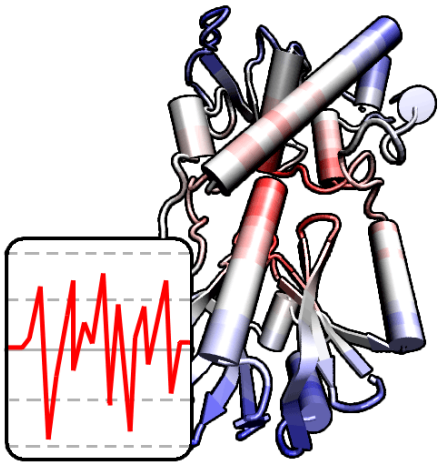
Molecular Dynamics Trajectory Reader & Analyzer User's Guide

Version 1.2

June 1, 2015



Description

The Molecular Dynamics Trajectory Reader & Analyzer (MDTRA) User's Guide describes how to run and use the PDB trajectory analyzing program MDTRA. This guide documents the user interfaces configuring parameters to analyze and displaying data computed, and describes how to perform some trajectory-related search using additional analysis tools.

Contents

List of Figures

List of Tables

1. Introduction

MDTRA is a graphical user interface (GUI) program designed for analysis of molecular dynamics trajectories in PDB format. Based on input data format widely used to describe three-dimensional structure of proteins and nucleic acids, the program implicates analysis of any chemical compounds until the PDB file is correct and contains all the necessary data. MDTRA runs on Microsoft Windows and on UNIX workstations with X-Windows graphics system (for example, Ubuntu). Online information about MDTRA is available from:

http://bison.niboch.nsc.ru/mdtra.html

1.1. List of key MDTRA features

- **General PDB trajectory analysis**

MDTRA manages trajectories as file streams, involving into calculations either full trajectory or its part. Data sources of different types (RMSD, RMSF, radius of gyration, distance, angle, torsion and dihedral angles, solvent-accessible surface area and occlusion, etc.) can be registered and further grouped into result collectors. Result collectors define data to be displayed on the graphical plot, with optional scale, bias and conversion of measure. Graphical plot data can be exported to an image file, and numeric data can be exported to a text file.

- **Statistical analysis**

MDTRA performs calculation of common statistical parameters along the trajectory (mean values, range, variance, etc.) and estimates Pearson linear correlation between data sources within a single result collector. These data can also be exported to a text file.

- **Friendly selection syntax**

Some data sources and tools require to specify a part of molecule to analyze expressed in distinct atom set. MDTRA uses RasMol-style selection syntax to define atoms, residues and chains, specific parts (protein, water, backbone, side chain, etc.) or those within another molecule part. The program also provides a tool to quickly estimate the result of certain selection term.

- **Visualizing a Trajectory**

MDTRA is designed to use an external PDB viewer (RasWin or VMD) to visually analyze a snapshot of a trajectory and compare different trajectories within a single result collector. Internal PDB viewer is under development.

- **Trajectory Related Search**

There are some tools designed for simplification of the valuable data search. MDTRA performs distance search along two trajectories using custom significance criterion to ascertain atoms with different positions, which can be useful to analyze trajectories of the same compound with minor structure changes. The trajectory can also be searched for hydrogen bonds revealing those participating in structure stabilization for a meaningful time scale. A two-dimensional root-mean-square deviation map (2D-RMSD) can be built, rendered and saved to an image file for a trajectory or its part and for selected region of a compound to detect relative mobility of molecule's parts. For calculated trajectory data, histograms can be build, rendered and saved to either image files, or numeric data tables.

- **User-defined Programs for Data Analysis**

MDTRA allows specifying user-defined programs to extend data analysis capabilities. There is a Lua programming language interpreter with a set of program built-ins that allows calculating custom trajectory parameters and statistical data. This feature requires user to have some programming skills.

1.2. Contacting the authors

The current developer of MDTRA is Alexander V. Popov. Dr. Dmitry O. Zharkov and Dr. Yury N. Vorobjev are acknowledged for valuable remarks and advices on MDTRA functionality, usability and feature set. We are very interested in and grateful for any user comments and reports of program faults and inaccuracies. If you have any suggestions, bug reports, or general comments about MDTRA, please send them to us at apopov@niboch.nsc.ru.

1.3. Citation reference

The authors request that any published work or images created using MDTRA, and/or any software developed using MDTRA source code, include the following reference:

Popov, A.V., Vorobjev, Y.N., and Zharkov, D.O. MDTRA: a molecular dynamics trajectory analyzer with a graphical user interface. // *J. Comput. Chem.* 2013 Feb 5. **34** (5):319-25. doi: 10.1002/jcc.23135.

1.4. Copyright and Disclaimer Notices

code is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

If for any reason you do not have the GNU General Public License file in your distribution, it can be downloaded from:

http://bison.niboch.nsc.ru/license/mdtra_license.txt

Some of the code and executables used by MDTRA have their own usage restrictions:

- **Qt**

Qt 4.8.6 dynamic-link libraries are used by MDTRA subject to the terms and conditions in LGPL version 2.1 licenses as described at Qt official website:

http://qt.nokia.com/products/licensing

- **Lua**

A Lua 5.1.2 dynamic-link library, with minor modifications, is used by and shipped with MDTRA subject to the terms and conditions in MIT license that is compatible with GPL, as described at Lua official website:

http://www.lua.org/license.html

- **CUDA Runtime**

32-bit CUDA Runtime Library v. 4.1.28 (cudart32_41_28.dll) is redistributed with MDTRA subject to the terms and conditions in EULA for NVIDIA CUDA Toolkit.

1.5. Integration with BioPASED Molecular Dynamics Modelling Program

MDTRA is fully compatible with trajectory files written by "BioPASED" molecular dynamics modelling program. BioPASED is Copyright © 2011 Yuri N. Vorobjev, Institute of Chemical Biology and Fundamental Medicine of Siberian Branch of Russian Academy of Sciences. BioPASED and GUI-BioPASED (a graphical superset of BioPASED designed to facilitate molecular modelling task creation) are available online:

http://biopased.niboch.nsc.ru/

2. Hardware and software requirements

2.1. Basic hardware and software requirements

MDTRA operates in graphics mode that requires an OpenGL-capable graphics accelerator with up-to-date drivers. However since the program uses mostly a basic OpenGL 1.1 feature set that is usually available even without graphics accelerator through software emulation, it should work even with integrated video solutions. But in order to use some features like plot multi-sample anti-aliasing a video accelerator is required (particularly, for multisampling support an OpenGL 1.3-compliant video card is required).

MDTRA is designed to run at a minimum screen resolution of 1024 x 768 pixels; for some tools, for example, 2D-RMSD Map, a vertical screen resolution of 960 pixels or larger is required (otherwise the tool window won't fit the screen).

The amount of system memory does not impact general performance of computational tasks, but some tools that load all PDB of the trajectory files into memory (for example, 2D-RMSD tool) may require a large amount of RAM.

The speed of hard disk drive is often a bottleneck of calculations; therefore a fast drive (for example, SSD drive) is recommended but not required.

MDTRA Win32 sources were compiled to run on Microsoft Windows® XP, Microsoft Windows® Vista, Microsoft Windows® 7. MDTRA UNIX sources were compiled to run on Linux with X-Server (for example, Debian and Ubuntu).

2.2. Multi-core, SIMD-featured CPUs

MDTRA makes use of modern processors' features: multi-core, or SIMD (single instruction multiple data), or both in computationally demanding tasks. Parallelization is used to compute result collectors' data (the primary purpose of MDTRA), to perform trajectory search (Distance Search and H-Bonds Search Tools) and to pre-process PDB files in 2D-RMSD module. Some procedures (for example, structure alignment with Kabsch algorithm, root-mean-square calculation) are SIMD-optimized for best performance. Usage of both features can be disabled in MDTRA settings.

2.3. GPGPU requirements

General-purpose computations on GPU in MDTRA are implemented using NVIDIA CUDA 4.1 technology and require a NVIDIA GPU, GeForce8xxx-class or better, with modern display drivers. Drivers can be downloaded from the official NVIDIA website:

http://www.nvidia.com/Download/index.aspx?lang=en-us

3. Trajectory File Formats

MDTRA reads trajectory files in PDB (Protein Data Bank) format. To use MDTRA with molecular dynamics programs that do not output PDB trajectories, one must convert output trajectories to PDB format using external tools. MDTRA reads the following information from PDB file: serial numbers of atoms; atom titles; serial numbers of residues; residue titles; chain identifiers; Cartesian coordinates of atoms; and, optionally, effective force vectors of atoms. There are two default PDB formats: generic format and BioPASED format, as described below. One may also define its own PDB format to maintain compatibility with programs that to not comply with PDB format standard.

3.1. Generic PDB File Format

Generic PDB file format is based on specification published by the wwPDB ("Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description", version 3.30) and complied with PDB Exchange Dictionary (PDBx). MDTRA exploits such fields, as: serial numbers of atoms; atom titles; serial numbers of residues; residue titles; chain identifiers; Cartesian coordinates of atoms. See the Table 3.1 for exact markup reference (please note that columns are indexed from 0).

Field Name	Field Type	Start Column Number	Number of Columns
Serial number	Integer	6	5
Atom title	String	12	4
Residue title	String	17	3
Chain identifier	Character	21	1
Residue number	Integer	22	4
Coordinate X	Float	30	8
Coordinate Y	Float	38	8
Coordinate Z	Float	46	8

Table 3.1. Generic PDB File Format Markup Reference.

All user-defined formats are initially based on Generic PDB file format. This is the default format to be used in Streams.

3.2. BioPASED PDB File Format

BioPASED molecular dynamics program outputs trajectories in PDB format a bit different from Generic format. MDTRA exploits such fields, as: serial numbers of atoms; atom titles; serial numbers of residues; residue titles; chain identifiers; Cartesian coordinates of atoms; effective

forces of atoms. See the Table 3.1 for exact markup reference (please note that columns are indexed from 0).

Field Name	Field Type	Start Column Number	Number of Columns
Serial number	Integer	6	5
Atom title	String	12	5
Residue title	String	17	4
Chain identifier	Character	21	1
Residue number	Integer	22	4
Coordinate X	Float	30	8
Coordinate Y	Float	38	8
Coordinate Z	Float	46	8
Force X	Float	55	8
Force Y	Float	63	8
Force Z	Float	71	8

Table 3.2. BioPASED PDB File Format Markup Reference.

To make this format default in all subsequently created Streams:

1. In the **Edit** menu, select **Preferences**. The **Preferences** window appears.
2. Switch to the **Formats** tab.
3. On the **Formats** tab, select **BioPASED Format** entry in the formats list.
4. Click the **Set Default** button. Entry title changes into **BioPASED Format [DEFAULT]**.
5. Click **OK** button to save the changes and to close the **Preferences** window.

3.3. Defining Custom PDB File Format

To maintain compatibility with programs that to not comply with PDB format standard one may define a custom PDB format markup. Custom format gets its own format identifier which is kept unique through all the MDTRA sessions. This means that if one deletes a custom format and creates a new one, the format identifier will not be the same. Custom formats are not saved within the Project Files, therefore streams that use custom formats can only be analyzed with MDTRA installation that contains a custom format definition. If a custom format is required to be used with another MDTRA installation, one must copy **conf/formats.dat** file to that installation, thus overwriting custom formats within that installation, if any.

To define a custom format:

1. In the **Edit** menu, select **Preferences**. The **Preferences** window appears.
2. Switch to the **Formats** tab.

3. On the **Formats** tab, click the **Add…** button. The **Add Format** dialog appears (see Figure 3.1).
4. In the **Format Title** text input field, enter new format title. Or, you may use the default title provided by MDTRA.
5. Check the fields that comprise your custom format. For fields checked, select **Field Type**, **Start** (first column in PDB ATOM line; indexed from 0), **Size** (number of columns to parse).
6. Click the **OK** button. The **Add Format** dialog disappears.
7. In the **Preferences** window, click **OK** button to save the changes and to close the **Preferences** window.

A custom format is added to the formats list. Click the **Edit** button to change a markup, or click the **Delete** button to completely remove a custom format. To use this format in all subsequent Streams by default, click the **Set Default** button.

Please note that only custom formats can be edited or deleted.

Custom formats are saved to the **conf/formats.dat** binary file when MDTRA program is closed.

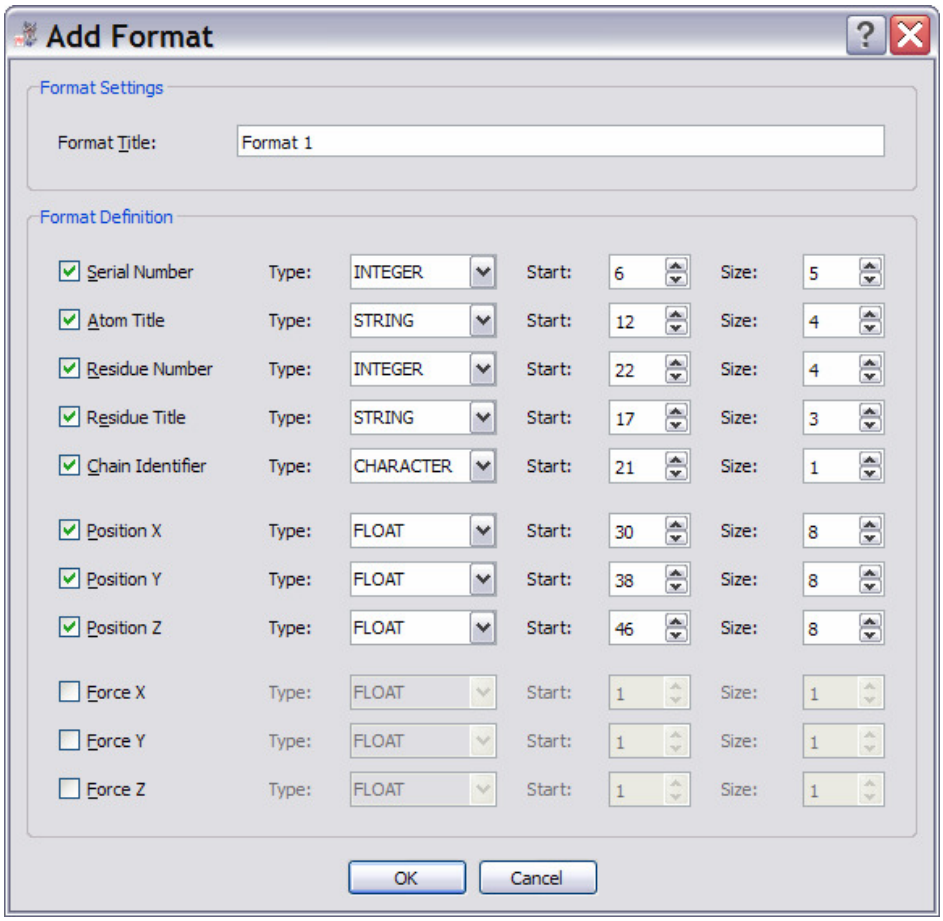


Figure 3.1. The **Add Format** dialog**.**

4. General Trajectory Analysis

General analysis of trajectories is performed in several steps. First, trajectories are registered in MDTRA as Streams. Second, corresponding parameters to compute are registered as Data Sources. Third, representations of these parameters are set up as Result Collectors. And finally a computation process is initiated and upon completion one can view raw data, statistical parameters and graphical representation on the plot, launch external viewer on an element of a trajectory, export data to files, etc. An illustrative guide along the steps of this process is provided herein.

4.1. Streams

The heart of MDTRA project is a trajectory – a set of PDB files representing a single molecular dynamics optimization experiment. Before running any analysis procedures one must register at least one trajectory as a Stream. Some tools require one or two valid Streams to be launched. Since trajectories are usually outsize, this is a crucial point for further work and it must be done carefully.

The New Stream window is shown on the Figure 4.1. A Stream consists of four major parts: 1) Stream title, used to identify the trajectory within some tables; 2) Stream data, an alphabetically sorted list of PDB files representing a trajectory; 3) Stream data format, which trajectory data is rendered into by molecular dynamics program; 4) trajectory time step, in picoseconds, or how much time does a single trajectory step (snapshot) take, so it can be rescaled to time on the Plot.

To register a new Stream:

1. In the **Edit** menu, select **Add Stream…**. The **New Stream** window appears. Or, click the **Add…** button in the **Streams** section of the main window.
2. Enter new Stream title in the text input field of the same name. You may leave the default title provided by MDTRA, but it is not recommended since Stream titles should be legible.
3. Select a format of PDB files of the Stream in the **PDB File Format** combo box.
4. Set a proper value of a time step of the trajectory, in picoseconds, in the **Trajectory Time Step, ps** spin box.
5. If your trajectory is of the moderate size (less than 1000 files), you may add them manually using the common **Open Files** dialog. Click the **Add…** button. The dialog appears. Select single file or multiple files of the trajectory and click the **Open** button to add them to the **Stream Data** list.

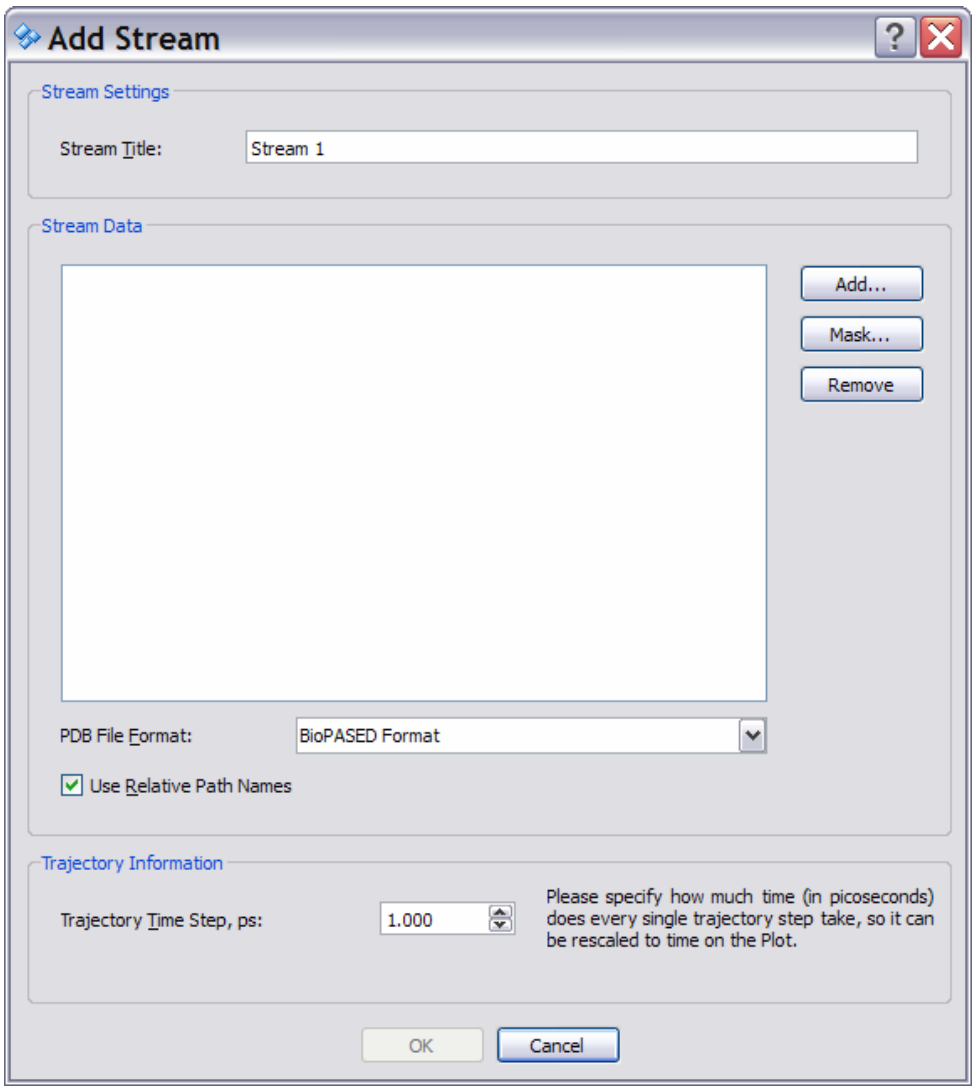


Figure 4.1. The **Add Stream** window layout.

6. Otherwise, if the trajectory is of the large size (more than 1000 files), the **Open Files** dialog may not process the files properly. You may either add files in parts, or use *File Masks*. Click the **Mask…** button. The **Add Files by Mask** window appears (see Figure 4.2). Click the **…** button (the button with three dots) and select any file that belongs to the trajectory. Since file names are usually composed of two parts – name text and serial number, MDTRA can load a whole range of trajectory files, replacing a special control character sequence with an increasing integer value. It tries to detect a serial number in a file name automatically, but one can edit it manually. For example, if you select a file named "TrajectoryStep0004.pdb", the mask will be "TrajectoryStep%04i.pdb", and that control character sequence, "%04i" (04 means that there are 4 characters and leading zeros are not stripped, and i means that this is an integer value) will be replaced with 0001, 0002, 0003 and so on, until a file with the new name is missing. A range for mask value may also be specified, using the **Use**

Mask Value Range frame with the checkbox and the spin boxes within. To close the dialog and to start scanning the directory with mask specified, click the **OK** button.

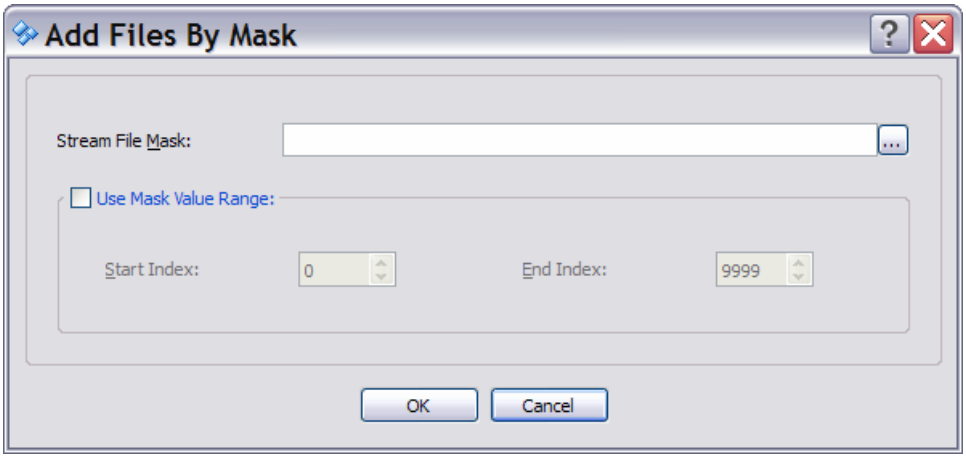


Figure 4.2. The **Add files by Mask** dialog.

7. The **Use Relative Path Names** check box instructs MDTRA to save file names to project file relative to its location. This is useful if you want to create a portable project. However, if a project file and a PDB file are on different disk drives (or mounting points), their names will be saved as absolute.
8. Click the **OK** button to register the Stream.

To edit the existing Stream, select the Stream in the list and click the **Edit…** button below, or double-click the Stream entry in the list. To remove the existing Stream, select the Stream in the list and click the **Remove** button. Please note that all the Data Sources and Result Collectors associated with the Stream are also removed and this action cannot be undone.

4.2. Data Sources

After some Streams are registered, Data Sources can be added. Data Source is a certain variable calculated along the Stream and consists from the following parts: 1) Data Source title, used to identify the Data Source within a Result Collector and the Plot; 2) Stream reference, or which Stream does this Data Source belong to; 3) data type; 4) data arguments, depending on data type. Refer to Table 4.1 for the list of available data types to calculate. The New Data Source window is shown on the Figure 4.3.

To register a new Data Source:

1. In the **Edit** menu, select **Add Data Source…**. The **New Data Source** window appears. Or, click the **Add…** button in the **Data Sources** section of the main window.

2. Enter new Data Source title in the text input field of the same name. You may leave the default title provided by MDTRA, but it is not recommended since Data Source titles should be legible.
3. In the **Stream Source** drop-down combo, select a Stream to use.
4. In the **Data Type** drop-down combo, select a data type to calculate. Depending on that type, some arguments become enabled. Define necessary atoms by selecting **Residue** and its **Atom** in the proper combos. For those data types requiring a selection term, enter it into the **Selection Expression** text input line. To update a selection, press the **Enter** key within that line, or click a button with the funnel picture.
5. Some data types require additional parameters to set, e.g. for "Occluded Area", one must define a selection term for occluder (refer to 5 for more details on how surface occlusion is calculated). Click the **Advanced** button to open a window where such parameters can be set.
6. Click the **OK** button to register the Data Source.

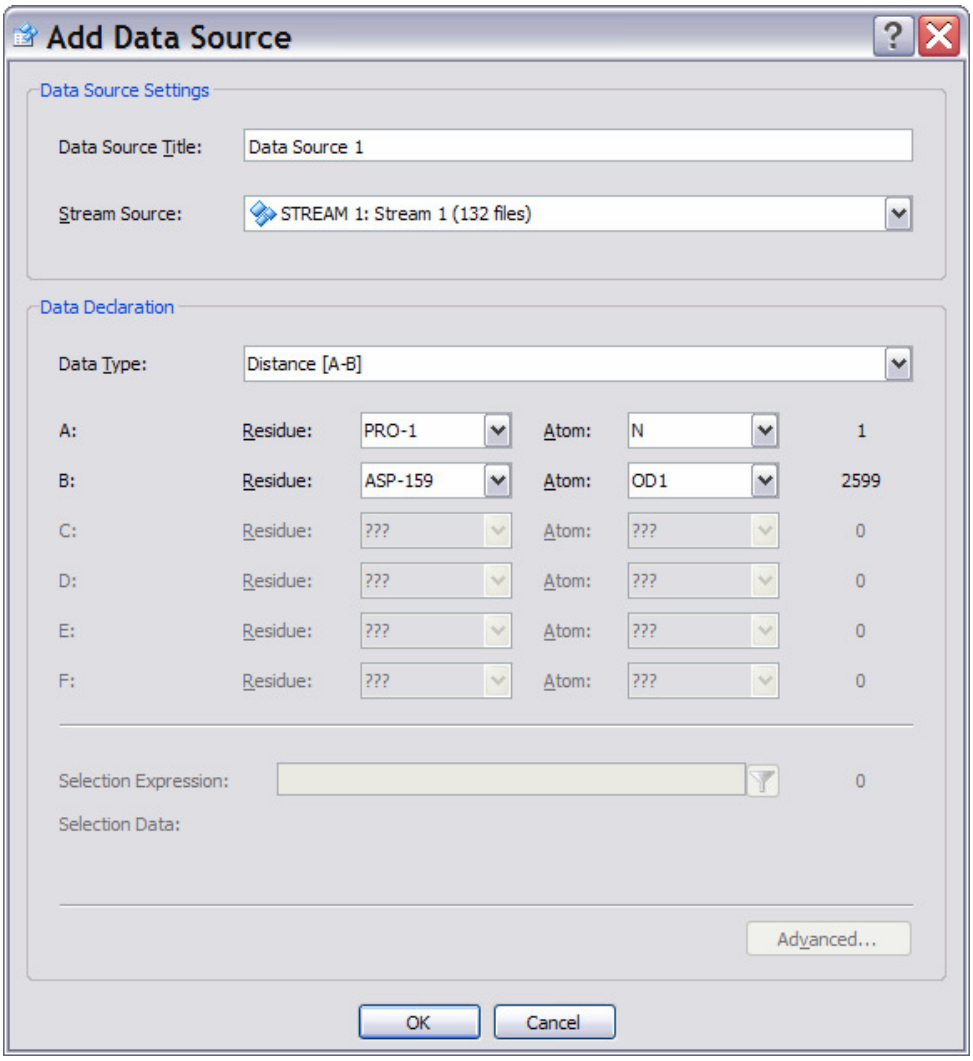


Figure 4.3. The **Add Data Source** window layout.

Data Type	**Number of Arguments**	**Remarks**
Root-mean-square deviation of backbone (RMSD)	0	Backbone is defined as N-Cα-C-O for proteins, O1P-O2P-P-C5'-C4'-O4'-C3'-O3'-C2'-O2'-C1' for nucleic acids; may be either time-based or residue-based
RMSD of Selection	0	Must also specify a selection term; may be either time-based or residue-based
Root-mean-square fluctuation of backbone (RMSF)	0	Backbone is defined as N-Cα-C-O for proteins, O1P-O2P-P-C5'-C4'-O4'-C3'-O3'-C2'-O2'-C1' for nucleic acids; residue-based
RMSF of Selection	0	Must also specify a selection term; residue-based
Radius of Gyration	0	Radius of gyration of the whole molecule (measure of compactness)
Distance	2	Euclidian distance A-B
Angle	3	Angle A-B-C
Angle between Sections	4	Angle between sections A-B and C-D
Torsion Angle ($-\pi \leq$ angle $\leq \pi$)	4	Torsion angle A-B-C-D
Torsion Angle ($0 \leq$ angle $\leq 2\pi$)	4	Torsion angle A-B-C-D, unsigned
Dihedral Angle ($-\pi \leq$ angle $\leq \pi$)	4	Dihedral angle A-B-C-D; depending on B-C direction (angle between planes A-B-C and B-C-D)
Dihedral Angle ($0 \leq$ angle $\leq \pi$)	4	Dihedral angle A-B-C-D; invariant
Angle between Planes	6	Angle between planes defined by atoms A-B-C and D-E-F
Force Magnitude	1	BioPASED or custom format only; force at atom A
Resultant Force	2	BioPASED or custom format only; forces are projected onto A-B direction and summed up; positive value refers to application of the resultant force towards the first atom
Solvent-accessible surface area (SAS)	0	May be either time-based or residue-based
SAS of Selection	0	Must also specify a selection term; may be either time-based or residue-based
Occluded Area	0	May be either time-based or residue-based; must also specify a selection term for occluder ("dna" by default)
Occluded Area of Selection	0	Must also specify a selection term; must also specify a selection term for occluder ("dna" by default); may be either time-based or residue-based
User-defined Type	any	User-defined Lua program will be invoked for such

	results. Refer to section 7 for the details.

Table 4.1. List of valid data types in Data Source.

It is sometimes bothersome to add a lot of Data Sources for almost identical Streams. There is a way to add multiple Data Sources with the same arguments for a number of registered Streams. The window looks nearly the same; the only change is that one can now select several streams in the **Stream Sources** list. Please note that atoms are referenced by residue and atom titles, thus if this atom is missing from the Stream selected, the result may be undefined.

To add multiple Data Sources: in the **Edit** menu, select **Add Data Source…**. The **Add Multiple Data Sources** window appears (Figure 4.4). Or, click the **Multiple…** button in the **Data Sources** section of the main window. The other steps are similar to those for a single Data Source described above.

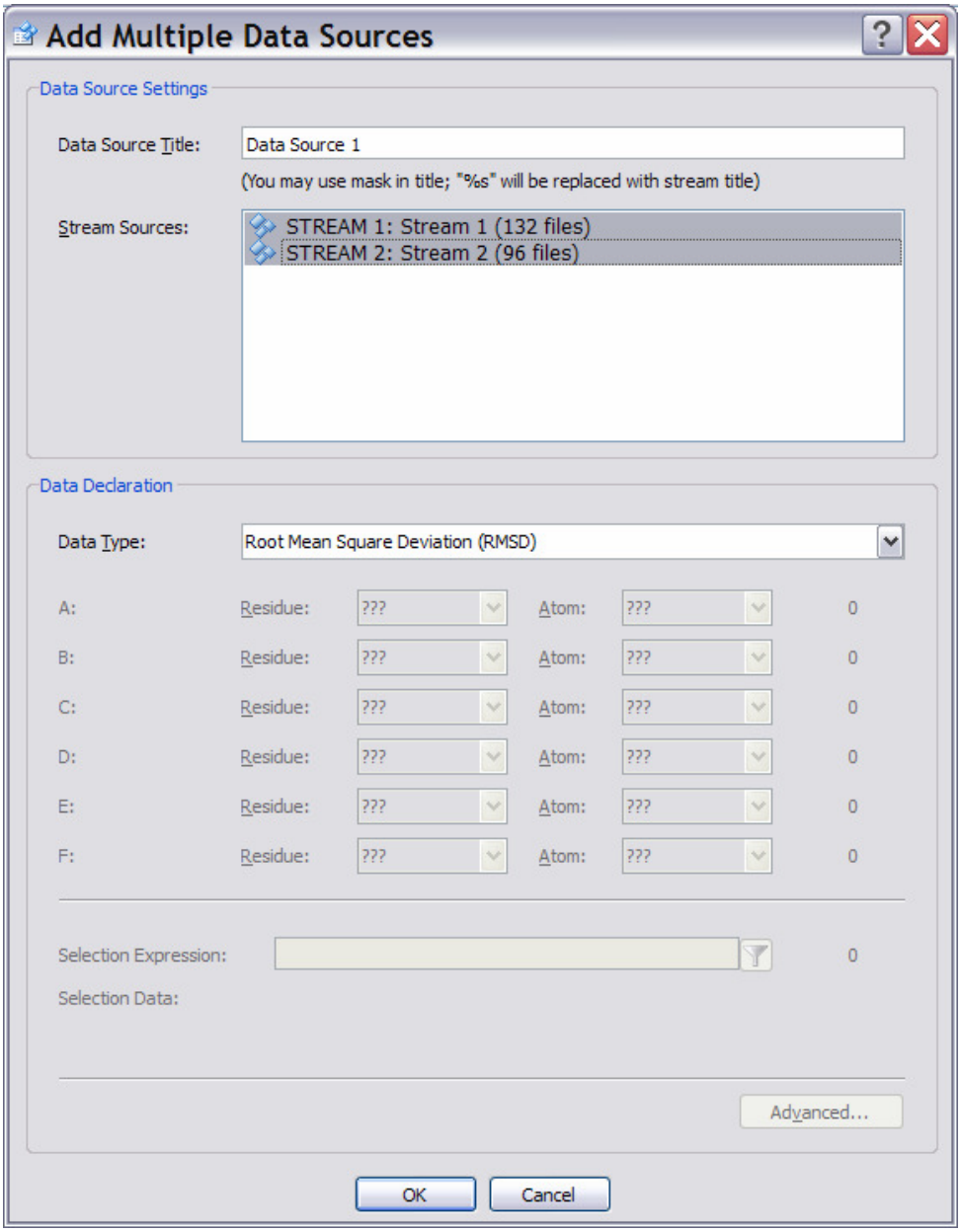


Figure 4.4. The **Add Multiple Data Sources** window layout.

To edit the existing Data Source, select the Data Source in the list and click the **Edit…** button below, or double-click the Data Source entry in the list. To remove the existing Data Source, select it in the list and click the **Remove** button. Please note that all the Result Data Sources referencing the Data Source are also removed (and if it is the only Result Data Source in the Result Collector, the latter is also removed) and this action cannot be undone.

4.3. Result Collectors

To view data calculated and plotted, Data Sources must be grouped, or "collected", into Result Collectors. A Result Collector represents a group of similar parameters, of the same type, presented on the common Plot. One or more Data Sources comprise a single Result Collector, and

their visibility may be toggled separately. Furthermore, Pearson linear correlation is calculated for each Data Source pair in a Result Collector.

All necessary Data Sources must be registered prior to adding Result Collectors that are supposed to use them.

To register a new Result Collector:

1. In the **Edit** menu, select **Add Result…**. The **Add Result Collector** window appears (see Figure 4.5). Or, click the **Add…** button in the **Result Collection** section of the main window.
2. Enter new Result title in the text input field of the same name. You may leave the default title provided by MDTRA, but it is not recommended since Result titles should be legible.
3. In the **Data Source Type** drop-down combo, select a type of data to be collected. Please note that some different data types may be used together; for example, "RMSD" and "RMSD of Selection".
4. To add a Data Source, click **Add…** button to the right of the **Data Source List**. The **Add Result Data Source** dialog appears (see Figure 4.6). Within this dialog, in **Data Source** drop-down combo, select a Data Source to add. Optionally, one may specify S**cale** and **Bias** to be applied to data at the calculation step. Finally, a **Visible on the plot** checkbox makes the Data Source initially visible (it may be toggled later).
5. To add multiple Data Sources at once, click **Multiple…** button to the right of the **Data Source List**. The **Add Multiple Result Data Sources** dialog appears (see Figure 4.7). It is generally the same as the dialog described at step 5, but allows selecting several Data Sources from the list.
6. Result Data Sources can be edited, removed, moved up and down in the list (this affects the position and the color on the Plot). To edit a Result Data Source, select it in the **Data Source List** and click the **Edit…** button; or, double-click the Result Data Source entry in the list. To remove a Result Data Source, select it in the **Data Source List** and click the **Remove** button. Moving of a Result Data Source up and down is performed by clicking the buttons **Up** and **Down** underneath.
7. In the **Scale Units** drop-down combo, select units of measure of data calculated. The choices available depend on **Data Source Type** selected at the step 3.
8. In the **Layout** choices, select the proper argument layout: either **Time-Based** (snapshots, picoseconds or nanoseconds, depending on program settings), **or Residue-Based** (data is averaged along the trajectory for each residue). The choices are available not for all data types.

9. Click the **OK** button to register the Result Collector.

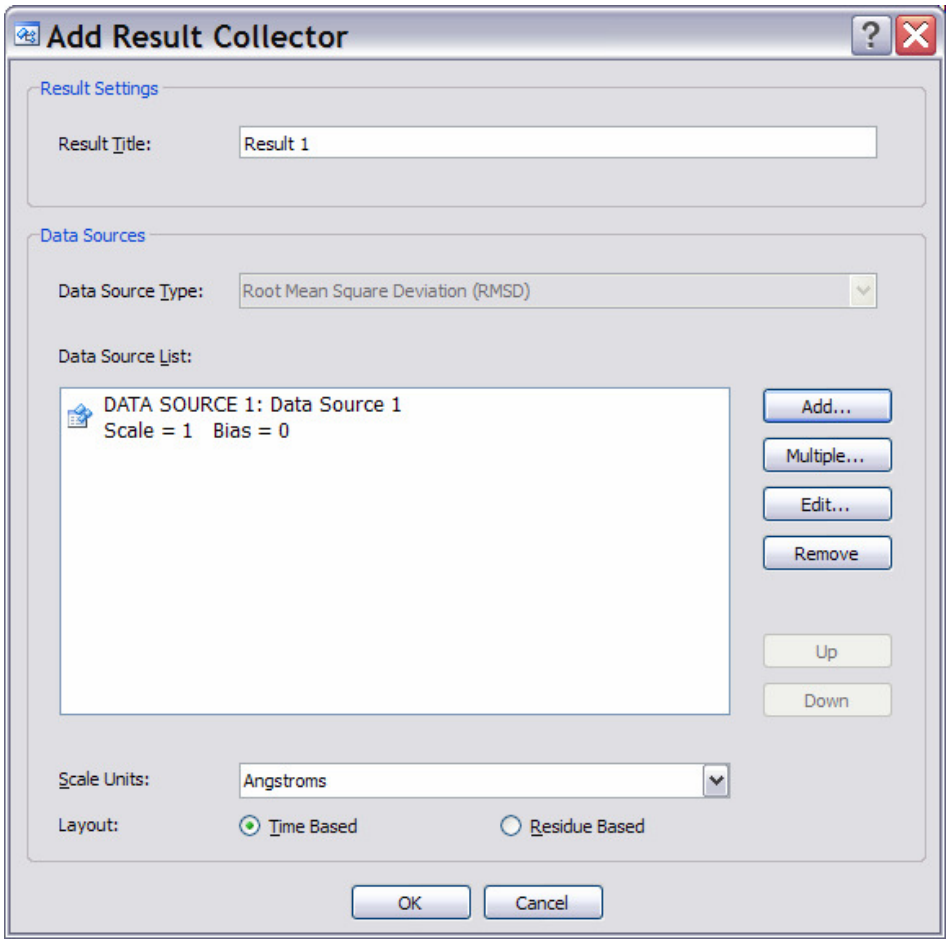


Figure 4.5. The **Add Result Collector** window layout.

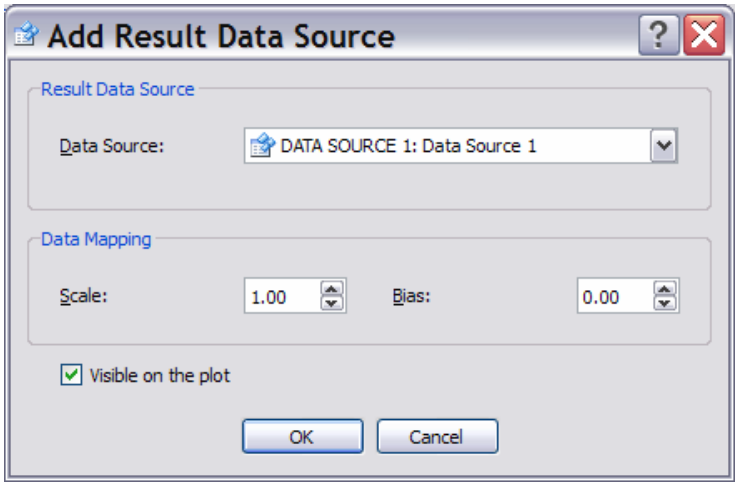


Figure 4.6. The **Add Result Data Source** dialog.

To edit the existing Result Collector, select the Result Collector in the list and click the **Edit…** button below, or double-click the Result Collector entry in the list. To remove the existing

Result Collector, select it in the list and click the **Remove** button. Please note that this action cannot be undone.

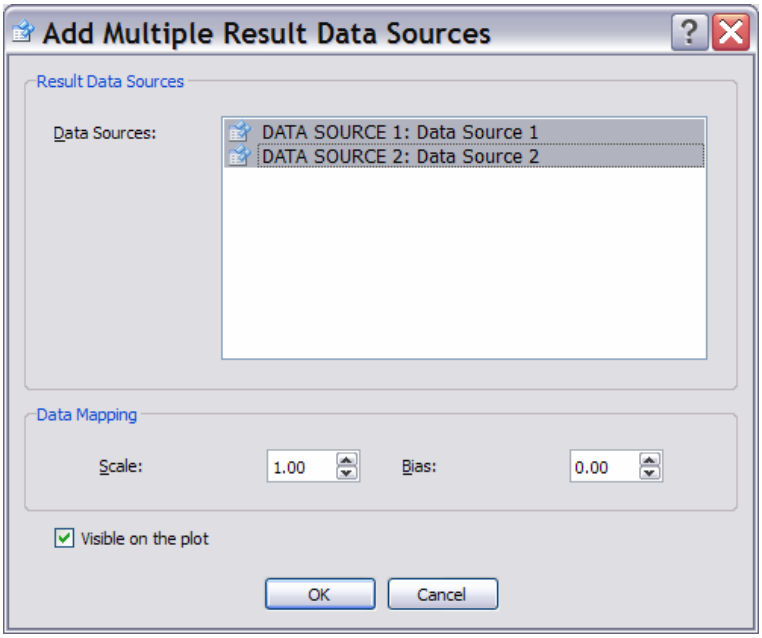


Figure 4.7. The **Add Multiple Result Data Sources** dialog.

4.4. Data Tables

Before data can be viewed, analyzed, and plotted, it must be calculated, or "built". Either invalid or all the result collectors can be built. To rebuild a valid result collector, it must first be invalidated (select it, and then in the **Edit** menu, select **Invalidate Result**; or, click the **Invalidate** button to the right of the **Result Collection** section of the main window).

To build invalid results, in the **Edit** menu, select **Build**. To rebuild all the results, in the **Edit** menu, select **Rebuild All**. It can also be done with the buttons in the bottom right corner of the **Result Collection** section of the main window).

One a build process starts, the **Build Progress** dialog appears (see Figure 4.8). The process may be interrupted by clicking the Cancel button, and only data already build is displayed in the Data Tables.

After data are built, it can be viewed in the **Data Table**. To view a trajectory snapshot associated with certain data, double-click the table row to launch the external viewer, if any.

Statistical parameters of the trajectory within the Result Collector are displayed in the Statistic Table. For the list of statistical parameters, refer to Table 4.2.

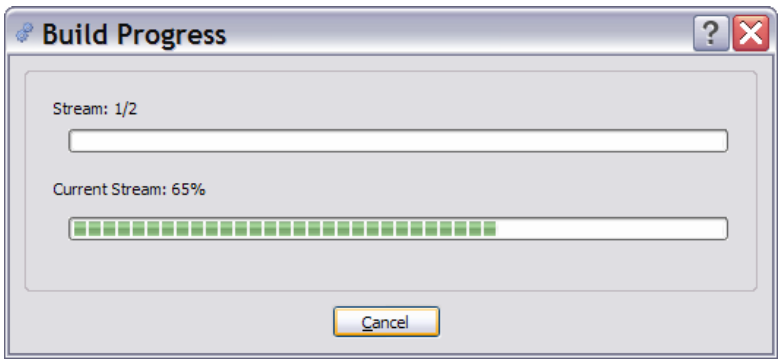


Figure 4.8. The **Build Progress** dialog.

Statistical Parameter	**Symbol**	**Formula**
Arithmetic mean	M	$M = \frac{1}{n}\sum_{i=0}^{n} x_i$
Geometric mean	Mg	$Mg = \sqrt[n]{\prod_{i=0}^{n} x_i}$
Harmonic mean	Mh	$Mh = \frac{1}{\sum_{i=0}^{n} \frac{1}{x_i}}$
Quadratic mean (Root-mean-square)	RMS	$RMS = \sqrt{\frac{1}{n}\sum_{i=0}^{n} x_i^{\,2}}$
Minimum value	Min	$Min = \min(x_i)$
Maximum value	Max	$Max = \max(x_i)$
Range	R	$R = Max - Min$
Midrange	Mr	$Mr = \frac{Max - Min}{2}$
Median	Med	$Med = median(x_i)$
Sample variance	S2	$S^2 = \frac{n}{n-1}\left(\frac{1}{n}\sum_{i=0}^{n} x_i^{\,2} - \left(\frac{1}{n}\sum_{i=0}^{n} x_i\right)^2\right)$
Sample standard deviation	S	$S = \sqrt{S^2} = \sqrt{\frac{n}{n-1}\left(\frac{1}{n}\sum_{i=0}^{n} x_i^{\,2} - \left(\frac{1}{n}\sum_{i=0}^{n} x_i\right)^2\right)}$
Sample standard error	SE	$SE = \frac{S}{\sqrt{n}} = \sqrt{\frac{1}{n-1}\left(\frac{1}{n}\sum_{i=0}^{n} x_i^{\,2} - \left(\frac{1}{n}\sum_{i=0}^{n} x_i\right)^2\right)}$
User-defined parameters	U1 – U6	Refer to section 8.6 for the details

Table 4.2. Statistical parameters calculated along the trajectory.

Data and statistical parameters can be exported to text (TXT) or comma-separated (CSV) formats. To export data:

1. Select the Result Collector in the Result Collection list.
2. In the **File** menu, select **Export Results…** The **Export Result** dialog appears.
3. Select a new file name and a desired file type (either TXT or CSV format), and then click the **Save** button.

To export statistical parameters, perform the same steps, but in the **File** menu, select **Export Statistics** (the **Export Statistics** dialog appears).

4.5. Plot

The data calculated as a function of either snapshot number/time (time-based plots), or residue number (residue-based plots), are displayed on the Plot. Data color can be adjusted in Settings (see Plot Colors). Data can be displayed selectively. To show or hide the data, one may either edit the Result Data Source in the Result Collector (**Visible on the plot** flag), or use the **Plot Data Visibility** dialog (in the Plot menu, **select Plot Data Visibility…**; changes are applied immediately on clicking the check boxes; see Figure 4.9).

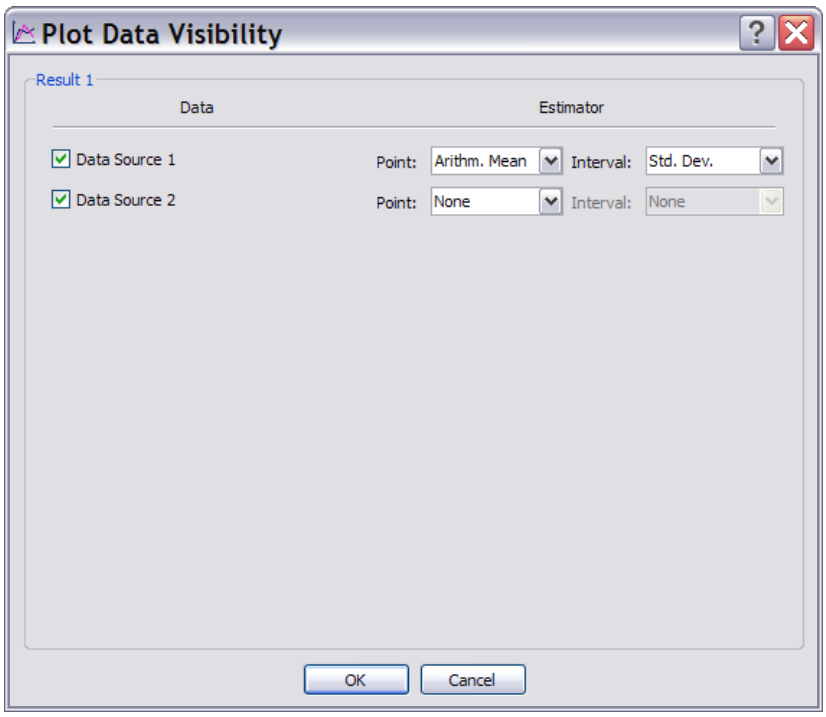


Figure 4.9. The **Plot Data Visibility** dialog.

There are four Plot options defining its visual appearance and behavior, and they can be found either in the **Plot** menu, or at the toolbar:

- **Track mouse cursor**: with this option enabled, a red dot is drawn under the mouse cursor, when it is over the Plot. There are also two dashed lines projecting the dot onto coordinate axes. It can be useful when estimating the data with the Plot.

- **Show grid lines**: toggles horizontal dashed grid lines at major marks of the ordinate axis.
- **Toggle plot labels**: toggles labels on the Plot. If labels are disabled, they cannot be edited, but still can be added.
- **Toggle plot legend**: toggles the legend under the Plot. Disabling the legend gives some extra vertical space for the Plot, but it is not recommended to export images without the legend.

To export the Plot as an image, PDF or PostScript file:

1. In the **Plot** menu, select **Save Image…** The **Save File** dialog appears.
2. Select the desired image format (the number of formats available depends on the operating system; PDF and PostScript files are rendered in a raster form and therefore are also referred to as images) and click **Save**.

Please note that image is saved with the dimensions of the Plot. If you need a higher image resolution, you may rescale the Plot window. For examples of images exported, see Figure 4.10.

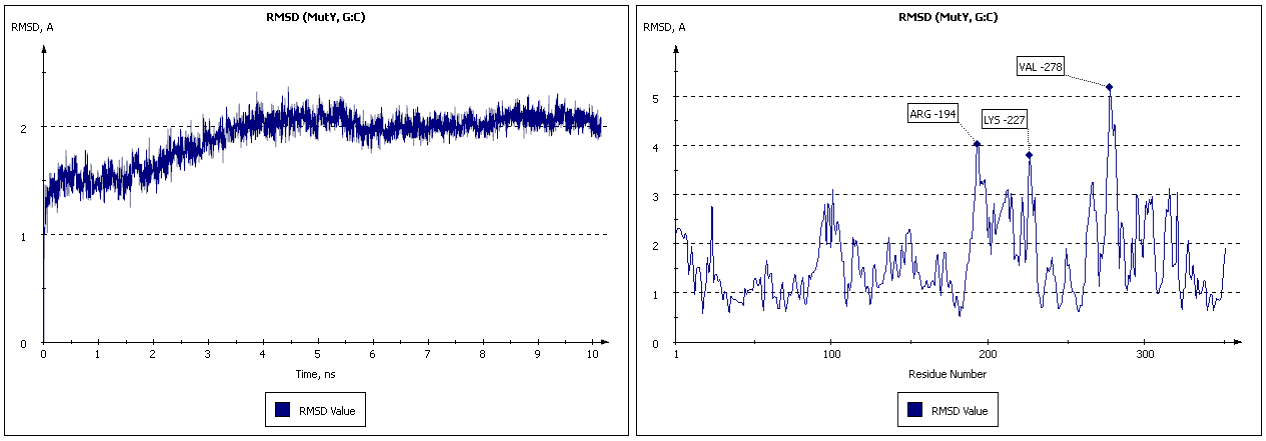


Figure 4.10. Examples of Plot images exported by MDTRA**.**

Plot can be annotated using **Plot Labels**. They are labels attached to a specific **Result Data Source** in the current **Result Collector**, and to a specific snapshot (either "real" snapshot, or residue number, for residue-based plots). To add a label to the Plot:

1. Right-click the Plot at the appropriate point. The context menu appears.
2. In the context menu, select **Add Label…** The **Add Label** dialog appears (see Figure 4.11).
3. The closest Result **Data Source** and **Snapshot** number are already selected, but you may change them. Label **Title** is also prompted automatically and can be altered. Multiple lines are allowed.
4. In the **Label Flags** group, select the proper label flags:

- **Display Border**: draw a border around label text.
- **Solid Background**: fill the label's background with the background color.
- **Connect to Marker**: draw a dotted connection line between the graph marker and the closest corner of the label.
- **Vertical line**: draw a solid line between marker and its projection to X axis.

5. Click the **OK** button to add the label.

To edit a label, one must first select it, either with a left click, or using a selection rectangle (click the Plot and move the mouse cursor to see it). Multiple labels can be selected at once. Hold **Shift** key to invert selection status of a label; hold **Ctrl** key to add a label to current selection. Labels can be dragged by mouse. If it is a single label, it can be edited in a similar way as it was added: invoke a context menu and select **Edit Label…** to open the **Edit Label** dialog. Selected labels can also be removed by selecting **Remove Label(s)** command in a context menu.

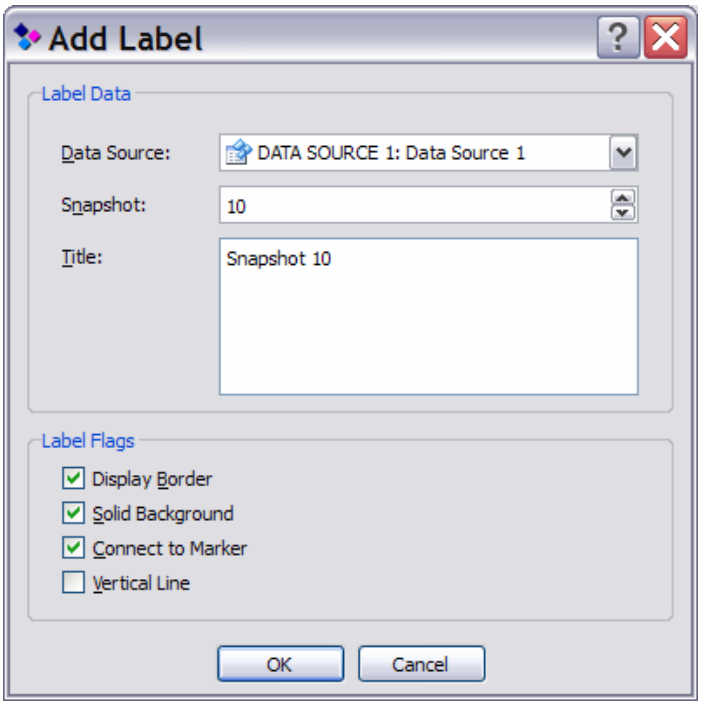


Figure 4.11. The **Add Label** dialog.

There is an ability to zoom in and out of the Plot (using a mouse wheel) and move along the X-axis (using a right mouse button). Images will be exported taking current zoom and offset into account. Please note that zoom and offset are not saved and are lost when one switches to another plot or terminates the MDTRA session.

4.6. Saving and Loading Project Files

The data set up and calculated can be saved into the MDTRA Project File format. This is an internal binary format of data used by the program. To save a project:

1. In the **File** menu, select **Save Project As…** The **Save Project File** dialog appears.
2. Select a directory to save a file, and then click the **Save** button.

You can also save an empty project and reopen it later. To save the changes under the same file name, you may use a Save Project command in the File menu, or a button with the diskette icon at the toolbar.

To open a previously saved project:

1. In the **File** menu, select **Open Project…** The **Open Project File** dialog appears.
2. Select a MDTRA Project File, and then click the **Open** button.

The following data is saved into MDTRA Project files:

- **Streams**: titles, settings and the absolute paths to trajectory files. Trajectory files are not saved within the project, and, if you move or delete them later, MDTRA will not be able to work with the Stream (refer to 4.7).
- **Data sources**: titles, settings, selection expressions if any.
- **Result collectors**: titles, settings, result data sources, plot labels, plot visibility information.
- **Data** and **statistical parameters** calculated.
- **DNA Data Mining** results (refer to 6.6).

The following data is not saved:

- **Distance search results** (refer to 6.1).
- **Torsion search results** (refer to 6.2).
- **H-Bonds search results** (refer to 6.3).
- **2D-RMSD** data (refer to 6.4).
- **Histogram** data (refer to 6.5).

4.7. Working without Trajectory Files

MDTRA can also function with an existing Project File, but without the trajectory files. However, the capabilities of analysis are very limited. Streams, Data Sources and Result Collectors are in a read-only mode, and no new ones can be added. The following operations are permitted in a read-only mode:

- Toggle plot data visibility.
- Add, edit, and remove plot labels.
- Calculation of histograms.
- Export of data and statistical parameters.
- Export of plot image.

All these operations do not depend on the trajectory files at all.

5. Solvent Accessible Surface Calculations

Solvent accessible surface (SAS) is a molecular surface accessible by solvent molecules. MDTRA has several **Data Sources** exploit this concept.

SAS area of the whole molecule or its part (defined by a selection term) is calculated using a Shrake-Rupley algorithm. Each atom is described as a sphere of dots uniformly distributed in space. Radius of the sphere is the sum of atomic van der Waals and solvent probe radii. Solvent probe radius by default is equal to 1.4 Å (an approximate radius of water molecule) and can be edited in MDTRA **Preferences**. Dot density depends on the accuracy selected and corresponds to the number of subdivisions of the initial icosahedron using a geosphere algorithm which produces a high-quality uniform distribution (see Table 5.1).

Accuracy	Subdivisions	# of dots	# of facets
Low	0	12	20
Medium	1	42	80
High	2	162	320
Very high	3	642	1280

Table 5.1. SAS computational accuracy and number of dots per atom.

MDTRA determines which dots are buried (i.e. inaccessible by solvent). SAS area of the atom is calculated using the following term:

$$S = 4\pi R^2 (1 - \frac{N_{buried}}{N_{total}}),$$

where **S** is surface area (in Å^2), **R** is radius of atomic sphere, $\mathbf{N_{buried}}$ is a number of buried dots, and $\mathbf{N_{total}}$ is a total number of dots.

One more concept based on SAS is Occluded Area. This is a part of molecular surface area occluded from solvent by its part (or another molecule). For example, in protein-DNA complexes some amino acid residues form a close contact with the DNA. In absence of DNA they are exposed to the solvent. Their summary surface area is defined as an area of protein's molecular surface occluded by DNA. One can also calculate an occluded area of particular residue, or even build a residue-based occlusion to determine such residues. Occluded area of the atom is calculated using the following term:

$$S = 4\pi R^2 \frac{N_{buried} - N'_{buried}}{N_{total}},$$

where **S** is occluded area (in Å^2), **R** is radius of atomic sphere, $\mathbf{N_{buried}}$ is a total number of buried dots, $\mathbf{N'_{buried}}$ is a number of dots buried by everything except atoms of the occluder, and $\mathbf{N_{total}}$ is a total number of dots.

6. Trajectory Related Search

Some MDTRA instruments do not comprise general data organization (data sources, result collectors) and therefore refer to as trajectory related search tools. They include the Distance Search Tool, the Torsion Search Tool, the H-Bonds Search Tool, the 2D-RMSD Calculation, and the Histogram Tool.

6.1. Distance Search Tool

The Distance Search Tool is a cross-trajectory notable distance estimation tool which may be helpful in a procedure of data source extraction. The main point of the tool is to analyze distance between pairs of atoms in two different trajectories and to decide whether the difference is significant or not. Distance Search results are suggestions that can be added to Result Collectors, or exported to a file, or ignored. The Distance Search dialog layout is shown on the Figure 6.1.

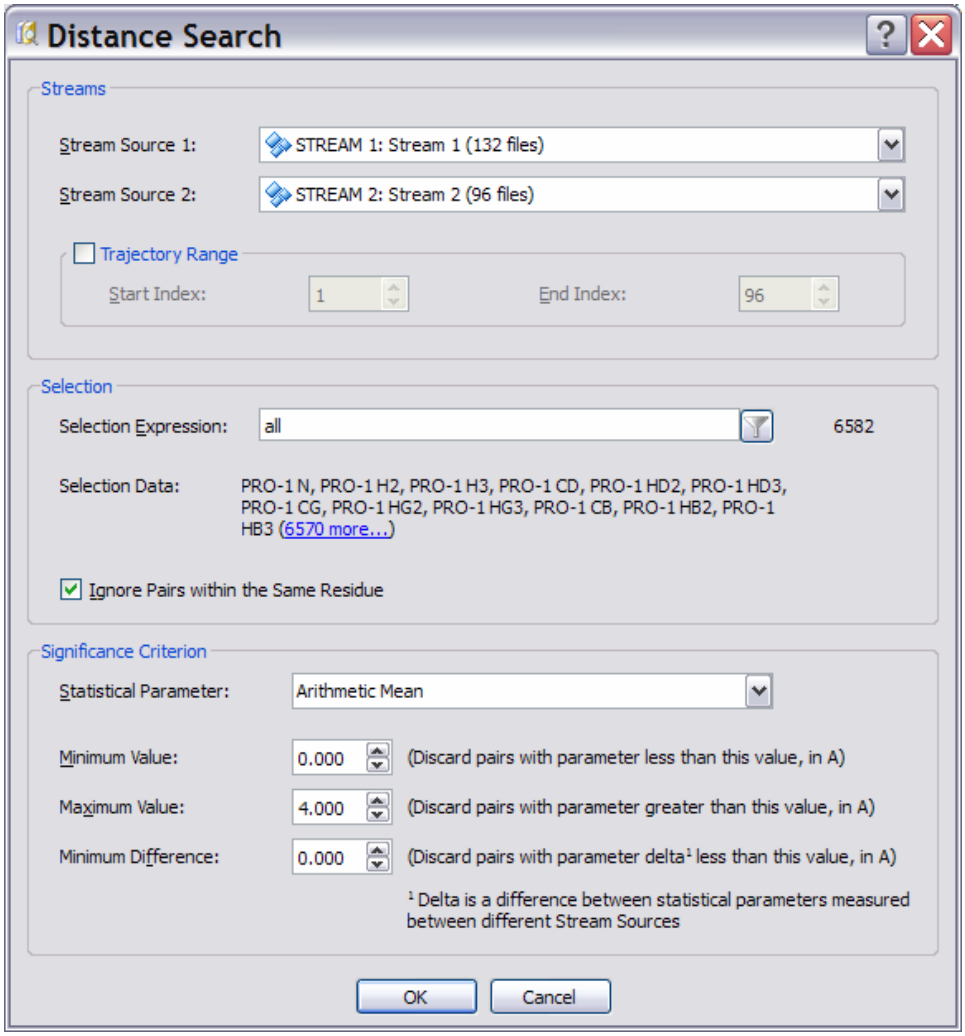
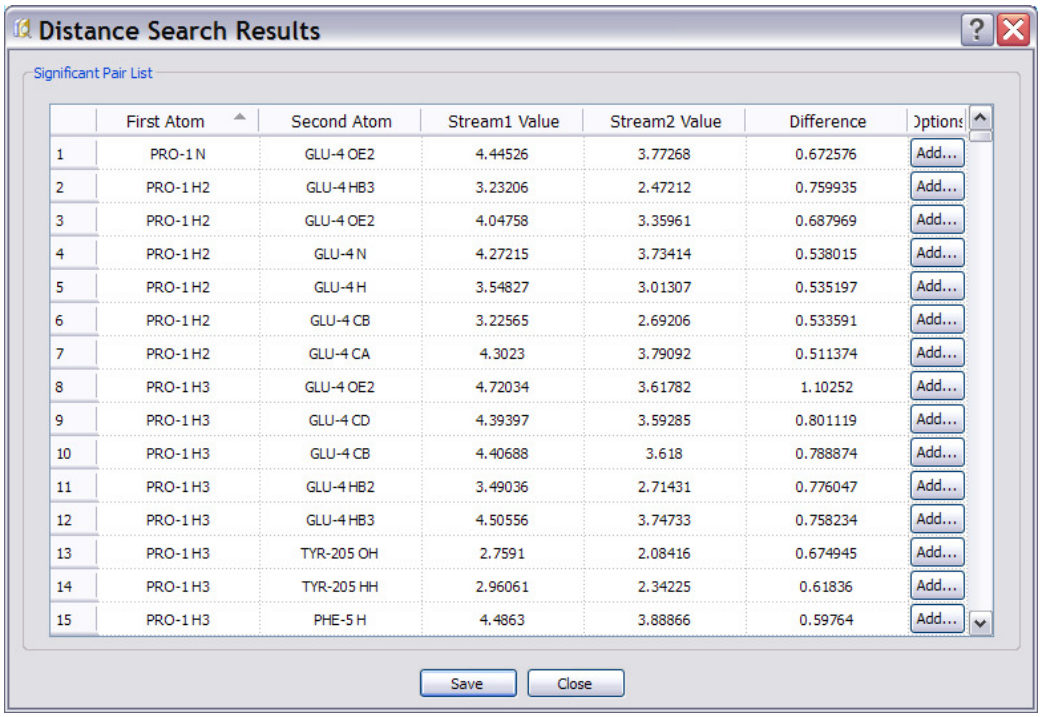


Figure 6.1. The **Distance Search** dialog.

To perform a distance search between two trajectories, both of them must be registered as Streams. Then the following steps should be performed:

1. In the **Tools** menu, select the **Distance Search…** If there are at least two Streams in the current project, the **Distance Search** dialog appears.
2. In the **Streams** section of the dialog, select streams to analyze in drop-down combos. If a trajectory fragment is to be analyzed, check the **Trajectory Range** checkbox and specify a first and a last snapshot number in the Stream list. Please note that this number is relative to the alphabetically sorted file list in the Stream, not the filename index, if any. Since trajectories may be of different size (this is not recommended however), an upper bound of a list (**End Index**) is limited to a size of a lesser file list.
3. In the **Selection** section, define a set of atoms to analyze in a pair wise mode. To avoid calculation of unnecessary distances within the same residue, the **Ignore Pairs within the Same Residue** checkbox is checked by default. It may be unchecked to alter the default behavior. Please note that the selection displayed belongs to the first Stream. The corresponding atoms in the second Stream are addressed using the atom title and residue title. If an atom is missing from either the first or the second Stream, it is not used in search although it matches the selection term.
4. In the **Significance Criterion** section, set up a significance criterion to filter the resulting distances. In the **Statistical Parameter** drop-down combo, select a parameter to calculate for both distance sets along the trajectories. The options available are: mean values (arithmetic, geometric, harmonic, and quadratic), limits (minimum and maximum values), range, midrange, and sample variance. For each atomic pair, the parameter is calculated independently for both trajectories. Then, to describe how they are compared at the final step of the search, three options should be defined: minimum value, maximum value and minimum difference (a brief description of them can be found to the right of the corresponding spin boxes). At the final step, the following distances are discarded:
   - if both of them are less than the **Minimum Value**, in angstroms;
   - if both of them are greater that the **Maximum Value**, in angstroms;
   - if an absolute value of difference between them is less than the **Minimum Difference** value, in angstroms.
5. Click the **OK** button to start the search process. The **Build Progress** dialog appears (see Figure 4.8).

6. After the build process is complete, a **Distance Search Results** dialog appears (see Figure 6.2). Please note that is there is a lot of atomic pairs matching the significance criterion, it may take a lot of time for the dialog to render.



	First Atom	Second Atom	Stream1 Value	Stream2 Value	Difference	Options
1	PRO-1 N	GLU-4 OE2	4.44526	3.77268	0.672576	Add...
2	PRO-1 H2	GLU-4 HB3	3.23206	2.47212	0.759935	Add...
3	PRO-1 H2	GLU-4 OE2	4.04758	3.35961	0.687969	Add...
4	PRO-1 H2	GLU-4 N	4.27215	3.73414	0.538015	Add...
5	PRO-1 H2	GLU-4 H	3.54827	3.01307	0.535197	Add...
6	PRO-1 H2	GLU-4 CB	3.22565	2.69206	0.533591	Add...
7	PRO-1 H2	GLU-4 CA	4.3023	3.79092	0.511374	Add...
8	PRO-1 H3	GLU-4 OE2	4.72034	3.61782	1.10252	Add...
9	PRO-1 H3	GLU-4 CD	4.39397	3.59285	0.801119	Add...
10	PRO-1 H3	GLU-4 CB	4.40688	3.618	0.788874	Add...
11	PRO-1 H3	GLU-4 HB2	3.49036	2.71431	0.776047	Add...
12	PRO-1 H3	GLU-4 HB3	4.50556	3.74733	0.758234	Add...
13	PRO-1 H3	TYR-205 OH	2.7591	2.08416	0.674945	Add...
14	PRO-1 H3	TYR-205 HH	2.96061	2.34225	0.61836	Add...
15	PRO-1 H3	PHE-5 H	4.4863	3.88866	0.59764	Add...

Figure 6.2. The **Distance Search Results** dialog.

In the **Distance Search Results** dialog, it is a list of atomic pairs matching the significance criterion, initially sorted by the Difference in the descending order. The sort order can be altered by clicking the corresponding table vertical header.

To further simplify the analysis, an easy registration of Result Collectors is provided. Click the **Add…** button to open the drop-down menu. From the menu, a new Result Collector containing the distances for both streams, can be registered, or these distances can be added to existing Result Collectors.

To export the results to a file (either TXT or CSV format), click the **Save** button and select an output file. Please note that the file will be overwritten.

After all the necessary distances are registered, click the **Close** button to close the **Distance Search Results** dialog.

6.2. Torsion Search Tool

The Torsion Search Tool is a cross-trajectory notable torsion angle estimation tool which allows locating of distinct backbone or side chain distortions. The main point of the tool is to analyze torsion angles in amino acids and DNA in two different trajectories and to decide whether the difference between two torsions is significant or not. The same as Distance Search results,

Torsion Search results are suggestions that can be added to Result Collectors, or exported to a file, or ignored. The Torsion Search dialog layout is shown on the Figure 6.3. There are nucleic (α, β, γ, δ, ε, ζ, η_0-η_4, χ), protein backbone (φ, ψ) and side chain torsion angles are monitored.

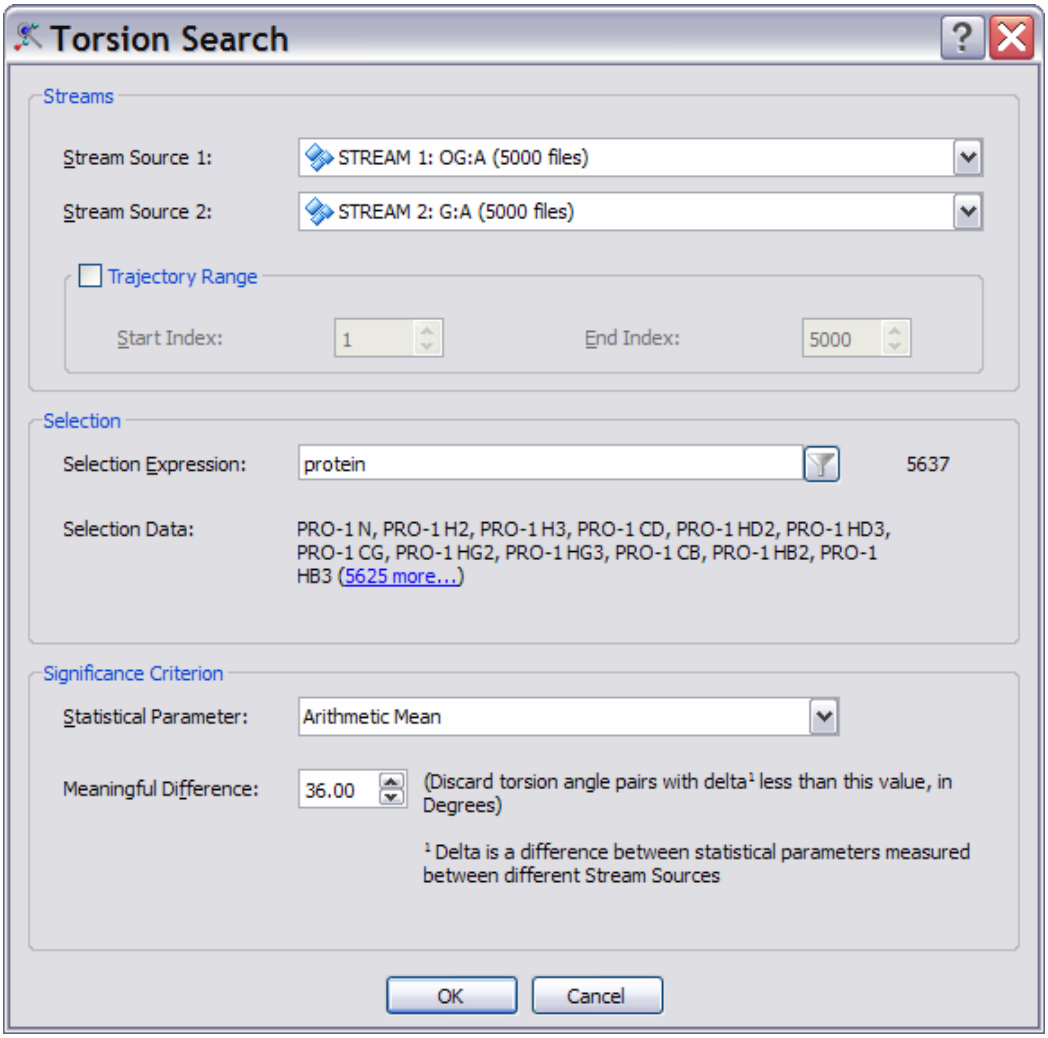


Figure 6.3. The **Torsion Search** dialog.

To perform a torsion search between two trajectories, both of them must be registered as Streams. Then the following steps should be performed:

1. In the **Tools** menu, select the **Torsion Search…** If there are at least two Streams in the current project, the **Torsion Search** dialog appears.
2. The **Streams** and the **Selection** sections are similar to those in the **Distance Search** dialog (refer to section 6.1).
3. In the **Significance Criterion** section, set up a significance criterion to filter the resulting distances. In the **Statistical Parameter** drop-down combo, select a parameter to calculate for both distance sets along the trajectories. The options available are: mean values (arithmetic, geometric, harmonic, and quadratic), limits (minimum and maximum values),

range, midrange, and sample variance. For each torsion angle, the parameter is calculated independently for both trajectories. Then define a meaningful difference value to determine which torsion angle pairs must be discarded.

4. Click the **OK** button to start the search process. The **Build Progress** dialog appears (see Figure 4.8).
5. After the build process is complete, a **Torsion Search Results** dialog appears (see Figure 6.4Figure 6.2). Please note that is there is a lot of torsion angle pairs matching the significance criterion, it may take a lot of time for the dialog to render.

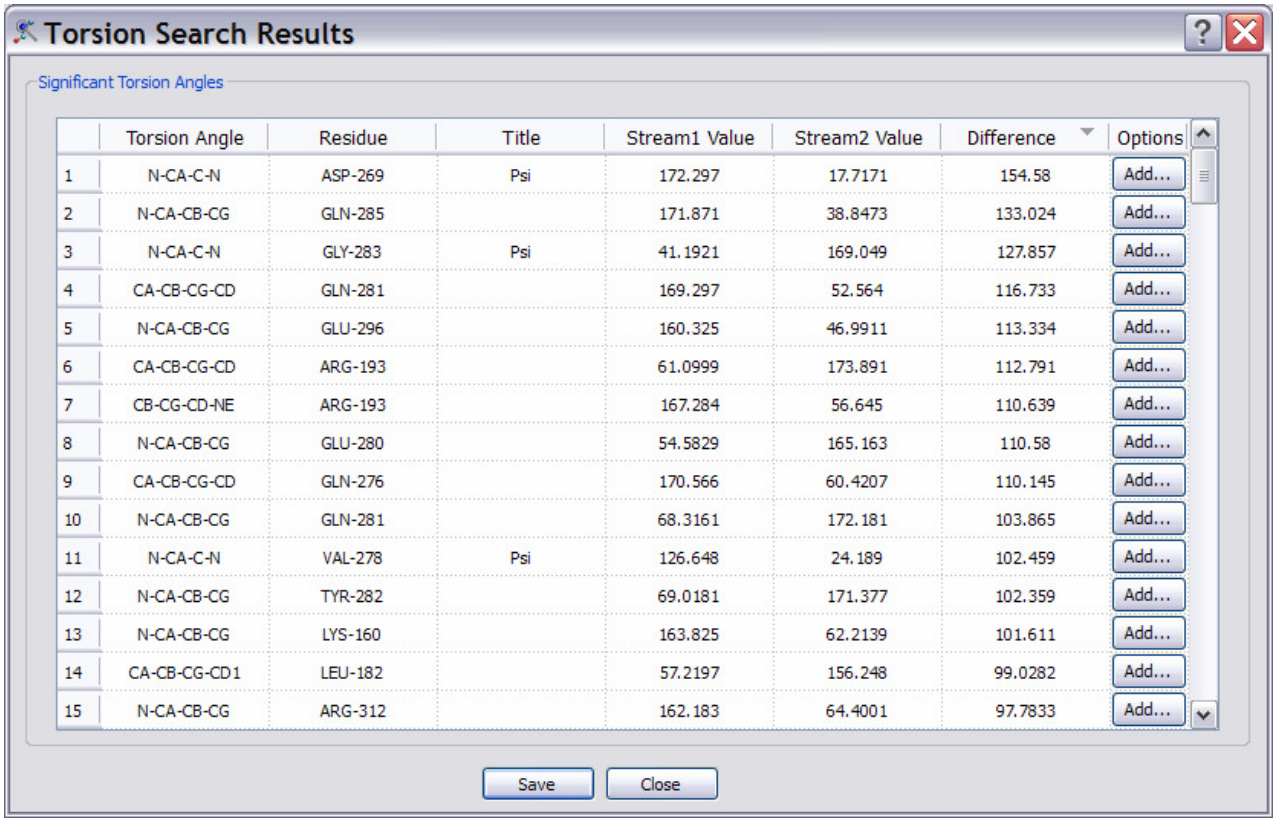


Figure 6.4. The **Torsion Search Results** dialog.

In the **Torsion Search Results** dialog, it is a list of torsion angles matching the significance criterion, initially sorted by the Residue in the ascending order. The sort order can be altered by clicking the corresponding table vertical header.

An easy registration of Result Collectors, and exporting features are also provided in the same way as in the Distance Search Results dialog (refer to section 6.1).

After all the necessary torsion angles are registered, click the **Close** button to close the **Torsion Search Results** dialog.

6.3. H-Bonds Search Tool

The H-Bonds Search Tool is a tool designed to find hydrogen bonds that stabilize the structure in meaningful time scale. It allows specifying a significance criterion to either accept or discard a bond based on estimated bond energy and occurrence along the trajectory. H-Bonds Search results are suggestions that can be added to Result Collectors or ignored. The H-Bonds Search dialog layout is shown on the Figure 6.5.

The process of hydrogen bond search can be described as follows. First, MDTRA builds a set of all possible X–H..Y triplets (where X is a donor, Y is an acceptor, H is a hydrogen, dash means a covalent bond and two dots mean a hydrogen bond) based on it's internal knowing about protein and DNA residues. Second, for each trajectory snapshot, MDTRA calculates an energy term (in kcal/mol) for each triplet using a 12-6 model:

$$E = E_m\left[\left(\frac{R_m}{r}\right)^{12} - 2\left(\frac{R_m}{r}\right)^{6}\right]e^{-\left(\frac{1+\cos\theta}{s}\right)^2},$$

where **E** is a hydrogen bond energy value, **r** is a distance between Y and H atoms (in angstroms), **θ** is a triplet angle, $\mathbf{R_m}$ is a van der Waals radius of an acceptor atom, in angstroms, and $\mathbf{E_m}$ is potential well depth, in kcal/mol. These parameters depend on AMBER codes of Y and H atoms and were fit for best matching of hydrogen bonding in native structures using BioPASED force field, a slightly modified AMBER99 force field. Parameter **s** is a constant of softness with a value of 0.134. If it is not a zero and its absolute value is greater than the value defined in significance criterion (if any), the bond is assumed to exist in the current snapshot.

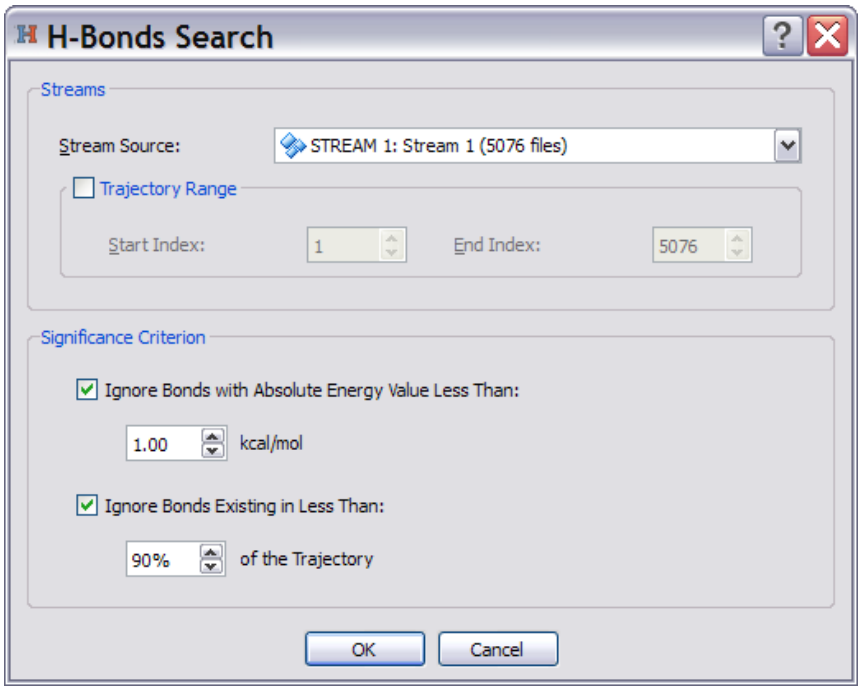


Figure 6.5. The **H-Bonds Search** dialog.

Finally, MDTRA collects all triplets with hydrogen bond occurrence greater than the value defined in significance criterion (if any) into a table of results.

To perform a hydrogen bonds search along a trajectory, it must be registered as Stream. Then the following steps should be performed:

1. In the **Tools** menu, select the **H-Bonds Search…** If there is a valid Stream in the current project, the **H-Bonds Search** dialog appears.
2. In the **Streams** section of the dialog, select a stream to analyze in a drop-down combo. If a trajectory fragment is to be analyzed, check the **Trajectory Range** checkbox and specify a first and a last snapshot number in the Stream list. Please note that this number is relative to the alphabetically sorted file list in the Stream, not the filename index, if any.
3. In the **Significance Criterion** section, set up a significance criterion to filter the bonds found. The **Ignore bonds with absolute energy value less than … kcal/mol** checkbox allows to define an energy filter (all bonds with lesser energy will be discarded). The **Ignore bonds existing in less than … of the trajectory** checkbox allows to define a meaningful percentage of the trajectory. If some bond exists in a smaller part of the trajectory, it will be also discarded.
4. Click the **OK** button to start the search process. The **Build Progress** dialog appears (see Figure 4.8).

After the build process is complete, a **H-Bonds Search Results** dialog appears (see Figure 6.5). Please note that is there is a lot of hydrogen bonds matching the significance criterion, it may take a lot of time for the dialog to render.

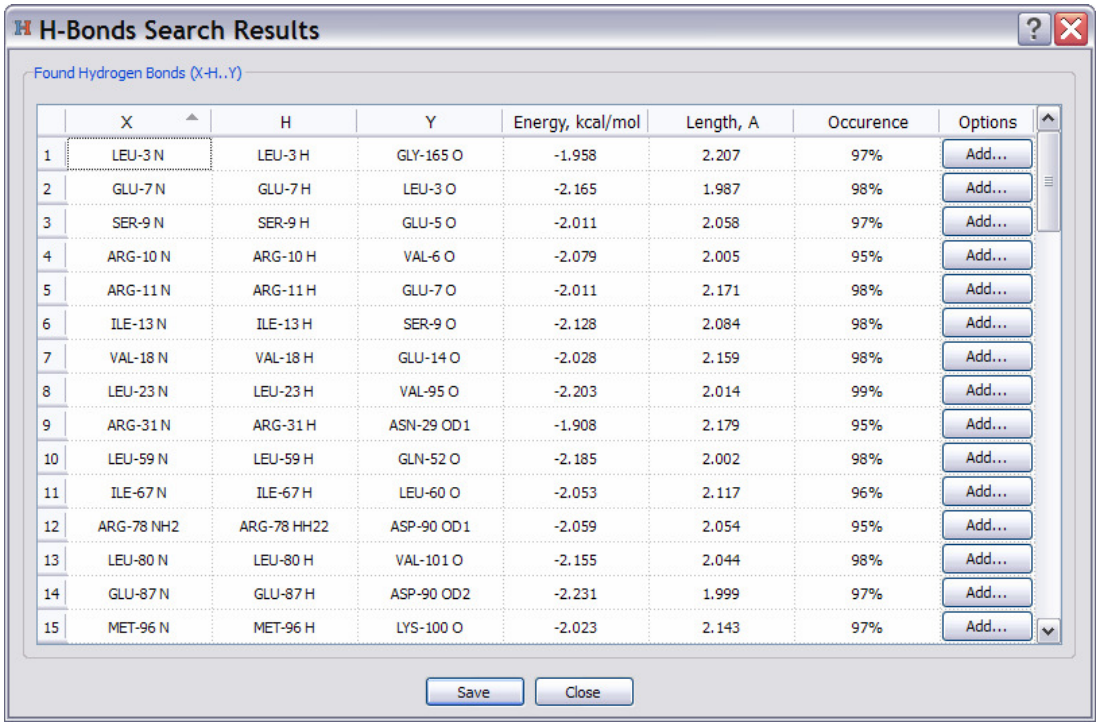


Figure 6.6. The **H-Bonds Search Results** dialog.

In the **H-Bonds Search Results** dialog, there is a list of existing hydrogen bonds matching the significance criterion, initially sorted by the X atom serial number (the first column). The sort order can be altered by clicking the corresponding table vertical header. Average hydrogen bond length (in angstroms) is defined as a mean value of X–Y distances in those snapshots matching the significance criterion.

To further simplify the analysis, an easy registration of Result Collectors is provided. Click the **Add…** button to open the drop-down menu. There are three submenus for different result types: 1) X–Y distance; 2) H–Y distance; 3) X–H–Y angle. From each submenu, a new Result Collector containing a distance, or an angle, can be registered, or these distances/angles can be added to existing Result Collectors.

To export the results to a file (either TXT or CSV format), click the **Save** button and select an output file. Please note that the file will be overwritten.

After all the necessary distances, or angles, are registered, click the **Close** button to close the **H-Bonds Search Results** dialog.

6.4. 2D-RMSD Calculation

With the 2D RMSD Tool, special RMSD diagrams that contain a relative deviation between snapshot pairs, can be built. The 2D RMSD dialog layout is shown on the Figure 6.7.

To build a 2D RMSD map for a trajectory, it must be registered as a Stream. Then the following steps should be performed:

1. In the **Tools** menu, select the **Calculate 2D-RMSD…** If there is a valid Stream in the current project, the **2D RMSD** dialog appears.
2. In the **Stream** section of the dialog, select a stream to use in the drop-down combo. If a trajectory fragment is to be analyzed, check the **Trajectory Range** checkbox and specify a first and a last snapshot number in the Stream list. Please note that this number is relative to the alphabetically sorted file list in the Stream, not the filename index, if any.
3. In the **Selection** section, define a set of atoms. A root-mean-square deviation is calculated for each atom in the set and then averaged to get a value for the entire snapshot.

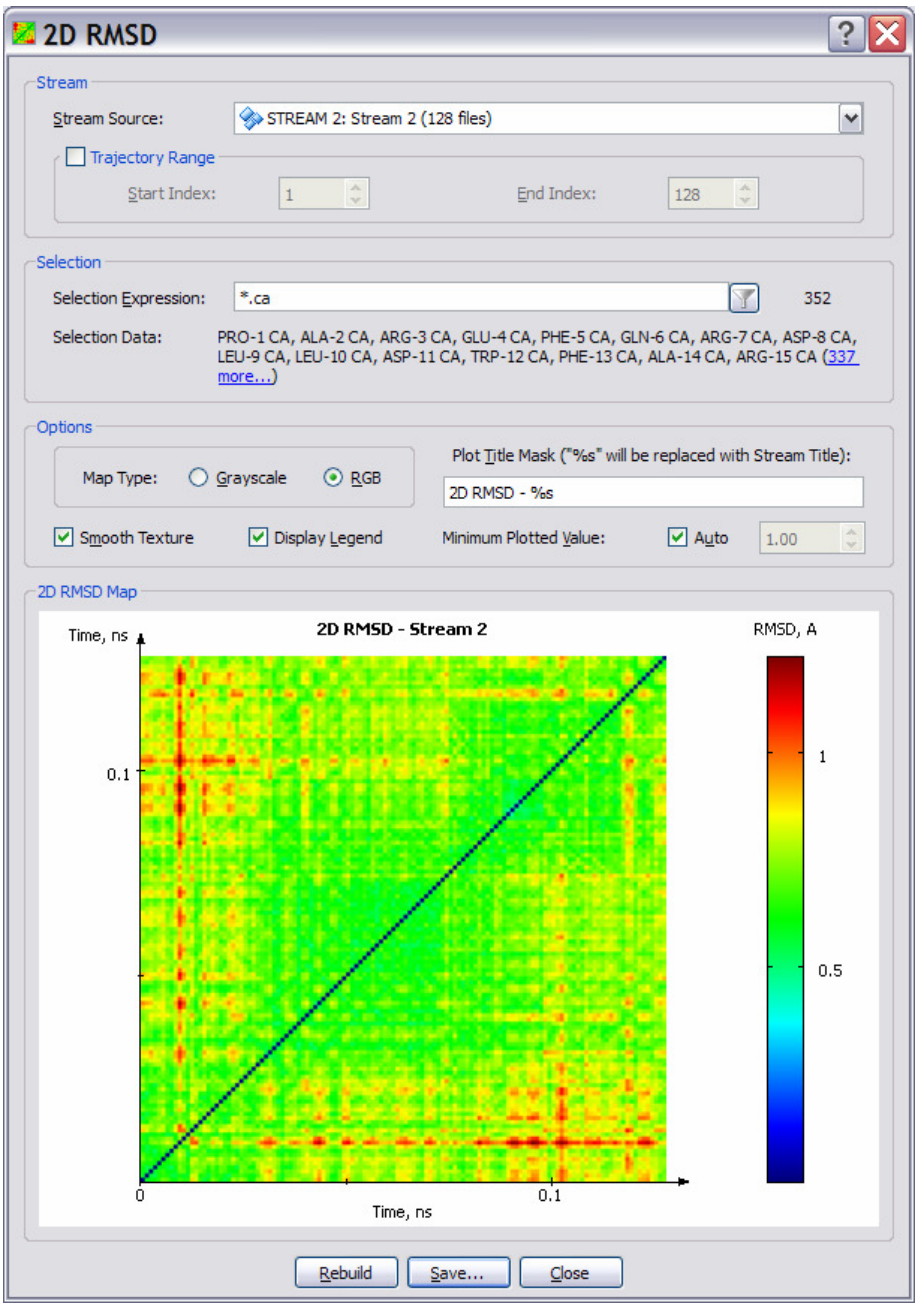


Figure 6.7. The **2D RMSD** dialog.

4. In the **Options** section, select the **Map Type**. It can be either grayscale (the **Grayscale** option) or colored (the **RGB** option, which stands for "Red, Green, Blue"). Provide a **Plot Title Mask**, if necessary (a "%s" control character sequence will be replaced by Stream title). By default, a bilinear filter is applied to the plot texture (uncheck the **Smooth Texture** checkbox to disable) and a legend is displayed (uncheck the **Display Legend** checkbox to hide a legend). The last setting, the **Minimum Plotted Value**, defines a minimum RMSD value, in angstroms, to be referenced by the plot (**Auto** means that MDTRA decides which value to choose; it is equal to 1.0 A in current release; otherwise you may specify your own minimum plotted value using the spin

box). The maximum value is determined by the actual RMSD data array. For example, if a maximum value in data array is equal to 1.4 A, and minimum plotted value is set to 1.0 A, the plot encodes colors in the range between 0.0 and 1.4 A (the actual maximum). If a maximum value in data array is equal to 0.8 A, and minimum plotted value is set to 1.0 A, the plot encodes colors in the range between 0.0 and 1.0 A (the minimum plotted value). This option is useful when rendering different 2D RMSD plots with the same color scales.

5. Click the **Rebuild** button to start the calculation process. The amount of actions included depends on changes made to the settings. For example, if a Stream was not changed, PDB files won't be reloaded, etc. For the first time the **Rebuild** button is pressed, everything is recalculated.
6. After a 2D RMSD plot is rendered, click the **Save…** button. The **Save 2D-RMSD Plot** dialog appears. Select a new file name and a desired image type, and then click the **Save** button.
7. Click the **Close** button to close the **2D RMSD** dialog. All its caches are flushed and memory is freed at this step.

Every pixel at the 2D-RMSD Plot with coordinates (X, Y) represents an average RMSD for the atom set selected, between trajectory snapshots X and Y. This elucidates a diagonal symmetry of the plot. The diagonal itself is either black (in grayscale mode) or dark blue (in colored mode), because the RMSD between identical snapshots is equal to zero.

Please note that 2D RMSD Tool attempts to load all the trajectory files used in calculation, into memory. This may require a huge amount of free RAM, depending on trajectory size (or trajectory fragment size). If there is not enough free memory to load the files, the behavior of MDTRA is undefined.

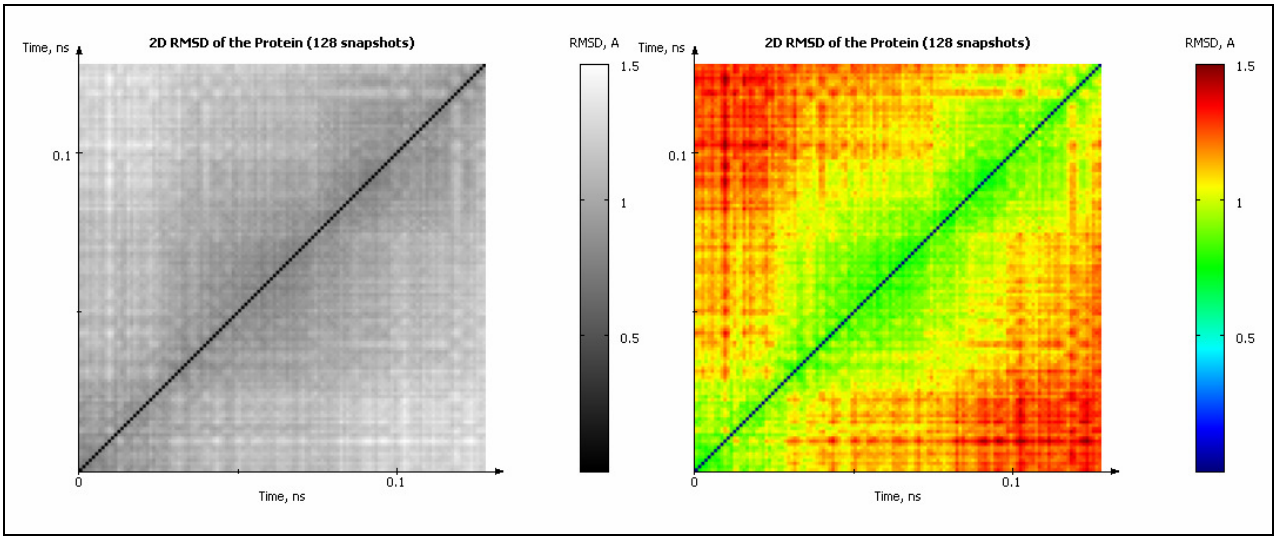


Figure 6.8. 2D RMSD plots, grayscale and colored, generated by MDTRA.

Examples of grayscale and colored plots generated by the 2D RMSD Tool are shown on the Figure 6.8.

6.5. Histogram Tool

With the Histogram Tool, data distribution histograms can be built. The Histogram dialog layout is shown on the Figure 6.9.

To build a histogram for a result data source, it must be set up within a result collector. Then the following steps should be performed:

1. In the **Tools** menu, select the **Build Histogram…** If there is a valid Result Collector in the current project, the **Histogram** dialog appears.
2. In the **Result** section of the dialog, select a Result Collector and a Result Data Source to use in the drop-down combos. If you have selected any Result Collector at the MDTRA result list, it will be also selected here. If histogram calculation takes a noticeable amount of time, you may uncheck the **Automatic Rebuild** checkbox and rebuild the histogram manually by clicking the **Rebuild** button.
3. In the **Options** section, select number of bins. You may allow MDTRA to select it automatically using one of predefined functions:
   - **Square-root choice**: $N = \sqrt{n}$ (this formula is simple and wide-spread; e.g. Microsoft Excel uses it).
   - **Cubic-root choice**: $N = \sqrt[3]{n}$.
   - **Sturges' formula**: $N = \lceil \log_2 n + 1 \rceil$ (brackets mean "ceiling" function).

   Here, **N** is number of bins, **n** is data size (e.g. number of snapshots in a stream).
   You may also specify number of bins manually.
4. In the **Options** section, select bin coloring mode. **Auto** assigns data colors to bins in order, **Grayscale** does the same but desaturates each color first, and **Single Color** allows using any single color for all bins. In the latter case, click the **…** button to specify the custom color (by default this is the first plot data color).
5. Click the **Rebuild** button to build the histogram. If the **Automatic Rebuild** checkbox is checked, it will be rebuilt automatically as soon as you change its parameters.
6. After a histogram plot is built, click the **Save…** button. The **Save Histogram Plot** dialog appears. Select a new file name and a desired image type, and then click the **Save** button. To save numeric histogram data, click the **Export…** button. The **Export Histogram Data** dialog appears. Select a new file name and a desired data format (either TXT or CSV format), and then click the **Save** button.
7. Click the **Close** button to close the **Histogram** dialog.

Histogram tool operates on data already calculated, therefore Result Collectors must be built prior to building of a histogram.

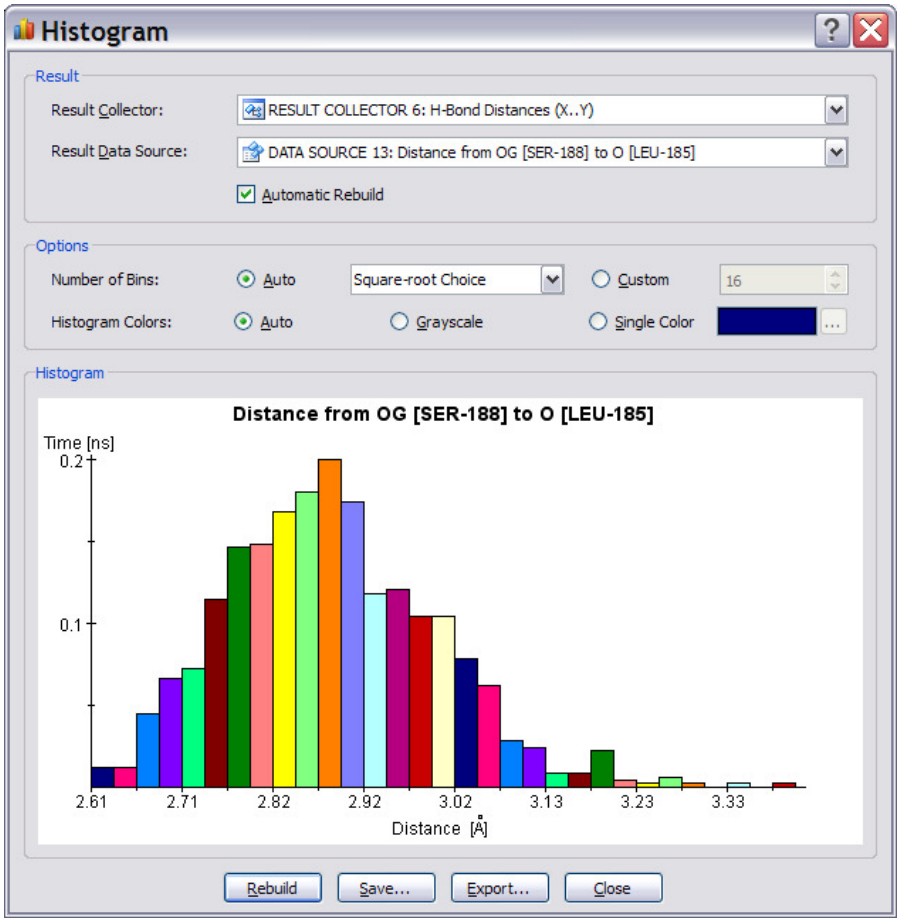


Figure 6.9. The **Histogram** dialog.

6.6. DNA Data Mining Tool

The DNA Data Mining tool is designed to facilitate analysis of DNA structural parameters. It enables quick massive building of Data Sources and Result Collectors of the parameters selected. The following parameters are available: nucleotide torsion angles (α, β, γ, δ, ε, ζ, η_0-η_4, χ), phase angle, Dickerson parameters for base pairs (shear, stretch, stagger, buckle, propeller, and opening) and for neighbor base pairs (shift, slide, rise, tilt, roll, and twist). The tool automatically determines which of the parameters selected are appropriate within the selection term, including base pair detection.

To build a set of Result Collectors of DNA parameters of one or more trajectories, they must be registered as Streams. Then the following steps should be performed:

1. In the **Tools** menu, select the **DNA Data Mining…** If there is a valid Stream in the current project, the **DNA Data Mining** dialog appears (see Figure 6.10).
2. In the **Streams** section of the dialog, select streams to use in the list.

3. In the **Selection** section, define a set of atoms. The tool actually operates on residues, not atoms, so it is enough to define one atom per residue to take that residue into account.
4. In the **DNA Data** section, select DNA parameters to mine. Clicking a checkbox of a subsection header allows disabling the whole group of parameters. The "**(All)**" checkbox inside a group allows to check or uncheck all the parameters within the group.
5. Click the OK button and confirm the number of Data Sources and Result Collectors to be added to the current project. After everything is registered (the process may take some time, depending on that number), the dialog box closes.

Please note that on subsequent calls to **DNA Data Mining** tool may generate duplicate Result Collectors, but Data Sources won't be duplicated.

After closing **DNA Data Mining** dialog, new Result Collectors must be rebuilt manually using the **Build** command.

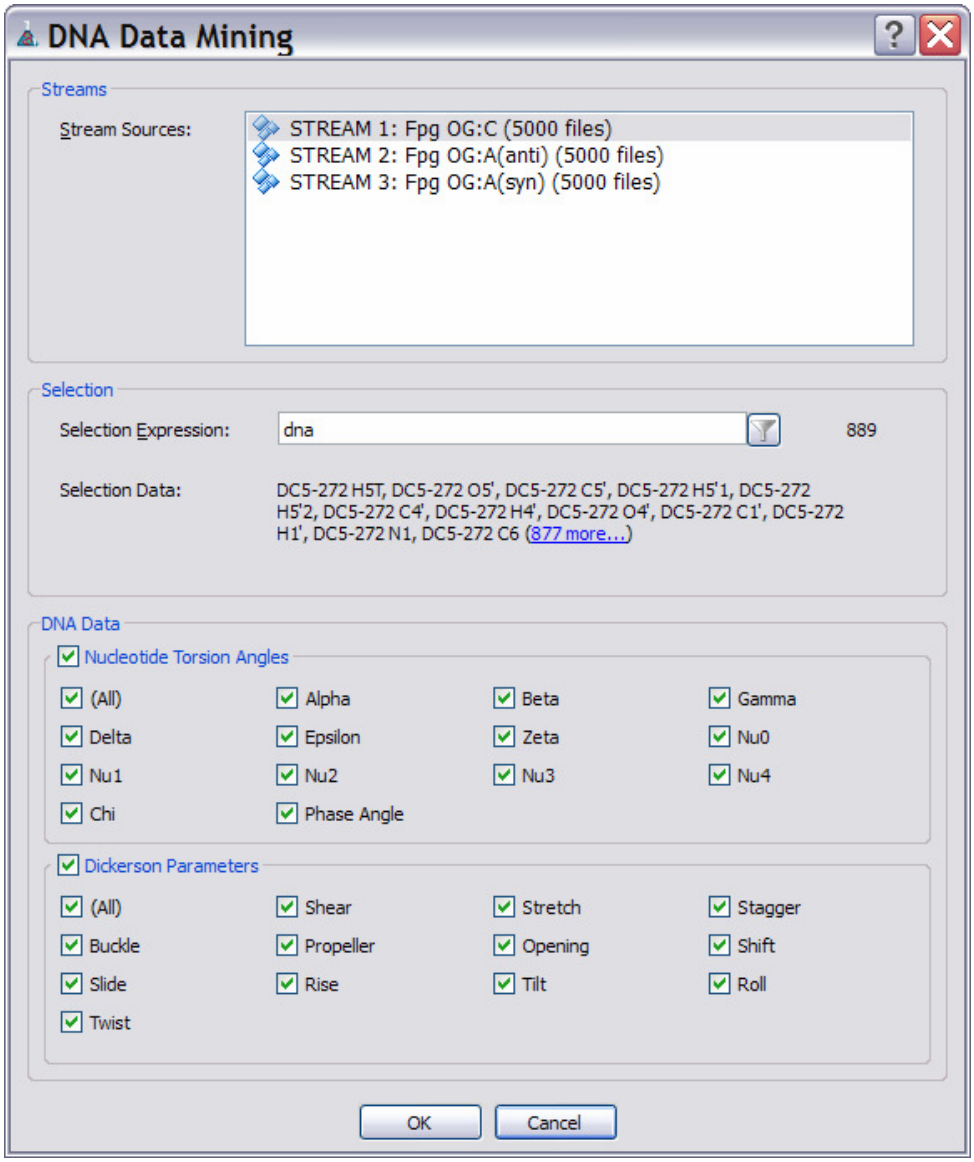


Figure 6.10. The **DNA Data Mining** dialog.

7. Model Preparation Tools

MDTRA provides an ability to perform some computational tasks during structure preparation process. First of all, the preparative trajectories should be obtained and loaded into MDTRA as Streams. Then prep tools can be used to refine, or somehow modify the input models using intensive analysis of preparative trajectories.

Currently, only the **Water Shell Optimization** tool is available.

7.1. Water Shell Optimization

To account for molecular effects, such as van der Waals interactions, hydrogen bonds and water bridges, but keeping computational cost at the minimum, one should use a reliable tool for searching of tightly bound water molecules in an adjacent solvent layer. These molecules can be modeled explicitly, and computational efficiency depends on careful minimization of molecule count. At the first step a molecule of a biopolymer should be solvated, i.e. surrounded by a layer of arbitrary oriented water molecules within some water shell radius. At the second step the derived structure is optimized using molecular dynamics method implemented in any molecular dynamics package (e.g. AMBER or BioPASED), gradually heating the system and modeling in equilibrium state for several nanoseconds. Moreover the structure of the biopolymer should be positionally restrained, because the point is to relax and optimize solvent only. Solvent is, on the contrary, must not be restrained, and water molecules feel free to sample the space for local energy minimums. The resulting trajectories are analyzed at the third step in MDTRA using **Water Shell Optimization** prep module (see Figure 7.1). MDTRA utilizes an external module to discriminate water molecules forming hydrogen bonds with two, three or more donors and acceptors of the biopolymer, along the trajectory. Finally a model of the biopolymer containing these explicitly associated, i.e. "tightly bound", water molecules will be built and output with the specified file name. The derived model can then be implicitly solvated, and contribution of associated water molecules is considered explicitly within the bounds of the selected explicit solvent model (TIP3P, TIP4P and so on).

To prepare an optimized water shell for a biopolymer, one should prepare an explicitly solvated trajectory using his favorite modeling package, then register the trajectory as Stream. Then the following steps should be performed:

1. In the **Prep** menu, select the **Water Shell Optimization…** If there is a valid Stream in the current project, the **Water Shell** dialog appears (see Figure 7.1).
2. In the **Streams** section of the dialog, select streams to analyze in drop-down combos. If a trajectory fragment is to be analyzed, check the **Trajectory Range** checkbox and specify a

first and a last snapshot number in the Stream list. Please note that this number is relative to the alphabetically sorted file list in the Stream, not the filename index, if any.

3. In the **Options** section, select output PDB file name. This will be the PDB containing only tightly bound water molecules found by the algorithm. If you leave the field intact, the default PDB name will be suggested by MDTRA.
4. Click the OK button. MDTRA will invoke external program (**wbr** in the "utils" directory). Please wait until the computation process is finished (it may take some time). Then you can safely close the dialog and, in case of successful finish, use the output PDB as new model.

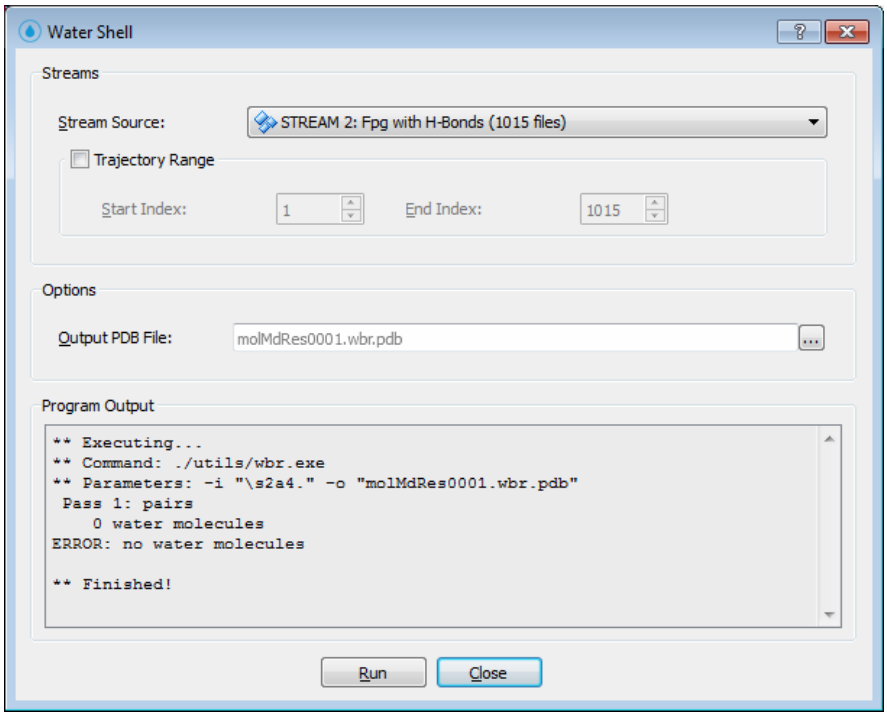


Figure 7.1. The **Water Shell Optimization** dialog.

Please note that if trajectory PDB files do not contain water molecules, the algorithm will bail out with an error "no water molecules". In this case no output PDB will be generated.

8. Programmable Data Sources

There are many different data sources and statistical parameters available within MDTRA, but it may be not enough for deep analysis. Fortunately, MDTRA introduces an extension mechanism: **programmable data sources**. It allows a user to define own data to calculate along the trajectory, and own statistical parameters to analyze. Custom programs can be written, within the limits of MDTRA API, and immediately executed. This doesn't require any third-party script interpreter or compiler; everything it done within a MDTRA project. Please note that custom data source programming requires mastering scripting or programming languages to some degree. The User's Guide assumes that a reader is familiar with a procedural programming paradigm.

8.1. Basic Concept of User-Defined Type

User-defined data types were designed to extend functionality of MDTRA. They actually are small programs written in **Lua** programming language, version 5.1, with some minor modifications (see 8.3). Such programs first need to be compiled to byte code which is executed during the build process. There are two possible entry functions: **main** and **reduce**, that do not accept any arguments. All necessary data can be accessed through MDTRA API (see 8.4).

The main function is called on every trajectory snapshot and is declared as follows:

```
function main()

  return 0.0;

end;
```

The function must return a floating-point value which is put into the Data Table. Please note that **Lua** programs perform calculation with double floating-point precision, but the MDTRA itself operates floating-point values of single precision. The result of **main** may be converted from double to float with a possible loss of data. Please note that in residue-based data layout a return value of **main** is ignored; instead, a per-residue data table should be filled using **rboutput**.

The **reduce** function is called after the trajectory data is calculated. It's purpose is to finalize trajectory-related calculations and compute custom statistical parameters (see 8.6). The function may also override a trajectory data (see 8.7). The default **reduce** function is declared as follows:

```
function reduce()

  return nil;

end;
```

This function does nothing and does not specify any user-defined statistical parameter.

The **main** function is mandatory, a program won't compile if it is not specified. The **reduce** function is optional.

8.2. Editing and Compiling a Program

A user-defined Data Source is added in the same way as a normal Data Source (see 4.2). After selecting a "User-defined Type" in the **Data Type** combo, click the **Advanced…** button to open the **Edit User Type** dialog (see Figure 8.1).

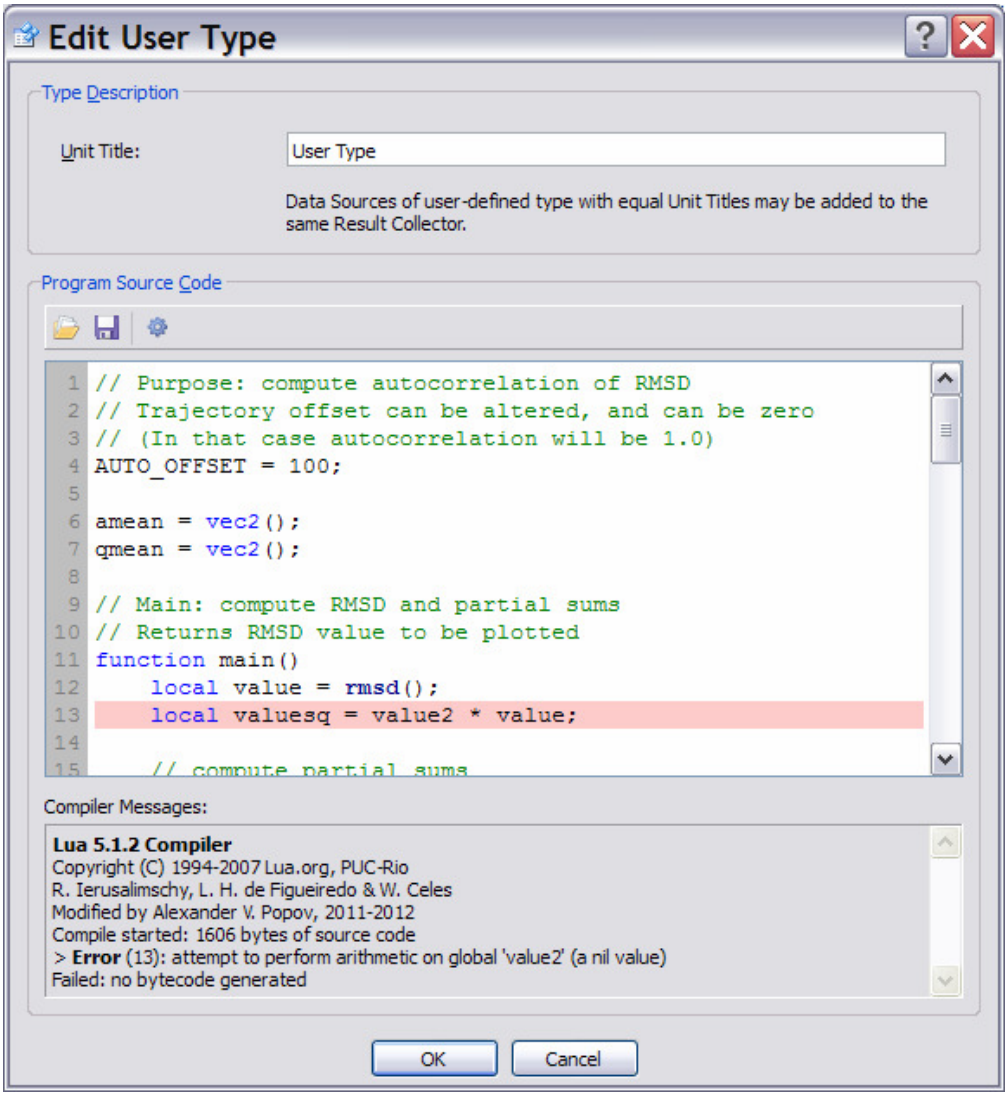


Figure 8.1. The **Edit User Type** dialog.

To declare a custom Data Source, the following steps should be performed:

1. In the **Type Description** section, enter the title of new unit. It usually refers to Y axis title (and it will be displayed as Y axis title on the plot, too), but this meaning is quite relative. The main point is that Data Sources of the similar **Unit Title** may be added to the same Result Collector.

2. In the **Program Source Code** section, write a **Lua** program with a **main** function returning some value. This value will represent a trajectory data of the Data Source.
3. Click the **Compile** button at the toolbar to compile program, perform a test run and generate a byte code. If there are errors, they will appear amongst the **Compiler Messages**. Please note that the User Type with invalid program cannot be saved. All errors must be corrected.
4. Click the **OK** button to save the User Type. If you did not compile the source code, it will be compiled. If there were no errors, the dialog disappears.

The code editing text field has some useful features: 1) automatic line numbering; 2) syntax highlighting (refer to 10.5 on the information how to change highlight colors); 3) referenced line highlighting by double-clicking the reference in **Compiler Messages** field.

The source code may be saved to external file (with a **.lua** extension) and loaded from such file. The default directory to save scripts into and load scripts from is the "scripts" directory. There are some predefined MDTRA scripts that can be used; for example, computing of autocorrelation.

Compiler messages may reference a line of the program's source code, or may not. If a message references a line (for example, an error message), double-click the message, and a highlighted line will appear in the code editing text field.

Please note that, for performance reasons, MDTRA does not interpret the **Lua** source code during the build process. It runs a precompiled byte code instead. The byte code is stored along with the source code within a MDTRA Project file.

8.3. Introduction to Lua Programming Language

Lua is a fast, light and portable scripting language. It combines a procedural programming paradigm with an object-oriented approach (MDTRA uses it only in a procedural manner, however). For those not familiar with **Lua**, there is a minute documentation at the official website:

http://www.lua.org/docs.html

Herein only the basics of **Lua** are described.

Lua is a case-sensitive language. For example, both "Return" and "RETURN" are valid identifiers, but "return" is a reserved word. Note that, along with keywords and built-in function names, all identifiers beginning with an underscore and followed by uppercase letters, are also reserved.

Lua is a language with dynamic typification. This means that there is no need to declare a type of a variable: it is determined after the first assignment instruction. Internal data types are: number (a double precision floating-point number), integer, boolean (true or false), string, nil, table (this is actually an associative array), and some others. MDTRA also defines some custom data types (see 8.4).

Lua is a language with implicit variable declaration. This means that there is no need to declare a variable before using it. However, every undeclared variable becomes a **global** variable within a full scope. **Local** variables are declared using a **local** keyword and are visible only at the local lexical scope (for example, within a function). Local function definitions are also legal.

Lua is a language with a built-in **garbage collector**. This means that there is no need to delete any variables allocated.

There is a associative array (also known as table) support in Lua. A table is declared as follows:

```
my_array = {};
```

To assign a value to array element, use an operator:

```
my_array[5] = 10;
```

To get a previously assigned value, write:

```
elem = my_array[5];
```

If there was no value assigned to a variable, it has a value of **nil**.

Important: array indices in MDTRA always start from 1, not 0!

Lua keywords are shown in Table 8.1. **Lua** operators are shown in Table 8.2. Some useful **Lua** build-in functions are shown in Table 8.3. Please refer to **Lua** manual on the information about how these keywords, operators and functions are used.

There are also some bound libraries with its own functions, for example, **string** library or **math** library. Such functions are called with a library name prefix: **string.reverse**, **math.sin**. The most useful functions in MDTRA Data Sources are math functions; please refer to the official website for the complete list and description:

http://www.lua.org/manual/5.1/manual.html#5.6

and	**break**	**do**	**else**	**elseif**	**end**	**false**
for	**function**	**if**	**in**	**local**	**nil**	**not**
or	**repeat**	**return**	**then**	**true**	**until**	**while**

Table 8.1. List of **Lua** keywords.

Lua Operator	C/C++ analogue	Описание
+	+	Addition
-	-	Subtraction, negation
*	*	Multiplication
/	/	Division
%	%	Reminder of division
^	^	Raise to power
==	==	Equal
~=	!=	Not equal
<	<	Less
>	>	Greater
<=	<=	Less or equal
>=	>=	Greater or equal
and	&&	Logical «AND»
or	||	Logical «OR»
not	!	Logical «NOT»
..	*none*	Concatenation of strings
#	*none*	Length of an array

Table 8.2. List of **Lua** operators.

Function Name	Description
tonumber	Converts an argument to a number (an integer or a floating-point, depending on what is appropriate).
tostring	Converts an argument to a string.
type	Returns a string describing a type of an argument (number, string, etc.).
print	Outputs a string to the Compiler Messages during a compilation process. Does nothing during a build process. See 8.8 for more details.
pairs, ipairs	Iterator function for tables. Refer to Lua manual for the details.

Table 8.3. Some useful built-in functions in **Lua.**

There are some minor modifications of **Lua** language that comes along with MDTRA:

1. **Lua** style comments ("-- text") are not supported. Instead, there is a support for **C** style ("/* text */") and **C++** style ("// text") comments.
2. Like in **C/C++**, each line may end with a semicolon. However, as opposed to **C/C++**, the semicolon is not mandatory.

And finally, a typical "Hello, World" example. It prints a message to the compiler output and returns zero as trajectory data value:

```
// A simple function that prints a message to output
// and return a value of zero.
function main()
 print('Hello, world!');
 return 0.0;
end;
```

8.4. MDTRA API: Data Types and Functions

There is a set of data types and functions that allow scripts to interact with MDTRA host program. These types and functions are not part of **Lua** language, but an necessary extension. There are also several predefined integer constants: **arg1** to **arg6**. They represent serial numbers of atoms selected in Data Source Arguments. If there were no atom selected, a corresponding arg is set to zero.

MDTRA defines two array-like types: a 2D vector (**vec2**) and a 3D vector (**vec3**). They are very much alike. The difference is that **vec2** holds two floating-point values, and **vec3** holds three floating-point values. Vector variables are created using a **constructor**, which also initializes them depending on arguments provided:

```
myvec2 = vec2();                //myvec2 now is a vector (0, 0)
myvec2 = vec2(1.0);             // myvec2 now is a vector (1, 1)
myvec2 = vec2(1.0,2.0);         // myvec2 now is a vector (1, 2)
myvec3 = vec3();                //myvec3 now is a vector (0, 0, 0)
myvec3 = vec3(1.0);             // myvec3 now is a vector (1, 1, 1)
myvec3 = vec3(1.0,2.0,4.0);     // myvec3 now is a vector (1, 2, 4)
```

MDTRA also defines a special 3x3 matrix type (**mat3**) which is actually a 9-component vector; matrix elements are stored in row-major order.

Vector variables support some operators: addition, subtraction, multiplication, division, negation, checks for equality or inequality, conversion to a string. The behavior of operators sometimes is different depending on what arguments they get. For example, you can add only two **vec2**'s or two **vec3**'s, and this is done componentwise. You may also perform a componentwise multiplication of two vectors; but along with this you may multiply vector by a number ("scale a vector"):

```
myvec3_1 = vec3(2,2,2);

myvec3_2 = vec3(3,1,4);

myvec3_mult = myvec3_1 * myvec3_2;    // result is (6,2,8)

myvec3_mult = myvec3_1 * 2.0;         // result is (4,4,4)
```

Matrix type supports addition, subtraction, multiplication, negation and converstion to a string. Multiplication operator accepts either another **mat3**, or **vec3**; the result will be of the appropriate type.

The length operator ("#") always returns 2 for **vec2**, 3 for **vec3**, and 9 for **mat3**.

There are some functions to deal with arguments of vector type: calculation of length of a vector, dot product and cross product, normalization of vector. See Table 8.4. These functions can be easily implemented directly in **Lua** user program, but it is recommended to use built-ins for performance reasons.

Function Name	Arguments	Return Value	Description
length	vec2	number	Calculates a length of a vector.
	vec3	number	
dot	vec2 (2)	number	Calculates a dot product of two vectors.
	vec3 (2)	number	
cross	vec3 (2)	vec3	Calculates a cross product of two vectors.
normalize	vec2	vec2	Returns a normalized vector. For a zero-length vector, an identity vector with y = 1.0 is returned.
	vec3	vec3	

Table 8.4. Functions dealing with custom data types.

Function Name	Arguments	Return Value	Description
numatoms[1]	-	integer	Returns a total number of atoms in current PDB file.
numresidues[1]	-	integer	Returns a total number of residues in current PDB file.
coord[1]	integer[3]	vec3	Returns a XYZ coordinate of an atom with a specified serial number, in Å.
	integer, string		Overloaded version specifies atom by residue number and atom title.
radius[1]	integer[3]	number	Returns a van der Waals of an atom with a specified serial number, in Å. Overloaded version specifies atom by residue number
	integer, string		

			and atom title.
residue[1]	integer[3]	integer	Returns a residue number of an atom with a specified serial number (indexing from 1).
atomflags[1]	integer[3]	integer	Returns flag mask of an atom with a specified serial number. Flag mask is a combination of single flags: 1 = protein; 2 = nucleic; 4 = backbone; 8 = water; 16 = hydrogen. It is better to use **checkflags** to check a single flag for existence in the mask (see below).
checkflags[1]	integer[3], integer	boolean	Returns a boolean value depending on whether a specified flag (argument 2) exists is a flag mask of an atom with a specified serial number (argument 1). Flags are listed in **atomflags** description (see above).
sqdev[1]	integer[3]	number	Returns a square deviation of atom between a reference PDB file and current PDB file aligned to it, in $Å^2$.
rmsd[1]	-	number	Returns a RMSD of backbone, in Å.
distance[1]	integer[3] (2)	number	Returns a distance between two atoms with specified serial numbers, in Å.
angle[1]	integer[3] (3)	number	Returns an angle between three atoms with specified serial numbers, in radians.
torsion[1]	integer[3] (4)	number	Returns a torsion angle between four atoms with specified serial numbers, in radians; ($-\pi \le$ angle $\le \pi$).
utorsion[1]	integer[3] (4)	number	Returns an unsigned torsion angle between four atoms with specified serial numbers, in radians; ($0 \le$ angle $\le 2\pi$).
dihedral[1]	integer[3] (4)	number	Returns a dihedral angle between four atoms with specified serial numbers, in radians; ($-\pi \le$ angle $\le \pi$).
force[1,4]	integer[3]	vec3	Returns a force vector for an atom with a specified serial number, in kcal/Å.
resultant[1,4]	integer[3] (2)	number	Returns a resultant force length between two atoms with specified serial numbers, in kcal/Å. Positive value refers to application of the resultant force towards the first atom.

sas[1]	-	number	Returns a solvent-accessible surface, in Å^2.
datapos	-	integer	Returns an index of a snapshot currently processed. This is a real snapshot index, not relative to the calling thread.
datasize	-	integer	Returns a total size of the trajectory.
dataread[2]	integer	number	Returns a calculated trajectory value for a specified snapshot index; (1 ≤ index ≤ **datasize**()).
datawrite[2]	integer, number	number	Replaces a calculated trajectory value for a specified snapshot index with a specified value; (1 ≤ index ≤ **datasize**()); returns a value written (0.0 if there was an error); refer to 8.7 for more details.
rboutput[1]	integer, number	number	Writes to residue-based output. First argument is a residue number (indexing from 1), second is value to be written.
get_residue_transform[1]	integer[3]	mat3, vec3	Returns transformation to the standard reference frame (rotation matrix and offset) for the residue specified by atom serial number. DNA residues only. This can be used in calculation of Dickerson DNA parameters.
build_rotation_matrix	vec3, number	mat3	Returns a general rotation matrix for specified axis and angle (in radians).

Table 8.5. Functions dealing with trajectory or PDB file data.

Remarks: [1] this function is not available inside **reduce**; [2] this function is not available outside **reduce**; [3] values of **arg1**-**arg6** are gladly accepted; [4] BioPASED format only.

The rest of MDTRA API functions operate on the trajectory or a PDB file data. The complete list of functions is presented in Table 8.5.

Please note that **there are no functions dealing with selection** at that moment. This feature is yet to be developed.

8.5. Multithreading: Thread Locals and Reducing

One of important MDTRA features is multithreading. The build process can be performed in a single thread (either if multithreading feature is disabled in Preferences, or there is only one processor core on a client's machine), or in multiple threads (up to 16). For each thread, there is a separate **Lua** state the program runs within. This actually means that all global variables are thread-specific, i.e. local to a specific thread (we will refer them to as "thread locals"). There is no interaction possible between threads in **main** function. Furthermore, **Lua** program doesn't know

whether it is a multi-threaded run, and how many threads are running. Information of such kind is redundant because of two primary design principles:

1. All executions of **main** function are independent and are not obliged to be consequent. The real position in the trajectory can be verified using a **datapos**() function.
2. All thread locals are collected into a global array before **reduce** function call. Size of that array is actually a number of threads used.

The name of **reduce** function now becomes clear: its main intention is to perform a reduction of thread data. One must always perform a reduction of thread locals, even if he runs MDTRA in a single-threaded mode; otherwise values calculated in single-threaded and multi-threaded modes will be different, which is not acceptable.

To give an illustration of how to perform a reduction, we will calculate an arithmetic mean of RMSD of a backbone. MDTRA already calculates an arithmetic mean, but this is just an example. We can also compare the value calculated with the reference value to determine whether our script is correct.

We use a concept of partial sums: main function calculates sum of values which is a thread local, and therefore is partial (it becomes a full sum in single-threaded mode; but this does not matter). Let p_sum be a declared thread local that will collect partial sums:

```
p_sum = 0.0;

function main()
  local value = rmsd();
  p_sum = p_sum + value;
  return value;
end;
```

We got partial sums and a trajectory filled with RMSD values. The second and the last step is reduction. We declare a local variable f_sum to collect partial sums, loop through all partial sums adding them to f_sum, and finally return a mean value by dividing full sum by the number of snapshots. Please note that in reduce function, p_sum is not a number, but an array of numbers, therefore the size operator (#) is applicable.

```
function reduce()
  local f_sum = 0.0;
  for i = 1, #p_sum, 1 do
        f_sum = f_sum + p_sum[i];
  end;
```

```
    return f_sum / datasize();

end;
```

If there are several thread locals, in reduce function they will all become arrays of the same size, so one doesn't need to create a for loop for each variable. Size of any such array can be used as an upper limit of the loop.

Please note that this code is also correct in a single-threaded mode. In that case, size of p_sum is 1, p_sum is actually a full sum, and loop is executed once, assigning a value of p_sum[1] to f_sum.

8.6. User-Defined Statistical Parameters

The **reduce** function may return up to 6 values which are referred to as user-defined statistical parameters (U1-U6). All these parameters, up to the last containing some value, will be displayed in a Statistics table. They are not obliged to be consequent: one may skip, for example, U1, by returning a nil value, and return a number into U2:

```
function reduce()

  return nil, 1.0;

end;
```

There will be a dash instead of the value for U1 in a Statistics table.

Please note that the return values will be converted to numbers, if possible. One may return a string ('1.0') instead of a number (1.0), but this is not recommended for performance reasons.

All assigned user-defined parameters can be exported to a file along with default statistical parameters.

8.7. Overriding Main Trajectory Data

Statistical parameters can also be plotted as a function of time (or, actually, a snapshot number). There is a function, **datawrite**(), available only in reduce function, that will overwrite trajectory data with a value provided. The function has some restrictions:

1. Default MDTRA statistical parameters are not overridden. They are calculated for the intact main trajectory. To calculate statistics for an overridden trajectory, use user-defined parameters U1-U6.
2. If there is a **datawrite**() call in any data source program within a Result Collector, Pearson correlation won't be calculated for such Collector.

If succeeded, **datawrite**() returns its second argument: a value written to the trajectory. If failed, trajectory is not modified, and the function returns 0.0.

8.8. Debugging

Debugging of **Lua** program should be performed during the compile process. You may use a debug function print to display any message to the compiler output. However, most of functions working with PDB files and trajectory return zero values since there is no trajectory and no PDB file bound.

Main difference between compile and build processes is that, in the first case, a source code is compiled and executed. There is a good error handling mechanism with an instructive error report. In the second case, there is a compiled byte code with low debugging capabilities. If there is a runtime error, MDTRA throws a generic warning and stops execution of a program (the rest of a trajectory will be filled with zeros). Nowadays, MDTRA's debugging features are yet to be developed.

To deal with multithreading issues, it is recommended to run a program in both single-threaded and multi-threaded modes and compare results. They must be the same. If not, possibly there is a reduction problem.

9. MDTRA Selection Syntax

9.1. Selection Quick Hint

Selection Quick Hint provides some useful examples of selection terms. To display the hint in any window that uses selection:

1. Click the **?** button (the button with a question mark) in the top right corner of the window (in the title bar). The mouse cursor changes into the arrow with the question mark.
2. Click the **Selection Expression** field. The selection hint appears.
3. Click the hint to close it.

Selection Quick Hint contents is listed in the Table 9.1.

atomno=100	select single atom
atomno>=100 and atomno<=200	select range of atoms
carbon	select all carbons
:a	select chain A
***.ca**	select all alpha-carbons
lys.cg	select all gamma-carbons in lysine residues
:a.cg	select all gamma-carbons in chain A
1-3.ca	select alpha-carbons in residues 1, 2 and 3
***.h?**	select hydrogens with 2-character name
***.h[12]**	select hydrogens named H1 and H2
***.h[1-3]**	select hydrogens named H1, H2 and H3
.h	select all hydrogens
lys	select all lysines
gl?	select all glutamates and glutamines
(lys, arg) and :b	select lysines and arginines in chain B
protein	select all protein atoms (see Table 9.2)
dna	select all nucleic atoms (may also specify **nucleic** keyword)
water and *.o	select all oxygen atoms in water molecules
within (3.5, lys)	select all atoms within 3.5 angstroms of any lysine residue
within (3.0, dna) and not dna	select non-nucleic atoms within 3.0 angstroms of nucleic acid

Table 9.1. Selection Quick Hint.

Keyword	Description
protein	All atoms that are part of a protein molecules. It is implied that everything except nucleic acid and water refers to as a protein.
dna	All atoms that are part of a nucleic acid molecules. Nucleic acids are discriminated from proteins by atom titles of the (deoxy)ribose ring that contain strokes or asterisks.
nucleic	Same as the "dna" keyword.
water	Water residues named HOH or H2O.
backbone	Backbone atoms of a protein or a nucleic acid. For a protein, they are C, CA, N, O atoms. For a nucleic acid, they are O1P, O2P, P, C5', C4', O4', C3', O3', C2', O2' (RNA only), and C1' atoms.
sidechain	All atoms except the backbone. Water molecules are implicitly treated as side chains.

Table 9.2. Keywords defining groups of atoms.

9.2. Testing Selection Terms

There is a utility tool to quickly test a selection term without invoking Data Source window, or Distance Search dialog, or 2D RMSD Tool dialog. There is a simple dialog, **Select Atoms**, with the common selection layout familiar by other MDTRA windows exploiting the selection functionality. To open it, in the **Tools** menu, select **Select Atoms…** The dialog appears (see Figure 9.1).

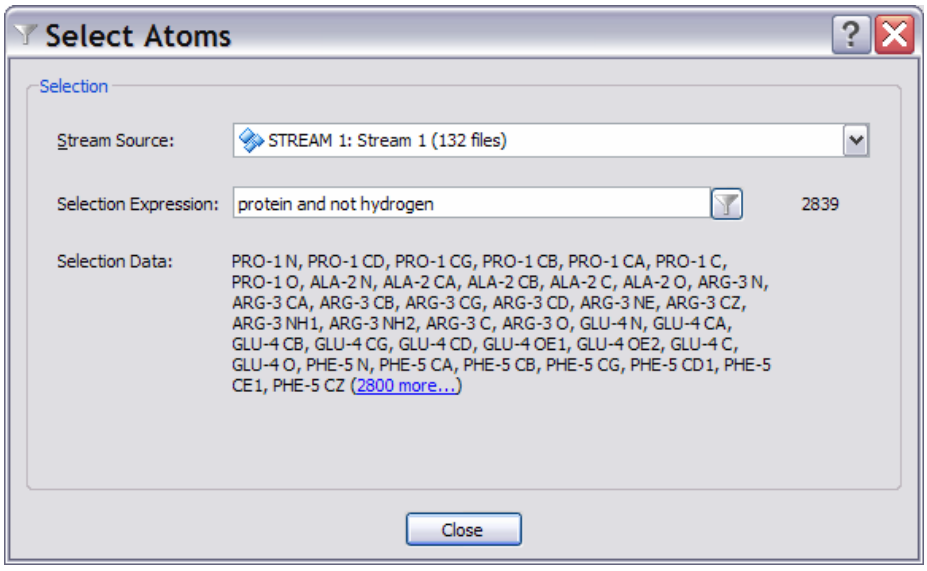


Figure 9.1. The **Select Atoms** dialog.

Like in other **Selection** sections of dialogs, a full selection can be viewed in the Selection **Data** label. If the selection is oversize, there is a hyperlink at the end of the selected atoms list. Click it to open a dialog with full selection list (see Figure 9.2).

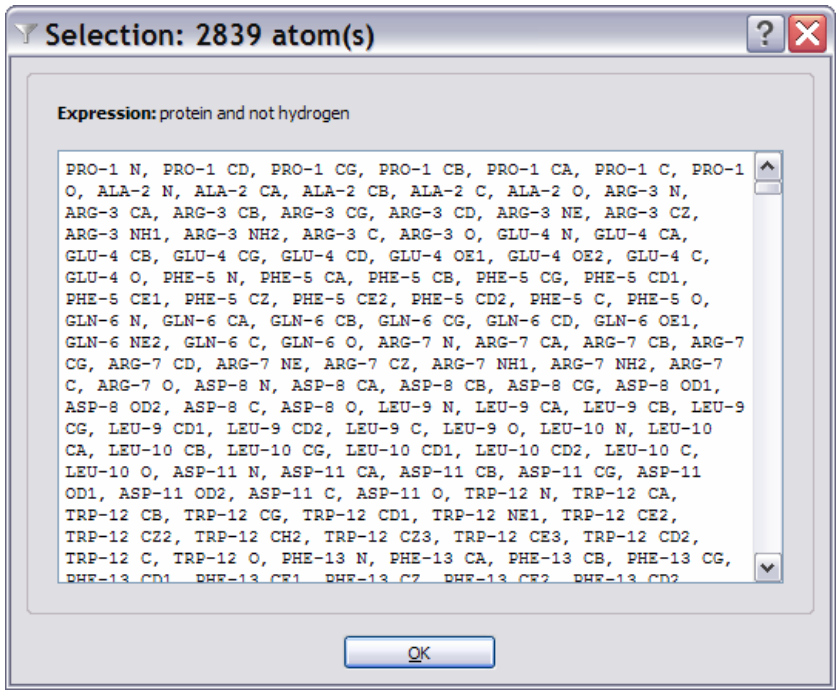


Figure 9.2. The **Selection** dialog.

This tool does not modify either trajectory files or data, and is provided only for a convenient visual inspection of selection term parsing and application result.

10. Customizing MDTRA Sessions

10.1. MDTRA Command-Line Options

The last command-line option, if given, and if it contains a dot (widely used to specify a filename extension), it treated as project file name and is loaded on startup. Thus, MDTRA project files may be associated with MDTRA in Microsoft Windows® to open with a double click.

MDTRA exploits Qt widget library to display GUI controls in platform-independent manner. Many common Qt command-line options may be used with MDTRA to customize widget look and feel. For example, "-style motif" or "-style plastique" commands (without the quotes) may be specified to the command line to change default Windows-style look and feel.

10.2. General Preferences

The list of options available in the **General Preferences**:

- **Multithreading** — enables or disables multi-core processor usage. Select "Single-threaded" to disable this feature, or select a number of threads up to 16 (it is not obligatory to use the number of threads less or equal to the number of processor cores). Select "Autodetect" to use the number of threads equal to the number of processor cores (this is recommended).
- **Use SSE instruction set if available** — enables or disables SSE enhancement instruction set usage. If SSE are not supported by the processor, the option is disabled.
- **Yield resources to other programs** – allows MDTRA to run tasks with low process priority, i.e. in background. Enable this feature if you want to use other programs while MDTRA is computing.
- **Use GPU computing if possible (NVIDIA CUDA)** — enables or disables GPGPU-optimized routines for some algorithms (for example, SAS calculation) using NVIDIA CUDA technology. If CUDA is not supported by graphics card or display driver currently installed, the option is disabled.

10.3. Plot Options

In the **Plot Options**, one may configure MDTRA optimization options, and change Plot X scale units.

To open the **Plot Options**:

2. In the **Edit** menu, select **Preferences**. The **Preferences** window appears.
3. Switch to the **Plot** tab.
4. After configuring options available, click **OK** button to save the changes and to close the **Preferences** window.

The list of options available in the **Plot Options**:

- **Trajectory Smoothing** — enables or disables smoothing of the trajectory on the plot using Central Moving Average algorithm. CMA width, in snapshots, can also be specified herein.
- **Plot multi-sample anti-aliasing** — enables or disables multisampling (FSAA) within the Plot. Video card must support multisampling in hardware to use this feature. If the feature was turned off when the program started, it requires a restart when turning on.
- **Present angles in polar coordinates** — enable rendering of a plot of angles in polar coordinates (time is plotted as distance along X and Y axes, angle value is plotted as a counter-clockwise angle between a direction to the point and positive X-axis direction).
- **Plot Language** — for publication purposes, one may change localization of the plots exported by MDTRA. Currently, only English and Russian (cp1251) languages are supported.
- **Time Scale Units** — selects X-axis units of measure for time-based plots. The snapshots, the picoseconds and the nanoseconds are available. As opposed to Y-axis scale units, changing this value does not require rebuilding the data.

10.4. Analysis Options

In the **Analysis** section of Preferences, one may configure MDTRA analysis tools. For now, only **SAS and Occlusion** calculation settings can be edited:

- **Probe radius** — specify radius of a solvent probe. Default is 1.4 angstroms which corresponds to the water molecule radius.
- **Accuracy** — balance SAS calculations between speed and accuracy. The more accurate calculation is, the more time it takes to complete. The value of "High" is recommended.
- **Exclude water molecules from calculation** — ignore water molecules that are part of source structure, for SAS calculations. This may be useful when analyzing a model with explicit solvation.

10.5. Plot Colors

Plot colors can be modified in MDTRA preferences. The following colors can be modified:

- plot background color;
- plot border color;
- plot axes color;
- plot data colors (data rows 1-14);
- plot text color;

- plot selection color;
- program syntax highlighting colors.

To modify a color:

1. In the **Edit** menu, select **Preferences**. The **Preferences** window appears.
2. Switch to the **Colors** tab.
3. On the **Colors** tab, select color title entry in the colors list.
4. Click the **…** button (the button with three dots). The **Choose Color** dialog appears. Or, double-click the color title entry to open the **Choose Color** dialog.
5. Select the desired color and click **OK** button to close the **Choose Color** dialog.
6. In the **Preferences** window, click **OK** button to save the changes and to close the **Preferences** window.

Colors are saved to the file with program settings when MDTRA program is closed.

10.6. PDB Viewer Options

In the **PDB Viewer Options**, one may configure MDTRA trajectory viewing options.

To open the **PDB Viewer Options**:

1. In the **Edit** menu, select **Preferences**. The **Preferences** window appears.
2. Switch to the **Viewer** tab.
3. After configuring options available, click **OK** button to save the changes and to close the **Preferences** window.

The following options are available:

- **No PDB Viewer** — do not use trajectory viewing functionality of MDTRA.
- **RasMol Viewer** — use **RasWin** (a RasMol-like program capable of managing multiple molecules together) to display snapshots (additionally, a path to **RasWin** executable must be specified; **RasWin** is not included within MDTRA installation and must be installed as a separate product).
- **VMD Viewer** — use **VMD** to display snapshots (additionally, a path to **VMD** executable must be specified; **VMD** is not included within MDTRA installation and must be installed as a separate product).

NOTES