GROMACS - Task #1010

Better support for multiple AnalysisData datasets

09/26/2012 06:49 AM - Teemu Murtola

Status:	In Progress	
Priority:	Normal	
Assignee:	Teemu Murtola	
Category:	analysis tools	
Target version:		
Difficulty:	uncategorized	

Description

The current framework is somewhat difficult to use to write tools and analysis data modules that process a variable number of datasets. Examples include g_rdf, where it would be nice to be able to calculate any number of RDFs in one go, and also tools like g_angle, where it would be nice to calculate angle histograms for each selection separately. In both cases, the nicest user interface would allow the user to provide any number of selections, and the tool would then produce the histograms a single file as columns. For g_angle, it could also be desirable to print all the angles from all the selections into a file; either one file, with the first columns giving the angles for the first selection, the next columns the angles for the next selection and so on. Or alternatively, angles from each selection into a separate file, although this will then generate quite a few files.

There are basically three alternatives for implementing this:

1. Add a third dimension to AbstractAnalysisData (e.g., datasetCount() in addition to columnCount() and frameCount()), and make this information and the different data sets accessible through the data structures declared in dataframe.h. Adds some complexity to the storage implementation and each AnalysisDataModule, but possibly not more than the other alternatives.
2. Represent each data set as a separate AbstractAnalysisData instance. registerDataset() functions in TrajectoryAnalysisModule would then need to be extended to either allow indexing, or the caller would need to generate names like "rdf1", "rdf2" etc. for the dataset. The name generation may be problematic, as currently it should be done as early as possible, but the number of datasets is only known after the selections have been parsed. Creating a AnalysisDataPlotModule derivative that plots from multiple input AbstractAnalysisData objects into a single output file may also turn out to be quite complicated, unless we want to sacrifice the possibility of doing this without storing all the data.
3. Encode the different datasets into a single AbstractAnalysisData object by pasting the columns. So columns 1-N would correspond to the first dataset, (N+1)-M to the second and so on. Keeps the storage implementation a bit simpler than the first alternative. But if the information on the column division is stored in the AbstractAnalysisData, this doesn't really differ much from the first alternative, except that using code will be less intuitive as it needs to do a lot of column indexing. If the information is not stored in the AbstractAnalysisData, then each module that supports this kind of data needs a separate interface to pass in this extra information, which is kind of ugly.

Related issues:

Related to GROMACS - Task #869: Make analysis data histogramming and multipoi...	**In Progress**

Associated revisions

Revision 6c31ffc4 - 06/17/2013 05:59 PM - Teemu Murtola

Uniform initialization for analysisdata tests.

Unit tests that use input data now initialize the AnalysisData object
using a common helper, which uses the properties of the input data.
This puts the initialization code into one place only, which makes it
easier to adapt to implementation of #1010.

Related to #1010.

Change-Id: I1ffe3a52a2b5edc7d6ac647cea669c250e67e71c

Revision 54919371 - 06/24/2013 07:11 AM - Teemu Murtola

More flexible input data for analysisdata tests.

Instead of somewhat inflexible parsing from a static real[] array,
construct the input data objects by explicit method calls to add frames
and point sets. Convert the existing tests to construct the input data
this way.

Increase coverage of multipoint tests for AnalysisData by using

point sets that do not cover the full range of columns.
Improve test error messages and fix issues found while doing this.

Prerequisite for unit tests for #869 and #1010.

Change-Id: Idd0831d9bbf8e59b6edfab758cd53881133e1f3a

Revision d7fb6e07 - 06/24/2013 07:15 AM - Teemu Murtola

Restructure AnalysisData unit tests.

Add test fixtures and other machinery that allows using the same test
logic for multiple different input data using typed tests from Google
Test.

Avoids code duplication in unit tests for #869 and #1010.

Change-Id: I74962aa4ed0741329b3dbcd40663fc94a3bc96b5

Revision 06a0c8a5 - 07/03/2013 07:54 PM - Teemu Murtola

Split AnalysisDataStorageFrame into two.

Main changes:
- Move parts of AnalysisDataStorageFrame that are only used internally
by the data storage into a separate class, which is internal to
datastorage.cpp. The remaining public interface now only contains
methods that are required to construct the in-progress frame/point
set.
- Make AnalysisDataStorageFrame have a separate vector of column values,
which then get transfered into the actual storage when necessary.
This part is required for handling multipoint storage.
- Only keep AnalysisDataStorageFrame objects for in-progress frames.
Without this part, the memory usage of full storage would double; now
the memory usage only increases proportional to the number of
concurrent data frames.
- Recycle the AnalysisDataStorageFrame objects for subsequent frames.
Avoids memory allocation, as the temporary column values don't need to
be reallocated for each frame.

These changes clarify the responsibilities in the code, and allow for a
much cleaner implementation of multipoint storage (subsequent commit).

The implementation may not be as clean as possible, but it will see
further changes in subsequent commits.

Refactoring in preparation for #869 and #1010, no functional change.

Change-Id: I40928ff10c5aded3b094604b37349b9fd174d267

Revision 3a18bf03 - 08/11/2013 12:54 PM - Teemu Murtola

Base support for multiple analysisdata data sets.

Make it possible for an AbstractAnalysisData object to contain multiple
data sets. See #1010 for the rationale for implementing this.

The main changes are:
- Support for specifying the number of data sets and the number of
columns for each data set in AbstractAnalysisData and classes deriving
from it. At the same time, improved checking the constraints for
analysis data modules, so that the restrictions for calling the
various set*() methods are now more uniform.
- Modeling multiple data sets in the dataframe.h interfaces: now the
point set has an additional attribute, specifying the data set.
Data with multiple data sets now always has multiple point sets.
There is exactly one per data set if it is not multipoint.
- Support for storage in AnalysisDataStorage. This is straighforward
extension of multipoint storage.
- Support for creating the data sets in AnalysisDataStorage and
AnalysisData.
- Adjust unit test framework to support at least basic testing of
multiple data sets.

Will implement support for multiple data sets in the modules, including

the angle trajectory analysis module, in a separate change to keep the amount of simultaneous changes limited.

Part of #1010.

Change-Id: Ieb373638d25ecd885d563c467bff893dd47b23b5

Revision 38f400d5 - 08/11/2013 01:56 PM - Teemu Murtola

Support for multiple analysis data sets in modules.

Main changes:
- Support for multiple data sets in the input data for most analysisdata modules. For the clearest output, standard deviation values were moved from a separate column into the error value instead, which required updating a lot of unit test reference data.
- Support for computing multiple sets of angles using 'gmx gangle'.
- Extend the analysis data test utilities to handle multiple data sets when comparing against reference data.
- Add some unit tests for the new functionality.

Supporting changes:
- Add support for providing error values through AnalysisDataTestInput.
- AbstractArrayData now provides the means to use the full functionality of AnalysisDataValue for constructing the data. AnalysisDataValue objects were already used internally for storage, so this amounts to exposing accessors for those objects instead of wrapping them in simple setters/getters for the value only.

Part of #1010.

Change-Id: I63abc05ab7e24362169e7d9ba0fa6ffc561be9d6

History

#1 - 10/24/2012 08:13 PM - Teemu Murtola

- Target version set to 5.0

I think this is an essential feature to have in the framework to make it flexible enough to write powerful tools; marking the target version as 5.0.

#2 - 04/26/2013 05:49 AM - Teemu Murtola

- Status changed from New to Accepted

#3 - 05/31/2013 12:47 PM - Teemu Murtola

- Assignee set to Teemu Murtola

#4 - 06/05/2013 06:49 PM - Teemu Murtola

- Status changed from Accepted to In Progress

#5 - 06/14/2013 08:56 PM - Teemu Murtola

https://gerrit.gromacs.org/#/c/2444/ and https://gerrit.gromacs.org/#/c/2445/ implement most of this along the first alternative outlined above. The only thing that is missing is that AbstractAnalysisData::addColumnModule() probably isn't flexible enough to support all potential uses, and it also doesn't support modules that request storage (a problem that it shares with multipoint data). Since there are currently no use cases that would require fixing these addColumnModule() problems, it is of lower priority. Will consider implementing it after looking at #869.

#6 - 06/28/2013 07:29 AM - Roland Schulz

This seems great. I haven't yet looked at the implementation in detail. In our case we have pairs. We have ~1000 selection and we want to calculate a quantity for all pairs. I assume one can use it for pairs by simply using the pair index for third dimension (which could be sel1Idx+sel2Idx*selCount). Given that we have so many pairs and the quantity is 0 for most pairs we would like to store only the non-zero values. Does it already support such sparse storage or have you thought about it?

#7 - 06/28/2013 09:19 AM - Teemu Murtola

Yes, that is the kind of case that this feature is intended for, in particular if you need to compute multiple values per frame per pair (so that you can't simply have one column per pair). That kind of sparse storage is not yet supported, but I did give it a thought as you mentioned it already earlier in #869. In the context of the new implementation, the storage part should be trivial to implement as a special case of multipoint data that simply does not allow the same column to appear more than once. Some extra effort is necessary to make everything else work nicely: there needs to be methods to enable this sort of storage and to propagate this flag to the storage, checks added so that modules that don't work with this kind of data can fail gracefully without implementing checks of their own etc.

#8 - 06/20/2014 10:45 AM - Erik Lindahl

- Target version changed from 5.0 to 5.x

#9 - 07/14/2014 11:52 AM - Teemu Murtola

- Project changed from Next-generation analysis tools to GROMACS

- Category set to analysis tools

#10 - 07/11/2016 08:23 PM - Mark Abraham

- Target version deleted (5.x)