МАТЕМАТИЧЕСКАЯ БИОЛОГИЯ И БИОИНФОРМАТИКА, 2007, том 2, №1, с.120-129,
http://www.matbio.org/downloads/Likhachev2007(2_120).pdf

==========МАТЕРИАЛЫ I МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ==========
==========«МАТЕМАТИЧЕСКАЯ БИОЛОГИЯ И БИОИНФОРМАТИКА»==========

УДК: 577.32:541.6

Анализатор траекторий молекулярной динамики

©2007 И.В. Лихачев[*], Н.К. Балабаев[**]

Институт математических проблем биологии, Российская академия наук, Пущино, Московская область, 142290, Россия

Аннотация. Представлена программа TAMD (Trajectory Analyzer of Molecular Dynamics), предназначенная для визуализации и нахождения простейших геометрических и динамических характеристик траекторных файлов молекулярных систем. Программа является удобным инструментом для первоначального анализа структуры и динамического поведения макромолекулы.

Ключевые слова: *анализатор траекторий, молекулярное моделирование, молекулярная динамика*

1. ВВЕДЕНИЕ

Результатом сложных молекулярно-динамических расчетов являются траектории движения всех атомов (≡ траектория молекулярной системы). Обычно координаты атомов записываются через определенные промежутки времени на магнитный диск в виде последовательного массива записей. Процесс расчета траектории и ее записи на диск порой длятся много часов, а иногда дней и месяцев. Как правило, такая траектория не является единственной, поскольку поведение молекулярной системы может отличаться в зависимости от условий среды (температура, давление, граничные условия), начальной структуры, изменений в химическом составе и т. п. Для работы с накопленными траекторными файлами необходимы удобные программные инструменты.

В настоящее время имеется достаточное число коммерческих и некоммерческих программ для визуализации структуры макромолекулы, записанной в формате Protein Data Bank [1]. Такие программы рассчитаны на анализ как статической структуры молекулярного объекта (RasMol, RasTop и др.), так и визуализации молекулярной динамики (VMD и др.). В то же время, имеется насущная потребность в простых и удобных инструментах для первоначального быстрого просмотра содержимого траекторных файлов (в форме молекулярного кино) и получения простых структурных и динамических характеристик моделируемой молекулярной системы. Проведение такого анализа в интерактивном режиме может служить как для первоначальной характеризации молекулярной системы, так и для планирования проводимых вычислительных экспериментов. Именно эти задачи и были поставлены при разработке программы TAMD – Анализатора траекторий молекулярной динамики.

В настоящей работе использованы траекторные файлы, получены с помощью программы ПУМА [2], ориентированной на моделирование молекулярной динамики полимерных и биополимерных систем.

[*] likhachev@impb.ru
[**] balabaev@impb.ru

2. ОСНОВНЫЕ ВОЗМОЖНОСТИ И ПРИМЕНЯЕМЫЕ АЛГОРИТМЫ

2.1. Предварительное описание

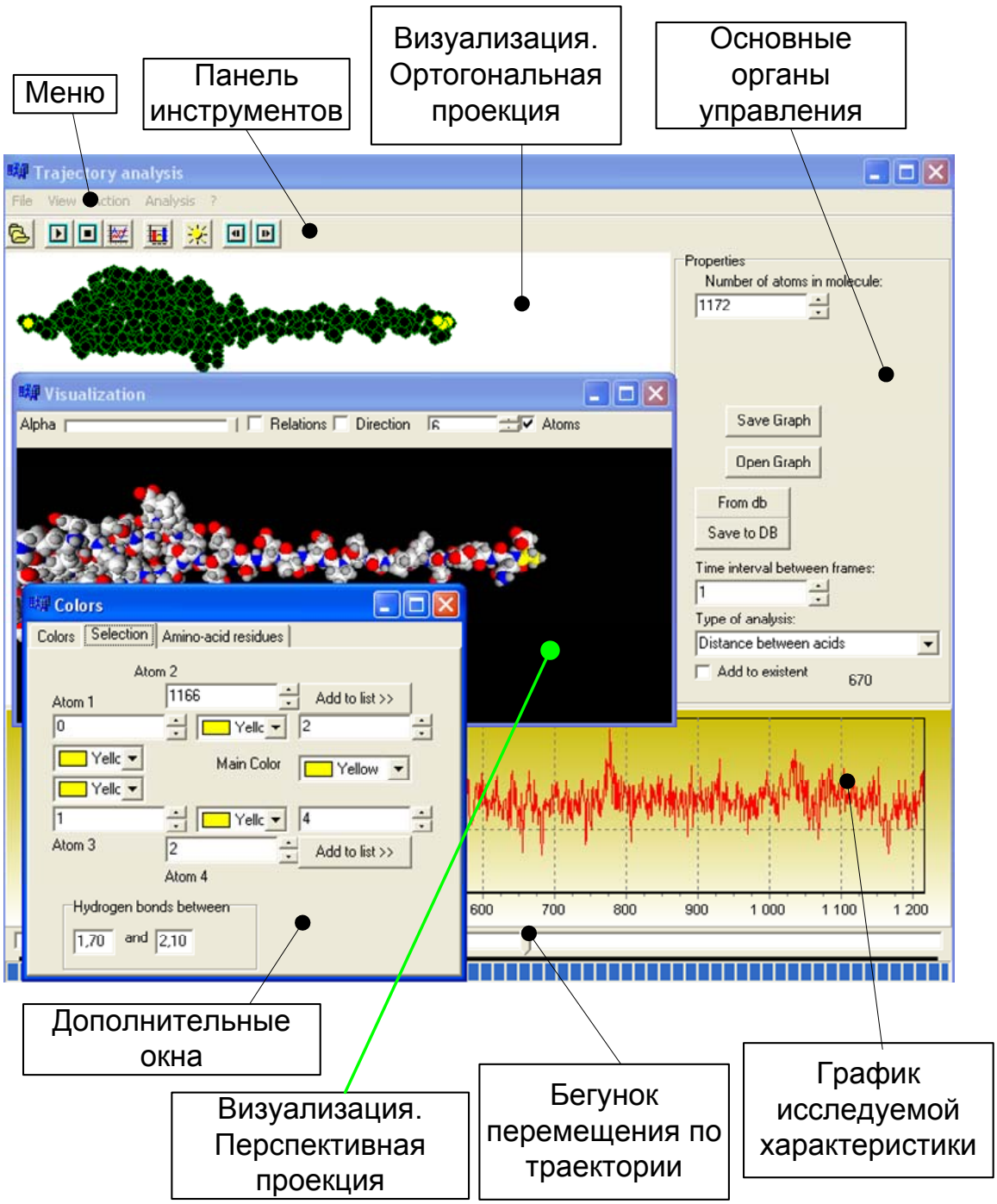

Рис. 1. Общая структура программы.

«Анализатор траектории» предназначен для анализа поведения различных геометрических характеристик молекулярной системы в процессе ее эволюции. Общее представление о работе с программой дает рис. 1. Анализатору доступны такие характеристики, как:

- углы между лучами, соединяющими данный атом с другими атомами, либо с центрами выделенных групп атомов;
- расстояния между заданными атомами, либо центрами выделенных групп атомов;

- двугранные углы между плоскостями, заданными списками атомов;
- тензор инерции макромолекулы;
- число водородных связей.

Полученная характеристика представляет собой некоторую величину, изменяющуюся во времени – сигнал. По полученному сигналу можно найти следующие его свойства (характеристики):

- плотность распределения;
- частотный спектр (Преобразование Фурье; Оконное преобразование Фурье);
- тренд (усредненные значения).

Возможности визуализации молекулярного кино:

- раскраска атомов по типам химических элементов;
- показ связей между атомами;
- возможность показа атомов в Ван-дер-ваальсовых радиусах;
- показ аминокислотных остатков;
- возможность группировки атомов по усмотрению пользователя.

2.2. Алгоритм работы с траекторией (Буферированное чтение)

Нередко расчет траектории занимает огромное количество времени, что приводит к необходимости работы с траекторными файлами большого объема. С другой стороны, при увеличении степени подробности даже малой траектории (уменьшении шага, через который координаты атомов записываются на диск) могут также существенно возрастать дисковые объемы данных.

Под большим объемом данных будем иметь в виду объем, превышающий физическую оперативную память ЭВМ. Подготовленный пользователь сразу спросит, а как же обстоит дело с виртуальной памятью? Поясним этот вопрос. Дело в том, что управлением памяти занимается операционная система. Напомним, что под операционной системой подразумевается аппаратно-программная начинка компьютера, которая предоставляет некий программный интерфейс для работы остальных программ, оборудования, а также взаимодействия с ними пользователя. Программа, в частности TAMD, дает процессору команды на чтение определенного адреса памяти, сама не зная о том, находится ли он в физической памяти, либо только на диске. Как известно, доступ к физической оперативной памяти (RAM) происходит намного быстрее, чем к дисковой. Использование виртуальной памяти приводит к появлению еще одной копии траектории на диске, что также неэффективно. С другой стороны, нельзя не согласиться, что описываемый алгоритм будет работать медленнее с короткой траекторией, чем алгоритм полной загрузки данных в оперативную память.

Перечислим основные приемы работы с дисковыми данными:

1. Полное чтение данных в память. Самое быстрое с точки зрения обработки данных. Однако данный способ полностью непригоден для большого объема данных;
2. Прямое чтение. Каждый кадр читается при необходимости его использования. Требуется минимум оперативной памяти. Однако данный способ неэффективен при повторном чтении данных, что постоянно происходит при просчете будущих координат атомов для определения направления их движения;
3. Буферированное чтение.

Остановимся подробнее на алгоритме последнего (см. Табл. 1).

Здесь номер кадра говорит только о времени доступа к нему и напрямую не зависит от кадра траектории.

Представим себе очередь, организованную по принципу FIFO (first input – first output; первым пришел – первым обработан), только усовершенствованную – кадры могут меняться местами в процессе доступа к ним.

Таблица 1. Организация очереди буфера.

Кадр 1
. . .
Рабочий кадр
. . .
Кадр n-1
Кадр n

Полный алгоритм работы буфера следующий:

- Если происходит попытка чтения кадра, которого нет в буфере, то он попадает на место рабочего кадра. При этом последний кадр затирается предпоследним;
- Если происходит обращение к кадру, который есть в памяти под номером m, то он меняется местами с кадром номер m-1. Таким образом, затирается всегда кадр, к которому не было доступа максимально длительное время;
- В буфере есть указатель на рабочий кадр – это последний кадр, к которому был доступ. Благодаря указателю, доступ к нему происходит наиболее быстро. При создании алгоритма работы буфера разумно было предположить, что наиболее часто происходит доступ к одному и тому же кадру.

Процедура смены кадров местами не приводит к полному обмену данными между ними. Просто происходит обмен адресами памяти, указывающими на соответствующие кадры.

Можно отметить, что разработанный и реализованный метод чтения данных позволяет достичь максимального быстродействия файлового ввода при ограниченном выделении оперативной памяти.

2.3. Возможности визуализации

Молекулярное кино в классическом понимании представляет собой последовательную смену кадров визуализации координат всех атомов молекулярной системы.

Программа предоставляет пользователю две проекции:

1. Ортогональную с использованием GDI (Graphics Device [Display] Interface). Интерфейс графических устройств, интерфейс GDI — набор API в Microsoft Windows, поддерживающих графический вывод растровых изображений на дисплей, графопостроители и ряд принтеров, отличается простотой и оставлен в программе для совместимости со старым оборудованием;
2. Перспективная проекция. Реализована при помощи OpenGL (Open Graphics Library [Language]), спецификация OpenGL - свободно распространяемая библиотека стандартизованных подпрограмм и функций трёхмерной графики. Программы, использующие OpenGL, отличаются особой реалистичностью. Эта библиотека использует аппаратную часть современных видеокарт, что во много раз ускоряет вывод на экран сложных графических объектов и снимает нагрузку с центрального процессора.

В перспективной проекции (рис. 2) каждый атом представляется размером и цветом соответствующего химического элемента. Цвет задается в формате RGBA. Помимо стандартных каналов: красный, зеленый, синий – в программе реализована возможность задать прозрачность для каждого типа атомов. Глубина цвета – 32 бита.

В программе предусмотрена возможность вывода химических связей между атомами. Делается это автоматически для всех пар атомов, расстояние между которыми не превышает заданное число.

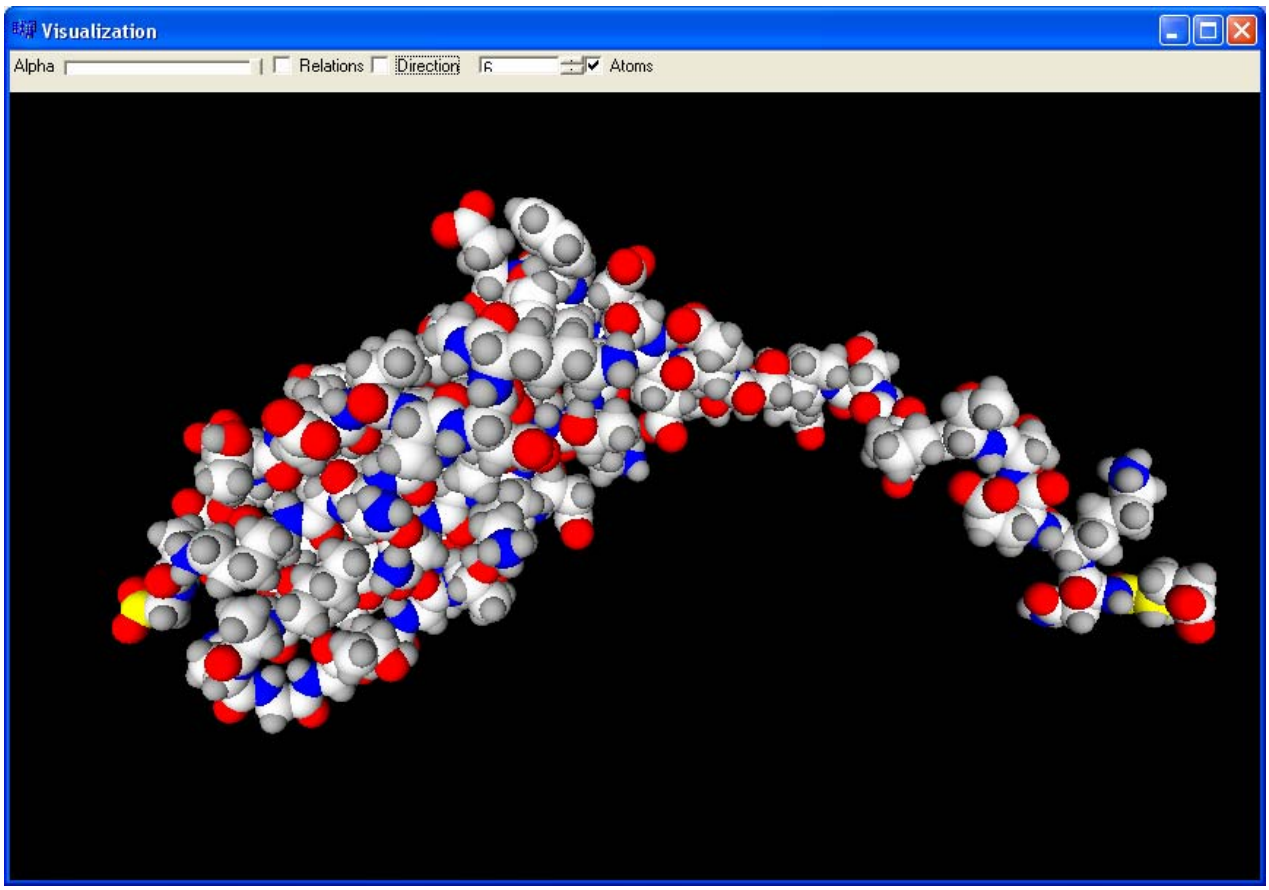

Рис. 2. Перспективная проекция.

Кроме того, показываются и направления движения атомов. Длина просчета будущих координат задается пользователем.

Однако для больших молекул такой возможности визуализации может оказаться недостаточно – химические связи могут показаться слишком мелкими на общем фоне, атомы представляют собой некое однородное вещество, в котором не видно отдельных его составляющих. Здесь мы предлагаем устанавливать пользовательские связи между атомами, а также выделять атомы в группы. Подобные структуры неудобно заводить каждый раз при запуске программы, и здесь мы предлагаем древовидную структуру хранения информации.

При анализе больших белковых структур может оказаться полезным вывод аминокислотных остатков (рис. 3). Глазу опытного специалиста поможет вывод связей между центрами остатков, насыщенность и цвет которых будет зависеть от расстояния между ними. Изменение цвета в динамике может помочь специалисту понять механизм тех или иных трансформаций.

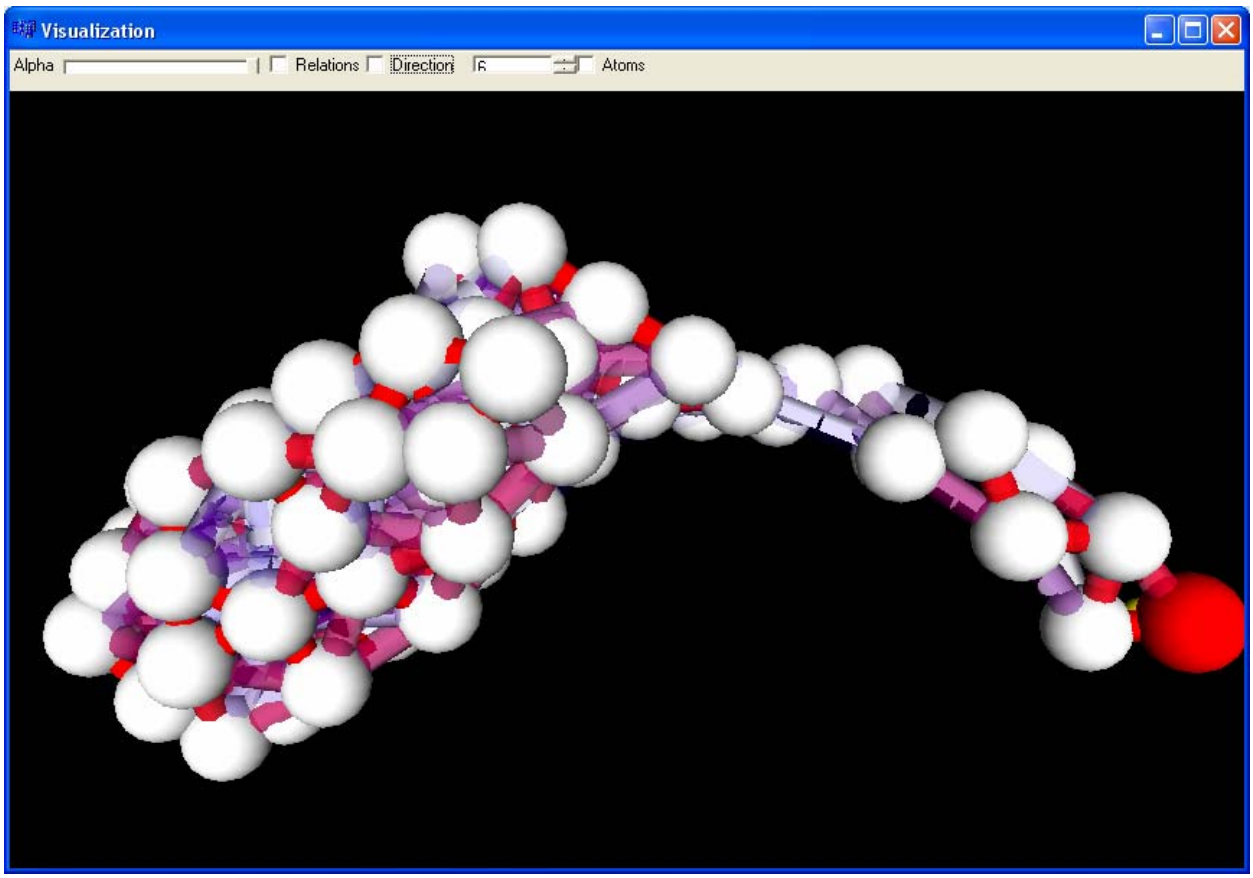

Рис. 3. Показ аминокислотных остатков со связями.

Здесь видно, как загромождённость рис. 2 легко интерпретируется новой визуализацией, показанной на рис. 3.

2.4. Использование базы данных для хранения пользовательской информации. Древовидная структура

Речь идет о многочисленных пользовательских настройках визуализации – запоминания пользовательских связей и объектов. Вбивать их заново каждый раз при запуске программы крайне неудобно. Кроме того, некоторые обработки способны генерировать большие списки пар атомов.

В настоящей программе реализован механизм хранения информации с использованием СУБД (системы управления базами данных). Разумеется, всё, что может храниться в базе данных, можно записывать в обыкновенных файлах (бинарных, либо текстовых). Но такой механизм не оправдывает себя ни как с точки зрения удобства программирования, ни использования – доступ к базе может быть осуществлен сторонними приложениями.

В программе реализован каталогизатор пользовательских настроек (рис. 4). Чтобы пользователю было максимально комфортно работать со своими данными, было принято решение построить каталогизатор в виде дерева, подобного дереву каталогов файловой системы.

Напомним читателю, что СУБД поддерживает хранение данных в виде таблиц, состоящих из некоторого числа однотипных записей. Каждая запись, в свою очередь, состоит из полей – данных определенного типа (символьных, целых, вещественных и т.д.).

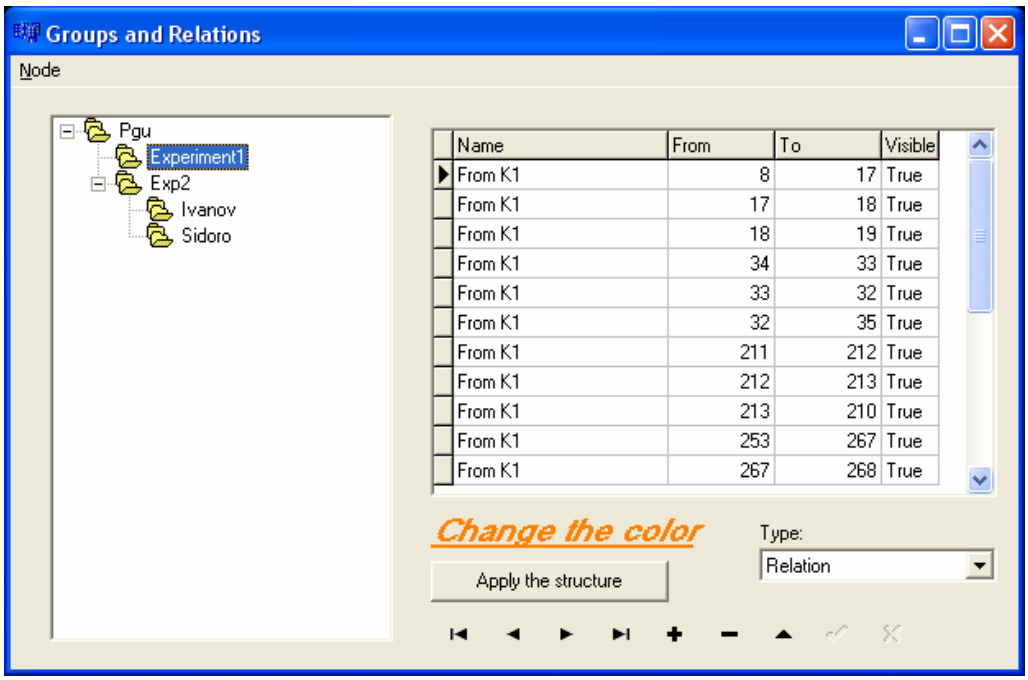

Рис. 4. Каталогизатор пользовательских настроек.

Кроме таблиц, СУБД может содержать связи между таблицами (такая система называется реляционной). Под связями понимается соответствие записи из одной таблицы (первичного ключа – primary key, PK) записям в другой таблице (внешнего ключа – foreign key, FK). Как правило, таким образом реализуют соответствие вида «один-ко-многим». Но некоторые СУБД применяют дополнительные виды отношений («один-к-одному», «многие-ко-многим»).

Большинство современных баз данных работает по технологии клиент-сервер, обсуждение которой выходит далеко за пределы этой статьи. Отметим лишь, что это придает большую надежность и защищенность данных (целостность), а также обеспечивает многопользовательскую работу. В текущей версии программы реализовано подключение к базе данных формата Microsoft Access (*.mdb). Однако программа может легко быть перенастроена на любую другую реляционную СУБД, включая MS SQL Server, а также Oracle.

2.4.1. Отношение один-ко-многим и реляционные базы данных

Требуется обеспечить хранение данных, используя реляционную СУБД. В реляционных БД, как правило, используется уникальное обозначение данных при помощи первичных ключей.

Учитывая вышесказанное можно было бы предложить следующую структуру хранения древовидной информации в БД (рис. 5):

Такая структура полностью пригодна для хранения любой древовидной информации. Дерево начинается с корня. Каждая запись (лист дерева) содержит сведения свой собственный уникальный на данном уровне вложенности идентификатор (число), ссылку на родительский элемент (у корневого элемента 0) и сами данные.

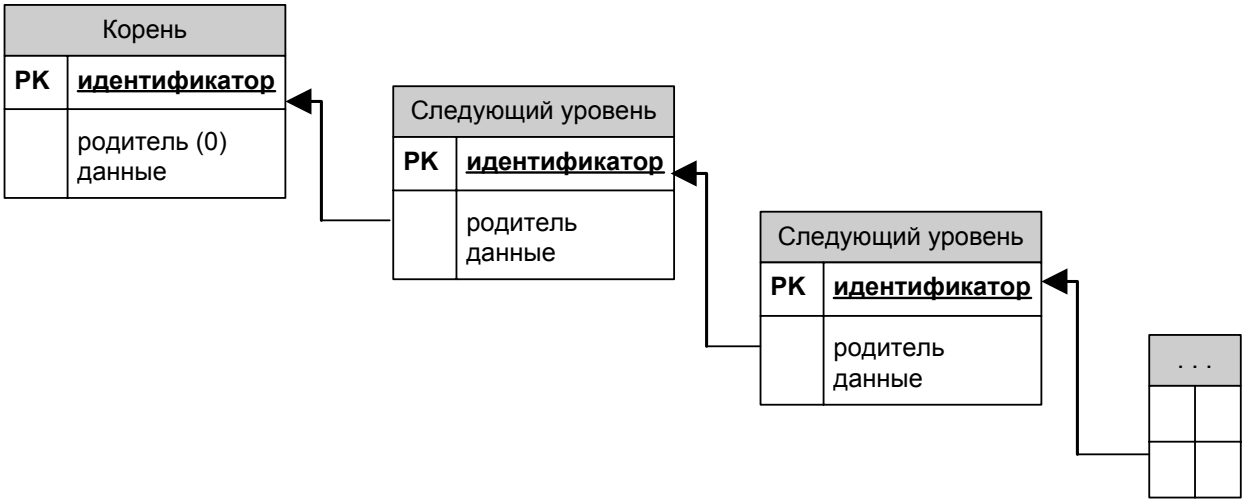

Рис. 5. Способ построения дерева, поддерживаемый СУБД.

Как видно из структуры, для каждого нового уровня требуется новая таблица. Здесь можно предложить использовать фиксированное количество уровней, либо можно создавать таблицы динамически во время работы программы при помощи структурированного языка запросов (SQL), поддерживаемого подавляющим большинством современных реляционных СУБД. Есть еще и третий выход, о нем рассказывается в следующем параграфе.

2.4.2. Разработка новой структуры хранения древовидной информации и структуры библиотечной БД

В данной работе предлагается способ хранения древовидной информации, не использующий отношение «один ко многим», поддерживаемое современными БД. Общая структура базы представлена на рис. 6.

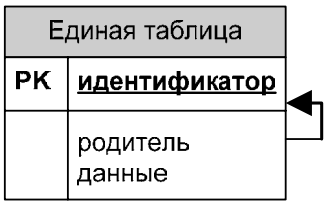

Рис. 6. Способ построения дерева, поддерживаемый программно**.**

Список полей не изменился. Каждый элемент дерева содержит свой уникальный идентификатор. Но ссылка на родительский элемент указывает на элемент, находящийся в той же самой таблице.

Нам потребуется всего одна таблица для данных любого уровня вложенности.

Разумеется, показанное отношение «один ко многим» придется реализовывать программным путем, что не является особо сложной задачей для программиста.

2.5. Методика проведения анализа

2.5.1. Характеристики

Обзор геометрических характеристик молекулярной системы, нахождение которых в настоящее время возможно в программе TAMD, уже дан в разделе 2.1. Здесь же мы остановимся на основном предназначении программы – обеспечить экспертов удобным инструментом анализа траекторий.

Все характеристики рассчитываются исходя из пространственной структуры молекулы. Они делятся на относительно простые, такие как углы и расстояния, и более сложные, например – водородные связи, анализ временных рядов и т.п.

Как известно, далеко не последнюю роль в структуре белка играют водородные связи. В программе реализовано два способа подсчета водородных связей: с фиксированной длиной связи и с учетом коридора, т.е., преодолев некоторое расстояние, связь считается образовавшейся, а, покинув другой, более удаленный барьер, считается разорвавшейся.

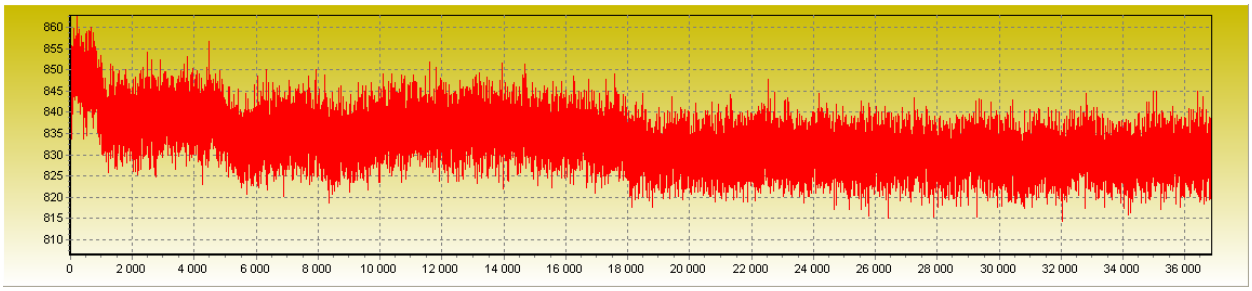

Рис. 7. График изменения количества водородных связей.

Такую характеристику удобно применять при слежении за трансформацией молекулы. Как правило, более компактному состоянию соответствует большее количество водородных связей. На рис. 7 в качестве примера приведен график изменения числа водородных связей в процессе разворачивания белковой глобулы при ее растяжении за концы.

Анализ изменения расстояний между геометрическими центрами аминокислотных остатков также может оказаться полезным при изучении динамики структуры.

2.5.2. Статистический и частотный анализ

После получения «траектории» исследуемой величины, становится доступным возможность ее анализа. Если мы говорим о временном ряде некоторых чисел, то разумно посмотреть их распределение. Пример такого распределения приведен на рис. 8.

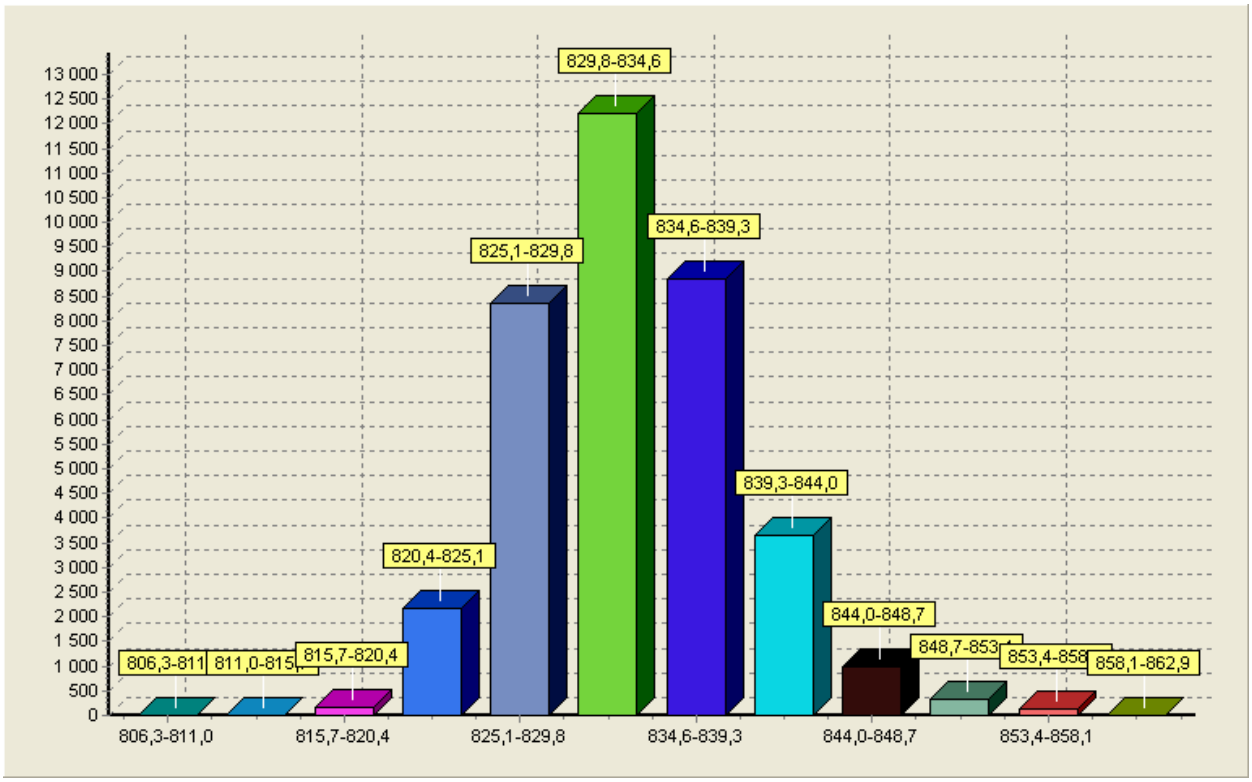

Рис. 8. Распределение полученной величины.

Стоит напомнить, что мы имеем дело с величиной, зависящей от времени. А раз так, то можно проводить частотный анализ. Однако бывают случаи, когда исследуемая характеристика ведет себя по-разному на разных участках траектории. Преобразование Фурье всей последовательности значений может не дать нам четкого результата. И тогда на помощь приходит оконное преобразование Фурье (рис. 9).

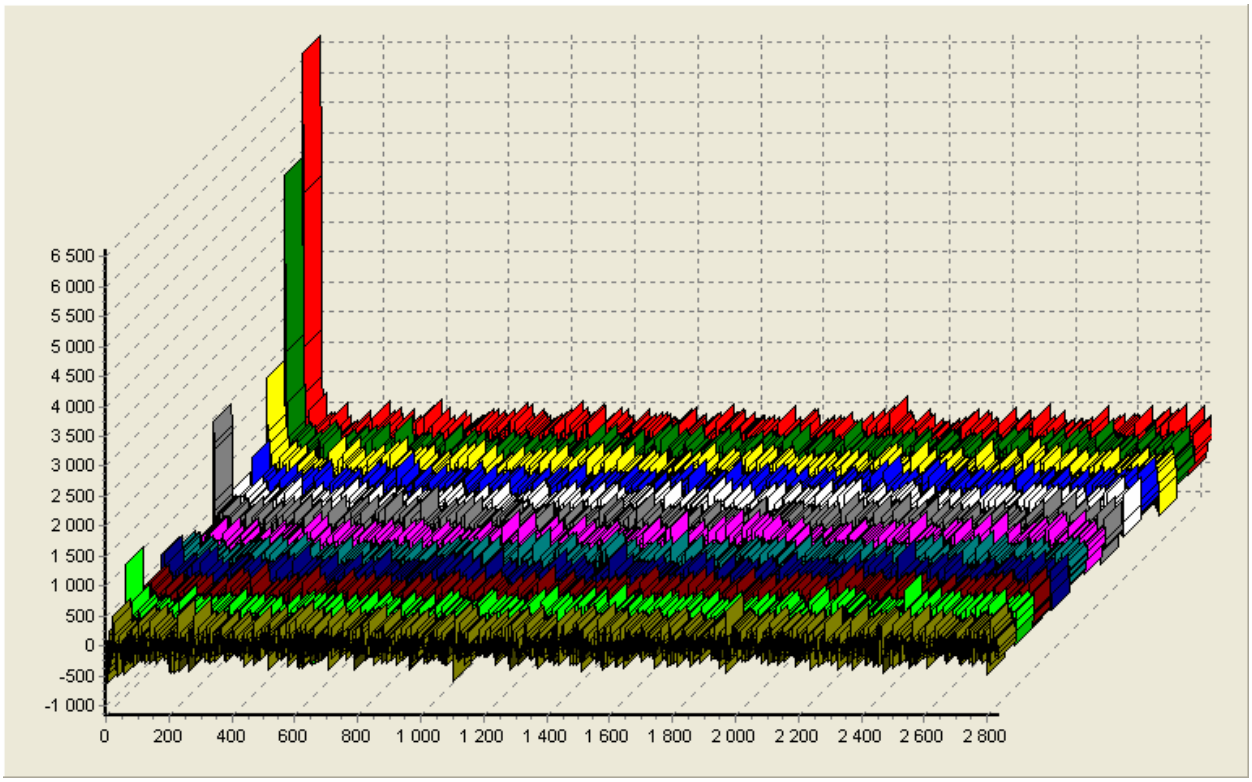

Рис. 9. Оконное преобразование Фурье.

3. ЗАКЛЮЧЕНИЕ

В настоящей статье представлена программа «Анализатор траекторий молекулярной динамики», прямым назначением которой является помощь экспертам в предметной области, обеспечивая их удобным, простым, но в то же время достаточно мощным и быстрым инструментом, способным обрабатывать большие массивы информации в виде траекторий молекулярной динамики.

Данный программный продукт следует также рассматривать как платформу с широким спектром базовой функциональности для быстрого построения (программистами) новых видов обработок траекторий молекулярной динамики.

Работа выполнена при финансовой поддержке РФФИ, гранты № 06-03-32814 и № 06-04-08136.

СПИСОК ЛИТЕРАТУРЫ

1. Bernstein F. C., Koetzle T. F., Williams G. J. B., Meyer Jr. E. F., Brice M. D., Rodgers J. R., Kennard O., Shimanouchi T., *et al*. Protein Data Bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.* 1977. **112**. 535-542.
2. Программный комплекс ПУМА для моделирования молекулярной динамики полимеров и биополимеров. – ИМПБ РАН, техническая документация.

Материал поступил в редакцию 30.03.2007, опубликован 04.05.2007