Научная визуализация, 2018, том 10, номер 1, страницы 110 - 127, DOI: 10.26583/sv.10.1.09

Библиотека aiwlib — инструмент для создания приложений численного моделирования, визуализации и анализа результатов

А. В. Иванов[1], С. А. Хилков[2]

[1] Институт прикладной математики им. М.В.Келдыша РАН
ORCID: 0000-0001-5132-3748, aivanov@keldysh.ru
[2] НОЦ Института прикладной математики им. М.В.Келдыша РАН, МФТИ (ГУ)
ORCID: 0000-0003-2702-5630, khilkov.s@gmail.com

Аннотация

Библиотека **aiwlib** предназначена для разработки высокопроизводительных приложений численного моделирования на языках **C++11** и **Python** для **OS Linux**, проведения массовых расчетов и последующего многопараметрического поиска, визуализации и анализа результатов.

Поддерживается визуализация данных заданных на равномерных прямоугольных сетках высокой размерности в виде двумерных срезов с эффективным изменением положения и ориентации среза; данных заданных на сферических сетках; произвольных проверхностей в трехмерном пространстве заданных на треугольных неструктурированных сетках; воксельная визуалиция данных заданных на равномерных трехмерных сетках; визуализация распределения намагниченности при микромагнитном или атомистическом моделировании.

Библиотека активно использовалась при разработке программных комплексов для нужд сейсморазведки, физики плазмы и оптики мутных сред, решении фундаментальных и прикладных задач по изучению магнитных материалов и созданию устройств спинтроники, моделировании процессов разработки керогеносодержащих нефтяных месторождений с учетом внутрипластового горения, моделировании задач пороупругости и гидроразрыва пласта.

Работа выполнена при финансовой поддержке Российского Научного Фонда (проект №15-11-00021).

ключевые слова: численное моделирование, визуализация, высокопроизводительные приложения

1 Введение

Существует по крайней мере три независимых способа ускорения процесса создания приложений численного моделирования: выбор правильной архитектуры, использование высокоуровневых библиотек и метапрограммирование (кодогенерация), в том числе с применением различных систем компьютерной алгербы.

Последние десятилетия показали, что скорость разработки на интерпретируемых мультипарадигменных языках программирования с динамической (т.н. “утиной”) типизацией (**Python**, **Ruby** и т.д.) на порядок выше, чем на традиционных компилируемых языках со статической типизацией (**C/C++**, **Pascal**, **Fortran**). Обратной стороной медали является низкая производительность получаемых приложений — по этому показателю **Python** поигрывает **C++** на один–три порядка в зависимости от задачи.

В приложениях численного моделирования как правило можно выделить вычислительное ядро и интерфейс. Вычислительное ядро пишется очень тщательно, модифицируется сравнительно редко и должно иметь максимально возможную прозводительность. Интерфейс, напротив, постоянно модифицируется

под различные постановки, но его производительность значения не имеет, поскольку основные затраты вычислительных ресурсов связаны с работой ядра.

Оптимальной архитектурой для приложения численного моделирования является комбинация интерфейса на интерпретируемым языке с динамической типизацией (мы выбрали **Python** [1, 2]), и вычислительного ядра на традиционном компилируемом языке со статической типизацией (в библиотеке **aiwlib** используется **C++**11). Для связывания интерфейса и вычислительного ядра используется утилита **SWIG** [3], ядро собирается в виде разделяемой библиотеки и импортируется в **Python** как пользовательский модуль. Такой подход, в частности, позволяет эффективно решать проблемы с заданием параметров расчета — в простейшем случае параметры задаются непосредственно в управляющем файле, написанном на языке **Python**, который редактируется по мере необходимости. Библиотека **aiwlib** предоставляет систему сборки, основанную на **GNU Make** [4], обеспечивающую компиляцию таких проектов на основе короткого (три–четыре строки) пользовательского **Makefile**.

Существуют пакеты, основанные на аналогичной архитектуре. Широко известная система **Matlab** использует свой интерпретируемый язык, ориентированный на реализацию сложных алгоритмов в терминах линейной алгебры, проведение расчетов и визуализацию результатов, при этом отдельные функции могут быть реализованы на языке **C** и подключены в виде разделяемой библиотеки. Библиотека numpy включает ряд алгоритмов прикладной математики на языке **C** и развитые средства визуализации, которые вызываются из **Python**.

Однако, система **Matlab** является коммерческой (с довольно дорогой лицензией), кроме того, написание расширений на **C** для **Matlab** — не вполне тривиальная задача. Библиотека numpy получила широкое распространение (и фактически является полноценной заменой системы Matlab), но накладные расходы при реализации сложных алгоритмов только на **Python**, даже несмотря на вызовы высокоуровневых функций на **C**, оказываются слишком высоки.

В отличии от аналогов, библиотека **aiwlib** инстанцирует в **Python** основные классы и функции ядра, написанного на **C++11**, что дает возможность использовать по мере необходимости практически одинаковый код в обоих языках, выбирая оптимальную с точки зрения соотношения “производительность кода/скорость разработки” декомпозицию разрабатываемого приложения.

Существует устойчивый миф о том, что приложения численного моделирования, написанные на языке **C++**, уступают по производительности приложениям, написанным на языке **Fortran**. В реальности на языке **C++** значительно проще, чем на **Fortran**, писать неэффективные приложения, но при соблюдении ряда простых правил [5] разница в производительности практически отсутствует. Напротив, множество средств, предоставляемых разработчику современным языком **C++**, значительно ускоряют эффективную реализацию сложных вычислительных алгоритмов. В частности, **LRnLA** (локально–рекурсивные нелокально–асинхронные) алгоритмы, обеспечивающие экстремально высокую производительность в задачах численного моделирования [6, 7, 8, 9], были реализованы именно на связке языков **C++** и **Python** с широким использованием механизма шаблонов **C++**.

Выбор стандарта **C++**11 в **aiwlib** связан с тем, что, с одной стороны, этот стандарт дает ряд новых возможностей (например шаблоны с переменным числом аргументов), с другой стороны, является уже хорошо устоявшимся и поддерживается относительно старыми

компиляторами на большинстве актуальных кластеров и суперкомпьютеров.

Второй способ ускорения создания приложений численного моделирования — использование высокоуровневых библиотек. В настоящий момент существует огромное количество библиотек, реализующих как достаточно сложные структуры данных (контейнеры, например, списки или разнообразные деревья), так и вычислительные алгоритмы (решение СЛАУ, быстрое преобразование Фурье и т.д.). Из библиотек на **C++** можно упомянуть такие библиотеки как **boost**, **Eigen**, **gmm++**, **MTL4**. Ядро библиотеки **aiwlib** ближе всего к библиотеке **blitz++.** Библиотека **aiwlib** дополняет традиционный функционал аналогичных библиотек развитыми средствами отладки, элементами линейной алгебры с рядом специфических операций, различными контейнерами (в том числе массивом произвольной размерности на основе Z–кривой Мортона и уникальной сферической сеткой на основе разбиения додекаэдра), средствами проведения массовых расчетов и развитыми средствами визуализации. К средствам визуализации относятся как оболочка для стандартного графопостроителя **gnuplot**, позволяющая с минимальными усилиями строить графики типографского качества, так и ряд специфических утилит для визуализации данных на сферической сетке, визуализации поверхностей, распределения магнитных моментов и т.д. В отличие от стандартных средств визуализации (например **paraview**), вьюверы библиотеки **aiwlib** имеют относительно бедный оконный интерфейс (что отчасти компенсируется развитым интерфейсом командной строки) и ориентированы на обработку больших массивов данных.

Третий способ ускорения создания приложений численного моделирования — использование метапрограммирования (автоматической кодогенерации) и различных систем компьютерной алгебры [10, 11]. В библиотеке **aiwlib** присутствуют некоторые средства для реализации этого подхода (конвертация алгебраических выражений из языка **Python** в форматы gnuplot, **C++** и **LaTeX** применяется в частности в утилите **gplt**), но эта достаточно сложная тема выходит за рамки настоящей статьи.

Предыдущая версия библиотеки **aivlib** (через букву **V**, [12]) успешно развивалась около десяти лет, пока не стало понятно, что устранение накопившегося списка недостатков требует слома обратной совместимости. Данная статья посвящена второй версии библиотеки — **aiwlib** (через букву **W**, [13]). Версии несовместимы и могут совместно использоваться в одном проекте (с точки зрения компилятора это разные библиотеки).

2 Ядро библиотеки

2.1 Средства отладки

Библиотека **aiwlib** предоставляет развитые средства отладки, учитывающие специфику приложений численного моделирования. В заголовочном файле **<aiwlib/debug>** на языке **C++** определена функция **init_segfault_hook()**, макросы отладочной печати **WOUT** и **WERR**, макрос **WCHK**, проверяющий значения выражений на **nan** и **inf**, макросы **WEXT** и **WEXC**, позволяющие выводить информацию со стека при возбуждении исключения или ошибке сегментирования, и макросы **WASSERT** и **WRAISE** для возбуждения исключительной ситуации.

Все макросы выводят сообщения на стандартный поток вывода или стандартный поток ошибок и указывают при этом исходный файл, номер строки и функцию. Перечисленные макросы (кроме макроса **WRAISE**) работают только если определен макрос **EBUG**, например при помощи опции компиляции **-DEBUG** или при помощи аргумента **debug=on** команды **make**, в противном случае макросы игнорируются компилятором.

Перечисленные макросы могут принимать в качестве аргументов произвольное количество выражений, например фрагмент кода

```
int a = 1; double b = 2.5; WOUT(a, a+b, a*b+3);
```

после компиляции и запуска выдаст в стандартный поток вывода

```
#test.cpp main() 14: a=1, a+b=3.5, a*b+3=5.5
```

Макрос **WERR** выводит информация в стандартный поток ошибок. Допускается использование любых выражений, результат которых может быть выведен в поток **std::ostream** при помощи оператора **<<**.

Макрос **WCHK** проверяет значения своих аргументов (результаты вычисления выражений с плавающей точкой) на значения **inf** и **nan**. Если для хотя бы одного аргумента проверка не пройдена, выводится сообщение об ошибке и возбуждается исключение.

Макросы **WEXC** и **WEXT** формируют на стеке объекты специального вида на основе шаблонов **std::tuple**, содержащие копии аргументов, что практически не влияет на производительность. При разрушении кадра стека (например нормальном выходе из функции) объекты с копиями аргументов макроса уничтожаются без побочных последствий. В случае возбуждения не перехваченного исключения либо ошибки сегментирования накопленная информация выводится в стандартный поток ошибок. Макрос **WEXC** допускает работу в многопоточном режиме, но не обрабатывает ошибки сегментирования. Макрос **WEXT** действует аналогично, но при этом регистрирует объекты в глобальной таблице, что не допускает его использование в многопоточном режиме. При возникновении ошибки сегментирования информация со всех зарегистрированных в глобальной таблице объектов выводится на стандартный поток ошибок. Для включения обработки ошибок сегментирования необходимо вызвать функцию

```
init_segfault_hook();
```

Такой подход зачастую оказывается удобнее, чем анализ дампа памяти при помощи отладчика. Во первых, отладчик не может показать историю изменений какой либо переменной, в отличии от макросов **WEXC** и **WEXT** (для этого придется вызвать несколько макросов). Во вторых, при запуске на суперкомпьютере объем дампов памяти может быть слишком велик для анализа.

При обработке ошибки сегментирования дополнительно выводится стек вызовов, который потом может быть проанализирован стандартной утилитой **addr2line**.

Макрос **WASSERT** принимает первым аргументом условие, если оно нарушается, возбуждается исключение.

2.2 Потоки ввода–вывода

Библиотека **aiwlib** предоставляет потоки ввода–вывода (абстрактный базовый класс **aiw::IOstream** и его наследники **aiw::File** и **aiw::GzFile**), основанные на стандартных потоках **FILE** и библиотеке **zlib**. Потоки отличаются от стандартных потоков **std::iostream** большей производительностью (за счет отказа от лишней ступени буферизации), наличием типобезопасного аналога метода **printf**, возможностью мапирования фрагментов файла с автоматической сборкой мусора и операциями **<** и **>**, перегруженными как операции бинарного ввода/вывода для всех актуальных типов данных и контейнеров, включая типы библиотеки **STL**.

2.3 Элементы линейной алгебры

Библиотека **aiwlib** определяет тип вектора **aiw::Vec<D, T=double>**, параметризованный по размерности (длине) **D** и типу ячейки **T** (по умолчанию **double**). Для этого типа перегружены как традиционные операции сложения, вычитания, скалярного умножения, так и операции сравнения (для эффективного

определения попадания точки в **D**-мерный параллелепипед), покомпонентного умножения, взятия модуля, вычисления максимума и т.д. Вектор с типом ячейки **int** объявлен как **aiw::Ind<D>** и используется в качестве индекса при доступе к ячейкам многомерных сеток.

Реализована операция обхода многомерной области (D вложенных циклов):

```
aiw::Ind<D> N = ...; // размер области
for(Ind<D> i; i^=N; ++i){ ... }
```

Как правило, инстанцирование шаблонов **C++** в **Python** при помощи **SWIG** требует указания специальной инструкции для **SWIG** под каждый вариант инстанцирования. Инстанцирование сопровождается автоматической генерацией и компиляцией большого объема **C++** кода, при этом возможности **SWIG** по обработке последних расширений стандарта **C++11** весьма ограничены, однако другие варианты связывания **C++** и **Python** значительно сложнее для пользователя. Поскольку внутреннее представление данных в объектах типа **Vec<D,T>** тривиально, в библиотеке **aiwlib** реализован специальный механизм инстанцирования таких объектов, основанный на взаимодействии с системой контроля типов системы **SWIG**. В итоге для векторов не требуется инстанцирование: в **Python** достаточно проимпортировать модуль **aiwlib.vec**, чтобы получить прозрачный доступ к объектам **Vec<D,T>** (для всех актуальных типов **T**) в **C++** коде со всеми их возможностями.

Для инстанцирования в **Python** описанных далее контейнеров (многомерных и сферических сеток) для каждого типа и размерности необходимо собирать отдельный модуль при помощи утилиты **make**, например

```
make MeshF3-float-3
```

приводит к сборке модуля **aiwlib.MeshF3**, содержащего трехмерную сетку с ячейками типа **float**.

2.4 Многомерные сетки (массивы)

При реализации многомерных массивов наиболее эффективным подходом является создание в памяти традиционного одномерного массива (вектора) и эмуляция многомерности на основе адресной арифметики. Для этого необходимо определить взаимно однозначное соответствие между индексом ячейки многомерной сетки $I=(I_x,I_y...)$ и смещением в одномерном массиве f.

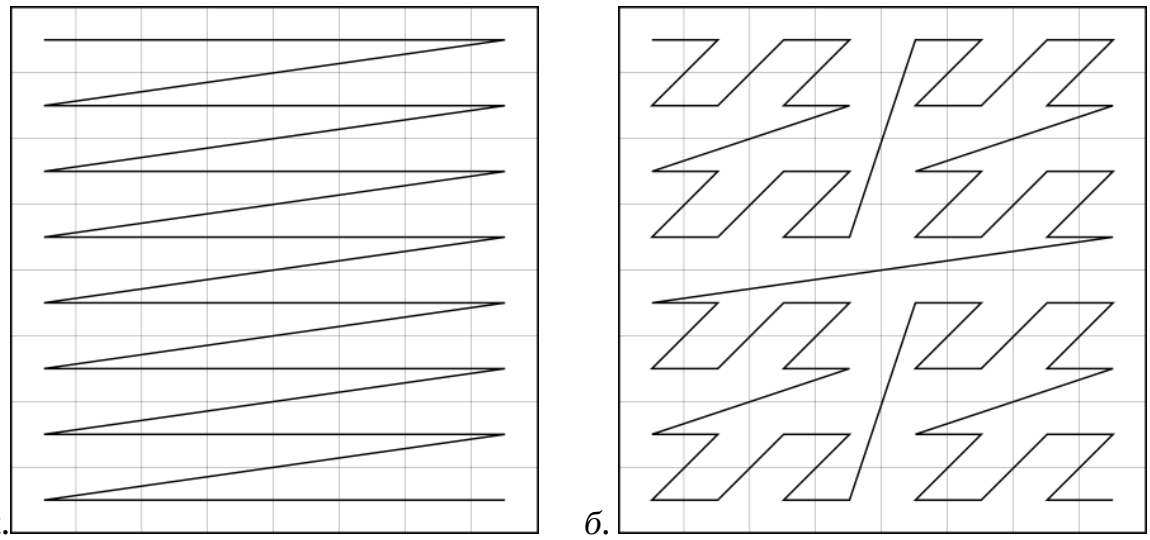
Очевидно, существует много различных решений этой задачи, однако традиционным является вариант с последовательным упорядочением ячеек

$$f = I_x + N_x I_y + N_x N_y I_z..., \qquad (1)$$

при этом ось x является самой “быстрой”, данные в памяти локализованы по оси x. Возможны вариации с последовательностью осей — например, в традиционных многомерных массивах языка **C** вида **T arr[Nx][Ny]...** ось x напротив является самой “медленной”. В самом общем случае традиционное решение может быть записано как

$$f = p_0 + \vec{I}\cdot\vec{S}, \qquad (2)$$

где p_0 — смещение нулевой ячейки, $\vec{S}$ — вектор смещений ячейки с индексом $(1,1,1...)$.



а. *б.*

Рис. 1. Традиционный обход многомерного массива (*а*) и обход на основе Z – кривой Мортона (*б*)

Если многомерный массив представляет собой D–мерный гиперкуб со стороной 2^R ячеек, традиционный подход сводится к построению смещения f в виде

$$f = I_x \cdot \mathbf{2^0} + I_y \cdot \mathbf{2}^R + I_z \cdot \mathbf{2}^{2R} + \ldots = \sum_{k=\mathbf{0}}^{D-1} I_k \cdot \mathbf{2}^{kD} = \overbrace{\ldots \underbrace{i_z^{R-1}\ldots i_z^0}_{R\ бит} \underbrace{i_y^{R-1}\ldots i_y^0}_{R\ бит} \underbrace{i_x^{R-1}\ldots i_x^0}_{R\ бит}}^{D\ групп\ бит}, \quad (3)$$

где k – номер оси координат (от 0 до $D-1$), i_k^l — l-й бит I_k. Можно предложить альтернативный вариант построения f

$$f = \overbrace{\underbrace{i_{D-1}^{R-1}\ldots i_y^{R-1} i_x^{R-1}}_{D\ бит} \ldots \underbrace{i_{D-1}^{1}\ldots i_y^{1} i_x^{1}}_{D\ бит} \underbrace{i_{D-1}^{0}\ldots i_y^{0} i_x^{0}}_{D\ бит}}^{R\ групп\ бит}, \quad (4)$$

приводящий в итоге к фрактальной Z –кривой Мортона (Лебега) [14], рис. 1.

В отличие от традиционного обхода, обход на основе Z –кривой существенно упрощает построение различных адаптивно–рекурсивных сеток и обеспечивает высокую локальность данных (ближайшие соседи в конфигурационном пространстве, как правило, расположены близко в памяти), что может увеличивать эффективность расчетов в так называемых memory–bound задачах [7, 8]. К недостаткам обхода на основе Z – кривой можно отнести фиксированный набор размеров массива и невысокую эффективность произвольного доступа: расчет смещения f на основе многомерного индекса ячейки $\vec{I}$ — относительно дорогая с вычислительной точки зрения операция.

Библиотека **aiwlib** предоставляет два класса **Mesh<T,D>** и **ZCube<T,D>**, параметризованных по типу ячейки **T** и размерности массива **D**. Класс **Mesh** реализует традиционный обход, класс **ZCube** реализует обход на основе Z – кривой. Оба класса реализуют произвольный доступ по многомерному индексу ячейки (объекту типа **aiw::Vec<D, int>**); эффективный обход массива и эффективный доступ к ближайшим соседям ячейки, в том числе с учетом периодических граничных условий; настройку равномерных сеток (пределов и шага) и логарифмического масштаба по отдельным осям, позволяющую вычислять координаты центров и углов ячеек в конфигурационном пространстве; кусочно–постоянную, линейную, локальную кубическую и B – сплайновую интерполяцию (может настраиваться независимо для отдельных осей); транспонирование (смену порядка следования) и разворот осей; эффективное сохранение данных на диск и чтение данных с диска в

бинарном и текстовом форматах для последующего анализа и визуализации.

Кроме того, класс **Mesh** дополнительно позволяет вырезать прямоугольные подобласти (сетку такого же типа, но меньшего размера) и строить срезы (сетки меньшей размерности). Все преобразования сеток (транспонирование, разворот осей, вырезание подобласти и построение среза) возвращают новый объект сетки, предоставляющий альтернативный доступ к той же области данных, выделения памяти и копирования больших объемов данных при этом не производится. Кроме увеличения производительности это открывает дополнительные возможности для обработки данных — например, двумерный срез трехмерной сетки может быть заполнен новыми данными, что приведет к изменению исходной трехмерной сетки.

Кроме того, при необходимости (подключив соответствующий заголовочный файл meshop) можно перегрузить операции “унарный минус”, бинарные +, -, *, /, ^ (как степень) и функции **abs, acos, asin, atan, ceil, cos, exp, fabs, floor, log, log10, sin, sinh, sqrt, tan, tanh, atan2, fmod, pow** над экземплярами классов **Mesh<T,D>.**

Допускаются выражения произвольной сложности, содержащие перегруженные операции и функции, операндами которых являются экземпляры классов **Mesh<T,D>** или любые другие данные, для которых выражение будет иметь смысл, если вместо экземпляров классов **Mesh<T,D>** подставляется значения из одной ячейки (типа **T**).

Само по себе выражение не приводит к выполнению каких либо действий, пока не будет выполнена операция **<<==** левым операндом которого должен быть экземпляр класса **Mesh<T,D>**, а правым построенное выражение. При этом запускается цикл по ячейкам сетки, стоящей слева от операции **<<==**, для каждой ячейки сетки слева отдельно вычисляется и записывается результат выражения справа. Реализация близка к сеточно–операторному подходу, развитому например в [15].

2.5 Сферические сетки

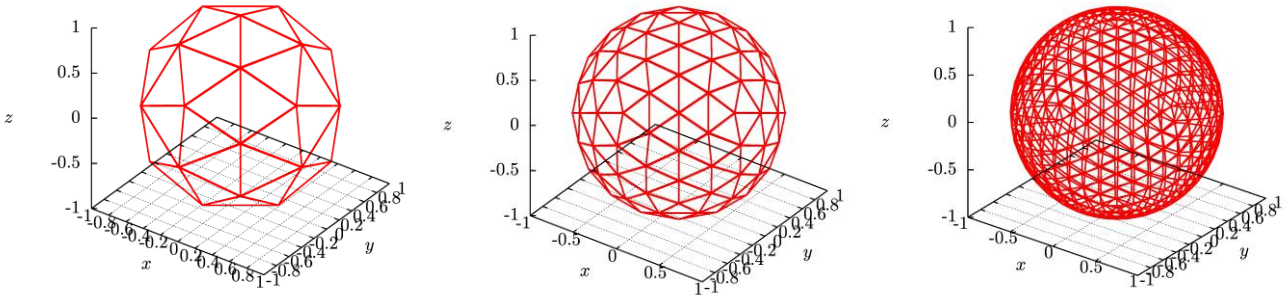


Рис. 2. Сферическая сетка на основе рекурсивного разбиения додекаэдра с различными рангами

Библиотека **aiwlib** предоставляет сетку на сфере, составленную из почти правильных треугольников, полученную путем рекурсивного разбиения додекаэдра [16, 17], рис. 2. В отличие от традиционных сферических координат с двумя сильными особенностями на полюсах, сферическая сетка на основе додекаэдра имеет 12 слабых особенностей, отвечающих центрам граней додекаэда — в этих точках узлам сетки инциндентны пять ячеек, а не

шесть (как для остальных узлов), и эти ячейки сильнее всего искажены (один из углов равен 72° а не 60°).

Сетка содержит 60×4^R ячеек и $30\times4^R+2$ узлов, где $R\geq0$ — ранг разбиения. Эффективно реализована операция поиска ячейки, в которую попадает трехмерный вектор, операции обхода сетки и доступа к соседям ячейки, операции вычисления координат вершин сетки, центров и площадей ячеек.

Сетка реализована в виде класса **Sphere<T>**, параметризованного по типу ячейки, обеспечивающего в том числе запись и чтение данных на диск в бинарном формате, кроме того библиотека предоставляет утилиту для визуализации таких данных. В большинстве случаев при численном моделировании сферическая сетка является удобной заменой традиционных сферических координат — кроме отсутствия сильных особенностей, сферическая сетка при той же точности (максимальном размере ячейки) требует примерно вдвое меньше ячеек за счет отсутствия сгущения узлов около полюсов.

Данные из сферической сетки могут сохранятся и загружаться с диска в бинарном формате, а также могут быть визуализированы при помощи описанного ниже вьювера **uplt**, рис. 3.

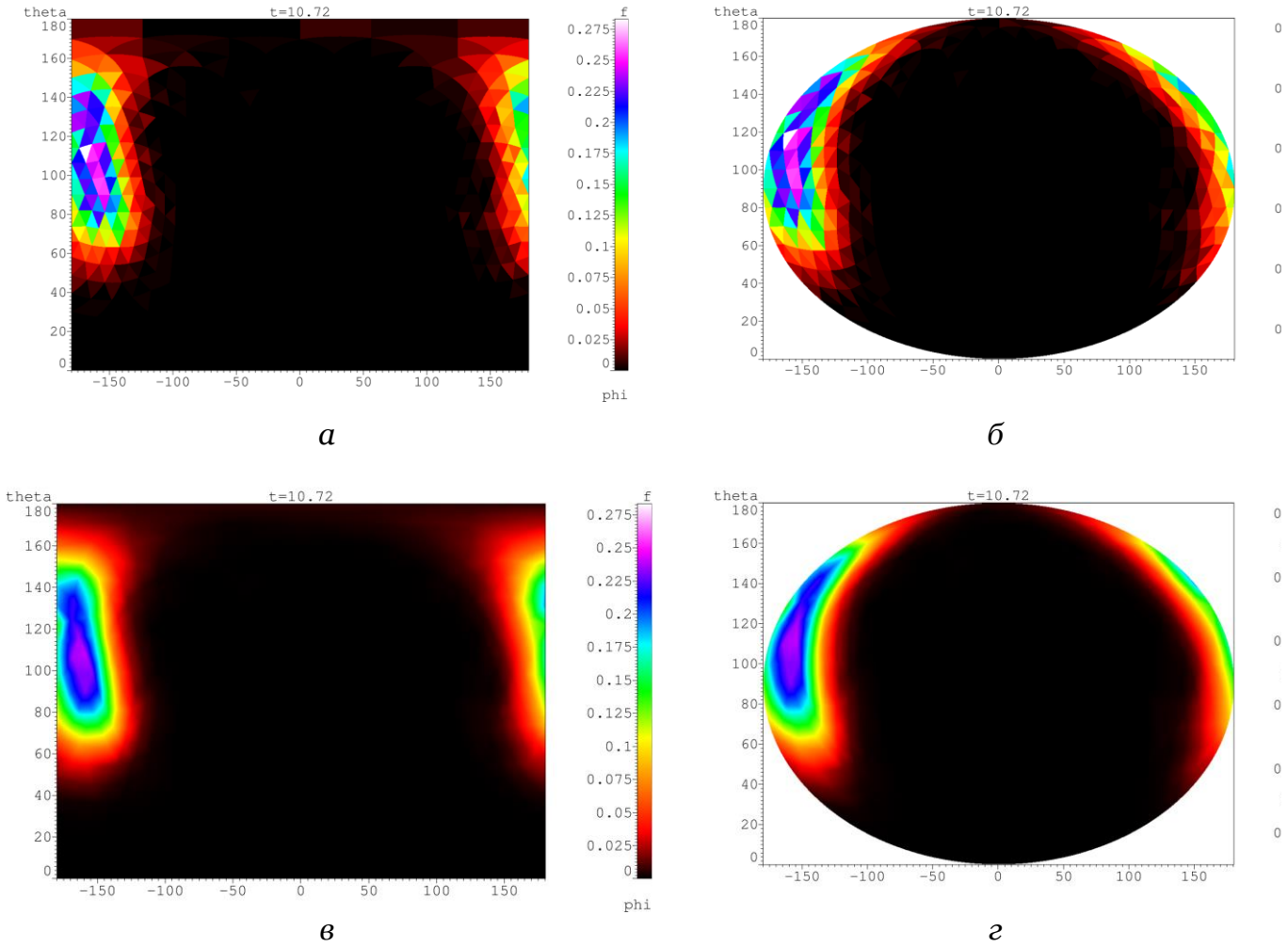


Рис. 3. Визуалиазация функции распределения магнитных моментов на сферической сетке третьего ранга в координатах θ,φ без интерполяции (*а, б*) и с линейной интерполяцией по ячейкам (*в, г*) в обычной проекции (*а, в*) и в проекции Молльвейде (*б, г*)

3 Массовый запуск расчетов и анализ результатов

При проведении численного моделирования результаты каждого расчета должны сопровождаться информацией об использованных параметрах расчета и алгоритмах, в противном случае через некоторое время результат превращается в абстрактную картинку. Если для сохранения параметров существует большое количество методик и библиотек, то сохранение алгоритмов является проблемой, и единственным приемлемым решением на сегодняшний день является сохранение исходного кода приложения.

При проведении массовых расчетов (например при анализе зависимости поведения устройства от нескольких параметров и построении фазовых диаграмм) требуется механизм, обеспечивающий многократный автоматический запуск приложения с меняющимися заданным образом параметрами, желательно с контролем распределения ресурсов в рамках локальной сети или на кластере.

Для анализа результатов необходим многопараметрический поиск по проведенным расчетам, для чего результаты расчетов должны храниться специальным, упорядоченным образом. Необходимо обеспечить возможность поиска по версиям исходного кода. Эту проблему можно решать вручную, например, размещая результаты расчетов на хорошо структурированном дереве каталогов — однако такой подход требует строгой внутренней культуры пользователя и усложняется тем, что в процессе расчетов критерии упорядоченности могут расширятся и изменяться кардинальным образом.

При массовых расчетах аккуратное решение вышеописанных проблем может отнимать значительное время и силы. В разных рабочих группах разработаны собственные библиотеки, позволяющие упростить процесс написания окружения [18, 19, 20], но единый подход до сих пор не выработан.

Описанная в данном разделе система **RACS** (Results & Algorithms Control System — система контроля результатов и алгоритмов) обеспечивает:

- задание параметров расчетов при запуске для приложений на языках **Python** и **C++**;
- автоматическое сохранение параметров и исходных кодов расчетов;
- пакетный запуск расчетов (циклы по значениям параметров) и балансировку загрузки как на локальных машинах, так и на кластерах под **MPI**;
- работу с контрольными точками для приложений **C++**, в том числе на кластерах под **MPI**;
- развитые средства для многопараметрического поиска, анализа и обработки результатов.

При разработке **RACS** делались следующие акценты:

- простота подключения (требуется минимальная модификация отлаженного кода);
- лаконичный и интуитивно понятный синтаксис при запуске расчетов;
- возможность обработки результатов средствами операционной системы и сторонними утилитами без потери целостности данных;
- интеграция с другими утилитами — вывод данных в формате **gnuplot** с заголовками **gplt**, чтение метаинформации о расчетах другими утилитами.

Даже для низкоквалифицированного пользователя **RACS** автоматически обеспечивает необходимый минимум “культуры” проведения расчетов (сохранение исходных кодов и параметров). В результате пользователь имеет возможность полностью сконцентрироваться на работе непосредственно над задачей.

При разработке системы **RACS** особый упор делался на простоту подключения к готовому приложению. **RACS** написан на языке **Python** и ориентирован в первую очередь на приложения, написанные на языках **C++** (высокопроизводительное вычислительное ядро) и **Python** (верхний управляющий слой приложения и интерфейсные части), связанные при помощи утилиты **SWIG** [3]. При этом для запуска расчетов **RACS** может быть подключен и к приложениям написанных только на языке **C++** без использования **Python**.

Как правило, расчет запускается из текущей (рабочей) директории, содержащей исходные коды, головные исполняемые файлы на **Python**-е или **C++** и т.д. Для каждого расчета в репозитории (некоторой директории) создается уникальная директория, в которой сохраняются параметры расчета, исходные коды и результаты моделирования. Набор параметров расчета образует словарь, который сохраняется в файле **.RACS** уникальной директории расчета, в формате стандартного модуля pickle языка **Python**.

Кроме файла **.RACS** в уникальной директории расчета могут создаваться файлы **.src.tgz** (архив с исходным кодом приложения выполнявшего расчет) и в случае “демонизации” расчета **logfile** (объединенные стандартный вывод и стандартный поток ошибок).

Репозиторий может быть структурирован произвольным образом, т.е. являться деревом директорий, в котором расчеты группируются согласно требованиям пользователя (например на основе значений ключевых параметров).

Отдельные расчеты и репозитории могут перемещаться средствами операционной системы, пересылаться по сети и т.д.

Расчет запускается из командной строки, при этом за счет подключения системы **RACS** аргументы командной строки позволяют изменять параметры расчета, производить серийный запуск расчетов и менять служебные параметры системы **RACS**. Допускается серийный запуск расчетов с перебором различных занчения парамеров и автоматической балансировкой загрузки компьютера.

Для анализа результатов моделирования используется утилита командной строки **racs**. В 2010-м году предпринимались попытки создать версию с оконным интерфейсом пользователя, однако быстро выяснилось, что это не дает никаких преимуществ, но при этом существенно усложняется взаимодействие с другими утилитами командной строки, кроме того, возникают проблемы с удаленной работой по **ssh**.

Утилита позволяет просматривать словарь параметров отдельного расчета (содержимое файла **.RACS**), строить выборки, отвечающие различным критериям, выводить выборки в различных форматах, модифицировать расчеты выборки, удалять расчеты выборки с диска.

Фактически, расчеты, помещенные под **RACS**, образуют нереляционную базу данных, при этом отдельные расчеты можно рассматривать как записи в базе, а репозитории как таблицы.

Утилита способна совместно обрабатывать нескольких репозиториев, репозитории обрабатываются последовательно (в том порядке, в котором были указаны), результаты сливаются в общую выборку (совокупность расчетов). Утилита **racs** дает пользователю широкие возможности по многопараметрическому поиску, анализу, совместной обработке и визуализации больших объемов результатов численного моделирования.

К настоящему моменту (первые версии появились в 2003 году, первая публикация [21] в 2007 году) **RACS** хорошо зарекомендовал себя при организации массовых расчетов в различных областях — сейсмике, моделировании разработки керогеносодержащих месторождений с

учетом внутрипластового горения, моделировании магнитных систем [22] и разработке устройств спинтроники [23], газодинамике горения [24], изучении резонансных свойств нелинейных систем [25] и т.д.

4 Средства визуализации

4.1 Утилита gplt — построение графиков типографского качества на основе gnuplot

Приложение **gnuplot** является одним из старейших средств визуализации проекта **GNU**. Несмотря на ряд недостатков (в первую очередь низкую производительность, особенно при визуализации поверхностей) **gnuplot** до сих пор пользуется популярностью благодаря гибкости, разнообразию форматов вывода и хорошему зарамочному оформлению графиков.

Для построения графиков типографского качества (с правильной шрифтовкой, подписями к осям и т.д.) **make** требует проведения довольно большой работы. Утилита **gplt** анализирует при запуске аргументы командной строки, формирует набор инструкций и запускает **gnuplot**. В итоге, за счет того что многие необходимые настройки заданы по умолчанию, и за счет лаконичного синтаксиса, для построения графика типографского качества в формате **.pdf** со шрифтовкой и формулами **LaTeX** необходим минимум действий (нажатий клавиш). При этом для каждого графика формируется **.gplt**–файл, содержащий аргументы соответствующего вызова утилиты **gplt** — при необходимости файл может быть отредактирован для внесения изменений в график.

Утилита **gplt** написана на языке **Python**, и при работе анализирует комментарии в **.dat**–файлах (текстовых файлах с данными для отрисовки). Комментарии специального вида (начинающиеся с символов #:) позволяют задавать имена для столбцов данных, задавать дополнительные численные константы, настраивать отображение названий столбцов для различных форматов вывода. Утилита **gplt** способна принимать достаточно произвольные выражения (фрагменты кода на языке **Python**), содержащие имена столбцов для отрисовки. При построении графиков для расчетов, находящихся под управлением системы **RACS**, утилита **gplt** также может использовать в выражениях метаинформацию о расчетах (содержимое **.RACS** файлов). Выражения по необходимости конвертируются через абстрактное синтаксическое дерево в форматы **gnuplot**, **LaTeX**, **EPS** и т.д.

4.2 Утилита uplt — визуализация данных на многомерных и сферических сетках в виде двумерных срезов

Для визуализации скалярных функций вида $f(x, y)$, заданных на равномерных прямоугольных или сферических сетках в бинарных форматах библиотеки **aiwlib**, применяется вьювер **uplt**, рисунок 4. Вьювер реализован на языках **C++11** и **Python** с использованием библиотек **Tkinter** и **Python Image Library**. Вьювер позволяет гибко настраивать палитру и пределы изображения, поддерживает задание логарифмических масштабов по всем осям и различные варианты интерполяции. При визуализации данных, записанных в один файл в виде последовательности независимых кадров (каждый кадр данные для одной сетки), вьювер позволяет перемещаться по кадрам. При визуализации равномерных сеток с размерностью больше двух данные отображаются в виде двумерного среза, при этом вьювер позволяет выбирать ориентацию и положение среза. Допускается запуск анимации (последовательная отрисовка файлов, кадров в одном файле или движение среза по одному кадру) и

автоматический монтаж видеоролика при помощи утилиты **ffmpeg**.

Вьювер допускает задание опций отрисовки как в графическом интерфейсе пользователя, так и при запуске через аргументы командной строки. Поддерживается импорт изображений в формат **PDF**.

Несмотря на то, что вьювер не использует графический ускоритель, обеспечивается приемлемая скорость визуализации для больших (гигабайты и более) объемов данных.

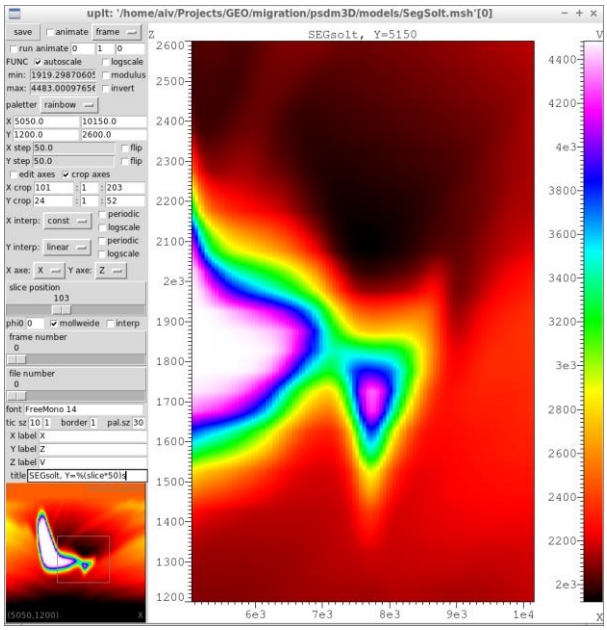
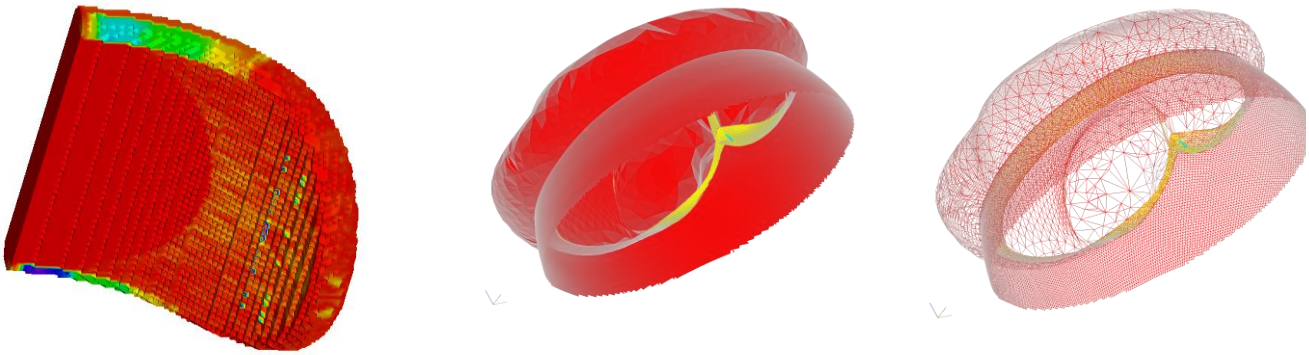


Рис. 4. Вертикальный разрез глубинно–скоростной модели известного комплекта международных синтетических сейсмических данных **SEGsalt**

4.3 Утилита splt — визуализация поверхностей, заданных треугольными неструктурированными сетками



а *б* *в*

Рис. 5. Уровень ошибки на неструктурированной тетраэдрической сетке при эволюции поверхности с подвижным краем (*а*) и волновой фронт при миграции на международном наборе сейсмических данных **SEGsalt** в виде поверхности (*б*) и так называемой “проволочной” модели сетки (*в*)

Три описанных далее вьювера (**splt**, **mplt** и **fplt**) построены на основе отдельного пакета **AbstractViewer**, не входящего непосредственно в библиотеку **aiwlib**. Пакет позволяет разрабатывать новые **3D** вьюверы на языках **C++** и **Python** с использованием библиотек **OpenGL** и **glut**, и встраивать их в приложения численного моделирования для отображения и анализа результатов непосредственно в процессе расчетов на **GPGPU**. Пакет разработан Хилковым С.А. и распространяется под лицензией **GPL-v3**.

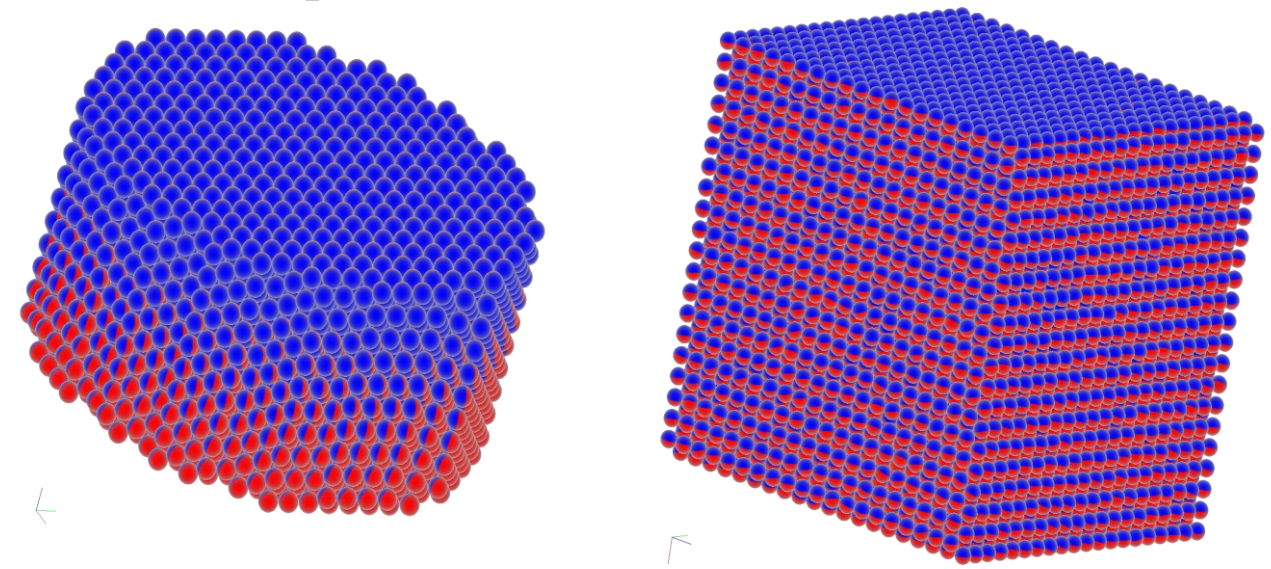
За счет эффективного использования дискретной видеокарты вьюверы имеют высокую производительность, интерфейс выполнен в нарочито “спартанском” стиле — манипулятор “мышь” позволяет разворачивать изображение, для остальных действий используются горячие клавиши и командная строка в терминале. Командная строка использует библиотеку **readline**, обеспечивающую контекстное дополнение буфера ввода и обращение к истории команд. Команды терминала обрабатываются при помощи языка **Python**, что позволяет при необходимости запускать целые сценарии по генерации серий изображений, созданию анимации и т.д.

Все вьюверы допускают смену палитр и пределов цветовой шкалы, использование плоскостей отсечения, масштабирование и поворот изображения.

Вьювер **splt** отображает достаточно произвольные поверхности, задаваемые на неструктурированной треугольной сетке (рис. 5), при этом могут отображаться данные с тетраэдрической сетки, заданной как совокупность граней. Сетки задаются в бинарном формате библиотеки **aiwlib**, в одном файле может размещаться последовательно произвольное количество независимых кадров (мгновенных состояний сетки), при этом у сеток в разных кадрах могут быть разные количества узлов и вершин. Вьювер обеспечивает эффективное переключение между кадрами.

С вершинами и ячейками треугольной сетки могут быть ассоциированы произвольные наборы данных типа **float4** (например, с вершинами ассоциируются координаты и компоненты скорости, с ячейками плотности вещества), каждое поле данных имеет уникальное текстовое имя. При отображении допускается выбор произвольных комбинаций полей — значения трех полей откладываются по осям *xyz*, значение четвертого поля показывается цветом. Это открывает широкие возможности по визуализации и анализу сложных данных, обнаружению различных зависимостей и изучению их эволюции в результатах расчетов.

4.4 Утилита mplt — визуализация распределений магнитных моментов и векторных полей



а *б*

Рис. 6. Распределение магнитных моментов для цилиндра из антиферромагнитного материала с неполной ГЦК решеткой (*а*) и для куба из ферромагнитного материала с ОЦК решеткой (*б*)

Для визуализации распределения магнитных моментов библиотека **aiwlib** предоставляет вьювер **mplt.** Намагниченность отдельной ячейки/атома отображается в виде красно–синей (по аналогии со стрелкой компаса) сферы, синий полюс которой ориентирован вдоль направления магнитного момента. Трехмерный массив таких сфер хорошо передает распределение намагниченности на поверхности образца (внутренние сферы практически не видны, рисунок 6), для анализа распределения намагниченности внутри образца используются срезы и/или плоскости отсечения.

Вьювер работает с данными в специальном бинарном формате библиотеки **aiwlib**. Результаты моделирования хранятся в одном файле. Файл состоит из заголовка, описывающего пространственное распределение магнитных моментов, и следующих за ним кадров, каждый кадр содержит значения магнитных моментов в некоторый момент времени. Для хранения ориентации одного магнитного момента используется либо вектор из трех чисел **float4** (12 байт), либо номер ячейки сферической сетки, построенной на основе рекурсивного разбиения додекаэдра пятого ранга (2 байта, точность порядка 1°).

Вьювер эффективно отображает массивы до 10^9 магнитных моментов, ограничением является лишь размер памяти видеокарты.

4.5 Утилита fplt — воксельная визуализация трехмерных равномерных сеток

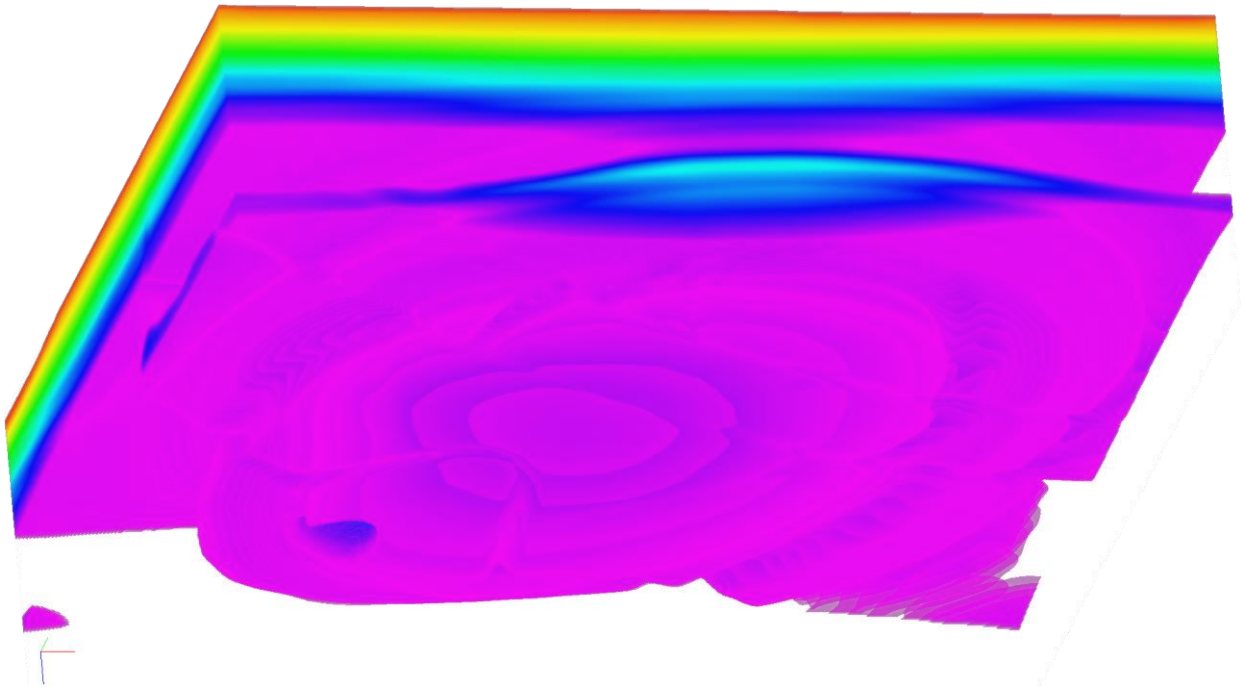


Рис.7. Трехмерное изображение глубинно–скоростной модели известного комплекта международных синтетических сейсмических данных SEGsalt

Вьювер **fplt** обеспечивает воксельную визуализацию равномерных трехмерных сеток сохраненных в формате библиотеки **aiwlib**, рисунок 7.

Пиксельный шейдер трассирует лучи с шагом порядка размера ячейки сетки, с учетом прозрачности. Начальное значение (цвет) считается невидимым (полностью прозрачным), трассировка каждого луча завершается после набора заданного значения в α – канале изображения (по умолчанию 0.95) [26] или выходе луча за пределы сетки.

5 Другие модули

Библиотека **aiwlib** предоставляет ряд дополнительных модулей для решения различных специфических задач.

На языке **C++** реализовано построение изолиний; контрольные точки для остановки и продолжения расчета; сериализация данных в формате **pickle**; дискретизация разбиения Вороного; сбор и сохранение метаинформации о пользовательских типах данных (структурах) для постобработки результатов расчетов; некоторые элементы линейной алгебры и аналитической геометрии, в том числе построение проекций, операции поворота; средства для чтения/записи тестовых конфигурационных файлов и настройки объектов пользователя.

На языке **Python** реализован эффективный разбор аргументов командной строки, позволяющий фактически создавать свои **DSL** (**Domain–Specific Languages** — предметно–ориентированные языки); высокоуровневая работа с датой и временем; автоматическое определение файлов с исходным кодом приложения, написанного на языках **C++** и **Python**, связанных утилитой **SWIG** и собранных при помощи **make**; экономичный протокол передачи данных через сокеты и высокоуровневые средства разработки несложных сетевых клиент–серверных приложений; конвертация алгебраических выражений из формата языка **Python** в другие форматы на

основе абстрактного синтаксического дерева.

6 Заключение

Первая версия библиотеки **aiwlib** была написана в 2008 году, а отдельные ее части (система **RACS** и утилита **gplt**) возникли еще в 2003 году. За прошедшие годы библиотека значительно изменилась, код ядра стал проще, компактнее, стабильнее а его функциональность существенно расширилась.

Библиотека **aiwlib** и ее предыдущая версия **aivlib** успешно использовались в целом ряде проектов: при разработке программных комплексов для нужд сейсморазведки [27, 28, 29], физики плазмы [6] и оптики мутных сред [30]; решении фундаментальных и прикладных задач по изучению магнитных материалов и созданию устройств спинтроники [22, 23]; изучении резонансных свойств суперапарамагнетиков [25]; моделировании газодинамики и горения [24], процессов разработки керогеносодержащих нефтяных месторождений с учетом внутрипластового горения, деградации материалов и блистеринга в приповерхностных слоях под влияем ионной бомбардировки [31]; моделировании задач пороупругости и гидроразрыва пласта [32]. При этом большую роль играли развитые средства визуализации библиотеки – за счет эффективного анализа результатов моделирования значительно упрощалась отладка кода и адаптация численных схем к специфике решаемых задач.

Ядро библиотеки распространяется под лицензией **Apache–2** (что допускает ее применение в коммерческих проектах с закрытым исходным кодом), утилиты визуализации распространяются под лицензией **GPL–3**.

Библиотека **aiwlib** продолжает активно развиваться. В связи с накоплением определенного опыта планируется включить в библиотеку модули для языка **CUDA**, а также заготовки (фреймворки) для создания кодов по моделированию магнитных материалов и различных вариантов **FEM/XFEM**.

Литература

[1] Г. Россум, Ф.Л.Дж. Дрейк, Д.С. Откидач. Язык программирования Python. 2001 — С. 1–454.

[2] М. Лутц. Программирование на Python. С-Пб.: <<Символ>>. 2002 — С. 1–1135.

[3] Simplified Wrapper and Interface Generator http://www.SWIG.org

[4] Р. Столлман, Р. МакГрат. GNU Make. Программа управления компиляцией. 1995.

[5] С. Мейерс. Эффективный и современный C++. Издательский дом <<Вильямс>>. — 2016. — С. 1–303.

[6] A. Yu. Perepelkina, I. A. Goryachev, V. D. Levchenko CFHall Code Validation with 3D3V Weibel Instability Simulation. //Journal of Physics: Conference Series. IOP Publishing. — 2013. — V. 441. — No. 1. — P. 012014

[7] A. Yu. Perepelkina, V. D. Levchenko, I. A. Goryachev Implementation of the Kinetic Plasma Code with Locally Recursive non-Locally Asynchronous Algorithms. //Journal of Physics: Conference Series. — IOP Publishing. — 2014. — V. 510. — No. 1. — P. 012042

[8] V. D. Levchenko, A. Yu. Perepelkina, A. V. Zakirov. DiamondTorre Algorithm for High-Performance Wave Modeling. //Computation 4.3 (2016): 29.

[9] V. D. Levchenko, A. yu. Perepelkina. The DiamondTetris Algorithm for Maximum Performance Vectorized Stencil Computation. //International

Conference on Parallel Computing Technologies. Springer, Cham. — 2017. — P. 124–135.

[10] К.Ш. Тан, В.-Х. Стиб, Й. Харди. Символьный C++: введение в компьютерную алгебру с использованием объектно–ориентированного программирования. — М.:<<Мир>>. — 2001. — 622 С.

[11] С.А. Жданов, А.В. Иванов Пример автоматической генерации кода приложения численного моделирования для решения уравнения Фоккера–Планка. //Математическое моделирование. — 2015. — Т. 27. — № 9. — С. 49–64.

[12] А.В. Иванов, С.А. Хилков, С.А. Жданов Библиотека aivlib. http://a-iv.ru/aivlib/

[13] А.В. Иванов, С.А. Хилков Библиотека aiwlib
https://github.com/aivn/aiwlib/blob/master/doc/aiwlib.pdf

[14] G. M. Morton. A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. — Ottawa, Canada: IBM Ltd., 1966. — (Technical Report).

[15] Краснов М.М. Метапрограммирование шаблонов C++ в задачах математической физики. — М.: ИПМ им. М.В.Келдыша, 2017. — 84 с.

[16] А.В. Иванов Кинетическое моделирование динамики магнетиков //Математическое моделирование. — 2007. — Т. 19. — № 10. — С. 89-104.

[17] S.A. Khilkov, A.V. Ivanov Numerical simulation of the magnetic moment distribution evolution for superparamagnetic materials //Препринты ИПМ им. М.В. Келдыша. — 2014. — № 29. — С. 1-16.

[18] А.Е. Александров, А.В. Тюрин Программные инструментальные средства для организации вычислительного эксперимента с целью проведения многовариантного анализа //Научно-технический вестник информационных технологий, механики и оптики. — 2015. — Т. 15. — № 5. — С. 907-915.

[19] Многовариантный анализ. Программный комплекс для автоматизации моделирования нестационарных процессов в механических системах и системах иной физической природы. http://www.laduga.ru/pradis/help/pradis/PRADIS_Multivaria_analysis.ru.htm

[20] B. M. Adams. at al. Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.6 User's Manual. —SAND2014-4633. — 2017. — С. 1–319. https://dakota.sandia.gov/

[21] А.В. Иванов Система контроля результатов и алгоритмов для задач численного моделирования //Автоматизация. Современные технологии. — 2007. — № 12. — С. 29–34.

[22] S.A. Khilkov, A.V. Ivanov, E.V. Zipunova Numerical simulation of strongly nonequilibrium processes in magnets based on physical kinetics equations //Mathematical Models and Computer Simulations. — 2016. — Т. 8. — № 6. — С. 703-708.

[23] И.М. Искандарова, А.В. Иванов, А.А. Книжник, А.Ф. Попков, Б.В. Потапкин, П.Н. Скирдков, К.А. Звездин Моделирование фазовых диаграмм переключения для термоассистированных наноприборов MRAM //Российские нанотехнологии. — 2015. — Т. 10. — № 11-12. — С. 112-117.

[24] M.A. Liberman, M. Kuznetsov, A. Ivanov A., I. Matsukov Formation of the preheated zone ahead of a propagating flame and the mechanism underlying the deflagration-to-detonation transition //Physics Letters A. — 2009. — Т. 373. — № 5. — С. 501-510.

[25] С.А. Хилков, А.В. Иванов Резонансные свойства суперпарамагнетиков при малых амплитудах внешнего периодического поля //Математическое моделирование. — 2015. — Т. 27. — № 8. — С. 96-110.

[26] CUDA Toolkit Documentation: Volume Rendering with 3D Textures http://docs.nvidia.com/cuda/cuda-samples/index.html#volume-rendering-with-3dtextures

[27] В.Д. Левченко, А.Ю. Перепёлкина, А.В. Иванов, А.В. Закиров, Т.В. Левченко, В.Е. Рок Высокопроизводительное динамическое 3d моделирование полноволнового сейсмического поля в задачах сейсморазведки. опыт применения в условиях различных сейсмо-геологических регионов // Суперкомпьютерные технологии в нефтегазовой отрасли. Математические методы, программное и аппаратное обеспечение Материалы научно-практической конференции. — 2017. — С. 49-53.

[28] А.В. Закиров, В.Д. Левченко, А.В. Иванов, А.Ю. Перепелкина, Т.В. Левченко, В.Е. Рок Высокопроизводительное 3d моделирование полноволнового сейсмического поля для задач сейсморазведки //Геоинформатика. — 2017. — № 3. — С. 34-45.

[29] А.Л. Плешкевич, А.В. Иванов, В.Д. Левченко, С.А. Хилков. Многолучевая 3D глубинная сейсмическая миграция до суммирования с сохранением амплитуд //Геофизика. спец. выпуск <<50 лет ЦГЭ>>. — 2017. — С. 89–97.

[30] A.V. Dmitriev, A.V. Ivanov, A.R. Khokhlov Numerical simulation of light propagation through a diffuser //Journal of Mathematical Sciences. — 2011. — Т. 172. — № 6. — С. 782-787.

[31] G.I. Zmievskaya, A.L. Bondareva Kinetics of the formation of pores and a change in the properties of materials in numerical model //Journal of Surface Investigation: X-Ray, Synchrotron and Neutron Techniques. — 2016. — Т. 10. — № 4. — С. 802-808.

[32] А.В. Иванов, Е.Б. Савенков. Моделирование и визуальное представление динамики поверхности с подвижным краем на стационарной неструктурированной сетке //Научная визуализация. —2017. —Т. 9. — № 2. — С. 64–81.