КОМБИНИРОВАНИЕ СИМВОЛЬНОЙ АЛГЕБРЫ И ГЕНЕРАЦИИ КОДА ДЛЯ РЕШЕНИЯ СЛОЖНЫХ СИСТЕМ НЕЛИНЕЙНЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

О.А. Хлыбов

Институт механики сплошных сред УрО РАН, Пермь, 614013, Россия

Разработан пакет, предназначенный для последовательного и параллельного численного решения систем нелинейных дифференциальных уравнений, который обеспечивает повышенную гибкость по сравнению с распространенными коммерческими пакетами численного моделирования. Изначально пакет ориентирован на решение задач гидродинамики, однако он способен решать любые другие задачи, которые можно описать в терминах систем (не)линейных дифференциальных уравнений. Ближайшими аналогами являются открытые и коммерческие пакеты FreeFEM++, OpenFOAM и FlexPDE, но они построены на несколько иных принципах.

APPLICATION OF THE ELEMENTS OF SYMBOLIC COMPUTATIONS AND AUTOMATIC CODE GENERATION TECHNIQUE TO THE SOLUTION OF COMPLEX PDE PROBLEMS

O.A. Khlybov

Institute of Continuous Media Mechanics UB RAS, Perm, 614013, Russia

A software package intended for serial and parallel numerical solution of PDE systems has been developed. The package provides extra flexibility compared to existing commercially available general-purpose computational systems. Although the package is initially aimed at solving CFD problems, it can however be applied to any field where the numerical solution of complex (non)linear PDE systems is required. The closest counterparts such as FreeFEM++, OpenFOAM or FlexPDE are based on similar ideas, but differ in certain important aspects.

1. Введение

Автором разработан программный пакет «Finita», предназначенный для последовательного и параллельного численного решения систем нелинейных дифференциальных уравнений, который обеспечивает повышенную гибкость по сравнению с распространенными коммерческими пакетами численного моделирования. Изначально пакет ориентирован на решение задач гидродинамики, однако он способен решать любые другие задачи, которые можно описать в терминах систем линейных и нелинейных дифференциальных уравнений. Ближайшими аналогами являются открытые и коммерческие пакеты FreeFEM++ [1], OpenFOAM [2] и FlexPDE [3], но они построены на несколько иных принципах. В частности, FreeFEM++ и FlexPDE используют исключительно метод конечных элементов, а OpenFOAM – методы конечных объёмов и конечных элементов; последний также использует язык программирования общего назначения (C++) в качестве языка описания задачи (проблемно-ориентированного языка), тогда как первые два используют стандартный подход для такого типа пакетов с собственным специализированным языком. Отличительной особенностью представленного пакета является использование технологии автоматической генерации исходного текста программ по заданному описанию задачи вместо непосредственного

исполнения последнего, как это сделано в названных аналогах, что ослабляет зависимость численного кода от программного окружения, в котором он исполняется, тем самым значительно улучшая его переносимость.

В настоящее время пакет способен решать стационарные и нестационарные задачи на двумерной четырёхугольной сетке с использованием полуавтоматической и автоматической дискретизации дифференциальных уравнений методами конечных разностей (МКР) и конечных объёмов (МКО). Под автоматической дискретизацией в данном случае понимается получение разностных аналогов дифференциальных уравнений, записанных с использованием стандартных дифференциальных операторов. При полуавтоматической дискретизации пакету (в случае МКО) требуются подсказки относительно вида интегрирования (либо по собственно контрольному объёму, либо по его поверхности) отдельных слагаемых, составляющих дифференциальное уравнение.

Реализованы автоматические и полуавтоматические конечно-разностные и конечно-объёмные [4] дискретизаторы, преобразующие исходные дифференциально-алгебраические уравнения в эквивалентные разностные аналоги. Дискретизация производится на прямоугольной сетке с фиксированным межузловым шагом, равным единице; отображение физической расчётной области на вычислительную сетку осуществляется с помощью преобразования координат, которое задаётся пользователем аналитически или является результатом работы внешнего генератора сеток.

Дискретизированные системы линейных и нелинейных алгебраических уравнений решаются методом Ньютона с привлечением сторонних решателей систем линейных алгебраических уравнений (СЛАУ), которые допускают выбор между последовательным и параллельным методами решения. В последнем случае возможно использование многопроцессорных систем с общей либо распределённой памятью (кластеров).

Дискретизированные системы нелинейных алгебраических уравнений разбиваются на подсистемы для получения различных схем решения — от полностью неявной (все неизвестные поля решаются совместно) до сегрегированной (неизвестным последовательно считается каждое поле и для него формируется своя процедура решения; на время работы этой процедуры все остальные поля считаются заданными). При этом каждая подсистема может иметь свои отдельные параметры и даже библиотеку решателя СЛАУ. Например, при решении гидродинамической задачи в силу значительного различия тепловых и гидродинамических времён может оказаться выгодным разделить систему определяющих уравнений на две части — совместную систему «скорость+давление» и отдельную систему «температура», и достигать решения последовательным итерированием этих двух подсистем.

В пакете в том или ином виде задействованы три языка программирования:

- Ruby [5] – Основной язык пакета, динамический интерпретируемый объектно-ориентированный язык высокого уровня, который используется в качестве проблемно-ориентированного языка для описания входного задания, в модуле дискретизации дифференциальных уравнений и в модуле генерации кода. Исполняемая система Ruby имеет невысокое быстродействие (что, впрочем, характерно для всех интерпретирующих систем) но доступна на всех наиболее распространённых платформах (Windows/UNIX/Mac).
- Eiffel [6] – статически типизируемый компилируемый объектно-ориентированный язык высокого уровня, используемый в модуле аналитических вычислений, оформленном в виде расширения для языка Ruby. В представленном пакете используется диалект GNU/SmartEiffel [7], особенностью которого является трансляция в ANSI/C всего кода вместе со стандартной библиотекой Eiffel, что даёт возможность избавиться от Eiffel-кода уже на этапе сборки собственно пакета. Диалект GNU/SmartEiffel является самоподдерживающимся (то есть написанным на себе самом), что гарантирует его

работу на различных платформах. Генерируемый же им ANSI/C код не имеет никакой привязки к платформе, на которой он создавался.

- ANSI/C – компилируемый язык низкого уровня, («переносимый ассемблер»), который используется как целевой язык для генерации кода и написания пользовательского кода, специфичного для решаемой проблемы, а также для организации взаимодействия Ruby/Eiffel.

2. Входное задание

Входное задание, подготавливаемое пользователем, представляет собой текст на проблемно-ориентированном языке (Ruby, в данном случае), загружаемый в специально сконфигурированное программное окружение, и состоит из четырёх блоков:

Блок 1 содержит описания скалярных констант, переменных и полей, имеющих отношение к решаемой задаче. Все описанные переменные доступны из пользовательского кода под заданными именами. В настоящее время доступны только действительные типы данных; комплексные пока не поддерживаются Для констант указывается обязательное значение, которое остаётся неизменным на всём протяжении выполнения кода. Для переменных начальное значение устанавливается до выполнения пользовательского кода и может быть изменено пользователем в любой момент; переменные, не инициализированные явно, получают нулевое значение. Скалярные поля имеют одинаковую размерность, равную размерности расчётной сетки и также инициализируются нулями. Доступ к элементам полей из пользовательского кода, написанного на языке Си (в дальнейшем Си-кода), осуществляется интуитивно понятным образом (например, T(i,j+1)).

Блок 2 описывает геометрию задачи. При формировании входного задания указывается размерность задачи и число узлов в каждом измерении; все скалярные поля получают эту размерность. Расчётная область представляется в виде набора перекрывающихся прямоугольных подобластей на расчётной сетке. В настоящее время реализованы подобласти трёх типов: открытая, закрытая и граничная. Разделение обусловлено различием дискретных аналогов дифференциальных операторов в зависимости от того, является ли конкретный узел граничным или внутренним. На двумерной сетке с толщиной границы в один узел можно выделить девять участков, в пределах которых дискретный аналог дифференциального оператора остаётся неизменным. В общем случае одно и то же дифференциальное выражение имеет различные дискретизации на разных участках сетки. При дискретизации дифференциального выражения на закрытой подобласти дискретному аналогу не позволяется ссылаться на узлы, находящиеся вне её. В применении к конечно-разностной дискретизации это может означать, например, использование направленных разностей вместо центральных в граничных узлах. Таким образом, в случае закрытой подобласти с толщиной границы в один узел, дифференциальное выражение будет иметь девять или менее дискретных аналогов. Открытая подобласть, напротив, не имеет такого ограничения (то есть дискретизация может захватывать узлы, лежащие за пределами этой подобласти), поэтому то же самое выражение будет иметь всего один дискретный аналог. В дифференциальных выражениях на подобных подобластях можно использовать операторы дифференцирования по координатным осям.

Граничная подобласть является расширением открытой подобласти и применяется для задания граничных условий. В двумерном случае она представляет собой вертикальный или горизонтальный ряд узлов, для которых определены направления нормалей и касательных. Поэтому, помимо операторов дифференцирования по координатным осям, в них определены также и операторы дифференцирования по

нормали и касательной. В случае использования конечно-разностной дискретизации такие операторы всегда аппроксимируются направленными разностями.

Блок 3 содержит системы дифференциально-алгебраических уравнений. Полная система уравнений и граничных условий, описывающих задачу, представляется в виде одного или более упорядоченных именованных списков, каждый из которых составляет систему уравнений, решаемую совместно. Элементом списка является само уравнение, представленное в аналитическом виде вкупе с подобластью, на которой оно определено, и полем, относительно которого оно разрешается; граничное условие ставится аналогично. На этапе генерации кода каждый такой список превращается в систему нелинейных алгебраических уравнений, решаемую отдельно. Имя списка задаёт название функции, видимой из пользовательского Си-кода, которая и запускает процедуру решения. Уравнение / граничное условие представляется в виде абстрактного синтаксического дерева для аналитического выражения вида $f = 0$; в f могут входить константы, переменные и поля, определённые ранее; нулевая правая часть опускается. В выражении используются как предопределённые дифференциальные операторы, так и относительные ссылки на соседние узлы конкретного поля (вида T(j+1)), что даёт возможность задавать пользовательские дискретные аналоги без переопределения уже существующих дифференциальных операторов. В настоящее время реализованы конечно-разностные и конечно-объёмные операторы на прямоугольной однородной сетке с единичным шагом. В случае конечно-разностных операторов определены операторы дифференцирования первого и второго порядка по координатам, а также касательным и нормальным направлениям (в случае задания выражения на граничной подобласти). В случае конечно-объёмных операторов реализованы операторы дифференцирования первого порядка по координатам (формирование операторов второго порядка производится рекурсивным вызовом, например $\frac{\partial^2}{\partial x^2} = \frac{\partial}{\partial x}\frac{\partial}{\partial x}$). Описание граничных условий в этом случае производится с помощью конечно-разностных операторов дифференцирования по нормали и касательной. Допускается смешивание различных методов дискретизации в разных уравнениях в пределах одной системы.

Блок 4 заканчивает сборку задания. Здесь производится установка общих параметров для всего генерируемого модуля – таких как список актуальных (то есть подлежащих обработке) систем уравнений, используемые решатели СЛАУ, имена и расположение генерируемых файлов и так далее.

3. Способ применения пакета

Процесс численного решения поставленной задачи с использованием пакета состоит из нескольких этапов:

1. Подготовка входного задания (*Приложение 1*). В наиболее простом случае пользователем подготавливается два исходных файла — входное задание, в которой описывается часть проблемы, подлежащая обработке пакетом, и исходный текст на языке Си, содержащий части кода, специфичные для решаемой задачи, такие как задание входных параметров, вызов генерированных решателей и вывод результатов расчётов.
2. Вызов пакета с передачей ему сформированного входного задания. При успешном завершении работы создаётся два исходных файла на Си: интерфейсный заголовочный, содержащий описания констант, переменных и полей, а также описания функций, допускающих вызов из пользовательского кода; Си-модуль, содержащий собственно реализацию.
3. Трансляция пользовательского Си-кода (*Приложение 2*) и автогенерированного модуля в объектные модули с помощью выбранного Си-компилятора.

4. Полученные объектные файлы собираются в исполняемую программу с помощью редактора связей. Последнему передаются все объектные файлы и библиотеки, необходимые для работы используемого решателя систем алгебраических уравнений с соответствующими параметрами командной строки.

5. Собранная программа запускается на выполнение. Запуск программы осуществляется с параметрами, управляющими используемым решателем систем алгебраических уравнений (не все поддерживаемые решатели имеют такую возможность).

Используемая схема работы предоставляет пользователю большую гибкость в выборе программно-аппаратных средств, на которых может осуществляться решение задачи. В частности, этапы 1 и 2 могут выполняться на локальной машине пользователя с мощностью, недостаточной для численного решения задачи, этапы же 3-5 – на хорошо оснащённой удалённой рабочей станции или кластере. В последнем случае также возможно разделение процесса решения задачи на 3 и 4 этапы, выполняемые на выделенной «фабрике компиляции», и 5 этап, выполняемый непосредственно на суперкомпьютере или кластере. В силу платформенной независимости пакета этапы 1 и 2 могут выполняться на компьютерах иной архитектуры, работающих под управлением другой операционной системы, нежели последующие.

В настоящее время в пакете реализована поддержка трёх решателей систем линейных и нелинейных алгебраических уравнений: SuperLU [8], UMFPACK [9] и PETSc [10]. Вся логика взаимодействия с выбранным решателем локализована в автогенерируемом модуле, пользовательский же код полностью изолирован от решателя и не требует изменения при смене последнего. Тем не менее, при смене решателя происходит перегенерация как заголовочного файла, так и модуля реализации, вследствие чего требуется перетрансляция всех исходных файлов, составляющих программу, с последующей пересборкой исполняемого модуля.

Из трёх поддерживаемых пакетов первые два (SuperLU, UMFPACK) реализуют прямые методы решения СЛАУ с возможностью уточнения результата итеративными методами. Третий решатель (PETSc) объединяет под общим интерфейсом множество различных алгоритмов решения систем алгебраических уравнений, как прямых, так и итеративных. Более того, он позволяет комбинировать различные методы «на ходу», выстраивая цепочки «предобуславливатель-решатель» [11] для получения наиболее эффективной схемы решения задачи.

Пакеты SuperLU и PETSc имеют как последовательную, так и параллельную версию решателя СЛАУ на системах с общей памятью и на системах с распределенной памятью (кластерах). Пакет UMFPACK имеет только последовательную версию.

4. Оценка эффективности

В качестве формального количественного критерия оценки эффективности пакета примем коэффициент размножения (КР), который равен отношению числа строк автогенерированного кода к числу строк входного задания. Использование числа строк кода в качестве метрики оправдано тем, что пакет предназначается для автоматизации кодирования программ, то есть автогенерированный код занимает место кода, который, в противном случае, пришлось бы написать вручную. Иными словами, увеличение коэффициента размножения отражает рост эффективности применения пакета. Следует отметить, что данная метрика – Source Lines of Code (SLOC), является самой распространённой в индустрии производства программного обеспечения. Конечно, подобный подход является достаточно формальным, поскольку число строк кода в каждом конкретном случае варьируется, например, в зависимости от стиля форматирования исходного текста, насыщенности комментариями и так далее, и не

учитывает другие параметры кода, такие как алгоритмическую и синтаксическую сложность, зависимость от языка реализации и пр. Тем не менее, он позволяет наглядно продемонстрировать практически достижимый эффект от применения представленного подхода к численному решению задач.

Полное описание задачи, подготавливаемое пользователем, как было сказано выше, состоит из двух частей: входное задание для пакета и управляющий модуль на языке Си. Управляющий модуль не учитывается при вычислении метрики, поскольку он, в соответствии с идеологией применения пакета, должен быть написан, вне зависимости от того, применяется ли пакет для автогенерации кода, или же последний написан вручную. Помимо собственно коэффициента размножения, интерес также представляет и размер входного задания, который показывает объём работы, непосредственно выполняемой пользователем в случае использования пакета.

Для оценки эффективности пакета рассмотрен широкий спектр задач, от тривиальных до весьма сложных в кодировании:

1. Тривиальное уравнение $f = 0$ на открытой подобласти.
2. Аналогичная предыдущей задача но на закрытой подобласти.
3. Уравнение Лапласа $\Delta f = 0$ с граничными условиями первого рода, дискретизированное методом конечных разностей.
4. Стационарная двумерная задача о свободной тепловой конвекции в прямоугольной области, подогреваемой сбоку. Дискретизация методом конечных разностей, совместное решение уравнений для температуры, завихренности и функции тока.
5. Задача моделирования течения и теплопереноса при выращивании полупроводникового кристалла методом Бриджмена [4], которая решается в псевдостационарной осесимметричной постановке. Форма и положение фронта кристаллизации не известны *a priori* и подлежат нахождению наряду с полями температуры, завихренности и функции тока. Уравнения дискретизированы методом конечных объёмов и решаются совместно.
6. Аналогичная предыдущей задача но с добавлением массопереноса и концентрационной конвекции.
7. Задача моделирования течения и тепло/массопереноса при выращивании полупроводникового кристалла методом погружённого нагревателя (SHM). Постановка аналогична задаче 6.
8. Комбинация задач 5 и 6; генерация двух систем уравнений на основе одного входного задания.

В таблице 1 представлены рассчитанные коэффициенты размножения и размеры входных заданий для задач 1–8, перечисленных выше.

Таблица. Коэффициенты размножения и размеры входных заданий для рассмотренных тестовых задач

Задача	Коэффициент размножения (КР)	Размер входного задания, строк
1	20	44
2	21	44
3	19	50
4	14	77
5	20	240
6	21	251
7	25	332
8	39	253

Все задачи сгенерированы с использованием библиотеки SuperLU в качестве решателя СЛАУ; использование других решателей не внесло каких-либо серьёзных качественных или количественных изменений в приведённые результаты.

Задачи 1 и 2 не представляют самостоятельной научной ценности, но они наглядно демонстрируют минимальный объём кода на входе и на выходе пакета.

В задаче 1 автогенерированный код состоит из двух исходных файлов и имеет суммарный объём 896 строк или 21 Кб текста на языке Си. С учётом практически нулевого полезного действия такого кода, последний целиком представлен «обвязочным» кодом, необходимым для нормального функционирования неявной схемы решения. Даже несмотря на неоптимальность автогенерированного кода (превосходящего по объёму код аналогичной функциональности, но написанный вручную), выгода в применении пакета состоит в несравненно меньших трудозатратах на этапе кодирования задачи. Что же касается входного задания, то из 44 строк кода, непустых строк было всего 20, а обязательных и того меньше – 11. Большая разница между общим числом строк и обязательным числом строк во входном задании объясняется выбранным «разреженным» стилем кодирования и наличием дополнительного кода, не принципиального с точки зрения результата; подобный стиль кодирования применялся и во всех остальных рассмотренных задачах. Таким образом, применение пакета позволяет двадцатикратно уменьшить объём ручной работы даже в подобном тривиальном случае.

При переходе от задачи 1 к задаче 2 происходит разбиение области на девять подобластей, с постановкой в каждой из них своей дискретизации исходного уравнения (одинаковых в данном конкретном случае в силу тривиальности рассматриваемой задачи). Незначительное изменение во входном задании, однако, приводит к увеличению объёма автогенерированного кода до 926 строк, что находит своё отражение в соответствующем увеличении КР.

Задача 3 моделирует реальную, хотя и простую ситуацию – решение уравнения Лапласа, дискретизированное методом конечных разностей и решаемое неявно. Несмотря на линейность системы «уравнение+граничные условия», решается она методом Ньютона. В целом, получаемый код аналогичен таковому же из задачи 2: вычислительная сетка также разбивается на 9 подобластей, но только в данном случае каждой подобласти ставятся в соответствие различные алгебраические уравнения, что приводит к увеличению размера автогенерированного кода до 975 строк. Тем не менее, увеличение размера входного задания до 50 строк приводит к некоторому уменьшению КР.

Задача 4 представляет решение классической проблемы, описываемой нелинейной системой дифференциальных уравнений для трёх неизвестных. Два дополнительных неизвестных и новые уравнения и граничные условия, добавленные к задаче 2, приводят к заметному увеличению размера входного задания до 77 строк и уменьшению КР до 14. Но даже и в этом случае автогенерация кода вполне оправдывает себя.

Задача 5 значительно сложнее предыдущих. Она включает в себя 34 скалярных параметра и 19 неизвестных полей. Основным осложняющим фактором является комплексная геометрия задачи: расчётная область представляет собой цилиндр с твёрдой боковой стенкой конечной толщины, заполненный материалом в двухфазном состоянии : снизу – твёрдая фаза (кристалл), сверху – жидкая (расплав). Форма и положение поверхности раздела (фронта кристаллизации) входит в число неизвестных. Вследствие использования неоднородной неортогональной четырёхугольной сетки стенка цилиндра также разбивается на две области. При этом уравнения, описывающие поведение системы, во всех четырёх подобластях зависят от четырёх геометрических факторов, что в сумме даёт 16 полей, полностью описывающих геометрию расчётной области. Использование для каждой из подобластей собственных полей геометрических факторов

обусловлено наличием их скачков на границах, поскольку в использованной схеме разбиения пространства на элементарные объёмы граница между подобластями проходит по узлам вычислительной сетки.

Усложнение задачи отразилось на размере входного задания, но разрастание автогенерированного кода до внушительных размеров (4814 строк) обеспечило значительный рост КР по отношению к предыдущей задаче и выявило один немаловажный факт: применение ручного кодирования для совместного решения сложных систем нелинейных уравнений нецелесообразно в силу быстрого роста размера исходного текста программ и, как следствие, трудозатрат на кодирование, отладку и поддержку такого кода. В то же время, применение пакета позволяет эффективно решать подобные и даже более сложные задачи. Подтверждением этому является задача 6 – развитие задачи 5 с добавлением одного неизвестного и соответствующих уравнений и граничных условий. Получающийся при этом код разрастается уже до 5311 строк при незначительном увеличении размера входного задания и вызывает дальнейший рост КР.

Задача 7 представляет собой дальнейшее развитие задачи 6 в части усложнения геометрии – в этом случае внутрь жидкой фазы помещается твёрдый цилиндр, в результате чего число подобластей с собственными коэффициентами преобразования возрастает до 9. Соответственно, внутри цилиндра и на его поверхности ставятся дополнительные уравнения и граничные условия. Описание геометрии занимает довольно значительную часть входного задания и последнее увеличивается в размерах до 332 строк. Но более сильное воздействие усложнение геометрии задачи оказывает на размер автогенерированного кода, который увеличивается до 8429 строк, и рост КР.

Наконец, задача 8 является слиянием задач 5 и 6 в рамках одного входного задания, из которого генерируются две системы уравнений, решаемых совместно: «температура+завихренность+функция тока» и «температура+завихренность+функция тока+концентрация примеси». Для получения решения приходится последовательно решать сначала первую систему уравнений, затем – вторую, для которой результаты решения первой служат начальным приближением. Данная ситуация объясняется сильной нелинейностью задачи 8 в интересуемом диапазоне параметров и прекрасно демонстрирует работоспособность и эффективность принципов, заложенных в пакет. Одновременно выявляются проблемы, возникающие при ручном кодировании систем уравнений, решаемых совместно: даже незначительные изменения во входном задании вызывают тектонические изменения в коде. Применительно к данной задаче добавление одного неизвестного и сопутствующих уравнений кардинальным образом меняет структуру Якобиана и функции, вычисляющие значения Якобиана и вектора невязок. В результате размер автогенерированного кода возрастает до 7346 строк, а коэффициент размножения достигает значения 39.

Что же касается скорости работы автогенерированного кода, то здесь следует учесть, что при решении задач методом Ньютона львиная доля машинного времени затрачивается на решение СЛАУ и основной прирост скорости достигается путём выбора подходящего решателя СЛАУ и его настройки. В частности, применение оптимизированной под конкретное аппаратное обеспечение библиотеки Basic Linear Algebra Subprograms (BLAS) обеспечивает 1,5–3-кратный прирост производительности при прочих равных условиях. Для рассмотренных выше задач время работы собственно автогенерированного кода, вычисляющего значения Якобиана и вектора невязок, находится в пределах 0,5…5% от общего времени счета задачи. Помимо этого, вычисление структуры Якобиана также происходит во время работы исполняемого кода, в фазе инициализации. Время, затраченное на эту работу, также не превышает 1% от общего времени счета и находится в диапазоне 0,1…2 сек. для задач с общим временем счета 30…60 мин.

Основным недостатком полностью неявных схем решения сеточных задач с привлечением решателей СЛАУ является большое потребление памяти, что является одним из факторов, сдерживающих применение совместных методов решения сложных задач, в частности, трёхмерных. При этом большое значение имеет выбор типа решателя СЛАУ. Как показывает практика, потребность в памяти у решателей, реализующих прямые методы, в 3-4 раза выше, чем у итеративных. Для примера, пиковое потребление памяти для задачи 6 на сетке 63×96 и четырёх неизвестных полях с использованием 32-битной версии SuperLU составляет порядка 400 Мбайт.

Для ряда тестовых задач вида 2 также проведены прогоны параллельного варианта кода (с использованием решателя PETSc), которые продемонстрировали близкое к линейному масштабирование производительности численного кода с ростом числа задействованных процессоров вплоть до 12…16 (в зависимости от размера сетки).

5. Выводы

Разработанный автором пакет «Finita» позволяет значительно упростить процесс разработки программ, реализующих численные алгоритмы решения сложных систем нелинейных дифференциальных уравнений сеточными методами. Ориентация пакета на использование элементов аналитических вычислений и автоматической генерации кода уменьшает объём ручного кодирования в 14 и более раз, причём этот показатель демонстрирует тенденцию к увеличению с ростом сложности решаемой проблемы. При применении пакета пользовательский код изолирован от деталей реализации схемы решения, вследствие чего смена решателя СЛАУ и даже переход с последовательной схемы решения на параллельную осуществляется без внесения изменений во входное задание для пакета и управляющий модуль.

Показано, что при совместном решении систем дифференциальных уравнений исходный код характеризуется большой громоздкостью и избыточностью, и это резко увеличивает его объём при усложнении задачи и трудозатраты на ручное написание и отладку численного кода. Применение пакета позволяет удержать объём работы, выполняемой пользователем на этапе кодирования задачи, в приемлемых рамках и переложить большую её часть на сам пакет.

Работа выполнена при частичной финансовой поддержке Программы Президиума РАН «Параллельные вычисления на многопроцессорных вычислительных системах» (раздел «Технологии параллельного программирования в задачах механики деформируемого твердого тела и механики жидкости»).

Литература

1. http://www.freefem.org
2. http://www.opencfd.co.uk/openfoam
3. http://www.pdesolutions.com
4. *Lan C.W., Chen F.C.* A finite volume method for solute segregation in directional solidification and comparison with a finite element method // Computational methods application in mechanical engineering. – 1996. - № 131. - P. 191-207.
5. http://www.ruby-lang.org
6. http://www.eiffel.com
7. http://smarteiffel.loria.fr
8. http://crd.lbl.gov/~xiaoye/SuperLU
9. http://www.cise.ufl.edu/research/sparse/umfpack
10. http://acts.nersc.gov/petsc
11. *M. Benzi.* Preconditioning techniques for large linear systems: a survey // J. of Comput. Physics. - 2002. - № 182. – P. 418-477.
12. *Gille P. ,1, S. Scharl, G. Mueller.* A generalized description of solute distribution in melt growth by the submerged heater method // J. of Crystal Growth. – 1995. – № 148. – P. 183-188.

Приложение 1

Данное приложение содержит пример исходной системы уравнений, описывающей течение и теплоперенос в несжимаемой жидкости (граничные условия опущены) и выдержку из входного задания для пакета, реализующего совместную схему её решения, при дискретизации уравнений методом конечных разностей.

Исходная система уравнений:

$$\begin{cases} \dfrac{\partial\varphi}{\partial x}\dfrac{\partial\psi}{\partial y} - \dfrac{\partial\varphi}{\partial y}\dfrac{\partial\psi}{\partial x} = \Delta\varphi - Gr\dfrac{\partial T}{\partial x}; \\ \varphi + \Delta\psi = 0; \\ \dfrac{\partial T}{\partial x}\dfrac{\partial\psi}{\partial y} - \dfrac{\partial T}{\partial y}\dfrac{\partial\psi}{\partial x} = \dfrac{1}{\mathrm{Pr}}\Delta T. \end{cases}$$

Выдержка из входного задания:

```
Gr = const :Gr, 1E5
Pr = const :Pr, 1
T = field :T
Psi = field :psi
Phi = field :phi
Cavity = Problem.new :cavity
Sys = System.new :Cavity; Cavity.systems << Sys
Sys.equations << [
    eqn(di(T)*dj(Psi)-dj(T)*di(Psi) - (d2i(T) + d2j(T))/Pr,
        T, domain.inner.area, false),
    eqn(di(Phi)*dj(Psi) - dj(Phi)*di(Psi) - (d2i(Phi) + d2j(Phi)) + Gr*di(T),
        Phi, domain.inner.area, false),
    eqn(Phi + d2i(Psi) + d2j(Psi), Psi, domain.inner.area, false)
]
```

Приложение 2

Фрагмент кода из управляющего Си-модуля, иллюстрирующий доступ к сеточным полям, определённым в соответствующем входном задании:

```
int i, j;
for(i = 1; i < grid_width-1; ++i) {
    for(j = 1; j < grid_height-1; ++j) {
        T(i,j) = (T1(i+1,j)+ T1(i-1,j)+ T1(i,j+1)+ T1(i,j-1))/4;
    }
}
```