УДК 532.529

РАЗРАБОТКА И РЕАЛИЗАЦИЯ АЛГОРИТМОВ ЧИСЛЕННОГО РЕШЕНИЯ ЗАДАЧ МЕХАНИКИ ЖИДКОСТИ И ГАЗА

К. Н. Волков[1]

Рассматриваются вопросы, связанные с разработкой программного обеспечения, предназначенного для численного решения задач механики жидкости и газа, и его программной реализацией на основе объектно-ориентированного подхода. Формулируются требования к системному и функциональному наполнению программного обеспечения. Обсуждаются инструментальный характер разработанных программных средств, структура организации классов и подход к их практической реализации.

1. Введение. К настоящему времени накоплен обширный фонд вычислительных алгоритмов, предназначенных для численного моделирования течений жидкости и газа, описываемых уравнениями Эйлера или Навье–Стокса.

Универсальные коммерческие CFD-пакеты (например, Fluent, Ansys CFX, Star-CD и др.) представляют собой сложные многокомпонентные системы, имеющие трехступенчатую структуру: сеточный генератор, расчетный модуль и графический интерпретатор результатов. В пакеты включаются широкие наборы математических моделей управляющих физических процессов, конечно-разностных схем, методов решения систем разностных уравнений, из элементов которых конструируется решение той или иной задачи. Многие пакеты допускают эксплуатацию не только на персональных компьютерах, но и на многопроцессорных вычислительных системах.

Широкое распространение вычислительных пакетов создает иллюзию того, что они позволяют решить любые задачи. На самом деле, используемые в них каталоги математических моделей и конечно-разностных схем далеки от совершенства, поскольку научные изыскания по ним не закончены. Приемлемость многих моделей для решения сложных задач и определение границ их применимости составляет предмет отдельного исследования (многие подходы тестируются на расчетах канонических течений, и их приемлемость в более широком диапазоне условий не очевидна). В существенной степени это относится к реализации современных подходов к моделированию турбулентных течений, например метода моделирования крупных вихрей.

В настоящей работе рассматриваются общие вопросы разработки программного обеспечения, предназначенного для численного моделирования течений вязкого сжимаемого газа, описываемых полными или фильтрованными уравнениями Навье–Стокса. Формулируется ряд требований к вычислительной процедуре, претендующей на расчет нестационарных течений вязкого сжимаемого газа в сложных пространственных областях, а также приводится ее системное и функциональное наполнение. Основные концептуальные положения, принятые при реализации программного кода, иллюстрируются парадигмами объектно-ориентированного программирования. Обсуждаются инструментальный характер разработанных программных средств, структура организации классов и их практическая реализация.

2. Системное и функциональное наполнение. Численное решение задач механики жидкости и газа подразумевает построение расчетной сетки, подготовку исходных данных, дискретизацию модельных уравнений, решение системы разностных уравнений и обработку полученных результатов. Данная схема не является простой последовательностью перечисленных действий, но обладает и обратной связью.

Системное и функциональное наполнение программного обеспечения, претендующего на расчет нестационарных течений вязкого сжимаемого газа в сложных пространственных областях, поясняет рис. 1.

2.1. Расчетные сетки. Построение сетки относится к одному из ключевых моментов численного эксперимента [1 – 3]. Сетки различаются идеологией построения и методами решения модельных уравнений (регулярные, блочные, неструктурированные, гибридные).

2.1.1. Регулярные сетки. При решении задач газовой динамики широко применяются регулярные сетки (структурированные сетки с четырехугольными ячейками на поверхности и шестигранными в пространстве). Регулярность заключается в том, что сетка представляет собой упорядоченную по

[1] Балтийский государственный технический университет “Военмех” им. Д.Ф. Устинова, физико-механический факультет, 1-я Красноармейская ул., д. 1, 190005, Санкт-Петербург; e-mail: dsci@mail.ru

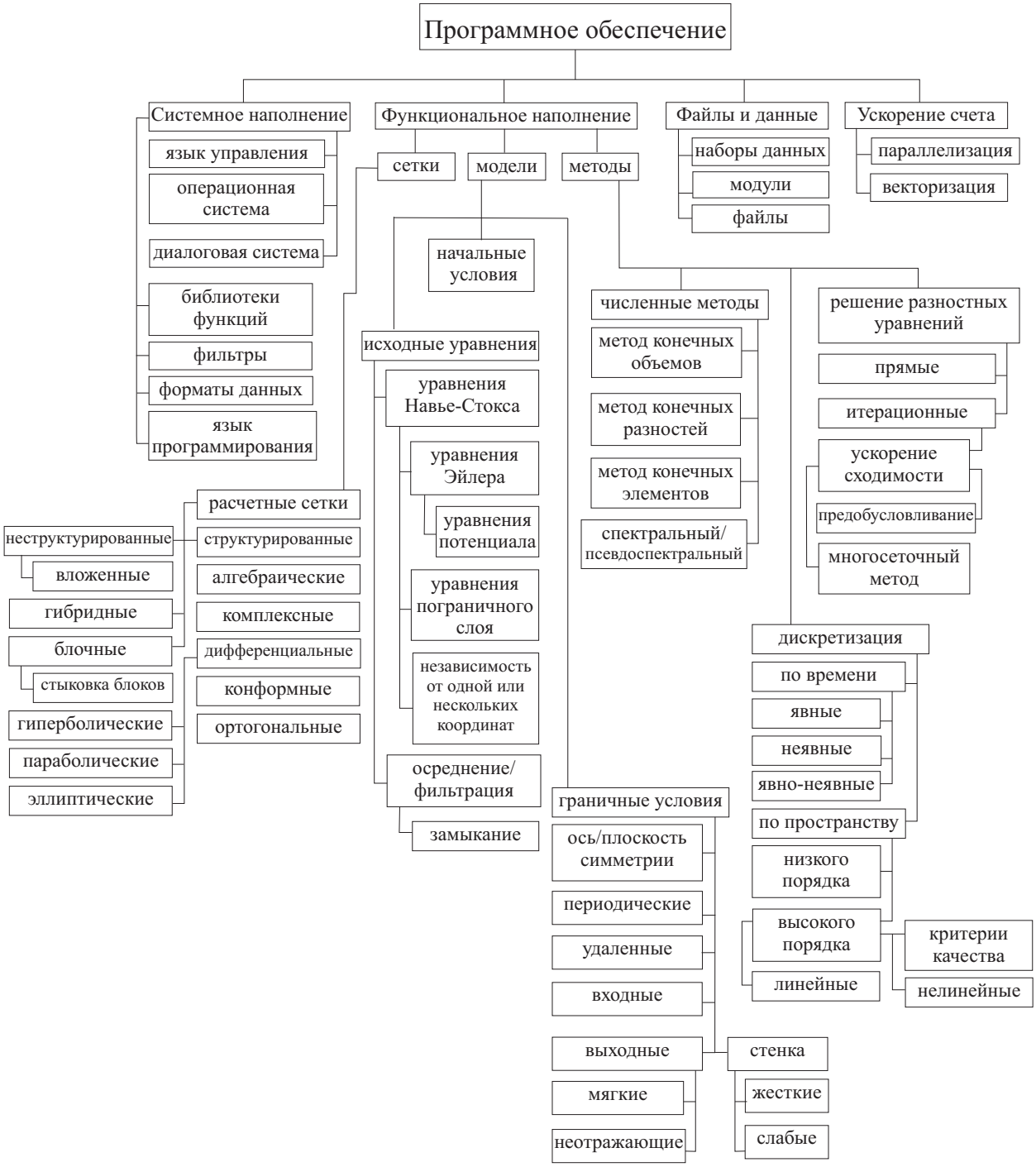

Рис. 1. Системное и функциональное наполнение программного обеспечения, предназначенного для решения задач механики жидкости и газа

определенным правилам структуру данных с выраженными сеточными направлениями. Для структурированных сеток (structured mesh) сравнительно легко реализуются вычислительные алгоритмы на основе метода конечных разностей или метода конечных объемов и современных монотонных методов высокого порядка точности. Регулярные сетки позволяют использовать методы расщепления для решения многомерных задач и реализовать сравнительно простую векторизацию программы.

Для построения регулярной сетки в сложной области применяется преобразование координат общего вида, основная цель которого состоит в получении равномерной сетки в вычислительном пространстве, представляющем собой прямоугольник [2, 3]. При этом физические границы расчетной области совпа-

дают с координатными линиями в вычислительном пространстве. В уравнениях, записанных в криволинейных координатах, появляются дополнительные члены (параметры преобразования), определяющие отображение физической области на пространство обобщенных координат. Параметры преобразования (компоненты метрического тензора) имеют форму производных, требующих дискретизации.

Дополнительные преимущества дают ортогональные и конформные сетки, поскольку преобразованные уравнения упрощаются. В случае ортогональной сетки компоненты метрического тензора, находящиеся не на главной диагонали, равняются нулю. Использование конформного преобразования позволяет сохранить такую же структуру модельных уравнений, записанных в вычислительном пространстве, как и в физической области (параметры преобразования равняются либо нулю, либо единице).

Методы построения регулярных сеток делятся на алгебраические, дифференциальные и методы с использованием теории функций комплексной переменной [1 – 3].

Основная идея алгебраических методов состоит в использовании интерполяции граничных данных для расчета внутренних узлов сетки. Контроль за размещением узлов сетки осуществляется с помощью функции растяжения (stretching function). Широкое применение находят метод двух границ (two-boundary method), метод многих поверхностей (multi-surface method) и метод трансфинитной интерполяции (transfinite interpolation).

Дифференциальные сетки строятся на основе решения дифференциальных уравнений в частных производных [2, 3]. В зависимости от типа решаемых уравнений выделяют гиперболические, параболические и эллиптические сетки.

Гиперболические уравнения решаются эффективными маршевыми методами и требуют сравнительно небольших затрат времени. Гиперболические уравнения лишены механизма диффузионного сглаживания, поэтому разрывы в начальных данных сохраняются во всей расчетной области, что неприемлемо при построении сетки. Граничные условия в физической области используются не в полной мере, что оказывается существенным для ряда задач.

Параболическая сетка также строится маршевым методом, что обеспечивает вычислительную эффективность. Вместе с тем, параболические уравнения имеют многие свойства эллиптических уравнений, в частности, механизм диффузионного сглаживания, что гарантирует отсутствие изломов и разрывов в решении. К их недостаткам относится невозможность использования всех граничных условий в физической области, поскольку для параболических уравнений в маршевом направлении граничные условия не ставятся.

Эллиптические уравнения (обычно используются уравнения типа Пуассона) позволяют получить гладкое решение и учесть граничные условия на всех границах физической области. В силу принципа максимума для эллиптических уравнений, обеспечивается взаимно-однозначность отображений физической и вычислительной областей. При этом реализуется достаточно гибкий механизм контроля за размещением внутренних узлов сетки [2, 3]. Недостатки подхода связаны с использованием итерационных методов решения системы разностных уравнений, что приводит к увеличению времени, необходимого для построения сетки.

Для построения конформных сеток применяется преобразование Шварца–Кристоффеля [3]. К недостаткам конформных сеток относятся: ограничение на размерность сетки (двумерные), нетривиальность выбора последовательности конформных отображений для области сложной конфигурации, сложность построения обратного преобразования (из вычислительного пространства в физическое).

Ортогональные сетки строятся либо методом конформных отображений с последующим растяжением узлов (конформность нарушается, но требование ортогональности соблюдается), либо дифференциальными методами [1] (во многих случаях они приводят к сильной деформации сетки в физической области). Уравнения для построения локально ортогональной сетки получаются на основе принципов вариационного исчисления.

С точки зрения вычислительной эффективности предпочтительнее применение алгебраических методов, которые позволяют обеспечить условие локальной ортогональности сетки и ее быструю перестройку. Взаимно-однозначность отображений физической и вычислительной областей обеспечивают методы последовательных конформных отображений и дифференциальный метод на основе решения эллиптических уравнений в частных производных [1]. В остальных случаях взаимно-однозначность отображений не гарантируется, поэтому требуется интерактивный процесс генерации сетки с использованием графических средств.

2.1.2. Блочные сетки. Для построения сетки в сложной области производится разделение поля течения на подобласти, в каждой из которых генерируется своя сетка. Выделяют метод многоблочных структур (multi-block structuring или zonal block) и метод иерархических блочных структур (embedding

grid). Сетки в разных блоках могут иметь различные топологические характеристики [4] (допускается также решение различных модельных уравнений в разных блоках).

В методе многоблочных структур физическая область разбивается на несколько зон, или блоков. В соответствии с граничными условиями для каждой подобласти, для каждого блока строится своя сетка (zonal grid). Различают два подхода к организации обмена данными между соседними блоками: сетки из разных блоков стыкуются по поверхности раздела физической области на зоны (patched grid) или сетки из соседних блоков пересекаются между собой (overlapped grid). В случае совпадения границ блоков при переходе от одной зоны к другой сохраняется консервативность разностной схемы и не требуется интерполяция между соседними блоками (однако требование точного совпадения границ блоков накладывает некоторые дополнительные условия на сетку). В случае пересечения границ блоков каждый блок допускает перемещение относительно других блоков, а при переходе от одного блока к другому консервативность схемы не гарантируется и требуется интерполяция искомых функций в пересекающихся областях [4].

Метод иерархических блочных структур подразумевает иерархическую вложенность блоков сетки друг в друга. Нижестоящие по иерархии сетки погружаются в вышестоящие. Реализация подхода требует, чтобы подобласти не были разъединены и включали одна другую полностью или частично.

2.1.3. Неструктурированные сетки. Особенностью неструктурированных сеток (unstructured mesh) является произвольное расположение узлов сетки в физической области. Произвольность расположения узлов понимается в том смысле, что отсутствуют выраженные сеточные направления и нет структуры сетки, подобной регулярным сеткам. Число ячеек, содержащих каждый конкретный узел, изменяется от узла к узлу. Узлы сетки объединяются в многоугольники (двумерный случай) или многогранники (трехмерный случай). Как правило, на плоскости используются треугольные и четырехугольные ячейки, а в пространстве — тетраэдры и призмы. Для дискретизации уравнений Навье–Стокса применяется метод конечных элементов или метод конечных объемов.

Для неструктурированных сеток необходимо хранить информацию о ячейках, гранях, узлах и ребрах, а в некоторых случаях — расстояние от центра контрольного объема до стенки.

В качестве основных структур данных рассматриваются множества (узлы, внутренние и граничные грани, ячейки различного типа), данные, связанные с множествами (координаты узлов сетки, объемы ячеек, нормали к граням), указатели между множествами (список узлов, формирующих ячейки того или иного типа, список граничных узлов), операции над множествами (цикл по множеству ячеек с использованием указателей на список узлов, формирующих ячейки, для расчета невязки в узлах сетки, цикл по ненулевым элементам разреженной матрицы).

Неструктурированные сетки требуют примерно в 5–6 раз больше ячеек, чем регулярные сетки, а для разрешения тонких пограничных слоев — достаточно мелких ячеек вблизи стенки, что ведет к увеличению их общего количества. Тем не менее процесс генерации неструктурированной сетки легче формализуется и автоматизируется по сравнению с регулярными сетками, занимая меньше времени, что обусловливает их широкое распространение на практике. При этом сравнительно легко реализуются локальные сгущения и адаптация сетки к решению [5].

2.1.4. Гибридные сетки. Гибридная сетка (hybrid mesh) предполагает объединение регулярных и неструктурированных сеток в различных подобластях расчетной области, позволяя сочетать достоинства и снизить влияние недостатков, присущих каждому типу сеток.

Гибридные сетки широко используются при решении задач механики жидкости и газа. Имеются многочисленные публикации, посвященные разработке и реализации вычислительных алгоритмов на таких сетках [5, 6], а также избранным вопросам дискретизации уравнений Эйлера и Навье–Стокса [7, 8]. В отличие от хорошо разработанных технологий метода конечных элементов, конечно-объемные технологии на неструктурированных сетках характеризуются отсутствием единых принципов, позволяющих провести дискретизацию конвективных и диффузионных потоков, источниковых членов, а также учет граничных условий. Достаточно часто способы дискретизации, имеющие различные характеристики, объединяются.

2.2. Основные уравнения. Стационарные или нестационарные течения вязкого сжимаемого газа описываются уравнениями Навье–Стокса, а течения идеальной среды — уравнениями Эйлера, которые для безвихревых течений сводятся к уравнениям потенциала.

В зависимости от размерности решаемой задачи используется трехмерная формулировка исходных уравнений или их сокращенный вариант — свойство независимости от одной или нескольких пространственных координат (плоские или осесимметричные течения). Моделирование течений с преимущественным направлением развития потока (течения в пограничных слоях или струях) проводится на основе

параболизованных уравнений Навье–Стокса.

Для расчетов турбулентных течений используется осреднение или фильтрация уравнений Навье–Стокса с последующим замыканием полученных уравнений. Прямое численное моделирование и моделирование крупных вихрей являются принципиально трехмерными подходами, поэтому использование двумерной формулировки задачи рассматривается как полезное с точки зрения сокращения затрат машинного времени, но необоснованное допущение [6].

2.3. Начальные условия. При моделировании стационарных течений начальные условия не играют роли. Тем не менее их удачное задание позволяет сократить затраты машинного времени на получение статистически стационарного решения.

2.4. Граничные условия. При решении полных уравнений Навье–Стокса требуется задание граничных условий на всех границах расчетной области.

На границе, через которую жидкость поступает в расчетную область, обычно задается распределение скорости, направление потока, распределения полного давления и полной температуры (или числа Маха), а также распределения характеристик турбулентности.

При решении задачи в неограниченной области из-за отсутствия точных граничных условий, заменяющих условия на бесконечности для исходной задачи с неограниченной областью, постановка граничных условий реализуется приближенным способом.

При расчете стационарных течений на выходной границе, через которую жидкость покидает расчетную область, обычно выставляются мягкие граничные условия, выражающие собой условие о равенстве нулю производной по нормали к границе.

При моделировании нестационарных дозвуковых течений возмущения, дойдя до внешней границы, частично отражаются от них, искажая решение внутри расчетной области. Широкое применение находят неотражающие граничные условия.

Для скорости на стенке используются граничные условия прилипания и непротекания (жесткие граничные условия). Для температуры задается температура стенки. При постановке граничных условий для характеристик турбулентности на стенке используется метод пристеночных функций [4, 7].

Использование слабых граничных условий позволяет избежать применения пристеночных функций и связанных с ними проблем. Влияние стенки на поток учитывается в виде сеточных напряжений сдвига и дополнительной сеточной генерации турбулентности за счет отличия профиля касательной скорости от логарифмического распределения около стенки [7, 8].

На удаленных границах расчетной области задаются условия скольжения или условия невозмущенного течения.

Периодические граничные условия выставляются для всех искомых функций на противоположных границах расчетной области при расчете течений в области, обладающей свойством симметрии.

2.5. Методы дискретизации. Для дискретизации уравнений Навье–Стокса используются: метод конечных разностей, метод конечных объемов, метод конечных элементов и спектральные методы.

Метод конечных разностей (finite difference method) основан на замене производных, входящих в исходные уравнения, их дискретными (разностными) аналогами. Его достоинствами являются эффективность и простота реализации, а также наглядность процедуры дискретизации, дающая возможность построения схем высокого порядка точности (эти достоинства реализуются при использовании структурированных сеток). Расчеты проводятся либо на совмещенной, либо на разнесенной сетке (staggered grid), когда скорость и давление рассчитываются в смежных узлах [9].

В методе конечных объемов (finite volume method) используется интегральная формулировка законов сохранения массы, импульса и энергии. Дискретный аналог балансовых соотношений, записанных для контрольного объема, получается суммированием по всем его граням потоков массы, импульса и энергии, вычисленных по каким-либо квадратурным формулам. Метод конечных объемов пригоден для дискретизации уравнений сохранения как на структурированных, так и на неструктурированных сетках с различной формой ячеек. В качестве контрольного объема выбирается контрольный объем, совпадающий с ячейкой, или контрольный объем, центрированный относительно узла [6].

Метод конечных элементов (finite element method) опирается на вариационную задачу о минимуме ошибки аппроксимации искомого решения базисными функциями, а не на исходные физические уравнения. Метод конечных элементов получил широкое распространение в механике деформируемого твердого тела. Отсутствие преимуществ перед методом конечных объемов и трудности обеспечения необходимой точности описания тонких пограничных слоев служат причинами низкой популярности метода конечных элементов при решении задач газовой динамики и теплообмена.

Для задач с достаточно гладким решением и мягкими граничными условиями для дискретизации

уравнений Навье–Стокса находят применение спектральные методы (spectral method), которые позволяют определить значения искомых функций более точно, чем локальные методы [3]. Важная роль граничных условий при построении спектральных методов привела к использованию псевдоспектральных подходов (pseudo-spectral method). Использование полиномов Чебышева для определения точек коллокации приводит к мелкой сетке вблизи границ и сравнительно грубой сетке внутри расчетной области, что удобно для моделирования течений при больших числах Рейнольдса.

Преимущество спектральных методов состоит в том, что высокая пространственная точность получается при сравнительно небольшом числе точек коллокации. Их основной недостаток связан с ограничениями на шаг по времени при расчете стационарных или слабо меняющихся со временем течений [3]. При моделировании нестационарных течений, в которых для достижения необходимой точности требуются малые шаги по времени, спектральные методы становятся конкурентоспособными с конечно-разностными и конечно-объемными методами (особенно для областей регулярной формы).

2.6. Дискретизация по времени. С точки зрения дискретизации по времени разностные схемы делятся на явные, неявные и смешанные явно-неявные [9].

Явные схемы (explicit scheme) лучше согласованы с конечной скоростью распространения возмущений, характерной для гиперболических уравнений, ограничивая их перенос одним шагом сетки за один шаг по времени. Для дискретизации параболических уравнений, которые характеризуются мгновенной скоростью распространения возмущений, используются неявные схемы (implicit scheme), снимающие жесткие ограничения на шаг интегрирования по времени. В явно-неявной схеме (explicit/implicit scheme) для дискретизации конвективных потоков применяется явная схема, а для дискретизации диффузионных потоков — неявная схема.

Одна из проблем численного решения уравнений Навье–Стокса и построения разностных схем на неструктурированных сетках связана с необходимостью обеспечения положительности искомых функций. Условие положительности выполняется для схем Рунге–Кутта различного порядка [10] (при условии, что выполняется условие Куранта–Фридрихса–Леви и все коэффициенты схемы являются неотрицательными).

Рост мощности современных многопроцессорных вычислительных систем делает оправданным использование простых явных конечно-разностных схем [11].

2.7. Дискретизация по пространству. Различные версии метода конечных объемов различаются способом вычисления потоков [9, 10, 12, 13].

Численные схемы, используемые для дискретизации конвективных потоков, должны сохранять монотонность и сходиться к физически корректному решению. Условие сохранения монотонности решения связано с идеей невозможности появления ложных максимумов или минимумов (нефизических осцилляций, развивающихся со временем).

Для получения монотонного решения применяется схема Годунова [1], предполагающая кусочно-постоянное распределение параметров течения на нижнем временно́м слое. В ней используется точное решение задачи о распаде произвольного разрыва (задача Римана). Для экономии машинных ресурсов применяются также приближенные подходы [10, 13] (среди наиболее популярных следует отметить методы Роэ и Ошера).

В силу теоремы Годунова не существует монотонных линейных разностных схем с порядком аппроксимации по пространству выше первого [3]. Принцип минимальных значений производных, используемый в схеме Колгана, позволяет повысить порядок схемы Годунова до второго по всем направлениям, за исключением направления интегрирования.

Прогресс в направлении улучшения диссипативных и дисперсионных свойств разностных схем, используемых для дискретизации конвективных потоков, связан с разработкой и реализацией разностных схем повышенной разрешающей способности (high resolution scheme). Общим во всех методах подобного класса является использование разнообразных монотонизирующих ограничителей потоков с переключателями, зависящими от локальных свойств решения [10, 12, 13].

Повышение точности конечно-разностных схем без потери их строгого теоретического обоснования достигается путем замены условия сохранения монотонности на условие уменьшения полной вариации [3] (Total Variation Diminishing, TVD). Для получения разностных схем, удовлетворяющих условию TVD, вводится ограничитель потока, зависящий от градиентов искомой функции. В зависимости от структуры ограничителя разностные схемы условно разделяются на линейные и нелинейные [12].

Для обеспечения ограниченности численного решения разностные схемы должны удовлетворять критерию конвективной ограниченности (Convection Boundedness Criterion, CBC). Критерий CBC представляет собой необходимое и достаточное условие для обеспечения ограниченности численного решения в

том случае, когда для дискретизации конвективных потоков на грани контрольного объема используется не более трех узлов против потока. Он находит подтверждение для неявных расчетов стационарных течений, но не гарантирует получение сходящегося решения [12].

Для того чтобы гарантировать получение сходящегося решения, используется универсальный ограничитель потока (Universal Limiter, ULTIMATE). Критерий ULTIMATE находит подтверждение для явных расчетов нестационарных течений и сводится к критерию CBC при числах Куранта [12].

Условия, выражаемые критериями CBC и ULTIMATE, оказываются более мягкими, чем условие TVD, но, как показывают многочисленные расчеты, позволяют получить монотонное и сходящееся решение [12].

Среди различных монотонизированных разностных схем повышенного порядка аппроксимации наиболее оптимальное соотношение между точностью, экономичностью, простотой и универсальностью имеет метод кусочно-параболической реконструкции (Piecewise Parabolic Method, PPM) [13], который считается наиболее перспективным для прямого численного моделирования газодинамической неустойчивости и турбулентности на современных многопроцессорных системах.

В то время как дискретизация конвективных потоков проводится при помощи разностных схем повышенной разрешающей способности, способ дискретизации диффузионных потоков влияет, скорее, на техническую сторону реализации подхода, а соответствующие разностные схемы не обязательно должны иметь повышенный порядок. При расчете турбулентных течений более низкий порядок дискретизации вязких членов интерпретируется как неточность в представлении сил вязкости, что вполне допустимо, когда турбулентная вязкость рассчитывается при помощи некоторой приближенной модели.

2.8. Решение разностных уравнений. Решение системы разностных уравнений представляет собой один из наиболее важных и доминирующих моментов вычислительной процедуры с точки зрения затрат машинных ресурсов.

Прямые методы (например, метод исключения Гаусса) предъявляют жесткие требования к быстродействию и памяти. Время вычислений оценивается как $O(n^3)$, а память, необходимая для хранения данных, как $O(n^2)$, где n — число неизвестных. Для итерационных методов время решения зависит от качества начального приближения, а требования к памяти значительно мягче и оцениваются как $O(n)$. Например, для методов Якоби и Гаусса–Зейделя время счета оценивается как $O(k^2)$, где k — количество итераций.

Для увеличения скорости сходимости итерационного процесса используются различные методы предобусловливания [3, 9]. Их применение приводит к удорожанию вычислительной процедуры приблизительно на 50 % [3].

2.9. Ускорение вычислений. При решении полных уравнений Навье–Стокса достаточно важной становится проблема сокращения затрат машинного времени. Данная проблема рассматривается с двух позиций — с точки зрения ускорения итерационного процесса и с точки зрения ускорения вычислений за счет эффективного использования ресурсов современных вычислительных систем.

Прогресс в ускорении сходимости итерационных процессов требует реализации и внедрения специальных вычислительных алгоритмов, обладающих достаточно сложной логикой счета [3]. Ускорение счета предполагает привлечение новых дорогостоящих вычислительных ресурсов и программного обеспечения, а также их своевременное обновление.

Среди итерационных методов решения систем разностных уравнений наибольший эффект ускорения сходимости позволяет получить многосеточный метод (multigrid method) [6, 14], а наиболее привлекательный способ ускорения счета состоит в использовании параллелизации или векторизации процесса вычислений [11].

Многосеточный метод сравнительно легко реализуется на структурированных сетках [14]. В случае неструктурированных сеток требуется построение последовательности неструктурированных сеток, позволяющих провести дискретизацию на различных уровнях [6, 15]. Для их построения используются метод схлопывающихся узлов (vertex collapse method) и метод схлопывающихся граней (edge collapse method).

2.10. Параллелизация. Для решения задач механики жидкости и газа требуется высокое быстродействие, а также обработка и хранение большого объема информации, что предъявляет повышенные требования к оперативной памяти. Такие ресурсы имеются в распоряжении высокопроизводительных вычислительных систем, или суперкомпьютеров (high performance computing).

Зависимость производительности вычислительных систем от времени их ввода в эксплуатацию приводится на рис. 2, взятом из работы [16]. Производительность современных суперкомпьютеров превышает 10 Тфлопс, а динамика ее роста является довольно оптимистичной. Сравнительно дешевые рабочие станции и персональные компьютеры достигли скорости в несколько сотен мегафлопс (их производительность

увеличивается по зависимости, близкой к линейной в логарифмических координатах).

Перспективы численного моделирования течений вязкого сжимаемого газа со стороны совершенствования вычислительных платформ выглядят достаточно привлекательными.

Подход к численному решению задач механики жидкости и газа при помощи современных вычислительных технологий и методов параллельного программирования основан на геометрической декомпозиции расчетной области на подобласти, количество которых равняется числу процессоров, расчете каждым процессором своей подобласти и обмене данными между ними на каждом шаге по времени. Связь подобластей обычно осуществляется при помощи введения фиктивных ячеек (их количество зависит от шаблона разностной схемы), которые находятся за границами подобластей и не обрабатываются кодом [11]. Показатели производительности зависят от метода декомпозиции, способа распределения данных по процессорам и реализации численных методов, применяемых для решения подзадач.

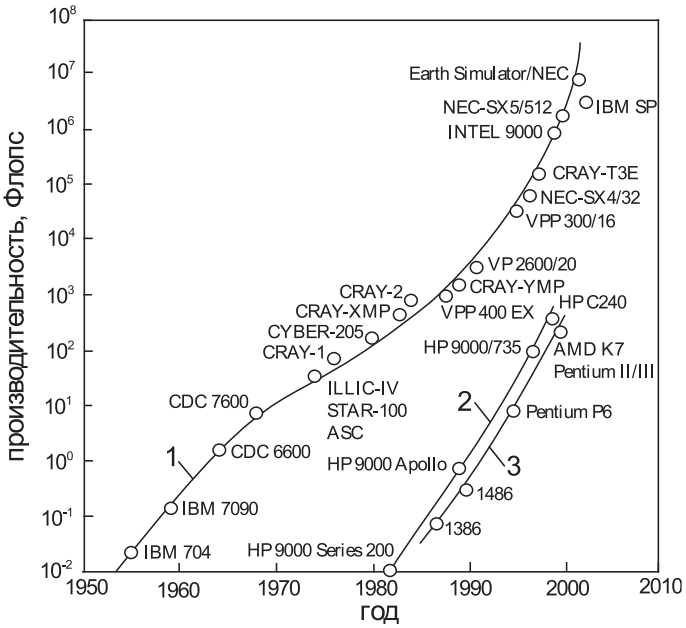

Рис. 2. Динамика увеличения производительности вычислительных систем (1 — суперкомпьютеры, 2 — рабочие станции, 3 — персональные компьютеры)

2.11. Файлы и данные. Файлы и данные, необходимые для численного решения задачи тем или иным способом, показаны на рис. 3 (считается, что программа работает в пакетном режиме, графический интерфейс не используется). Файл сценария (файл исходных данных) содержит информацию, необходимую для решения задачи, в частности, имена файлов с геометрией области, тип граничных условий, параметры разностной схемы, теплофизические характеристики жидкости и другие параметры. Для параметров, значения которых не определяются в файле сценария, используются значения по умолчанию.

Бинарные файлы сетки и поля течения содержат координаты узлов расчетной сетки, начальные распределения искомых функций, а также форму всех ячеек сетки, тип выставляемых граничных условий и ряд других параметров. Сетка хранится отдельно от поля течения. При этом для хранения каждого уровня неструктурированной сетки используется отдельный файл (уровень 1 соответствует сетке наилучшей разрешающей способности). Начальное и конечное поле течения хранятся либо в отдельных файлах (файл начального поля течения сохраняется неизменным), либо в одном и том же файле (файл начального поля течения заменяется на новый).

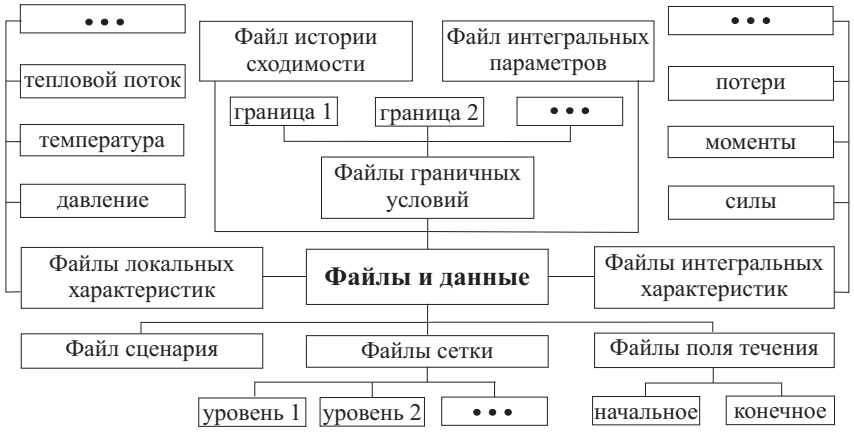

Рис. 3. Файлы и данные

Файлы граничных условий хранят профили функций, задаваемые на границах расчетной области. Для проверки сходимости итерационного процесса требуется файл, содержащий значения невязок. Для обработки результатов расчетов значения локальных и интегральных параметров потока сохраняются в файлах, имена которых указываются в файле сценария.

2.12. Программная реализация. Развитие программного обеспечения осуществляется, в основном, не за счет замены имеющихся модулей на их более совершенные версии, а за счет расширения и

включения в программу новых модулей, отражающих различные решения, принимаемые в ходе вычислительного эксперимента. Ключом к эффективному программированию вычислительных задач является рациональный подход к организации создания и модификации программного обеспечения.

Традиционная его организация в виде библиотек подпрограмм соответствует технологии разработки, опирающейся на идеи процедурного программирования. Принципы процедурного программирования отражают концепцию построения программного обеспечения на основе выделения структурируемых и самостоятельно значимых подпрограмм, выполняющих некоторую последовательность операций над данными и решающих независимые подзадачи. К его недостаткам относятся: необходимость унификации внутренних форматов данных, которые используются импортируемыми библиотечными модулями, в связи с чем реализуется избыточная поддержка нескольких эквивалентных представлений данных; низкая модифицируемость, наглядность и выразительность процедурных средств программирования; отсутствие внутренней структуризации программы несмотря на процедурную завершенность отдельных подпрограмм.

Мощную и гибкую технологию разработки и развития программного обеспечения предоставляют средства объектно-ориентированного программирования (Object-Oriented Programming, OOP). Его отличительные черты состоят в прозрачности и возможности доступа к деталям реализации конкретного метода и алгоритма, математической ясности описания нового метода для разработчика и пользователя, открытости и возможности дополнения библиотеки новыми процедурами, а также простоте использования разработанных методов.

Инкапсуляция методов в классах математических объектов способствует повышению наглядности применяемых численных методов и регламентированию корректной дисциплины работы с данными объектов без нарушения их целостности (особенно это относится к динамически размещаемым данным математических объектов, в частности, к векторным и матричным объектам).

Иерархическое упорядочивание позволяет выделить общие свойства объектов в базовых классах, а в производных классах дополнять переменные состояния и поведение объектов новыми свойствами. Некоторые из наследуемых свойств в объектах производных классов допускают переопределение, а указатели на объект производного класса — приведение к указателям на объект базового класса.

При обращении к виртуальным методам иерархии по указателям на объекты базового класса вызываются методы тех классов, на объекты которых они на самом деле указывают.

Наследование классов и полиморфизм методов, инкапсулируемых соответствующими классами, предоставляют возможность гибкой модернизации и развития математического обеспечения как с учетом проблемной ориентации, так и в соответствии с детальной классификацией объектов, выделенных по их математическим и вычислительным свойствам.

Объектно-ориентированный подход обеспечивает компромисс между необходимостью иметь надежное функционально полное и унифицированное алгоритмическое ядро и возможностью замещения универсальных работоспособных методов на эффективные алгоритмические версии, применимые только в частных случаях. Он широко применяется для создания текстовых редакторов, графических интерфейсов и библиотек вычислительной математики [17]. Меньше внимания уделяется вопросам использования объектно-ориентированных технологий при разработке программного обеспечения, предназначенного для численного решения задач в различных предметных областях. В области механики жидкости и газа объектно-ориентированный подход находит применение для разработки модели декомпозиции расчетной области [18, 19], конечно-элементного подхода к дискретизации законов сохранения [20 – 22], методов решения операторно-сеточных задач двумерной газовой динамики [23], системы подготовки данных, управления процессом счета и визуализации результатов численного моделирования [24, 25].

Выбор системы объектов, специфичных для данной предметной области, и принципы, лежащие в основе построения иерархии классов, определяют характеристики и показатели качества разрабатываемого программного обеспечения. Несмотря на имеющиеся примеры использования объектно-ориентированного подхода для реализации алгоритмов решения задач механики жидкости и газа, общепринятые шаблоны построения системы объектов и иерархии классов отсутствуют [26], а применяемые шаблоны зависят от субъективных представлений [18, 22, 26].

Полная реализация классов с использованием функций генерации и обработки исключительных ситуаций при задании некорректных исходных данных или при неправильном обращении к методам классов увеличивает размер исходного кода программы и предъявляет повышенные требования к производительности компьютера.

2.13. Средства реализации. Выбор средств реализации вычислительного алгоритма во многом определяет показатели производительности программного обеспечения, переносимость и возможности

его последующего расширения.

Среди наиболее часто используемых языков программирования следует отметить Fortran (процедурный подход) и C++ (объектно-ориентированный подход). Применение языка Fortran объясняется возможностью использования мощных библиотек функций, а C++ — его универсальностью.

Немаловажным представляется вопрос о переносимости программного обеспечения с одной платформы на другую (например, с Windows на Unix). В основном, это касается графического интерфейса. Проблем с переносимостью расчетных модулей, как правило, не возникает.

Проведение вычислений на современных параллельных программных системах в пакетном режиме или режиме удаленного терминала делает нерациональным создание программного обеспечения с графическим интерфейсом и развитыми средствами визуализации. Для этих целей используются специализированные коммерческие или свободно распространяемые пакеты и открытые форматы данных.

Для сопряжения с коммерческими пакетами требуется знание соответствующих форматов данных и разработка специальных фильтров, осуществляющих трансляцию из одного формата в другой (или запись результатов в нужном формате данных).

Программа решения уравнений Навье–Стокса реализует только расчетные функции (консольное приложение). Для задания входных данных и параметров используется встроенный макроязык. Такой подход позволяет добиться переносимости программных модулей и их независимости от конкретных особенностей операционной системы.

Для создания параллельных приложений широкое применение находит интерфейс межпроцессорного взаимодействия (Message Passing Interface, MPI). Технология MPI предназначена для поддержки работы параллельных процессов в терминах передачи сообщений. Интерфейс MPI предоставляет единый механизм взаимодействия ветвей внутри параллельного приложения независимо от машинной архитектуры (однопроцессорные/многопроцессорные системы с общей/распределенной памятью), взаимного расположения ветвей (на одном процессоре/на разных). Во многих случаях предпочтительно использовать программные средства более высокого уровня, чем MPI, облегчающие создание параллельного приложения.

Рис. 4. Функциональные подсистемы и их взаимодействие

2.14. Тестирование. Вычислительная процедура требует тестирования на широком диапазоне пространственных и временны́х масштабов путем решения совокупности задач модельного плана, для которых имеются надежные экспериментальные и расчетные данные.

3. Организация программного кода. Сложность математических моделей, необходимость построения многовариантных программных модулей, обслуживающих вычислительный эксперимент, а также существенный объем программного кода ставят вопрос организации его конфигурационных построений.

3.1. Функциональные подсистемы. Разработка универсальной программной среды подготовки данных, управления процессом расчета и визуализации результатов представляет достаточно сложную задачу.

В соответствии с функциональным назначением в системе выделяются три основные подсистемы: подсистема подготовки данных, подсистема управления процессом счета, подсистема визуализации полученных результатов. Управление подсистемами и их связь между собой осуществляются путем обмена сообщениями с управляющей программой (приложением). Подсистемы включают в себя следующие компоненты (их взаимодействие поясняет рис. 4).

1. Подсистема подготовки данных и создания твердотельной модели. Импорт твердотельной модели и сетки из коммерческих пакетов.

1.1. Задание геометрии расчетной области. Информация о геометрии хранится отдельно от поля течения.

1.2. Построение сетки.

1.3. Задание граничных условий. Указываются допустимые типы граничных условий и данные, необ-

ходимые для задания каждого из этих типов. Привязка граничных условий к отдельным сеточным линиям и формирование диалога для задания данных на каждой границе производится на основе коммерческих сеточных генераторов.

1.4. Задание начальных условий. Начальные условия задаются либо аналитически, либо с использованием интерполяции табличных данных, либо при помощи считывания из файла.

1.5. Задание параметров, относящихся к физической постановке задачи и реализации вычислительного алгоритма.

2. Подсистема управления процессом счета. Возможны несколько вариантов управления:

— не допускается управление во время счета (управление производится через систему подготовки данных путем задания продолжительности счета и шага вывода промежуточных результатов, которые накапливаются в файлах);

— частичное управление (предусматривается возможность наблюдения за изменением какого-либо параметра во времени);

— полное управление (приостановка счета и изменение управляющих параметров, сохранение текущего состояния с последующим рестартом, получение промежуточных результатов, выдача результатов при достижении заданного значения какого-либо параметра).

3. Подсистема визуализации результатов (графическое представление результатов в виде изолиний, векторных полей, графиков) и экспорт результатов в ряд форматов, поддерживаемых распространенными коммерческими пакетами.

С математической точки зрения расчет течения жидкости или газа представляет собой решение начально-краевой задачи некоторым численным методом (рис. 5).

Рис. 5. Структурно-функциональная схема решения задачи

Специфика задач механики жидкости и газа подразумевает зависимость результатов от геометрии расчетной области, что накладывает на подсистему задания геометрии особые условия (параметрическое описание геометрических объектов и возможность их легкого редактирования). Для задач разной размерности этапы создания геометрии и построения сетки различаются, что осложняет реализацию системы.

Создание твердотельной модели и построение сетки, задание типа граничных условий и обработка результатов производятся на локальном компьютере. Для этого используются коммерческие системы пре- и постпроцессорной обработки данных.

Расчетная сетка выгружается в текстовом формате. Препроцессор считывает файл сетки и формирует файл статистики, содержащий необходимую информацию о количестве и размерах векторов, которые выделяются в памяти на расчетном этапе.

Расчетный модуль считывает файл статистики, файл, содержащий граничные, начальные условия и физические параметры среды. Выполняется расчет, после которого расчетные данные выгружаются в файл для последующей обработки в системе визуализации. Память выделяется на этапе загрузки данных, и итерационный расчет выполняется без перевыделения памяти. В этом случае удается практически избежать дефрагментации оперативной памяти (указатели адресов в памяти располагаются упорядоченно). Достигаются эффективное использование памяти и надежность работы расчетного модуля.

3.2. Выбор объектов. В качестве основы объектной реализации программного обеспечения рас-

сматривается триада: математический объект, вычислительный алгоритм, проблема. Соответствующая классовая поддержка обеспечивает общность математических методов и программных средств решения вычислительной задачи. С каждым из выделенных понятий связывается математический класс, а с обобщенной постановкой — проблемный класс (рис. 6).

Математический объект представляет собой сущность, выражающую некоторую математическую категорию и составляющую объект вычислений. В качестве объектов, составляющих классовое ядро библиотеки, рассматриваются векторы, матрицы, а в более общем случае — геометрические примитивы, из которых строится твердотельная модель. Каждый математический объект обладает набором математических признаков, являющихся основой для классификационных построений. Однако сами по себе математические объекты не составляют вычислительной задачи и являются лишь инструментальным средством для ее постановки и решения.

При реализации математических классов вычислительно мощные методы реализуются на конкретной структуре данных, а их виртуальное использование переносится на абстрактный уровень. В этом случае затраты на вызовы виртуальных функций оказываются малыми по сравнению с объемом вычислительной работы, а сложно организованные вычислительные процессы реализуются на самых верхних уровнях классовой иерархии.

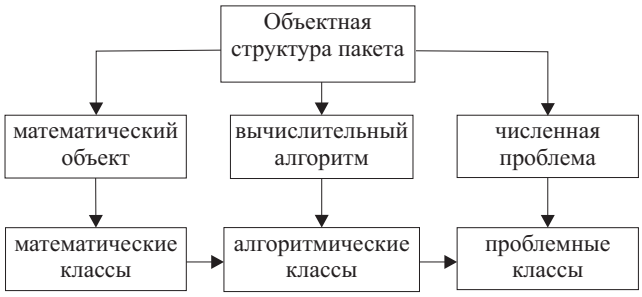

Рис. 6. Объектная структура пакета и иерархия классов

Математические классы выражают общие проблемно-инвариантные понятия, а математическая объектно-ориентированная библиотека представляет собой базовую инструментальную среду для разработки вычислительных приложений.

Под вычислительным алгоритмом понимаются методы вычислительной математики и вспомогательная информация, определяющая условия их алгоритмического использования. Каждый алгоритм предназначен для решения одной проблемы, хотя и может косвенно использоваться для решения других задач (для решения численных проблем одного класса могут использоваться разные подходы). Кроме параметров численного метода, алгоритмические объекты содержат информацию о точности решения и вычислительных ресурсах, имеющихся в наличии (они выражаются, например, в виде предельного числа итераций и времени счета). Данная информация определяет условия организации вычислительного процесса.

В качестве численной проблемы рассматривается задача вычислительной математики, представленная в унифицированной форме (например, проблема решения системы линейных или дифференциальных уравнений). Организация проблемных классов обеспечивает желаемую общность программной реализации близких постановок задач, отличающихся типами математических объектов.

Наряду с решением задачи важно иметь информацию о корректности и эффективности использования алгоритма в конкретной ситуации. Такая информация ассоциируется с решаемой проблемой, а не с используемыми объектами вычислений. Поскольку объекты численных проблем имеют большее время жизни, чем базовые объекты, управление вычислительными ресурсами производится на этом уровне.

3.3. Связи между классами. Специфическая для данной предметной области система понятий, их свойств и отношений находит отражение в системе классов, построенной по принципу иерархической детализации. Свойства ООР, такие как инкапсуляция (объединение данных и методов), наследование (сохранение объектом следующего уровня всех свойств предыдущего) и полиморфизм (возможность выбора нужного действия в зависимости от ситуации), отображаются на газодинамические явления и их взаимосвязи.

Классы делятся на следующие группы:

— классы твердотельной модели (узлы, граничные условия, область);

— классы метода декомпозиции, необходимые для реализации параллельной версии программного кода (подобласть, модуль решения для подобласти);

— численные классы, выполняющие вычисления и хранение данных задачи (векторы, матрицы, графы);

— классы решения задачи (алгоритм решения, схема наложения граничных условий, способы упорядочивания уравнений), облегчающие повторное использование программного кода.

Существуют три основных типа объектов: объекты-операции, объекты-данные и объект-расчетный модуль. Операции и данные являются внутренними объектами системы, а расчетный модуль — внеш-

ним. Объекты-данные соответствуют понятиям предметной области и иерархически упорядочены в соответствии с уровнем абстракции, выделенной при декомпозиции области (например, геометрические, топологические и сеточные объекты). Объекты-операции хранятся в контейнере операций, который представляет собой многосвязный граф. Связи в этом графе отражают отношение использования. Объекты-данные возникают в результате выполнения операций порождения данного объекта и могут использоваться только через изменение соответствующей операции порождения.

Классы подразделяются на базовые и проблемно-ориентированные. Базовые классы не ориентированы на конкретную область применения и носят универсальный характер. Проблемно-ориентированные классы представляют собой высокоуровневые конструкции, характеризуемые узкой областью применения. Они строятся в результате обобщения группы вычислительных методов.

Базовые классы реализуются с помощью шаблонов-заготовок, в которых фиксируется инвариантная часть. Настройка шаблона на конкретную область применения осуществляется фиксацией структурных и функциональных параметров. Структурные параметры определяют топологию параллельной программы, в то время как функциональная часть определяет вычислительное содержание алгоритма.

Для хранения граничных значений искомых функций используются списки с одиночными связями. Связные списки допускают гибкие методы доступа, поскольку каждый элемент данных содержит ссылку на следующий элемент данных в цепочке.

Базовые классы, реализующие работу с динамическими структурами данных, а также с одномерными, двух- и трехмерными массивами данных, оформляются в виде соответствующих обобщенных классов и их шаблонов, что позволяет использовать эти классы в предметных модулях прикладной программы.

Классы векторов и матриц предназначены для хранения информации о системах уравнений и характеристиках узлов. Интерфейсы этих классов предоставляют набор операций в форме перегружаемых операторов. От базового класса матрицы наследуются специальные классы матриц (симметричные, ленточные и др.).

Векторы и матрицы представляют собой объекты класса, в основе которого лежит хранение компонент в контейнере — динамическом массиве. В этом случае векторы имеют любую “разумную” размерность. Исключительным случаем являются векторы нулевой размерности, образующиеся в результате ошибочных действий программы. В отличие от массивов, для векторов и матриц определяются операции линейной алгебры. Различий между векторами-столбцами и векторами-строками не делается.

Контейнерный класс динамических массивов реализуется как шаблон классов. Класс настраивается на хранение значений любого арифметического типа, и в дальнейшем используются различные реализации шаблона. Для объектов этого класса определяются операции изменения размерности, заполнения, индексации и присваивания. Класс векторов наследует все свойства вещественных динамических массивов и определяет новые свойства — операции линейной алгебры. Класс двумерных матриц реализует хранение значений элементов матриц в виде массива векторов-строк. Для объектов этого класса определяются операции изменения размерности матрицы, заполнения, индексации, присваивания и ряд операций линейной алгебры, включая вычисление определителя и получение обратной матрицы. Интерфейсы векторных и матричных классов предоставляют набор векторных и матричных операций в формате перегружаемых операций. В соответствии со стандартом программирования на C++, определяются конструктор по умолчанию, инициирующий конструктор, а также деструктор.

Классы решения задачи предназначены для проверки корректности объектов и установки необходимых связей между ними, а также для управления шагами решения задачи и определения последовательности операций.

Данные, необходимые для параллелизации задачи, инкапсулированы в специальном классе, который встраивает функциональность MPI в иерархию классов для обеспечения простого интерфейса.

Для реализации обменов данными, представленными в памяти не последовательно, а с определенным шаблоном, а также для абстрагирования от специфики шаблона и обмениваемых данных применяются пользовательские типы данных MPI. Типы данных, описывающие шаблон, определяются в локализованном блоке в процессе инициализации программы и допускают легкую модификацию. В функциях обмена тип данных используется в качестве параметра и позволяет реализовать логику обмена абстрактно: например, обмены в разных направлениях выполняет один и тот же программный код.

3.4. Файл сценария и макроязык. Прикладная система классов наряду со своим функциональным назначением обеспечивает поддержку текстового интерфейса, необходимого для конечного пользователя. Для описания параметров задачи и задания их значений в файле сценария используется встроенный макроязык. Реализация макроязыка при помощи объектно-ориентированного подхода позволяет добиться преемственности синтаксических конструкций.

Макроязык включает описание комментариев (для их обозначения используется символ !) и блоков (признаком начала блока служит последовательность символов ***). Каждый блок имеет имя (поддерживаются имена **nonlinear**, **namelist**, **output**, **monitor**). Синтаксический разбор блоков производится в порядке их появления в файле сценария. Исходные данные задаются в размерном виде, а уравнения решаются в безразмерном виде.

Приведем шаблон файла сценария (в квадратных скобках указываются необязательные параметры).

```
*** nonlinear
nlevel  nl_fmg  nl_pre  nl_post
gridfile.1.adf
gridfile.2.adf
...
[flowfile.ini.adf]  flowfile.adf
[visual] [histfile] [outfile] [append]
ncycle  nitsav  nprint  cfl
model
gamma
bc_group.1  bc_type.1      ! name    ! keyword
[boundary conditions]
bc_group.2  bc_type.2      ! name    ! keyword
[bc_cond.1  bc_cond.2 ...]
...
*** namelist
variable = value
*** output
bc_group.1  out_type.1
out_file.1
...
*** monitor
bc_group.1  mon_type.1  mon_save
mon_file.1
...
```

В блоке **nonlinear** указываются параметры, необходимые для инициализации исходных данных и проведения расчетов.

Параметр **nlevel** определяет число уровней сетки, параметр **nl_fmg** — уровень сетки, с которого начинаются многосеточные итерации, параметр **nl_pre** — число итераций для предварительного сглаживания, параметр **nl_post** — число итераций для постсглаживания. Далее следуют имена файлов, содержащих информацию о сетках различного уровня (**gridfile.1.adf**, **gridfile.2.adf**, ...), начиная с сетки хорошей разрешающей способности (всего **nlevel** штук). После этого указываются имена файлов с начальным и конечным полем течения — файлы **flowfile.ini.adf** и **flowfile.adf** (если указывается только один файл, то он используется для инициализации поля течения, а потом затирается). Число переменных, сохраняемых в файле **flowfile.adf**, зависит от типа течения (например, 6 при использовании модели Спаларта–Аллмараса и 7 при использовании k–ε модели).

Следующие параметры указывают на необходимость визуализации поля течения в процессе расчета (параметр **visual**), имя файла для сохранения истории сходимости итерационного процесса (файл **histfile**), имя файла, в котором сохраняются интегральные характеристики потока на каждой итерации (файл **outfile**), а также указание на то, сохранять ранее созданные файлы или нет (параметр **append**).

Параметр **ncycle** определяет число многосеточных итераций, параметр **nitsav** — число итераций, через которое сохраняется поле течения, параметр **nprint** — число итераций, через которое выводится информация о сходимости итерационного процесса, параметр **cfl** — число Куранта–Фридрихса–Леви.

Выбор модели турбулентности происходит в зависимости от значения параметра **model** (невязкое течение, ламинарное течение или турбулентное течение).

Параметр **gamma** определяет род рабочего газа и имеет смысл отношения удельных теплоемкостей (равняется 1.4 для воздуха).

Каждая граница расчетной области характеризуется порядковым номером (параметр **bc_group**), типом выставляемых граничных условий (параметр **bc_type**), именем (для этого служит комментарий **name**), ключевым словом (комментарий **keyword**) и граничными условиями (параметры **bc_cond**). Порядковый номер используется для определения общего числа границ и ссылок на границу в других блоках. Тип

каждой границы, представляющий собой целое число, хранится в файле сетки и определяет дополнительную информацию, которая записывается в файл сетки. Например, для стенки, на которой выставляются граничные условия прилипания и непротекания, в файл сетки записывается расстояние до ближайшей стенки, а при использовании граничных условий скольжения — направления нормали к стенке. Такой подход позволяет сэкономить место на диске, но требует перестройки сетки при смене типа границы (точнее, повторного преформатирования сетки из формата сеточного генератора в формат библиотеки ADF). Имя границы обычно определяется при создании сетки. Ключевое слово содержит необходимые пояснения для пользователя.

Граничные условия задаются двумя способами. В простейшем случае задаются соответствующие численные значения (их количество зависит от типа границы) в файле сценария. При необходимости задания профилей характеристик потока граничные условия определяются в отдельном файле. Файл граничных условий имеет следующий формат:

```
nprof  icoord
coord(1)  var(1,1)  var(1,2)  ...
coord(2)  var(2,1)  var(2,2)  ...
...       ...       ...       ...
```

При этом `nprof` представляет собой число точек, определяющих профиль, `icoord` — направление, в котором задаются граничные условия, `coord(i)` — значения координат, `var1(i,j)` — граничные значения (первый индекс соответствует координате, а второй — функции). Максимальное значение индекса i равняется `nprof`, а максимальное значение индекса j зависит от типа граничных условий. Полагается, что значения `icoord` = 1, 2, 3 и 4 соответствуют координатным направлениям x, y, z и r (при использовании цилиндрической системы координат). При необходимости используется кубическая сплайн-интерполяция.

Блок `namelist` используется для присваивания значений переменным с предопределенным именем (в виде `variable=value`). Например, элементы массива `ncycle(i)` имеют смысл числа сглаживающих итераций на сетках различной разрешающей способности (число элементов массива равняется `nlevel`), а элементы массива `reslev(i)` содержат минимальные значения невязок для всех уровней сетки.

Блок `output` используется для вывода информации о локальных характеристиках потока после окончания расчета. Для этого указывается номер границы (параметр `bc_group`). Параметр `out_type` определяет сохраняемую характеристику в файле с именем `out_file` (1 — для объема ячейки, 2 — для температуры, 3 — для статического давления, 4 — для теплового потока).

Блок `monitor` служит для сохранения интегральных характеристик потока в файле с именем `mon_file` через `mon_save` итераций. Параметр `bc_group` указывает на порядковый номер границы, а параметр `mon_type` определяет сохраняемую характеристику потока (1 — для силы, 2 — для момента).

4. Программные и технические средства. Программный код написан на языках программирования Fortran и C++. Используется модель программирования SPMD (Single Program/Multiple Data), в которой параллельно исполняется несколько копий одной программы на нескольких процессорах. Поддерживается одна версия программного кода, которая не зависит от числа процессоров.

Вопрос построения сетки отделяется от проблемы дискретизации уравнений Навье–Стокса. Расчетная сетка (структурированная или неструктурированная) считается заданной, в частности, построенной при помощи одного из коммерческих сеточных генераторов, таких как Gambit или Ansys ICEM CFD.

Разработанные программные средства используют трансляцию сетки из формата сеточного генератора в формат общедоступной библиотеки ADF (Advanced Data Format), которая является частью библиотеки CGNS (CFD General Notation System). Формат CGNS поддерживается многими вычислительными пакетами (соответствующая опция имеется, например, в пакете Fluent). Для генерации файлов объемом, превышающим 2 Гб (максимальный размер файла ограничивается 4 Гб, но максимальный размер указателя не превышает 2 Гб), что необходимо для проведения расчетов с использованием полной геометрической модели, на системе с 32-разрядной архитектурой используется расширенная версия библиотеки CGNS LFS (Large File System), разработанная корпорацией Роллс–Ройс (Rolls–Royce) и университетом Суррея (University of Surrey). Размер фйала в 2 Гб приблизительно соответствует неструктурированной сетке, содержащей 7×10^6 шестигранных ячеек.

Для визуализации результатов используются пакеты Tecplot и FieldView. Число поддерживаемых форматов данных расширяется за счет реализации и подключения соответствующих фильтров. Визуализация результатов расчетов, сохраненных в формате ADF, проводится при помощи библиотеки VISUAL 3, разработанной в Массачусетском технологическом институте (Massachusetts Institute of Technology, MIT). Она использует X и OpenGL (Open Graphics Library) библиотеки и предназначена для визуализации результатов, полученных на неструктурированных сетках.

Для распараллеливания используется библиотека OPlus (Oxford Parallel Library for Unstructured Solvers), реализованная на языке Fortran 77 в вычислительной лаборатории университета Оксфорда (Oxford University Computing Laboratory) для платформ Win32, Linux, SGI и AIX. К достоинствам библиотеки OPlus относятся: использование обобщенных структур данных (в качестве элемента данных рассматриваются грани и ячейки, которые имеют различный тип — шестигранники, призмы, параллелепипеды, тетраэдры), производительность (посылка сообщений происходит только при модификации данных, а для уменьшения задержек при посылке сообщений данные объединяются в пакеты), переносимость (для посылки сообщений используются функции MPI).

Операции над множествами применяются в любом порядке без влияния на конечный результат, что ограничивает использование библиотеки OPlus для некоторых численных алгоритмов (например, метода Гаусса–Зейделя). Тем не менее многие современные вычислительные алгоритмы удовлетворяют этому ограничению. Множества и указатели декларируются в начале исполняемой программы и не изменяются в дальнейшем, что не позволяет провести динамическое изменение сетки. В то же время, такой подход делает алгоритм независимым от порядка циклов и числа разбиений.

Функции библиотеки OPlus делятся на функции управления процессами, функции инициализации указателей, функции управления вводом/выводом, функции управления вводом/выводом в файлы и функции управлениями циклами.

Использование библиотеки OPlus позволяет запустить программу как в последовательном режиме без вызова функций передачи сообщений (что удобно, например, для отладки кода), так и в параллельном режиме. Последовательная версия библиотеки содержит функции с теми же аргументами, что и ее параллельная версия, но которые не выполняют действий по передаче сообщений (пустое тело функции).

5. Проведение расчетов. Вычисления производятся в режиме удаленного терминала на кластере и суперкомпьютере.

Вычислительные модули кластера связаны между собой высокоскоростной сетью Myrinet (пропускная способность составляет 1 Гбит/с) и сетью Fast Ethernet (пропускная способность составляет 100 Мбит/с). Сеть Myrinet предназначена для высокоскоростного обмена между модулями в ходе вычислений, а сеть Fast Ethernet — для начальной загрузки программ и данных в модуль, передачи служебной информации о ходе вычислительного процесса и соединения решающего поля с управляющим узлом и дисковым массивом.

Суперкомпьютерный центр находится в Центральной лаборатории Совета по научным исследованиям (Central Laboratory for the Research Councils, CLRC). Суперкомпьютер IBM SP/1600 (рис. 2) состоит из 40 SMP-узлов pSeries 690 Regatta, каждый из которых содержит 32 процессора Power 4+ с 64-разрядной архитектурой и тактовой частотой 1.7 ГГц, обеспечивая теоретическую пиковую производительность 6.66 Тфлопс. Для оптимизации передачи данных между фреймами каждый из них разделяется на 4 узла по 8 процессоров (Logical Partitions, LPAR). С точки зрения программирования система представляет собой 160 узлов SMP-типа по 8 процессоров каждый.

Результаты тестирования вычислительной процедуры на ряде модельных задач газовой динамики и задач, связанных с расчетом внутренних турбулентных течений, а также сведения по ускорению и производительности программного кода приводятся в работах [6–12].

6. Заключение. Приведены особенности построения и основные концептуальные положения, принятые при реализации параллельного объектно-ориентированного программного комплекса, предназначенного для численного решения задач механики жидкости и газа. Приводится и обсуждается системное и функциональное наполнение программного комплекса, инструментальный характер разработанных программных средств, а также структура организации классов и подход к их практической реализации.

Обоснован выбор системы объектов, которые лежат в основе построения классов программного комплекса и выражают основные предметные понятия и их свойства. С каждым из выделенных понятий связан базовый класс, а с обобщенной постановкой — необходимый проблемный класс. Базовые классы не ориентированы на конкретную область применения и носят универсальный характер. Проблемно-ориентированные классы представляют собой высокоуровневые конструкции, которые характеризуются узкой областью применения и строятся в результате обобщения группы вычислительных методов.

Применение разработанного подхода представляется перспективным для построения параллельной численной библиотеки и прикладных вычислительных систем, к которым предъявляются повышенные требования к расширяемости, модифицируемости и масштабируемости.

СПИСОК ЛИТЕРАТУРЫ

1. Численное решение многомерных задач газовой динамики / Под ред. С.К. Годунова. М.: Наука, 1976.

2. *Андерсон Д., Таннехилл Дж., Плетчер Р.* Вычислительная гидромеханика и теплообмен. М.: Мир, 1990.
3. *Флетчер К.* Вычислительные методы в динамике жидкостей. М.: Мир, 1991.
4. Управление обтеканием тел с вихревыми ячейками в приложении к летательным аппаратам интегральной компоновки (численное и физическое моделирование) / Под ред. А. В. Еримишина и С. А. Исаева. М., СПб, 2001.
5. *Barth T.J.* Aspects of unstructured grids and finite-volume solvers for the Euler and Navier–Stokes equations // VKI Lecture Series. N 1994-05. Brussels: Von Karman Institute for Fluid Dynamics, 1994.
6. *Волков К.Н.* Применение метода контрольного объема для решения задач механики жидкости и газа на неструктурированных сетках // Вычислительные методы и программирование. 2005. **6**, № 1. 47–64.
7. *Волков К.Н.* Граничные условия на стенке и сеточная зависимость решения в расчетах турбулентных течений на неструктурированных сетках // Вычислительные методы и программирование. 2006. **7**, № 2. 83–95.
8. *Волков К.Н.* Пристеночное моделирование в расчетах турбулентных течений на неструктурированных сетках // Теплофизика и аэромеханика. 2007. **14**, № 1. 113–129.
9. *Волков К.Н.* Реализация схемы расщепления на разнесенной сетке для расчета нестационарных течений вязкой несжимаемой жидкости // Вычислительные методы и программирование. 2005. **6**, № 2. 146–159.
10. *Волков К.Н.* Разностные схемы расчета потоков повышенной разрешающей способности и их применение для решения задач газовой динамики // Вычислительные методы и программирование. 2005. **6**, № 2. 23–44.
11. *Волков К.Н.* Применение средств параллельного программирования для решения задач механики жидкости и газа на многопроцессорных вычислительных системах // Вычислительные методы и программирование. 2006. **7**, № 1. 73–88.
12. *Волков К.Н.* Дискретизация конвективных потоков в уравнениях Навье–Стокса на основе разностных схем высокой разрешающей способности // Вычислительные методы и программирование. 2004. **5**, № 2. 10–26.
13. *Бондаренко Ю.А., Башуров В.В., Янилкин Ю.В.* Математические модели и численные методы для решения задач нестационарной газовой газодинамики. Обзор зарубежной литературы. Препринт РФЯЦ ВНИИЭФ 88-2003. Саров, 2003.
14. *Hackbusch W.* Multi-grid convergence theory // Lecture Notes in Mathematics. N 960. Berlin: Springer-Verlag, 1982. 177–219.
15. *Müller J.-D.* Coarsening 3-D hybrid meshes for multigrid methods // Proceedings of the 9th Copper Mountain Multigrid Conference. Copper Mountain (Colorado, USA), 1999.
16. *Rodi W.* Simulation of turbulence in practical flow calculations // Proceedings of European Congress on Computational Methods in Applied Sciences and Engineering. Barcelona, 2000.
17. *Семенов В.А.* Объектная систематизация и парадигмы вычислительной математики // Программирование. 1997. № 4. 14–25.
18. *Luksch P.* Parallel and distributed implementation of large industrial applications // Future Generation Computer Systems. 2000. **16**. 649–663.
19. *Копысов С.П., Красноперов И.В., Рычков В.Н.* Объектно-ориентированный метод декомпозиции области // Вычислительные методы и программирование. 2003. **4**, № 1. 176–193.
20. *Dubois-Pelerin Y., Pegon P.* Improving modularity in object-oriented finite element programming // Communications in Numerical Methods in Engineering. 1997. **13**, N 3. 193–198.
21. *Mukunda G.R., Sotelino E.D., Hsieh S.H.* Distributed finite element computations using object-oriented techniques // Engineering with Computers. 1998. **14**, N 1. 59–72.
22. *Munthe O., Langtangen H.P.* Finite element and object-oriented implementation techniques in computational fluid dynamics // Computational Methods in Applied Mechanics Engineering. 2000. **190**. 865–888.
23. *Саблин М.Н.* Программная реализация алгоритмов численного решения операторно-разностных сеточных задач двумерной газовой динамики с использованием системы классов C++ // Вычислительные методы и программирование. 2006. **7**, № 1. 144–154.
24. *Иванов И.Э., Крюков И.А., Терехов И.В.* Объектно-ориентированная программная система подготовки данных и визуализации результатов газодинамических расчетов // Математическое моделирование. 2001. **13**, № 7. 110–115.
25. *Иванов И.Э., Крюков И.А., Терехов И.В.* Особенности построения программной среды обеспечения газодинамических расчетов // Математическое моделирование. 2002. **14**, № 8. 28–30.
26. *Peskin A.P., Hardin G.R.* An object-oriented approach to general purpose fluid dynamics software // Computers and Chemical Engineering. 1996. **20**, N 8. 1043–1058.

Поступила в редакцию
28.03.2007