УДК 519.63; 004.272.26 doi 10.26089/NumMet.v20r215

ЧИСЛЕННЫЕ МЕТОДЫ ДЛЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, УСТРОЕННОГО ПО ПРИНЦИПУ “ЧЕРНОГО ЯЩИКА”

С. И. Мартыненко[1,2]

Сформулированы требования к вычислительным алгоритмам для перспективного программного обеспечения, устроенного по принципу “черного ящика” и предназначенного для математического моделирования в механике сплошных сред. Выполнен анализ прикладных свойств классических многосеточных методов и универсальной многосеточной технологии в рамках проблемы “универсальность–эффективность–параллелизм”. Показано, что близкая к оптимальной трудоемкость при минимуме проблемно-зависимых компонентов и высокая эффективность параллелизма достижимы при использовании универсальной многосеточной технологии на глобально структурированных сетках. Применение неструктурированных сеток потребует определения двух проблемно-зависимых компонентов (межсеточных операторов), которые значительно влияют на трудоемкость алгоритма.

Ключевые слова: параллельные и высокопроизводительные вычисления, краевые задачи, многосеточные методы, программное обеспечение.

1. Введение. Сущность математического моделирования состоит в замене исходного объекта или процесса его математическим “аналогом” (математической моделью) и дальнейшем изучении модели с помощью реализуемых на компьютерах вычислительно-логических алгоритмов [11]. С одной стороны, математическая модель позволяет сравнительно легко исследовать свойства и поведение исходного объекта или процесса в любых возможных ситуациях, что характерно для теоретических методов исследования. С другой стороны, вычислительные эксперименты с моделями объектов или процессов позволяют подробно и глубоко изучать объекты в полноте, недоступной чисто теоретическим подходам, что характерно для экспериментальных методов исследования. Поэтому методология математического моделирования продолжает развиваться и охватывать все новые проблемные области — от разработки технических систем и управления ими до анализа сложнейших экономических и социальных процессов [11].

Вычислительный эксперимент наряду с физическим и натурными экспериментами является одним из основных способов исследования и получения новых знаний в различных областях науки и техники. Практическая реализация возможностей математического моделирования существенно повышает эффективность инженерных разработок — особенно при создании принципиально новых, не имеющих прототипов машин и приборов, материалов и технологий, что позволяет сократить существенно затраты времени и средств. В настоящее время математическое моделирование используют не только в качестве расчетно-теоретического сопровождения на стадии отработки технического устройства, но и при его проектировании, подборе и оптимизации его эксплуатационных режимов, анализе его надежности и прогнозировании отказов и аварийных ситуаций, а также при оценке возможностей улучшения характеристик и модернизации технических устройств.

Математическое моделирование какого-либо технического объекта или процесса условно представляют как совокупность трех основных этапов: модель–алгоритм–программа [11]. На первом этапе выбирают (или строят) систему уравнений, отражающую в математической форме важнейшие свойства моделируемого объекта или процесса — законы, которым он подчиняется, связи, присущие составляющим его частям, и др. Второй этап состоит в выборе или разработке алгоритма для реализации модели на компьютере. Модель представляется в форме, удобной для применения численных методов, определяется последовательность вычислительных и логических операций, которые нужно произвести, чтобы найти искомые величины с заданной точностью. Вычислительные алгоритмы должны не искажать основные свойства модели и, следовательно, исходного объекта, быть экономичными и адаптирующимися к особенностям решаемых задач и используемых компьютеров. На третьем этапе создаются программы, “пере-

[1] Центральный институт авиационного моторостроения им. П. И. Баранова, ул. Авиамоторная, д. 2, 111116, Москва; науч. сотр., e-mail: Martynenko@ciam.ru

[2] Институт проблем химической физики РАН, просп. Семенова, д. 1, 142432, Московская область, г. Черноголовка; ст. науч. сотр., e-mail: Martynenko@icp.ac.ru

водящие" модель и алгоритм на доступный компьютеру язык. К ним также предъявляются требования экономичности и адаптивности [11].

Во времена ламповых компьютеров каждый машинный расчет был заметным событием. Однако в конце прошлого века появились персональные компьютеры, достаточно мощные для проведения научно-технических расчетов. К вычислительной технике получили широкий доступ инженеры, физики, химики и специалисты из других проблемных областей, которые не обладали достаточной подготовкой в области вычислительной математики, но сталкивались с необходимостью решения прикладных задач, требующих большого объема вычислений. Сразу стало понятно, что наибольший эффект от применения методов математического моделирования в решении научно-технических задач достигается при использовании специализированного программного обеспечения, позволяющего существенно снизить усилия на написание и отладку компьютерных программ. Как правило, математические модели в естественнонаучных и технических приложениях являются системами (не)линейных (интегро-)дифференциальных уравнений, численное решение которых требует напряженного труда коллектива высококвалифицированных программистов и математиков. Поэтому пользователям, обладающим должной квалификацией только в своей проблемной области, лучше работать с готовым программным обеспечением, созданным профессиональными разработчиками.

Все проблемы, связанные с развитием современного программного обеспечения для математического моделирования физико-химических процессов, можно условно разделить на три группы: "физические", "математические" и "компьютерные".

"Физические" проблемы связаны с трудностью математического описания сложных физико-химических процессов, таких как гидродинамика и тепломассообмен в многофазных реагирующих средах, турбулентное горение и др.

"Компьютерные" проблемы возникают из-за трудностей совместимости разнообразных и быстро обновляемых программных и аппаратных средств.

"Математические" проблемы связаны со сложностью формализации основных этапов вычислительного эксперимента: построения вычислительной сетки, аппроксимации основополагающих (не)линейных (интегро-)дифференциальных уравнений и эффективного решения систем нелинейных уравнений высокого порядка на последовательном или параллельном компьютере.

И если сложность "физических" проблем обусловлена многообразием моделируемых процессов и глубиной их математического описания, то "математические" проблемы являются следствием недостаточной изученности основополагающих (не)линейных (интегро-)дифференциальных уравнений и недостаточным совершенством методов численного решения уравнений математической физики.

В настоящей статье рассмотрены "математические" проблемы моделирования, связанные с решением основополагающих уравнений моделей, и выполнен анализ наиболее распространенных численных методов с точки зрения применения их к решению прикладных задач математического моделирования. Целью работы является формулирование требований к численным методам для перспективного программного обеспечения, устроенного по принципу "черного ящика", и формирование облика вычислительной технологии для унифицированного решения широкого класса задач механики сплошных сред.

2. Требования к вычислительным алгоритмам. Механика сплошной среды — это обширная часть механики, посвященная движению газообразных, жидких и твердых деформируемых тел. Помимо обычных материальных тел в механике сплошной среды рассматриваются также особые среды — поля: электромагнитное поле, поле излучений, гравитационное поле (поле тяготения) и др. Моделирование полей имеет как фундаментальный, так и прикладной характер, что придает результатам вычислительных экспериментов особую актуальность.

Рассмотрим подробнее математические модели, используемые для описания физико-химических процессов, протекающих в области Ω. Предположим, что данная математическая модель представима в виде системы нелинейных (интегро-)дифференциальных уравнений

$$\begin{cases} \mathcal{N}_1(u_1, u_2, u_3, \ldots, u_M; \ldots, P_{1l}, \ldots) = f_1, \\ \mathcal{N}_2(u_1, u_2, u_3, \ldots, u_M; \ldots, P_{2l}, \ldots) = f_2, \\ \ldots\ldots\ldots\ldots\ldots \\ \mathcal{N}_M(u_1, u_2, u_3, \ldots, u_M; \ldots, P_{Ml}, \ldots) = f_M, \end{cases} \tag{1}$$

где $\mathcal{N}_m$, $m = 1, 2, \ldots, M$, — нелинейные (интегро-)дифференциальные операторы, f_m, $m = 1, 2, \ldots, M$, — известные функции, u_m, $m = 1, 2, \ldots, M$, — искомые функции, а P — некоторые известные параметры.

Пусть $\mathcal{G}_k$, $k = 1, 2, \ldots, K$ $(K \geqslant M)$, — некоторые нелинейные (интегро-)дифференциальные операторы и g_k, $k = 1, 2, \ldots, K$, — известные функции, а Q — некоторые известные параметры. Тогда система (1) может быть дополнена граничными условиями

$$\begin{cases} \mathcal{G}_1(u_1, u_2, u_3, \ldots, u_M; \ldots, Q_{1l}, \ldots) = g_1, \\ \mathcal{G}_2(u_1, u_2, u_3, \ldots, u_M; \ldots, Q_{2l}, \ldots) = g_2, \\ \ldots\ldots\ldots\ldots\ldots \\ \mathcal{G}_K(u_1, u_2, u_3, \ldots, u_M; \ldots, Q_{Ml}, \ldots) = g_K. \end{cases} \tag{2}$$

Если система (1) описывает нестационарное поведение какого-либо технического объекта или процесса, то в дополнение к граничным условиям (2) необходимо добавить начальное условие, т.е. описание состояния моделируемого объекта или процесса в некоторый (начальный) момент времени.

Будем считать, что система (1) с граничными условиями (2) (и начальными условиями) есть некоторая краевая (или начально-краевая) задача из механики сплошных сред (гидродинамика, электромагнетизм, теплопередача и др.). Далее всегда будем полагать, что поставленная задача имеет единственное решение.

Предположим, что в области Ω построена некоторая вычислительная сетка G с количеством узлов N_G для аппроксимации системы (1) с граничными условиями (2). Дискретный аналог (1), полученный в результате некоторой аппроксимации исходной задачи на сетке G, принимает вид

$$\begin{cases} \mathcal{N}_1^h\big(u_1^h, u_2^h, u_3^h, \ldots, u_M^h; \ldots, P_{1l}, \ldots\big) = f_1^h, \\ \mathcal{N}_2^h\big(u_1^h, u_2^h, u_3^h, \ldots, u_M^h; \ldots, P_{2l}, \ldots\big) = f_2^h, \\ \ldots\ldots\ldots\ldots\ldots \\ \mathcal{N}_M^h\big(u_1^h, u_2^h, u_3^h, \ldots, u_M^h; \ldots, P_{Ml}, \ldots\big) = f_M^h, \end{cases} \tag{3}$$

где $\mathcal{N}_m^h$, $m = 1, 2, \ldots, M$, — дискретные аналоги нелинейных (интегро-)дифференциальных операторов $\mathcal{N}_m$, $m = 1, 2, \ldots, M$, а f_m^h и u_m^h, $m = 1, 2, \ldots, M$, — дискретные аналоги функций f_m и u_m соответственно.

Аналогично дискретный аналог граничных условий (2), полученный в результате некоторой аппроксимации на сетке G, имеет вид

$$\begin{cases} \mathcal{G}_1^h\big(u_1^h, u_2^h, u_3^h, \ldots, u_M^h; \ldots, Q_{1l}, \ldots\big) = g_1^h, \\ \mathcal{G}_2^h\big(u_1^h, u_2^h, u_3^h, \ldots, u_M^h; \ldots, Q_{2l}, \ldots\big) = g_2^h, \\ \ldots\ldots\ldots\ldots\ldots \\ \mathcal{G}_K^h\big(u_1^h, u_2^h, u_3^h, \ldots, u_M^h; \ldots, Q_{Ml}, \ldots\big) = g_K^h, \end{cases} \tag{4}$$

где $\mathcal{G}_k^h$, $k = 1, 2, \ldots, K$ $(K \geqslant M)$, — дискретные аналоги нелинейных (интегро-)дифференциальных операторов $\mathcal{G}_k$ и g_k^h, $k = 1, 2, \ldots, K$, — дискретные аналоги функций g_k. Далее будем полагать, что система (3) с граничными условиями (4) имеет единственное решение.

Осуществим глобальную линеаризацию (3)–(4) и выберем некоторое упорядочение уравнений и неизвестных. Обычно упорядочение выбирают таким образом, чтобы на главной диагонали матрицы коэффициентов располагались максимальные по модулю элементы [19]. Тогда линеаризованная система (3) и граничные условия (4) представимы в виде системы линейных алгебраических уравнений (СЛАУ)

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{5}$$

где матрица коэффициентов A и вектор правых частей $\boldsymbol{b}$ определяются вычислительной сеткой G, типом и порядком аппроксимации на ней исходной краевой или начально-краевой задачи (1)–(2), методом линеаризации системы нелинейных уравнений (3)–(4), а также упорядочением уравнений и неизвестных. Зачастую матрица A содержит много нулевых элементов (т.е. является разреженной). Решение СЛАУ (5) означает отыскание вектора неизвестных $\boldsymbol{x}$, который желательно найти, выполняя наименьшее количество арифметических и логических операций. В отдельных случаях (например, для анизотропных задач) выбор упорядочения сильно влияет на объем вычислений, поэтому упорядочение уравнений и неизвестных является проблемно-зависимой компонентой алгоритма.

Решение нестационарных задач подразумевает решение СЛАУ (5) на каждом временно́м слое, а стационарных — только на одном. Далее, для общности стационарные задачи будем считать нестационарными с одним временны́м слоем и бесконечно большим шагом по времени, а линейные задачи — нелинейными, но для решения которых потребуется одна итерация по нелинейности.

Сейчас мы не будем детализировать способы построения вычислительной сетки, тип и порядок аппроксимации исходной (интегро-)дифференциальной задачи (1)–(2) и метод линеаризации системы нелинейных уравнений (3)–(4), однако заметим, что линеаризованные граничные условия включены в результирующую СЛАУ (5) и поэтому далее отдельно рассматриваться не будут.

Отличительной особенностью результирующей СЛАУ (5) является большое количество уравнений N. В самом деле, матрица коэффициентов A СЛАУ (5) имеет размер $\approx N_G M \times N_G M$, где N_G — количество узлов вычислительной сетки G, а M — количество уравнений системы (1). В приложениях часто встречаются три случая.

1) Многомасштабные задачи (multiscale problems): количество узлов N_G вычислительной сетки G достаточно велико, а количество уравнений M в исходной системе (1) достаточно мало. Данный случай связан с процессами, которые характеризуются большим диапазоном изменения масштабов решений. Например, при прямом моделировании турбулентности необходимо использовать вычислительные сетки с малым шагом по пространственным направлениям для разрешения самых маленьких вихрей в турбулентном потоке. При этом отношение характерных геометрических размеров области к размерам мельчайших вихрей может достигать значений 10^3–10^5 и выше. В данном случае $M = 4$ в (1) (три уравнения движения и уравнение неразрывности в системе Навье–Стокса), но количество узлов сетки становится чересчур большим из-за трехмерности моделируемого процесса, т.е. $N_G = 10^9$–10^{15} и выше, поэтому количество уравнений N СЛАУ (5), приблизительно равное $N_G M$, будет чересчур большим. Следует также помнить, что турбулентность является сугубо трехмерным и сугубо нестационарным явлением, поэтому СЛАУ (5) необходимо решать на каждом временно́м слое, количество которых тоже достаточно велико.

2) Многофизичные задачи (multiphysics problems): количество узлов N_G вычислительной сетки G достаточно мало, а количество уравнений M в исходной системе (1) достаточно велико. Данный случай связан с изучением сложных явлений, которые характеризуются большим количеством одновременно протекающих физико-химических процессов. Например, при нагреве до высоких температур авиационный керосин начинает разлагаться на более простые углеводородные соединения, причем математические модели термодеструкции керосина могут состоять из сотен, а иногда и тысяч, уравнений [12]: при $M = 10^2$–10^4 и $N_G = 10^7$–10^9 количество уравнений в результирующей СЛАУ (5), приблизительно равное $N_G M$, тоже будет чересчур большим.

3) Многомасштабные и многофизичные проблемы: количество узлов N_G вычислительной сетки G и количество уравнений M в исходной системе (1) достаточно велики. Например, 4D математическая модель авиационного двигателя является примером многомасштабной и многофизичной проблемы: различие масштабов обусловлено точностью моделирования турбулентного переноса и горения топлива при расчете совокупности одновременно протекающих физико-химических процессов (гидродинамика, сопряженный теплообмен, химические реакции, деформация элементов конструкции двигателя и т.д.). Поскольку вычислительная техника не позволит в ближайшем будущем решать подобные задачи, то сейчас их решение можно отложить до лучших времен.

Проблемы численного моделирования, связанные с разработкой и теоретическим обоснованием универсальных, эффективных и параллельных алгоритмов, не снимаются сами собой по мере появления все более мощных и дешевых компьютеров. Это связано, по меньшей мере, с двумя причинами: усложнением выдвигаемых как практикой, так и теорией задач и необходимостью проведения большого числа серий вычислительных экспериментов для достаточно полного изучения объекта. Поэтому разработка вычислительных алгоритмов еще долго останется одной из ключевых проблем математического моделирования [4].

Уровень развития математического моделирования во многом определяется количеством уравнений N результирующей СЛАУ (5). В ряде частных случаев решение СЛАУ (5) тривиально, если матрица коэффициентов A является диагональной (явные схемы) или треугольной (полунеявные схемы). Однако в общем случае именно численное решение СЛАУ (5) является наиболее трудоемкой частью вычислительного эксперимента, которая вместе с быстродействием компьютера и объемом оперативной памяти определяет максимальный размер решаемой задачи. Сравнительно недавно для прикладных задач было характерно $N = N_G M = 10^5$–10^6 уравнений, а современному уровню соответствуют значения $N = N_G M = 10^7$–10^{10}. Зачастую отсутствие эффективного численного метода вынуждает упрощать решаемую задачу с целью снижения объема вычислительной работы или рассматривать частные случаи,

что снижает практическую ценность получаемых результатов.

Линеаризация нелинейной дискретной задачи (3)–(4) может быть глобальной (линеаризация всех членов всех уравнений) или локальной (линеаризация отдельных членов отдельных уравнений). Итерационное решение линеаризованной задачи (3)–(4) может быть раздельным (неизвестную функцию u_m^h отыскивают из m-го уравнения системы (3) при фиксированных, т.е. взятых с предыдущей итерации, значениях остальных неизвестных u_k^h, $k \neq m$) или совместным (неизвестные функции u_m^h объединяют в блок неизвестных и новые их значения отыскивают одновременно). Как правило, наименее трудоемким является совместный алгоритм с локальной линеаризацией.

Воспользуемся простейшим и наиболее исследованным случаем задачи (1)–(2), а именно двумерной задачей Дирихле для уравнения Пуассона, чтобы показать характерные трудности численного решения результирующей СЛАУ (5). Пусть в области Ω, которая является единичным квадратом, необходимо отыскать решение уравнения Пуассона

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f(x,y)$$

с однородными граничными условиями

$$u(0,y) = u(1,y) = u(x,0) = u(x,1) = 0.$$

Построим в единичном квадрате Ω равномерную вычислительную сетку посредством разбиения сторон квадрата на n частей. Тогда узлы $\bigl(x_i^{\mathrm{v}}, y_j^{\mathrm{v}}\bigr)$ сетки определены соотношениями

$$x_i^{\mathrm{v}} = (i-1)h = \frac{i-1}{n}, \quad i = 1,2,\ldots,n+1, \qquad y_j^{\mathrm{v}} = (j-1)h = \frac{j-1}{n}, \quad j = 1,2,\ldots,n+1,$$

где $h = 1/n$ — параметр дискретизации (шаг сетки). В данном случае количество арифметических операций $\mathcal{W}$ (трудоемкость метода), необходимых для решения задачи Дирихле некоторым численным методом, будет зависеть только от одного параметра — шага сетки h (или количества уравнений N результирующей СЛАУ (5)). Стандартная пятиточечная аппроксимация уравнения Пуассона имеет вид

$$\frac{u_{i-1j}^h - 2u_{ij}^h + u_{i+1j}^h}{h^2} + \frac{u_{ij-1}^h - 2u_{ij}^h + u_{ij+1}^h}{h^2} = -f\bigl(x_i^{\mathrm{v}}, y_j^{\mathrm{v}}\bigr), \tag{6}$$

где сеточная u_{ij}^h есть аналог функции $u(x,y)$, т.е. $u_{ij}^h = u\bigl(x_i^{\mathrm{v}}, y_j^{\mathrm{v}}\bigr)$. Используя естественное упорядочение неизвестных и исключая граничные условия $u_{1j}^h = u_{n+1j}^h = u_{i1}^h = u_{in+1}^h = 0$, запишем разностную краевую задачу (6) в матричном виде (5). В данном случае $M = 1$ (одно уравнение Пуассона), поэтому количество уравнений результирующей СЛАУ (5) составит: $N = MN_G = N_G = (n-1)^2 \approx h^{-2}$.

Рассмотрим различные методы решения разностной краевой задачи (6).

Прямые методы решения СЛАУ (5) основаны на преобразовании исходной системы к треугольному виду [1, 3–5, 7, 10, 13, 20]. К достоинствам прямых методов следует отнести возможность точного решения СЛАУ, выполняя конечное количество арифметических операций (при отсутствии погрешностей округления) и слабую зависимость алгоритма от структуры матрицы A. Недостатком является чересчур большой объем вычислительной работы (т.е. большое количество арифметических операций, которые нужно выполнить для сведения матрицы коэффициентов к треугольному виду): как правило, количество арифметических операций (ао) пропорционально кубу количества уравнений (в рассматриваемом случае $\mathcal{W} \sim N_G^3$ ао). Поэтому прямые методы (обычно метод Гаусса с выбором главного элемента) чаще всего используют для решения СЛАУ малого размера ($MN_G \leqslant 100$) или в итерационных методах для решения СЛАУ с матрицей коэффициентов особого вида [7, 9]. С точки зрения программ, устроенных по принципу “черного ящика”, главным достоинством прямых методов является отсутствие проблемно-зависимых компонентов.

Итерационные методы решения СЛАУ (методы последовательных приближений) производят последовательность приближений $\boldsymbol{x}^{(\nu)}$ к точному решению $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ [1, 3–5, 7, 10, 13, 20]. Простейшие (одношаговые) итерационные методы можно представить в виде

$$W\,\frac{\boldsymbol{x}^{(\nu+1)} - \boldsymbol{x}^{(\nu)}}{\tau} = \boldsymbol{b} - A\boldsymbol{x}^{(\nu)}, \quad \nu = 0,1,2,\ldots, \tag{7}$$

где матрица расщепления W и релаксационный параметр τ определяют итерационный метод, а ν — счетчик итераций. Чтобы начать вычисления по (7), необходимо задать начальное приближение $\boldsymbol{x}^{(0)}$, что

не всегда является тривиальной задачей. Далее новое приближение $\boldsymbol{x}^{(\nu+1)}$ к точному решению $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ отыскивают как решение вспомогательной СЛАУ

$$W\bar{\boldsymbol{x}} = \bar{\boldsymbol{b}},$$

где $\bar{\boldsymbol{x}} = \boldsymbol{x}^{(\nu+1)} - \boldsymbol{x}^{(\nu)}$ и $\bar{\boldsymbol{b}} = \tau\big(\boldsymbol{b} - A\boldsymbol{x}^{(\nu)}\big)$. Очевидно, что матрица расщепления W должна быть легко обратимой, т.е. диагональной или треугольной, чтобы уменьшить объем вычислений, необходимых для отыскания нового приближения $\boldsymbol{x}^{(\nu+1)} = \boldsymbol{x}^{(\nu)} + \bar{\boldsymbol{x}} = \boldsymbol{x}^{(\nu)} + W^{-1}\bar{\boldsymbol{b}}$ к решению СЛАУ (5). Сходимость итераций (7) означает монотонное убывание ошибки приближений

$$\big\|\boldsymbol{x}^{(\nu+1)} - \boldsymbol{x}\big\| < \big\|\boldsymbol{x}^{(\nu)} - \boldsymbol{x}\big\|.$$

Каждый итерационный метод содержит, как минимум, одну проблемно-зависимую компоненту — критерий останова итераций. Итерации стараются прервать при достижении условия (критерия останова)

$$\big\|\boldsymbol{x}^{(\nu+1)} - \boldsymbol{x}\big\| \leqslant \varepsilon,$$

где ε — наперед заданное малое число (как правило, зависящее от h). Сформулировать "универсальный" критерий останова итераций трудно, поскольку вектор $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ неизвестен. Кроме того, критерий останова должен быть согласован с порядком аппроксимации исходной (интегро-)дифференциальной задачи. Недостаточное количество выполненных итераций приводит к потере точности численного решения, а избыточное — к неоправданному росту объема вычислений.

Если исходная задача является линейной, то матрица A и вектор правых частей $\boldsymbol{b}$ результирующей СЛАУ (5) являются постоянными, в противном случае матрица A и вектор $\boldsymbol{b}$ изменяются в процессе сходимости итераций по нелинейности.

Необходимые и достаточные условия сходимости итерационных методов сформулированы в многочисленных публикациях [1, 3–5, 7, 10, 13, 20], хотя обычно их трудно проверить при решении прикладных задач.

Положим, что количество итераций (7), которые необходимо выполнить для достижения критерия останова, составит $\sim n^k$, где $k \geqslant 0$ — некоторая константа, зависящая от конкретного метода, а вычислительная стоимость каждой итерации пропорциональна количеству уравнений $N = n^d$, $d = 2, 3$, линейной системы (5). Тогда трудоемкость итерационного алгоритма (7) составит $\mathcal{W} \sim n^k N = N^{1+k/d}$ арифметических операций (ао).

В опубликованной в 1874 году работе Филипп Людвиг фон Зейдель предложил итерационный метод решения СЛАУ, ныне известный как метод Зейделя. Представим матрицу коэффициентов A в (5) в виде

$$A = L + D + U,$$

где L — нижняя треугольная матрица с нулевыми элементами на главной диагонали, D — матрица, образованная диагональными элементами матрицы A, и U — верхняя треугольная матрица с нулевыми элементами на главной диагонали. Полагая $\tau = 1$ и $W = L + D$ в (7), итерационный метод Зейделя можно записать в форме

$$(L + D)\big(\boldsymbol{x}^{(\nu+1)} - \boldsymbol{x}^{(\nu)}\big) = \boldsymbol{b} - A\boldsymbol{x}^{(\nu)}, \quad \nu = 0, 1, 2, \ldots. \tag{8}$$

Теоретический анализ показывает, что для задачи (6) $k = 2$, т.е. объем вычислений, который необходимо выполнить для достижения критерия останова, составит $\mathcal{W} \sim N^2$ ао в двухмерном случае ($d = 2$) и $\mathcal{W} \sim N^{5/3}$ ао в трехмерном случае ($d = 3$). Трудоемкость метода Зейделя при надлежащем выборе проблемно-зависимых компонентов (упорядочения уравнений и неизвестных, а также критерия останова) меньше, чем трудоемкость прямого метода Гаусса с выбором главного элемента ($\mathcal{W} \sim N^3$ ао), но все еще остается высокой. Зейдель использовал свой метод при ручных расчетах, обратил внимание на достаточно медленную сходимость итераций (8) и даже не рекомендовал им пользоваться.

Метод последовательной верхней релаксации (SOR) является самой простой попыткой снизить трудоемкость метода Зейделя (8). Чтобы уменьшить объем вычислительной работы, в итерации Зейделя вводят параметр релаксации τ

$$(L + \tau D)\,\frac{\boldsymbol{x}^{(\nu+1)} - \boldsymbol{x}^{(\nu)}}{\tau} = \boldsymbol{b} - A\boldsymbol{x}^{(\nu)}, \quad \nu = 0, 1, 2, \ldots. \tag{9}$$

Теоретически показано, что оптимальное значение параметра релаксации τ_{opt} лежит в интервале (0, 2). При этом объем вычислений, который необходимо выполнить при решении задачи (6), равен $\mathcal{W} \sim N^{3/2}$

арифметических операций [8, 10]. Трудоемкость метода SOR (9) меньше, чем метода Зейделя (8), при надлежащем выборе трех проблемно-зависимых компонентов — упорядочения неизвестных, параметра верхней релаксации τ и критерия останова.

Оптимальное значение параметра верхней релаксации τ_{opt} можно определить теоретическими методами лишь в самых простейших случаях, а в общем случае этот вопрос так и остался открытым [19]. Проблемно-зависимые компоненты породили проблему оптимизации итерационных алгоритмов, т.е. проблему выбора их оптимальных значений посредством использования некоторой априорной информации о решаемой задаче.

Таким образом, трудоемкость рассмотренных выше методов Гаусса, Зейделя (8) и SOR (9) при решении разностной краевой задачи (6) составила $\mathcal{W} \sim N^3$ ао, $\mathcal{W} \sim N^2$ ао и $\mathcal{W} \sim N^{3/2}$ ао соответственно. Поэтому еще в начале 60-х годов прошлого века многие математики задумывались над возможностью построения оптимального (с минимальной трудоемкостью) алгоритма для решения СЛАУ (5), т.е. алгоритма, позволяющего численно решать разностные краевые задачи, выполняя $\mathcal{W} \sim N$ ао. Поскольку вычислительная стоимость одной итерации (7) составит $\mathcal{W} \sim N$ ао, то количество итераций оптимального алгоритма не должно зависеть от шага сетки!

Спустя 87 лет после опубликования работы Зейделя совершенно неожиданно выяснилось, что потенциал итерационного метода (8) далеко не исчерпан. Выдающийся отечественный математик Радий Петрович Федоренко обратил внимание на сложный характер сходимости метода Зейделя: на первых итерациях ошибка приближений к решению (т.е. разность между приближением и точным решением) быстро уменьшается, причем независимо от количества неизвестных, а на последующих итерациях ошибка уменьшается гораздо медленнее и сильно зависит от N [9, 23]. Разлагая ошибку в ряд Фурье, нетрудно показать, что коротковолновые гармоники можно быстро удалить на текущей сетке за малое количество итераций, а длинноволновые гармоники можно аппроксимировать на более грубых сетках, т.е. на сетках с бóльшим шагом. В западной литературе данное утверждение получило название *основного многосеточного принципа*[1]. В ставшей уже легендарной статье [14], опубликованной в 1961 году, Р.П. Федоренко предложил использовать быструю сходимость метода Зейделя на первых итерациях в многосеточном алгоритме для достижения оптимальной (минимальной) трудоемкости.

Значительный вклад в развитие многосеточных методов внесли Н.С. Бахвалов и Г.П. Астраханцев. В 70-е годы многосеточные методы получили развитие в США, Европе и Израиле, где были сформулированы классические многосеточные алгоритмы и исследована их сходимость [9, 23]. Оптимальная трудоемкость многосеточных методов со временем привлекла к ним широкий интерес, и область их приложений постоянно расширяется на новые научно-технические задачи.

В классических многосеточных методах (КММ) наряду с исходной СЛАУ используют вспомогательные СЛАУ меньшей размерности, по способу построения которых многосеточные алгоритмы разделяют на геометрические и алгебраические [23]. В геометрических многосеточных методах (ГММ) построение вспомогательных СЛАУ осуществляется с привлечением информации о вычислительной сетке, а в алгебраических многосеточных методах (АММ) построение вспомогательных СЛАУ осуществляется из исходной СЛАУ алгебраическими методами, т.е. без использования информации о вычислительной сетке. С АММ связаны надежды разработать высокоэффективный алгоритм для численного решения краевых задач на неструктурированных сетках [23].

Однако сразу же были выявлены слабые стороны КММ: оптимальная (минимальная) трудоемкость ($\mathcal{W} \sim N$ ао) достигается за счет оптимальной адаптации проблемно-зависимых компонентов (сглаживающая процедура, тип грубых сеток, способ задания оператора на грубых сетках, операторы межсеточных переходов, многосеточный цикл и т.д.) к решаемой задаче. Такой подход оправдан лишь при наличии некоторой априорной информации о решаемой задаче или при решении большой серии однотипных задач. Фактически, КММ являются набором проблемно-зависимым компонентов, оптимальный выбор которых и обусловливает оптимальную сходимость алгоритма в целом[2]. В общем случае оптимальный выбор компонентов КММ является отдельной и достаточно сложной задачей.

КММ разделяют на оптимизированные (optimized multigrid algorithms) и робастные[3] (robust multigrid algorithms). В оптимизированных многосеточных алгоритмах стараются адаптировать все компоненты

[1]The essential multigrid principle is to approximate the smooth (long wavelength) part of the error on coarser grids. The non-smooth or rough part is reduced with a small number (independent of h) of iterations with a basic iterative method on the fine grid [25].

[2]However, the multi-grid method cannot be understood as a fixed algorithm [18].

[3]В иностранной литературе итерационный метод, эффективный для решения широкого класса задач, называется робастным (when a smoother is efficient for a large class of problems it is called robust [25]). Данное определение представляется неудачным, поскольку не оговариваются причины эффективности метода.

(сглаживающая процедура, тип грубых сеток, способ задания оператора на грубых сетках, операторы межсеточных переходов, многосеточный цикл и т.д.) к рассматриваемой проблеме, чтобы получить минимально возможную трудоемкость. В робастных многосеточных алгоритмах стараются выбирать эти компоненты независимо от заданной проблемы, но одинаково для максимально большого класса задач. Данные методы часто используют в программных пакетах, где минимальная трудоемкость для одной задачи не так важна [23]. Как правило, робастные алгоритмы пытались построить последовательно путем совершенствования различных вариантов КММ. Это напоминало попытку переделать гоночный автомобиль в вездеход и к успеху не привело, несмотря на обилие публикаций по данной теме. Нужны были новые идеи.

Другим недостатком явилась невозможность эффективного распараллеливания КММ [7, 20]. Классическая многосеточная итерация связана с решением ряда СЛАУ разного размера — от самой большой, соответствующей самой мелкой сетке, до самой малой, соответствующей самой грубой сетке. При распараллеливании многосеточной итерации разный объем вычислений распределяется между одинаковым количеством независимых вычислителей (процессоров, ядер и т.д.), поэтому при сглаживании на грубых сетках (минимум вычислительной работы) эффективность параллелизма снижается из-за увеличения издержек, связанных с обменом данными и возможного простоя ряда независимых вычислителей [23].

Теперь проанализируем достоинства и недостатки рассмотренных численных методов (метод Гаусса, метод Зейделя (8), SOR (9) и КММ) с точки зрения использования их в перспективном программном обеспечении. Но сначала дадим ряд определений.

Определение 1. Универсальность вычислительного алгоритма определяется количеством проблемно-зависимых компонентов. Среди алгоритмов, предназначенных для решения определенного класса задач, универсальным будет тот, который содержит наименьшее количество проблемно-зависимых компонентов.

Вычислительный алгоритм для решения СЛАУ (5), полученной в результате аппроксимации одиночного (интегро-)дифференциального уравнения, называется эффективным, если он обладает оптимальной трудоемкостью (т.е. позволяет решать СЛАУ, выполняя $O(N)$ ао, где N — количество уравнений) или близкой к ней (т.е. позволяет решать СЛАУ, выполняя $O(N \log N)$ ао).

Математические модели в задачах механики сплошных сред, как правило, являются системами нелинейных (интегро-)дифференциального уравнений, описывающими совокупность одновременно протекающих физико-химических процессов. Проще всего решать уравнения модели раздельно, т.е. последовательно решать нелинейные уравнения (3) с граничными условиями (4), отыскивая неизвестную функцию u_m^h из m-го уравнения системы (3). Раздельный алгоритм нарушает взаимосвязь между отдельными уравнениями системы (3) в процессе их численного решения, что приводит к увеличению объема вычислений. В совместном алгоритме неизвестные функции u_m^h объединяют в блок неизвестных, с каждым блоком связана СЛАУ с плотно заполненной матрицей коэффициентов, получаемая в результате локальной линеаризации системы (3). Решение этой СЛАУ прямым методом позволяет получить обновленные значения всех неизвестных, образующих данный блок [24]. Как правило, общая трудоемкость совместного алгоритма меньше, чем раздельного, но трудоемкость итерации может оказаться выше. Пусть каждый из n_b блоков образован N_b неизвестными, т.е. $N = n_b N_b$, где N — количество уравнений системы (3). Тогда для выполнения одной итерации необходимо решить n_b СЛАУ размера $N_b \times N_b$, т.е. трудоемкость этой итерации составит $\mathcal{W} = O\big((n_b/N)^{1-\varsigma} N\big)$ ао, где $\varsigma = 1$ для точечного упорядочения неизвестных ($n_b = N$) и $\varsigma = 3$ для блочного упорядочения ($n_b < N$).

Определение 2: Вычислительный алгоритм для совместного решения нелинейной системы (3)–(4) называется *эффективным*, если он обладает оптимальной трудоемкостью (т.е. позволяет решать нелинейную систему (3)–(4), выполняя $O\big((n_b/N)^{1-\varsigma} N\big)$ ао, где N — количество уравнений нелинейной системы (3), n_b — количество блоков неизвестных). Также будем считать алгоритм эффективным, если его трудоемкость близка к оптимальной (т.е. позволяет решать нелинейную систему (3)–(4), выполняя $O\big((n_b/N)^{1-\varsigma} N \log N\big)$ ао).

К данным определениям следует сделать следующие замечания.

1) Минимальное количество проблемно-зависимых компонентов алгоритма означает минимальное использование априорной информации о решаемой задаче, т.е. конструкция универсального алгоритма должна быть исключительно проста.

2) Из всех рассмотренных выше методов решения разностной краевой задачи (6) прямой метод Гаусса является наиболее универсальным (отсутствие проблемно-зависимых компонентов), далее в порядке убывания следуют метод Зейделя (8) (две проблемно-зависимые компоненты — упорядочение уравнений и неизвестных, а также критерий останова), SOR (9) (три проблемно-зависимые компоненты — упоря-

дочение уравнений и неизвестных, критерий останова и параметр верхней релаксации) и КММ (набор проблемно-зависимых компонентов: сглаживающая процедура, тип грубых сеток, способ задания оператора на грубых сетках, операторы межсеточных переходов, многосеточный цикл и т.д.). С точки зрения эффективности наблюдается обратный порядок: наиболее эффективны КММ ($\mathcal{W} \sim N$ ао), далее следуют метод SOR (9) ($\mathcal{W} \sim N^{3/2}$ ао), метод Зейделя (8) ($\mathcal{W} \sim N^2$ ао) и прямой метод Гаусса ($\mathcal{W} \sim N^3$ ао). Таким образом, требования универсальности и эффективности вычислительного алгоритма являются взаимоисключающими.

3) Оптимальная трудоемкость вычислительного алгоритма ($\mathcal{W} \sim N$ ао) достижима лишь при решении простейших задач типа (6), а при решении сложных прикладных задач, как правило, достижима лишь трудоемкость, близкая к оптимальной ($\mathcal{W} \sim N \log N$ ао). Далее будет показано, что алгоритм с трудоемкостью $\mathcal{W} \sim N \log^2 N$ ао уже не позволит эффективно распараллеливать вычисления.

Современные математические модели предназначены для исследования сложных природных явлений или физико-химических процессов, протекающих в технологическом оборудовании. Многодисциплинарность моделей, сложный математический аппарат решения основополагающих уравнений и суперкомпьютерные технологии не позволяют одному специалисту охватить все аспекты математического моделирования. Особенно неразумно тратить силы на написание и отладку программ, если необходимо выполнить лишь несколько расчетов. Поэтому неоднократно предпринимались попытки разработать программные комплексы, устроенные по принципу “черного ящика” (black box software), которые существенно облегчат составление математических моделей и численное решение основополагающих уравнений.

Определение 3 (“математическое”). Программа или подпрограмма называется автономной (или устроенной по принципу “черного ящика”), если она не требует какого-либо дополнительных данных от пользователя, кроме определения задачи, состоящей из решаемой линейной дискретной системы уравнений и правой части (матрицы коэффициентов A и вектора правых частей $\boldsymbol{b}$ в (5)). Пользователь ничего не знает о многосеточных методах и воспринимает подпрограмму как некий метод линейной алгебры[4].

Данное определение программ, устроенных по принципу “черного ящика”, произошло от АММ и итерационных методов на подпространствах Крылова [22, 23]. Первоначально предполагалось, что некоторая задача типа (1)–(2) в конечном итоге сводится к СЛАУ (5), которая может быть решена АММ без контроля вычислительного процесса со стороны пользователя. Однако неуниверсальность, необходимость глобальной линеаризации системы нелинейных уравнений (3)–(4) и трудности распараллеливания вычислений остались неустранимыми недостатками АММ.

В [7, 20] было высказано предположение, что программы, устроенные по принципу “черного ящика”, следует разрабатывать, не разделяя отдельные этапы вычислительного эксперимента (построение вычислительной сетки, аппроксимация уравнений математической модели, локальная линеаризация нелинейных сеточных задач и численное решение результирующих систем нелинейных уравнений), а наоборот — объединяя их в единую вычислительную технологию. Термин “технология” в [7, 20] определен традиционным образом как совокупность приемов обработки, и было дано следующее “инженерное” определение.

Определение 4 (“инженерное”). Программа называется автономной (или устроенной по принципу “черного ящика”), если она не требует каких-либо дополнительных данных от пользователя, кроме определения физической задачи, состоящей из геометрии области, граничных и начальных условий, перечня уравнений, которые необходимо решить (уравнение теплопроводности, уравнения Навье–Стокса, уравнения Максвелла и т.д.), и рабочих сред. Пользователю не нужно ничего знать о численных методах, высокопроизводительных и параллельных вычислениях[5].

Очевидно, что подобные программы (в смысле “инженерного” определения) будут очень удобны для специалистов-нематематиков, однако они накладывают жесткие ограничения на используемые в них численные методы. Ранее уже упоминалось о проблеме “универсальность–эффективность”, когда универсальные методы решения разностных краевых задач не эффективны, а эффективные — не универсальны. Фактически прогресса в данном направлении вычислительной математики не было, поскольку исходная проблема низкой эффективности вычислительных алгоритмов заменялась на чуть более простую проблему оптимизации проблемно-зависимых компонентов.

[4] A program or subroutine may be called autonomous if it does not require any additional input from the user apart from problem specification, consisting of the linear discrete system of equations to be solved and the right-hand side. The user does not need to know anything about multigrid methods. The subroutine is perceived by the user as if it were just another linear algebra solution method [25].

[5] We define software to be black-box if it does not require any additional input from the user apart from the physical problem specification consisting of the domain geometry, boundary and initial conditions, the enumeration of equations to be solved (heat conductivity equation, Navier–Stokes equations, Maxwell equations, etc.) and mediums. The user does not need to know anything about numerical methods, or high-performance and parallel computing [20].

Первые же попытки разработать программы, устроенные по принципу “черного ящика”, столкнулись с проблемой “универсальность–эффективность”, т.е. быстро выяснилось, что необходимых численных методов просто нет. Современные пакеты инженерного анализа обычно имеют модульную структуру, обязательно графический интерфейс и интерпретаторы текстовых команд, предоставляющих пользователю набор удобных инструментов для управления формированием виртуальной модели, проведением расчетов и отображением полученных результатов. В области анализа гидрогазодинамических процессов и тепломассопереноса в России нашли распространение такие коммерческие пакеты, как STAR-CD/STARCCM+, Fluent, CFX, FlowVision и GasDynamicsTool [2]. По уровню полноты реализуемых физических и математических моделей и необходимому уровню профессиональной подготовки пользователей первые три относят к группе так называемых пакетов “тяжелого класса”. STAR-CD/STARCCM+, Fluent, CFX — это коммерческие профессиональные CFD-комплексы для решения широкого спектра задач механики сплошных сред и тепломассообмена. Моделирование процессов, протекающих в жидких и газообразных средах, в пакетах STAR-CD, Fluent и CFX осуществляется на основе численного решения полных трехмерных нестационарных уравнений Навье–Стокса. Пакеты обеспечивают возможность анализа течений вязкой ньютоновской и неньютоновской жидкости и газа в широком диапазоне скоростей от ползучих до гиперзвуковых течений при ламинарном и турбулентном режимах. Однако современные пакеты прикладных программ еще очень далеки от совершенства и фактически предлагают пользователю инструментарий, который лишь облегчает математическое моделирование. Разработчики коммерческих программ не стали особо заморачиваться математическими проблемами, а просто запрограммировали наиболее распространенные численные методы, используемые в математическом моделировании, снабдив их дружественным пользовательским интерфейсом. Пользователь, расставляя галочки в выпадающих меню, может быстро “собрать” из предлагаемого набора подпрограмм вычислительный алгоритм для решения своей задачи. Фактически, проблема оптимизации алгоритмов была переложена на пользователей, что не сильно напрягает разработчиков, больше озабоченных прибылью, а не качеством программного продукта. Пользователи тоже особо не напрягаются оптимизацией алгоритмов и чаще всего используют настройки “по умолчанию”. В результате повсеместно слышны жалобы на нехватку вычислительных мощностей, что на самом деле означает отсутствие должного решения проблемы “универсальность–эффективность”.

Ситуация с развитием программ, устроенных по принципу “черного ящика”, обострилась в 90-е годы, когда многопроцессорная вычислительная техника стала получать широкое распространение. Сразу выяснилось, что эффективные численные методы не только плохо поддаются формализации, но и не позволяют эффективно распараллеливать вычисления. Напомним основные количественные характеристики параллельных алгоритмов.

Определение 5. *Ускорением* $\left(\bar{\mathcal{S}}\right)$ и *эффективностью* $\left(\bar{\mathcal{E}}\right)$ параллельного алгоритма называется величина

$$\bar{\mathcal{S}} = p\bar{\mathcal{E}} = \frac{\bar{T}(1)}{T(p)}\,, \tag{10}$$

где $\bar{T}(1)$ — время выполнения алгоритма на одном процессоре, а $T(p)$ — время выполнения параллельного алгоритма на системе из p процессоров.

Определение 6. *Ускорением* $\left(\tilde{\mathcal{S}}\right)$ и *эффективностью* $\left(\tilde{\mathcal{E}}\right)$ параллельного алгоритма по отношению к наилучшему последовательному алгоритму называется величина

$$\tilde{\mathcal{S}} = p\tilde{\mathcal{E}} = \frac{\tilde{T}(1)}{T(p)}\,, \tag{11}$$

где $\tilde{T}(1)$ — время выполнения быстрейшего последовательного алгоритма на одном процессоре, а $T(p)$ — время выполнения параллельного алгоритма на системе из p процессоров.

Эффективность $\bar{\mathcal{E}}$ представляет теоретический интерес и характеризует параллелизм конкретного алгоритма, а ускорение $\tilde{\mathcal{S}}$ представляет практический интерес и характеризует уменьшение времени решения задачи по сравнению с наиболее быстрым последовательным алгоритмом (как правило, это оптимизированный вариант КММ). Для приложений наибольший интерес представляют параллельные алгоритмы с ускорением $1 < \tilde{\mathcal{S}}(p) < p$.

Предположим, что время выполнения пропорционально трудоемкости, т.е. $T(1) \sim \mathcal{W}$, и в качестве итерационного метода выберем метод Якоби с параметром (damped Jacobi method), который является сглаживателем [9, 23]. Тогда

$$\bar{T}(1) = CN^2, \quad T(p) = C\,\frac{1}{p}\,N^2 + T^*,$$

где N — количество уравнений в СЛАУ (5), C — некоторая константа, зависящая от итерационного метода, p — количество независимых вычислителей, T^* — время, затрачиваемое на издержки параллельного исполнения (обмены данными и т.д.). Тогда из определения (10) следует

$$\bar{\mathcal{S}} = p\bar{\mathcal{E}} = \frac{CN^2}{C\,\dfrac{1}{p}\,N^2 + T^*}\,.$$

Полагая $T^* = 0$, получим $\bar{\mathcal{S}} = p$ и $\bar{\mathcal{E}} = 1$, т.е. метод Якоби с параметром теоретически обладает полным параллелизмом.

Совершенно иные оценки получаются при сравнении параллельного метода Якоби с параметром с последовательным многосеточным методом, для которого $\tilde{T}(1) = C^*N$, где C^* — некоторая константа. Полагая $T^* = 0$, из определения (11) получим $\tilde{\mathcal{S}} = O(p/N)$ и $\tilde{\mathcal{E}} = O(1/N)$, т.е. $\tilde{\mathcal{E}} \to 0$ при достаточно больших N. Это означает, что параллельный метод Якоби с параметром будет уступать последовательному многосеточному алгоритму, в котором метод Якоби с параметром использован в качестве сглаживателя.

Аналогично можно рассмотреть параллельный итерационный метод с трудоемкостью, близкой к оптимальной ($\mathcal{W} \sim N \log N$ ао). Тогда $\tilde{\mathcal{S}} = O(p/\log N)$ и $\tilde{\mathcal{E}} = O(1/\log N)$, т.е. при $p > \log N$ такой параллельный метод обеспечит ускорение по сравнению с быстрейшим последовательным алгоритмом. Очевидно, что итерационный метод с трудоемкостью $\mathcal{W} \sim N \log^2 N$ ао уже не позволит достичь ускорения $\tilde{\mathcal{S}} > 1$ при достаточно больших N.

Совокупность всех требований к численным методам для программных комплексов, устроенных по принципу “черного ящика”, образует проблему “универсальность–эффективность–параллелизм”. *Формализация вычислений в перспективных программных комплексах означает разработку и применение универсальных* (*т.е. с минимальным количеством проблемно-зависимых компонентов*)*, эффективных* (*т.е. с трудоемкостью не более* $\mathcal{W} = O\big((n_b/N)^{1-\varsigma} N \log N\big)$ *ао, где* $\varsigma = 1$ *для точечного упорядочения неизвестных* ($n_b = N$) *и* $\varsigma = 3$ *для блочного упорядочения* ($n_b < N$) *и параллельных* (*т.е. выполняемых быстрее наилучшего последовательного алгоритма:* $\tilde{\mathcal{S}} > 1$) *численных методов для решения задач, которые мы умеем решать.* Разработка методов решения новых задач является процессом творческим и не поддающимся формализации. Изучение этих новых задач обязательно начинается средствами теоретической математики, которыми стараются установить необходимые вопросы существования, единственности, корректности и основные свойства решений.

Напомним ряд особенностей прикладных задач, которые необходимо учитывать при разработке численных методов для программных комплексов, устроенных по принципу “черного ящика”.

1) *Неопределенный порядок аппроксимации исходной задачи* (1)–(2). В настоящее время происходит интенсивное развитие методов высокоточной аппроксимации (интегро-)дифференциальных операторов. Предполагается, что высокий порядок аппроксимации позволит уменьшить количество узлов вычислительной сетки и, как следствие, уменьшить размер результирующей СЛАУ (5). Однако подобный подход зачастую бесполезен при решении прикладных задач. Во-первых, исходная задача (1)–(2), как правило, сформулирована при ряде допущений физического характера. В задачах гидродинамики упрощенно часто ставят граничные условия на входе и выходе из области, игнорируют или весьма своеобразно учитывают шероховатость омываемых поверхностей. Зачастую параметры P и Q, входящие в систему (1) и граничные условия (2), известны весьма приближенно. Поэтому при неточно заданных исходных данных формальное повышение порядка аппроксимации (интегро-)дифференциальной задачи не всегда приводит к повышению точности численного решения. Во-вторых, высокий порядок аппроксимации задачи (1)–(2) подразумевает достаточную гладкость решения, которая зависит от многих факторов и характерна не для всех задач. В-третьих, минимальный размер шага сетки определяется минимальным масштабом решения, и возможности уменьшения количества узлов сетки посредством повышения порядка аппроксимации в ряде приложений весьма ограничены. Поэтому в общем случае следует ожидать, что порядок аппроксимации разностной схемы будет не выше второго. Однако в частных случаях возможно применение высокоточных схем. Поэтому в программных комплексах, устроенных по принципу “черного ящика”, должна быть предусмотрена возможность гибкого изменения порядка аппроксимации в зависимости от особенностей исходной задачи (1)–(2).

2) *Неопределенный метод аппроксимации исходной задачи* (1)–(2). В настоящее время метод контрольного объема (МКО) и метод конечных элементов (МКЭ) получили широкое распространение для решения прикладных задач из различных проблемных областей. Для схем низкого порядка аппроксимации МКО ни в чем не уступает МКЭ, но существенно проще и технологичнее. Тем не менее, в программных комплексах, устроенных по принципу “черного ящика”, должна быть предусмотрена возможность не только гибкого изменения порядка, но и типа аппроксимации.

3) *Широкий диапазон изменения шагов сетки и параметров задачи.* В приложениях часто встречаются математические модели, описывающие как эволюцию некоторого объекта или процесса, так и стационарное состояние. В качестве иллюстрации рассмотрим некоторую начально-краевую задачу для уравнения теплопроводности

$$\frac{\partial u}{\partial t} = a\,\frac{\partial^2 u}{\partial x^2} + a\,\frac{\partial^2 u}{\partial y^2} + f(x, y, t),$$

где a — коэффициент температуропроводности.

Трудоемкость численного решения данной начально-краевой задачи зависит от соотношения величин шага сетки по времени h_t, шага по пространству h и коэффициента температуропроводности a. Если данные параметры удовлетворяют неравенству $4ah_t < h^2$, то можно использовать явную схему, которая сводится к СЛАУ с диагональной матрицей коэффициентов и ее решение потребует одной итерации.

Если $4ah_t \sim h^2$, то можно использовать неявную схему, которая сводится к СЛАУ с пятидиагональной матрицей коэффициентов и ее решение потребует нескольких итерации из-за хорошей обусловленности матрицы коэффициентов A (5), вызванной сильным диагональным преобладанием при малых $h_t \to 0$.

Если $h_t = h$ и $a = O(h)$, то трудоемкость метода Зейделя (8) будет оптимальной ($\mathcal{W} \sim N$ ао) и количество итераций не будет зависеть от шага сетки.

Наконец, если $h_t \gg h^2/a$, то трудоемкость метода Зейделя (8) составит $\mathcal{W} \sim N^2$ ао, т.е. количество итераций Зейделя будет пропорционально количеству узлов сетки.

Таким образом, количество итераций метода Зейделя (8), необходимое для решения начально-краевой задачи для уравнения теплопроводности на каждом временно́м слое, может изменяться от одной до $O(N)$ в зависимости от соотношения шагов сетки h_t/h и величины коэффициента температуропроводности a. Поэтому при выборе численных методов для перспективного программного обеспечения необходимо рассматривать в первую очередь наихудший случай $h_t = \infty$. Очевидно, что в общем случае счет на установление не позволит получать решение краевых задач с трудоемкостью, близкой к оптимальной.

4) *Нелинейность прикладных задач.* Подавляющее большинство прикладных задач являются нелинейными, что крайне затрудняет теоретический анализ, разработку универсальных алгоритмов и приводит к увеличению требуемой вычислительной работы. Решение нелинейных задач посредством некоторого способа линеаризации сводят к решению совокупности линейных задач. Наиболее распространенным является метод Ньютона, который обладает квадратичной скоростью сходимости при условии, что начальное приближение выбрано достаточно близко к искомому корню [3, 5]. В противном случае итерации по нелинейности могут расходиться [7]. Самым простым способом контроля сходимости итераций по нелинейности при решении системы нелинейных уравнений (3) является метод последовательной нижней релаксации, например (9) с параметром $\tau \in (0, 1)$. Если метод Ньютона определяет правильное направление движения к корню, то надлежащий выбор параметра нижней релаксации τ позволит контролировать погрешность очередного приближения к решению. В достаточной близости от корня принимают $\tau = 1$. Основная сложность состоит в определении стратегии выбора параметра τ, которая позволит минимизировать количество итераций по нелинейности. Итерационные методы, которые сходятся при достаточно произвольно выбранном начальном приближении к решению нелинейной задачи, достаточно сложны и трудоемки [15]. В общем случае невозможно формализовать численное решение нелинейных задач, поэтому всегда надо быть готовым к различным сюрпризам.

5) *Построение вычислительной сетки.* В настоящее время именно отсутствие универсальных технологий построения вычислительных сеток является той самой проблемой, которая существенно ограничивает формализацию вычислительных алгоритмов для математического моделирования физико-химических процессов и сдерживает развитие перспективных программных комплексов. Многообразие возникающих здесь вопросов позволяет оценить простое перечисление часто применяемых типов сеток: адаптивные, структурированные, неструктурированные и квазиструктурированные, согласованные, несогласованные и мортарные, регулярные и нерегулярные, статические и динамические и др. Важно отметить, что современные наиболее эффективные подходы связаны с достаточно сложными дискретными объектами и их преобразованиями, включая последовательности иерархических сеток и их локальные сгущения, декомпозиции сеточных областей на подобласти, динамическую перестройку сеток, апостериорный и/или априорный учет свойств искомых решений и т.д. [4].

Н.Е. Жуковский часто говорил, что “механика есть искусство выражать задачи о движении уравнениями, которые до конца интегрируются”. В рассматриваемой проблематике данное утверждение можно переформулировать так: математическое моделирование есть искусство описывать объекты и процессы уравнениями, а также строить вычислительные сетки, на которых эти уравнения эффективно решаются численными методами.

6) *Ограниченность ресурсов компьютера.* Разрабатывая численные методы для перспективного программного обеспечения, всегда следует помнить, что ресурсы компьютера (в первую очередь, оперативная память) являются ограниченными. Поэтому объем оперативной памяти, необходимый для реализации численных методов, должен быть намного меньше, чем объем памяти для хранения исходных данных.

В завершение кратко сформулируем основные требования к численным методам для программного обеспечения, устроенного по принципу "черного ящика":

— универсальность (минимальное количество проблемно-зависимых компонентов);

— эффективность (трудоемкость не более $O\big((n_b/N)^{1-\varsigma} N \log N\big)$ ао в широком диапазоне изменения параметров задачи);

— параллелизм (ускорение по сравнению с наилучшим последовательным алгоритмом);

— адаптивность (возможность гибкого изменения способа и порядка аппроксимации);

— минимальное использование ресурсов компьютера.

3. Универсальная многосеточная технология. В 80-е годы прошлого века наблюдался бурный рост публикаций по многосеточным методам. Каждая новая задача приводила к новым многосеточным алгоритмам или к дальнейшему совершенствованию основных компонентов. Перспектива решать краевые задачи с оптимальными вычислительными усилиями выглядела очень заманчиво, однако попытки построить робастный многосеточный метод на основе КММ оказались неудачными.

Сильные (оптимальная трудоемкость) и слабые (необходимость оптимизации и трудности распараллеливания многосеточных итераций) стороны КММ имеют одинаковую причину — вспомогательные СЛАУ меньшего размера, которые используют для отыскания поправки (coarse grid correction). Не сразу пришло понимание, что робастный многосеточный метод может быть основан только на принципиально иной форме алгоритмизации основополагающей идеи Р.П. Федоренко, а не как последовательное совершенствование известных алгоритмов.

Тем не менее, уже в 1990 г. была высказана смелая идея применить основной многосеточный принцип в односеточном алгоритме, т.е. отказаться от грубых сеток так, чтобы максимально сохранить достоинства КММ и устранить их недостатки. Разработанный метод численного решения краевых задач получил название *Универсальная*[6] *Многосеточная*[7] *Технология*[8] (УМТ) [7, 20]. Фактически, УМТ состоит из МКО, метода Ньютона (для нелинейных задач) и псевдомногосеточного метода Зейделя с блочным упорядочением неизвестных, трудоемкость которого снижена до близкой к оптимальной без использования проблемно-зависимых компонентов.

Поскольку УМТ является разновидностью геометрических многосеточных методов, то сначала рассмотрим вычислительные сетки простейшей топологии.

Определение 7. Сетка называется регулярной, если ее топология полностью определяется некоторым набором правил[9]. Это означает, что, зная только индексы узла, можно определить все соседние узлы, а также вычислить их координаты [3].

Определение 8. Регулярная сетка G_1^0, образованная $(N+1)^d$ узлами $(d = 2, 3)$, называется структурированной, если ее подсетки G_k^l, $l = 1, 2, \ldots, L^+$, $k = 1, 2, \ldots, 3^{dl}$, обладают следующими свойствами.

Свойство 1. Каждая сетка G_k^l ($l \neq L^+$, где L^+ — уровень с самыми грубыми подсетками) представима в виде объединения 3^d соответствующих ей грубых подсеток уровня $l+1$. Как следствие, исходная сетка G_1^0 представима в виде объединения всех подсеток одного уровня l:

$$G_1^0 = \bigcup_{k=1}^{3^{dl}} G_k^l\,, \quad l = 0, \ldots, L^+\,.$$

Свойство 2. Подсетки одного уровня l не имеют общих точек, т.е. $G_n^l \cap G_m^l = \varnothing\,,\ n \neq m,\ l = 1, \ldots, L^+$.

Свойство 3. Каждый контрольный объем на подсетках G_k^l представим в виде объединения 3^{dl} контрольных объемов на исходной сетке G_1^0.

[6]Термин "универсальный" указывает на минимальное количество проблемно-зависимых компонентов.

[7]Термин "многосеточный" является неудачным для односеточного алгоритма, более уместно говорить "псевдомногосеточный", но, к сожалению, данная терминология уже устоялась.

[8]Термин "технология" использован в традиционном смысле как совокупность приемов обработки: адаптации краевых и начально-краевых задач к УМТ, аппроксимации адаптированных задач на вычислительной сетке и решение сеточных уравнений псевдомногосеточным методом с минимальным количеством проблемно-зависимых компонентов.

[9]Это важное свойство позволяет существенно экономить компьютерные ресурсы.

Совокупность всех сеток G_k^l называется *многосеточной структурой*, где $l = 0, 1, \ldots, L^+$ — сеточные уровни, состоящие из 3^{dl} сеток. На первый взгляд, многосеточная структура похожа на иерархию сеток, используемых в ГММ ($k = 1$), однако в УМТ вспомогательные подсетки G_k^l, $l > 0$, явно не строят и никакие сеточные функции (coarse grid variables) с ними не связаны.

Напомним основные положения УМТ на примере линейной краевой задачи [7, 20]

$$L_\Omega u(\boldsymbol{x}) = f_\Omega(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \qquad L_{\partial\Omega} u(\boldsymbol{x}) = f_{\partial\Omega}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega.$$

Здесь $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ и $\Omega \in \mathbb{R}^d$ — заданная открытая область с границей $\partial\Omega$, L_Ω и $L_{\partial\Omega}$ — дифференциальные операторы, определенные в области Ω и ее границе $\partial\Omega$ соответственно, f_Ω и $f_{\partial\Omega}$ обозначают данные функции на Ω и $\partial\Omega$. Будем кратко записывать краевую задачу $Lu = f$ в предположении, что она имеет решение $u = L^{-1} f$.

Данная краевая задача может быть переписана в Σ-модифицированной форме. Представление искомого решения в виде суммы двух функций

$$u = c + \hat{u}$$

приводит к следующей Σ-модифицированной форме краевой задачи:

$$Lc = f - L\hat{u}.$$

В последующих многосеточных итерациях[10] сеточный аналог функции $\hat{u}(x)$ будет служить приближением к решению разностной краевой задачи, а сеточный аналог функции $c(x)$ — поправкой, вычисляемой на грубых сетках. Представление решения в виде суммы двух функций, называемое Σ-модификацией решения, является одной из форм адаптации решаемых краевых задач к УМТ [7, 20].

Интегрирование Σ-модифицированной краевой задачи по контрольному объему V на сетке G_k^l приведет к следующему соотношению:

$$\frac{1}{V}\int\limits_V Lc\,dv = \frac{1}{V}\int\limits_V \bigl(f - L\hat{u}\bigr)\,dv. \tag{12}$$

В результате аппроксимации МКО сеточный аналог уравнения (12) примет окончательный вид

$$A_l \boldsymbol{c}_l^h = \mathcal{R}_{0\to l}\bigl(\boldsymbol{b} - A\hat{\boldsymbol{u}}^{\,h}\bigr)_0, \quad l = 0, 1, \ldots, L^+, \tag{13}$$

где A_l — блочно-диагональная матрица коэффициентов результирующей СЛАУ, причем количество блоков (3^{dl}) равно количеству сеток G_k^l, образующих данный уровень l; $\boldsymbol{c}_l^h$ — вектор неизвестных (поправок); $\mathcal{R}_{0\to l}$ — проблемно-независимый оператор сужения (restriction operator), который проецирует невязку $\bigl(\boldsymbol{b} - A\hat{\boldsymbol{u}}^{\,h}\bigr)_0$ (сеточный аналог $f - L\hat{u}$) с исходной (самой мелкой) сетки G_1^0 на сетки уровня l, L^+ — уровень с самыми грубыми подсетками.

Основные различия между ГММ и УМТ состоят в следующем.

1) В ГММ сглаживание связано со СЛАУ разного размера, а в УМТ аппроксимация Σ-модифицированной краевой задачи МКО осуществляется на исходной (самой мелкой) сетке G_1^0, поэтому сглаживание связано с $L^+ + 1$ СЛАУ одинакового размера.

2) Аппроксимация модифицированных краевых задач на многосеточной структуре МКО использует аддитивность определенного интеграла относительно подобластей (свойство 3). Оператор сужения $\mathcal{R}_{0\to l}$ однозначно определяется свойством аддитивности определенного интеграла и способом аппроксимации интегралов на исходной (самой мелкой) сетке G_1^0, поэтому не зависит от решаемой задачи (свойство 3). Очевидно, что $\mathcal{R}_{0\to 0} = I$, где I — единичный оператор. Правая часть (13) есть осредненная невязка, причем осреднение осуществляется в узлах исходной (самой мелкой) сетки G_1^0, расположенных в реальной части данного контрольного объема.

3) Свойство 2 обеспечивает высокий параллелизм УМТ, поскольку сглаживание на сетках одного уровня можно проводить независимо друг от друга.

4) Для реализации УМТ необходим только один дополнительный массив данных для хранения поправки $\boldsymbol{c}_l^h$ (свойство 1).

Сглаживающие итерации Зейделя на сетках G_k^l определены следующим образом:

$$(L + D)\Bigl(\bigl(\boldsymbol{c}_l^h\bigr)^{(\nu+1)} - \bigl(\boldsymbol{c}_l^h\bigr)^{(\nu)}\Bigr) = \mathcal{R}_{0\to l}\bigl(\boldsymbol{b} - A\hat{\boldsymbol{u}}^{\,h}\bigr)_0 - A_l\bigl(\boldsymbol{c}_l^h\bigr)^{(\nu)}, \quad \nu = 0, 1, 2, \ldots. \tag{14}$$

[10] Данные наименования функций $\hat{u}$ (approximation to the solution) и c (coarse grid correction) пришлось позаимствовать из КММ для сохранения устоявшейся терминологии.

В итерациях Зейделя (14) используют блочное упорядочение неизвестных, которое позволяет применять данную сглаживающую процедуру к решению широкого класса задач, включая совместное решение систем особого вида [24]. Обычно выполняют несколько сглаживающих итераций для удаления высокочастотных компонентов ошибки.

После выполнения сглаживающих итераций на уровне l осуществляется переход к уровню $l-1$ с более мелкими сетками, который представим в следующем матричном виде:

$$\boldsymbol{c}_{l-1}^h = \mathcal{P}_{l\to l-1}\boldsymbol{c}_l^h,$$

где проблемно-независимый оператор пролонгации $\mathcal{P}_{l\to l-1}$ переставляет компоненты вектора неизвестных, его конструкция однозначно определяется только количеством узлов исходной сетки G_1^0. После завершения многосеточной итерации осуществляют пересчет приближения к решению $\big(\hat{\boldsymbol{u}}^{\,h} := \hat{\boldsymbol{u}}^{\,h} + \boldsymbol{c}_l^h$, где ":=" означает равенство в смысле присвоения$\big)$ и обнуление поправки $\big(\boldsymbol{c}_l^h = 0\big)$. УМТ использует простейший (пилообразный[11]) многосеточный цикл для отыскания поправки $\boldsymbol{c}_0^h$.

Результаты теоретического анализа показали, что количество многосеточных итераций УМТ не зависит от величины шага сетки, однако вычислительная стоимость многосеточной итерации несколько выше классической и составляет $O(N\log N)$ ао [7, 20]. Незначительное увеличение стоимости многосеточных итераций УМТ по сравнению с КММ обусловлено ее проблемно-независимыми компонентами, которые не надо оптимизировать.

Сделаем несколько замечаний о решении нелинейных задач. Поскольку для нелинейного оператора $\mathcal{N}$ справедливо $\mathcal{N}\big(c+\hat{u}\big) \neq \mathcal{N}(c)+\mathcal{N}\big(\hat{u}\big)$, то численное решение нелинейных задач требует индивидуального рассмотрения.

1) В ряде случаев нелинейный оператор $\mathcal{N}$ обладает свойством

$$\mathcal{N}\big(c+\hat{u}\big) = \mathcal{N}^*(c)+\mathcal{N}\big(\hat{u}\big),$$

где $\mathcal{N}^* \neq \mathcal{N}$. Тогда Σ-модифицированная нелинейная задача принимает вид

$$\mathcal{N}^*(c) = f - \mathcal{N}\big(\hat{u}\big).$$

Эта Σ-модификация применима к решению уравнений Навье–Стокса [7, 20].

2) Для отдельных нелинейных задач (например, нелинейных конвективно-диффузионных уравнений) удобнее использовать Π-модификацию, т.е. представлять решение в виде произведения двух функций. Σ- и Π-модификации краевых задач являются частными случаями аддитивного и мультипликативного выделения особенностей [3].

3) В общем случае можно применить подход, аналогичный FAS (Full Approximation Storage) [17, 23]. Прибавим к обеим частям Σ-модифицированной нелинейной задачи $0 = f - \mathcal{N}\big(c+\hat{u}\big)$ слагаемое $\mathcal{N}(c)$:

$$\mathcal{N}(c) = f + \mathcal{N}(c) - \mathcal{N}\big(c+\hat{u}\big).$$

Правую часть вычисляют по значениям, взятым с предыдущего сеточного уровня с более крупными сетками, т.е. с запаздыванием на один уровень. В результате аппроксимации МКО сеточный аналог принимает вид

$$\mathcal{N}^h\big(\boldsymbol{c}_l^h\big) = \mathcal{R}_{0\to l}\Big(f^h + \mathcal{N}^h\big(\boldsymbol{c}^h\big) - \mathcal{N}^h\big(\boldsymbol{c}^h + \hat{\boldsymbol{u}}^{\,h}\big)\Big)_0, \quad l = 0,1,\ldots,L^+.$$

Используемый в УМТ в качестве сглаживателя метод Зейделя с блочным упорядочением неизвестных для совместного решения систем (не)линейных уравнений является обобщением метода Ванки [24], первоначально разработанного для численного решения уравнений Навье–Стокса [9]. В данном случае сглаживающие итерации связаны с локальной линеаризацией исходной системы нелинейных уравнений и решением СЛАУ сравнительного малого размера ($N < 30$) с плотно заполненной матрицей коэффициентов прямыми методами. Именно применение прямых методов позволяет использовать сглаживатели на основе метода Ванки для решения широкого класса прикладных задач, начиная от уравнения Пуассона до совместного решения систем нелинейных уравнений в частных производных [7, 20]. К сожалению, сходимость метода Ванки не доказана, однако распространено мнение, что это лучший сглаживатель при решении уравнений Навье–Стокса.

[11] Пилообразный цикл (sawtooth cycle) является разновидностью V-цикла, в котором отсутствует предварительное сглаживание [25].

Метод Зейделя в традиционном односеточном исполнении подразумевает итерационное решение СЛАУ

$$A_0\hat{\boldsymbol{u}}_0^h = \boldsymbol{b}_0$$

на исходной (самой мелкой) сетке G_1^0. Метод Зейделя в псевдомногосеточном исполнении (УМТ) подразумевает итерационное решение СЛАУ (13) на 3^{dl} сетках уровня l, начиная с уровня L^+ с самыми грубыми сетками и заканчивая исходной (самой мелкой) сеткой G_1^0. При этом представление исходной СЛАУ в виде совокупности 3^{dl} СЛАУ меньшего размера и сборка полученных приближений к решению осуществляется проблемно-независимым образом (свойства 1 и 3). Использование основного многосеточного принципа в односеточном алгоритме позволило снизить трудоемкость метода Зейделя с $\mathcal{W} \sim N^{1+2/d}$ ао до $\mathcal{W} \sim N \log N$ ао фактически без использования проблемно-зависимых компонентов. Как и ожидалось, псевдомногосеточная вычислительная технология для перспективных комплексов программ, устроенных по принципу "черного ящика", оказалось очень простой — МКО, линеаризация (по Ньютону) и метод Зейделя.

Теперь необходимо сделать замечания об адаптивной оптимизации УМТ, т.е. о способности подстраивать отдельные параметры конкретной задачи и/или компоненты алгоритма с целью снижения трудоемкости вычислений и/или повышения точности численного решения. В отдельных приложениях возникает необходимость использования различных параметров, оптимальные значения которых заранее неизвестны. Основная идея адаптивной оптимизации УМТ состоит в проведении дополнительных вычислительных экспериментов на уровнях с грубыми сетками и анализе трудоемкости в зависимости от величин оптимизируемых параметров. В [7] получена следующая оценка увеличения времени счета при адаптивной оптимизации УМТ

$$\frac{T_\Sigma^*}{T_\Sigma} \approx 1 + \frac{\varrho}{L^+ + 1}\,\frac{1}{3^d - 1}\,,$$

где T_Σ — время счета с известными оптимальными значениями проблемно-зависимых компонентов УМТ, T_Σ^* — время счета с учетом затрат на адаптивную оптимизацию УМТ, ϱ — количество дополнительных задач, которые необходимо решить на грубых сетках для адаптивной оптимизации УМТ, L^+ — номер сеточного уровня с грубыми сетками. Очевидно, что в трехмерном случае ($d = 3$) увеличение объема вычислительной работы будет меньше, чем в двухмерном ($d = 2$), поскольку сеточный уровень состоит из большего числа сеток.

Аналогичным образом можно выбрать упорядочение уравнений и неизвестных при решении анизотропных задач, которое обеспечит наиболее высокую скорость сходимости УМТ.

Положим, что некоторая начально-краевая задача может быть решена на каждом временно́м слое традиционным методом Зейделя. Трудоемкость алгоритма на каждом слое составит $\mathcal{W}_0 = CN\nu_0$ арифметических операций, где ν_0 — количество итераций метода Зейделя, необходимых для достижения критерия останова. Аналогично трудоемкость сглаживания составит $\mathcal{W}_l = CN(l+1)\nu$ ао, где $l = 0, 1, 2, \ldots, L^+$ — номер сеточного уровня с которого начинают сглаживание. Трудоемкость УМТ будет меньше, чем трудоемкость метода Зейделя, если

$$\frac{\mathcal{W}_l}{\mathcal{W}_0} = (l+1)\,\frac{\nu}{\nu_0} < 1,$$

откуда номер сеточного уровня, с которого надо начинать сглаживание, будет равен

$$l = \max\left(0; \left[\frac{\nu_0}{\nu} - 1\right]\right),$$

где квадратные скобки означают целую часть числа. Таким образом, если шаг по времени достаточно мал, то УМТ трансформируется в традиционный метод Зейделя.

Таким образом, если традиционный метод Зейделя применим к решению результирующей СЛАУ, полученной в результате аппроксимации краевой задачи на сетке G_1^0, то УМТ (как метод Зейделя в псевдомногосеточном исполнении) применима к решению 3^{dl} СЛАУ, полученных в результате аппроксимации модифицированной краевой задачи на сетке G_1^0. Аппроксимация на многосеточной структуре и межуровневые переходы в УМТ не зависят от решаемой задачи. Теоретически УМТ имеет дополнительную проблемно-зависимую компоненту — количество сглаживающих итераций, выполняемых на грубых сетках ($l > 0$). Практически увеличение количества сглаживающих итераций слабо влияет на общий объем вычислений, поскольку в этом случае уменьшается количество многосеточных итераций. Кроме того, оптимальное количество сглаживающих итераций можно определить по приведенной выше методике адаптивной оптимизации УМТ.

В различных проблемных областях существует ряд задач, решение которых требует чрезмерного времени даже с привлечением самых совершенных процессоров и численных методов. Поэтому неуклонно возрастает интерес к возможности распараллеливания вычислений. В настоящее время еще не утвердились доминирующий алгоритм, предназначенный для параллельного численного решения краевых задач, и доминирующая параллельная архитектура.

УМТ обладает следующими привлекательными свойствами для параллельных вычислений:

1) грубые сетки каждого уровня не имеют общих точек (свойство 2), поэтому сглаживающие итерации на этих сетках могут проводиться параллельно вне зависимости от используемой сглаживающей процедуры;

2) фиксированное количество сеток (3^{dl}) на каждом уровне l позволяет заранее предсказать необходимое число процессоров для оптимального распараллеливания УМТ;

3) отсутствие погрешности интерполяции (свойство 1) позволяет снизить требования, предъявляемые к сглаживающим процедурам, и использовать слабые, но обладающие высоким параллелизмом сглаживатели в многосеточных итерациях УМТ;

4) почти одинаковое количество точек на каждой сетке одного уровня позволяет добиться равномерной загрузки процессоров.

Поскольку каждый уровень l состоит из 3^{dl} сеток ($d = 2, 3$; $l = 0, 1, \ldots, L^+$), количество процессоров должно определяться равенством $p = 3^{d\kappa}$ ($\kappa = 1, 2, \ldots$) для их равномерной загрузки. В дальнейшем величину κ будем называть *глубиной распараллеливания* УМТ. Значение $\kappa = 0$ ($p = 1$) соответствует последовательной УМТ.

Параллельная УМТ обладает следующими свойствами.

1. *Геометрический параллелизм:* $l \geqslant \kappa$ (количество сеток, образующих данные уровни, больше количества процессоров p). Благодаря оригинальной иерархии вычислительных сеток (свойство 2) приближение к решению можно получить посредством решения $p = 3^{d\kappa}$ ($\kappa = 1, 2, \ldots$) независимых задач (без обмена данными) практически одинакового размера, причем для схем второго порядка аппроксимации разница между приближением и решением составит κ значащих цифр. Геометрический параллелизм определяется вычислительной сеткой и не зависит от сглаживателя.

2. *Алгебраический параллелизм:* $l < \kappa$ (количество сеток, образующих данные уровни, меньше количества процессоров p). Алгебраический параллелизм определяется сглаживателем и не зависит от вычислительной сетки. Параллельные варианты метода Зейделя с многоцветными упорядочениями неизвестных подробно описаны в многочисленной литературе [21].

Традиционный способ построения параллельных КММ, основанный на только алгебраическом параллелизме (т.е. на распараллеливании сглаживающей процедуры и операторов переходов), не позволяет достичь высокой эффективности параллелизма из-за уменьшения отношения объема вычислительной работы к объему пересылаемых данных и возможного простоя процессоров на грубых сетках [23]. Именно геометрический параллелизм позволяет выполнять сглаживание на уровнях с грубыми сетками параллельно, причем без обмена данными и независимо от выбора сглаживающей процедуры. Сочетая преимущества геометрического ($l \geqslant \kappa$) и алгебраического ($l < \kappa$) параллелизмов, можно добиться параллельного ускорения УМТ по сравнению с последовательными КММ ($\tilde{\mathcal{S}} > 1$) при $p = 3^{d\kappa}$, соответствующие оценки ускорения и эффективности параллелизма УМТ приведены в [7, 20].

Заметим, что параллельная УМТ эффективно реализуется на многопроцессорных компьютерах с общей памятью с помощью программных средств типа OpenMP, причем все данные хранят в одних массивах, но в силу свойства 2 это не приводит к конфликтным ситуациям. Информационная структура геометрического параллелизма позволяет выполнять независимое сглаживание на сетках уровней $l \geqslant \kappa$ в разных вычислительных потоках (threads), при этом каждый поток обрабатывается на своем процессоре без коммуникационных потерь. Информационная структура алгебраического параллелизма позволяет разделять сглаживающую итерацию на независимые подзадачи на каждой сетке уровней $l < \kappa$, которые обрабатывается в разных вычислительных потоках (threads) на своем процессоре с последующими обменами данными и коммуникационными потерями.

УМТ прошла сложный и тернистый путь своего развития: применение основного многосеточного принципа в односеточном алгоритме, описание односеточного алгоритма как многосеточного и нестандартные псевдомногосеточные компоненты односеточного алгоритма долгое время не встречали должного понимания у специалистов в области вычислительной математики. В настоящее время даже трудно

подсчитать общее количество публикаций по многосеточным методам в мире, однако в УМТ могут быть использованы аналоги следующих подходов из КММ:

1) матричный подход к анализу сходимости, основанный на свойствах сглаживания и аппроксимации [17];

2) сглаживатели Ванки для совместного решения систем уравнений [24];

3) метод вспомогательного пространства для обобщения на неструктурированные сетки [26];

4) метод FAS для решения нелинейных задач [23].

Все остальные компоненты УМТ не имеют аналогов в КММ. Неудивительно, что первая статья по УМТ была опубликована только в 2000 году, спустя 10 лет после первого сглаживания на многосеточной структуре [6].

В завершение кратко сформулируем основные свойства УМТ при решении широкого класса (не)линейных (начально-)краевых задач на глобально структурированных сетках.

1. *Универсальность.* УМТ на основе МКО, метода Ньютона и псевдомногосеточного метода Зейделя практически не содержит новых проблемно-зависимых компонентов по сравнению с традиционным односеточным методом Зейделя. Теоретически количество сглаживающих итераций в УМТ зависит от решаемой задачи, однако это слабо влияет на общую трудоемкость алгоритма. Кроме того, оптимальное количество сглаживающих итераций и оптимальное упорядочение уравнений и неизвестных может быть определено с минимальными издержками в результате адаптивной оптимизации УМТ. Используемое блочное упорядочение неизвестных позволяет унифицированно применять УМТ как к решению уравнения Пуассона, так и к совместному решению нелинейных систем дифференциальных уравнений в частных производных особого вида (уравнения Навье–Стокса для несжимаемых сред). Фактически, единственным проблемно-зависимым компонентом УМТ при решении линейных задач является критерий останова. Для разностных схем второго порядка аппроксимации, записанных в матричной форме (5), может быть рекомендован следующий критерий останова многосеточных итераций УМТ
$$\|A\boldsymbol{x} - \boldsymbol{b}\|_\infty < 10^{-6}.$$

2. *Эффективность.* Трудоемкость УМТ ($\mathcal{W} \sim N \log N$ ао) существенно ниже трудоемкости традиционного метода Зейделя ($\mathcal{W} \sim N^{1+2/d}$ ао), что достигнуто практически без дополнительных проблемно-зависимых компонентов. Проигрыш в трудоемкости КММ ($\sim \log N$ ао) обусловлен минимумом проблемно-зависимых компонентов в УМТ. Трудоемкость многосеточной итерации УМТ при совместном решении систем (не)линейных уравнений возрастает до $O\big((n_b/N)^{1-\varsigma} N \log N\big)$ ао.

3. *Параллелизм.* Для распараллеливания УМТ необходимо $p = 3^{d\kappa}$ ($\kappa = 1, 2, \ldots$) процессоров. Геометрический параллелизм позволяет получить достаточно точное приближение к решению на уровнях с грубыми сетками, причем без обмена данными между процессорами и независимо от используемого сглаживателя. Разница между приближением и решением составит κ значащих цифр для схем второго порядка аппроксимации. Алгебраический параллелизм основан на параллельном методе Зейделя с многоцветными упорядочениями неизвестных: оставшиеся κ значащих цифр уточняют посредством сглаживания на мелких сетках уровней $l < \kappa$ с обменом данными. Сочетание геометрического и алгебраического параллелизма позволяет достичь ускорения параллельной УМТ по сравнению с наилучшим последовательным алгоритмом (11).

4. *Адаптивная оптимизация.* Отдельные параметры конкретной задачи и/или компоненты алгоритма возможно оптимизировать посредством дополнительных вычислительных экспериментов на уровнях с грубыми сетками. Вычислительные издержки оптимизации невелики, особенно в трехмерном случае. Возможность изменения способа и порядка аппроксимации будет показана в следующем разделе.

5. *Минимальное использование ресурсов компьютера.* Реализация УМТ подразумевает использование дополнительной оперативной памяти только для хранения поправок.

Следующим по сложности является случай, когда область Ω представима в виде $\Omega = \bigcup\limits_{k=1}^{K} \Omega_k$, т.е. область Ω является объединением подобластей Ω_k, в каждой из которых возможно построение структурированной сетки G_k^h, однако их объединение $G^h = \bigcup\limits_{k=1}^{K} G_k^h$ не является глобально структурированной

сеткой. Сетки данного типа будут называться локально структурированными. В этом случае возможно применение метода декомпозиции области (domain decomposition) для решения краевых задач [16].

4. Неструктурированные сетки. Теперь рассмотрим наиболее трудный случай, когда вычислительная сетка является неструктурированной. Попытки построения ГММ оказались безуспешными из-за трудностей построения иерархии неструктурированных сеток. Положим, что в некоторой области Ω построена неструктурированная сетка, осуществлена аппроксимация МКЭ нелинейной краевой задачи, выполнена глобальная линеаризация системы нелинейных уравнений и результирующая СЛАУ записана в матричной форме (5). Наиболее эффективной группой итерационных методов, применяемой в настоящее время в линейной алгебре для решения разреженных СЛАУ, являются методы подпространств Крылова с предобусловливанием. К этой группе принадлежат следующие методы: метод сопряженных градиентов (CG), квадратичный метод сопряженных градиентов (CGS), стабилизированный метод бисопряженных градиентов (BiCGSTAB), обобщенный метод минимальных невязок (GMRES) и др. [22]. Однако трудоемкость данных итерационных методов достаточно велика, а для их реализации необходимы существенные ресурсы компьютера. Поэтому методы подпространств Крылова с предобусловливанием используют преимущественно для решения СЛАУ малого размера.

Аналогичные недостатки (необходимость глобальной линеаризации системы нелинейных уравнений, использование существенных ресурсов компьютера и трудности распараллеливания сглаживающих итераций на грубых сетках) характерны и для АММ [23]. Хотя продолжаются интенсивные разработки новых вариантов АММ, очевидно, что начинать решение надо не со СЛАУ, а с исходной нелинейной системы (интегро-)дифференциальных уравнений.

Обобщим УМТ на неструктурированные сетки с учетом возможного изменения порядка и способа аппроксимации исходной системы (не)линейной (интегро-)дифференциальных уравнений. Основным методом обобщения является аналог коррекции дефекта (defect correction) [23] и метода вспомогательного пространств (auxiliary space method) [26]. Запишем исходное нелинейное уравнений $\mathcal{N}(u) = f$ в Σ-модифицированном виде

$$\mathcal{N}(c) = f + \mathcal{N}(c) - \mathcal{N}\big(c + \hat{u}\big),$$

где $\mathcal{N}(0) = 0$. Предположим, что для численного решения поставленной задачи построены вспомогательная структурированная ($G_\square$) и исходная неструктурированная ($G_\triangle$) сетки в области Ω. Граничные узлы вспомогательной сетки $G_\square$, порождающей многосеточную структуру, могут не совпадать с границами области Ω. Методика построения структурированной сетки $G_\square$ приведена в [7, 20]. Левую часть данного уравнения аппроксимируем МКО на многосеточной структуре, а правую часть — на неструктурированной сетке $G_\triangle$ МКЭ или МКО

$$\underbrace{\Big[\mathcal{N}_l^h\big(c_l^h\big)\Big]_\square}_{\text{МКО}} = \mathcal{R}_{0\to l} \prod_{\Delta\to\square} \underbrace{\Big[f^h + \mathcal{N}^h\big(c^h\big) - \mathcal{N}^h\big(c^h + \hat{u}^{\,h}\big)\Big]_\triangle}_{\text{МКЭ или МКО}},$$

индексы $\square$ и $\triangle$ соответствуют структурированной и неструктурированной сеткам, оператор $\prod$ проецирует правую часть $f^h + \mathcal{N}^h\big(c^h\big) - \mathcal{N}^h\big(c^h + \hat{u}^{\,h}\big)$ с неструктурированной сетки $G_\triangle$ на структурированную сетку $G_\square$, а оператор $\mathcal{R}_{0\to l}$ проецирует правую часть $\prod\limits_{\Delta\to\square}\Big[f^h + \mathcal{N}^h\big(c^h\big) - \mathcal{N}^h\big(c^h + \hat{u}^{\,h}\big)\Big]_\triangle$ с сетки $G_\square$ на грубые сетки уровня l. Порядок аппроксимации МКО левой части не выше второго, а порядок аппроксимации правой части МКЭ или МКО может быть произвольным. Основная идея данного метода состоит в следующем: по мере сходимости многосеточных итераций УМТ поправка стремится к нулю ($c_0^h \to 0$); следовательно, $f^h - \mathcal{N}^h\big(\hat{u}^{\,h}\big) \to 0$, т.е. приближение к решению $\hat{u}^{\,h}$ будет стремиться к решению исходной задачи $u^h = \big(\mathcal{N}^h\big)^{-1} f^h$. Таким образом можно гибко изменять не только порядок, но и способ аппроксимации, причем при минимальных изменениях вычислительного алгоритма.

В общем случае УМТ представима в виде следующей последовательности действий.

`do` $l = L^+, 0, -1$

1. Вычисление правой части $f^h + \mathcal{N}^h\big(c^h\big) - \mathcal{N}^h\big(c^h + \hat{u}^{\,h}\big)$ на неструктурированной сетке $G_\triangle$. В начале многосеточной итерации полагаем $c^h = 0$.

2. Проекция правой части на структурированную сетку $G_\square$, т.е. вычисление

$$\prod_{\Delta\to\square} \underbrace{\Big[f^h + \mathcal{N}^h\big(c^h\big) - \mathcal{N}^h\big(c^h + \hat{u}^{\,h}\big)\Big]_\triangle}_{\text{МКЭ или МКО}}.$$

3. Проекция $\prod\limits_{\Delta\to\square}\left[f^h+\mathcal{N}^h\big(c^h\big)-\mathcal{N}^h\big(c^h+\hat{u}^{\,h}\big)\right]_{\triangle}$ с сетки $G_\square$ на грубые сетки уровня l, т.е. вычисление

$$\mathcal{R}_{0\to l}\prod_{\Delta\to\square}\underbrace{\left[f^h+\mathcal{N}^h\big(c^h\big)-\mathcal{N}^h\big(c^h+\hat{u}^{\,h}\big)\right]_{\triangle}}_{\text{МКЭ или МКО}}.$$

4. Выполнение сглаживающих итераций метода Зейделя с блочным упорядочением неизвестных на сетках уровня l.

5. Пролонгация поправки на неструктурированную сетку $G_\triangle$: $c^h_\triangle=\prod\limits_{\square\to\Delta}c^h_\square$.

6. Пролонгация поправки на уровень с более мелкими сетками: $\boldsymbol{c}^\square_{l-1}=\mathcal{P}_{l\to l-1}\boldsymbol{c}^\square_l$.

`end do`

Далее на неструктурированной сетке $G_\triangle$ осуществляют пересчет приближения к решению

$$\hat{u}^{\,h}_\triangle:=\hat{u}^{\,h}_\triangle+c^h_\triangle,$$

где := означает равенство в смысле присвоения, и обнуляют поправку $c^h_\triangle=0$. После чего выполняют сглаживание на неструктурированной сетке $G_\triangle$ при помощи метода Зейделя с блочным упорядочением неизвестных, чтобы уменьшить влияние погрешности интерполяции поправки на точность вычисления приближения к решению, и проверяют критерий останова.

Операторы межсеточных переходов $\prod\limits_{\Delta\to\square}$ и $\prod\limits_{\square\to\Delta}$ являются проблемно-зависимыми компонентами алгоритма. Если использована только одна сетка (т.е. $G_\square\equiv G_\triangle$) для аппроксимации Σ-модифицированной задачи

$$\prod_{\Delta\to\square}=\prod_{\square\to\Delta}=I,$$

то гибкое изменение порядка и типа аппроксимации не требует привлечения проблемно-зависимых компонентов.

Если $\mathcal{N}$ является линейным оператором, то легко выполнить анализ сходимости данного двухсеточного алгоритма. В [7, 20] показано, что если справедливы свойства сглаживания и аппроксимации на неструктурированной сетке $G_\triangle$

$$\left\|A^{-1}_\Delta-\prod_{\square\to\Delta}A^{-1}_\square\prod_{\Delta\to\square}\right\|\leqslant C_{A_\Delta}\|A_\Delta\|^{-1},\tag{15}$$

то данный двухсеточный алгоритм сходится, причем количество "межсеточных" итераций не зависит от параметра дискретизации. Свойства аппроксимации определяет точность интерполяции поправки и невязки. Впрочем, подобные ограничения характерны для ГММ [9, 23].

Трудоемкость одной итерации составит

$$\mathcal{W}=C_q[\mathcal{W}_\square+C_\square N_\square+C_\Delta N_\Delta]\log N_\square+\bar{C}_\Delta N_\Delta+\mathcal{W}_\Delta\text{ ao},$$

где

$$\mathcal{W}_\square=C^*_\square\nu_\square\left(\frac{n^\square_b}{N_\square}\right)^{1-\varsigma_\square}N_\square,\quad\mathcal{W}_\Delta=C^*_\Delta\nu_\Delta\left(\frac{n^\Delta_b}{N_\Delta}\right)^{1-\varsigma_\Delta}N_\Delta,$$

есть трудоемкость сглаживающих итераций на структурированной и неструктурированной сетках, $\nu_\square$ и ν_Δ — количество сглаживающих итераций, $N_\square$ и N_Δ — количество узлов, $n^\square$ и n^Δ_b — количество неизвестных в блоке, C_q, $C_\square$, C_Δ и др. — константы.

Нетрудно показать, что использование вспомогательной структурированной сетки есть предобусловливание исходной СЛАУ

$$A_\Delta\boldsymbol{x}_\Delta=\boldsymbol{b}_\Delta.$$

Предобусловленная СЛАУ имеет вид

$$K^{-1}A_\Delta\boldsymbol{x}_\Delta=K^{-1}\boldsymbol{b}_\Delta,$$

где

$$K^{-1} = A_{\triangle}^{-1} - S_{\triangle}^{\nu}\left(A_{\triangle}^{-1} - \prod_{\square\to\triangle} A_{\square}^{-1} \prod_{\triangle\to\square}\right).$$

Здесь $S_{\triangle}$ есть матрица сглаживающих итераций на неструктурированной сетке $G_{\triangle}$. Если справедливы свойства сглаживания и аппроксимации, то

$$\left\|I - K^{-1}A_{\triangle}\right\| \leqslant C\eta(\nu),$$

где ν — количество сглаживающих итераций на неструктурированной сетке $G_{\triangle}$, $\eta(\nu)$ — монотонно убывающая функция, C — некоторая константа, не зависящая от параметра дискретизации.

Теперь кратко сформулируем основные адаптивные свойства УМТ при решении широкого класса (не)линейных (начально-)краевых задач.

1. *Гибкое изменение порядка и способа аппроксимации.* УМТ позволяет легко изменять порядок и способ аппроксимации решаемой задачи посредством модификации правой части результирующей системы (не)линейных уравнений, что делает возможным использование широкого арсенала численных методов для решения краевых и начально-краевых задач в комплексах программ, устроенных по принципу “черного ящика”.

2. *Коррекция дефекта.* Численное решение нелинейных краевых и начально-краевых задач на неструктурированных сетках не требует глобальной линеаризации, однако межсеточные операторы для обмена данными между исходной неструктурированной сеткой и вспомогательной структурированной сеткой являются проблемно-зависимыми компонентами технологии. Если интерполяция достаточно точна (15), то трудоемкость алгоритма будет близка к оптимальной.

5. Заключение. УМТ является доведенной до совершенства вычислительной технологией для последующего применения в программных продуктах, устроенных по принципу “черного ящика”. Однако предел этого совершенства ограничен топологией вычислительной сетки.

1. *Глобально структурированные сетки.* Если вычислительная сетка является глобально структурированной, то УМТ содержит минимальное количество проблемно-зависимых компонентов (упорядочение неизвестных, обеспечение сходимости итераций по нелинейности и критерий останова многосеточных итераций), обладает близкой к оптимальной трудоемкостью $\left(O\left((n_b/N)^{1-\varsigma} N\log N\right)\right.$ ао в широком диапазоне параметров задачи), высоким параллелизмом (ускорением по сравнению с наилучшим последовательным алгоритмом), адаптивностью (возможностью гибкого изменения способа и порядка аппроксимации) и минимальным использованием ресурсов компьютера.

2. *Локально структурированные сетки.* Если вычислительная сетка является локально структурированной, то дополнительные трудности в реализации и увеличение трудоемкости УМТ обусловлены необходимостью декомпозиции области.

3. *Неструктурированные сетки.* Если вычислительная сетка является неструктурированной, то дополнительные трудности в реализации и увеличение трудоемкости УМТ обусловлены необходимостью использования вспомогательной структурированной сетки. В данном случае УМТ содержит два дополнительных проблемно-зависимых компонента, связанных с обменом данными между исходной неструктурированной и вспомогательной структурированной сетками.

УМТ является комплексным решением проблемы “универсальность–эффективность–параллелизм”, причем начиная с исходной (интегро-)дифференциальной задачи, а не с результирующей СЛАУ. С точки зрения практических приложений, в настоящее время наиболее важной проблемой является формализация построения структурированных сеток в областях со сложной геометрией.

6. Благодарности. Исследовательские работы проводились при финансовой поддержке РНФ по соглашению № 15–11–30012 от 08.07.2015 по теме “Суперкомпьютерное моделирование физико-химических процессов в высокоскоростном прямоточном воздушно-реактивном двигателе гиперзвукового летательного аппарата на твердых топливах”. Автор выражает глубокую признательность профессору М. П. Галанину за проявленный интерес к работе.

СПИСОК ЛИТЕРАТУРЫ

1. *Бахвалов Н.С., Жидков Н.П., Кобельков Г.М.* Численные методы. М.: Бином, 2007.
2. *Васильев В.А., Калмыкова М.А.* Анализ и выбор программных продуктов для решения инженерных задач приборостроения // Современная техника и технологии. 2013. № 3. URL: http://technology.snauka.ru/2013/03/1702.
3. *Галанин М.П., Савенков Е.Б.* Методы численного анализа математических моделей. М.: Изд-во МГТУ им. Н.Э. Баумана, 2010.
4. *Ильин В.П.* Математическое моделирование. Часть 1. Непрерывные и дискретные модели. Новосибирск: Изд-во СО РАН, 2017.
5. *Калиткин Н.Н.* Численные методы. М.: Наука, 1978.
6. *Мартыненко С.И.* Универсальная многосеточная технология для численного решения дифференциальных уравнений в частных производных на структурированных сетках // Вычислительные методы и программирование. 2000. **1**. 83–102.
7. *Мартыненко С.И.* Многосеточная технология: теория и приложения / Под. ред. М.П. Галанина. М.: Физматлит, 2015.
8. *Марчук Г.И.* Методы вычислительной математики. М.: Наука, 1989.
9. *Ольшанский М.А.* Лекции и упражнения по многосеточным методам. М.: Физматлит, 2005.
10. *Самарский А.А., Гулин А.В.* Численные методы. М.: Наука, 1989.
11. *Самарский А.А., Михайлов А.П.* Математическое моделирование: Идеи. Методы. Примеры. М.: Физматлит, 2002.
12. *Токталиев П.Д., Мартыненко С.И., Яновский Л.С., Волохов В.М., Волохов А.В.* Особенности окисления модельного углеводородного топлива в канале под воздействием электростатического поля // Изв. АН. Сер. хим. 2016. № 8. 2011–2017.
13. *Турчак Л.И.* Основы численных методов. М.: Наука, 1987.
14. *Федоренко Р.П.* Релаксационный метод решения разностных эллиптических уравнений // Журн. вычисл. матем. и матем. физ. 1961. **1**, № 5. 922–927.
15. *Дэннис Дж., Шнабель Р.* Численные методы безусловной оптимизации и решения нелинейных уравнений. М.: Мир, 1988.
16. *Dolean V., Jolivet P., Nataf F.* An introduction to domain decomposition methods: algorithms, theory, and parallel implementation. Philadelphia: SIAM Press, 2015.
17. *Hackbusch W.* Multi-grid methods and applications. Berlin: Springer, 1985.
18. *Hackbusch W.* Robust multi-grid methods, the frequency decomposition multi-grid algorithm // Notes on Numerical Fluid Mechanics. Vol. 123. Braunschweig: Vieweg, 1989. 96–104.
19. *Хейгеман Л., Янг Д.* Прикладные итерационные методы. М.: Мир, 1986.
20. *Martynenko S.I.* The robust multigrid technique: For black-box software. Berlin: De Gruyter, 2017.
21. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. М: Мир, 1991.
22. *Саад Ю.* Итерационные методы для разреженных линейных систем. М.: Изд-во Моск. ун-та, 2013.
23. *Trottenberg U., Oosterlee C.W., Schüller A.* Multigrid. London: Academic Press, 2001.
24. *Vanka S.P.* Block-implicit multigrid solution of Navier–Stokes equations in primitive variables // J. Comput. Physics. 1986. **65**, N 1. 138–158.
25. *Wesseling P.* An introduction to multigrid methods. Chichester: Wiley, 1992.
26. *Xu J.* The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids // Computing. 1996. **56**. 215–235.

Поступила в редакцию
14.04.2019

Numerical Methods for Black Box Software

S. I. Martynenko [1, 2]

[1] *Baranov Central Institute of Aviation Motors; ulitsa Aviamotornaya 2, Moscow, 111116, Russia; Dr. Sci., Scientist, e-mail: Martynenko@ciam.ru*

[2] *Institute of Problems of Chemical Physics, Russian Academy of Sciences; prospekt Semenova 1, Chernogolovka, Moscow region, 142432, Russia; Dr. Sci., Senior Scientist, e-mail: Martynenko@icp.ac.ru*

Received April 4, 2019

Abstract: A number of requirements are formulated to the numerical algorithms for black box software intended for mathematical modeling in continuum mechanics. An analysis of applied properties of the classical multigrid methods and robust multigrid technique in the framework of "robustness-efficiency-parallelism"problem is performed. It is shown that a close-to-optimal complexity with the least number of problem-dependent components and high parallel efficiency can be achieved with the robust multigrid technique on globally structured grids. Application of unstructured grids requires the accurate definition of two problem-dependent components (intergrid operators) that strongly affect on the complexity of an algorithm.

Keywords: parallel and high performance computing, boundary value problems, multigrid methods, black box software.

References

1. N. S. Bakhvalov, N. P. Zhidkov, and G. M. Kobel'kov, *Numerical Methods* (Binom, Moscow, 2007) [in Russian].

2. V. A. Vasilev and M. A. Kalmykova, "Analysis and Selection of Software Products for Instrument Engineering," Sovremen. Tekhnika Tekhnolog. (2013). http://technology.snauka.ru/2013/03/1702. Cited May 7, 2019.

3. M. P. Galanin and E. B. Savenkov, *Procedures of Numerical Analysis of Mathematical Models* (Bauman Gos. Tekh. Univ., Moscow, 2010) [in Russian].

4. V. P. Il'in, *Mathematical Modeling,* Part I: *Continuous and Discrete Models* (Ross. Akad. Nauk, Novosibirsk, 2017) [in Russian].

5. N. N. Kalitkin, *Numerical Methods* (Nauka, Moscow, 1978) [in Russian].

6. S. I. Martynenko, "Robust Multigrid Technique for Solving Partial Differential Equations on Structured Grids," Vychisl. Metody Programm. **1**, 83–102 (2000).

7. S. I. Martynenko, *Multigrid Technology: Theory and Applications* (Fizmatlit, Moscow, 2015) [in Russian].

8. G. I. Marchuk, *Methods of Numerical Mathematics* (Nauka, Moscow, 1989; Springer, New York, 1982).

9. M. A. Ol'shanskii, *Lectures and Exercises on Multigrid Methods* (Fizmatlit, Moscow, 2005) [in Russian].

10. A. A. Samarskii and A. V. Gulin, *Numerical Methods* (Nauka, Moscow, 1989) [in Russian].

11. A. A. Samarskii and A. P. Mikhailov, *Mathematical Modeling: Ideas, Methods, Examples* (Fizmatlit, Moscow, 2002) [in Russian].

12. P. D. Toktaliev, S. I. Martynenko, L. S. Yanovskiy, et al., "Features of Model Hydrocarbon Fuel Oxidation for Channel Flow in the Presence of Electrostatic Field," Izv. Ross. Akad. Nauk, Ser. Khimich., No. 8, 2011–2017 (2016) [Russ. Chem. Bull. **65** (8), 2011–2017 (2016)].

13. L. I. Turchak, *Fundamentals of Numerical Methods* (Nauka, Moscow, 1987) [in Russian].

14. R. P. Fedorenko, "A Relaxation Method for Solving Elliptic Difference Equations," Zh. Vychisl. Mat. Mat. Fiz. **1** (5), 922–927 (1961) [USSR Comput. Math. Math. Phys. **1** (4), 1092–1096 (1962)].

15. J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Englewood Cliffs, 1983; Mir, Moscow, 1988).

16. V. Dolean, P. Jolivet, and F. Nataf, *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation* (SIAM Press, Philadelphia, 2015).

17. W. Hackbusch, *Multi-Grid Methods and Applications* (Springer, Berlin, 1985).

18. W. Hackbusch, "Robust Multi-Grid Methods, the Frequency Decomposition Multi-Grid algorithm," in *Notes on Numerical Fluid Mechanics* (Viewig, Braunschweig, 1989), Vol. 123, pp. 96–104.

19. L. A. Hageman and D. M. Young, *Applied Iterative Methods* (Academic Press, New York, 1981; Mir, Moscow, 1986).

20. S. I. Martynenko, *The Robust Multigrid Technique: For Black-Box Software* (De Gruyter, Berlin, 2017).

21. J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems* (Springer, New York, 1988; Mir, Moscow, 1991).

22. Y. Saad, *Iterative Methods for Sparse Linear Systems* (SIAM, Philadelphia, 2003; Mosk. Gos. Univ., Moscow, 2013).

23. U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid* (Academic Press, London, 2001).

24. S. P. Vanka, "Block-Implicit Multigrid Solution of Navier–Stokes Equations in Primitive Variables," J. Comput. Phys. **65** (1), 138–158 (1986).

25. P. Wesseling, *An Introduction to Multigrid Methods* (Wiley, Chichester, 1992).

26. J. Xu, "The Auxiliary Space Method and Optimal Multigrid Preconditioning Techniques for Unstructured Grids," Computing **56**, 215–235 (1996).