ОБ ОБЪЕКТНО-ОРИЕНТИРОВАННОМ ПОДХОДЕ К РАЗРАБОТКЕ ЧИСЛЕННОГО МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ

В.А. Семенов

В работе рассматриваются общие аспекты применения ООП к математическому обеспечению, ориентированному на решение задач вычислительного характера. Проводится объектный анализ вычислительной проблематики, а также описывается возможная организация математической объектно-ориентированной библиотеки, допускающей унифицированную разработку программных приложений в разных научных областях. Основные концептуальные положения, принятые при построении библиотеки, иллюстрируются парадигмами объектно-ориентированного математческого программирования.

1. Введение

Объектно-ориентированный подход (ООП), получивший широкое распространение как мощная программная технология, составляет в настоящее время весомую альтернативу традиционным процедурным методам [1]. Популярность ООП во многом обусловлена концептуальной целостностью и более сильной формой структуризации программного обеспечения (ПО), разрабатываемого на его основе. В конечном итоге использование ООП выражается в более быстрой и надежной разработке программ, а также в возможности гибкой и естественной модификации существующего ПО при создании новых программных приложений. Данные обстоятельства оказываются решающими в условиях динамично развивающихся отраслей программирования.

С конца 1990 года ведутся достаточно интенсивные исследования в области создания объектно-ориентированных операционных систем, баз данных, графических интерфейсов [2]. В значительно меньшей мере внимания уделяется вопросам применения объектных технологий к математическому обеспечению, ориентированному на решение задач вычислительного характера.

До сих пор наиболее употребительным языком для программ, связанных с решением численных задач, считается Фортран. Значимое многообразие высокоинтеллектуальных математических и прикладных библиотек, созданных на этом языке в течение трех последних десятилетий, а также несомненные традиции пакетной деятельности являются существенными факторами, сдерживающими широкое внедрение объектных технологий в данную область.

Настоящая работа посвящена анализу общих аспектов применения ООП к разработке математического обеспечения. Излагается взгляд автора на организацию математической объектно-ориентированной библиотеки, допускающей унифицированную разработку программных приложений в разнообразных областях науки и техники. Основные принципы организации библиотеки иллюстрируются примерами объектного программирования задач нелинейной оптимизации, занимающих важное место в теории численных методов.

2. Система классов вместо пакетов прикладных программ

Традиционная организация прикладного ПО в виде отдельных программ, библиотек, пакетов и интегрированных систем соответствует прежде всего проблемной и процедурной ориентации технологии разработки, опирающейся на идеи структурного программирования и программной декомпозиции. Все перечисленные виды ПО, отличаясь по существу степенью интеграции и условной масштабностью решаемых задач, отражают общую концепцию построения ПО на основе выделения хорошо структурируемых и самостоятельно значимых программных модулей — программных единиц, выполняющих определенную последовательность операций над данными и решающими некоторые независимые подзадачи.

При таком подходе оказывается довольно проблематичной модификация уже существующего ПО и его адаптация к новым программным приложениям. Рассмотрим наиболее важные особенности процедурной, а затем и объектной реализации математического обеспечения, формально следуя привычной технологии использования универсальных математических библиотек, таких как NAG, IMSL, при создании пакета прикладных программ (ППП), предназначенного, например, для конечно-элементного анализа и по функциональным возможностям подобного известным программам PAFEC и SPICE.

Первым препятствием к интеграции ПО является необходимость унификации всех внутренних форматов данных приложения, которые будут использоваться импортируемыми библиотечными модулями. Это не всегда возможно, поскольку часто оказывается важным расширить методический репертуар одной универсальной библиотеки специализированными средствами из других библиотек, например, итеративными методами решения задач линейной алгебры из пакета ITPACK, методами условной нелинейной оптимизации из MINPACK или методами вычисления квадратур из QUADPACK и т. п. (необходимые ссылки на упомянутые библиотеки и пакеты можно найти, например, в обзоре математического обеспечения [3]).

Единственной возможностью формального объединения модулей в этом случае является одновременная избыточная поддержка нескольких эквивалентных представлений данных на основе их предварительного тщательного изучения и программирования. Заметим, что даже в случае удачно специфицируемых форматов разных библиотек разработчик все равно необходимо поддерживает все внутренние данные, имея, естественно, непосредственный доступ к последним. Данная дисциплина часто является источником ошибок и, безусловно, не может считаться удовлетворительной для больших программных проектов.

Другим не менее существенным недостатком процедурного подхода в данном случае оказывается низкая модифицируемость программных модулей. Даже незначительное изменение одной алгоритмической ветви в вычислительном модуле обычно требует детального ознакомления с организацией всего вычислительного процесса, со структурой всей программы и часто сопряжено со значительными усилиями по перепрограммированию. С учетом крайне низкой наглядности и выразительности процедурных языковых средств такая техника доступна только высококвалифицированным специалистам как в области вычислительных методов, так и программирования.

Типичными примерами подобных модификаций могут служить, например, изменение стратегии выбора главного элемента в методах факторизации, когда существует дополнительная априорная информация о матричной обусловленности или разреженности, или усиление условий остановки итеративного процесса для более надежного контроля сходимости в нелинейных алгебраических задачах, или изменение эвристического сценария выбора шага при численном интегрировании дифференциальных уравнений определенного сорта и т. п.

Важно заметить, что ориентация только на процедурную завершенность не способствует внутренней структуризации программ и чрезвычайно затрудняет их модификацию и использование. Применение же концепции ООП для разработки математического обеспечения представляется чрезвычайно привлекательным. Система классов или, что то же самое, объектная библиотека, являющаяся фундаментальной формой объектно-ориентированного ПО, оказывается мощным и выразительным средством реализации математических библиотек и прикладных вычислительных систем.

Обсудим главные возможности, предоставляемые объектной технологией при реализации вычислительного математического обеспечения.

Прежде всего, инкапсуляция методов, в данном случае численных методов, в классах математических объектов является более сильной формой структуризации вычислительных модулей, повышающей концептуальную наглядность применяемых численных подходов и регламентирующей корректную дисциплину работы с данными объектов без нарушения их целостности. Разработчик прикладного ПО выступает в качестве пользователя библиотечных классов и вместо поддержки внутренних данных может сконцентрировать усилия на предметном аспекте разрабатываемых программ. Принимая во внимание сложность динамически размещаемых данных математических объектов, в частности, разреженных матричных и функциональных объектов, данное преимущество кажется вполне ощутимым.

Механизм защиты данных, обеспечиваемый объектной технологией, также способствует повышению надежности разрабатываемых приложений.

Наконец, наследование математических классов и полиморфизм численных методов, инкапсулируемых ими, предоставляют важную для многих численных приложений возможность гибкой модернизации и развития математического обеспечения как с учетом проблемной ориентации, так и в соответствии с детальной и строгой классификацией объектов, выделенных по их математическим и вычислительным признакам и свойствам. Реализация численных методов в виде наследуемых методов родственных математических классов позволяет естественным образом выразить меру общности математических понятий и степень универсальности применения тех или иных численных подходов в каждом конкретном случае.

Базовые объектные парадигмы, используемые при реализации математической библиотеки, порождают качественно новую технологию разработки прикладного ПО. Поскольку объектный подход обеспечивает мощную и гибкую технологию развития уже существующего программного обеспечения, а математические классы выражают самые общие проблемно-инвариантные понятия, математическая объектно-ориентированная библиотека может рассматриваться в качестве базовой инструментальной среды для разработки разнообразных программных приложений.

Во-первых, теряется необходимость начинать новую проблемную разработку с нуля. Базовые классы типа векторов, списков, таблиц, матриц и т. п., с необходимостью используемые самой математической библиотекой и оформленные в виде соответствующих абстрактных обобщенных классов или их шаблонов, могут быть непосредственно использованы в предметных модулях прикладной программы. Инструментальный характер базовых классов тем более осознается при необходимости конкретизации какого-либо вида объектов и переопределения группы методов.

Во-вторых, основные математические классы, выражающие мощные концептуальные понятия, образуют своеобразную языковую среду, используя которую, пользователь может программировать проблемную часть практически в терминах, общепринятых в математических и вычислительных науках.

В-третьих, расширением множества математических понятий на предметную сферу и соответствующей классовой поддержкой последних достигается реализация прикладной системы классов с унифицированным модифицируемым математическим ядром. Специфическая для проблемной области система понятий, свойств, отношений и т. п. находит свое отражение в системе классов, построенной по принципу иерархической детализации. Поскольку аппарат понятий может все более уточняться, и тем самым приобретать все более мощное содержание, удается выразить проблематику приложения в терминах конечного пользователя — инженера, специализирующегося в соответствующей предметной области.

Опыт подобных разработок показывает, что не представляет большого труда эмулировать входные языки вычислительных ППП соответствующими объектно-ориентированными средствами, достигая при этом даже преемственности синтаксических конструкций. Таким образом, прикладная система классов наряду со своим функциональным назначением может обеспечивать поддержку текстового интерфейса, необходимую для конечного пользователя, и избавлять тем самым разработчика от реализации транслятора входного языка, столь свойственного ППП.

При реализации интегрированной программной системы, содержащей необходимые операционные средства, графический пользовательский интерфейс, базу данных, может использоваться та же прикладная система классов в результате придания классифицированным сущностям соответствующего операционного, графического и информационного содержания. Хотя в настоящее время объектно-ориентированные базы данных, графические интерфейсы развиваются как самостоятельные программные продукты, возможно, механизм множественного наследования обеспечит требуемый уровень интеграции ПО при создании законченных программных систем.

Таким образом, реализация математического обеспечения в виде библиотеки или системы классов имеет существенные преимущества над традиционной процедурно-ориентированной формой ППП.

3. Общие принципы разработки объектно-ориентированного математического обеспечения

Обсудим общий подход к организации математического обеспечения, следуя условной схеме проектирования объектно-ориентированного ПО, представляемой пятью этапами:

— выбор объектов и определение их классов,

— идентификация атрибутов объектов и определение данных классов,

— идентификация свойств объектов и операций над ними и определение методов классов,

— выделение общностей объектов, установление отношений наследования между ними и организация классовых иерархий с использованием полиморфизма методов,

— анализ взаимодействия между объектами, тестирование сценария работы с системой классов, возможное их критическое переосмысление и, как следствие, возврат к предыдущим стадиям разработки.

Применение объектного подхода предполагает прежде всего проведение предварительного объектного анализа рассматриваемой проблемной области. Результатом такого анализа должна стать система общностей, выражающая основные предметные понятия, их свойства и устанавливающая необходимые классификационные отношения между ними. Объектный анализ вычислительной математики показывает, что выбор такой системы объектов крайне неоднозначен и в значительной степени определяется субъективным представлением о той роли, которую играют выделенные общности в самой математике и в тех концептуальных построениях, на которых основывается теория численных методов [4, 8, 10].

Неотъемлемой составной частью объектного анализа является выявление возможных отношений наследования между объектами, при которых множество классов может представляться единой целостной иерархией родственных объектов. Естественно, в основе построения любой подобной иерархии лежит та или иная классификационная стратегия, идентифицирующая каждый объект в соответствии с принятой системой признаков. Поскольку способы выделения математических и вычислительных признаков допускают широкое многообразие, при организации классовых математических иерархий всегда будем исходить из тех вычислительно-конструктивных свойств объектов, которые определяют возможность эффективного применения численных подходов к наиболее важным и распространенным постановкам рассматриваемых проблем.

В этом случае любой вычислительный алгоритм, применимый к объектам некоторого математического класса, также может быть программно применен и к объектам любого наследуемого частного класса, причем с высокой равномерной эффективностью на множествах эквивалентных задач, различающихся только типами математических объектов. При этом может быть обеспечена полезная дисциплина контроля корректности применения численных методов к частным математическим постановкам.

Вместе с тем, всякий раз, когда наличие специальных математических свойств объекта, принятых в качестве классификационных критериев, допускает применение более эффективного вычислительного подхода, соответствующий метод общего класса может быть полиморфно переопределен. Тем более это становится оправданным, если математические особенности частного объекта позволяют заменить вычислительную процедуру соответствующим аналитическим преобразованием. В определенном смысле данная парадигма может рассматриваться как средство синтеза основных подходов вычислительной математики с известными теоретическими результатами математической науки.

Рассмотренный аспект применения объектной технологии в равной степени распространяется и на структуры данных алгоритмов, зачастую определяющие эффективность вычислительных процедур. Поскольку ООП предполагает возможность определения абстрактных классов, сложные вычислительные алгоритмы, будучи редуцированными к базовым операциям, могут быть реализованы на абстрактном уровне без конкретизации форматов данных. В тех случаях, когда требования к эффективности высоки и частое использование виртуальных функций нежелательно, алгоритм всегда может быть целиком перепрограммирован и под конкретную структуру данных.

По-видимому, наиболее выверенным способом эффективной реализации математических классов является дисциплина, при которой вычислительно мощные методы реализуются непосредственно на конкретной структуре данных, а их виртуальное использование переносится на абстрактный уровень. В этом случае затраты на вызовы виртуальных функций будут пренебрежимо малы по сравнению с объемом выполненной

вычислительной работы, а сложно организованные вычислительные процессы окажутся реализованными уже на самых верхних общих уровнях классовой иерархии.

Таким образом, наследование и полиморфизм численных методов, инкапсулируемых соответствующими математическими классами, обеспечивает важный для многих проблемных приложений компромисс между необходимостью иметь надежное функционально полное и унифицированное алгоритмическое ядро и возможностью замещения универсальных робастных методов на эффективные алгоритмические версии, применимые только в частных математических случаях.

4. Объектная триада вычислительной математики

Поиск объектов, которые могли бы составить базовое классовое ядро объектной математической библиотеки, в данном случае неоднозначен. На наш взгляд объектная триада «математический объект — вычислительный алгоритм — численная проблема» является фундаментальной и определяет достаточно общую концепцию объектной реализации математического и прикладного ПО (см. рис. 1). Поскольку данные виды объектов выражают основные конструктивные понятия вычислительной математики, соответствующая классовая поддержка обеспечивает желаемую общность математических методов и программных средств решения вычислительных задач.

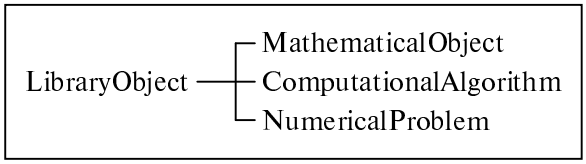

Рис. 1. Объектная вычислительная триада

Под математическим объектом MathematicalObject будем понимать самостоятельную сущность, выражающую некоторую математическую категорию и составляющую объект вычислений. Математическими объектами являются целые, вещественные и комплексные числа. К ним будем относить также их множества, векторы, матрицы, функции, последовательности, ряды, области пространств, кривые, поверхности и т. п. Каждый математический объект обладает набором математических признаков, являющихся основой для классификационных построений. Заметим, что сами по себе математические объекты еще не составляют вычислительной задачи и являются лишь инструментальным средством для ее постановки и решения.

Под вычислительным алгоритмом ComputationalAlgorithm будем понимать, собственно, методы вычислительной математики и некоторую вспомогательную информацию, определяющую условия их алгоритмического использования. Будем считать, что каждый алгоритм предназначен для решения строго одной проблемы, хотя и может использоваться косвенно при решении задач разного сорта. Например, любой метод интерполяции может служить алгоритмической основой для построения семейства квадратурных формул. Однако для численного интегрирования, кроме интерполяционного метода, необходимо определить методы оценок погрешности приближения и устойчивости, адаптивную стратегию выбора шага и многое другое. Поэтому всякий раз вычислительный алгоритм будем связывать с одной единственной проблемой. Обратное, конечно, неверно, поскольку для решения численных проблем одного класса могут использоваться разные подходы.

Кроме параметров численного метода алгоритмические объекты содержат необходимую информацию о требуемой точности, о располагаемых пользователем вычислительных ресурсах, которые выражаются обычно предельным числом итераций, максимально допустимым количеством вычислений функций и их производных и т. п. Данная информация определяет конкретные условия организации вычислительного процесса и всегда необходимо используется в математическом обеспечении.

Определим, наконец, объекты типа численная проблема NumericalProblem. Под численной проблемой будем понимать классическую задачу вычислительной математики, представленную в стандартной унифицированной форме. В рамках обсуждения универсальной математической библиотеки имеет смысл, естественно, ограничиться именно

таким кругом задач, хотя методологически такой взгляд может быть перенесен и на прикладную сферу.

Примерами численных проблем могут служить системы линейных уравнений, проблема собственных значений, задачи линейного и квадратичного программирования. Среди нелинейных алгебраических проблем наиболее важными являются системы уравнений, задачи условной и безусловной оптимизации. Среди дифференциальных проблем выделим задачи Коши, краевые задачи для обыкновенных дифференциальных уравнений, интегральные уравнения и уравнения в частных производных. Естественно, приведенные примеры составляют лишь самое общее перечисление вычислительной проблематики и нуждаются в разработке более тщательных классификаций, обсуждаемых в следующих разделах.

Таким образом, определены три основных вида вычислительных объектов и, соответственно, три связанных с ними способа инкапсуляции численных методов как методов математических, алгоритмических и проблемных классов:

Solution=Object.SolveProblem(Object2, …, Algorithm),
Solution=Algorithm.ApplyTo(Object1, …),
Solution=Problem.SolveBy(Algorithm).

В первых двух вариантах задается множество математических объектов, определяющих корректную постановку задачи, а в первом и третьем варианте — алгоритм решения, иногда отсутствующий в спецификации и используемый по умолчанию.

Введение специальных проблемных классов может показаться избыточным. Тем не менее, на наш взгляд, наряду с целесообразностью определения самостоятельной классовой общности существуют и другие весомые предпосылки для организации подобных классов.

Во-первых, поскольку постановка и решение численной задачи выражается в определении соответствующих математических и алгоритмических объектов, организация проблемных классов в виде системы ассоциативных ссылок на данные объекты обеспечивает желаемую общность программной реализации близких постановок задач, отличающихся только типами математических объектов. Например, для постановки задачи условной оптимизации необходимо определить оптимизируемый функционал, его область определения, систему условий типа равенств и неравенств, иногда и начальное приближение. Для корректной постановки дифференциальной задачи необходимо задать сами дифференциальные уравнения, систему начальных и граничных условий и т. п.

Во-вторых, для решения одной и той же проблемы могут применяться разные численные алгоритмы и наряду с решением задачи часто оказывается важным иметь информацию о корректности и эффективности использования алгоритма в каждой конкретной ситуации. Естественнее выглядит реализация, при которой такая информация ассоциируется с решаемой проблемой, а не с используемыми объектами вычислений.

Более того, наличие такой информации в проблемном классе позволяет организовать экспертную оценку эффективности применения алгоритмов к частным задачам. Поскольку данный аспект часто оказываются предметом серьезных научных исследований, программные экспертные средства должны иметь достаточно мощное интеллектуальное наполнение, объединяющее фундаментальные теоретические результаты и практические знания о возможных реализациях широких алгоритмических многообразий. Полиморфизм методов математических и алгоритмических классов может служить полезным механизмом, обеспечивающим автоматический выбор необходимой алгоритмической версии в соответствии со свойствами объектов, принятых в качестве классификационных критериев.

В-третьих, довольно часто в приложениях вычислительные процедуры используются в определенном контексте с операционной ситуацией ЭВМ, характеризуемой наличием свободных ресурсов памяти, времени и т. п. Поскольку объекты численных проблем имеют большее время жизни, чем базовые математические объекты, гораздо проще управлять вычислительными ресурсами именно на этом уровне. Например, при многократном решении системы линейных уравнений с изменяемой правой частью или малоранговыми матричными модификациями и при наличии необходимых ресурсов памяти целесообразно хранить и использовать промежуточное представление матричной факторизации при последующих решениях системы вместо инициации полного вычислительного цикла. Анализ операционной ситуации и поддержку внутренних представлений промежуточных данных следует отнести к функциям проблемных классов. Реализация же их в математических классах выглядит излишне громоздкой.

В дальнейшем реализацию методов решения сложных вычислительных проблем будем связывать с организацией специальных проблемных классов. В тех случаях, когда задача носит вспомогательный характер и определяет лишь базовые операции математического или алгоритмического объекта с детерминированной сложностной оценкой, введение дополнительного проблемного объекта лишь усложнит программирование.

5. Общая структура математической библиотеки

Главным требованием, предъявляемым к организации библиотеки, по-видимому, является возможность непосредственного создания на ее основе различных программных приложений, что необходимо приводит к дальнейшему развитию и модернизации ее проблемного и методического репертуара. Объектная технология в данном случае предоставляет важные возможности для такой реализации при определении библиотечного класса и использовании парадигм наследования и полиморфизма. Рассмотрим более подробно возможную структуру математической объектно-ориентированной библиотеки, изображенную на рис. 2, не конкретизируя частные вычислительные аспекты.

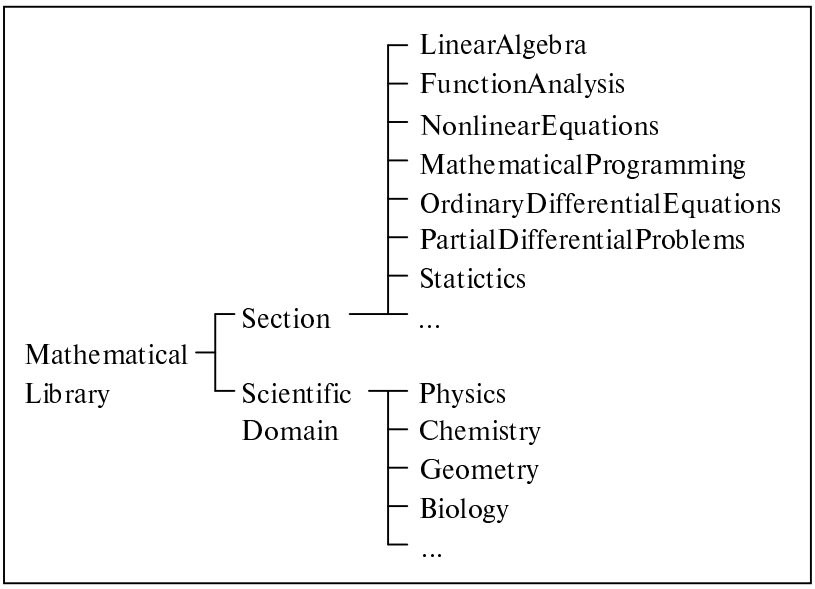

Рис. 2. Структура объектно-ориентированной математической библиотеки

Вершиной предлагаемой объектной иерархии является класс MathematicalLibrary, выражающий наибольшую общность выделенных объектов, и который удобно ассоциировать с атрибутами и системными средствами самой библиотеки. Например, статические члены суперкласса могли бы содержать необходимые для большинства приложений машинно-зависимые и математические константы, а методы — реализовать общесистемные функции библиотеки: обработку ошибок, сбор статистики, поддержку стандартных потоков ввода/вывода, информационную помощь и операционные средства управления вычислительными ресурсами на пользовательском уровне.

Непосредственными наследниками библиотечного класса являются классы разделов самой математической библиотеки Section и классы основных научных направлений ScientificDomain, в рамках которых планируется создание программных приложений. Классы разделов и областей имеют то же назначение, что и библиотечный класс. Определение же их связано с необходимостью введения дополнительных системных атрибутов и средств, специфичных для отдельных разделов вычислительной математики и проблемных областей и не удовлетворяющих общим концептуальным построениям и системным сценариям, принятым для библиотечного суперкласса.

Естественно, с учетом необходимости расширения самой библиотеки и разработки на ее основе новых приложений всякие попытки полной унификации подобных программных средств для достаточно самостоятельных направлений прикладных наук выглядят довольно сомнительными. Тем не менее, тщательное проектирование библиотечного математического класса, инвариантного по отношению к различным проблемным областям, позволяет

надеяться, что в большинстве практически важных случаев можно ограничиться только разработанными системными средствами. В других ситуациях всегда возможно доопределение или переопределение подобных унифицированных средств.

Организация классов Section в определенной степени отражает принятую проблемную классификацию разделов вычислительной математики [4, 8, 10].

Классы научной ориентации ScientificDomain реализуют общие методические концепции, характерные для соответствующих научных областей. Например, методы физического класса Physics могли бы поддерживать концепцию моделирования Кирхгофа и реализовывать технику конечно-элементного анализа. С учетом того, что данные подходы применимы к широким классам физических проблем, включая электронное моделирование, гидро- и аэродинамику, прочностной анализ, физику полупроводников и т. п., класс Physics нуждается в дальнейшем уточнении и организации специализированных библиотек классов математических моделей элементов (см. рис. 3). Естественно, разработка таких библиотек может осуществляться унифицированным образом, но для каждого физического приложения отдельно.

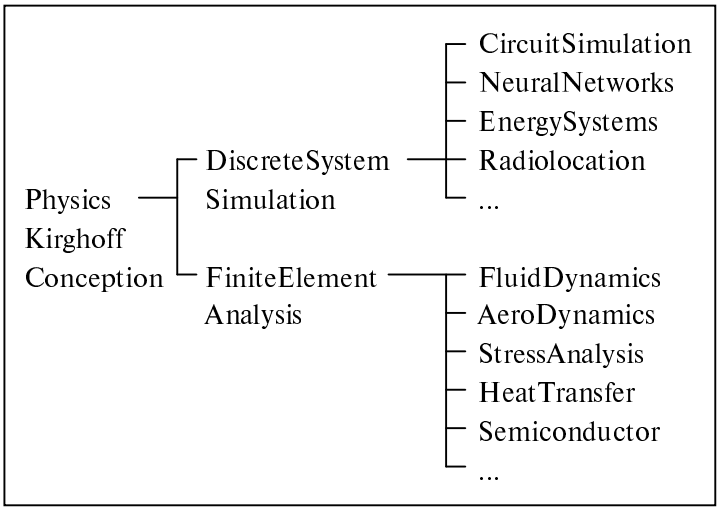

Рис. 3. Объектная организация физических приложений

Рассмотренные библиотечные классы реализуют самые общие понятия и составляют верхнюю часть классовой иерархии, которая продолжается наследуемыми объектами фундаментальной вычислительной триады. При этом наследуемые математические, алгоритмические и проблемные классы образуют самостоятельные подиерархии в соответствии с математическими и вычислительными признаками объектов, принятыми в качестве классификационных.

Может показаться странным, что внутренняя организация математической библиотеки в большей степени отражает проблемную ориентацию разработки, а не объектную. Данное обстоятельство объясняется тем, что сами проблемы и алгоритмы их решения также являются вычислительными объектами, происхождение которых строго определяется соответствующими разделами математической библиотеки. При этом способ наследования математических классов не всегда однозначен, поскольку для постановки и решения любой вычислительной задачи используются примерно одни и те же базовые математические понятия.

6. Объектно-ориентированное математическое программирование

Проиллюстрируем основные принципы разработанного подхода примерами решения некоторых задач математического программирования [5, 6, 7, 9]. Данная предметная область допускает достаточно глубокую проблемную классификацию и предоставляет практически содержательные случаи разработки и применения объектных вычислительных парадигм.

Математическое программирование в самом общем виде определяется как задача оптимизации с ограничениями в пространстве $\mathbf{R}^n$:

$\min f(x)$,

$g_i(x) \le 0, \quad i=1, \ldots, m, \quad x \in S \subset \mathbf{R}^n$.

Нетрудно заметить, что постановка задачи осуществляется с использованием следующих математических объектов, соответствующих понятиям пространства вещественных чисел $\mathbf{R}^n$, вектора $x \in \mathbf{R}^n$, компоненты которого являются неизвестными задачи, функционала $f(x)$, называемого в данном случае целевой функцией, а также векторной функции $g(x)$: $\mathbf{R}^n \to \mathbf{R}^m$ и множества $S \subset \mathbf{R}^n$, определяющих систему ограничений задачи. Следуя намеченному подходу, с каждым из выделенных понятий будем связывать соответствующий математический класс, а с обобщенной постановкой — необходимый проблемный класс.

В зависимости от типов математических объектов, а именно от свойств функций f, g_i и множества S, используют различную терминологию задач математического программирования, неформально отражающую возможности построения и применения соответствующих численных методов для решения частных классов задач. Данный аспект является основополагающим для проблемной классификации и корректной реализации целостных иерархий математических и алгоритмических классов.

Так, непрерывное программирование предполагает задание произвольных нелинейных непрерывных функций и связного компактного множества. Частный случай отсутствия ограничений соответствует задачам непрерывной безусловной оптимизации, а варианты с выпуклыми функциями и множествами определяют постановки и методы выпуклого программирования. К практически важным случаям следует отнести также задачи квадратичного и линейного программирования, и задачи одномерной оптимизации, алгоритмически участвующие в решении большинства многомерных постановок. Наконец, случаи задания дискретного множества и не обязательно непрерывных функций определяют постановки дискретного и целочисленного программирования.

В дальнейшем ограничимся случаем непрерывного программирования и рассмотрим два характерных примера объектного программирования задач одномерной оптимизации и многомерной оптимизации с ограничениями, задаваемыми выпуклой областью.

Итак, пусть класс Function реализует понятие скалярной функции одной переменной и определяет метод вычисления ее значения по заданному аргументу. Наличие свойства унимодальности на рассматриваемом отрезке позволяет поставить задачу одномерной минимизации и решить ее в самом общем виде, соответствующем итеративному циклу с этапами инициализации вычислений, включающей выбор начального приближения, реализации итераций и проверки достижения заданной точности и превышения отведенных задаче вычислительных ресурсов. Все перечисленные операции могут быть реализованы как методы некоторого алгоритмического класса OneVariateAlgorithm, предназначенного для решения задач заданного класса Problem.

Известные методы деления отрезка пополам, последовательности Фибоначчи и золотого сечения гарантированно решают задачу минимизации для унимодальных функций. Если исследуемая функция является кроме того и непрерывной, то множество применимых методов расширяется методами парабол и кубической интерполяции. Для дифференцируемых функций применимы также методы дихотомии и интерполяции с производными, методы секущих, касательных, а также алгоритмы Голдстейна, Армийо и Вольфе–Пауэлла, реализующие так называемый экономичный поиск. Наконец, условие дважды дифференцируемости функции определяет возможность использования метода Ньютона и близких ему вариантов.

Поскольку свойство дважды дифференцируемости необходимо влечет непрерывную дифференцируемость, а она в свою очередь — непрерывность функции, то введением иерархии классов, изображенной на рис. 4, достигается математически содержательная классификация функциональных объектов, для которых последовательно определяются и наследуются методы вычисления значений функций, поиска экстремума, вычисления значений первой и второй производной.

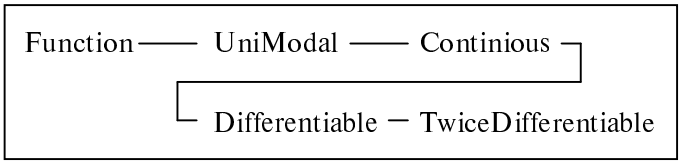

Рис. 4. Объектная классификация функций одной переменной

Аналогичную иерархию образуют методы одномерной оптимизации (см. рис. 5), которые строго сориентированы на выделенные функциональные свойства.

Таким образом, выстроены две параллельные иерархии математических и алгоритмических классов. Функциональные и алгоритмические классы с одноименными названиями не реализуют полностью необходимые вычислительные процедуры и могут быть определены только как абстрактные, при этом наследуемые ими классы функций и вычислительных методов организуются как конкретные. Тогда определение методов минимизации как методов абстрактных функциональных классов с использованием механизма ссылок на соответствующие абстрактные алгоритмические классы обеспечивает важную парадигму контроля корректности применения вычислительных методов для частных математических постановок, при которой каждый алгоритм может быть применен ко всем функциональным объектам, располагающимся в общей иерархии ниже него, и, наоборот, минимизация каждой функции может осуществляться любым из алгоритмов, занимающим в иерархии более высокое положение.

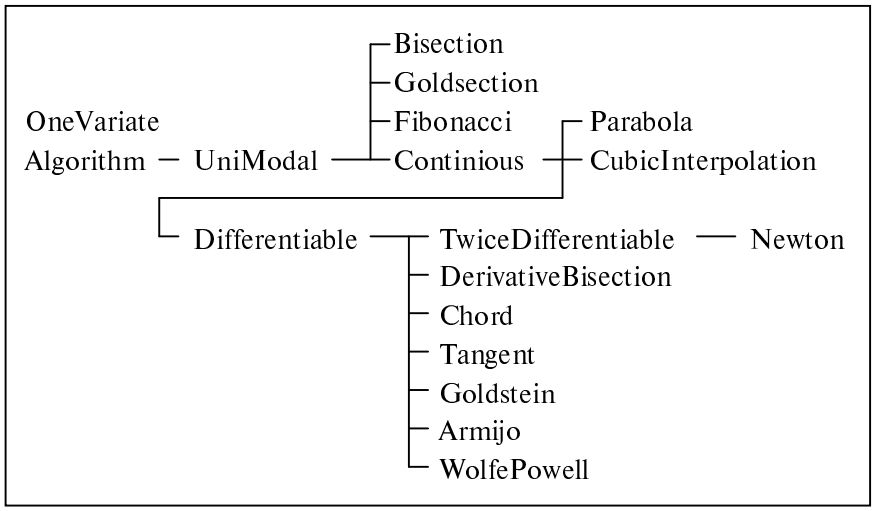

Рис. 5. Объектная классификация методов одномерной оптимизации

Другую интересную вычислительную парадигму иллюстрирует пример решения задачи оптимизации функционала с ограничениями в виде заданной выпуклой области.

Пусть класс Functional реализует понятие дважды дифференцируемой функции, для которой определены методы вычисления значения фнкционала, его градиента и гессиана, а класс Domain соответствует понятию выпуклой области и определяет методы выбора произвольной внутренней точки области, установления принадлежности точки области и вычисления проекции точки на область.

Прежде всего укажем конкретные варианты областей, для которых такие задачи обычно возникают. Это прежде всего шар, гиперплоскость, параллелепипед, полупространство, положительный октант, политоп, полиэдр (см. рис. 6). Удобно к объектам данного класса относить и само пространство $\mathbf{R}^n$, достигая в этом случае полной унификации постановок задач условной и безусловной оптимизации. Ясно, что методы класса пространства реализуют только тривиальные операции.

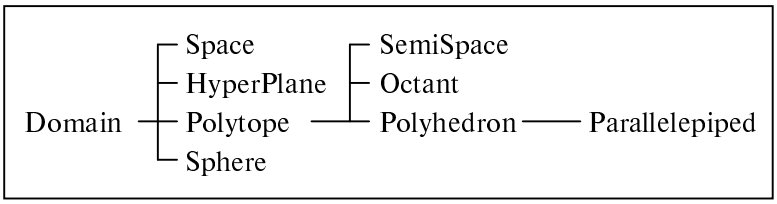

Рис. 6. Объектная классификация характерных областей оптимизации

Попытаемся сконструировать некоторый обобщенный класс Algorithm, который бы отражал то значительное многообразие алгоритмов, разработанное в теории оптимизации для решения данного класса задач. Будем следовать идеологии метода возможных

направлений и выделим основные алгоритмические этапы, с которыми и будем связывать соответствующие методы класса. Как в предыдущем случае, для класса Algorithm можно ввести методы инициализации вычислений и выбора начального приближения, реализации одной итерации, проверки сходимости и достижения заданной точности, а также анализа отведенных задаче ресурсов.

Однако теперь процедуру выполнения одной итерации можно представить более подробно с использованием стадий выбора направления спуска, формирования функции переменной шага, соответствующей минимизируемому функционалу в выбранном направлении при условии проектировании приближения на заданную область, а также ее минимизации одним из рассмотренных выше одномерных алгоритмов. С основными алгоритмическими стадиями будем связывать некоторые вспомогательные семейства классов. Так, будем считать, что вычисление направления спуска реализуется методами класса DescentAlgorithm (см. рис. 7), а определение длины шага — методами класса OneVariateAlgorithm.

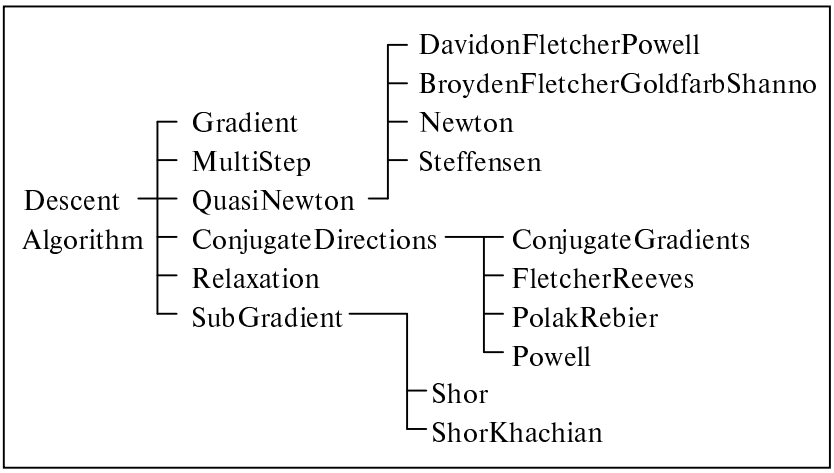

Рис. 7. Объектная классификация методов безусловной оптимизации

Приведенную иерархию методов спуска составляют градиентный метод, многошаговый овражный метод, квазиньютоновские методы, к наиболее важным из которых следует отнести методы Ньютона, Стеффенсена, Давидона–Флетчера–Пауэлла, Бройдена–Флетчера–Гольдфарба–Шанно, методы сопряженных направлений Флетчера–Ривса, Пауэлла, а также вариант Полака–Рибьера. Для недифференцируемых функций наиболее распространены методы покоординатного спуска и субградиентные подходы, в частности, методы Шора и Шора–Хачияна.

Тогда интересующий нас класс Algorithm может быть организован через систему ссылок на соответствующие объекты классов DescentAlgorithm и OneVariateAlgorithm и тем самым определять обобщенную алгоритмическую реализацию, при которой любой метод спуска может сочетаться с любым методом одномерной оптимизации. Данная вычислительная парадигма предоставляет важную для многих приложений возможность конструирования частных, но сложных алгоритмических версий из имеющихся в математической библиотеке базовых алгоритмов без какого-либо предварительного программирования.

При таком, вообще говоря, вольном обращении с численными методами трудно гарантировать их вычислительную корректность всякий раз. Однако вопросы применимости даже самых распространенных алгоритмических вариантов составляют, как правило, серьезную теоретическую задачу, разрешимую только в самых частных случаях. Поэтому описанный подход к организации обобщенных алгоритмических классов только расширяет вычислительные возможности, не суживая множества разрешимых численных постановок.

В заключение укажем и на другую важную особенность применения объектного подхода к рассмотренной оптимизационной задаче, заключающуюся в ее обобщенном представлении. В самом деле, объектная организация классов Functional и Domain позволила унифицировать все необходимые для численного решения базовые методы и определить класс Algorithm, применимый к широкому классу проблем, различающихся типами оптимизируемых функционалов и областей поиска.

Таким образом, применение объектной технологии к разработке математического обеспечения и, в частности, к решению задач математического программирования обнаруживает важные преимущества над традиционными процедурными методами и представляется достаточно перспективным. Предмет дальнейших исследований составят вопросы развития предложенного подхода для основных разделов вычислительной математики и проблемы создания законченных программных приложений.

Работа поддержана Российским Фондом фундаментальных исследований (грант 95-01-01239).

ЛИТЕРАТУРА

1. Harmon Paul, and Taylor David A. Objects in action: commercial applications of object-oriented technologies. Addison-Wesley, Reading, MA, 1993.
2. Proceedings of the Conference on Object-Oriented Programming: Systems, Languages, and Applications, ACM, 1994.
3. John Carroll, The role of computer Software in numerical analysis teaching, SIGNUM Newsletter, 2 (1992).
4. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. — М.: Наука, 1987.
5. Васильев Ф.П. Численные методы решения экстремальных задач. — М.: Наука, 1988.
6. Евтушенко Ю.Г. Методы решения экстремальных задач и их применение в системах оптимизации. — М.: Наука, 1982.
7. Карманов В.Г. Математическое программирование. — М.: Наука, 1986.
8. Марчук Г.И. Методы вычислительной математики. — М.: Наука, 1989.
9. Мину М. Математическое программирование. Теория и алгоритмы: Пер. с фр. — М.: Наука, 1990.
10. Самарский А.А., Гулин А.В. Численные методы. — М.: Наука, 1989.