ОБЪЕКТНАЯ КЛАССИФИКАЦИЯ ЗАДАЧ И МЕТОДОВ НЕЛИНЕЙНОЙ БЕЗУСЛОВНОЙ ОПТИМИЗАЦИИ

В.А. Семенов, Е.Ю. Ширяева

В работе рассматриваются общие аспекты применения объектно-ориентированной технологии к программированию одномерных и многомерных задач нелинейной безусловной оптимизации, составляющих важнейшие разделы математического программирования. В результате проводимого объектного анализа строятся детальные объектные классификации задач и методов нелинейной оптимизации, которые могут использоваться в качестве теоретической и практической основы для разнообразных программных реализаций. Приводятся некоторые результаты проектирования и разработки объектно-ориентированной библиотеки для решения стандартных оптимизационных задач на языке Си++ с использованием предложенных классификаций. Обсуждается инструментальный характер разработанной библиотеки, допускающей как расширение проблемно-методического репертуара, так и его адаптацию к конкретным прикладным задачам.

1. Введение

В настоящее время существует ряд пакетов программ, предназначенных для решения задач нелинейной безусловной оптимизации. К числу таких программ следует отнести, прежде всего, известные пакеты MINPACK [1], TNPACK [2], а также программы разделов одномерной минимизации, безусловной и условной оптимизации в составе универсальных математических библиотек NAG [3], IMSL [4]. Большинство пакетов представляет собой набор стандартизованных автономных программных модулей, написанных на процедурно-ориентированных языках, главным образом на Фортране и Си. Как правило, программы используют несколько наиболее надежных и зарекомендовавших себя методов решения стандартных задач. Так, например, библиотека NAG реализует два алгоритма одномерной минимизации и три алгоритма многомерной безусловной оптимизации, соответствующих классическим вариантам квазиньютоновского метода и метода сопряженных градиентов. Близкие алгоритмические наборы содержат и другие упомянутые пакеты и библиотеки.

Процедурные модули пакетов могут быть непосредственно включены в любую авторскую программу. При этом предполагается, что прикладная

задача представима в стандартной форме, методики пакета обеспечивают удовлетворительную вычислительную эффективность на данном классе задач, а разработчик имеет квалификацию математика-программиста, необходимую для корректного использования модулей при программной реализации средств постановки и решения задачи. Однако на практике подобные допущения часто не оправданы, и разработчик сталкивается с серьезными трудностями, связанными с необходимостью адаптации модулей к конкретным прикладным задачам, которые могут отличаться от стандартных и постановкой, и алгоритмами решения. Процедурные модули крайне плохо приспособлены для необходимых модификаций и разработчику нередко приходится перепрограммировать их заново.

Важное значение в связи с этим приобретает применение объектно-ориентированного подхода (ООП) [5], обеспечивающего широкие инструментальные возможности для развития и сопровождения уже созданного программного обеспечения. Вместе с тем, преимущества использования ООП могут быть достигнуты только в результате глубокого объектного анализа рассматриваемой предметной области, предполагающего построение необходимых объектных классификаций, и тщательного объектного проектирования, обеспечивающего практические возможности для эффективной программной реализации.

Объектные классификации по существу являются формой систематизации выделенных групп объектов, для которых устанавливаются необходимые отношения общности. Обычно объектные классификации представляются деревьями или иерархиями классов. Выбранные группы объектов должны быть достаточно значимыми, чтобы выражать наиболее содержательные понятия предметной области, и достаточно конструктивными, чтобы быть программно реализуемыми. Организация классовых иерархий тесно связана с возможностями неформального применения известных объектных парадигм и во многом определяет качество объектно-ориентированного математического обеспечения.

В настоящей работе представлены некоторые результаты проектирования и реализации объектно-ориентированной библиотеки на языке Си++, предназначенной для постановки и решения одномерных и многомерных задач нелинейной безусловной оптимизации. Заметим, что теоретическую и практическую основу библиотеки составляют разработанные детальные объектные классификации задач и методов оптимизации, которые и являются ее систематическим представлением. Особое внимание в работе уделяется проблеме создания единой библиотеки классов, допускающей унифицированную разработку программ для постановки различных классов оптимизационных задач и их

решения с использованием широкого многообразия оптимизационных алгоритмов.

2. Математические постановки задач оптимизации и их объектный анализ

Для проведения необходимого объектного анализа рассмотрим, прежде всего, математические постановки задач оптимизации и попытаемся выделить наиболее содержательные группы объектов, участвующие в них. Обычно задачи нелинейной безусловной оптимизации формулируются несколькими способами [6–11]. Наиболее распространенная постановка задачи минимизации функции $f\colon R^n \to R$ состоит в поиске точки $x^* \in R^n$ такой, что $f(x^*) \le f(x)$ для всех $x \in R^n$.

Данная постановка является частным вариантом более общей задачи условной минимизации с ограничениями типа замкнутой связной области

$$\min_{x \in D \subset R^n} f\colon R^n \to R.$$

Принято считать, что если решение лежит внутри области *D*, то данную задачу следует рассматривать как задачу безусловной минимизации. В противном случае решение является граничной точкой области и минимизация функции на области становится задачей условной оптимизации.

Практически важный класс задач безусловной оптимизации составляют нелинейные задачи о наименьших квадратах. Последние формулируются как задачи минимизации функционала специального вида $\frac{1}{2}F(x)^T F(x)$, для которого задана функция невязки $F\colon R^n \to R^m$, причем $m > n$.

Заметим, что задачи одномерной оптимизации являются частными случаями приведенных выше обобщенных формулировок при $n = 1$ и их можно было бы не рассматривать отдельно. Однако существование специальных алгоритмов одномерной минимизации, ориентированных на данный размерный фактор и не допускающих редукции соответствующих многомерных алгоритмических вариантов, требует отдельного анализа таких задач, который проводится в следующих разделах.

Приведенные постановки задач оптимизации предполагают, прежде всего, наличие в проектируемой объектной системе классов векторов или точек пространства, областей пространства, а также классов скалярных и векторных функций. С учетом специфики постановки задачи о наименьших квадратах, а также в силу отмеченных алгоритмических отличий при решении одномерных и многомерных задач будем различать скалярные и векторные функции одной и нескольких переменных и

использовать достаточно естественную для указанного функционального

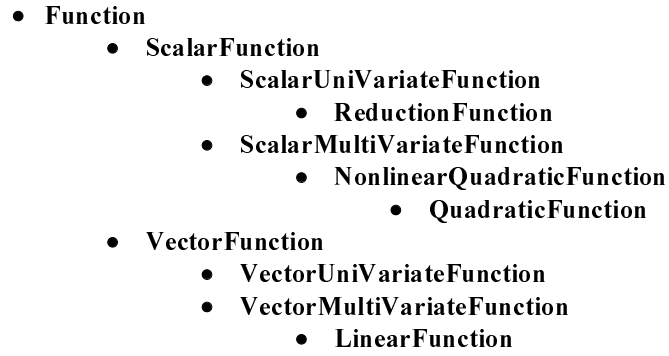

Рис. 1. Объектная классификация математических функций

многообразия классификацию, изображенную на рис. 1.

Данная функциональная классификация в определенной степени отражает имеющееся разнообразие постановок оптимизационных задач, которые связаны с определенными типами или видами функций [12]. Вершиной иерархии является функциональный суперкласс **Function**, формально объединяющий все функциональные объекты и специфицирующий методы определения размерностей результата и аргумента. Абстрактные классы **ScalarFunction** и **VectorFunction**, наследуемые от класса **Function**, подразделяют функции на скалярные и векторные в соответствии с размерностью возвращаемого ими результата.

Более содержательными в математическом отношении являются абстрактные классы **ScalarUniVariateFunction**, **ScalarMultiVariateFunction** и **VectorMultiVariateFunction**. Для данных классов целесообразно определить набор базовых методов, чисто виртуально реализующих основные операции численного анализа и, прежде всего, операции вычисления значения функции, а также ее производных. Тогда большинство оптимизационных алгоритмов, редуцируемых к базовым операциям, могут быть реализованы как методы данных абстрактных классов. Причем конкретизация функциональных объектов, в конечном итоге выражающаяся в реализации всех базовых методов, делает возможным применение всего методического репертуара данных абстрактных классов ко всем наследуемым частным объектам.

В дальнейшем будем считать, что класс скалярных функций одной переменной **ScalarUniVariateFunction** определяет методы, которые реализуют следующие базовые операции и процедуры над функциональными объектами $f: R \to R$:

— вычисление значения функции в заданной точке $f(x)$,

— вычисление первой и второй производной $f'(x)$, $f''(x)$ в заданной точке,

— локализация и нахождение нулей функции на заданном интервале $f(x)=0,\ x\in[a,b]$,

— локализация и нахождение экстремумов функции на заданном интервале $\min\limits_{x\in[a,b]} f\colon R\to R$.

С классом скалярных функций многих переменных **ScalarMultiVariateFunction** будем связывать методы, реализующие процедуры:

— вычисление значения функции в заданной точке $f(x)$,

— вычисление градиента и гессиана функции в заданной точке $\nabla f(x)$, $\nabla^2 f(x)$,

— оптимизация функционала на заданной области $\min\limits_{x\in D\subset R^n} f\colon R^n\to R$.

В функциональной иерархии определен также конкретный класс **ReductionFunction**, наследуемый от класса **ScalarUniVariateFunction**. Данный класс предназначен для преобразования многомерного функционала $f(x)\colon R^n\to R$ в функцию одной переменной $g(\alpha)=f(x+\alpha d)$, в которой переменная α определяет длину шага в заданном направлении $d\in R^n$ от заданной точки $x\in R^n$. Использование данного класса существенно упрощает реализацию многих многомерных алгоритмов, обсуждаемых в дальнейших разделах. Класс **ReductionFunction** определяет необходимые методы для установки фиксированного направления d и определения начальной точки x, а также непосредственно реализует все базовые методы родительского класса.

Наконец, для класса векторных функций многих переменных **VectorMultiVariateFunction** определим методы, которые реализуют операции и процедуры над функциональными объектами $F\colon R^n\to R^m$, а именно:

— вычисление значения отображения $F(x)$,

— вычисление якобиана и второй производной отображения $F'(x)$, $F''(x)$,

— решение системы уравнений $F(x)=0$, определяемой данным отображением.

Приведенный набор базовых методов вычисления функций и их производных позволяет реализовать большинство известных вычислительных оптимизационных алгоритмов в виде методов рассматриваемых классов функциональных отображений. Вместе с тем, конкретизация функциональных классов и связанная с ней возможная специализация численных подходов делают целесообразными рассмотрение и введение в функциональную иерархию частных классов **NonlinearQuadraticFunction** и **QuadraticFunction**. Данные классы соответствуют специальным видам функций, участвующих в постановке

задач о наименьших квадратах. Кроме того, в иерархию вводится класс **LinearFunction**, представляющий подмножество линейных векторных функций.

Так класс **NonlinearQuadraticFunction** определяет множество нелинейных квадратичных функций вида $\frac{1}{2}F(x)^T F(x)$, $F\colon R^n \to R^m$, класс **QuadraticFunction** — множество квадратичных функций вида $\frac{1}{2}x^T Ax + b^T x + c$, где $A \in L(R^n)$, и, наконец, класс **LinearFunction** — множество векторных функций $F(x) \equiv Ax + b$, соответствующих линейным операторам $A \in L(R^n, R^m)$. При этом класс квадратичных функций мы рассматриваем как частный производный класс нелинейных квадратичных функций, для которого функция невязки представляется линейным оператором. Хотя в этом случае возникает необходимость поддержки соответствия нелинейных и линейных функциональных компонент на программном уровне, такой способ классовой организации выглядит более привлекательным с точки зрения логической целостности всей функциональной системы.

Приведенная функциональная иерархия продолжается конкретными классами, которые образуют самостоятельные подиерархии родственных функциональных объектов и тем самым определяют возможное многообразие постановок задач, связанных с конкретными типами или видами оптимизируемых функционалов. В конкретных функциональных классах необходимо реализовать лишь методы вычисления самих отображений и их производных. Заметим, что методы вычисления производных допускают конкретную реализацию и на абстрактном уровне в виде процедур численного дифференцирования. Тем не менее, при наличии для производных несложных явных аналитических выражений целесообразно переопределить данные методы в конкретных классах с целью повышения точности и надежности применяемых вычислительных алгоритмов.

Все вычислительные процедуры, за исключением базовых операций, реализуются как методы абстрактных функциональных классов. Тем не менее, оформление в виде виртуальных функций позволяет переопределить их при необходимости в частных классах, возможно, более эффективным образом. Такую возможность мы используем при реализации алгоритмов решения линейных и нелинейных задач о наименьших квадратах. В этом случае для решения задач данного класса могут применяться как общие оптимизационные алгоритмы, реализуемые методами класса **ScalarMultiVariateFunction**, так и специальные алгоритмы, ориентированные на особенности квадратичного представления и реализуемые методами классов **NonlinearQuadraticFunction**, **QuadraticFunction**.

Для полноты объектной системы необходимо ввести также класс векторов или точек пространства **Vector**, класс областей пространства **Domain**, которые непосредственно участвуют в постановках задач оптимизации. Кроме того, для представления объектов, являющихся результатами вычисления гессиана или якобиана функциональных отображений необходим также матричный класс **Matrix**. Вопросы организации векторных и матричных классов, предназначенных для решения задач линейной алгебры, достаточно хорошо проработаны [13].

Здесь же перечислим предполагаемые методы класса **Domain**, которые реализуют следующие процедуры:

— определение принадлежности точки заданной области $x \in D$,

— выбор произвольной точки области $x \in R^n$,

— определение принадлежности области заданной области $D_1 \subset D_2$,

— теоретико-множественные операции объединения, пересечения и вычитания областей.

Таким образом, выделены классы объектов, участвующих в различных постановках нелинейных оптимизационных задач. Основную часть объектной системы составляют функциональные классы, представляемые единой многоуровневой классовой иерархией. Иерархия в определенной мере отражает сложившуюся классификацию задач оптимизации и может использоваться в качестве конструктивной объектной основы для разнообразных программных реализаций математического обеспечения обсуждаемой направленности.

3. Объектная интерпретация методов оптимизации

Как указывалось выше, при создании вычислительных приложений важно, чтобы используемое математическое обеспечение предоставляло инструментальные возможности для дальнейшего развития программных средств, предназначенных как для постановки, так и для решения оптимизационных задач. В связи с этим интерес представляет организация объектно-ориентированного математического обеспечения, которое допускало бы реализацию различных оптимизационных методов на единой инструментальной основе. Следуя подходу, предложенному в работе [14], будем рассматривать сами вычислительные методы и алгоритмы в качестве объектов и попытаемся построить единую объектную классификацию оптимизационных алгоритмов в виде целостной иерархии наследуемых алгоритмических классов.

В определенном смысле построение подобной иерархии может следовать изложению проблем оптимизации в известных монографиях [6–11]. Причем логическая организация классовой иерархии может отражать или повторять методологические построения, используемые самой

теорией оптимизации. Обеспечение логической целостности является достаточно важным для правильного восприятия пользователями основных алгоритмических и программных возможностей разрабатываемой библиотеки классов. Другим существенным аспектом подобного проектирования является обеспечение неформального применения объектной парадигмы, что, в конечном счете, упростило бы программную реализацию самой библиотеки и обеспечило бы необходимые инструментальные возможности для ее развития.

Алгоритмы нелинейной оптимизации по своей природе являются алгоритмами итерационного характера. Поэтому имеет смысл построить суперкласс итерационных алгоритмов **IterativeAlgorithm**, определяющий основные алгоритмические компоненты организации обобщенного итерационного вычислительного процесса и являющийся родительским для всех классов оптимизационных алгоритмов. Возможная реализация такого класса, специфицирующего основные этапы организации итерационного вычислительного процесса и предоставляющего необходимые средства оперирования с точностью вычислений и вычислительными ресурсами, описана в работе [15].

Определим также класс **OptimizationAlgorithm**, формально объединяющий все классы оптимизационных алгоритмов, и классы **OptimizationUniVariateAlgorithm**, **OptimizationMultiVariateAlgorithm**, подразделяющие все алгоритмы на одномерные и многомерные. Описанная организация алгоритмических классов в виде иерархии

- **IterativeAlgorithm**
  - **OptimizationAlgorithm**
    - **OptimizationUniVariateAlgorithm**
    - **OptimizationMultiVariateAlgorithm**
    - **NonlinearQuadraticOptimizationAlgorithm**

Рис. 2. Объектная классификация методов оптимизации

наследуемых классов изображена на рис. 2.

Принятое деление является принципиальным в силу замечаний сделанных выше, несмотря на то, что многомерные алгоритмы могут быть использованы и при решении одномерных задач, а часто являются простыми обобщениями одномерных вариантов. В общем же случае это неверно, поскольку не все многомерные алгоритмы автоматически редуцируются к одномерным.

Выделение самостоятельного класса **OptimizationAlgorithm** связано с необходимостью определения дополнительных данных и методов для оперирования с точностью вычислений. Для оптимизационных задач, наряду с оценкой точности самого решения x^*, важным является анализ точности выполнения функционального условия экстремума $\nabla f(x^*) = 0$.

Поэтому классом инкапсулируются дополнительные данные, соответствующие заданной и достигнутой точности для функционального условия, а также определяются необходимые методы оперирования с ними. Данную организацию алгоритмического класса следует признать достаточно общей, поскольку подобный анализ точности осуществляется как для одномерных, так и многомерных задач оптимизации.

Класс алгоритмов одномерной оптимизации **OptimizationUniVariateAlgorithm** непосредственно наследуется от класса оптимизационных алгоритмов. Дополнительными членами класса являются следующие:

— ссылка на оптимизируемую функцию $f(x)$ класса **ScalarUniVariateFunction**,

— ссылка на интервал поиска экстремума $[a, b]$,

— начальное приближение x^0, текущее и предыдущее приближенные решения x^{k+1}, x^k.

В данном классе непосредственно реализуются методы анализа сходимости итераций и методы оценки точности приближенного решения. Для оценки точности используются принятые нормированные критерии относительной точности для приближенного решения и функционального условия экстремума

$$\frac{\left|x^{k+1}-x^k\right|}{\max\left\{\text{typ}\,x, \left|x^{k+1}\right|\right\}} < \tau_1 \qquad \text{и} \qquad \left|\frac{f'(x^{k+1})}{f(x^{k+1})}x^{k+1}\right| < \tau_2 .$$

В классе определены также виртуальные методы, позволяющие применить требуемый алгоритм к оптимизируемой функции, а также методы, осуществляющие редукцию задачи оптимизации к поиску минимума или максимума. В качестве параметров методов указываются оптимизируемый функциональный объект, интервал поиска экстремума и возможное начальное приближение.

Аналогичным образом строится класс алгоритмов многомерной оптимизации **OptimizationMultiVariateAlgorithm**, наследуемый от класса оптимизационных алгоритмов.

Членами класса **OptimizationMultiVariateAlgorithm** являются:

— ссылка на оптимизируемый функционал $f(x)$ класса **ScalarMultiVariateFunction**,

— ссылка на область поиска экстремума D класса **Domain**,

— начальное приближение x^0, текущий и предыдущий вектор приближенного решения x^{k+1}, x^k класса **Vector**,

— ссылка на соответствующий одномерный алгоритм класса **OptimizationUniVariateAlgorithm**.

Последняя ссылка необходима для реализации большинства методов многомерной оптимизации, редуцируемых к последовательности одномерных задач и использующих одномерную минимизацию в качестве вспомогательной процедуры. Например, в глобально сходящихся методах линейного поиска одномерная минимизация используется для вычисления оптимальной длины шага вдоль выбранного направления, в релаксационных методах — для оптимизации функционала относительно текущей компоненты приближенного решения и т.п.

В классе реализуются методы анализа сходимости итераций и методы оценки точности приближенного решения на основе критериев, подобных одномерным условиям

$$\max_{1\le i\le n}\left|\frac{x_i^{k+1}-x_i^k}{\max\left\{\left|x_i^{k+1}\right|,\ \mathrm{typ}\ x_i\right\}}\right|\le\tau_1 \quad \text{и} \quad \max_{1\le i\le n}\left|\frac{\nabla f(x^{k+1})_i\ \max\left\{\left|x_i^{k+1}\right|,\ \mathrm{typ}\ x_i\right\}}{\max\left\{\left|f(x^{k+1})\right|,\ \mathrm{typ}\ f\right\}}\right|\le\tau_2 .$$

Заметим, что на практике часто используются и другие критерии остановки итераций. В любом случае, данный класс допускает возможность переопределения виртуальных методов в наследуемых алгоритмических классах.

Наконец, класс **OptimizationMultiVariateAlgorithm** специфицирует метод, реализующий саму процедуру оптимизации. В качестве параметров метода указываются оптимизируемый функционал, область поиска, а также возможное начальное приближение и вспомогательный одномерный алгоритм.

Аналогичным образом строится класс **NonlinearQuadraticOptimizationAlgorithm**, ориентированный на решение задач нелинейной квадратичной оптимизации. Главным отличием данного абстрактного метода является наличие ссылки не на произвольный оптимизируемый функционал класса **ScalarMultiVariateFunction**, а на нелинейный квадратичный функционал класса **NonlinearQuadraticFunction**. Данное замечание существенно, поскольку при реализации алгоритмов данного класса наряду с доступом к общим методам функционального класса необходим явный доступ к методам соответствующей векторной компоненты квадратичного функционала. Остальные данные и методы имеют аналогичное предназначение.

Таким образом, описаны абстрактные классы алгоритмов одномерной и многомерной оптимизации. Классы наследуют данные и методы родительских классов **IterativeAlgorithm**, **OptimizationAlgorithm**, определяющие и частично реализующие основные этапы обобщенного итерационного процесса оптимизационного алгоритма. При этом значительная часть методов, связанная с организацией всего вычислительного процесса, оценкой и контролем точности, анализом

сходимости итерационного алгоритма, контролем вычислительных ресурсов, непосредственно реализуется уже в абстрактных классах. Остальная часть методов определяется чисто виртуально и нуждается в конкретной реализации в наследуемых алгоритмических классах. К этой части методов относятся, прежде всего, метод вычисления самой итерации и методы сложностной оценки вычислительных ресурсов, связанные с конкретными оптимизационными алгоритмами.

4. Объектная классификация методов одномерной оптимизации

Обсудим вопросы построения объектной классификации одномерных оптимизационных алгоритмов в виде целостной иерархии наследуемых классов (рис. 3). Алгоритмы одномерной оптимизации по цели применения и по способу организации вычислений условно разделим на следующие три группы:

— алгоритмы уточнения приближенной точки экстремума,

— алгоритмы сужения интервала, на котором осуществляется поиск оптимума,

— алгоритмы экономичного поиска.

Первые две группы отражают возможные способы организации итераций. Последняя группа алгоритмов предназначена главным образом для использования в многомерных алгоритмах линейного поиска и имеет целью не нахождение экстремума с той или иной точностью, а лишь улучшение приближенного решения в некоторой мере, определяемой соответствующим правилом. По способу организации экономичные алгоритмы, как правило, комбинируют технику двух предыдущих групп.

Выделенные группы алгоритмов представлены в иерархии абстрактными классами **CorrectionAlgorithm**, **IntervalAlgorithm** и **EconomicalRule**. Введенные абстрактные классы наследуются в иерархии частными алгоритмическими реализациями, допускающими в свою очередь продолжение соответствующими модифицированными вариантами.

- **OptimizationAlgorithm**
  - **OptimizationUniVariateAlgorithm**
    - **CorrectionAlgorithm**
      - **ModifiedNewton**
        - **NewtonRaphson**
        - **NewtonBisection**
        - **Secant**
    - **IntervalAlgorithm**
      - **Parabola**
      - **HyperbolaWithDerivative**
      - **ProportionalSection**
        - **Bisection**
        - **BisectionWithDerivative**
        - **Fibonacci1**
        - **Fibonacci2**
        - **GoldenSection**
    - **EconomicalRule**
      - **Goldstein**
      - **Armijo**
      - **WolfePowell**
      - **Curry**
      - **Altman**

Рис. 3. Объектная классификация методов одномерной оптимизации

Рассмотрим более подробно конкретные группы одномерных алгоритмов.

4.1. Класс алгоритмов уточнения CorrectionAlgorithm

Абстрактный класс **CorrectionAlgorithm** объединяет алгоритмы уточнения и специфицирует их свойства. Данная группа алгоритмов реализует обычный итерационный процесс и использует принцип поиска решения путем последовательного уточнения предыдущих приближений. Для этого решается нелинейное уравнение $f'(x) = 0$ одним из итерационных методов, приводящих к следующей обобщенной формуле:

$$x^{k+1} = x^k - M_k \, f'(x^k),$$

в которой значение M_k определяется выбранным конкретным алгоритмом рассматриваемого класса. Поскольку выбор значения M_k неоднозначен, в классе **CorrectionAlgorithm** определен чисто виртуальный метод вычисления этого значения. Например, в методе Ньютона–Рафсона это значение выбирается обратным второй производной в предыдущей точке, а в методе секущих — обратным какой-либо ее аппроксимации.

Методы уточнения представлены в предложенной иерархии семейством квазиньютоновских методов **ModifiedNewton**, включающим метод Ньютона–Рафсона **NewtonRaphson**, гибридный вариант метода Ньютона и бисекции **NewtonBisection**, а также метод секущих **Secant**.

Абстрактный класс **ModifiedNewton** реализует квазиньютоновский метод для одномерной оптимизации. Будучи наследованным от абстрактного класса **CorrectionAlgorithm**, класс определяет метод вычисления значения M_k в обобщенном алгоритме по формуле

$$M_k = \frac{\lambda_k}{f''(x^k)},$$

где параметр λ_k в свою очередь определяется более частными алгоритмическими вариантами метода. Для этого в интерфейсе класса предусмотрен чисто виртуальный метод вычисления этого параметра.

Конкретный класс **NewtonRaphson**, наследуемый от класса квазиньютоновских методов **ModifiedNewton**, реализует классический вариант метода Ньютона–Рафсона (Ньютона–Канторовича), в котором выбирается значение параметра $\lambda_k = 1$.

Иную алгоритмическую реализацию предоставляет класс **NewtonBisection**, соответствующий гибридному варианту метода Ньютона и бисекции. Сочетание метода Ньютона, обеспечивающего высокую локальную скорость сходимости, с методом бисекции, обеспечивающим быстрый и надежный глобальный поиск хорошего приближения, является достаточно эффективным для практического применения. В этом алгоритме с помощью половинного деления вычисляется значение λ_k, которое улучшает приближенное решение и удовлетворяет условию спуска $f(x^{k+1}) < f(x^k)$. Алгоритм начинается с выполнения полного шага $\lambda_k = 1$, соответствующего методу Ньютона–Рафсона. В случае если $f''(x^k) < 0$, длина шага выбирается $\lambda_k = -1$.

Наконец, конкретный класс **Secant** реализует алгоритм секущих (хорд). Алгоритм использует для вычисления минимума тот же обобщенный итерационный процесс, однако, вместо значения второй производной берется ее аппроксимация по двум предыдущим значениям аргумента

$$M_k = \frac{x^k - x^{k-1}}{f'(x^k) - f'(x^{k-1})}.$$

При реализации класса учитывается необходимость хранения двух текущих значений приближенного решения, а также задание или определение двух начальных точек для инициализации цикла вычислений.

4.2. Класс алгоритмов сужения интервала IntervalAlgorithm

Абстрактный класс **IntervalAlgorithm** определяет множество алгоритмов, осуществляющих решение путем сужения интервала поиска.

Алгоритмы этой группы используют ту или иную стратегию разбиения интервала и определения тех его частей, на которых заведомо не может быть экстремума. Тем самым достигается сужение области поиска решения и обеспечивается сходимость к нему.

Методы класса **IntervalAlgorithm** определяют и частично реализуют процедуры:

— разбиения интервала поиска минимума,

— анализа интервалов и локализации минимума,

— уточнения приближенного решения.

Для проверки наличия минимума на текущем интервале, заданном тройкой точек, в классе предусмотрен вспомогательный метод, реализующий такую проверку. Для заданных трех точек $x_1 \le x_2 \le x_3$ метод проверяет выполнение условий: $f(x_1) \ge f(x_2)$ и $f(x_3) \ge f(x_2)$.

Кроме того, в классе переопределен метод оценки точности, реализуемый родительским классом **OptimizationUniVariateAlgorithm**. Вместо оценки, основанной на анализе сходимости итераций, вводится оценка точности, определяемая границами интервала поиска.

Методы данной группы представлены в иерархии классами метода квадратичной интерполяции (параболы) **Parabola**, метода кубической интерполяции с использованием производной **HyperbolaWithDerivative**, двумя вариантами метода бисекции (половинного деления) **Bisection**, **BisectionWithDerivative** и его вариациями, реализующими деление интервала оптимизации на неравные части, — методами Фибоначчи **Fibonacci1**, **Fibonacci2** и методом золотого сечения **GoldenSection** [6, 8, 9].

Класс **Parabola** реализует классический метод параболы. Для решения задачи алгоритму необходимо задать три точки, удовлетворяющие условию локализации минимума. На каждой итерации алгоритма отыскивается значение минимума параболы построенной по трем точкам, принадлежащим минимизируемой функции. Из четырех имеющихся значений выбирают три точки, удовлетворяющие указанному условию. При программной реализации данного алгоритмического класса достаточно естественным оказывается применение класса квадратичных степенных полиномов **QuadraticPolynomial**, входящего в состав математической объектно-ориентированной библиотеки [15] и инкапсулирующего необходимые методы интерполирования и вычисления минимума.

Более сложную алгоритмическую реализацию предоставляет класс **HyperbolaWithDerivative**. На каждой итерации алгоритм также использует три точки, удовлетворяющие условию локализации минимума, однако применяет кубическую интерполяцию для определения приближенного минимума функции. С этой целью используется значение

производной в одной из точек, в которых функция имеет отрицательную производную. Как и в предыдущем случае, программная реализация алгоритмического класса опирается на класс кубических степенных полиномов **CubicPolynomial**, для которого определены методы интерполирования и вычисления минимума.

Важную группу алгоритмов сужения интервала составляют алгоритмы пропорционального деления класса **ProportionalSection**. Общим для этих алгоритмов является то, что интервал дробится в определенной пропорции, задаваемой параметром метода γ. С помощью данного параметра на каждой итерации рассчитываются интервал деления $\Delta = (b-a)/\gamma$ и значения промежуточных точек на интервале поиска: $c = b - \Delta$ и $d = a + \Delta$. На каждой итерации из полученных четырех точек a, c, d и b выбираются три, удовлетворяющие условию локализации минимума. Затем на оставшемся интервале заново определяются промежуточные точки.

Абстрактный класс **ProportionalSection** реализует обобщенный итерационный процесс, характерный для всех алгоритмов пропорционального деления. Поскольку процедура нахождения вспомогательных точек у всех алгоритмов различна, в классе **ProportionalSection** определен соответствующий чисто виртуальный метод, реализуемый конкретными наследуемыми алгоритмическими классами. Данный класс алгоритмов представлен в алгоритмической иерархии конкретными классами **Bisection**, **BisectionWithDerivative**, **Fibonacci1**, **Fibonacci2** и **GoldenSection**.

Класс **Bisection** реализует классический алгоритм половинного деления. На первом шаге алгоритма, реализуемом в методе инициализации, находится средняя точка заданного интервала $c = (a+b)/2$. В методе вычисления итерации определяются еще две промежуточные точки интервала поиска: $d = (a+c)/2$ и $e = (c+b)/2$. Из имеющихся точек выбираются три смежные точки, удовлетворяющие условию локализации минимума.

Класс **BisectionWithDerivative** реализует алгоритм половинного деления с использованием производных. Как и в методе бисекции при вычислении итерации находится средняя точка. Однако уточнение интервала поиска происходит не на основе разностной оценки локализации минимума, а с использованием аналитической производной $f'(c)$. Если значение производной в средней точке положительно, то в качестве уточненного интервала поиска берется левая половина, в противном случае — правая.

Классы **Fibonacci1**, **Fibonacci2** реализуют два варианта алгоритма последовательности Фибоначчи, различающиеся условиями поиска экстремума.

Класс **Fibonacci1** осуществляет нахождение оптимума функции на заданном интервале $[a, b]$ с заданной абсолютной точностью ε. На этапе инициализации рекуррентным образом находится первое значение F_n последовательности Фибоначчи $\{F_k: F_1 = 1, F_2 = 2, F_k = F_{k-1} + F_{k-2}\}$, удовлетворяющее условию $F_n \geq (b-a)/\varepsilon$. Затем определяется параметр метода $\gamma = F_n / F_{n-1}$ и отыскиваются начальные значения промежуточных точек. Из имеющихся точек отбираются необходимые три, удовлетворяющие условию локализации минимума. Новая промежуточная точка берется симметрично к имеющейся относительно центра нового интервала.

Класс **Fibonacci2** определяет процедуру нахождения оптимума функции на заданном интервале $[a, b]$ по заданному предельному числу возможных итераций N. Для этого находится значение F_{N-1} последовательности Фибоначчи. Вычисляется $\Delta = (b-a) / F_{N-1}$. Дальше алгоритм выполняется, как и в первом случае. Полученная точность будет равна соответственно Δ.

Класс **GoldenSection** осуществляет поиск экстремума методом золотого сечения. Метод является упрощенным вариантом предыдущего алгоритма и использует вместо значений последовательности Фибоначчи, зависящих от требуемой точности вычисления или требуемого количества произведенных итераций, постоянное число $\gamma = (\sqrt{5}+1)/2 \approx 1{,}618$ — золотое сечение. Все вычисления идентичны первому варианту метода последовательности Фибоначчи.

4.3. Класс алгоритмов экономичного поиска EconomicalRule

Третий класс алгоритмов **EconomicalRule**, непосредственно наследуемый от класса **OptimizationUniVariateAlgorithm**, составляют алгоритмы так называемого «экономичного», или «наилучшего» поиска. Назначением этих алгоритмов является не приближенное нахождение точки оптимума, а лишь улучшение начального приближения в некоторой заданной мере. Данные алгоритмы имеют важное практическое значение, поскольку позволяют существенно ускорить работу методов многомерной оптимизации, использующих быструю одномерную оптимизацию.

Класс методов экономичного поиска **EconomicalRule** составляют конкретные классы **Goldstein**, **Armijo**, **WolfePowell**, **Curry** и **Altman**,

реализующие правила Голдстейна, Армийо, Вольфе–Пауэлла, а также принципы Карри и Альтмана соответственно [6, 7].

Вообще говоря, данный класс методов можно было бы организовать самостоятельно, поскольку он не обеспечивает поиска экстремума в обычном понимании. Тем не менее, включение этого класса в общую иерархию позволяет унифицировать методы решения одномерных задач и тем самым использовать при многомерной оптимизации различные алгоритмические возможности. В частности, обычный поиск минимума функционала в заданном направлении может определять важную группу классических методов наискорейшего спуска.

В основе рассматриваемых методов лежит общий алгоритм нахождения точки, близкой к оптимальной. При этом неявно предполагается, что поиск ведется на положительной полуоси $[x_0, +\infty)$ для убывающей функции $f'(x_0) < 0$. Заметим, что именно этот случай представляет практический интерес, поскольку минимизация многомерного функционала всегда осуществляется в направлении спуска. С точки зрения эффективности алгоритм должен делать достаточно большой шаг, чтобы обеспечивать быструю сходимость. Вместе с тем, длина шага должна быть ограничена с тем, чтобы обеспечивать реальное уменьшение функционала в направлении спуска. Оба условия определяют общие правила для рассматриваемой группы алгоритмов и реализуются абстрактным классом **EconomicalRule**.

Методы класса **EconomicalRule** определяют и частично реализуют следующие процедуры:

— установку параметров алгоритма m_1 и m_2, доступ к ним, а также проверку их корректности,

— проверку условия сходимости для предполагаемых итераций многомерного алгоритма,

— проверку условия для длины шага многомерного алгоритма,

— переопределения интервала поиска и приближенного значения минимума.

Хотя алгоритмы предназначены для решения многомерных задач, сформулируем правила данного класса в терминах постановки одномерной задачи.

На этапе инициализации вычислений задается интервал поиска $[x_1, x_2]$ и некоторое приближенное значение минимума $x \in [x_1, x_2]$. Обычно в качестве начальных значений выбираются $x_1 = x_0$ и $x_2 = +\infty$, а значение x экстраполируется на основе предыдущих успешных шагов многомерного алгоритма.

Далее, при вычислении итераций определяется значение $f(x)$ и осуществляется проверка первого правила

$$f(x) \le f(x_0) + m_1 (x - x_0)\, f'(x_0),$$

в котором параметр алгоритма $m_1 \in (0, 1)$. В случае невыполнения условия переопределяется правая граница интервала $x_2 = x$ и вычисляется новое значение x. Поскольку данное правило является общим для всех алгоритмов рассматриваемого класса **EconomicalRule**, в данном классе непосредственно реализуется метод проверки условия, а также метод переопределения границ интервала и определения нового значения минимума. В случае выполнения первого общего правила переходят ко второй проверке, которая различается для каждого конкретного алгоритма и реализуется соответствующим виртуальным методом класса **EconomicalRule**.

В классе **Goldstein** данный метод осуществляет проверку правила Голдстейна

$$f(x) \ge f(x_0) + m_2 (x - x_0)\, f'(x_0),$$

для которого параметр $m_2 \in (m_1, 1)$.

Класс **Armijo** реализует правило Армийо

$$f(x_0 + m_2 (x - x_0)) \ge f(x_0) + m_1 m_2 (x - x_0) f'(x_0),$$

где параметр метода $m_2 > 1$.

Наконец, класс **WolfePowell** осуществляет проверку условий правила Вольфе–Пауэлла

$$f'(x) \ge m_2 f'(x_0),$$

где параметр $m_2 \in (m_1, 1)$.

В случае невыполнения соответствующего условия конкретного алгоритма, переопределяется левая граница интервала поиска $x_1 = x$ и ищется новое значение $x \in [x_1, x_2]$. В противном случае задача считается решенной и x — искомая точка приближенного минимума.

Аналогичным образом организуются классы **Curry**, **Altman**, реализующие принципы Карри и Альтмана. Данные принципы вряд ли следует рассматривать в качестве конструктивной практической процедуры, поскольку они в большей степени представляют полезный теоретический инструмент для исследования других алгоритмов. Для целостного представления всех известных экономичных стратегий данные классы были введены в разрабатываемую алгоритмическую классификацию.

Таким образом, рассмотрена объектная классификация основных методов одномерной оптимизации. Рассмотрим возможности объектно-ориентированного подхода при программировании многомерных алгоритмов на основе их объектной классификации.

5. Объектная классификация методов многомерной безусловной оптимизации

Упомянутый выше суперкласс алгоритмов оптимизации **OptimizationAlgorithm** может служить базовым родительским классом для разнообразных реализаций многомерных оптимизационных алгоритмов, поскольку его методы определяют достаточно общие алгоритмические компоненты и оформлены в виде виртуальных процедур, допускающих переопределение в конкретных наследуемых алгоритмических классах.

Предлагается использовать следующую иерархию алгоритмических классов, основанную на данном суперклассе (рис. 4). Классовая иерархия в значительной степени отражает общую классификацию методов, принятую в теории оптимизации [6–11].

- **OptimizationAlgorithm**
  - **OptimizationMultiVariateAlgorithm**
    - **DescentAlgorithm**
      - **MultiStepGradient**
        - **Gradient**
          - **DivergentSeries**
          - **GradientNewton**
      - **QuasiNewton**
        - **TruncatedNewton**
          - **ModifiedNewton**
            - **Newton**
            - **NewtonConjugateGradient**
          - **DiscreteNewton**
            - **DiscreteNewton1**
            - **DiscreteNewton2**
        - **SecantUpdate**
          - **PowellBroyden**
          - **GenericBroyden**
            - **UniRankUpdate**
            - **DavidonFletcherPowell**
            - **BroydenFletcherGoldfarbShanno**
    - **ConjugateDirections**
      - **Daniel**
      - **FletcherReeves**
      - **PolakRibiere**
      - **ParallelTangents**
      - **Powell**
      - **Rosenbrock**
    - **Relaxation**
      - **SuccesiveOverRelaxation**
        - **GaussSeidel**
        - **SuccesiveOverRelaxationNewton**
      - **FastRelaxation**
  - **NonlinearQuadraticOptimizationAlgorithm**
    - **ModifiedGaussNewton**
      - **DampedGaussNewton**
        - **GaussNewton**
      - **LevenbergMarquardt**
    - **QuadraticOptimizationNewton**

Рис. 4. Объектная классификация методов многомерной оптимизации

Данная классификация представлена двумя основными абстрактными классами методов **OptimizationMultiVariateAlgorithm** и **NonlinearQuadraticOptimizationAlgorithm**, ориентированными на решение задач оптимизации в общей постановке и в постановке задач нелинейной квадратичной оптимизации. Общие алгоритмы подразделяются на методы спуска **DescentAlgorithm**, методы сопряженных направлений **ConjugateDirections**, а также релаксационные алгоритмы **Relaxation**. Алгоритмы нелинейной квадратичной оптимизации представлены классом методов типа Гаусса–Ньютона **QuasiGaussNewton**.

Степень детализации предлагаемой объектной классификации различна для разных семейств методов. Поэтому в иерархии представлены, прежде всего, классические варианты методов, составляющие предмет теоретических исследований, и методы, хорошо зарекомендовавшие себя в многочисленных практических приложениях.

Основную группу методов составляет класс методов спуска **DescentAlgorithm**. Класс виртуально определяет и реализует основную итерационную формулу методов спуска

$$x^{k+1} = x^k - \lambda_k p^k, \quad \lambda_k > 0,$$

в которой λ_k — длина шага в направлении p^k. Все алгоритмы данного класса различаются способами построения направления спуска и выбора оптимальной длины шага. Поэтому в классе определены виртуальные методы для вычислительных процедур, осуществляющих:

— построение направления спуска p^k,

— выбор оптимальной длины шага λ_k.

Причем метод построения направления спуска определяется как чисто виртуальный и нуждается в уточнении в конкретных алгоритмических реализациях. Метод выбора длины шага непосредственно реализуется в данном классе с использованием дополнительного одномерного алгоритма класса **OptimizationUniVariateAlgorithm**. Заметим, что в случае задания алгоритмического объекта для обычной одномерной оптимизации реализуется схема метода наискорейшего спуска. А в случае задания алгоритмического объекта частного класса **EconomicalRule** реализуется схема экономичного спуска. Тем самым, данная объектная реализация обеспечивает применение различных оптимизационных подходов.

Рассмотрим реализацию наиболее важных групп методов спуска, представленных в иерархии классом многошаговых градиентных методов **MultiStepGradient**, классом квазиньютоновских методов **QuasiNewton** и более частным классом методов секущих **SecantUpdate**, который следует отнести также к классу квазиньютоновских методов.

5.1. Класс многошаговых градиентных методов MultiStepGradient

Класс **Gradient** реализует семейство классических градиентных методов. Общая итерационная формула, используемая градиентными методами, имеет вид

$$x^{k+1} = x^k - \lambda_k p^k, \quad p^k = \frac{\nabla f(x^k)}{\left\| \nabla f(x^k) \right\|}.$$

Поскольку родительский класс определяет метод выбора шага λ_k как виртуальный, допустимо переопределение этого метода в частных наследуемых классах градиентных методов. В частности, известны варианты метода, в которых выбор перемещений определяется априорно. Примером таких методов, представленным в алгоритмической иерархии классом **DivergentSeries**, является метод расходящегося ряда, в котором шаг вычисляется как $\lambda_k = 1/k$.

Как известно, градиентные методы обеспечивают линейную скорость сходимости, которая, ввиду низкой вычислительной стоимости итераций, может оказаться вполне удовлетворительной для практического использования. Вместе с тем, градиентные методы обладают очень низкой сходимостью для функционалов, представляющих овражные поверхности. Это связано с необходимостью точной оптимизации функционала в градиентных направлениях, перпендикулярных к направлению экстремума. Одним из способов ускорения основных итераций является введение многошагового процесса, при котором оптимальная длина шага на общей итерации определяется после нескольких пробных шагов. Данное алгоритмическое обобщение градиентного метода реализуется классом **MultiStepGradient**, являющимся родительским для всех классов градиентных методов. В классе определен дополнительный параметр числа пробных шагов, а также необходимые методы доступа к нему. Метод вычисления основной итерации реализуется циклом пробных градиентных шагов.

5.2. Класс квазиньютоновских методов QuasiNewton

Класс **QuasiNewton** определяет наиболее значимую и развитую группу методов, реализующую обобщенный квазиньютоновский подход в соответствии с общей формулой методов этого семейства

$$x^{k+1} = x^k - \lambda_k H_k^{-1} \nabla f(x^k),$$

в которой матрица H_k является тем или иным приближением гессиана $\nabla^2 f(x)$. Выбор аппроксимации гессиана H_k определяет многообразие частных квазиньютоновских алгоритмов и реализуется соответствующим виртуальным методом класса **QuasiNewton**. Поскольку при практическом использовании неявной итерационной формулы предпочтительнее решать соответствующую линейную систему, а не обращать аппроксимирующую матрицу, в классе **QuasiNewton** предусмотрены альтернативные методы, реализующие вычисление направления спуска и итерации целиком на основе заданной матрицы.

Одним из условий применения метода Ньютона и его сильным ограничением является требование положительной определенности

гессиана. Поэтому на практике применяются различные способы построения аппроксимирующей матрицы на основе возмущения гессиана. Одним из наиболее распространенных методов данного типа является так называемый модифицированный метод Ньютона, в котором матрица выбирается как $H_k = \mu_k I + \nabla^2 f(x^k)$.

Класс **ModifiedNewton** реализует данный метод, определяя необходимые виртуальные процедуры для выбора параметров λ_k, μ_k. Выбор параметра μ_k осуществляется соответствующим методом класса **ModifiedNewton**, исходя из условия положительной определенности аппроксимации гессиана при минимальном параметре $\mu_k > 0$.

Метод выбора шага λ_k непосредственно реализуется в родительском классе **DescentAlgorithm**. Поскольку данный метод использует соответствующий алгоритмический объект для одномерной оптимизации, подобная программная организация обеспечивает возможность конструирования комбинированных алгоритмов, сочетающих в данном случае метод Ньютона с любым из методов наискорейшего или экономичного спуска, которые обсуждаются в предыдущих разделах. При этом не исключается возможность переопределения этого метода и построения частных алгоритмических вариантов. Так, например, классический вариант метода Ньютона, представленный в иерархии классом **Newton**, реализуется в результате конструирования частного производного класса от **ModifiedNewton**, в котором методы выбора параметров переопределены тривиальным образом $\lambda_k = 1$, $\mu_k = 0$.

Рассмотренные выше методы основаны на аналитическом вычислении градиента и гессиана функционала. Часто на практике такие операции оказываются чрезмерно дорогими, а иногда и просто невозможными. Поэтому в подиерархию квазиньютоновских методов внесены классы дискретных методов Ньютона **DiscreteNewton**, основанные на простейших конечно-разностных аппроксимациях гессиана. Класс **DiscreteNewton1** реализует метод аппроксимации гессиана с использованием аналитически заданного градиента

$$\left(H_k\right)_{ij} = \tfrac{1}{2}\left(\frac{\nabla f(x^k + h_j e_j)_i - \nabla f(x^k)_i}{h_j} + \frac{\nabla f(x^k + h_i e_i)_j - \nabla f(x^k)_j}{h_i}\right).$$

Класс **DiscreteNewton2** предоставляет другой способ аппроксимации гессиана, основанный на непосредственном использовании самого функционала

$$\left(H_k\right)_{ij} = \frac{\left[f(x^k + h_i e_i + h_j e_j) - f(x^k + h_i e_i)\right] - \left[f(x^k + h_j e_j) - f(x^k)\right]}{h_i h_j}.$$

В классе **DiscreteNewton** определены необходимые методы для конструирования вектора длин шага h. Практический выбор шага обычно основывается на известных оценках, использующих константы машинной арифметики.

Наконец, вспомним, что независимо от конкретного способа аппроксимации гессиана для вычисления ньютоновского направления необходимо на каждой итерации (или с той или иной периодичностью) решать соответствующую систему линейных уравнений. Вообще говоря, решение линейных систем, особенно систем высокой размерности, составляет самостоятельную вычислительную проблему, для решения которой могут применяться достаточно разные численные подходы, например, прямые или итерационные методы. Причем приближенное решение линейных систем в методах ньютоновского типа носит принципиальный характер и определяет практически важное семейство так называемых «усеченных» алгоритмов [2]. Для унифицированной программной реализации всех алгоритмических возможностей решения линейных систем в иерархию введен специальный родительский класс **TruncatedNewton**. В данном классе определена ссылка на алгоритмический объект класса **LinearSystemAlgorithm** и реализованы необходимые методы установки алгоритма, предназначенного для решения линейных систем.

5.3. Класс методов секущих SecantUpdate

Важную группу квазиньютоновских методов образуют методы секущих, представленные в иерархии классом **SecantUpdate**. Методы секущих применяют аппроксимацию гессиана с использованием явных формул коррекции вида $H_{k+1} = H_k + \Delta_k$. В качестве матриц коррекции Δ_k обычно используются малоранговые симметричные преобразования, что позволяет эффективно применять известные формулы пересчета при обращении или факторизации гессиана. Для обеспечения глобальной сходимости используется оптимальный выбор шага λ_k, а также периодическое обновление итерационного процесса. Для этого после каждых p итераций в качестве начальной точки x^{p+1} выбирается последняя полученная точка x^p, а в качестве обновленной матрицы H_{p+1} выбирается первоначальная матрица H_0. Простейшим способом обновления является выбор в качестве H_0 единичной матрицы.

Предположим, что класс **SecantUpdate** определяет чисто виртуальный метод вычисления коррекции Δ_k и реализует методы установки шага p и матрицы обновления H_0, методы пересчета матриц H_k^{-1} и их факторизаций и метод вычисления итерации. Тогда реализация

частных методов секущих класса **SecantUpdate** может быть сведена к определению лишь одного метода вычисления самой коррекции.

Методы секущих представлены в иерархии классами метода Пауэлла–Бройдена **PowellBroyden**, обобщенного метода Бройдена **GenericBroyden**, метода симметричной одноранговой модификации **UniRankUpdate**, метода Давидона–Флетчера–Пауэлла **DavidonFletcherPowell** и метода Бройдена–Флетчера–Гольдфарба–Шанно **BroydenFletcherGoldfarbShanno**.

Класс **PowellBroyden** реализует симметричную формулу коррекции ранга два

$$\Delta_k = \frac{(\gamma_k - H_k\delta_k)\,\delta_k^T + \delta_k\,(\gamma_k - H_k\delta_k)^T}{\delta_k^T\delta_k} - \frac{\langle \gamma_k - H_k\delta_k,\ \delta_k \rangle\,\delta_k\delta_k^T}{(\delta_k^T\delta_k)^2},$$

где $\delta_k = x^{k+1} - x^k$, $\gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k)$.

Класс **UniRankUpdate** предоставляет алгоритмическую возможность симметричной одноранговой коррекции

$$\Delta_k = \frac{(\gamma_k - H_k\delta_k)(\gamma_k - H_k\delta_k)^T}{(\gamma_k - H_k\delta_k)^T\delta_k}.$$

Обе приведенные формулы сохраняют симметричность аппроксимации гессиана, что является, безусловно, важным для эффективного практического использования. Вместе с тем, хорошо известны симметричные формулы коррекции, сохраняющие и положительную определенность аппроксимации гессиана. Рассмотрим наиболее известные из них.

Класс **DavidonFletcherPowell** реализует обратную положительно определенную формулу секущих ранга два

$$\Delta_k = \frac{(\gamma_k - H_k\delta_k)\,\gamma_k^T + \gamma_k\,(\gamma_k - H_k\delta_k)^T}{\gamma_k^T\delta_k} - \frac{\langle \gamma_k - H_k\delta_k,\ \delta_k \rangle\,\gamma_k\gamma_k^T}{(\gamma_k^T\delta_k)^2}.$$

Формула сохраняет положительную определенность матрицы H_{k+1}, если только матрица H_k была положительно определена и $\gamma_k^T\delta_k > 0$.

Класс **BroydenFletcherGoldfarbShanno** реализует иную положительно определенную формулу коррекции ранга два

$$\Delta_k = \frac{\gamma_k\gamma_k^T}{\gamma_k^T\delta_k} - \frac{H_k\delta_k\delta_k^T H_k}{\delta_k^T H_k\delta_k}.$$

Данный метод хорошо зарекомендовал себя во многих практических приложениях и признан одним из наиболее эффективных методов секущих.

В заключение раздела приведем еще одну формулу секущих, предложенную Бройденом

$$\Delta_k = \frac{(\gamma_k - H_k\delta_k)\,\gamma_k^T + \gamma_k\,(\gamma_k - H_k\delta_k)^T}{\gamma_k^T\delta_k} - \frac{\langle \gamma_k - H_k\delta_k,\ \delta_k \rangle\,\gamma_k\gamma_k^T}{(\gamma_k^T\delta_k)^2} -$$
$$- \varphi\ (\delta_k^T H_k\delta_k)\left[\frac{\gamma_k}{\gamma_k^T\delta_k} - \frac{H_k\delta_k}{\delta_k^T H_k\delta_k}\right]\left[\frac{\gamma_k}{\gamma_k^T\delta_k} - \frac{H_k\delta_k}{\delta_k^T H_k\delta_k}\right]^T,$$

где φ — некоторый параметр метода. Данная формула является обобщением рассмотренных выше методов и включает в себя формулу Давидона–Флетчера–Пауэлла при $\varphi = 0$, формулу Бройдена–Флетчера–Гольдфарба–Шанно при $\varphi = 1$ и формулу симметричной одноранговой коррекции при $\varphi = \delta_k^T H_k\delta_k \,/\, (\delta_k^T H_k\delta_k - \delta_k^T\gamma_k)$. Обобщенный метод Бройдена реализуется классом иерархии **GenericBroyden**, который является родительским для рассмотренных выше классов **UniRankUpdate**, **DavidonFletcherPowell**, **BroydenFletcherGoldfarbShanno**. В интерфейсе класса **GenericBroyden** предусмотрен виртуальный метод вычисления параметра φ. Рассмотренные выше частные классы методов секущих реализуются путем переопределения метода вычисления данного параметра. Заметим, что для эффективной реализации целесообразнее переопределить не метод вычисления параметра, а сам метод вычисления коррекции. Предложенный подход к программной реализации обеспечивает обе отмеченные возможности.

5.4. Класс методов сопряженных направлений ConjugateDirections

Важное место в теории оптимизации занимают методы сопряженных направлений. В случае квадратичных функций данные методы обеспечивают сходимость к решению за конечное число шагов. В связи с этим определенный практический и теоретический интерес приобретает применение данных методов для произвольных нелинейных функционалов. Абстрактный класс **ConjugateDirections**, наследуемый непосредственно от класса **OptimizationMultiVariateAlgorithm**, реализует обобщенный вариант метода сопряженных направлений. Задание начального приближения и вычисление итерации происходит в соответствии с общей схемой, реализуемой классом **OptimizationMultiVariateAlgorithm**. Выбор длины шага на основе правил наискорейшего или экономичного спуска также осуществляется родительским классом.

В классе **ConjugateDirections** определен виртуальный метод вычисления очередного независимого направления p^{k+1} на основе следующего рекуррентного соотношения:

$$p^{k+1} = -g^{k+1} + \beta_k\, p^k.$$

Реализация данного метода основана на вспомогательном виртуальном методе конструирования градиентного направления $g^{k+1} = \nabla f(x^{k+1})$ и чисто виртуальном методе расчета коэффициента β_k, конкретизируемом в наследуемых классах. Для обеспечения глобальной сходимости методов сопряженных направлений в классе **ConjugateDirections** определены и реализованы необходимые методы установки периода обновления итераций и выбора начального направления перемещения $p^0 = -\nabla f(x^0)$.

В предлагаемой иерархии методы сопряженных направлений представлены методом Дэниэла, вариантами Флетчера–Ривза и Полака–Рибьера, методом параллельных касательных и методами Розенброка и Пауэлла, реализуемых классами **Daniel**, **FletcherReeves**, **PolakRibiere**, **ParallelTangents**, **Rosenbrock** и **Powell** соответственно.

Реализация классов сводится к программированию метода расчета коэффициента рекуррентной формулы β_k. В алгоритме Дэниэла **Daniel** метод вычисления коэффициента основывается на формуле

$$\beta_k = -((g')^{k+1} g^k)^T p^k / ((g')^{k+1} p^k)^T p^k ,$$

в которой необходимо используется гессиан функционала. Данного недостатка лишены варианты Флетчера–Ривза и Полака–Рибьера, которые реализуются аналогичным образом. Поскольку основные методы вычислений реализуются уже в родительском классе **ConjugateDirections**, в классах **FletcherReeves**, **PolakRibiere** лишь конкретизируются методы вычисления коэффициентов β_k на основе соответствующих формул:

$$\beta_k = \frac{(g^{k+1})^T g^{k+1}}{(g^k)^T g^k} \quad \text{и} \quad \beta_k = \frac{(g^{k+1})^T (g^{k+1} - g^k)}{(g^k)^T g^k} .$$

Заметим, что все рассмотренные выше методы сопряженных направлений по существу являются обобщениями метода сопряженных градиентов и автоматически сводятся к нему в случае квадратичных функционалов. Тем не менее, существуют методы, которые не сводятся к методу сопряженных градиентов, хотя и носят черты сходства с предыдущими алгоритмами. К числу таких методов следует отнести, прежде всего, методы Розенброка и Пауэлла, реализуемые классами **Rosenbrock** и **Powell**.

Обсудим возможности объектной реализации данных методов на примере более значимого метода Пауэлла. В алгоритме задается начальная точка x^0 и n линейно независимых направлений $p^1, p^2, ..., p^n$. В простейшем случае задаются направления координатных осей. Далее вычисляется последовательность точек $x^1, x^2, ..., x^n$ путем последовательного решения одномерных минимизационных задач, как и в общей схеме методов сопряженных направлений. Затем система

направлений модифицируется путем переопределения одного из направлений, в котором функционал был минимизирован наименее успешно. Для этого анализируются последовательные разности $f(x^{i-1}) - f(x^i)$, $i = 1, \ldots, n$. Следующие итерации проходят в обновленной системе направлений. Для реализации класса **Powell** достаточно определить методы установки, поддержки и модификации системы направлений, а также переопределить метод вычисления основной итерации. Тем же самым образом осуществляется программирование класса **Rosenbrock**.

5.5. Класс релаксационных методов Relaxation

Релаксационные методы относятся к группе методов оптимизации, не использующих явно производные, и, как правило, применяются в тех случаях, когда оптимизируемая функция недифференцируема и невозможно вычислить градиент или субградиент. Другой ситуацией, благоприятной для применения релаксационных методов, является слабая зависимость компонент решения между собой. Релаксационный процесс относительно легко реализуется, однако может приводить к серьезным проблемам, связанным с обеспечением как глобальной, так и локальной сходимости.

Абстрактный класс **Relaxation** реализует обобщенный релаксационный процесс. Алгоритм стартует из начальной точки x^0 и последовательно минимизирует функционал по одной из переменных при фиксировании остальных, используя один из одномерных алгоритмов.

Для унифицированной реализации данного алгоритмического семейства в интерфейсе класса **Relaxation** определены чисто виртуальные методы выборки текущей и фиксируемых компонент приближенного решения и метод пересчета приближенного решения при переходе к следующей итерации. Основная итерация непосредственно реализуется в данном классе. Релаксационные методы подразделяются на методы последовательной верхней релаксации (ПВР) класса **SuccesiveOverRelaxation** и методы быстрой релаксации класса **FastRelaxation**.

Класс **SuccesiveOverRelaxation** реализует циклический последовательный выбор компонент решения, относительно которых проводится одномерная минимизация. В качестве фиксируемых компонент берутся соответствующие компоненты текущего решения на полной итерации $\bar{x}$, а следующее приближение пересчитывается по формуле: $x^{k+1} = x^k + \omega(\bar{x} - x^k)$. Выбор параметра релаксации ω определяется соответствующим виртуальным методом класса. Классический метод Гаусса–Зейделя **GaussSeidel** реализуется как частный

наследуемый класс **SuccesiveOverRelaxation**, в котором метод выбора релаксационного параметра переопределен тривиальным образом $\omega = 1$.

Одним из способов улучшения последовательного релаксационного процесса является выбор оптимизируемых компонент не в естественном порядке, а в порядке, соответствующем самой высокой абсолютной компоненте градиента. Данный алгоритм реализуется классом **FastRelaxation**, в котором переопределяется метод выбора очередной компоненты решения, относительно которой осуществляется одномерная оптимизация.

5.6. Класс методов нелинейной квадратичной оптимизации

Одной из частных задач нелинейной оптимизации является нелинейная задача о наименьших квадратах, состоящая в минимизации квадратичного функционала специального вида $g(x) = \frac{1}{2} F(x)^T F(x)$. Для решения данного класса задач может применяться любой из методов многомерной оптимизации, рассмотренных выше. Причем предлагаемая система классов позволяет это осуществить и программным образом, поскольку в функциональной иерархии класс нелинейных квадратичных функций **NonlinearQuadraticFunction** наследуется от класса произвольных функций нескольких переменных **ScalarMultiVariateFunction**.

Вместе с тем, существуют более эффективные подходы к решению задач данного класса. Один из них состоит в применении метода Ньютона к градиентному уравнению для экстремума квадратичного функционала $g'(x) = F'(x)^T F(x) = 0$. Данный подход приводит к итерации

$$x^{k+1} = x^k - \omega_k \left[F''(x^k) F(x^k) + F'(x^k)^T F'(x^k) \right]^{-1} F'(x^k)^T F(x^k)$$

и представлен в иерархии классом **QuadraticOptimizationNewton**. Данный класс наследуется от класса **NonlinearQuadraticOptimizationAlgorithm** и использует его методы доступа к самой квадратичной функции $g(x)$ и к ее функциональной компоненте $F(x)$.

Другая возможность заключается в применении метода Ньютона непосредственно к системе уравнений $F(x) = 0$, что приводит к методу Гаусса–Ньютона, представленному в иерархии классом **GaussNewton**. Обобщенный вариант метода класса **ModifiedGaussNewton** выражается следующей формулой

$$x^{k+1} = x^k - \omega_k \left[F'(x^k)^T F'(x^k) + \lambda_k I \right]^{-1} F'(x^k)^T F(x^k).$$

Как и в методах ньютоновского типа, значение параметра ω_k определяется путем минимизации функционала в рассчитанном

направлении спуска, а значение λ_k выбирается достаточно большим, чтобы обеспечить положительную определенность аппроксимации якобиана отображения $F(x)$.

Частные классы **GaussNewton**, **DampedGaussNewton** и **LevenbergMarquardt** определяют основной, демпфированный вариант метода Гаусса–Ньютона и метод Левенберга–Марквардта, реализуемые при частном выборе параметров $\omega_k = 1$, $\lambda_k = 1$; $\lambda_k = 0$ и $\omega_k = 1$.

Построение данных классов полностью аналогично рассмотренной выше организации классов, реализующих методы ньютоновского типа.

6. Результаты проектирования и реализации математической объектно-ориентированной библиотеки

Таким образом, рассмотрены основные результаты объектной систематизации и классификации задач и методов нелинейной безусловной оптимизации. Предложенная функциональная классификация включает основные типы математических функций и определяет имеющееся многообразие постановок задач оптимизации. Разработанная алгоритмическая классификация охватывает наиболее важные семейства методов оптимизации, предназначенных для решения как одномерных, так и многомерных задач. Вопрос о включении в классификацию того или иного метода решался достаточно субъективно — на основе нашей оценки его практической ценности и той методологической роли, которую он играет в самой теории оптимизации. Естественно, что многие известные оптимизационные алгоритмы оказались обойденными в ходе объектного анализа. Тем не менее, развитие построенной классификации может быть продолжено естественным путем на основе включения в ее иерархическую структуру других методов и алгоритмов.

Разрабатываемая структура объектной классификации допускала довольно широкое множество приемлемых проектных решений. В конечном счете, она определялась последовательностью тех математических, вычислительных и программных свойств у выделенных групп функциональных и алгоритмических объектов, которые принимались в качестве классификационных критериев. Важно заметить, что требование логической организации классовой иерархии, предполагающее строгое соблюдение отношений общности между наследуемыми классами объектов, и стремление к ее оптимальной программной реализации часто конфликтовали между собой.

Наглядным примером может служить введение в иерархию метода Бройдена класса **GenericBroyden**, обобщающего три различных метода пересчета секущих классов **UniRankUpdate**, **DavidonFletcherPowell** и **BroydenFletcherGoldfarbShanno**. С математической точки зрения метод

Бройдена действительно является обычным параметрическим обобщением перечисленных методов и класс **GenericBroyden** следует рассматривать в качестве родительского для частных классов методов секущих. Программно реализовав класс **GenericBroyden**, мы можем построить все остальные классы путем тривиального переопределения единственного метода вычисления параметра. Вместе с тем, описанную программную реализацию вряд ли стоит считать оптимальной для частных наследуемых классов. Во-первых, она привела к избыточному хранению данных — в данном случае параметра метода. Во-вторых, использование обобщенной математической формулы метода Бройдена с конкретизацией значения параметра крайне неэффективно для частных методов. Например, для метода симметричной одноранговой модификации класса **UniRankUpdate** данная формула могла бы быть существенно упрощена, а главное, позволила бы при пересчете отказаться от двухранговой модификации, присущей методу Бройдена.

Всякий раз, когда возникали подобные противоречия мы разрешали их все же в сторону концептуальной логической целостности разрабатываемой объектной классификации. При этом в первую очередь преследовалась цель обеспечить наглядность и смысловую наполненность классификации, делающих ее привлекательной для потенциальных пользователей, к числу которых в этом случае можно было бы отнести и математиков, профессионально не владеющих программированием. Вместе с тем, анализировались возможности эффективной программной реализации объектной классификации и практически во всех проблемных ситуациях находились альтернативные решения, позволяющие совместить ее семантическую целостность с приемлемой эффективностью. В приведенном примере подобная альтернатива могла бы состоять в переопределении не метода вычисления параметра класса **GenericBroyden**, а одного из его виртуальных методов, реализующих более мощные алгоритмические ветви, например, метода вычисления всей итерации. При этом хранение одного лишнего параметра для каждого алгоритмического объекта не является серьезным ограничением даже для программных приложений, в которых могут участвовать одновременно несколько оптимизационных подходов.

Важной конструктивной стороной построенной объектной классификации является ее практическая направленность и программная реализуемость. Фактически, описанная объектная классификация, в которой определяются не только основные классы объектов и отношения наследования между ними, но и полный набор инкапсулируемых ими данных и методов, представляет достаточно детальный проект математического обеспечения для решения широкого класса оптимизационных задач. Более того, значительная часть предложенной

объектной классификации, включающая функциональные классы, большинство одномерных алгоритмов и некоторые многомерные методы ньютоновского типа, была программно реализована на языке Си++ и включена в состав разрабатываемой математической объектно-ориентированной библиотеки. Планируется реализация и других многомерных оптимизационных алгоритмов, представленных в классификации.

Существенной особенностью математической библиотеки является ее инструментальный характер, допускающий дальнейшее развитие проблемно-методического репертуара на единой объектной основе. Данная черта особенно важна для разработки практических приложений, нуждающихся в определенной адаптации используемых библиотечных средств. Применяемая объектная технология обеспечивает при этом значительную алгоритмическую и программную общность, заключающуюся в возможности унифицированной модификации отдельных компонент алгоритма. Например, для методов спуска возможно изменить и направление поиска и стратегию выбора шага, а при необходимости и вспомогательный алгоритм решения линейных систем. Проиллюстрируем данное свойство на примере конструирования известных комбинированных алгоритмов, также включенных в алгоритмическую классификацию.

Класс методов ПВР–Ньютона **SuccesiveOverRelaxationNewton** может быть построен путем наследования класса **SuccesiveOverRelaxation** и переопределения виртуального метода установки вспомогательного алгоритма одномерной минимизации — в данном случае установки на одномерный метод Ньютона **Newton**. Аналогично реализуется известный градиентный метод Ньютона **GradientNewton**, использующий обычные градиентные направления в сочетании с одномерным методом Ньютона для выбора оптимальной длины шага [7]. Реализация такого метода сводится к созданию класса, наследуемого от **Gradient** и переопределяющего метод выбор длины шага путем задания одномерного метода Ньютона **Newton**.

Не менее важная группа комбинированных оптимизационных алгоритмов может быть построена в результате сочетания основного метода с тем или иным необходимым алгоритмом решения линейных систем. В частности, практически важное семейство комбинированных методов Ньютона и сопряженных градиентов **NewtonConjugateGradient**, которое составляет основу известного пакета TNPACK [2], может быть реализовано путем наследования основного метода класса **Newton** и установки вспомогательного метода решения линейной системы на алгоритмический объект класса **ConjugateGradient**, принадлежащего в свою очередь классу **LinearSystemAlgorithm**. С учетом того, что алгоритм

класса **ConjugateGradient** может применяться к линейным системам с различной формой матричной разреженности, достигается еще большая общность, которая также представляет серьезный практический и исследовательский интерес [8].

7. Заключение

Таким образом, предложен подход к объектной систематизации и классификации задач и методов нелинейной безусловной оптимизации. Разработанная объектная классификация охватывает наиболее важные группы математических функций, одномерных и многомерных алгоритмов оптимизации и в значительной степени отражает методологические построения самой теории оптимизации. Вместе с тем, спроектированная классификация носит конструктивный характер и допускает непосредственные программные реализации. Значительная часть объектной системы была реализована на языке Си++ и включена в математическую объектно-ориентированную библиотеку. Мощные инструментальные возможности математической библиотеки определяют перспективу дальнейшего развития проблемно-методического репертуара и адаптации его к разнообразным вычислительным приложениям.

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

ЛИТЕРАТУРА

1. More J., Sorensen D., Garbow B., Hillstrom K. The MINPACK Project. // Sources and Development Of Mathematical Software, ed. W. Cowell. Prentice Hall, 1984.
2. Schlick T., Fogelson A. TNPACK — A truncated Newton minimization package for large-scale problems. // ACM Trans. Math. Softw., v. 18, № 1 (March 1992), pp. 46–111.
3. Hopkins T., Phillips C. Numerical Methods in Practice: Using the NAG Library. Addison-Wesley, Reading, MA, 1988.
4. Rice J.R. Numerical Methods, Software and Analysis: IMSL Reference Edition. McGraw-Hill, New-York, 1983.
5. Readings in Object-Oriented Systems and Applications, ed. David C. Rine. IEEE Computer Society Press, 1995.
6. Мину М. Математическое программирование. Теория и алгоритмы: Пер. с фр. — М.: Мир, 1990.
7. Ортега Дж., Рейнболдт В. Итерационные методы решения нелинейных систем уравнений со многими неизвестными: Пер. с англ. — М.: Мир, 1975.

8. Дэннис Дж.-мл., Шнабель Р. Численные методы безусловной оптимизации и решения нелинейных уравнений: Пер. с англ. — М.: Мир, 1988.
9. Васильев Ф.П. Численные методы решения экстремальных задач. — М.: Наука, 1988.
10. Евтушенко Ю.Г. Численные методы решения экстремальных задач и их применение в системах оптимизации. — М.: Наука, 1982.
11. Карманов В.Г. Математическое программирование. — М.: Наука, 1986.
12. Морозов С.В., Семенов В.А. Объектно-ориентированное программирование задач численного анализа. // Вопросы кибернетики. Приложения системного программирования. — М.: НСК РАН, 1995, с. 189–211.
13. Семенов В.А., Тарлапан О.А. Технологии реализации разреженных матричных классов. // Вопросы кибернетики. Приложения системного программирования. — М.: НСК РАН, 1995, с. 164–188.
14. Семенов В.А. Об объектно-ориентированном подходе к разработке численного математического обеспечения. // Вопросы кибернетики. Приложения системного программирования. — М.: НСК РАН, 1995, с. 140–163.
15. Семенов В.А., Морозов С.В. Объектно-ориентированное программирование квадратурных методов. // Настоящий сборник.