ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 39 за 2016 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Самотохин А.С.

Численный метод условной минимизации с использованием кэширования и модифицированной функции Лагранжа

Рекомендуемая форма библиографической ссылки: Самотохин А.С. Численный метод условной минимизации с использованием кэширования и модифицированной функции Лагранжа // Препринты ИПМ им. М.В.Келдыша. 2016. № 39. 25 с. doi:10.20948/prepr-2016-39
URL: http://library.keldysh.ru/preprint.asp?id=2016-39

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Российской академии наук

А.С. Самотохин

Численный метод условной минимизации с использованием кэширования и модифицированной функции Лагранжа

Москва — 2016

Самотохин А.С.

Численный метод условной минимизации с использованием кэширования и модифицированной функции Лагранжа

Рассматривается численный метод минимизации функции нескольких переменных при наличии ограничений в форме равенств и/или неравенств. Метод основан на использовании модифицированной функции Лагранжа и кэшировании результатов вычислений. Подробно описана вычислительная схема предлагаемого метода. Предложен оригинальный метод оценки частных производных минимизируемой функции.

Ключевые слова: численный метод, условная минимизация, кэширование.

Alexander Sergeevich Samotokhin

Numerical method of conditional minimization with using caching and modified Lagrange function

The numerical method to minimize function of several variables supplied by limitations in forms of equalities or/and inequalities is considered. The method is based on using of modified Lagrange function and caching of calculation. Computation scheme is described in details. The original method to estimate partial derivative is proposed.

Key words: numerical method, conditional minimization, caching.

Постановка задачи

В проектной и оперативной баллистике часто возникает необходимость в решении задач условной минимизации функции нескольких переменных. Примером такой задачи является задача выбора оптимальной схемы возврата с орбиты искусственного спутника Луны с последующей посадкой в заданном районе земной поверхности, которая рассматривалась в работе [1]. В этой задаче требуется найти оптимальный импульс скорости, который бы обеспечивал переход на траекторию возврата с заданной высотой условного перицентра и плоскостью, проходящей через центральную точку полигона посадки.

Рассмотрим формальную постановку задачи условной минимизации функции нескольких переменных при наличии ограничений:

$$\begin{aligned} & f(\mathbf{x}) \to \min; \\ & g_j(\mathbf{x}) \le 0,\ j = 1,\ldots,s; \\ & g_j(\mathbf{x}) = 0,\ j = s+1,\ldots,m. \end{aligned} \qquad (1)$$

Здесь $f(\mathbf{x}), g_j(\mathbf{x})$ – известные функции аргумента $\mathbf{x}$ размерности n. В дальнейшем, не нарушая общности, будем считать, что все ограничения задаются в форме неравенств. Поскольку речь идет о численном методе, любое ограничение в виде равенства может быть выполнено с некоторой конечной точностью, что эквивалентно ограничениям в форме неравенств.

Известно [4], что решением поставленной задачи является седловая точка функции Лагранжа

$$L(\mathbf{x},\lambda) = f(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}), \qquad (2)$$

в которой достигается минимум функции Лагранжа по аргументу $\mathbf{x}$ и максимум по положительным множителям λ_j. Однако на практике поиск седловой точки функции Лагранжа с помощью итерационных процедур, использующих ее явную форму, редко приводит к успеху из-за проблем со сходимостью.

Ф.П. Васильевым в работе [2] был предложен метод, основанный на использовании модифицированной функции Лагранжа следующего вида:

$$M(\mathbf{x},\lambda) = f(\mathbf{x}) + \frac{1}{2A} \sum_{j=1}^{m} \left[\left(\lambda_j + A g_j(\mathbf{x}) \right)^{+} \right]^2. \qquad (3)$$

Здесь A – некоторая положительная константа, а под обозначением a^+ здесь и далее понимается выражение

$$a^+ = \max(0, a). \tag{4}$$

Предложенный в работе [2] метод представляет собой следующую итерационную процедуру. Пусть на k-ой итерации имеется некоторое приближение для вектора аргументов $\mathbf{x}_k$ и вектора множителей Лагранжа λ_k. В этом случае следующее приближение для вектора аргументов $\mathbf{x}_{k+1}$ получается в результате решения задачи безусловной минимизации для вспомогательной функции следующего вида:

$$\Phi(\mathbf{x}) = \frac{1}{2}|\mathbf{x} - \mathbf{x}_k|^2 + \alpha M(\mathbf{x}, \lambda_k), \tag{5}$$

где α – некоторая положительная константа. Следующее приближение для вектора множителей Лагранжа вычисляется по формуле

$$\lambda_{k+1} = \left(\lambda_k + Ag(x_{k+1})\right)^+. \tag{6}$$

Изложенный выше метод был принят за основу при построении вычислительной схемы и создании программного обеспечения, ориентированного на решение практических задач в баллистическом центре ИПМ.

Основные допущения

При разработке вычислительной схемы, которая будет рассмотрена в последующих разделах, во внимание были приняты следующие особенности, присущие задачам проектной и оперативной баллистики.

1. Вычисление целевой функции $f(\mathbf{x})$, а также функций ограничений $g(\mathbf{x})$ в задачах баллистики, как правило, связано с численным интегрированием уравнений движения и является операцией, требующей наибольших вычислительных затрат. Как следствие, при использовании метода, описанного в предыдущем разделе, целесообразно предусмотреть кэширование значений этих функций, чтобы избежать повторного их вычисления в одних и тех же (или близко расположенных) точках.
2. Вычисление частных производных для функций $f(\mathbf{x})$ и $g(\mathbf{x})$, которые нужны для выполнения безусловной минимизации, также требует значительных вычислительных ресурсов. Поэтому было принято решение использовать для их оценки метод наименьших квадратов с использованием значений, ранее накопленных в кэше.

3. Как правило, в задачах баллистики величины, входящие в вектор аргументов $\mathbf{x}$, имеют разную физическую размерность. Вследствие этого целесообразно предусмотреть механизм, который позволял бы эффективно выполнять минимизацию вне зависимости от состава вектора аргументов.
4. В качестве основного критерия достижения искомого решения используется точность по аргументу минимизации $\mathbf{x}$, а не по целевой функции $f(\mathbf{x})$.

Пространство аргументов

Чтобы обеспечить эффективное кэширование, для пространства аргументов вводятся правила дискретизации и предусматривается использование так называемых регулярных сеток (см. ниже). Перед началом оптимизации пользователь должен задать следующие константы, которые однозначно определяют эти правила.

$\delta\mathbf{x}_0 = \{\delta x_{0i}, i = 1,...,n\}$ – вектор, определяющий начальный номинальный шаг в пространстве аргументов.

s – фактор масштабирования: целое число, больше 1.

$k_{\max}$ – максимальный уровень масштабирования: положительное целое число.

Поясним смысл этих констант. На каждой итерации в предлагаемом алгоритме используется понятие уровня масштабирования аргумента. Это положительное целое число *k*, удовлетворяющее условию

$$0 \le k \le k_{\max}. \qquad (7)$$

На первой итерации *k* полагается равным 0. В дальнейшем вычислительный алгоритм может принимать решение об изменении значения *k* в большую или в меньшую сторону в зависимости от поведения целевой функции.

При известном уровне масштабирования *k* номинальный шаг по аргументу вычисляется по формуле

$$\delta\mathbf{x}_k = s^{-k}\delta\mathbf{x}_0. \qquad (8)$$

Полученное по формуле (8) значение $\delta\mathbf{x}_k$ используется в качестве вариаций при вычислении частных производных целевой функции, при ограничении длины фактического шага и при определении факта достижения минимума.

Значение начального номинального шага $\delta\mathbf{x}_0$ используется также для оценки расстояний между точками в пространстве аргументов, которое вычисляется по формуле

$$d(\mathbf{x}_1,\mathbf{x}_2)=\|\mathbf{x}_1-\mathbf{x}_2\|=\max_i\left(\frac{|x_{1i}-x_{2i}|}{\delta x_{0i}}\right). \qquad (9)$$

Отметим, что значение d, получаемое по этой формуле, всегда является безразмерным.

Рассмотрим конкретный пример. Пусть вектор аргумента минимизации представляет собой 3 компоненты вектора скорости и пользователем были установлены следующие значения: $\delta x_{0i}=1$ м/с, $s=10$, $k_{\max}=3$. Это означает, что на первой итерации для вычисления частных производных будут использоваться вариации по аргументу, равные 1 м/с, а окончательное решение будет получено с точностью до 10^{-3} м/с.

Регулярные сетки в пространстве аргументов

Пусть имеется некоторая точка $\mathbf{x}_0$ в пространстве аргументов. Под регулярной сеткой с шагом $\delta\mathbf{x}$ будем понимать такое множество точек в пространстве аргументов, для которых

$$\mathbf{x}=\mathbf{x}_0+\Delta\mathbf{x}, \qquad (10)$$

где Δx_i равны либо 0, либо $\pm\delta x_i$.

Очевидно, число точек в такой регулярной сетке зависит от размерности аргумента n и в общем случае равно 3^n (включая центральную точку). Если потребовать, чтобы число ненулевых приращений Δx_i в сетке не превышало заданного значения $p\le n$, то число точек будет меньшим. Практический интерес представляет случай $p=2$. В этом случае число точек сетки равно $2n^2+1$.

Поясним сказанное на конкретных примерах.

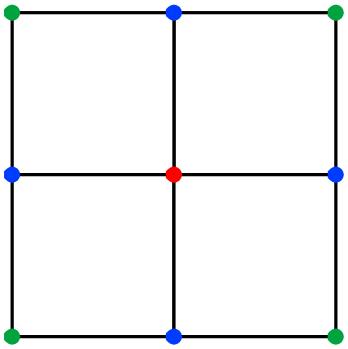

Рис. 1. Регулярная сетка при n=2.

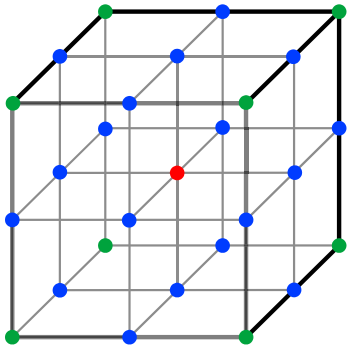

Рис. 2. Регулярная сетка при n=3.

На рис. 1 показана регулярная сетка при размерности аргумента n=2. Здесь общее число точек сетки равно 9, а число точек, которые удовлетворяют p=1,

равно 5 (эти точки обозначены синим и красным цветом). Аналогично на рис. 2 показана регулярная сетка для n=3 с 27 точками. Здесь синим и красным цветом выделены точки, которые образуют сетку с p=2 (число таких точек равно 19).

В дальнейшем будем использовать только сетки с p=n и p=2.

Алгоритм оценки частных производных

При выполнении минимизации с помощью методов второго порядка необходима оценка частных производных 2-го порядка. Наиболее часто для этой цели используются формулы, основанные на методе конечных разностей:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{f(\mathbf{x}+\delta x_i) - f(\mathbf{x}-\delta x_i)}{2\delta x_i}, \tag{11}$$

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i^{\,2}} = \frac{f(\mathbf{x}+\delta x_i) + f(\mathbf{x}-\delta x_i) - 2f(\mathbf{x})}{\delta x_i^{\,2}}, \tag{12}$$

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{1}{4\delta x_i \delta x_j}\begin{pmatrix} f(\mathbf{x}+\delta x_i+\delta x_j) + f(\mathbf{x}-\delta x_i-\delta x_{sj}) \\ -f(\mathbf{x}+\delta x_i-\delta x_j) - f(\mathbf{x}-\delta x_i+\delta x_j) \end{pmatrix}. \tag{13}$$

Однако использование этих формул может давать очень серьезную погрешность в случае, когда целевая функция имеет «овраги». Рассмотрим конкретный пример.

На рис. 3 показано поведение функции, у которой изолинии ориентированы примерно под углом 45 градусов к осям координат. Значения функции в узловых точках показаны на рисунке цифрами. Применение формул (11)-(13) для оценки частных производных в центральной точке приводит к следующим результатам:

$$\frac{\partial f}{\partial x_1} = 0, \quad \frac{\partial f}{\partial x_2} = 0, \frac{\partial^2 f(\mathbf{x})}{\partial x_1^{\,2}} = 4, \quad \frac{\partial^2 f(\mathbf{x})}{\partial x_2^{\,2}} = 4$$

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} = -1.5.$$

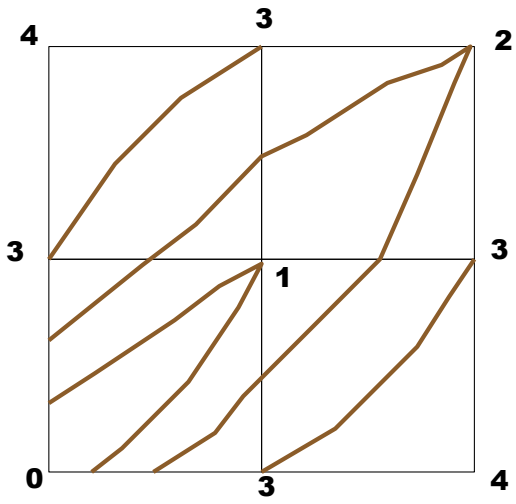

Рис. 3. Пример овражной функции

Очевидно, при таком способе оценки частных производных применение методов оптимизации 1-го или даже 2-го порядка не позволит сдвинуться от центральной точки. Хотя приведенный выше пример является несколько

утрированным, в задачах условной минимизации овражные функции встречаются достаточно часто.

Рассмотрим альтернативный способ оценки частных производных, основанный на использовании метода наименьших квадратов. Будем использовать следующую аппроксимацию поведения функции $f(\mathbf{x})$ в окрестности точки $\mathbf{x}_0$:

$$f(\mathbf{x}_0+\Delta\mathbf{x})=f(\mathbf{x}_0)+\mathbf{g}(\mathbf{x}_0)\cdot\Delta\mathbf{x}+\frac{1}{2}\Delta\mathbf{x}^T\mathbf{G}(\mathbf{x}_0)\Delta\mathbf{x}. \qquad (14)$$

Здесь $\mathbf{g}(\mathbf{x}_0)$ – вектор градиента функции $f(\mathbf{x})$, $\mathbf{G}(\mathbf{x}_0)$ – матрица частных производных 2-го порядка этой функции.

Введем в рассмотрение вектор оцениваемых параметров $\mathbf{s}=\left\{\frac{\partial f}{\partial x_i},\frac{\partial^2 f}{\partial x_i\partial x_j},i=1..n,\ j=i...n\right\}$, который включает в себя частные производные функции $f(\mathbf{x})$ 1-го и 2-го порядков. Очевидно, размерность этого вектора равна $N=\frac{n(n+3)}{2}$.

С учетом введенного обозначения уравнение (14) может быть записано в следующей форме:

$$f(\mathbf{x}_0+\Delta\mathbf{x})=f(\mathbf{x}_0)+\sum_{j=1}^{N}s_j\varphi_j(\Delta\mathbf{x}). \qquad (15)$$

Здесь $\varphi_j(\Delta\mathbf{x})$ – простые аналитические функции, форма которых зависит от индекса j, т.е. от оцениваемой частной производной:

$\varphi(\Delta\mathbf{x})=\Delta x_i$ для первых производных $\frac{\partial f}{\partial x_i}$,

$\varphi(\Delta\mathbf{x})=\frac{1}{2}\Delta x_i^2$ для производных $\frac{\partial^2 f}{\partial x_i^2}$ и

$\varphi(\Delta\mathbf{x})=\Delta x_i\Delta x_j$ для производных $\frac{\partial^2 f}{\partial x_i\partial x_j}$.

Рассмотрим следующую постановку задачи. Пусть требуется оценить значения частных производных в точке $\mathbf{x}_0$ по $M\geq N$ точкам, для которых известны координаты $\mathbf{x}_k=\mathbf{x}_0+\Delta\mathbf{x}_k$, $k=1,...,M$ и значения функций в этих точках $f_k=f(\mathbf{x}_k)$. Получим такую оценку из условия минимума функционала

$$\Phi = \sum_{k=1}^{M} w(\Delta \mathbf{x}_k) \left(f_k - f_0 - \sum_{j=1}^{N} s_j \varphi_j (\Delta \mathbf{x}_k) \right)^2 . \qquad (16)$$

Здесь $w(\Delta \mathbf{x}_k)$ – положительная весовая функция, зависящая от $|\Delta \mathbf{x}_k|$. Предполагается, что эта функция должна убывать с ростом $|\Delta \mathbf{x}_k|$. Введение такой весовой функции допускает большие погрешности аппроксимации (15) по мере удаления от $\mathbf{x}_0$.

Поскольку оцениваемые параметры s_j входят в Ф линейно, решение поставленной задачи сводится к решению системы линейных уравнений вида

$$\mathbf{As} = \mathbf{b}, \qquad (17)$$

где $\mathbf{A}$ – симметричная матрица размерности $N \times N$, элементы которой вычисляются по формулам

$$a_{ij} = \sum_{k=1}^{M} w(\Delta \mathbf{x}_k) \varphi_i (\Delta \mathbf{x}_k) \varphi_i (\Delta \mathbf{x}_k), \qquad (18)$$

а $\mathbf{b}$ – вектор размерности N, элементы которого равны

$$b_i = \sum_{k=1}^{M} w(\Delta \mathbf{x}_k)(f_k - f_0) \varphi_i (\Delta \mathbf{x}_k). \qquad (19)$$

Таким образом, решение поставленной задачи сводится к решению системы линейных уравнений (17), в которой матрица $\mathbf{A}$ зависит только от аргумента $\Delta \mathbf{x}_k$ и не зависит от значения функции в этих точках.

Вернемся к примеру, показанному выше на рис. 3. Применим описанную выше методику для оценки частных производных в центральной точке, использовав в качестве исходных 10 точек, показанных на рисунке. Для простоты примем $w(\Delta x_k) = 1$. Результатом будет следующая оценка:

$$\frac{\partial f}{\partial x_1} = 0.333, \quad \frac{\partial f}{\partial x_2} = 0.333, \quad \frac{\partial^2 f(\mathbf{x})}{\partial x_1^{\,2}} = 2, \quad \frac{\partial^2 f(\mathbf{x})}{\partial x_2^{\,2}} = 2, \quad \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} = -1.5 .$$

Следующее приближение, полученное с помощью метода Ньютона, при такой оценке производных даст в результате $\Delta x_1 = -0.667$, $\Delta x_2 = -0.667$ относительно центральной точки.

Пример использования уровней масштабирования

Прежде чем перейти к формальному описанию вычислительной схемы, используемой для минимизации функционала (5), рассмотрим простой пример, который иллюстрирует приемы, описанные в предыдущих разделах.

Будем рассматривать минимизацию функции двух переменных ($n=2$) при следующих условиях: компоненты номинального шага равны друг другу $\delta x_{01}=\delta x_{02}=\delta x_0$, фактор масштабирования $s=5$, максимальный уровень масштабирования $k_{\max}=1$. На представленных ниже рисунках утолщенные линии сетки соответствуют уровню масштабирования $k=0$, а тонкие линии – уровню $k=k_{\max}=1$.

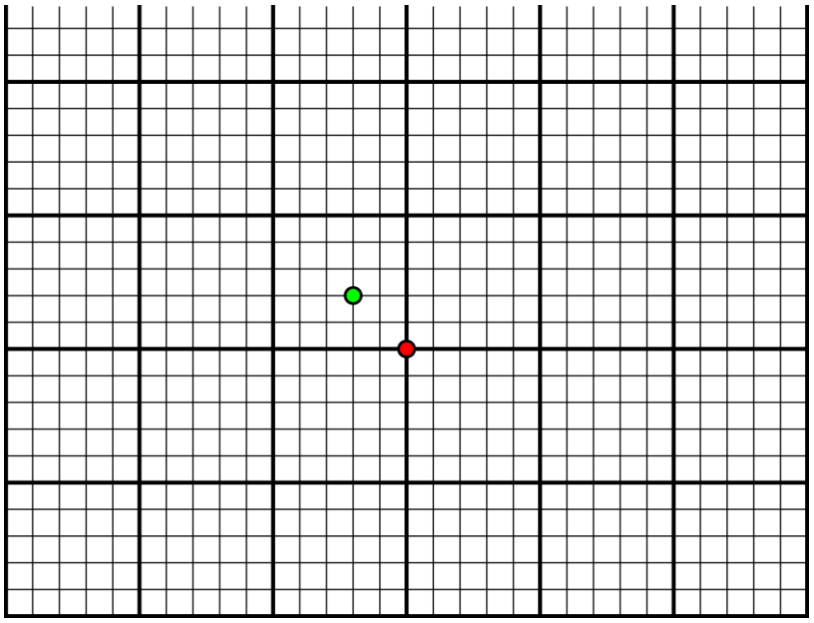

Рис. 4. Начальное приближение и базовая точка.

На рис. 4 зеленым цветом показана точка, соответствующая начальному приближению. Координаты этой точки (как и любой другой) выравниваются на шаг, соответствующий максимальному уровню масштабирования и в ней выполняется вычисление целевой функции. Координаты точки и вычисленное значение сохраняются в кэше. Далее определяются координаты ближайшей точки, координаты которой кратны шагу текущего уровня масштабирования. Назовем эту точку базовой, она обозначена на рис. 4 красным цветом. В ней также выполняется вычисление значения целевой функции с сохранением в кэше координат и вычисленного значения.

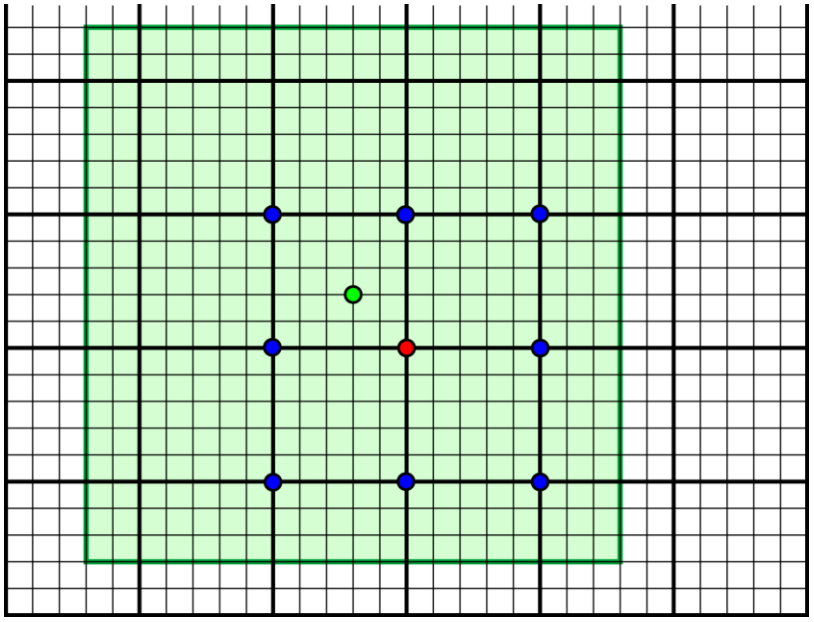

Рис. 5. Первая итерация

Для получения следующего приближения необходимо выполнить оценку частных производных в точке начального приближения, которая обозначена на рис. 4 и 5 зеленым цветом. С этой целью формируется набор точек, образующих регулярную сетку с $p=2$ и шагом δx_0 вокруг базовой точки (которая обозначена на рис. 4 и 5 красным цветом). Точки, образующие такую сетку, показаны на рис. 5 синим цветом. Проводится вычисление значений целевой функции в этих точках, и результат вычислений также сохраняется в кэше.

Оценка частных производных выполняется путем решения системы линейных уравнений (17), в которой элементы матрицы $\mathbf{A}$ и вектора $\mathbf{b}$ вычисляются по формулам (18) и (19). При вычислении матрицы $\mathbf{A}$ и вектора $\mathbf{b}$ используются все имеющиеся точки кэша, для которых расстояние от точки начального приближения, вычисленное по формуле (9), не превышает $2\cdot s^{-k}$. Область, которая удовлетворяет этому условию, выделена на рис. 5 с помощью прямоугольника светло-зеленого цвета. Таким образом, в обсуждаемом примере для оценки частных производных целевой функции будет использовано 10 точек, которые показаны на рис. 5.

После того как проведена оценка частных производных, вычисляется точка следующего приближения (например, с помощью широко известного метода Ньютона). Дальнейший ход вычислений зависит от того, насколько далеко новое приближение будет отстоять от предыдущего. Рассмотрим два возможных случая.

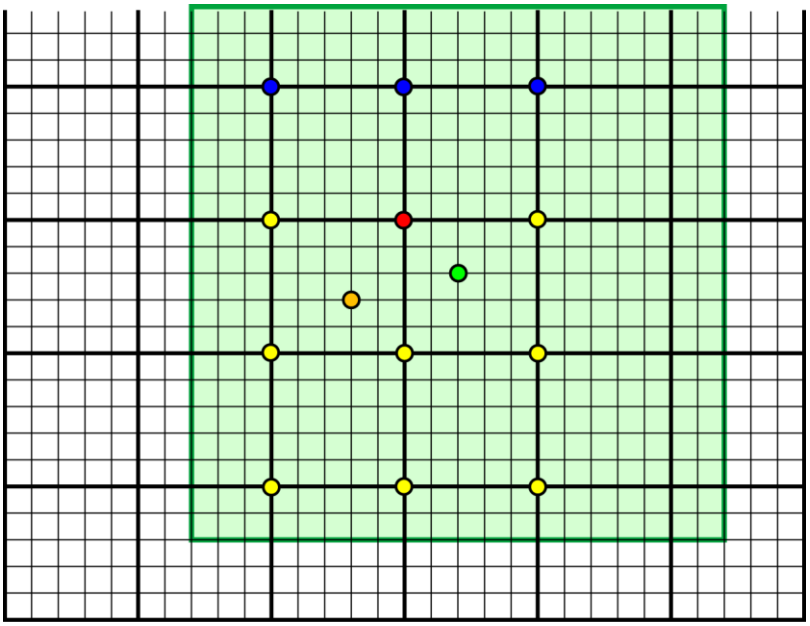

Рис. 6. Вторая итерация с сохранением уровня масштабирования

На рис. 6 показан случай, когда расстояние между точками предыдущего приближения (оранжевый цвет) и текущего (зеленый цвет) больше, чем $0.5 \cdot s^{-k}$. В этом случае уровень масштабирования k сохраняется неизменным. Базовой становится точка, обозначенная красным цветом (она извлекается из кэша). Кэш дополняется 3 точками, которые обозначены на рисунке синим цветом (остальные точки регулярной сетки уже есть в кэше). Как видно из рисунка, в данном случае оценка частных производных будет проводиться уже по 14 точкам.

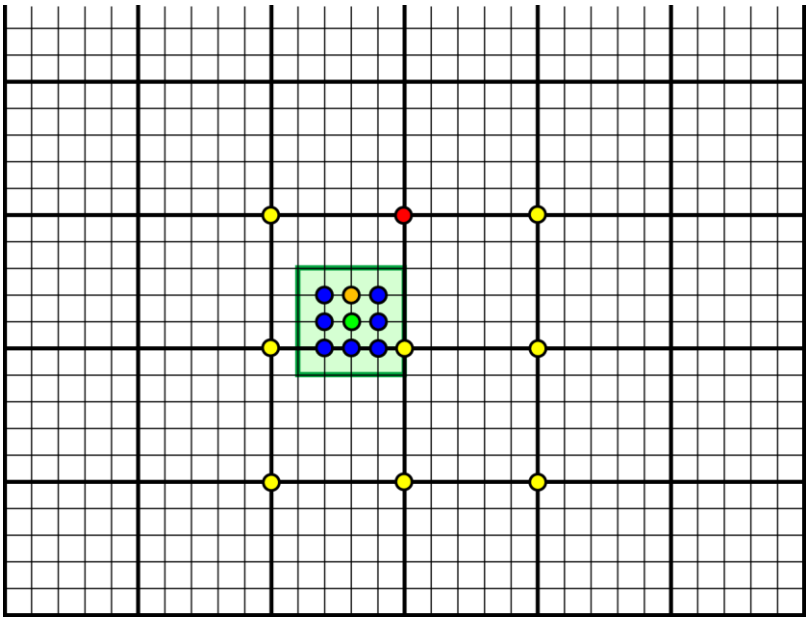

Рис. 7. Вторая итерация с изменением уровня масштабирования

На рис. 7 показан случай, когда следующее приближение (зеленая точка) отстоит от предыдущего (оранжевая точка) менее чем на $0.5 \cdot s^{-k}$. В этом случае принимается решение об увеличении уровня масштабирования аргумента на 1 (т.е. в данном случае *k* полагается равным 1). На этом уровне точка следующего приближения становится базовой для построения регулярной сетки с $p = 2$ и шагом $s^{-1} \cdot \delta x_0$. Точки этой сетки обозначены на рис. 7 синим цветом. Для оценки частных производных из кэша извлекаются все точки, которые отстоят от центральной на расстояние, не превышающее $0.5 \cdot s^{-k}$. На рис. 7 эта область показана с помощью прямоугольника светло-зеленого цвета.

Алгоритм определения точки минимума

Алгоритм, описанный ниже, используется для определения факта достижения точки минимума в задаче безусловной минимизации. В алгоритме используется вспомогательный массив данных размерности $3^n - 1$. Алгоритм включает в себя две части. Первая часть вызывается однократно перед началом безусловной оптимизации и выполняет начальное заполнение вспомогательного массива. Вторая часть алгоритма вызывается всякий раз, когда необходимо проверить факт достижения точки минимума.

Вспомогательная массив состоит из $3^n - 1$ записей, каждая из которых включает в себя вектор $\Delta\mathbf{x}$ размерности n и величину Δf. При вызове первой части алгоритма значения Δf всех записей обнуляются, в вектора $\Delta\mathbf{x}$ помещаются значения координат регулярной сетки с $p = n$ и шагом $s^{-k_{\max}} \delta\mathbf{x}_0$, сформированной в окрестности точки с нулевыми координатами аргументов.

Заполнение векторов $\Delta\mathbf{x}$ выполняется с помощью двух вложенных циклов. Во внешнем цикле переменная *i* изменяется от 0 до $3^n - 1$. Внутри цикла выполняются следующие действия.

1. Присвоение $k = i + 1$
2. Цикл по переменной *j* от 0 до *n*. Внутри цикла выполняются следующие действия.
   2.1. Переменной *m* присваивается значение, равное остатку от целочисленного деления *k* на 3.
   2.2. Если $m = 0$, то *j*-ой компоненте вектора $\Delta\mathbf{x}$ в *i*-той записи присваивается нулевое значение: $\Delta x_{ij} = 0$.
   2.3. Если $m = 1$, то $\Delta x_{ij} = +s^{-k_{\max}} \delta x_{0j}$.
   2.4. Если $m = 2$, то $\Delta x_{ij} = -s^{-k_{\max}} \delta x_{0j}$.
   2.5. Присвоение $k = k/3$ (целочисленная операция).

Вторая часть алгоритма вызывается, когда нужно определить, является ли текущая точка точкой минимума. В качестве исходных данных передаются координаты текущей точки $\mathbf{x}_c$ и значение целевой функции f_c в этой точке.

Алгоритм возвращает значение true, если текущая точка является точкой минимума, и false – в противном случае. Алгоритм включает в себя следующие шаги.

1. Если текущий уровень масштабирования $k < k_{\max}$, то возврат false.
2. Организуется цикл по переменной i от 0 до $3^n - 1$. Внутри цикла выполняются следующие действия.
   2.1. Вычисляются координаты i-ой точки регулярной сетки в окрестности $\mathbf{x}_c$: $\mathbf{x}_i = \mathbf{x}_c + \Delta\mathbf{x}_i$.
   2.2. Вычисляется значение целевой функции в i-ой точке: $f_i = f(\mathbf{x}_i)$ (если в кэше есть точка с координатами $\mathbf{x}_i$, то значение f_i просто извлекается из кэша).
   2.3. В i-ой записи вспомогательного массива полю Δf присваивается значение $\Delta f_i = f_i - f_c$.
   2.4. Если $f_i < f_c$, то выполняется сортировка записей вспомогательного массива в порядке возрастания поля Δf. Работа алгоритма прерывается и происходит возврат false.
3. Выполняется сортировка записей вспомогательного массива в порядке возрастания поля Δf, после чего возвращается значение true.

Цель сортировки, которая предусмотрена на шаге 2.4, состоит в том, что опрос точек регулярной сетки начинается с тех направлений, где появление точки с меньшим значением функционала является наиболее вероятным.

Алгоритм безусловной оптимизации

Приведем формальное описание алгоритма безусловной оптимизации функционала (5), который используется для получения очередного приближения $\mathbf{x}_k$ при фиксированных значениях множителей Лагранжа λ_k. Этот алгоритм представляет собой итерационную процедуру, на каждом шаге которой для получения могут использоваться как широко известные методы минимизации 1-го или 2-го порядка (см., например, [3]), так и методы, связанные с изменением уровня масштабирования.

В алгоритме используются следующие структуры данных:

- кэш, содержащий координаты точек, в которых ранее выполнялись вычисления целевой функции, и вычисленные значения этой функции,
- k – текущий уровень масштабирования,
- t – триггер используемого метода оптимизации: целочисленная переменная, которая может принимать значения 0, 1 или 2.

Перед началом оптимизации кэш является пустым, k=0, t=0. Также подразумевается, что определены вектор начального номинального шага $\delta\mathbf{x}_0$, фактор масштабирования s и максимальный уровень масштабирования $k_{\max}$ (см. выше раздел «Пространство аргументов»).

На вход в процедуру минимизации передается вектор начального приближения $\mathbf{x}_0$. Координаты этого вектор выравниваются на величину шага $s^{-k_{\max}}\delta\mathbf{x}_0$, и происходит вычисление целевой функции в полученной точке. Вычисленное значение и координаты аргумента запоминаются в кэше. Дальнейшие действия представляют собой итерационную процедуру, на каждом шаге которой выполняются следующие действия.

1. Из кэша извлекается точка $\mathbf{x}_c$ с минимальным значением целевой функции f_c. Это значение запоминается во вспомогательной переменной.
2. С помощью алгоритма, описанного в предыдущем разделе, выполняется проверка достижения минимума целевой функции. Если этот алгоритм возвращает true, то $\mathbf{x}_c$ является решением и происходит выход процедуры минимизации.
3. Вызывается блок изменения текущего уровня масштабирования *k*. Если $k = k_{\max}$ или если кэш содержит только 1 точку, уровень масштабирования остается неизменным. В противном случае с использованием формулы (9) вычисляется расстояние между двумя точками кэша с минимальными значениями целевой функции. Если вычисленное значение оказывается меньше $0.5 \cdot s^{-k}$, то значение *k* увеличивается на 1.
4. С помощью процедуры, которая была ранее описана в разделе «Алгоритм оценки частных производных», проводится оценка частных производных целевой функции в точке $\mathbf{x}_c$ и формируются вектор градиента $\mathbf{g}$ и матрицы вторых частных производных $\mathbf{G}$.
5. Регуляризация матрицы вторых частных производных $\mathbf{G}$. Если матрица $\mathbf{G}$ не является положительно определенной, с помощью модифицированного разложения Халецкого (см. [3]) формируется положительно определенная матрица $\mathbf{G}_p$, которая отличается от исходной матрицы $\mathbf{G}$ только значениями диагональных элементов. В этом случае переменная $z = \text{false}$. Если $\mathbf{G}$ является положительно определенной, то $\mathbf{G} = \mathbf{G}_p$ и $z = \text{true}$.
6. Происходит вычисление следующих вспомогательных величин:
   $l_{\max} = s^{-k+1}$ – максимально допустимая длина шага по аргументу,
   $\Delta\mathbf{x}_N = \mathbf{G}_p^{-1}\mathbf{g}$ – шаг по аргументу с использованием метода Ньютона,
   $l_N = \|\Delta\mathbf{x}_N\|$ – его длина (с использованием формулы (9))
   $\Delta\mathbf{x}_G$ - шаг по аргументу с использованием метода скорейшего спуска,
   $l_G = \|\Delta\mathbf{x}_G\|$ – его длина (с использованием формулы (9)).
7. Если $k > 0$ и $l_N > l_{\max}$ и $l_G > l_{\max}$, то уровень масштабирования *k* уменьшается на 1, после чего повторяются действия, выполняемые на шагах 4-6 (т.е. вновь вычисляются $\mathbf{g}$, $\mathbf{G}$, $\mathbf{G}_p$, z, $l_{\max}$, $\Delta\mathbf{x}_N$, l_N, $\Delta\mathbf{x}_G$, l_G).
8. Если $t = 0$, то вызывается блок выбора метода оптимизации. В результате работы этого блока переменная *t* получает значение 1 или 2. *t* полагается

равным 1, если $l_N > s \cdot l_{\max}$ или если $\{l_N > l_{\max}$ и $z = \text{false}\}$. Во всех остальных случаях $t = 2$.

9. Если $t = 2$, то выполняется шаг по методу Ньютона. Если $l_N < l_{\max}$, то $\mathbf{x} = \mathbf{x}_c + \Delta\mathbf{x}_N$. В противном случае $\mathbf{x} = \mathbf{x}_c + \frac{l_N}{l_{\max}}\Delta\mathbf{x}_N$. Далее вычисляется значение целевой функции в точке $\mathbf{x}$. Шаг считается успешным, если в результате действий, выполненных на шагах 2-9, в кэше появляется точка с функционалом, меньшим, чем f_c. В этом случае полагается $t = 0$, иначе $t = 1$. Выполняется переход к следующей итерации (т.е. к шагу 1).
10. Если $t = 1$, то выполняется шаг с использованием метода скорейшего спуска. Если $l_G < l_{\max}$, то $\mathbf{x} = \mathbf{x}_c + \Delta\mathbf{x}_G$. В противном случае $\mathbf{x} = \mathbf{x}_c + \frac{l_G}{l_{\max}}\Delta\mathbf{x}_G$. Далее вычисляется значение целевой функции в точке $\mathbf{x}$. Шаг считается успешным, если в результате действий, выполненных на шагах 2-10, в кэше появляется точка с функционалом, меньшим, чем f_c. В этом случае полагается $t = 0$ и выполняется переход к следующей итерации (т.е. к шагу 1). В противном случае переход к шагу 11.
11. Принудительное увеличение уровня масштабирования *k*. Если $k < k_{\max}$, то *k* увеличивается на 1. Создается регулярная сетка в окрестности $\mathbf{x}_c$ с шагом $s^{-k}\delta\mathbf{x}_0$, и производится вычисление значения функционала в точках этой сетки. Выполняется присвоение $t = 0$ и переход к следующей итерации.

Получение очередного приближения для множителей Лагранжа

Как уже отмечалось выше, в работе [2] для получения следующего приближения множителей Лагранжа λ_k предлагается использовать формулу (6). Однако, поскольку значения аргумента $\mathbf{x}$ меняются с дискретностью, кратной $s^{-k_{\max}}\delta\mathbf{x}_0$, прямое использование такой формулы может приводить к ситуациям, когда при изменении значений λ_k точка минимума функционала Ф (см. (5)) остается на прежнем месте. Для того чтобы избежать таких ситуаций, использовался более сложный алгоритм, который будет описан ниже.

В этом алгоритме предполагается наличие кэша записей, каждая из которых содержит координаты аргумента $\mathbf{x}$, значение целевой функции $f(\mathbf{x})$ и ограничений $g_j(\mathbf{x})$. При известных значениях предыдущего приближения для аргумента $\mathbf{x}_k$ и множителей Лагранжа λ_k значение функционала Ф для каждой записи кэша может быть пересчитано по формуле (5).

На первом шаге этого алгоритма, после того как в результате безусловной минимизации функционала Ф при известных значениях $\mathbf{x}_k$ и λ_k определено

новое приближение для аргумента $\mathbf{x}_{k+1}$, значение λ_{k+1} вычисляется по формуле (6). Далее при известных значениях $\mathbf{x}_{k+1}$ и λ_{k+1} с использованием (5) проводится вычисление функционала Φ для всех записей кэша и определяется такая запись с координатами $\mathbf{x}_m$, для которой $\Phi(\mathbf{x}_m, \lambda_{k+1})$ минимально. Если в результате этой операции $\mathbf{x}_m$ не совпадает с $\mathbf{x}_{k+1}$, то происходит переход к следующей итерации, которая подразумевает минимизацию Φ при $\mathbf{x}_{k+1}$ и λ_{k+1}. Точка $\mathbf{x}_m$ используется в этой процедуре в качестве начального приближения.

Если же $\mathbf{x}_m$ совпадает с $\mathbf{x}_{k+1}$, то на втором шаге алгоритма в окрестности $\mathbf{x}_{k+1}$ строится регулярная сетка с $p = 2$ и шагом $s^{-k_{\max}} \delta \mathbf{x}_0$. Отметим, что в общем случае при построении такой сетки записи кэша могут быть обновлены. Если в результате выполнения этой операции удается найти точку, для которой $\Phi(\mathbf{x}_m, \lambda_{k+1}) < \Phi(\mathbf{x}_{k+1}, \lambda_{k+1})$, то, как и на первом шаге, происходит переход к следующей итерации со значениями $\mathbf{x}_{k+1}$ и λ_{k+1} для Φ и $\mathbf{x}_m$ в качестве начального приближения.

Если первые два шага обсуждаемого алгоритма не приводят к успеху, то используется алгоритм, основанный на переборе точек регулярной сетки, построенной на предыдущем шаге. В этом алгоритме значения множителей Лагранжа задаются в форме

$$\lambda_q = \left(\lambda_k + qAg\left(\mathbf{x}_{k+1}\right)\right)^+, \tag{20}$$

где q – искомое положительное число. Пусть $\mathbf{x}_r$ – очередная точка регулярной сетки. Найдем такое минимальное значение q_r, при котором выполняется условие

$$\Delta\Phi(q) = \Phi(\mathbf{x}_r, \lambda_q) - \Phi(\mathbf{x}_{k+1}, \lambda_q) < 0. \tag{21}$$

Опуская промежуточные выкладки, можно показать, что функция $\Delta\Phi(q)$ является непрерывной и состоит из участков, на которых $\Delta\Phi$ зависит от q линейно или по квадратичному закону. Точки стыковки этих участков определяются уравнениями:

$$q_j = -\frac{\lambda_{kj}}{Ag_j\left(\mathbf{x}_{k+1}\right)}, \tag{22}$$

$$q_{n+j} = -\frac{\lambda_{kj} + Ag_j\left(\mathbf{x}_k\right)}{Ag_j\left(\mathbf{x}_{k+1}\right)}. \tag{23}$$

Возможны два случая. Среди точек регулярной сетки существует хотя бы одна, для которой решение уравнения $\Delta\Phi(q)=0$ является положительным. В этом случае среди таких точек выбирается минимальное значение $q_{\min}$, и тогда для следующего приближения множителей Лагранжа используется формула

$$\lambda_{k+1}=\left(\lambda_k+\lceil q_{\min}\rceil Ag(x_{k+1})\right)^{+}. \qquad (24)$$

Если таких точек нет (т.е. для всех точек регулярной сетки решения уравнения $\Delta\Phi(q)=0$ получаются только при отрицательных q), то анализируются значения ограничений $g_j(\mathbf{x}_{k+1})$. Если все эти ограничения отрицательны, то точка $\mathbf{x}_{k+1}$ является решением поставленной задачи. В противном случае продолжение поиска с использованием рассматриваемого метода невозможно.

Ограничение на значение целевой функции

Чтобы избежать возможной расходимости итерационного процесса при выполнении безусловной минимизации, на практике часто прибегают к следующему приему. Если выполнение очередной итерации приводит к увеличению функционала, то такая итерация считается неудачной. В этом случае используются ограничение длины шага или переход на иной способ получения следующего приближения.

Опасность возникновения расходящегося итерационного процесса может существовать и при условной минимизации. Однако в этом случае критерии расходимости являются более сложными, поскольку включают в себя не только целевую функцию f, но и значения функций ограничений g_j.

Введем следующие определения. Будем называть *точкой возможного решения* такую точку $\mathbf{x}$, для которой все $g_j(\mathbf{x})\le 0$. Будем называть *точкой локального минимума* такое возможное решение, в окрестности которого нет других точек, для которых удовлетворяются ограничения $g_j(\mathbf{x})\le 0$ при меньшем значении целевой функции $f(\mathbf{x})$. Под окрестностью здесь понимается совокупность точек регулярной сетки с $p=n$ и минимально допустимым шагом по аргументу.

Обозначим с помощью $f_{\min}$ минимальное значение целевой функции среди всех известных точек возможного решения, которые были получены ранее. Введем дополнительное ограничение следующего вида:

$g_{m+1}(\mathbf{x})=-1$, если не было получено ни одного возможного решения,

$g_{m+1}(\mathbf{x})=-\sqrt{f_{\min}-f(\mathbf{x})}$, если $f_{\min}>f(\mathbf{x})$,

$g_{m+1}(\mathbf{x})=\sqrt{f(\mathbf{x})-f_{\min}}$, если $f_{\min}\le f(\mathbf{x})$.

Очевидно, что при таком дополнительном ограничении в каждой очередной найденной точке возможного решения значение целевой функции будет меньше или равно значению целевой функции в предыдущей.

Парирование колебаний

Другой опасностью, которая может возникнуть при выполнении условной минимизации, является возникновение колебаний, при которых в ходе итерационного процесса происходит циклический обход повторяющихся точек в пространстве аргументов. Для предотвращения таких ситуаций мы использовали эмпирический алгоритм, который включает в себя две части и использует вспомогательный массив.

Всякий раз по окончании безусловной минимизации функционала Φ проводится проверка: не является ли полученная точка $\mathbf{x}_k$ точкой возможного решения. Если хотя бы для одного j не выполняется условие $g_j(\mathbf{x}_k) \le 0$, то такая точка помещается в конец вспомогательного массива. В этом массиве сохраняются не только координаты $\mathbf{x}_k$, но и значения λ_k и $g_j(\mathbf{x}_k)$.

Задача первой части алгоритма парирования колебаний заключается в установлении факта появления возможного зацикливания. Эта часть возвращает положительный ответ, если во вспомогательном массиве уже есть точка, координаты которой совпадают с $\mathbf{x}_k$.

Вторая часть алгоритма вызывается только в том случае, когда факт возможного зацикливания был установлен. Задачей второй части алгоритма является получение очередных приближений для $\mathbf{x}_{k+1}$ и λ_{k+1} на основании следующих действий.

1. Среди точек вспомогательного массива находится точка с минимальным значением p, для которой $\|\mathbf{x}_k - \mathbf{x}_{k-p}\| = 0$. Такая точка заведомо существует, если первая часть алгоритма дает положительный результат.
2. Среди точек вспомогательного массива находится точка с таким индексом $0 < q < p$, для которой $\|\mathbf{x}_k - \mathbf{x}_{k-q}\|$ максимально.
3. Формирование следующего приближения для $\mathbf{x}_{k+1}$. Алгоритм зависит от наличия в кэше точек возможного решения. Если такие точки есть, то из них выбирается такая точка $\mathbf{x}_c$, для которой $f(\mathbf{x}_c)$ минимально. В этом случае $\mathbf{x}_{k+1} = \dfrac{\mathbf{x}_k + \mathbf{x}_{k-q} + \mathbf{x}_c}{3}$. В противном случае $\mathbf{x}_{k+1} = \dfrac{\mathbf{x}_k + \mathbf{x}_{k-q}}{2}$. В полученной точке вычисляются значения ограничений $g_j(\mathbf{x}_{k+1})$.
4. Вычисление нового приближения для λ_{k+1} по формуле (6), т.е $\lambda_{k+1} = \left(\lambda_k + A\mathbf{g}(\mathbf{x}_{k+1})\right)^+$.

5. Покомпонентная коррекция множителей Лагранжа по следующим правилам. $\lambda_{k+1,j} = 2\lambda_{k+1,j}$, если $g_j(\mathbf{x}_k) > 0$ и $g_j(\mathbf{x}_{k+q}) > 0$. В противном случае $\lambda_{k+1,j} = 0$, если $g_j(\mathbf{x}_k) > 0$ или $g_j(\mathbf{x}_{k+q}) > 0$. Если $g_j(\mathbf{x}_k) \le 0$ и $g_j(\mathbf{x}_{k+q}) \le 0$, то коррекция не выполняется.

Алгоритм быстрого поиска в окрестности возможного решения

В случае, когда в результате минимизации функционала Φ была получена точка возможного решения $\mathbf{x}_k$, для которой все $g_j(\mathbf{x}_k) \le 0$, применяется простой алгоритм быстрого поиска в окрестности полученного приближения. Как показала практика, использование этого алгоритма позволяет несколько ускорить поиск решения.

Алгоритм основан на опросе значений $f(\mathbf{x})$ и $g_j(\mathbf{x})$ в точках регулярной сетки с $p = n$ в окрестности $\mathbf{x}_k$. В алгоритме используется вспомогательный массив данных, который идентичен вспомогательному массиву, описанному ранее в разделе «Алгоритм определения точки минимума». Этот массив определяет порядок опроса точек сетки.

Алгоритм представляет собой итерационную процедуру, с ограниченным числом итераций. Если на очередной итерации в результате опроса находится такая точка сетки $\mathbf{x}_c$, для которой $f(\mathbf{x}_c) < f(\mathbf{x}_k)$ и $g_j(\mathbf{x}_c) \le 0$ для любого j, то полагается $\mathbf{x}_k = \mathbf{x}_c$ и происходит переход к следующей итерации. Если такой точки найти не удается, то алгоритм заканчивает свою работу.

Вычислительная схема условной минимизации

Приведем полное формальное описание предлагаемого алгоритма условной минимизации. Здесь предполагается, что процедуры вычисления целевой функции $f(\mathbf{x})$ и ограничений $g_j(\mathbf{x})$ известны, а для пространства аргументов были заданы вектор начального шага $\delta\mathbf{x}_0$, фактор масштабирования s и его максимальный уровень $k_{\max}$ (см. раздел «Пространство аргументов»).

На вход в алгоритм передается вектор начального приближения $\mathbf{x}_0$, предельно допустимое число итераций $N_{\max}$ и требуемый режим останова поиска (см. ниже). В случае успешного завершения алгоритм возвращает точку найденного решения $\mathbf{x}_s$. Существуют две ситуации, в которых выполнение алгоритма может завершиться без нахождения искомой точки: исчерпание допустимого числа итераций и останов из-за невозможности получения очередного приближения для множителей Лагранжа (см. раздел «Получение очередного приближения для множителей Лагранжа»).

В алгоритме предусмотрено 3 режима останова итераций, которые далее будем обозначать числами от 1 до 3:

1. Остановка поиска, если найдено хотя бы одно возможное решение (т.е. такая точка, для которой $g_j(\mathbf{x}_s) \le 0$ для любого j).
2. Остановка поиска, если найдена точка локального минимума.
3. Режим полного поиска. Итерации останавливаются, если найдена точка локального минимума и попытка использования ее в качестве начального приближения не приводит к улучшению решения.

В обсуждаемом алгоритме используются 3 вспомогательных массива данных, которые заполняются динамически по ходу работы алгоритма (в начальный момент все эти массивы пусты).

- Кэш ранее вычисленных значений $\mathbf{x}, f(\mathbf{x}), g_j(\mathbf{x})$. Обновление этого кэша происходит всякий раз, когда требуется вычисление значений целевой функции и ограничений.
- Массив решений, получаемых после безусловной минимизации функционала Φ. Этот массив используется для парирования возможных колебаний, и его емкость ограничена $N_{\max}$ записями (см. выше раздел «Парирование колебаний»).
- Массив локальных минимумов из 3 записей, в который заносятся получаемые в процессе решения точки локальных минимумов.

Рассматриваемый алгоритм представляет собой итерационную процедуру, на каждой *k*-ой итерации которой уточняются следующие приближения для вектора аргумента $\mathbf{x}_k$ и множителей Лагранжа λ_k. Перед началом первой итерации выполняются следующие начальные действия.

1. В качестве начального приближения все множители Лагранжа полагаются равными 0.
2. Вычисляются значения целевой функции $f(\mathbf{x}_0)$ и ограничений $g_j(\mathbf{x}_0)$ в точке начального приближения. Результат вычислений запоминается в кэше.
3. Если для любого j выполняются условия $g_j(\mathbf{x}_0) \le 0$, то полагается $f_{\min} = f(\mathbf{x}_0)$. Если при этом был установлен режим возврата 1, то выполняется присвоение $\mathbf{x}_s = \mathbf{x}_0$ и завершение работы алгоритма.

Дальнейшие действия выполняются в цикле, который повторяется не более $N_{\max}$ раз. Внутри цикла выполняются следующие действия.

1. С помощью алгоритма безусловной минимизации, который был ранее рассмотрен в разделе «Алгоритм безусловной оптимизации», выполняется безусловная минимизация функционала Φ при фиксированных значениях $\mathbf{x}_k$ и λ_k. Результатом этой операции является вектор $\mathbf{x}_{k+1}$.
2. Если кэш содержит хотя бы одну точку возможного решения, то выполняются следующие действия.

2.1. В кэше находится точка возможного решения с минимальным значением целевой функции.

2.2. Если был установлен режим 1, то эта точка принимается в качестве найденного решения $\mathbf{x}_s$ и работа алгоритма завершается.

2.3. В противном случае происходит обновление значения $f_{\min}$ и пересчет значений $g_{m+1}(\mathbf{x})$ в записях кэша (см. раздел «Ограничение на значение целевой функции»).

3. Если полученная точка $\mathbf{x}_{k+1}$ не является точкой возможного решения, а в кэше такие точки есть, то выполняются следующие действия.

3.1. Среди точек кэша выбирается точка возможного решения $\mathbf{x}_m$ с минимальным значением целевой функции.

3.2. Если $\|\mathbf{x}_{k+1}-\mathbf{x}_m\|<0.5s^{-k_{\max}+1}$ или если $\|\mathbf{x}_{k+1}-\mathbf{x}_m\|<s^{-k_{\max}+1}$ и $\|\mathbf{x}_{k+1}-\mathbf{x}_k\|<s^{-k_{\max}+1}$, то $\mathbf{x}_{k+1}=\mathbf{x}_m$.

4. Если полученная точка $\mathbf{x}_{k+1}$ не является точкой возможного решения, она заносится во вспомогательный массив решений, после чего вызывается алгоритм, описанный выше в разделе «Парирование колебаний». Если первая часть этого алгоритма возвращает положительный ответ, то новые приближения $\mathbf{x}_{k+1}$ и λ_{k+1} вычисляются с помощью второй части алгоритма и происходит переход к следующей итерации.

5. Если полученная точка $\mathbf{x}_{k+1}$ является точкой возможного решения, вызывается алгоритм быстрого поиска в ее окрестности (см. выше соответствующий раздел). В результате выполнения этого алгоритма значение $\mathbf{x}_{k+1}$ может быть изменено.

6. Если полученная точка $\mathbf{x}_{k+1}$ является точкой локального минимума, то выполняются следующие действия.

6.1. Если был установлен режим 2 (останов при достижении локального минимума), то точка $\mathbf{x}_{k+1}$ считается решением и работа алгоритма завершается.

6.2. Если массив локальных минимумов не пуст и точка $\mathbf{x}_{k+1}$ уже присутствует в этом массиве, то она считается решением и работа алгоритма завершается.

6.3. Точка $\mathbf{x}_{k+1}$ заносится в массив локальных минимумов, все множители Лагранжа λ_{k+1} полагаются равными 0, и происходит переход к следующей итерации.

7. Вычисляется следующее приближение для λ_{k+1} по формуле (6).

8. Происходит пересчет значения функционала Φ для всех записей кэша при значениях $\mathbf{x}_{k+1}$, λ_{k+1} и $f_{\min}$, полученных в ходе текущей итерации.

9. Среди записей кэша находится точка $\mathbf{x}_m$, для которой Φ минимально. Если $\|\mathbf{x}_{k+1}-\mathbf{x}_m\|$ отлично от 0, то происходит переход к следующей итерации.

10. Вызывается алгоритм получения следующего приближения для λ_{k+1}, описанный ранее в разделе «Получение очередного приближения для множителей Лагранжа».

Тестирование алгоритма условной минимизации

Ранее в формулах (3) и (5) были введены две положительные константы: A и α, которые определяют вид модифицированной функции Лагранжа M и функционала оптимизации Φ. В работе [2] отсутствуют какие-либо рекомендации по выбору этих констант.

В качестве тестовой задачи для проверки предлагаемого алгоритма мы использовали задачу проектной баллистики, в которой требуется вычислить оптимальный импульс скорости при возврате космического аппарата с орбиты искусственного спутника Луны с последующей посадкой в заданном районе земной поверхности (см. [1]). В этой задаче в качестве аргумента оптимизации использовались 3 компонента вектора асимптотической скорости аппарата относительно Луны. На правом конце траектории возврата накладывались ограничения на высоту условного перигея и предельно допустимые отклонения центра полигона посадки от плоскости траектории возврата. Расчет траектории возврата от лунной орбиты до момента входа в атмосферу Земли выполнялся с помощью численного интегрирования уравнений движения.

Таблица 1

Скорость сходимости задачи условной минимизации

	$\alpha=10^{-2}$	$\alpha=10^{-3}$	$\alpha=10^{-4}$	$\alpha=5\cdot10^{-4}$
$A=10^4$	1358	402	333	404
$A=10^3$	2609	474	311	341
$A=10^2$	3146	579	437	551
$A=10$	2993	1384	582	1012
$A=2$	2477	1937	2640	1336

В таблице 1 приведено число обращений к процедуре численного интегрирования траектории возврата, которое требуется для получения решения при одинаковом начальном приближении в зависимости от значений констант A и α. Здесь в качестве начальных условий рассматривалась круговая орбита искусственного спутника Луны с наклонением 10 градусов, долготой восходящего узла 90 градусов и высотой 100 км над поверхностью Луны. Склонение Луны принималось равным –28 градусов. Время исполнения импульса для перехода с лунной орбиты на траекторию возврата к Земле

выбиралось в течение 1 витка таким образом, чтобы обеспечить требуемую асимптотическую скорость с минимальными затратами топлива.

Как видно из приведенных результатов, алгоритм обеспечивает сходимость в широком диапазоне значений A и α, даже когда их значения меняются на несколько порядков. Однако при этом скорость сходимости также может меняться в весьма широких пределах. Далее мы ограничимся лишь самыми общими рекомендациями относительно выбора этих констант.

Константа α де-факто служит ограничителем на область поиска в пространстве аргументов при получении следующего приближения для $\mathbf{x}_{k+1}$. Так, при $\alpha = 0$ согласно (5) точкой минимума функционала Ф всегда будет $\mathbf{x}_{k+1} = \mathbf{x}_k$. Поэтому рекомендуется использовать меньшие значения α при наличии хорошего начального приближения и большие – при его отсутствии.

Константа A играет роль весового коэффициента, устанавливающего относительный вклад ограничений в функционал Ф по сравнению с целевой функцией (чем больше значение A, тем этот вклад сильнее). Как правило, при больших значениях A алгоритм быстрее находит точки возможного решения. Однако при неоправданном увеличении этой константы существует опасность остановки процесса поиска решения на промежуточных локальных минимумах.

Отсутствие формальных критериев обоснованного выбора констант A и α при решении конкретной задачи с одной стороны является недостатком предлагаемого метода. С другой стороны при отсутствии хорошего начального приближения метод позволяет строить гибкие приложения, когда искомое решение может быть найдено в результате нескольких последовательных обращений к алгоритму условной минимизации с разными настройками точности и при разном значении обсуждаемых констант.

Выводы

1. Разработан метод условной минимизации, учитывающий специфику практических задач проектной и оперативной баллистики.
2. Работоспособность разработанного метода была проверена на задаче возврата с орбиты искусственного спутника Луны с посадкой в заданном районе земной поверхности.
3. Недостатком разработанного метода является отсутствие формальных критериев обоснованного выбора констант A и α, которые существенно влияют на скорость сходимости.

Библиографический список

[1] С. Н. Евдокимов, С. И. Климанов, А. Н. Корчагин, Е. А. Микрин, Ю. Г. Сихарулидзе, А. Г. Тучин, Д. А. Тучин. Обеспечение посадки спускаемого аппарата на космодром “Восточный” после возвращения от Луны. // Известия РАН. Теория и системы управления, 2014, № 6, с. 136–152.

[2] Ф. П. Васильев. Методы оптимизации. // М., Факториал Пресс, — 2002.
[3] P. E. Gill, W. Murray, and M. H.Wright. Practical Optimization. // Academic Press, London, — 1981.
[4] P. E. Gill and W. Murray. Numerical Methods of Constrained Optimization. // Academic Press, London, — 1974.

Оглавление