2.5. НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ (НП – ПРОГРАММИРОВАНИЕ)

Постановка НП-задачи формулируется как нахождения оптимума целевой функции *f(X)* при ограничениях и задаётся моделью вида

$$\begin{cases} f(x_1,x_2,...,x_n) \to \max, \\ g_1(x_1,x_2,...,x_n) \ge 0, \\ g_2(x_1,x_2,...,x_n) \ge 0, \\ ... \\ g_m(x_1,x_2,...,x_n) \ge 0. \end{cases} \qquad (2.67)$$

где функции $f(X)$ и $g_i(X)$, $i = 1, m$, в общем случае, нелинейные.

НП-задачи существенно отличаются от ЗЛП неформализованностью методов их решения. Нелинейность приводит к тому, что:

- область принятия решения может быть невыпуклая;
- область может иметь бесконечное число крайних точек.

Поэтому для решения НП-задач разработаны методы, которые ориентированы на классы задач в их конкретной постановке.

Общего подхода, являющегося универсальным во всех случаях, ***создать не удалось*** [12, 46, 67, 68].

2.5.1. Аналитические методы определения экстремумов

Указанные методы основываются на известных вам методах классического математического анализа, базируясь на ряд теорем [33].

Теорема 1 (о существовании экстремума). Если функция многих переменных $f(x_1, x_2, \ldots, x_n)$ непрерывна и определена на замкнутом множестве $\mathfrak{R}$, то она достигает на этом множестве, ***по крайней мере, один раз*** своего минимального и максимального значений.

Теорема 2 (о местоположении экстремума). Если $f(x_1, x_2, \ldots, x_n)$ является функцией нескольких переменных, определённой на допустимой области $\mathfrak{R}$, то экстремальное значение f (если оно существует) достигается в одной или нескольких точках, принадлежащих:

- множеству стационарных точек $S(X)$;
- множеству точек границы $G(X)$;

- множеству точек, в которых (где) функция $f(x_1, x_2, \ldots, x_n)$ не дифференцируема.

Множество точек $S(X)$ функции $f(x_1, x_2, \ldots, x_n)$ называется множеством стационарных точек, если его элементы удовлетворяю условию

$$\nabla f(X) = \left\{ \frac{\partial f(X)}{\partial x_j}, j = \overline{1,n} \right\} = 0 . \qquad (2.68)$$

Вектор $\nabla f(X)$ – называют градиентом функции.

Находящий в стационарной точке ***минимум*** или ***максимум*** функции, может быть как ***абсолютным***, так и ***относительным***.

Относительный максимум функции $f(X)$ достигается в точке X^0 с координатами $(x_1^0, x_2^0, \ldots, x_n^0)$, если для всех точек, лежащих в малой окрестности точки X^0, имеет место неравенство $f(X^0) \geq f(X^0 + H)$, где $H = \{h_1, h_2, \ldots, h_n\}$.

Относительный максимум называется ещё ***локальным*** максимумом.

Абсолютный максимум функции $f(X)$ достигается в точке X^* с координатами $(x_1^*, x_2^*, \ldots, x_n^*)$, если для всех точек, принадлежащих множеству ограничений $\mathfrak{R}$ справедливо неравенство $f(X^*) \geq f(X)$, где $X = \{x_1, x_2, \ldots, x_n\} \in \mathfrak{R}$.

Абсолютный максимум называется ещё ***глобальным*** максимумом.

Аналогично, с точностью до знака неравенства, формулируются определения абсолютного и относительного минимумов.

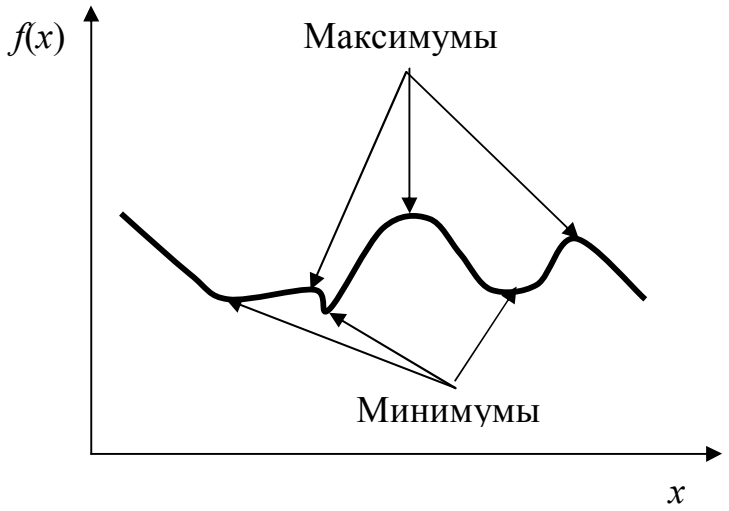

Рисунок 2.12 – Локальные и глобальные экстремумы

Характер экстремума, минимум или максимум, характеризуется выпуклостью или вогнутостью функции (рисунок 2.12).

Пусть $\mathfrak{R}$ – выпуклое множество точек n-мерного пространства.

Если для произвольного множителя $k \in [0, 1]$ и некоторого приращения ΔX выполняется неравенство

$$f(X + k\Delta X) \geq f(X) + k[f(X) - f(X + \Delta X)], \tag{2.69}$$

то функция называется ***вогнутой*** (обращена выпуклостью вверх, в отличие от математического анализа!!!), а если

$$f(X + k\Delta X) \leq f(X) + k[f(X) - f(X + \Delta X)], \tag{2.70}$$

то функция называется ***выпуклой***.

Если неравенства (2.69) и (2.70) строгие, говорят о **строгой вогнутости** и ***строгой выпуклости***. Для одномерного случая указанные неравенства интерпретируются графически, как это представлено на рисунке 2.13.

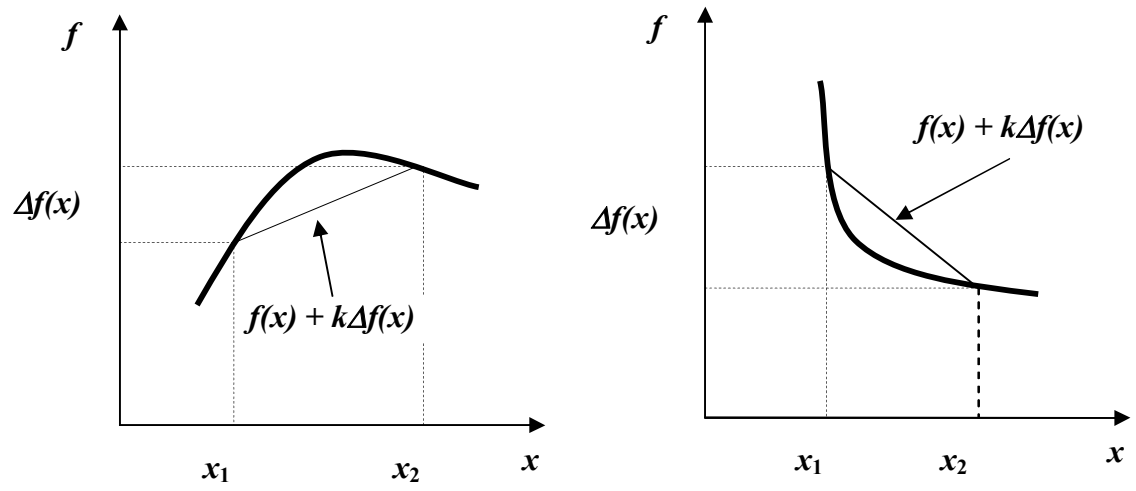

Рисунок 2.13 – Вогнутости и выпуклости функций

В многомерном случае непосредственное применение (2.69) или (2.70) является проблематичным. С этой целью применяется матрица Гёссе (или Гессе), элементы которой составляются из производных второго порядка и определяются так:

$$h_{i,j} = \left[\frac{\partial^2 f(X)}{\partial x_i \partial x_j} \right]_{x=x_0}, i = \overline{1,n}, j = \overline{1,m}. \tag{2.71}$$

Матрица Гёссе (обозначается как $\nabla^2 f(X)$ и $H(X)$) называется ***положительно определённой***, если её главные угловые миноры

положительны, и ***отрицательно определённой***, если её главные угловые миноры имеют знак $(-1)^k$, k – номер углового минора.

В качестве напоминания [18, 20, 35, 36]: *минором элемента $a_{i,j}$ матрицы A называется определитель, построенный из элементов этой матрицы, оставшихся после вычёркивания i-ой строки и j-ого столбца.*

Главные угловые миноры матрицы соответствуют элементам, расположенным вдоль главной (с Северо-запада на Юго-восток) её диагонали.

$$\mu_1 = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{pmatrix} = \begin{vmatrix} h_{2,2} & h_{23} \\ h_{3,2} & h_{3,3} \end{vmatrix} = h_{2,2} \cdot h_{3,3} - h_{3,2} \cdot h_{2,3}.$$

Положительная определённость матрицы Гёссе соответствует ***выпуклости*** функции, отрицательная определённость – ***вогнутости***. Случай, когда знаки чередуются не по порядку, соответствует ***перегибу***.

Теорема 3. Для того, чтобы в точке X_0 достигался внутренний относительный максимум, достаточно ***равенства нулю всех первых*** производных и ***строгой вогнутости*** функции в окрестностях X_0.

Теорема 4. Для того, чтобы в точке X_0 достигался внутренний относительный минимум, достаточно ***равенства нулю всех первых*** производных и ***строгой выпуклости*** функции в окрестностях X_0.

Пример. Исследовать на экстремум функцию без ограничений

$$f(x_1, x_2, x_3) = x_1 + 2x_3 + x_2 \cdot x_3 - x_1^2 - x_2^2 - x_3^2.$$

Градиент этой функции есть

$$\nabla f(x_1, x_2, x_3) = \{1 - 2x_1; x_3 - 2x_2; 2 + x_2 - 2x_3\}.$$

Используя условие стационарной точки (2.68), получим её координаты

$$X_0 = \left\{\frac{1}{2}; \frac{2}{3}; \frac{4}{3}\right\}.$$

Матрица Гёссе, построенная для рассматриваемого случая

$$\nabla^2 f(X) = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{pmatrix}.$$

Её угловые миноры суть

$$\mu_1 = \begin{vmatrix} -2 & 1 \\ 1 & -2 \end{vmatrix} = 3, \mu_2 = \begin{vmatrix} -2 & 0 \\ 0 & -2 \end{vmatrix} = 4, \mu_3 = \begin{vmatrix} -2 & 0 \\ 0 & -2 \end{vmatrix} = 4.$$

Они положительны, следовательно, матрица Гессе положительно определена, функция выпукла, и в точке X_0 с координатами $\left\{\frac{1}{2};\frac{2}{3};\frac{4}{3}\right\}$ достигается минимум.

2.5.2. Методы поиска экстремумов в задачах без ограничений или в случае ограничений с разделяющимися переменными

Ограничение $g_i(x_1, x_2, \ldots, x_n) = 0$ является функцией с ***разделяемыми переменными (сепарабельной)***, если его можно представить в виде

$$x_i = \varphi_i(\{x_j\}), \qquad i \neq j, \;\; j = \overline{1,n}.$$

Общий алгоритм решения

- Отыскивается множество всех стационарных точек S функции $f(X)$ внутри допустимого множества $\mathfrak{R}$ и выбираются координаты точки, в наибольшей степени отвечающие направлению оптимизации задачи.
- Рассматривается множество точек границы G. Для этого выполняют разделение переменных по каждому из ограничений, с последующей подстановкой выражений переменных в функцию цели $f(X)$. Полученные функции исследуются на экстремумы. Выбираются координаты интересующего нас оптимума.
- Определяется и подвергается исследованию множество точек, принадлежащих $\mathfrak{R}$, где функция не дифференцируема.
- Из результатов предыдущих шагов выбирается наилучшее решение.

Замечания.

1. Метод требует значительных вычислительных затрат и аналитических преобразований.

2. Не отвечает должной формализации для использования вычислительной техники.

3. Применение ограничивается задачами, область поиска решений которых описывается функциями с сепарабельными переменными.

Тем не менее, были разработаны многочисленные методы, направлены на решение задачи и ориентированные на применение ЭВМ [9, 29]. “Движение” текущей точки к оптимуму может происходить различными способами. В зависимости от этого, имеет место следующая классификация, методы подразделяются на:

- прямые методы или методы, использующие лишь значения функции;
- методы первого порядка, использующие, наряду со значениями функции значения первых частных производных;
- методы второго порядка, использующие дополнительно к значениям функции и первых частных производных и прочие частные вторые производные и выше.

Причём производные могут вычисляться как аналитически, так и численно.

2.5.2.1. Прямые методы поиска [9]

Одномерный поиск

При одномерном поиске функция *f(x)*, экстремум которой ищется, зависит от одной переменной. В дальнейшем будем предполагать, что ***решается задача минимизации***. Совершенно аналогичный алгоритм может быть использован и для случая поиска максимума. Отличия будут заключаться в знаках неравенств в проверках, производимых в процессе функционирования алгоритма.

Одномерный поиск является составной частью многих алгоритмов многомерного поиска в качестве вспомогательного. Алгоритмов одномерного поиска разработано изрядно, мы рассмотрим только самые интересные из них как с точки зрения скорости вычислений, так и их структуры.

Дихотомический поиск

Так же называется методом половинного деления. Это очень древний алгоритм, известный со времён раннего средневековья Ближнего Востока под названием “Поимка льва в пустыне”.

Есть пустыня, в ней лев. Делим пустыню пополам, лев окажется в одной из половинок. Часть пустыни, где находится лев, снова подвергаем делению… В конце концов, размеры пустыни окажутся чуть больше величины льва, и царя зверей остаётся только заключить в клетку.

Входные данные. Интервал поиска (сиречь, пустыня) [*a, b*]; допустимая конечная длина интервала, определяющая точность расчётов $l > 0$; константа различимости ε.

Условие окончания: $b - a \le l$.

Дадим пояснение алгоритма с использованием рисунков 2.14 и 2.15.

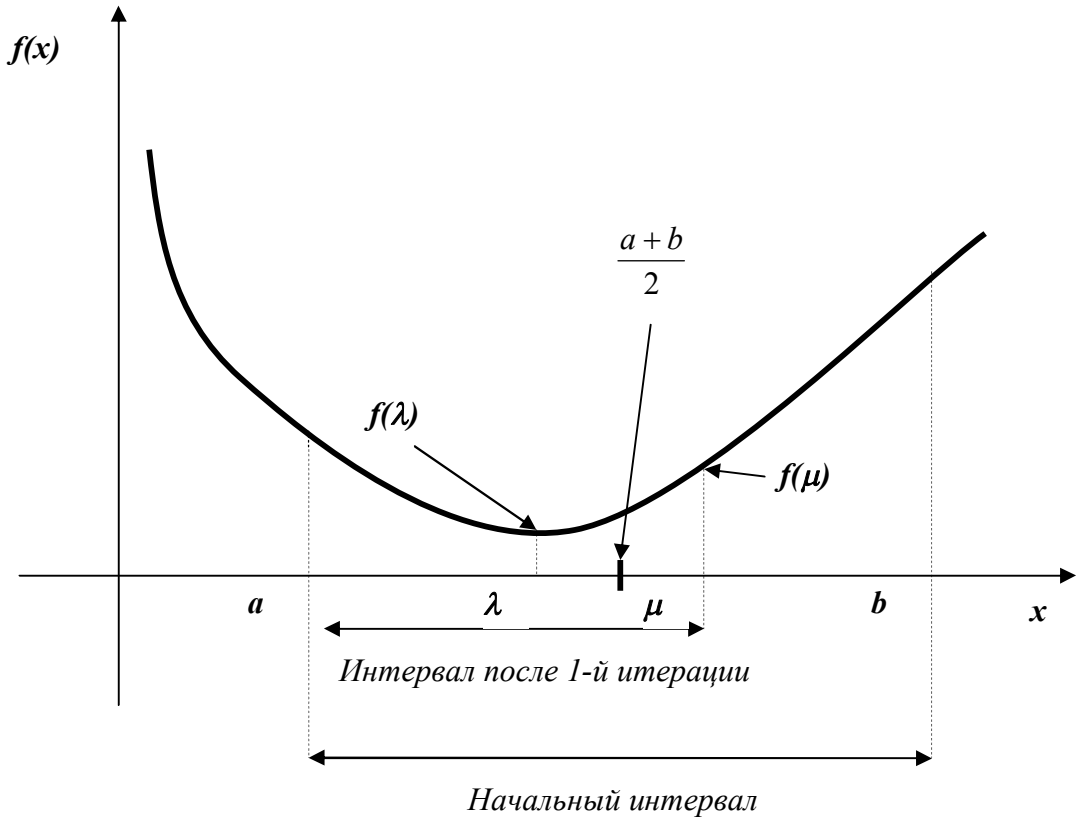

Рисунок 2.14 – Сужение интервала по мере дихотомии

Первоначально вычисляется пара величин: $\lambda = \frac{a+b}{2} - \varepsilon$ и $\mu = \frac{a+b}{2} + \varepsilon$, а тако же соответствующие им значения функций $f(\lambda)$ и $f(\mu)$. Полученные значения функций сравниваются между собой.

Пусть, например, $f(\lambda) \le f(\mu)$ (условие № 1 на схеме алгоритма, рисунок 2.14), выход «да». В этом случае сдвигается граница b, как это показано на рисунке 2.13. В противном случае, выход «нет», граница a сдвигается по направлению к середине интервала $\frac{a+b}{2}$.

Длительность вычислительного процесса контролируется условием № 2: вычисления заканчиваются при возникновении ситуации $b - a \le l$, которая означает, что границы интервала поиска сузились до заданного точностного размера l.

Имеется выражение, которое позволяет определить число итераций, необходимое для получения решения с заданной точностью:

$$b_k - a_k = \frac{(b-k)}{2^k} + 2 \cdot \varepsilon \left(1 - \frac{1}{2^k} \right),$$

где k – число итераций.

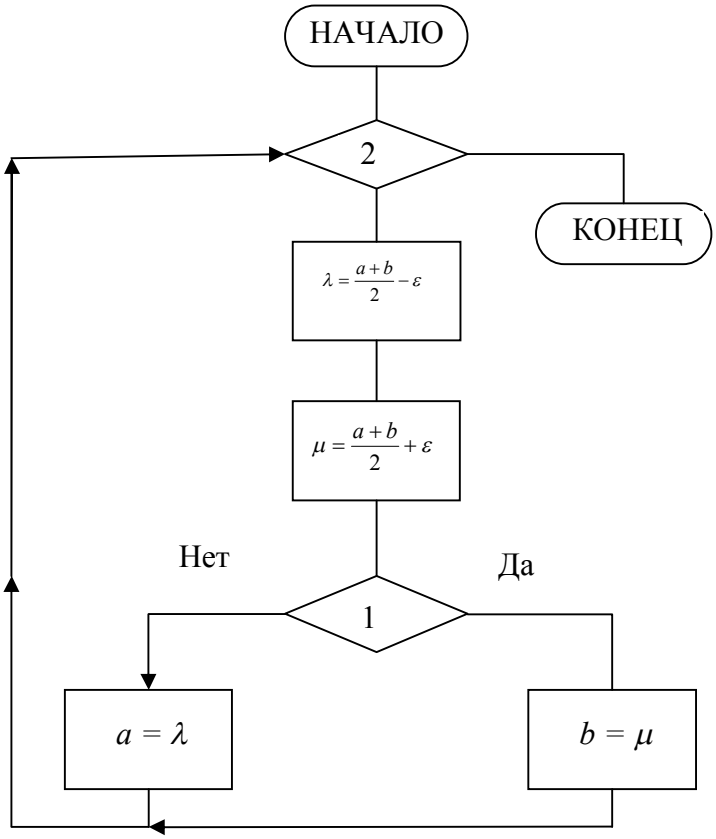

Рисунок 2.15 – Схема алгоритма дихотомического метода

Алгоритм, описанный выше, обладает самой меньшей скоростью сходимости, но, тем не менее, пользуется популярностью у программистов за простоту его реализации.

Метод золотого сечения

При функционировании метода выполняются условия:

- интервал поиска сужается равномерно;
- параметры точек сравнения и концов интервала соотносятся следующим образом

$$b - \lambda = \mu - a. \qquad (2.72)$$

Выполнение (2.72) достижимо, если значения λ и μ рассчитывать по формулам

$$\lambda = a + (1 - \alpha) \times (b - a) \text{ и} \quad (2.73)$$
$$\mu = a + \alpha \times (b - a), \quad (2.74)$$

где $|\alpha| < 1$.

Действительно, из (2.74) получается

$$\mu - a = \alpha \times (b - a). \quad (2.75)$$

Если λ, определяемое (2.73), подставить в левую часть (2.72), то

$$\boldsymbol{b - a} - (1 - \alpha) \times (\boldsymbol{b - a}) = \alpha \times (b - a). \quad (2.76)$$

Сопоставляя (2.75) и (2.76) видим, что условие (2.72) будет неизменно выполняться при определении λ и μ по формулам (2.73) и (2.74).

В ходе вычислений новые границы переставляются таким образом, чтобы либо $\lambda^{\nabla} = \mu$, либо $\mu^{\nabla} = \lambda$. Таким образом, пересчёт значений координат и функции в одной из этих точек не производится.

Коэффициент золотого сечения $\alpha = 0{,}618\ldots$ есть предел отношения соседних членов ряда Фибоначчи при их числе, стремящемся бесконечности.

Интервал поиска сужается равномерно, пропорционально α по закону

$$b_k - a_k = \alpha^k \ (b - a),$$

что позволяет оценить значение k – число итераций, необходимых до нахождения оптимума с заданной точностью.

Чертёж, поясняющий ход расчетов, и схема алгоритма представлены на рисунках 2.15 и 2.16.

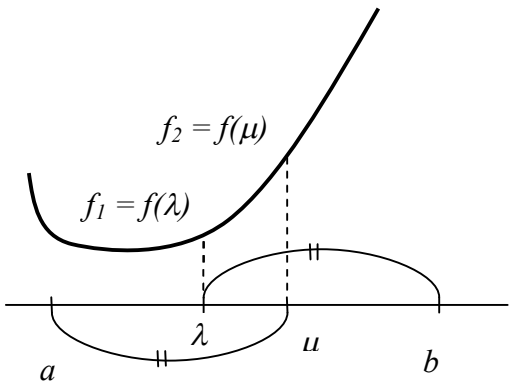

Рисунок 2.15 – Интервалы в методе золотого сечения

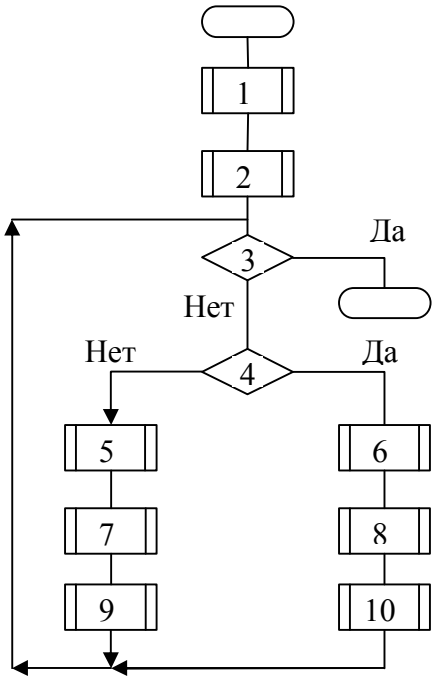

Рисунок 2.16 – Алгоритм метода золотого сечения

Входные данные. Интервал поиска [*a, b*]; точность расчётов $l > 0$.

Блоки № 1 рассчитывает параметры λ и μ по формулам (2.73) и (2.74), а блок № 2 определяет значения функции $f_1 = f(\lambda)$ и $f_2 = f(\mu)$, им соответствующие. Условие 3 $b - a \le l$ определяет остановку вычислительного процесса при достижении заданной погрешности. Условие 4 $f_1 \le f_2$ ответственно за сдвиги границ интервала поиска, аналогично дихотомическому методу. Блоки №№ 5, 7, 9 выполняют операции, связанные со сдвигом границы *a*.

- 5: $a = \lambda;\ \lambda = \mu$.
- 7: Расчёт μ по формуле (2.74)
- 9: Перестановка пересчёт значений функций $f_1 = f_2;\ f_2 = f(\mu)$

Блоки №№ 6, 8, 10 выполняют аналогичные операции, связанные со сдвигом границы *b*.

- 6: $b = \mu;\ \mu = \lambda$.
- 8: Расчёт λ по формуле (2.73)
- 10: Перестановка пересчёт значений функций $f_2 = f_1;\ f_1 = f(\lambda)$.

Метод Фибоначчи

В отличие от метода золотого сечения, указанный метод

- на каждой итерации изменяет коэффициент сжатия интервала и
- выполняется заранее известное числа шагов.

Основывается на использовании ряда Фибоначчи, элементы которого определяются рекуррентным соотношением

$$F_{n+1} = F_n + F_{n-1},\ F_0 = F_1 = 1, \tag{2.77}$$

определяющим, что каждый последующий член ряда образован суммой двух предыдущих:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 …

Входные данные. Интервал поиска [*a, b*], параметр, задающий точность расчётов $l > 0$, константа различимости $\varepsilon << l$.

Условие окончания: поиск заканчивается через определённое число шагов (итераций), определяемых по неравенству

$$\frac{b-a}{l} < F_n \Rightarrow n\,. \tag{2.78}$$

Координаты точек λ_k и μ_k, где k – номер итерации, рассчитываются по формулам

$$\lambda_k = a_k + \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k)\,, \tag{2.79}$$

$$\mu_k = a_k + \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k)\,. \tag{2.80}$$

Согласно формулам (2.79) и (2.80), от итерации к итерации, интервал поиска меняется по закону

$$b_{k+1} - a_{k+1} = \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k) = b_k - \lambda_k = \mu_k - a_k\,, \tag{2.61}$$

причём, перемещение границ интервала поиска и пересчёт $f(\lambda_k)$ и $f(\mu_k)$ происходит с соблюдением тех же принципов, что и в методе золотого сечения.

Константа различимости ε используется после окончания циклической части алгоритма для уточнения полученного результата. Схема алгоритма представлена рисунком 2.18.

Блок № 1 выполняет подготовительные операции: осуществляется расчёт ряда Фибоначчи (2.77), попутно определяется (2.78) – число членов ряда n, необходимых для расчёта; начальные значения

$$\lambda_0 = a + \frac{F_{n-2}}{F_n}(b-a), \quad \mu_0 = a + \frac{F_{n-1}}{F_n}(b-a), \quad f_1 = f(\lambda_0) \text{ и } f_2 = f(\mu_0).$$

Блок № 2 есть заголовок цикла изменения счётчика k от 1 до $n - 2$, тело цикла представляется блоками №№ 3 – 5.

Условие, управляющее сдвигом границ, $f_1 \le f_2$, представлено блоком № 3.

Сдвиг левой границы, выход «нет» блока № 3, выполняется в блоке № 4:

- $a = \lambda_k$; $\lambda_k = \mu_k$.
- Расчёт μ_k по формуле (2.80)
- Перестановка пересчёт значений функций $f_1 = f_2$; $f_2 = f(\mu_k)$.

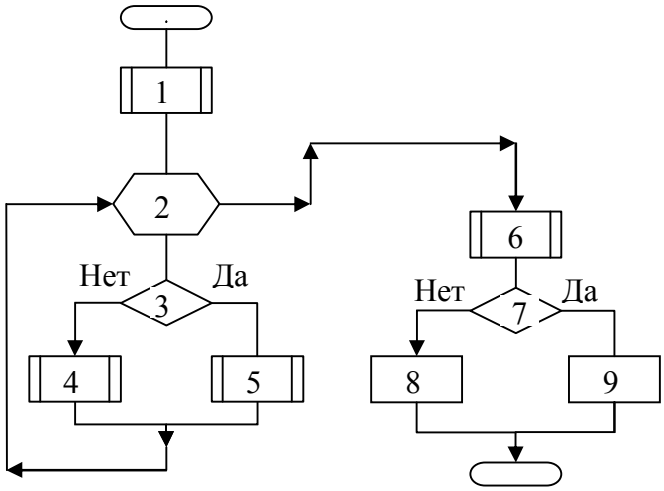

Рисунок 2.18– Алгоритм метода Фибоначчи

Сдвиг правой границы, выход «да» блока № 3, выполняется в блоке № 5:

- $b = \mu_k$; $\mu_k = \lambda_k$.
- Расчёт λ_k по формуле (2.79)
- Перестановка пересчёт значений функций $f_2 = f_1$; $f_1 = f(\lambda_k)$.

Блоки №№ 6, 7, 8 и 9 осуществляют уточнение результата методом половинного деления. Блок № 6 выполняет операции $\mu = \lambda + \varepsilon$ и $f_2 = f(\mu)$. Блок № 7 проверяет условие $f_1 \le f_2$ и, в зависимости от его выполнения, сдвигает левую границу, выход «нет» блок № 8, $a = \lambda$, либо правую, выход «да», блок № 9, $b = \mu$.

Многомерный поиск [8, 5]

При многомерном поиске функция *f(X)*, экстремум которой ищется, зависит от нескольких переменных. Алгоритмы многомерного поиска эвристические. Хотя некоторые из них носят имена своих первооткрывателей, не факт, что они не переизобретаются заново многочисленными последователями, хотя и более невежественными в плане оптимизации, но, тем не менее, более подкованными в плане алгоритмизации.

Метод конфигураций

Метод конфигураций называется тако же методом траекторий или, по имени авторов, методом Хука и Дживса.

Идея метода состоит в следующем.

- Направления поиска ориентированы, так сказать, по сторонам света или вдоль осей координат в обе стороны.
- Вокруг текущей точки производится поиск направления, в котором функция убывает.
- Если такое направление найдено, то шаг поиска увеличивается, а поиск продолжается в выбранном направлении – устанавливается так называемый тренд поиска, и осуществляется до тех пор, пока функция в этом направлении убывает.
- Если факта убывания функции не обнаружено, то шаг поиска уменьшается.

Таким образом, делается попытка найти «овраг» (при минимизации) функции и двигаться вдоль него к точке минимума.

В задачи максимизации ищется направление возрастания функции и «хребет».

Алгоритм метода показан на рисунке 2.19.

Входные данные. Начальное значение шага λ, координаты начальной точки X_0 ускоряющий множитель α, точность нахождения решения ε, список возможных направлений поиска $S = \{s_1, s_2, \ldots, s_n\}$. Для числа переменных равного двум, это орты: $s_1 = (0, 1)$ и $s_2 = (1,0)$.

Условие окончания: значение текущего шага укладывается в точность, $\varepsilon \geq \lambda$.

Блок № 1. Координаты начальной точки переприсваивается текущей переменной: $Y = X_0$.

Блок № 2. Организуется цикл перебора направлений поиска $j = 1, n$, тело цикла составляют блоки №№ 3 – 7.

Сравнение в блоке № 3 проверяется факт убывания функции в выбранном направлении: $f(Y + \lambda \times s_j) < f(Y)$. Если функция убывает, выход «да» блока № 3, то текущая точка перемещается в выбранном направлении $Y = Y + \lambda \times s_j$ (блок № 4).

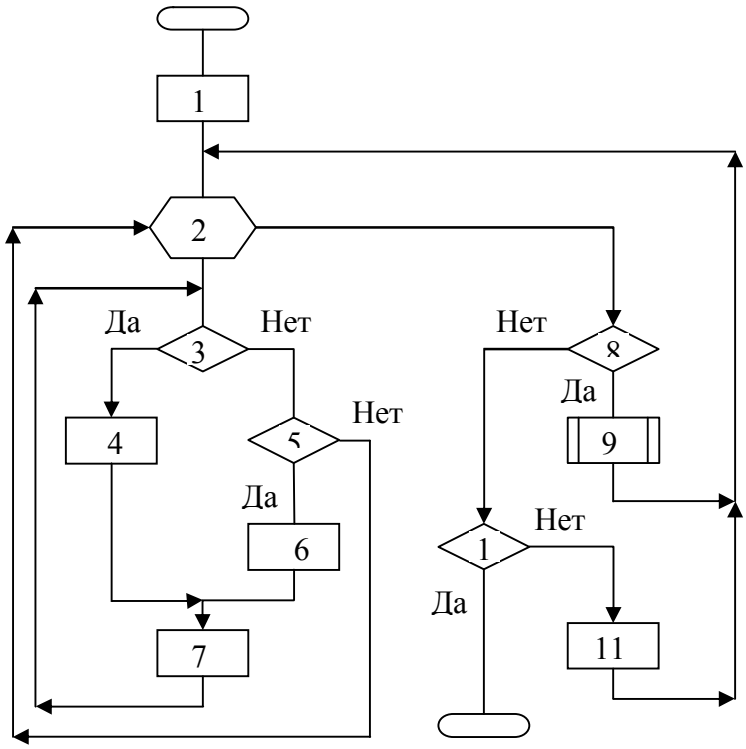

Рисунок 2.19 – Алгоритм метода Хука-Дживса

В противном случае, выход «нет» блока № 3, в блоке № 5 проверяется факт убывания функции в направлении, противоположном выбранному: $f(Y - \lambda \times s_j) < f(Y)$.

Если функция не убывает ни в выбранном, ни в противоположном направлениях (выход «нет», блок № 5), то выбирается очередное направление.

Если сравнение в блоке № 5 произошло удачно (выход «да»), то текущая точка решения перемещается в блоке № 6 по закону $Y = Y - \lambda \times s_j$.

Блок № 7 выполняет факультативное действие $\lambda = 2 \times \lambda$, которое увеличивает шаг, ускоряя тем самым поиск. В ряде реализаций алгоритма данное действие отсутствует.

Блок № 8 достигается в ходе работы алгоритма, после перебора всех направлений поиска и попадания в ситуацию, когда при заданном значении λ в окрестностях точки X_0 убывания функции не обнаружено.

Блок № 9 выполняет пересчёт координат текущей точки по формуле

$$Y = X_0 + \alpha \times (Y - X_0); \; X_0 = Y.$$

Блок № 10 осуществляет проверку окончания: $\varepsilon \geq \lambda$? Если «да» то алгоритм заканчивает работу, если «нет», то шаг поиска уменьшается

$$\lambda = \frac{\lambda}{2},$$

после чего начинается поиск вокруг текущей точки X_0 с уменьшенным шагом.

Пример траектории поиска представлен ниже, на рисунке 2.20.

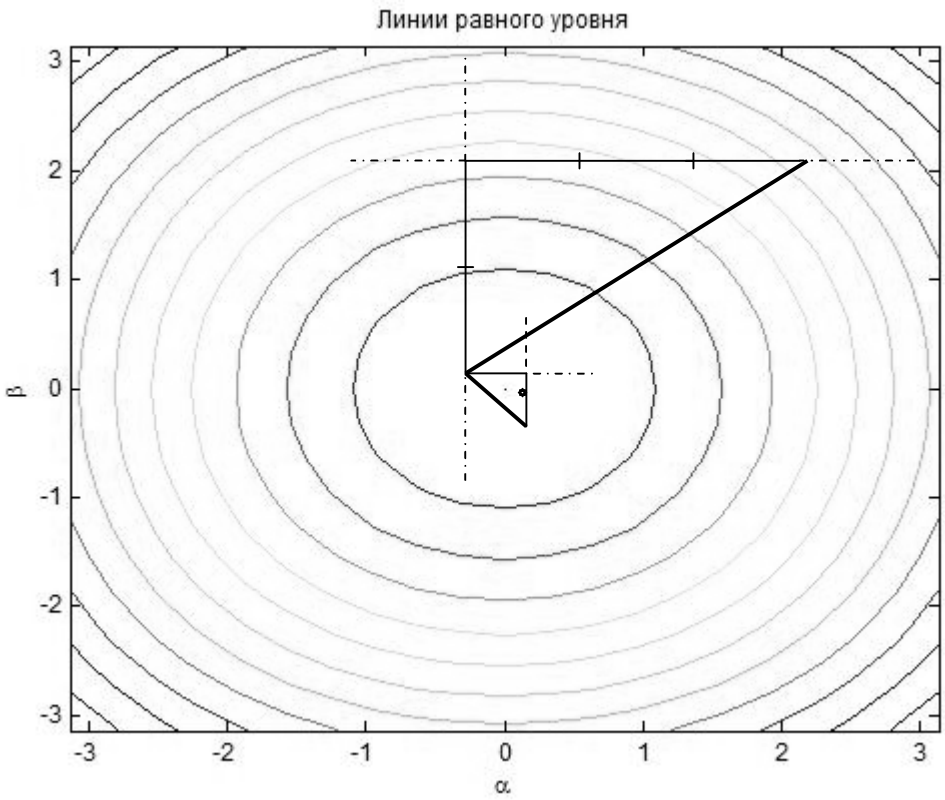

Рисунок 2.20– Поиск решения методом траекторий

Для демонстрации использована функция вида $f(\alpha,\beta) = -\frac{\sin(\alpha+\beta)}{\alpha+\beta}$, которая достигает своего «дна» в точке с координатами (0, 0), это положение отмечено точечкой в середине рисунка, пунктиром показаны неудачные пробы, жирным – траектория спуска.

Замечания.

К безусловным ***достоинствам*** метода следует отнести следующие:

- способность восстанавливать направление движения, когда при искривлении «оврага» тренд поиска теряется;
- явного задания функции не требуется;
- легко учитываются ограничения на переменные и область поиска.

В качестве недостатка отмечается, что при попадании в область размытого локального минимума есть риск остановиться, не достигнув глобального минимума.

Метод Розенброка [46, 47]

Идея метода состоит в поиске по взаимно-ортогональным направлениям на каждом шаге алгоритма, в отличие от алгоритма Хука-Дживса, где поиск осуществляется вдоль координатных осей.

Если поиск, в каком либо из направлений, оказывается удачным, то шаг поиска по этому направлению увеличивается, в противном случае шаг уменьшается.

Поисковая процедура заканчивается, когда на каждом из направлений поиска одна проба окажется успешной, а одна – неудачной.

Данный метод ***особенно хорош*** при работе с функциями, обладающими длинными, узкими и искривлёнными гребнями и оврагами.

Топология алгоритма представлена на рисунке 2.21.

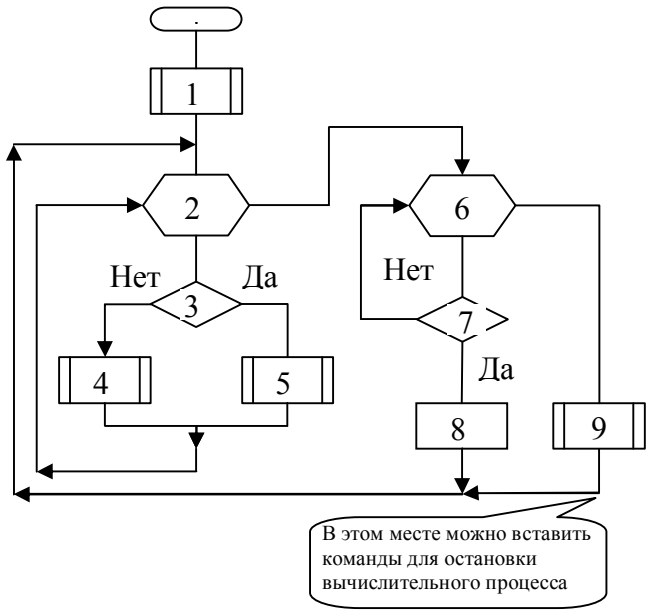

Рисунок 2.21 – Алгоритм метода Розенброка

Входные данные. Координаты начальной точки X_0; список возможных направлений поиска, первоначально набор ортогональных векторов $S = \{s_1, s_2, \ldots, s_n\}$; набор значений шагов поиска $\Lambda^T = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$; ускоряющий множитель α; замедляющий множитель β.

Условие окончания: автором не предусмотрено, ибо, скорее всего, считал вручную и вычислительный процесс контролировал визуально. При компьютерной реализации используют один из возможных подходов:

- выполняют несколько фиксированных шагов, после чего алгоритм прекращает свою работу;
- модуль вектора приращений (расстояние между смежными текущими точками на итерации) $|\Delta| \le \varepsilon$ несколько раз подряд.

Розенброком были предложены следующие значения констант: $\alpha = 3$, $\beta = -0{,}5$.

Блок № 1 осуществляет подготовительные операции, $Y = X_0$.

Блоки №№ 2 – 5 выполняют сканирование направлений: заголовок цикла перебора (блок № 2); вычисление точки в окрестностях текущей точки в выбранном направлении, значения функции и сравнение значений : $f(Y + \lambda_i s_i) < f(Y)$(блок № 3).

Если значение функции в выбранном направлении увеличивается или неизменно, выход «нет», то текущее направление поиска штрафуется, путём умножения $\lambda_i = \beta \times \lambda_i$ и запоминается соответствующие координаты в окрестностях текущей точки $P_i = Y + \lambda_i s_i$ – блок № 4.

Если функции в выбранном направлении убывает, выход «да», то направление поиска поощряется $\lambda_i = \alpha \times \lambda_i$, и также запоминаются координаты $P_i = Y + \lambda_i s_i$ – блок № 5.

Блоки №№ 6 и 7 осуществляют поиск удачного направления среди просканированных направлений. Блок № 6 – организует циклический перебор, блок № 7 проверяет условие $f(P_i) < f(Y)$.

Если первое же попавшееся направление удачное, текущая точка перемещается блоком № 8, $X_0 = P_i$, после чего вычислительный процесс возобновляется со сканирования направлений.

Если удачное направление поиска не найдено, то необходимо модифицировать систему направлений, что осуществляется блоком № 9:

- вычисляется приращение $\Delta = X_0 - Y$;
- переопределяется новая точка начала итерации $Y = X_0$;
- новое направление s_1 берется совпадающим с направлением вектора Δ (единичные значения по ненулевым координатам), а остальные направления пересчитываются с учётом условия ортогональности.

2.5.2.2. Градиентные методы поиска [40, 74, 83]

Эти методы основываются на использовании значений градиентов целевой функции, как следует из их названия.

В зависимости от порядка используемых градиентов бывают первого и второго порядков. Далее мы рассмотрим

- метод наискорейшего спуска (подъёма) – 1-го порядка;
- метод Ньютона (вторых производных) – 2-го порядка;
- метод сопряжённых направлений (сопряжённого градиента), предложенный Флетчером и Ривсом, 1-го порядка;
- методы переменной метрики, квазиньютоновские, 1-го порядка.

Общая идея, положенная в основу этих методов, для случая функции одной переменной, пояснена с помощью рисунка. 2.22.

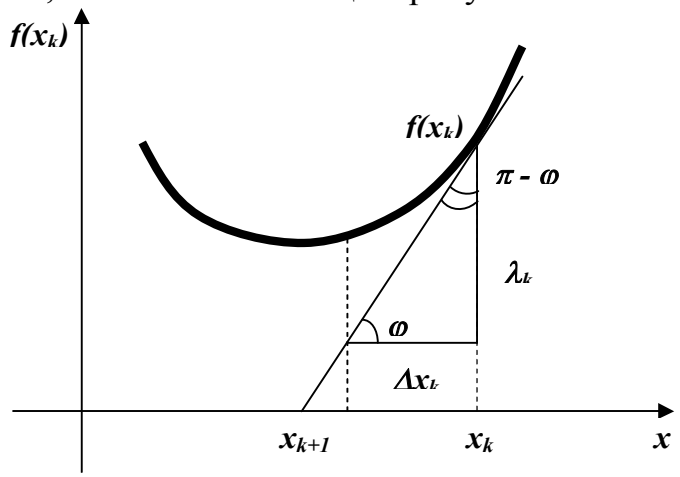

Рисунок 2.22 – Использование градиента

По рисунку видно, что $x_{k+1} = x_k - \Delta x_k$ и $\tan\varphi = f'(x_k)$ – геометрический смысл первой производной функции одной переменной.

Рассмотрение прямоугольного треугольника позволяет найти $\Delta x_k = \lambda_k \cdot f'(x_k)$ при заданном значении λ_k. В этом случае нами использованы формулы связи значений тригонометрических функций при изменении аргумента на величину $\pm\pi$.

Для многомерного случая справедливо аналогичное выражение

$$x_{k+1} = x_k - \lambda_k \times \nabla f(x_k), \qquad (2.81)$$

где $\lambda_k > 0$ – величина шага на k-ой итерации. Особенности отдельных градиентных методов заключаются в приёмах определения λ_k на каждой итерации.

Метод наискорейшего спуска (подъёма) [9, 85, 87]

При решении задачи на минимизацию речь идёт о спуске, а в случае максимизации – о подъёме. В алгоритмах указанное отличие отражается в знаках при выборе направления, которое совпадает с направлением градиента в задачах максимизации и противоположно ему (знак «минус») в задачах минимизации.

Луи Огюст Коши предложил выбирать значение λ_k путём решения задачи одномерной минимизации из условия

$$\lambda_k^* \Rightarrow \min_{\lambda>0} f(X_k + \lambda s_k), \tag{2.82}$$

где $s_k = -\nabla f(X_k)$ определяется значением градиента. При этом непосредственно значение λ_k может быть определено из уравнения

$$\frac{\partial f(X_k + \lambda s_k)}{\partial \lambda} = 0 \tag{2.83}$$

аналитически, либо путём численного решения задачи одномерной минимизации. Алгоритм метода тривиальный, изображён на рисунке 2.23.

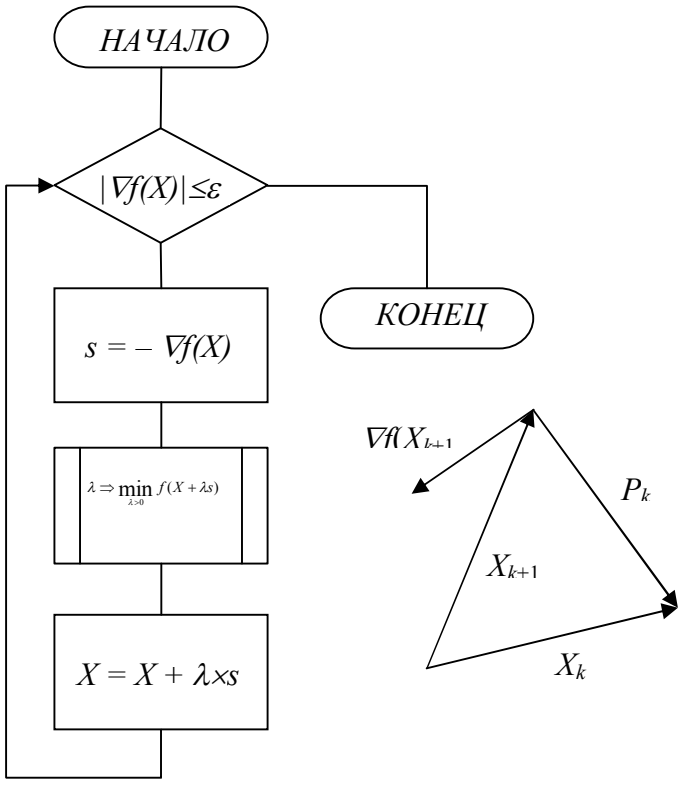

Рисунок 2.23 – Алгоритм метода наискорейшего спуска

Если ввести вектор, определяемый как $\vec{P}_k = \vec{X}_{k+1} - \vec{X}_k$, то вектор-градиент $\nabla\vec{f}(X_{k+1})$ перпендикулярен вектору приращения $\vec{P}_k$, то есть $\nabla\vec{f}(X_{k+1}) \perp \vec{P}_k$. Поэтому траектория спуска будет иметь, в общем случае, вид зигзага.

Тут уместно заметить, что метод наискорейшего спуска ***не обладает*** оптимальными свойствами, ***не обеспечивая минимум шагов*** при минимуме ***вычислений***, если только ***направление спуска не совпадает*** с направлением на минимум, локальный или глобальный.

Последнее справедливо только для ограниченного класса функций, поэтому метод считается ***полношаговым***. Наискорейшим спуск будет, например, для нелинейных функций чётных степеней.

Метод Ньютона (вторых производных) [9, 89 – 91]

Для итерационных методов, как мы знаем, характерна следующая связь между новыми и старыми координатами в пределах одной итерации

$$X_{k+1} = X_k + \Delta X_k. \tag{2.84}$$

Отыщем значение ΔX_k в ходе решения оптимизационной задачи с использованием ряда Тейлора 2-го порядка. Для функции одной переменной ряд, ограниченный до трёх членов, в окрестностях точки X_0 будет иметь вид

$$f(x) \cong f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2,$$

а для многомерного случая

$$f(X) \cong f(X_0) + \nabla^T f(X_0)(X - X_0) + \frac{1}{2}(X - X_0)^T \nabla^2 f(X_0)(X - X_0),$$

где $\nabla^T f(X_0)$ – матрица Гёссе, составленная из частных производных второго порядка. Определяя ΔX_k ка (2.84), придём к формуле

$$f(X_{k+1}) = f(X_k) + \nabla^T f(X_k)\Delta X_k + \frac{1}{2}\Delta X_0^T \nabla^2 f(X_k)\Delta X_k.$$

Для отыскания оптимального значения ΔX_n последнее выражение подвергнем дифференцированию, а результат положим равным нулю. Отсюда

$$\Delta X_k = -\frac{\nabla f(X_k)}{\nabla^2 f(X_k)}.$$

Окончательно рекуррентное выражение для расчётов

$$X_{k+1} = X_k - \frac{\nabla f(X_k)}{\nabla^2 f(X_k)}. \qquad (2.85)$$

Если сравнить (2.85) и (2.81), видно, что для данного случая

$$\lambda_k = \left[\nabla^2 f(X_k)\right]^{-1},$$

то есть, λ_k на каждой итерации равно обращённой матрице Гёссе в текущей точке.

Выражение (2.85) есть задание алгоритма метода Ньютона в виде формулы, которая применяется циклически применяется циклически, до удовлетворения необходимой точности расчётов, то есть $|\Delta X_k| \le \varepsilon$.

Замечание. Обращение матрицы Гёссе на каждом шаге – основное неудобство метода, ведущее в временным затратам и росту погрешности вычисления. Последнее, отчасти, препятствует широкому использованию метода в практических приложениях.

Метод сопряжённого градиента (Флетчера – Ривса) [9, 34]

Понятие о сопряжённых направлениях

В общем случае, система линейно-назависимых направлений поиска S_1, S_2, $\ldots S_{n-1}$ называется ***сопряжённой*** по отношению к некоторой положительно определённой матрице Q если

$$S_i^T Q S_j = 0; \quad i \neq j; \quad i = \overline{1, n-1}, \quad j = \overline{1, n-1}.$$

Если $Q = I$ – единичная матрица, то вектора S_i и S_j, $i \neq j$ ортогональны.

Направление спуска S_1 будет сопряжено направлению S_0, если выполняется тождество:

$$S_0^T \cdot \nabla^2 f(X_0) \cdot S_1 = 0.$$

Как известно из математики [20, 21], для положительно определённой матрицы её собственные вектора образуют систему из $n - 1$ направлений, и такая система всегда существует, по крайней мере, одна.

Пусть исходное направление поиска в некоторой точке X_n есть

$$S_n = -\nabla f(x_n). \tag{2.86}$$

Сопряжённое направление будем строить по правилу

$$S_{n+1} = -\nabla f(x_{n+1}) + \omega_{n+1} S_n, \tag{2.87}$$

где ω_{n+1} – весовой коэффициент сопряжения. Этот коэффициент должен обеспечивать условие сопряжённости, то есть

$$S_n^T \nabla^2 f(x_n) S_{n+1} = 0. \tag{2.68}$$

Алгоритм движения точки в ходе поиска подчиняется известному выражению

$$x_{n+1} = x_n + \lambda S_n \Rightarrow \Delta x_n = \lambda S_n. \tag{2.89}$$

Необходимо найти нормирующий множитель ω_{n+1}.

Воспользуемся для начала аппроксимацией матрицы Гёссе по методу Л. Эйлера

$$\frac{\nabla f(x_{n+1}) - \nabla f(x_n)}{x_{n+1} - x_n} \cong H(x_n) = \nabla^2 f(x_n).$$

Из последнего выражения, определив знаменатель дроби через направление S_n по выражению (2.89), последует

$$\nabla f(x_{n+1}) - \nabla f(x_n) = H(x_n) \lambda S_n,$$

откуда получим

$$S_n^T = \frac{1}{\lambda} \left[\nabla f(x_{n+1}) - \nabla f(x_n)\right]^T H^{-1}(x_n). \tag{2.90}$$

Подстановка (2.86) в (2.87)

$$S_{n+1} = -\nabla f(x_{n+1}) - \omega_{n+1} \nabla f(x_n),$$

а, в дальнейшем, полученного результата совместно с (2.90) – в условие сопряжённости (2.88), даёт

$$\frac{1}{\lambda}\left[\nabla f(x_{n+1}) - \nabla f(x_n)\right]^T H^{-1}(x_n) H(x_n)\left[-\nabla f(x_{n+1}) - \omega_{n+1}\nabla f(x_n)\right] = 0.$$

Из последней записи, принимая во внимание, что $H^{-1}(x_n)H(x_n) = I$, выразим весовой коэффициент сопряжения:

$$\omega_{n+1} = \frac{\nabla^T f(x_{n+1}) \cdot \nabla f(x_{n+1})}{\nabla^T f(x_n) \cdot \nabla f(x_n)} = \frac{\left\|\nabla f(x_{n+1})\right\|^2}{\left\|\nabla f(x_n)\right\|^2}. \qquad (2.91)$$

Результат показывает, что коэффициент сопряжения (2.91) равен отношению квадратов норм (длин) векторов текущего и предшествующего градиентов.

Алгоритм расчётов представлен ниже, на рисунке 2.24. Шаг λ^* отыскивается по методу Коши, как и в методе наискорейшего спуска.

2.5.2.3. Методы поиска переменной метрики (квазиНьютоновские)

Обращение матрицы Гёссе на каждом шаге итерации есть основной недостаток метода второго порядка, предложенного Ньютоном и рассмотренного нами выше. Указанная матрица может быть плохо обусловленной, а численное обращение трудоёмким и приводить к погрешности – число операций умножения пропорционально n^3, где n – размерность матрицы. Методы переменной метрики или ***квазиньютоновские*** производят построение аппроксимирующей матрицы $\left[H(X_k)\right]^{-1}$ в процессе спуска [34, 67, 68].

Пусть алгоритм движения точки подчиняется закону

$$x_{n+1} = x_n - \lambda_n D_n \nabla f(x_n),$$

где D_n - аппроксимирующая матрица, λ_n - стабилизирующий множитель.

Аппроксимирующую матрицу D_n строят по правилу:

$$H^{-1}(x_n) \approx D_{n+1} = \omega(D_n + \Delta D_n), \qquad (2.92)$$

ω - нормирующий множитель.

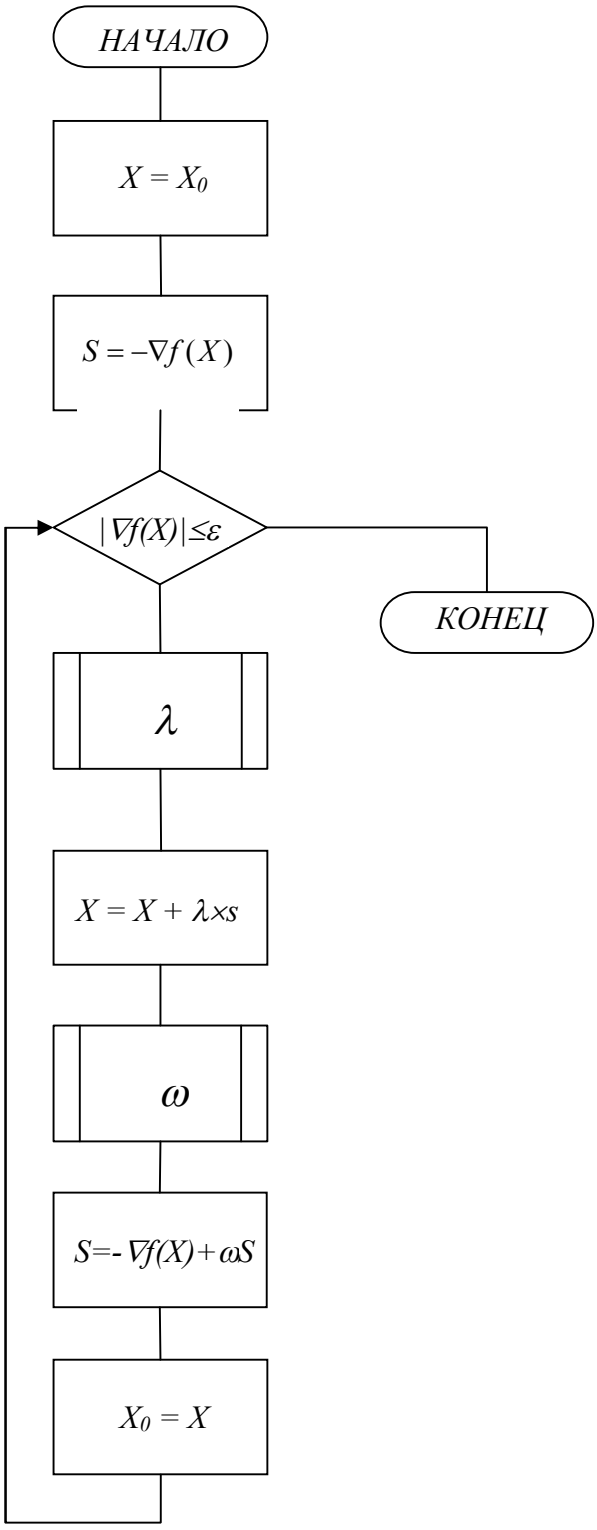

Рисунок 2.24 – Алгоритм метода Флетчера – Ривса

Определим приращение ΔD_n, обеспечивающее корректное построение матрицы $\left[H(X_k)\right]^{-1}$.

Воспользуемся аппроксимацией матрицы Гёссе

$$H(x_n) = \frac{\Delta g_n}{\Delta x_n}, \qquad (2.93)$$

где $\Delta g_n = \nabla f(x_{n+1}) - \nabla f(x_n)$, а $\Delta x_n = x_{n+1} - x_n$.

Приращение Δx_n определим из (2.93):

$$\Delta x_n = H^{-1}(x_n) \cdot \Delta g_n .$$

Подстановка сюда определения (2.92) даёт выражение

$$\Delta x_n = \omega(D_n + \Delta D_n)\Delta g_n \Rightarrow \Delta D_n \Delta g_n = \frac{1}{\omega}\Delta x_n - D_n \Delta g_n ,$$

откуда

$$\Delta D_n = \frac{1}{\omega} \cdot \frac{\Delta x_n}{\Delta g_n^T} \cdot \frac{Y^T}{Y} - \frac{D_n \Delta g_n}{\Delta g_n^T} \cdot \frac{Z^T}{Z}.$$

Фиктивные добавки – векторы Y и Z, размерности n, решают задачи определимости матричных операций, сходимости D_n к $H^{-1}(x_n)$ и её положительной определённости.

В алгоритме, предложенном ***Бройденом***, положено: $\omega = 1,\ Y = Z = \Delta x_n - D_n \Delta g_n$.

В алгоритме ***Давидона-Флетчера-Пауэлла*** аналогичные добавки определяются как: $\omega = 1,\ Y = \Delta x_n,\ Z = D_n \Delta g_n$.

В обоих случаях, в качестве начального приближения используется единичная матрица I.

Для последнего алгоритма

$$D_{n+1} = D_n + \frac{\Delta x_n}{\Delta g_n^T} \cdot \frac{\Delta x_n^T}{\Delta x} - \frac{D_n \Delta g_n}{\Delta g_n^T} \cdot \frac{(D_n \Delta g_n)^T}{D_n \Delta g_n} = D_n + A_n - B_n$$

или

$$D_{n+1} = I + \sum_{i=1}^{n} A_i - \sum_{i=1}^{n} B_i ,$$

в котором A_n обеспечивает сходимость D_n к $H^{-1}(x_n)$, а B_n положительную определённость.

Схема работы алгоритма – типовая для градиентных методов.

Параметр $\lambda_n^* \Rightarrow \min_{\lambda > 0} f(X_n + \lambda_n s_n)$, где $s_n = -\nabla f(X_n)$, вычисляется по методу Коши.

Условия окончания работы алгоритма задаются одним из нижеследующих способов.

1. Каждая из компонент векторов $D_n \nabla f(x_n)$ и $\lambda_n D_n \nabla f(x_n)$ меньше по модулю некоторой заданной величины ε.

2. Длина (норма) каждого из векторов $D_n \nabla f(x_n)$ и $\lambda_n D_n \nabla f(x_n)$ меньше величины ε.

2.5.3. Поиск экстремумов в задачах нелинейного программирования при ограничениях типа “равенство” (метод Лагранжа) [46, 69]

В основу метода положена идея сведения задачи поиска условного экстремума к поиску безусловного экстремума специальной функции. Пусть математическая модель представлена в виде

$$\begin{cases} f(x_1, x_2, ..., x_n) \to \max, \\ h_1(x_1, x_2, ..., x_n) \ge 0, \\ h_2(x_1, x_2, ..., x_n) \ge 0, \\ ... \\ h_m(x_1, x_2, ..., x_n) \ge 0. \end{cases} \qquad (2.94)$$

Допустим, что функции $f(X)$ и $h_i(X)$, $i = 1, m$, входящие в (2.94), нелинейные и имеют непрерывные частные производные.

Введём набор множителей $\lambda_1, \lambda_2, ..., \lambda_m$, по числу ограничений в модели (2.94), и составим функцию вида

$$L(x_1, x_2, ..., x_n, \lambda_1, \lambda_2, ..., \lambda_m) = f(x_1, x_2, ..., x_n) + \sum_{i=1}^{m} \lambda_i h_i(x_1, x_2, ..., x_n) \qquad (2.95)$$

Полученная ***функция*** называется функцией ***Лагранжа***, а множители λ, $i = 1, m$, – ***коэффициентами*** или ***множителями Лагранжа***.

Часть выражения, стоящего под знаком суммы в (2.95), можно интерпретировать как результат решения системы уравнений, в ходе которого происходит сложение или вычитание отдельных её уравнений, помноженных на коэффициенты.

Очевидно, что экстремум (2.95) будет при таких значениях $X^* = \{x_1^*, x_2^*, ..., x_n^*\}$ и $\Lambda^* = \{\lambda_1^*, \lambda_2^*, ..., \lambda_m^*\}$, которые будет удовлетворять системе уравнений

$$\left.\begin{aligned}\frac{\partial L(X^*,\Lambda^*)}{\partial x_j}=0, j=\overline{1,n},\\ \frac{\partial L(X^*,\Lambda^*)}{\partial \lambda_i}=0, i=\overline{1,m}.\end{aligned}\right\} \quad (2.96)$$

Эта система компактно записана в форме

$$\nabla f(X^*)+\sum_{i=1}^{m}\lambda_i^*\nabla h_i(X^*)=0. \quad (2.97)$$

Условия (2.96) суть ***необходимые*** условия существования экстремума, а запись (2.97) – отражает ***достаточность***.

Вектор $\nabla f(X^*)$ лежит в гиперплоскости, "натянутой на вектора $\nabla h_i(X^*)$" [33], эдакий зонтик. Известна следующая теорема, носящая имя Лагранжа.

Теорема [33]. Пусть существует точка X^*, в которой достигается экстремум функции *f(X)* при ограничениях $h_i(X) = 0,\ i = \overline{1,m}$. Если ранг матрицы $I=\left[\frac{\partial h_i}{\partial x_j}\right], i=\overline{1,m}; j=\overline{1,n}$ в точке X^* равен m, то существует m вещественных чисел $\lambda_1, \lambda_2, \ldots, \lambda_m$, ***не все из которых равны нулю одновременно***, при которых выполняется условие

$$\nabla f(X^*)+\sum_{i=1}^{m}\lambda_i^*\nabla h_i(X^*)=0.$$

Алгоритм метода Лагранжа

1. Составить функцию Лагранжа $L(X, \Lambda)$.
2. Найти частные производные функции Лагранжа (2.96).
3. Решить систему уравнений для определения X^* и Λ^*.
4. Исследовать полученные точки на максимум или минимум.

Пример практического применения метода Лагранжа

Содержательная постановка [6].

Цикл эксплуатации блока датчиков информационного зонда включает следующие фазы: $\tau_П$ – подготовки к применению; $\tau_Г$ – готовности к применению, $\tau_Р$ – работы (применения). Требование к надёжности формулируется в виде показателя безотказной работы R. Каковы должны

быть требования по надёжности к каждой фазе эксплуатации, если известны потери от отказа на каждой фазе, задаваемые неотрицательными величинами $C_i > 0$, $i = П, Г, Р$?

Время эксплуатации изделия в целом есть

$$\tau_\Sigma = \tau_П + \tau_Г + \tau_Р,$$

а надёжность характеризуется показателем

$$R = P_П \cdot P_Г \cdot P_Р,$$

где P_i, $i = П, Г, Р$ – вероятность безотказной работы на соответствующей фазе.

Если последствия отказов равнозначны, то задача имеет тривиальное решение:

$$P_i = \sqrt[3]{R}\ ,\ i = П, Г, Р.$$

В противном случае имеем оптимизационную задачу: необходимо рассчитать вероятностные характеристики таким образом, чтобы общие потери были минимальные. Функция цели или функция потерь имеет вид

$$C_\Sigma = C_П \cdot (1 - P_П) \cdot P_Г \cdot P_Р + C_Г \cdot P_П \cdot (1 - P_Г) \cdot P_Р + C_Р \cdot P_П\ P_Г \cdot (1 - P_р),$$

при ограничении

$$R = P_П\, P_Г\, P_Р\ .$$

Таким образом, имеем оптимизационную задачу, которую будем решать изложенным выше методом.

Функция Лагранжа получается такова:

$$L = C_\Sigma + \lambda[P_П \cdot P_Г \cdot P_Р - R].$$

В ходе применения метода, λ из первого, например, уравнения для частных производных $\dfrac{\partial L}{\partial P_i}$, подставляется в остальные.

$$\begin{cases} C_Г \cdot P_П - C_П \cdot P_Г = 0, \\ C_Г \cdot P_Р - C_Р \cdot C_Г = 0, \\ R = P_П \cdot P_Г \cdot P_Р. \end{cases} \Rightarrow \begin{cases} P_Р = \dfrac{C_П \cdot R}{C_Г \cdot P_П^2}, \\ P_П = \dfrac{C_Р \cdot R}{C_Г \cdot P_Р^2}. \end{cases}$$

Далее следуют алгебраические преобразования. Окончательно имеем оптимум при переменных

$$P_П^* = \sqrt[3]{\frac{C_П^2 R}{C_Г C_Р}}, \qquad P_Г^* = \sqrt[3]{\frac{C_Г^2 R}{C_П C_Р}}, \qquad P_Р^* = \sqrt[3]{\frac{C_Р^2 R}{C_П C_Г}}.$$

Иногда стоимость потерь при отказах удобно выражать в относительных единицах:

$$a_1 = \frac{C_Г}{C_П}, \quad a_2 = \frac{C_П}{C_Р}, \quad a_3 = \frac{C_Р}{C_Г}, \text{ причём } a_1 a_2 a_3 = 1.$$

Тогда решение в оптимуме примет вид

$$P_П^* = \sqrt[3]{\frac{Ra_2}{a_1}}, \; P_Г^* = \sqrt[3]{\frac{Ra_1}{a_3}}, \; P_Р^* = \sqrt[3]{\frac{Ra_3}{a_2}}.$$

2.5.4. Общий случай задачи нелинейного программирования [12, 68]

Задачи общего вида подразделяются на задачи ***выпуклого*** и ***вогнутого*** программирования, но, независимо от их типа, на переменные накладываются условия неотрицательности $x_j \geq 0$, $j = 1,n$.

Математическая модель для задачи выпуклого программирования имеет вид

$$\begin{aligned} & f(x_1, x_2, \ldots, x_n) \to \min, \\ & \begin{cases} g_1(x_1, x_2, \ldots, x_n) \leq 0, \\ g_2(x_1, x_2, \ldots, x_n) \leq 0, \\ \ldots \\ g_m(x_1, x_2, \ldots, x_n) \leq 0, \\ x_j \geq 0, \qquad j = \overline{1, n}. \end{cases} \end{aligned} \tag{2.98}$$

когда функции $f(X)$ и все $g_i(X)$, $i = 1, m$ выпуклы (в смысле НП-задач).

Для случая вогнутого программирования характерно

$$f(x_1,x_2,...,x_n) \to \max,$$
$$\begin{cases} g_1(x_1,x_2,...,x_n) \ge 0, \\ g_2(x_1,x_2,...,x_n) \ge 0, \\ ... \\ g_m(x_1,x_2,...,x_n) \ge 0, \\ x_j \ge 0, \qquad j = \overline{1,n}. \end{cases}$$

при функциях $f(X)$ и всех $g_i(X)$, $i = 1,m$ вогнутых (выпуклых вверх).

Связь между задачами выпуклого и вогнутого программирования в части целевой функции и системы ограничений определяется множителем (– 1). То есть

$$\max U(X) = \min [-U(X)].$$

Поэтому, обозначая

$$\hat{f}(X) = -f(X),$$
$$\hat{g}_i(X) = -g_i(X), \quad i = \overline{1,n}.$$

перейдём к следующей задаче выпуклого программирования:

$$\hat{f}(x_1,x_2,...,x_n) \to \min,$$
$$\begin{cases} \hat{g}_1(x_1,x_2,...,x_n) \le 0, \\ \hat{g}_2(x_1,x_2,...,x_n) \le 0, \\ ... \\ \hat{g}_m(x_1,x_2,...,x_n) \le 0, \\ x_j \ge 0, \qquad j = \overline{1,n}. \end{cases}$$

Рассмотрим общий случай задачи НП-программирования (2.98):

Если в точке минимума X^* неравенство $g_i(X)$ выполняется как ***равенство***, то оно называется ***активным***. В этом случае, согласно теореме Лагранжа, формула (2.97), для активных ограничений имеем

$$\nabla f(X^*) + \sum_I \lambda_i \nabla g_i(X^*) = 0, \qquad I = \{i : g_i(X^*) = 0\}. \tag{2.99}$$

Введём в рассмотрение вектор Λ с неотрицательными компонентами, обеспечивающими ***условие линейной независимости***

$$\sum_{i=1}^{m} \lambda_i g_i(X^*) = 0. \tag{2.100}$$

Условия (2.99) и (2.100) – суть условия ***регулярности*** ограничений.

Существует теорема Куна – Таккера.

Теорема[33]. Пусть функции $f(X)$ и $g_i(X)$, $i = 1, m$ обладают частными производными на некоторой области $\mathfrak{R}$, содержащей X^*. Точка X^* будет являться точкой минимума функции $f(X)$ при ограничениях $g_i(X) \le 0$, $i = 1, m$, удовлетворяющих условиям регулярности в виде линейной независимости, если существуют такие неотрицательные множители Лагранжа $\lambda_1, \lambda_2, \ldots, \lambda_m$, что

$$\begin{cases} \nabla f(X^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(X^*) = 0; \\ \sum_{i=1}^{m} \lambda_i g_i(X^*) = 0; \\ \lambda_i \ge 0, \ i = \overline{1, m}. \end{cases} \tag{2.101}$$

Ограничения (2.100) называются ***условиями дополняющей нежёсткости*** или условиями Слейтера, а точка X^* – точкой Куна-Таккера.

Смысл множителей Лагранжа в данной ситуации состоит в том, что они позволяют “нейтрализовать” (аннулировать) неактивные ограничения, которые и без того выполняются. Тем самым, задача нелинейного программирования общего вида сводится к задаче Лагранжа.

2.5.4.1. Седловая точка в НП-задачах

Определение. Пара векторов X^* и Λ^* называется ***седловой точкой*** функции $L(X, \Lambda)$ на области $\mathfrak{R}$, если для всех $\lambda \ge 0$ и $X \in \mathfrak{R}$ выполняется условие

$$L(X^*, \Lambda) \le L(X^*, \Lambda^*) \le L(X, \Lambda^*), \tag{2.102}$$

называемое ***неравенством седловой точки***.

В самой седловой точке выполняется равенство, называемое ещё неравенством(условием) максимина или мимнимакса.

$$\max_{\lambda_i \geq 0} \min_{X \in \Re} L(X, \Lambda) = \min_{X \in \Re} \max_{\lambda_i \geq 0} L(X, \Lambda), \quad i = \overline{1, n}.$$

Пример вида простейшей функции Лагранжа с седловой точкой показан на рисунке 2.25.

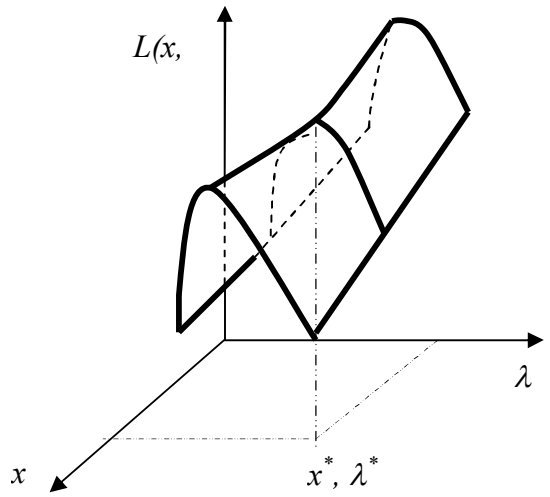

Рисунок 2.25 – Седло нелинейной функции

Существует нижеследующая теорема о седловой точке.

Теорема [33]. Пусть $f(X)$ и все $g_i(X)$, $i = 1, m$ выпуклы, а функции $g_i(X)$ удовлетворяют условиям ***регулярности Слейтера*** (вида $\exists_X \forall_i g_i(X) < 0$). Вектор X^* является решением задачи нелинейного программирования

$$\min f(X),$$
$$g_i(X) \leq 0, \quad i = \overline{1, m}.$$

тогда и только тогда, когда существует такой вектор Λ с неотрицательными компонентами, что выполняются неравенство ***седловой точки*** (2.102) и ***условие линейной независимости*** (2.100).

Условие регулярности Слейтера означает наличие в области ограничений точек, в которых ограничения задачи выполняются строго.

Таким образом, НП-задача может быть, в принципе, сведена к задаче поиска седловой точки функции Лагранжа. При этом преференции будут отданы тому решению, ход решения которого более прост.

2.5.4.2. Применение теоремы Куна-Таккера к НП-задачам [68]

Имеем выпуклую задачу НП-программирования

$$\min f(X);$$
$$\begin{cases} g_i(X) \le 0, \quad i = \overline{1,m}; \\ x_j \ge 0, \qquad\qquad j = \overline{1,n}. \end{cases}$$

Введём дополнительные ограничения, учитывающие неотрицательность переменных x_j в явном виде: $x_j = -h_j(x_j)$, $j = 1, n$, откуда непосредственно следует, что $h_j(x_j) \le 0$, $j = 1, n$.

Для такой расширенной системы построим функцию Лагранжа

$$L(X,\Lambda,U) = f(X) + \sum_{i=1}^{m} \lambda_i g_i(X) + \sum_{j=1}^{n} u_j h_j(x_j),$$

к которой применим теорему Куна-Таккера, найдём $\nabla L(X, \Lambda, U)$. Это эквивалентно системе уравнений

$$\begin{cases} \dfrac{\partial L(X,\Lambda,U)}{\partial x_j} = \dfrac{\partial f(X)}{\partial x_j} + \sum_{i=1}^{m} \lambda_i \dfrac{\partial g_i(X)}{\partial x_j} - u_j = 0, \qquad j = \overline{1,n}; \\ u_j x_j = 0, \qquad j = \overline{1,n}; \\ \lambda_i g_i(X) = 0, \; i = \overline{1,m} \\ \lambda_i \ge 0, \qquad\qquad i = \overline{1,m}. \end{cases}$$

В точке Куна-Таккера будем иметь следующее:

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} \dfrac{\partial L(X^\circ,\Lambda^\circ)}{\partial x_j} \ge 0, \qquad j = \overline{1,n}; \\ \dfrac{\partial L(X^\circ,\Lambda^\circ)}{\partial x_j} \cdot x_j^\circ = 0, \; j = \overline{1,n}; \end{array} \right\} \\ \left\{ \begin{array}{l} \dfrac{\partial L(X^\circ,\Lambda^\circ)}{\partial \lambda_i} = g_i(X^\circ) \le 0, \; i = \overline{1,m}; \\ \lambda_i g_i(X^\circ) = 0, \qquad i = \overline{1,m}. \end{array} \right\} \end{array} \right.$$

Знак “≥” в первом неравенстве записан, потому, что не все значения u_j равны нулю, а первых два ограничения, выделенные “}”, есть условия дополняющей нежёсткости по u_j.

Для вогнутого программирования, благодаря симметрии задач получается

$$\left\{\begin{array}{l}\left.\begin{array}{ll}\dfrac{\partial L(X^{\circ},\Lambda^{\circ})}{\partial x_j}\le 0, & j=\overline{1,n};\\ \dfrac{\partial L(X^{\circ},\Lambda^{\circ})}{\partial x_j}\cdot x_j^{\circ}=0, & j=\overline{1,n};\end{array}\right\}\\ \left.\begin{array}{ll}\dfrac{\partial L(X^{\circ},\Lambda^{\circ})}{\partial \lambda_i}=g_i(X^{\circ})\ge 0, & i=\overline{1,m};\\ \dfrac{\partial L(X^{\circ},\Lambda^{\circ})}{\partial \lambda_i}\cdot \lambda_j^{\circ}=0,, & i=\overline{1,m}.\end{array}\right\}\end{array}\right.$$

Таким образом, теорема Куна-Таккера предоставляет исследователю удобный инструмент, позволяющий по условию задачи нелинейного программирования построить систему уравнений, решение которой даёт систему векторов X и Λ. Эти векторы дополнительно проверяются на выполнение условий линейной независимости и неотрицательности.

2.5.5. Методы возможных направлений

Указанные методы применяются для ***численного*** решения НП-задач.

Определение. Вектор с ненулевыми компонентами (ненулевой) S называется ***возможным направлением*** спуска в точке $X \in \mathfrak{R}$, если существует $\delta > 0$, такое, что $X + \lambda S \in \mathfrak{R}$ для всех $\lambda \in (0, \delta)$ и $f(X + \lambda S) < f(X)$.

Определение такого направления составляет краеугольный камень, лежащий в основании методов возможных направлений. Типичными представителями таковых являются алгоритмы, предложенные Зойтендейком (есть написание Заутендайка, в оригинале G. Zoutendijk) и Розеном.

2.5.5.1. Метод Зойтендейка [33, 34, 67, 68]

Существуют три разновидности указанного метода:

- при линейных ограничениях;
- при нелинейных ограничениях и
- улучшенной сходимости при нелинейных ограничениях.

Случай линейных ограничений

Постановка задачи, в этом случае, такова

$$\min f(X)$$
$$\begin{cases} AX \le b, \\ HX = h, \end{cases}$$

где $f(X)$ – нелинейна, $A[m \times n]$, $H[l \times n]$ – матрицы, а $b[m]$ и $h[l]$ – вектора системы линейных ограничений.

Пусть в текущей точке все ограничения со знаком “≤” представимы в виде

$$\begin{cases} A_1 X = b_1, \\ A_2 X < b_2. \end{cases}$$

При этом считаем, что исходные матрицы блочные (вернее подлежат разделению или перестановке) по знакам отношений

$$\begin{cases} A^T = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}, \\ b^T = \begin{bmatrix} b_1^T & b_2^T \end{bmatrix}. \end{cases}$$

Вектор S будет являться направлением спуска в точке X при выполнении условий:

$$\begin{cases} A_1 S \le 0, \\ HS = 0, \end{cases}$$

как это показано на рисунке 2.26 ниже.

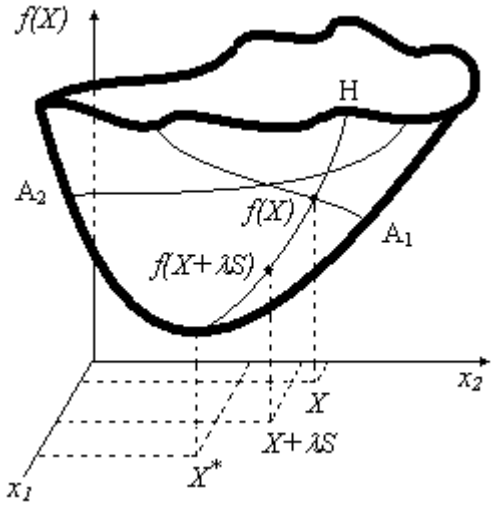

Рисунок 2.26 – К пояснению выбора направления

На изображении целевая функция показана жирной линией. Она напоминает кубок, линии на кубке есть проекции области ограничений на поверхность, описываемую целевой функцией. Из эскиза видно, что спуск должен выполняться «вдоль линии» (или не покидая гиперплоскости, в многомерном случае), по которой выполняются исходные ограничения вида «равенство», в сторону от активных ограничений (неравенства, выполняющиеся в точке в виде равенства, как нами было определено выше).

То есть, выбранное направление спуска S должно обеспечивать выполнение исходных ограничений модели $HX = h$ и устранять активность ограничений $A_1 X = b_1$.

На компоненты вектора S налагаются дополнительные ограничения (требования) нормировки:

- на величину элементов

$$-1 \le s_i \le 1,\ j = \overline{1,n}; \tag{2.103}$$

- на модуль вектора возможного направления S

$$S^T S \le 1, \|S\|^2 \le 1, \tag{2.104}$$

это ограничения объединяет условия $-1 \le s_j \le 1, j = \overline{1,n}$ и $\sum_{j=1}^{n} s_i \le 1, \sum_{j=1}^{n} s_i \ge -1;$

- на величину целевой функции ЗЛП

$$\nabla^T f(X) S \ge -1. \tag{2.105}$$

Поэтому возможна одна из трёх постановок задачи минимизации, которая может быть решена соответствующим методом линейного программирования:

$$\begin{array}{lll}
\min \nabla^T f(X) S & \min \nabla^T f(X) S & \min \nabla^T f(X) S \\
\begin{cases} A_1 S \le 0, \\ HS = 0, \\ s_j \le 1, \\ s_j \ge -1,\ j = 1, m. \end{cases} & \begin{cases} A_1 S \le 0, \\ HS = 0, \\ S^T S \le 1. \end{cases} & \begin{cases} A_1 S \le 0, \\ HS = 0, \\ \nabla^T f(X) S \ge -1. \end{cases} \\
1) & 2) & 3)
\end{array} \tag{2.106}$$

Алгоритм Зойтендейка для случая линейных ограничений

Предварительный этап.

Нахождение точки X, удовлетворяющей системе ограничений задачи.

Итерация.

1. Найти множество активных ограничений в текущей точке и композицию матрицы системы ограничений $A^T = \left[A_1^T \quad A_2^T \right]$ и решить ЗЛП (2.106) вида 1, 2 или 3, по желанию.

2. *Проверка условия окончания*.

Если оптимальное значение целевой функции ЗЛП равно нулю, ***то*** текущие координаты X определяют точку Куна-Таккера.

3. ***Иначе*** решить задачу одномерной минимизации Коши: $\min\limits_{0\le\lambda\le\lambda_{\max}} f(X+\lambda S)$, в которой граничное значение параметра λ определяется как

$$\lambda_{\max}=\begin{cases}\min\limits_k\left\{\dfrac{\widetilde{b}_k}{\widetilde{s}_k}\middle|\widetilde{s}_k>0\right\}, & \exists_k\widetilde{s}_k>0,\\ \infty, & \forall_k\widetilde{s}_k\le 0.\end{cases}\quad\text{где } \widetilde{b}=b_2-A_2X,\qquad \widetilde{S}=A_2S.$$

4. Перейти в новую текущую точку $X = X + \lambda \times S$.

Алгоритм не сложен, он представлен на рисунке 2.25.

Алгоритм Зойтендейка для случая нелинейных ограничений

Рассмотрим задачу НП-программирования вида

$$\begin{cases}\min f(X);\\ g_i(X)\le 0, \quad i=\overline{1,m};\\ x_j\ge 0, \qquad\qquad j=\overline{1,n}.\end{cases}$$

Пусть текущая точка X является допустимой точкой, а $I=\{i: g_i(X)=0\}$ – множество ограничений, активных в этой точке. Если

$$\begin{cases}\nabla^T f(X)S<0,\\ \nabla^T g_i(X)S<0,\ i\in I,\end{cases}$$

то S – вектор возможного направления спуска в этой точке (смотри рисунок 2.26).

Алгоритм имеет топологию, аналогичную представленной рисунком 2.27.

Предварительный этап.

Найти начальную точку X, для которой выполняются ограничения задачи: $g_i(X)\le 0, \quad i=\overline{1,m}$.

Основной этап (итерация).

1. Найти множество активных ограничений задачи в текущей точке $I=\{i: g_i(X)=0\}$ и решить ЗЛП вида

$$\min z$$
$$\begin{cases} \nabla^T f(X)S - z \le 0, \\ \nabla^T g_i(X)S - z \le 0,\ i \in I = \{i : g_i(X) = 0\}. \end{cases}$$

при любом, оговоренном выше, условии нормировки компонентов вектора возможного направления S (2.103) – (2.105).

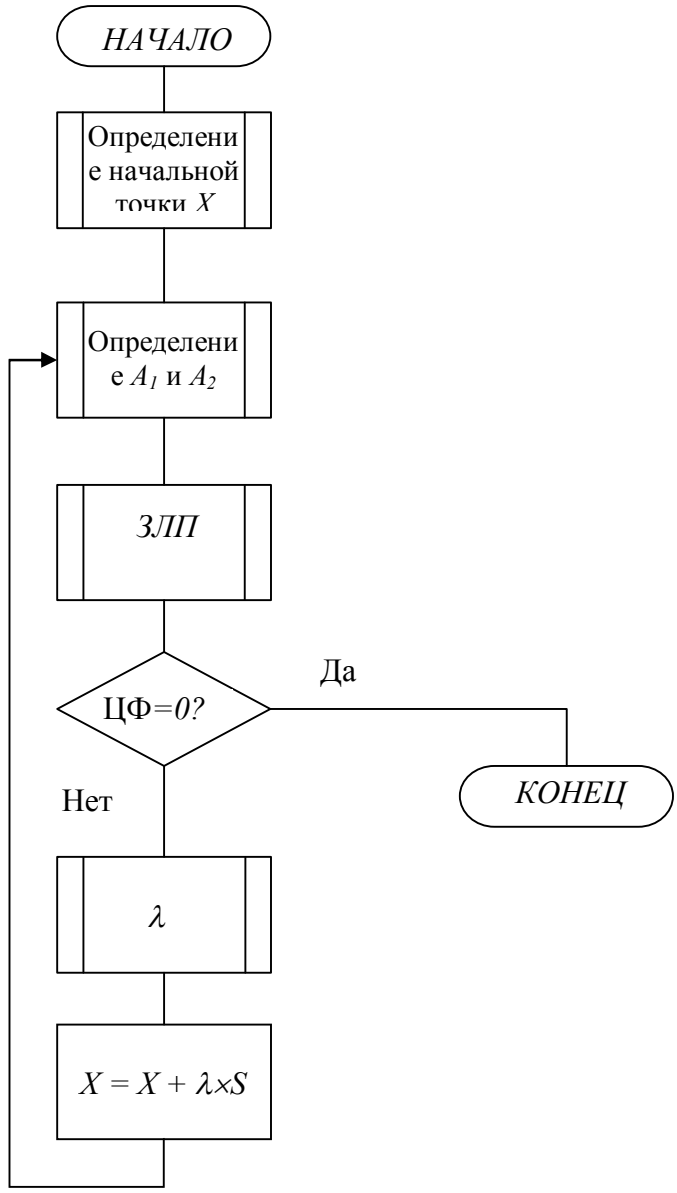

Рисунок 2.27– Граф-схема алгоритма Зойтендейка

2. *Проверка условия окончания*.

Если оптимальное значение целевой функции ЗЛП равно нулю, ***то*** X – оптимальное решение (точка Куна-Таккера) исходной НП-задачи.

3. *В противном случае* необходимо решить задачу одномерной минимизации Коши: $\min\limits_{0\le\lambda\le\lambda_{\max}} f(X+\lambda S)$, где параметр λ определяется из условия

$$\lambda_{\max} = \max\{\lambda : g_i(X+\lambda S) \le 0,\ i=1,m\}.$$

4. Переместится в следующую точку $X = X + \lambda \times S$ и продолжить решение.

Алгоритм Зойтендейка для случая нелинейных ограничений улучшенной сходимости

Шаги, генерируемые вдоль направления поиска, стремятся, по мере итераций, к нулю. Это может, в отдельных случаях, вызвать остановку вычислительного процесса без достижения оптимума. Чтобы избежать несанкционированной остановки, ограничения эквивалентной ЗЛП усиливают.

Модифицированная ЗЛП имеет вид

$$\min z$$
$$\begin{cases} \nabla^T f(X)S - z \le 0, \\ \nabla^T g_i(X)S - z \le -g_i(X),\ i=1,m. \end{cases}$$

При этом нововведения позволяют учитывать как активные, так и обычные ограничения в точке поиска. ЗЛП снабжается одним из условий нормировки (2.103) – (2.105).

2.5.5.2. Метод проекции градиента Розена [9, 68]

Метод предназначен для решения задач нелинейного программирования с линейной системой ограничений

$$\min f(X)$$
$$\begin{cases} AX \le b, \\ HX = h, \end{cases}$$

где $f(X)$ – нелинейная и ***дифференцируемая*** функция, A – матрица размером $[m \times n]$, H – $[l \times n]$-мерная матрица , а b – m-мерный вектор, а h – l-мерный вектор системы линейных ограничений.

Идея, лежащая в основе метода, состоит в следующем.

Пусть X – допустимая точка, удовлетворяющая системе ограничений, а направление спуска определяется градиентом $S = -\nabla f(X)$.

Движение вдоль направления спуска может, гипотетически, привести к ***нарушению допустимости***. Поэтому Розен предложил направление спуска строить по правилу

$$S = -P \times \nabla f(X),$$

где P – матрица проецирования или проецирующая (проектирующая) матрица, которая бы гарантировала сохранение допустимости текущей точки.

Пусть в допустимой точке часть неравенств активна, а другая часть – нет, то допускается представление системы ограничений по признаку активности в виде блочной матрицы:

$$\begin{aligned} &A_1 X = b_1, \quad A^T = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}, \\ &A_2 X < b_2, \quad b^T = \begin{bmatrix} b_1^T & b_2^T \end{bmatrix} \end{aligned} \tag{2.107}$$

Матрица проецирования P должна удовлетворять условию $P \times \nabla f(X) \neq 0$ и блокировать возможные направления в сторону активных ограничений. Это будет соблюдаться, когда проектирующая матрица определена как

$$P = I - M^T \times (M \times M^T)^{-1} \times M, \tag{2.108}$$

где I – единичная матрица, $M^T = [A_1{}^T H^T]$ – невырождена. Отметим, что выполняется тождество $M \times P = 0$, то есть $A_1 \times P = 0$ и $H \times P = 0$, таким образом, спуск в направлении ограничений, выполняемых как равенство, не производится.

Рассмотрим точку, в которой выполняется условие $P \times \nabla f(X) = 0$. Имеем

$$[I - M^T \times (M \times M^T)^{-1} \times M] \times \nabla f(X) = \nabla f(X) + M^T \times W = 0,$$

где $W^T = [U^T V^T]$, а

$$\nabla f(X) + A_1{}^T \times U + H^T \times V = 0.$$

Если компоненты вектора U неотрицательны, то X является точкой Куна-Таккера. В противном случае, можно построить новое направление спуска

$$S = -\widetilde{P} \times \nabla f(X),$$

где $\widetilde{P} = I - \widetilde{M}^T \times (\widetilde{M} \times \widetilde{M}^T)^{-1} \times \widetilde{M}$, в котором $\widetilde{M}^T = [\widetilde{A}_1^T H^T]$, а $\widetilde{A}_1$ получено из A_1 путём вычёркивания строк, соответствующих компонентам $u_j < 0$.

Схема алгоритма показана на рисунке 2.28.

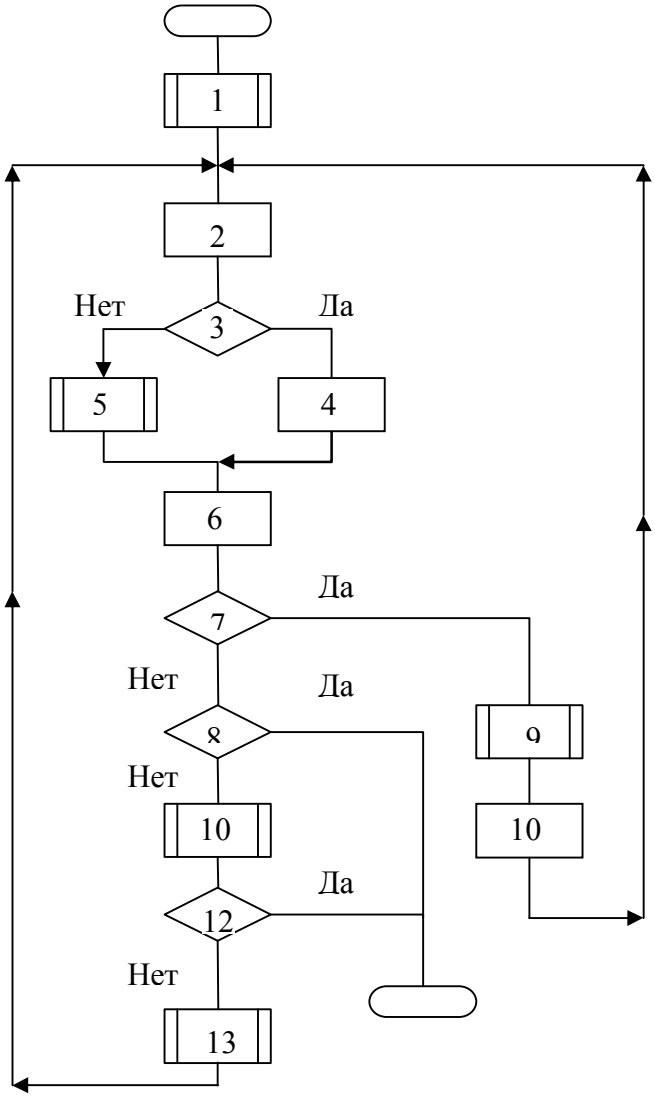

Рисунок 2.28 – Граф-схема алгоритма Розена

Блок № 1 выполняет выбор начальной точки X, удовлетворяющей системе ограничений задачи, отыскивает активные ограничения и разложение ограничений с неравенствами (2.107).

В блоке № 2 конструируется матрица $M^T = [A_1{}^T H^T]$, которая проверяется на равенство нулю всех её элементов условием блока № 3: $M = 0?$, если матрица вся нулевая (выход «да» блока № 3), то проецирующая матрица полагается равной единичной (блок № 4).

В противном случае (выход «нет» блока № 3), выполняется расчёт проецирующей матрицы в блоке № 5 по формуле (2.108).

После определения элементов проецирующей матрицы, рассчитывается возможное направление спуска $S = -P \times \nabla f(X)$ в блоке № 6.

Блок № 7 выполняет проверку: $S \neq 0?$, то есть, не является ли текущая точка точкой Куна-Таккера.

Если все компоненты вектора возможного направления нулевые (выход «нет» блока № 7), то это, возможно, искомое решение. Чтобы убедиться в этом, проверяется равенство нулю всех компонентов блочной матрицы M (блок № 8, $M = 0?$), и если это так (выход «да» блока № 8), то решение успешно заканчивается.

Если среди компонентов матрицы M есть ненулевые элементы (выход «нет» блока № 8), то ищется разложение $W^T = [U^T V^T]$, блок № 10.

Если компоненты подматрицы U неотрицательны, что проверяется блоком № 12 ($\forall_i u_i \geq 0?$), то X является точкой Куна-Таккера (выход «да»).

Если компоненты U отрицательны (выход «нет» блока № 12), то переопределяется матрицы $A_1 = \widetilde{A}_1$ и блочная матрица $\widetilde{M}^T = [\widetilde{A}_1^T H^T]$ (блок № 13).

Если не все элементы вектора возможного направления нулевые (выход «да» блока № 7), рассчитывается новая текущая точка. Сперва выбирается параметр λ решением задачи одномерной минимизации Коши (блок № 9): $\min\limits_{0 \leq \lambda \leq \lambda_{\max}} f(X + \lambda S)$, в которой граничное значение параметра λ определяется, как и в методе Зойтендейка:

$$\lambda_{\max} = \begin{cases} \min\limits_k \left\{ \dfrac{\widetilde{b}_k}{\widetilde{s}_k} \middle| \widetilde{s}_k > 0 \right\}, & \exists_k \widetilde{s}_k > 0, \\ \infty, & \forall_k \widetilde{s}_k \leq 0. \end{cases} \quad \text{где } \widetilde{b} = b_2 - A_2 X, \qquad \widetilde{S} = A_2 S.$$

Значение очередной текущей точки рассчитывается в блоке № 11 по рекуррентной формуле $X = X + \lambda \times S$.

2.5.6. Методы штрафных функций [46, 67, 68]

Методы штрафных функций основаны на практике перехода от задачи ***условной минимизации*** к задаче ***безусловной минимизации*** путём построения специальной штрафной функции.

Известны [3, 9, 24, 33, 40, 56] ***параметрические*** и ***непараметрические*** методы построения штрафных функций в виде полинома

При использовании ***параметрических*** методов, штрафной полином конструируется с использованием выражений, описывающих ограничения, в качестве параметров других функций (функционалов) и весовых коэффициентов.

В *непараметрических* методах функция рассматривается в качестве дополнительного ограничения, которое постоянно усиливается в процессе решения.

По характеру перемещения точки к оптимуму различают:

- методы внутренней точки,
- методы внешней точки и
- комбинированные методы.

При использовании методов ***внутренней точки***, последовательные приближения к оптимуму производятся ***внутри области***, определяемой ограничениями, благодаря специальной функции штрафа, называемой ***барьерной***.

Когда поиск оптимума осуществляется по методу ***внешней точки***, текущее решение находится ***за пределами области*** ограничений, попадая вовнутрь её на последнем шаге итераций.

Комбинированные методы используют, когда большинство ограничений задачи имеют вид равенства, и, в процессе решения, попеременно, одни ограничения выполняются, а другие нет.

Искомое решение получается, при удовлетворении заданных условий, в пределах отведённого допуска.

Пусть условия задачи имеют вид

$$\min f(X);$$
$$\begin{cases} h_i(X) = 0, & i = \overline{1, m}; \\ g_i(X) \geq 0, & i = \overline{m+1, l}. \end{cases}$$

Пользуясь параметрическим методом, по этим условиям можно построить следующую фунцию без ограничений:

$$P(X,\rho) = f(X) + \sum_{i=1}^{m} \rho_i H[h_i(X)] + \sum_{i=m+1}^{l} \rho_i G[g_i(X)],$$

где $P(X,\rho)$ – штрафная функция; ρ_i, $i = \overline{1,l}$ – весовые коэффициенты, отражающие значимость соблюдения того или иного ограничения; $H[h_i(X)]$ и $G[g_i(X)]$ – некоторые функционалы.

Рассмотрим, какими свойствами должны обладать эти функционалы.

Функционал $H[h_i(X)]$ должен сделать невыгодным любое отклонение аргумента X от поверхности $h_i(X) = 0$, то есть

$$\lim_{h_i(X)\to 0} H[h_i(X)] = 0, \qquad i = \overline{1,m}.$$

Поэтому в качестве функционала выбирают чётную степенную функцию

$$H[y] = y^p, \ p = 2,4,... \text{ либо } H[y] = |y|^p, \ p = 1,2,...$$

Функционал $G[g_i(X)]$ зависим от местоположения текущей точки в процессе решения.

- **Метод внутренней точки**:

$$\lim_{g_i(X)\to 0^+} G[g_i(X)] = \infty, \qquad i = \overline{m+1,l} \text{ для } g_i(X) > 0.$$

- **Метод внешней точки**:

$$\lim_{g_i(X)\to 0^-} G[g_i(X)] = 0, \qquad i = \overline{m+1,l} \text{ для } g_i(X) < 0.$$

- **Комбинированный метод**:

$$G[g_i(X)] > 0 \text{ для } g_i(X) < 0, \ i = \overline{m+1,l},$$
$$G[g_i(X)] = 0 \text{ для } g_i(X) = 0, \ i = \overline{m+1,l}.$$

При этом, не зависимо от траектории движения точки, в процессе решения задачи необходимо обеспечение выполнения условий:

$$\lim_{k\to\infty} \sum_{i=1}^{m} \rho_{i,k} H[h_i(X_k)] = 0,$$

$$\lim_{k\to\infty}\sum_{i=m+1}^{l}\rho_{i,k}G[g_i(X_k)]=0,$$
$$\lim_{k\to\infty}\left|P(X_k,\rho_k)-f(X_k)\right|=0,$$

где k – номер итерации.

Особо подчеркнём, что штрафная функция должна быть построена таким образом, что ***невыполнение*** какого либо их ограничений должно приводить ***к резкому возрастанию*** её в целом, и, по мере приближению к оптимуму, ***влияние*** штрафных добавок ***уменьшается***.

2.5.6.1 Метод барьерных поверхностей (МБП) [9, 34, 46, 68]

Данный алгоритм (рисунок 2.29) относится к группе методов внутренней точки, используется для решения задач вида

$$\min f(X);$$
$$\begin{cases} g_i(X)\ge 0, & i=\overline{1,m};\\ x_j\ge 0, & j=\overline{1,n}. \end{cases}$$

По условиям задачи строится штрафная функция вида

$$P(X,r,\Omega)=f(X)+r\cdot\sum_{i=1}^{m}\omega_i G[g_i(X)]. \qquad (2.109)$$

В формуле (2.109) использованы: параметр $r > 0$, убывающий по мере вычислений и Ω – вектор положительных весовых коэффициентов, учитывающих важность (значимость) ограничений модели.

Функции барьера $G[g_i(X)]$ выбирается из альтернатив

$$G[Y]=1/Y \text{ или } G[Y]=-\ln[Y].$$

Легко видеть, что $\lim_{Y\to 0^+}G[Y]=\infty$, то есть функция барьера, при приближении к нему изнутри области, неограниченно возрастает. Штрафная добавка к функции цели определяется формулой

$$\Delta = r \cdot \sum_{i=1}^{m} \omega_i G[g_i(X)]$$

Функционирование алгоритма представлено на рисунке 2.29, в его работе использована уменьшающая константа $0 < \beta < 1$, точность расчётов задаётся константой ε.

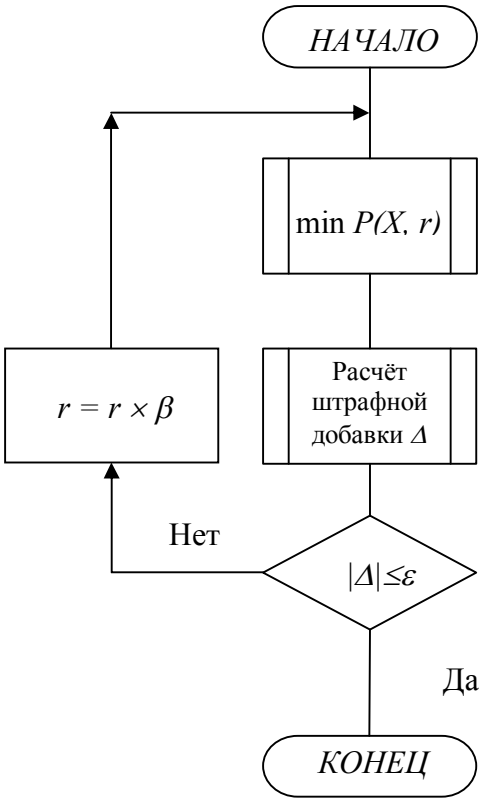

Рисунок 2.29 – Схема алгоритма МБП

Замечания:

- область ограничений должна быть непустой: $X_{opt} \in \{X: g_i(X) > 0, i = 1, m\}$;
- начальное значение X должно быть допустимой точкой;
- метод критичен по отношению к параметру r: если значение r мало, то будет отыскан оптимум функции, не совпадающий с истинным, а если значение r велико, то возрастёт число итераций;
- аналогично, близкие к единице значения β могут привести к преждевременному прекращению вычислений.

2.5.6.2 Метод внешней точки [3, 34, 46, 68]

Ориентируется на решение НП-задач в общей постановке

$$\min f(X);$$
$$\begin{cases} h_i(X) = 0, & i = 1, \overline{m}; \\ g_i(X) \geq 0, & i = m+1, \bar{l}. \end{cases}$$

Штрафная функция описывается выражением

$$P(X, r) = f(X) + rL(X),$$

где r – коэффициент штрафа, а величина штрафа в текущей точке X есть

$$L(X) = \sum_{i=1}^{m} H[h_i(X)] + \sum_{i=m+1}^{l} G[g_i(X)].$$

Используются функционалы:

$$G[Y] = [\max\{0; -Y\}]^P, \; P > 0, \text{ целое},$$
$$H[Y] = |Y|^P, \; P > 0, \text{ целое}.$$

Легко видеть, что функционал $G[Y]$ обращается в нуль при попадании текущей точки в область ограничений.

Алгоритм метода полностью дублирует алгоритм метода барьерных поверхностей, показанный на рисунке 2.29, но, в отличие от него, используется параметр увеличения штрафа β больше единицы.

Если взять параметры r и β достаточно большИми, то минимизация штрафной функции будет выполняться за счёт функции цели и величины штрафа. Поэтому, в ходе расчётов, коэффициент штрафа будет постоянно возрастать, а штраф – убывать. Следовательно, положив r и β порядка десятков, сотен, а то и тысяч, можно получить решение задачи с приемлемой точностью.

Алгоритм, сам по себе, обладает хорошей сходимостью и устойчивостью. Очевидно, что вельми проблемным местом его функционирования будет процедура минимизации штрафной функции, что является нетривиальной задачей.

2.5.7. Задачи квадратичного программирования

Квадратичное программирование – специальный класс НП-задач, в которых $f(X)$ – квадратичная (не выше второй степени переменных) вогнутая (выпуклая вверх) функция, а все ограничения $g_i(X)$, $i = 1,m$ – линейны [33].

Математическая модель такой задачи выглядит следующим образом [3, 23, 31, 33, 34, 36, 68]

$$\begin{aligned} &f(x) = b^T X + \frac{1}{2} X^T C X \to \max, \\ &AX \le A_0,\ X \ge 0, \end{aligned} \tag{2.110}$$

где C — симметричная отрицательно определённая матрица размерностью $[n \times n]$, b^T — вектор-строка размерностью $[1 \times n]$, A — матрица системы ограничений размерностью $[m \times n]$, A_0 — вектор свободных членов системы ограничений размерностью $[m \times 1]$, n — число переменных, m – число ограничений.

Задача решается путём применения теоремы Куна-Таккера, в результате чего получается система линейных ограничений, которую можно решить симплекс-методом.

Функция Лагранжа, построенная по условиям задачи, имеет вид

$$L(X,\Lambda) = b^T \cdot X + \frac{1}{2} X^T \cdot C \cdot X + \Lambda^T (A_0 - A \cdot X).$$

Воспользуемся ранее полученными результатами из пункта 2.5.4.2. Имеем систему уравнений:

$$\begin{cases} \dfrac{\partial L(X,\Lambda)}{\partial X} = b + C \cdot X - A^T \Lambda \le 0, \\ \dfrac{\partial L(X,\Lambda)}{\partial \Lambda} = A_0 - A \cdot X \ge 0. \end{cases}$$

Приведя её к каноническому виду и добавив условия дополняющей нежёсткости, получим констатирующую формулировку нижеследующей теоремы.

Теорема квадратичного программирования [33]. Вектор $X_0 \ge 0$ является оптимальным решением задачи квадратичного программирования

тогда и только тогда, когда существуют такие m-мерные вектора $\Lambda > 0$, $W \geq 0$ и n-мерный вектор $V \geq 0$, что выполняются следующие условия

$$\left.\begin{array}{ll} b + C \cdot X_0 - A^T \Lambda + V = 0, & \text{а)} \\ A_0 - AX_0 - W = 0, & \text{б)} \\ V^T X_0 = 0 & \\ W^T \Lambda = 0 & \text{в)} \end{array}\right\}. \qquad (2.111)$$

Компоненты всех векторов Λ, W и V — неотрицательны, а вектора W и V могут быть нулевыми. Условия (2.111, а) и (2.111, б) образуют систему из $n + m$ уравнений для $2 \times (n + m)$ неизвестных компонентов X, Λ, V и W. Ограничения (2.111, в) есть условия дополняющей нежёсткости.

Алгоритм решения задачи квадратичного программирования

1. Условия (2.111, а) и (2.111, б) необходимо представить в форме, обеспечивающей положительность элементов столбцов свободных членов

$$\begin{cases} A^T \Lambda - C \cdot X - V = b, \\ AX + W = A_0. \end{cases} \qquad (2.112)$$

2. Поскольку знак в ограничениях (2.112) – равенство, следует воспользоваться методом искусственного базиса. Необходимо добавить искусственные переменные $\{y_i\}$ в первую группу уравнений, и $\{z_i\}$, при этом система ограничений примет вид

$$\begin{cases} A^T \Lambda - C \cdot X - V + Z = b, \\ AX + W + Y = A_0. \end{cases} \qquad (2.113)$$

3. Конструируется псевдоцелевая функция [3] из искусственных переменных

$$\sum_{i=1}^{m} \mu \cdot y_i + \sum_{j=1}^{n} \mu \cdot z_j \to \min . \qquad (2.114)$$

4. Решается эквивалентная ЗЛП с функцией цели (2.114) при ограничениях (2.113).

Если в ходе решения ЗЛП векторы Y и Z будут выведены из базиса в полном составе (достигнут оптимум), а полученные значения X, Λ, V и W не отрицательны, а так же удовлетворяют (2.111, в), то компоненты вектора X представляют собой оптимальное решение исходной задачи квадратичного программирования (2.110).

5. Рассчитывается значение целевой функции.

Пример.

Пусть компоненты условия (2.110) выглядят так

$$A=\begin{pmatrix}2 & 5\\ 3 & 2\\ 1 & 1\end{pmatrix},\ A_0=\begin{pmatrix}10\\ 6\\ 4\end{pmatrix},\quad b^T=\begin{pmatrix}2 & 3\end{pmatrix},\ C=\begin{pmatrix}-\frac{1}{2} & \frac{1}{4}\\ \frac{1}{4} & -\frac{1}{2}\end{pmatrix}.$$

Матрица C симметричная, а об отрицательной определённости можно судить по критерию Сильвестра – знаки минорных определителей должны чередоваться [20, 35, 36].

$$C=\begin{pmatrix}-\frac{1}{2} & \frac{1}{4}\\ \frac{1}{4} & -\frac{1}{2}\end{pmatrix},\quad \Delta_1=c_{11}=-\frac{1}{2}<0,\quad \Delta_2=\begin{vmatrix}-\frac{1}{2} & \frac{1}{4}\\ \frac{1}{4} & -\frac{1}{2}\end{vmatrix}=\frac{1}{4}-\frac{1}{16}=\frac{3}{16}>0.$$

Следовательно, матрица C определена отрицательно.

Построение развёрнутой модели (2.110), соответствующей нашим условиям, даёт результат

$$f(x_1,x_2)=2\cdot x_1+3\cdot x_2+\frac{1}{4}\cdot x_1\cdot x_2-\frac{1}{4}\cdot x_1^2-\frac{1}{4}\cdot x_2^2\to\max$$

$$\begin{cases}2\cdot x_1+5\cdot x_2\le 10,\\ 3\cdot x_1+2\cdot x_2\le 6,\\ 1\cdot x_1+1\cdot x_2\le 4,\\ x_1\ge 0, x_2\ge 0,\end{cases}\quad\text{или}\quad\begin{cases}10-2\cdot x_1-5\cdot x_2\ge 0,\\ 6-3\cdot x_1-2\cdot x_2\ge 0,\\ 4-1\cdot x_1-1\cdot x_2\ge 0,\\ x_1\ge 0, x_2\ge 0.\end{cases}$$

Правая запись системы ограничений соответствует формальному виду, применяемому для представления ограничений в задачах вогнутого программирования.

Исследуем целевую функцию данной задачи на экстремум, найдём координаты стационарной точки.

$$\begin{cases} \dfrac{\partial f(x_1,x_2)}{\partial x_1} = 2 + \dfrac{1}{4}\cdot x_2 - \dfrac{1}{2}\cdot x_1 = 0, \\ \dfrac{\partial f(x_1,x_2)}{\partial x_2} = 3 + \dfrac{1}{4}\cdot x_1 - \dfrac{1}{2}\cdot x_2 = 0. \end{cases} \Rightarrow \begin{cases} x_1 = 2\left(2 + \dfrac{1}{4}\cdot x_2\right), \\ 3 + \dfrac{1}{2}\left(2 + \dfrac{1}{4}\cdot x_2\right) - \dfrac{1}{2}\cdot x_2 = 0. \end{cases} \Rightarrow X^* = \begin{pmatrix} \dfrac{28}{3} \\ \dfrac{32}{4} \end{pmatrix}$$

Полученная точка, в которой наблюдается безусловный глобальный максимум, лежит за пределами области ограничений, значение целевой функции в этой точке есть $f(X^*) = \dfrac{196}{9} \cong 21{,}7$.

По условиям задачи может быть составлена функция Лагранжа

$$L(X,\Lambda) = 2\cdot x_1 + 3\cdot x_2 + \frac{1}{4}\cdot x_1\cdot x_2 - \frac{1}{4}\cdot x_1^2 - \frac{1}{4}\cdot x_2^2 \quad +\lambda_1(10 - 2\cdot x_1 - 5\cdot x_2) + \\ +\lambda_2(6 - 3\cdot x_1 - 2\cdot x_2) + \\ +\lambda_3(4 - 1\cdot x_1 - 1\cdot x_2).$$

Применение подхода, связанного с поисками безусловного экстремума функции без ограничений (функции Лагранжа) в задачах вогнутого программирования, позволяет получить систему ограничений для поиска значений X и Λ, являющихся координатами экстремальной точки.

$$\begin{cases} \dfrac{\partial L(X,\Lambda)}{\partial x_1} = 2 + \dfrac{1}{4}\cdot x_2 - \dfrac{1}{2}\cdot x_1 - 2\cdot\lambda_1 - 3\cdot\lambda_2 - 1\cdot\lambda_3 \le 0, \\ \dfrac{\partial L(X,\Lambda)}{\partial x_1} = 3 + \dfrac{1}{4}\cdot x_1 - \dfrac{1}{2}\cdot x_2 - 5\cdot\lambda_1 - 2\cdot\lambda_2 - 1\cdot\lambda_3 \le 0, \\ \dfrac{\partial L(X,\Lambda)}{\partial \lambda_1} = 10 - 2\cdot x_1 - 5\cdot x_2 \ge 0, \\ \dfrac{\partial L(X,\Lambda)}{\partial \lambda_2} = 6 - 3\cdot x_1 - 2\cdot x_2 \ge 0, \\ \dfrac{\partial L(X,\Lambda)}{\partial \lambda_3} = 4 - 1\cdot x_1 - 1\cdot x_2 \ge 0. \end{cases}$$

Для канонизации системы ограничений добавим с соответствующими знаками компоненты векторов V и W. Получим следующую каноническую форму системы ограничений:

$$\begin{cases} 2+\frac{1}{4}\cdot x_2-\frac{1}{2}\cdot x_1-2\cdot\lambda_1-3\cdot\lambda_2-1\cdot\lambda_3+1\cdot v_1=0, \\ 3+\frac{1}{4}\cdot x_1-\frac{1}{2}\cdot x_2-5\cdot\lambda_1-2\cdot\lambda_2-1\cdot\lambda_3+1\cdot v_2=0, \\ 10-2\cdot x_1-5\cdot x_2-1\cdot w_1=0, \\ 6-3\cdot x_1-2\cdot x_2-1\cdot w_2=0, \\ 4-1\cdot x_1-1\cdot x_2-1\cdot w_3=0, \\ x_1\ge 0, x_2\ge 0,\ \Lambda>0, V\ge 0, W\ge 0. \end{cases}$$

Сопоставив полученные выражения с выражениями (2.111, а, б) теоремы квадратичного программирования, отметим их полное соответствие.

Пункт № 1 алгоритма.

Преобразуем данную каноническую форму системы ограничений, так, чтобы в столбце свободных членов находились бы положительные элементы.

$$\begin{cases} \frac{1}{2}\cdot x_1-\frac{1}{4}\cdot x_2+2\cdot\lambda_1+3\cdot\lambda_2+1\cdot\lambda_3-1\cdot v_1=2, \\ -\frac{1}{4}\cdot x_1+\frac{1}{2}\cdot x_2+5\cdot\lambda_1+2\cdot\lambda_2+1\cdot\lambda_3-1\cdot v_2=3, \\ 2\cdot x_1+5\cdot x_2+1\cdot w_1=10, \\ 3\cdot x_1-2\cdot x_2+1\cdot w_2=6, \\ 1\cdot x_1+1\cdot x_2+1\cdot w_3=4, \\ x_1\ge 0, x_2\ge 0,\ \Lambda>0, V\ge 0, W\ge 0. \end{cases}$$

Пункты № 2,3 алгоритма.

Построим эквивалентную ЗЛП, введя псевдоцелевую функцию и модифицировав систему ограничений. Функция

$$\mu\cdot y_1+\mu\cdot y_2+\mu\cdot z_1+\mu\cdot z_2+\mu\cdot z_3\to\min$$

Наименование “псевдоцелевая” функция получила, поскольку составляется только из искусственных переменных, а оптимальное решение, в случае его существования, приведёт к обращению её в нуль.

Система ограничений после введения искусственных переменных примет вид.

$$\begin{cases} \frac{1}{2}\cdot x_1 - \frac{1}{4}\cdot x_2 + 2\cdot\lambda_1 + 3\cdot\lambda_2 + 1\cdot\lambda_3 - 1\cdot v_1 + y_1 = 2, \\ -\frac{1}{4}\cdot x_1 + \frac{1}{2}\cdot x_2 + 5\cdot\lambda_1 + 2\cdot\lambda_2 + 1\cdot\lambda_3 - 1\cdot v_2 + y_2 = 3, \\ 2\cdot x_1 + 5\cdot x_2 + 1\cdot w_1 + z_1 = 10, \\ 3\cdot x_1 - 2\cdot x_2 + 1\cdot w_2 + z_2 = 6, \\ 1\cdot x_1 + 1\cdot x_2 + 1\cdot w_3 + z_3 = 4, \\ x_1 \ge 0, x_2 \ge 0,\ \Lambda > 0, V \ge 0, W \ge 0,\ Y \ge 0, Z \ge 0. \end{cases}$$

Поскольку переменных *X*, *Λ,* V, W, Z много больше числа ограничений, в качестве алгоритма решения ЗЛП следует модифицированный симплекс-метод.

Пункт № 4 алгоритма.

Подробный ход расчётов опускается, приводится конечный результат решения, опровергнуть или подтвердить который Читатель может самостоятельно, перерешав ЗЛП.

Координаты оптимума составляют:

$$x_1 = \frac{10}{11}, x_2 = \frac{18}{11}, \lambda_1 = \frac{103}{242}, \lambda_2 = \frac{109}{242}, \lambda_3 = 0, v_1 = 0, v_2 = 0, w_1 = 0, w_2 = 0, w_3 = \frac{16}{11}.$$

Элементы векторов *V* и *W* неотрицательны, среди элементов вектора *Λ* – есть положительные (не все равны нулю по Куну-Таккеру).

Так как $x_1 \cdot v_1 = 0,\ x_2 \cdot v_2 = 0,\ \lambda_1 \cdot w_1 = 0,\ \lambda_2 \cdot w_2 = 0,\ \lambda_3 \cdot w_3 = 0,$ то условие дополняющей нежёсткости выполнено, и решение задачи квадратичного программирования находится в крайней точке множества допустимых стратегий, значение функции равно $f\left(\frac{10}{11}, \frac{18}{11}\right) = \frac{753}{121} \cong 6{,}223.$

Вопросы для самоконтроля

1. Как выглядит в общем виде модель для задачи нелинейного программирования?
2. Какие существуют типы задачи НП, чем они различаются?
3. Как формулируются необходимые условия оптимальности в задачах безусловной оптимизации?

4. Как найти наибольшее и наименьшее значения функции нескольких переменных в замкнутой ограниченной области?
5. Что называется частной производной первого порядка функции нескольких переменных?
6. Как следует понимать термины «выпуклая» и «вогнутая» функции применительно к задачам нелинейного программирования?
7. Что называется линией уровня функции двух переменных?
8. Что называется поверхностью уровня функции трех переменных?
9. Что такое частная производная второго порядка функции нескольких пе-
10. ременных?
11. Что называется стационарной точкой?
12. Какие численные методы существуют для вычисления частных производных?
13. Что такое критическая точка функции нескольких переменных?
14. Обязана ли критическая точка быть точкой экстремума?
15. В чем состоит схема исследования функции нескольких переменных на экстремум?
16. Что такое условный экстремум функции нескольких переменных?
17. Какие задачи относят к выпуклому программированию, а какие – к вогнутому?
18. Где отмечаются минимальные и максимальные значения функций в НП-задачах?
19. Как используется матрица Гёссе при определении экстремумов целевой функции?
20. Для чего в задачах многомерной оптимизации применяются методы одномерной минимизации?
21. Какие методы поиска относятся к прямым, а какие – к градиентным?
22. Что такое градиент функции нескольких переменных?
23. Как определить составляющие вектора градиента?
24. Когда будет происходить наискорейший спуск, а когда – полношаговый?
25. Каковы основные свойства градиента функции в некой точке?
26. В чем заключается градиентные методы при поиске минимума функции?
27. Как определяется шаг поиска в градиентных методах?
28. Что такое сопряженные направления?
29. В чём состоят достоинства и недостатки метода Ньютона?
30. В чём предназначение квазиньютоновских методов?
31. Как сконструировать функцию Лагранжа?
32. Что представляют собой множители Лагранжа?
33. В чем заключается метод множителей Лагранжа?

34. В каких случаях необходимые условия оптимальности в задаче НП также являются и достаточными?
35. Какое ограничение НП-задачи называется активным?
36. Как формулируется теорема Куна-Таккера?
37. В чём заключается условие дополняющей нежёсткости?
38. Каковы особенности седловой точки функции многих переменных?
39. Как формулируется теорема о седловой точке?
40. Что вкладывается в понятие “возможное направление”?
41. Для чего строится проецирующая матрица в методе Зойтендейка?
42. Каковы общие идеи, положенные в основу методов штрафных функций?
43. Какие требования предъявляют к штрафным функционалам методов барьерных поверхностей?
44. Какие требования предъявляют к штрафным функционалам методов внешней точки?
45. За счёт чего происходит решение НП-задачи в методах внешней точки?
46. Как формулируется теорема квадратичного программирования?
47. Как по «внешнему» виду математической модели определить: относится ли она к задачам квадратичного программирования?
48. Что означает термин “симметричная” матрица?
49. Что означают термины “отрицательно” и “положительно” определённые матрицы?
50. Когда задача квадратичного программирования неразрешима?
51. Можно ли решить задачу квадратичного программирования для случая с функцией цели вида $f(x) = b^T X + \frac{1}{2} X^T C X \to \max$, когда C — симметричная, положительно определённая матрица, и, если Вы полагаете, что можно, поясните, каким образом?
52. Какие шаги предпринять, чтобы пользуясь изложенным выше методом решить НП-задачу с целевой функцией вида $f(x) = b^T X + \frac{1}{2} X^T C X$ на минимум?