ЧИСЛЕННЫЕ МЕТОДЫ ОПТИМИЗАЦИИ

Среди важнейших методов оптимизации широко распространены следующие методы.

1. Безусловной минимизации функций одной переменной:

— прямые методы (перебора, половинного деления, золотого сечения);

— методы, использующие производные первого и второго порядка (Ньютона, ломаных, касательных).

2. Безусловной минимизации функций многих переменных:

— градиентного спуска;

— наискорейшего спуска;

— сопряженных направлений.

3. Условной оптимизации:

— линейное и квадратичное программирование;

— нелинейное программирование;

— динамическое программирование.

Под *минимизацией* функции $f(x)$ на множестве $X \subset R$ будем понимать следующую задачу: найти хотя бы одну *точку минимума* x^* и *минимум* $f^* = f(x^*)$ на множестве X.

Задача нахождения точки максимума x^* функции $f(x)$ и максимального значения функции $f(x^*)$ сводится к задаче минимизации функции $-f(x)$, поэтому ниже будем рассматривать только задачу минимизации.

Число $x^* \in X$ называется *точкой абсолютного* (*глобального*) *минимума* функции $f(x)$, а значение $f(x^*)$ — *глобальным минимумом*, если $f(x^*) \leqslant f(x)$ для всех $x \in X$.

Число $\widetilde{x} \in X$ называется *точкой локального минимума*, если существует такое число $\delta > 0$, что выполняется неравенство $f(\widetilde{x}) \leqslant f(x)$ для всех точек x, удовлетворяющих условию $|x - \widetilde{x}| < \delta$; значение $f(\widetilde{x})$ называется *локальным минимумом*.

Будем рассматривать *унимодальные функции* $f(x)$ — функции, имеющие один минимум в области X.

Функция $f(x)$ называется унимодальной на отрезке $[a, b]$, если она непрерывна на $[a, b]$ и существуют числа α и β, $a \leqslant \alpha \leqslant \beta \leqslant b$, такие, что

1) на отрезке $[a, \alpha]$ $f(x)$ монотонно убывает;

2) на отрезке $[\beta, b]$ $f(x)$ монотонно возрастает;

3) на отрезке $x \in [\alpha, \beta]$ $f(x)$ имеет минимум $f^* = \min\limits_{x \in [\alpha, \beta]} f(x)$.

Существует два критерия унимодальности, используемые на практике:

1) если функция $f(x)$ дифференцируема на отрезке $[a, b]$ и производная $f'(x)$ *не убывает* на этом отрезке, то $f(x)$ *унимодальна.*

2) Если функция $f(x)$ дважды дифференцируема на отрезке $[a, b]$ и $f''(x) \geqslant 0$ на этом отрезке, то $f(x)$ *унимодальна.*

Подавляющее большинство численных методов оптимизации относится к классу *итерационных*, т.е. порождающих последовательность точек в соответствии с предписанным набором правил, включающим критерий окончания. При заданной начальной точке x^0 методы генерируют последовательность $x^0, x^1, x^2, \ldots$. Преобразование точки x^k в x^{k+1} представляет собой *итерацию*.

Для определенности рассмотрим задачу поиска безусловного локального минимума:

$$f(x^*) = \min_{x \in R^n} f(x). \qquad (1)$$

Численное решение задачи (1), как правило, связано с построением последовательности $\{x^k\}$ точек, обладающих свойством

$$f(x^{k+1}) < f(x^k), \qquad k = 0,1,\ldots\ . \qquad (2)$$

Общее правило построения последовательности $\{x^k\}$ имеет вид

$$x^{k+1} = x^k + t_k d^k, \qquad k = 0,1,\ldots\ , \qquad (3)$$

где точка x^0 - *начальная точка* поиска; d^k - приемлемое направление перехода из точки x^k в точку x^{k+1}, обеспечивающее выполнение условия (2) и называемое *направлением спуска*; t_k - *величина шага*.

Начальная точка поиска x^0 задается, исходя из физического содержания решаемой задачи и наличия априорной информации о положении точек экстремума.

Приемлемое направление спуска d^k должно удовлетворять условию

$$\left(\nabla f\left(x^k\right), d^k\right) < 0, \quad k = 0,1,\ldots, \tag{4}$$

обеспечивающему убывание функции $f(x)$. Примером приемлемого направления является направление вектора антиградиента: $d^k = -\nabla f\left(x^k\right)$.

Величина шага $t_k > 0$ выбирается либо из условия (2), либо из условия минимума функции вдоль направления спуска:

$$f\left(x^k + t_k\, d^k\right) \to \min_{t_k}. \tag{5}$$

Выбор шага t_k из условия (5) делает спуск в направлении d^k *наискорейшим.*

Определение (1). Последовательность $\left\{x^k\right\}$ называется *минимизирующей,* если $\lim\limits_{k\to\infty} f\left(x^k\right) = f^*$, т.е. последовательность сходится к нижней грани $f^* = \inf\limits_{x\in R^n} f(x)$.

Определение (2). Последовательность $\left\{x^k\right\}$ называется *сходящейся к точке минимума* x^*, если $\left\|x^k - x^*\right\| \to 0$ при $k \to \infty$.

Сходимость последовательности $\left\{x^k\right\}$ при выборе приемлемого направления d^k и величины шага t_k из условия (2) или (5) зависит от характера функции $f(x)$ и от выбора начальной точки x^0.

В зависимости от наивысшего порядка частных производных функции $f(x)$, используемых для формирования d^k и t_k, численные методы решения задачи безусловной минимизации **(1)** принято делить на три группы.

1. ***Методы нулевого порядка***, использующие только информацию о значении функции $f(x)$.

2. ***Методы первого порядка***, использующие информацию о первых производных функции $f(x)$.

3. ***Методы второго порядка***, требующие для своей реализации знания вторых производных функции $f(x)$.

Работоспособность метода еще не гарантирована доказательством сходимости соответствующей последовательности - нужна определенная скорость сходимости.

Задачу называют **детерминированной**, если погрешностью вычисления (или экспериментального определения) функции *f(x)* можно пренебречь. В противном случае задачу называют **стохастической**. Все изложенные далее методы применимы только к детерминированным задачам.

Итак, приступая к поиску минимума функции, необходимо определить интервал, на котором функция могла бы иметь минимум.
Для этого можно использовать:
1) графическое представление функции;
2) аналитический анализ аппроксимирующей функции;
3) сведения о математической модели исследуемого процесса (т.е. законы Поведения данной функции).

Методы поиска минимума по нахождению корней уравнений

Если функция $f(x)$ аналитически дифференцируема, то решаем уравнение $f^{/}(x) = 0$. При этом условие $f^{//}(x) > 0$ в найденной точке указывает нам на минимум. Для использования этих методов необходимо знать либо аналитический вид первой и второй производных, либо рассчитать их численно, если это не приведет к потере точности.

Метод перебора. Метод перебора является простейшим методом из прямых методов минимизации, но он является и самым надежным методом в смысле гарантированного нахождения минимального значения.

Пусть $f(x)$ унимодальна на $x \in [a, b]$. Требуется найти точку минимума x^* и минимум $f(x^*)$ с точностью $\varepsilon > 0$.

Отрезок $[a, b]$ разбивается точками $x_i = a + i \cdot \dfrac{b-a}{n}$, $i = \overline{0, n}$, на n равных частей, где $n \geqslant \dfrac{b-a}{\varepsilon}$ (в простейшем случае $n = \dfrac{b-a}{\varepsilon}$ или $\dfrac{b-a}{n} = \varepsilon$).

Вычислив значения $f(x_i)$, $i = \overline{0, n}$, путем сравнения находим

$$f(x_m) \approx \min f(x_i); \quad x^* \approx x_m; \quad f^* \approx f(x_m).$$

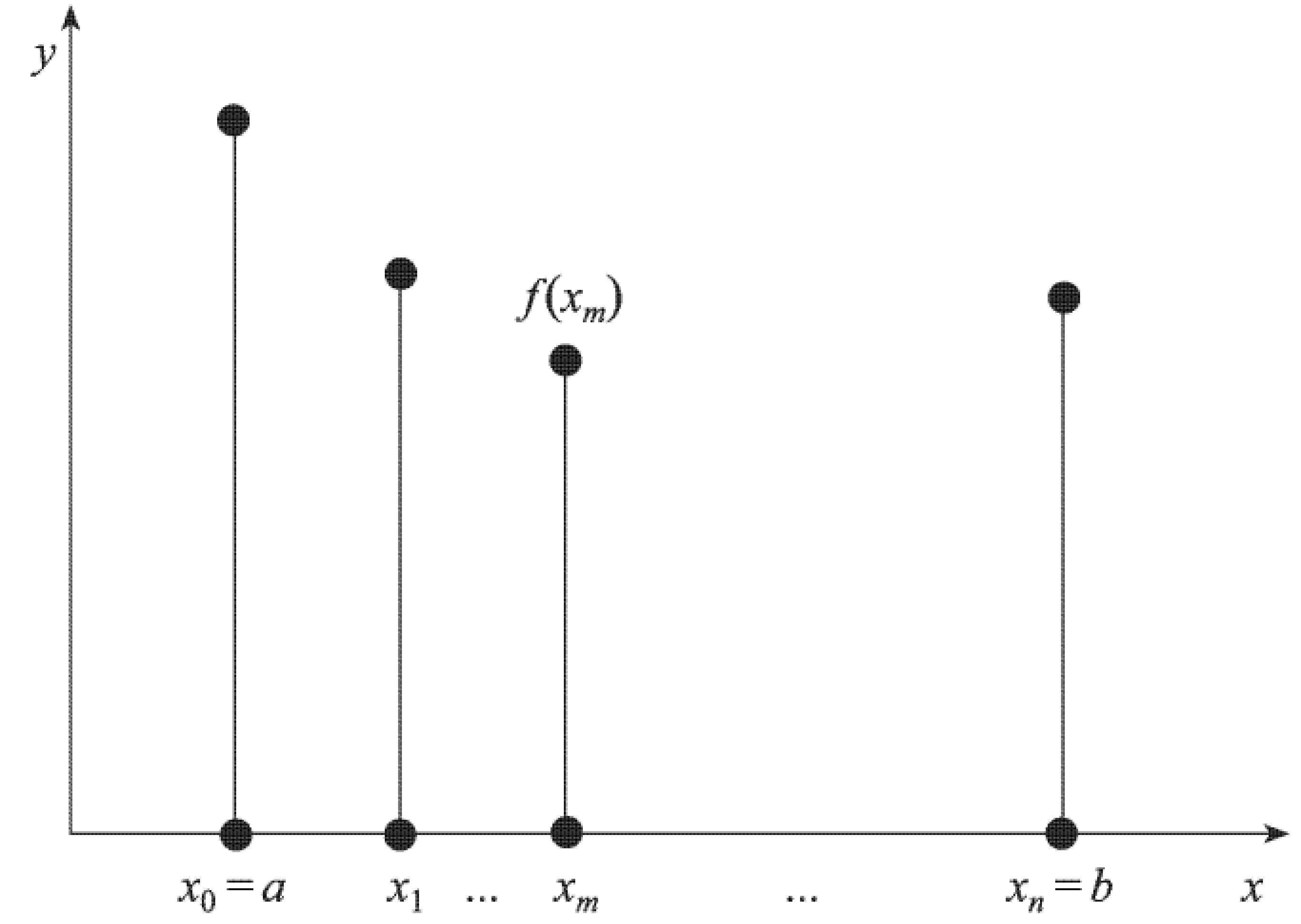
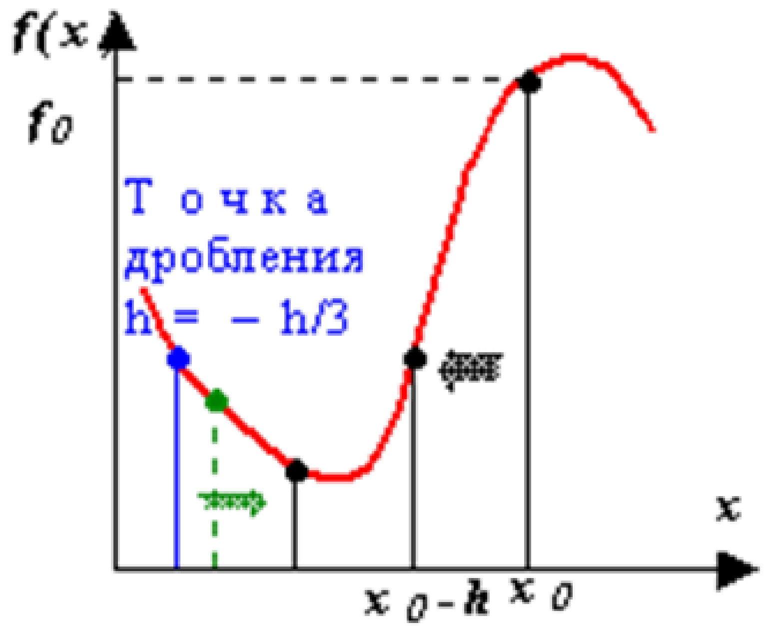


Рис. К методу перебора

Ясно, что значения x^* и $f(x^*)$ являются приближенными. Поэтому для вычисления минимума с большей точностью ε_1 ($\varepsilon_1 < \varepsilon$) рассматривается отрезок $x \in [x_{m-1}, x_{m+1}]$, где находится минимальное значение $f(x^*)$, на нем вычисляются значения $f(x_j)$, $j = \overline{0, k}$, $k = (x_{m+1} - x_{m-1})/\varepsilon_1$, и путем сравнения $f(x_j)$ находим более точное значение минимума функции.

Достоинства метода — простота и надежность, недостаток — большое число вычислений значений функции $f(x)$.

Метод дробления



Пусть дана начальная точка x_0, а также величина и знак шага h, определяющие движение из этой точки в сторону предполагаемого минимума $f(x)$. Метод заключается в последовательном дроблении исходного шага h с изменением его знака при выполнении условия $f(x_{k+1}) > f(x_k)$, где k – порядковый номер вычисляемой точки. Например, как только очередное значение функции стало больше предыдущего, выполняется $h = -h/3$ и процесс продолжается до тех пор, пока

$$|x_{k+1} - x_k| \le \varepsilon \,. \qquad (6)$$

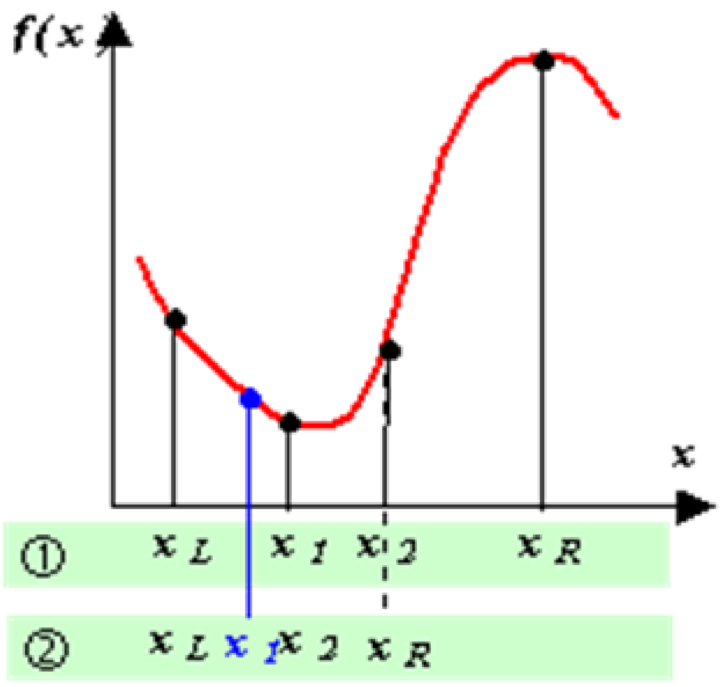
Данный метод является одним из самых медленных для поиска минимума, однако он использует значительно меньшее количество вычислений функции $f(x)$, чем метод перебора. Основное достоинство данного алгоритма – возможность использования в программах управления экспериментальными исследованиями, когда значения функции $f(x)$ последовательно измеряются с шагом $h \ge h_{\min}$.

Метод золотого сечения

Пусть $f(x)$ задана и кусочно-непрерывна на $[x_L, x_R]$, и имеет на этом отрезке только один локальный минимум. Золотое сечение, о котором упоминал ещё Евклид, состоит в разбиении интервала $[x_L, x_R]$ точкой x_1 на две части таким образом, что отношение длины всего отрезка к его большей части равно отношению большей части к меньшей:

$$\frac{x_R - x_L}{x_R - x_1} = \frac{x_R - x_1}{x_1 - x_L} \quad (7)$$

Таким образом, возьмем на отрезке две точки x_1 и x_2, симметрично относительно границ делящие исходный отрезок в отношении золотого сечения:



$$x_1 = x_L + (1-\tau)\cdot(x_R - x_L),$$

$$x_2 = x_L + \tau(x_R - x_L),$$

где коэффициент $\tau = \dfrac{\sqrt{5}-1}{2} \approx 0.618034$.

Если $f(x)_1 < f(x)_2$, мы должны сузить отрезок справа,
т.е. новое значение $x_R = x_2$, в противном случае $x_L = x_1$. Оставшаяся внутри нового отрезка точка является первым приближением к минимуму и делит этот отрезок в отношении золотого сечения. Таким образом, на каждой итерации приближения к минимуму (см. рисунок) нам нужно ставить только одну точку (x_1 или x_2), в которой считать значение функции и сравнивать его с предыдущим. Условием выхода из итерационного процесса будет, подобно предыдущему случаю, условие $|x_2 - x_1| \leq \xi$.

Метод не отличается высокой скоростью сходимости, однако, всегда сходится, прост и экономичен. Если на заданном интервале несколько локальных минимумов, то один из них непременно будет найден, хотя не обязательно - наименьший. Данный метод применим в том числе и к недифференцируемым функциям.

Программа Data_sheet1.m иллюстрирует работу метода золотого сечения для поиска локальных экстремумов кусочно-дифференцируемой функции. За счёт незначительного изменения положения левой границы исходного отрезка удаётся последовательно получить все три локальных минимума функции.

Погрешность метода золотого сечения оценивается по формуле

$$|x^* - \overline{x}| \leqslant \left(\frac{\sqrt{5}-1}{2}\right)^k (b-a)\,,$$

и если задана точность ε, то, поскольку погрешность не превышает точности, из неравенства

$$\left(\frac{\sqrt{5}-1}{2}\right)^k (b-a) \leqslant \varepsilon$$

до начала процесса вычислений находим нижнюю оценку числа шагов k:

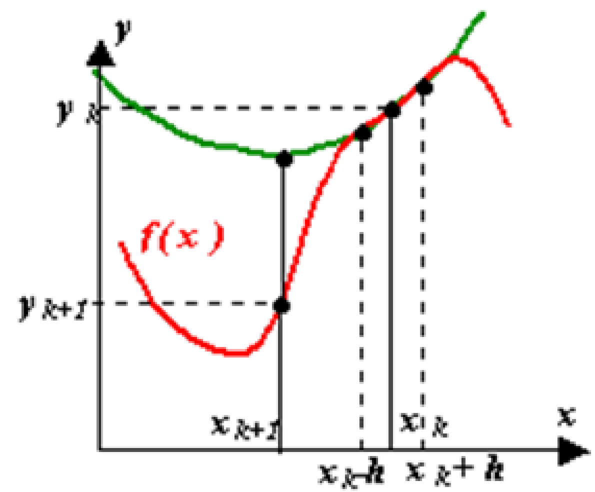
$$k \geqslant \ln\frac{\varepsilon}{b-a} \Big/ \ln\left(\frac{\sqrt{5}-1}{2}\right) \approx -2{,}1 \cdot \ln\frac{\varepsilon}{b-a}.$$

Метод парабол

Пусть *f(x)* имеет первую и вторую производную. Разложим *f(x)* в ряд Тейлора в некоторой точке x_k, ограничиваясь при этом тремя первыми членами разложения:

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2 \qquad (8)$$

Иными словами, аппроксимируем нашу функцию в точке x_k параболой. Для этой параболы можно аналитически вычислить положение экстремума как корень уравнения первой производной $f'(x) = 0$ функции, представленной ур.(8), а именно: $f'(x_k) + (x - x_k) f''(x_k) = 0$. Пусть минимум аппроксимирующей параболы находится в точке x_{k+1}. Тогда вычислив значение функции $f(x_{k+1})$, мы получаем новую точку приближения к минимуму.



Обычно в практических реализациях данного метода не используют аналитический вид первой и второй производных *f(x)*. Их заменяют конечно-разностными аппроксимациями. Наиболее часто берут симметричные разности с постоянным шагом *h*:

$$f'(x_k) \approx \frac{1}{2h}\left(f(x_k + h) - f(x_k - h)\right) \qquad f''(x_k) \approx \frac{1}{h^2}\left(f(x_k + h) - 2f(x_k) + f(x_k - h)\right)$$

Это эквивалентно аппроксимации функции параболой, проходящей через три близкие точки x_k+h, x_k, x_k-h. Окончательное выражение, по которому можно строить итерационный процесс, таково:

$$x_{k+1} = x_k - \frac{h}{2} \cdot \frac{f(x_k+h) - f(x_k-h)}{f(x_k+h) - 2f(x_k) + f(x_k-h)} \quad (9)$$

Рассмотрим, как оно получено. Запишем интерполяционную параболу: $y=a + bx + cx^2$. Для нахождения неизвестных коэффициентов a, b, c составим систему уравнений:

$$\begin{cases} a + b(x_k - h) + c(x_k - h)^2 = f(x_k - h) \\ a + bx_k + cx_k^2 = f(x_k) \\ a + b(x_k + h) + c(x_k + h)^2 = f(x_k + h) \end{cases} \quad (10)$$

Решив эту систему уравнений, найдём неизвестные коэффициенты, а затем, подставив их в уравнение параболы, найдём её экстремум. Это и есть уравнение (9). Решение системы (10) и нахождение её экстремума (9) реализовано в программе Data_sheet2.m: xps=x_{k+1}, Fsm=f(x_k-h), Fs=f(x_k), Fsp=f(x_k +h).

Данный метод отличается от вышеизложенных высокой скоростью сходимости. Вблизи экстремума, вплоть до расстояний $\sim h^2$, сходимость практически не отличается от квадратичной. Однако алгоритм требует постоянного контроля сходимости. Итерационный процесс будет сходиться к минимуму, если

1. знаменатель формулы (9) должен быть >0. Если это не так, нужно сделать шаг в обратном направлении, причем достаточно большой. Обычно в итерационном процессе полагают $h << |x_{k+1} - x_k|$.

   Иногда ради упрощения расчетов полагают $h = |x_{k+1} - x_k|$, однако это существенно уменьшает скорость сходимости.

2. $f(x_{k+1}) < f(x_k)$. Если это не так, то от x_k следует сделать шаг $\tau(x_{k+1} - x_k)$ с $\tau = ½$. Если и при этом условие убывания не выполнено, уменьшают τ и вновь делают шаг.

Программа Data_sheet3.m иллюстрирует работу метода парабол. Показана очень быстрая сходимость метода к одному из локальных экстремумов $x_k = \pm\sqrt{k}$, k=0,1,2,…, функции $F(x) = -\cos(\pi x^2)$. График иллюстрирует довольно сложную зависимость между начальным приближением и выходом на то или иное экстремальное значение. Был рассмотрен отрезок [-1,6; 1,6], при этом найдено 13 локальных минимумов: 0, ±1, $\pm\sqrt{2}$, $\pm\sqrt{3}$, $\pm\sqrt{5}$, $\pm\sqrt{6}$, $\pm\sqrt{45}$.

Методы минимизации, использующие производные. Метод Ньютона

Метод Ньютона, как метод с квадратичной скоростью сходимости, используется на завершающем этапе какого-либо более грубого метода.

Он использует производные 1-го и 2-го порядков, что обеспечивает очень быструю сходимость.

Множество точек $X \subset R$ называется *выпуклым*, если отрезок $[x_1, x_2]$, соединяющий любые две точки x_1, $x_2 \in X$, принадлежит множеству X, т. е. точки $\alpha\, x_1 + (1-\alpha)x_2 \in X$, $0 < \alpha < 1$.

Функция $f(x)$ называется *выпуклой* на множестве точек $x \in \in X$, если для двух точек x_1, $x_2 \in X$ выполняется неравенство

$$f(\alpha\, x_1 + (1-\alpha)x_2) \leqslant \alpha\, f(x_1) + (1-\alpha) f(x_2), \quad 0 < \alpha < 1.$$

Будем рассматривать *выпуклые функции* $f(x)$ на отрезке $x \in \in [a, b]$.

Для того чтобы *дважды дифференцируемая* функция была *выпуклой* на отрезке $x \in [a, b]$, необходимо и достаточно, чтобы выполнялось неравенство $f''(x) \geqslant 0$ для всех точек.

Для выпуклой дважды дифференцируемой функции $f(x)$ на $x \in [a, b]$ *метод Ньютона* заключается в построении следующей итерационной последовательности (сравнить с методом Ньютона уточнения корней нелинейных уравнений):

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}, \qquad k = 0, 1, 2 \ldots$$

Отсюда видно, что производная второго порядка $f''(x)$ на отрезке $x \in [a, b]$ не должна быть равна нулю. Для выпуклых функций $f(x)$ это действительно так, в противном случае точки, в которых $f''(x) = 0$, являлись бы точками перегиба функции, а в этих точках минимум (или максимум) функции $f(x)$ отсутствует.

Если задана точность ε, то останов осуществляется при выполнении условия

$$f'(x_k) \leqslant \varepsilon,$$

т. е. когда касательная к графику функции в точке $(x_k,\ f(x_k))$ почти горизонтальна. Тогда

$$x^* \approx x_k \ \text{ и } \ f(x^*) \approx f(x_k).$$

Можно показать, что верхняя оценка погрешности в методе Ньютона представима в виде следующего неравенства:

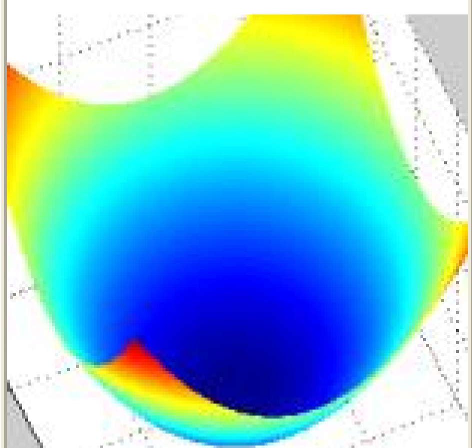
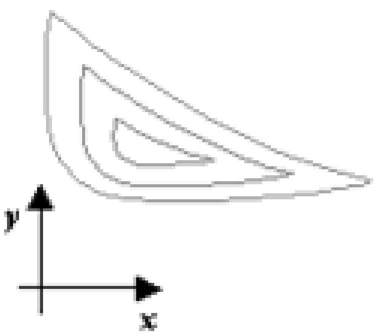
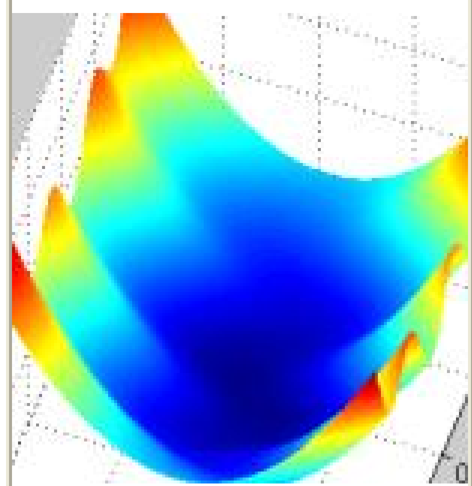
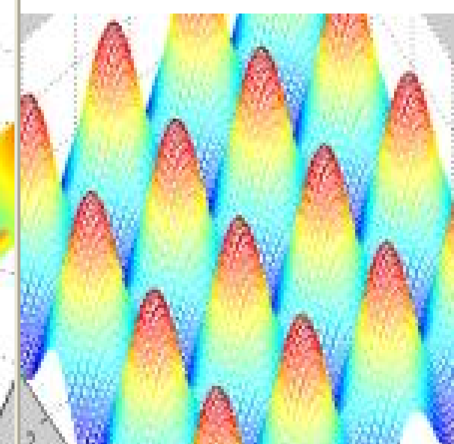
$$|x^* - x_k| \leqslant \frac{2m_2}{M_1} q^{2^k}, \quad q = \frac{M_1}{2m_2} \left| f'(x_0) \right|,$$

$$k = 0,\ 1,\ 2,\ \dots\ ; \quad M_1 = \max_{x \in [a,\, b]} \left| f'(x) \right|; \quad m_2 = \min_{x \in [a, b]} \left| f''(x) \right|,$$

причем для сходимости метода Ньютона достаточно, чтобы начальное приближение x_0 удовлетворяло условию $q < 1$.

Численные методы поиска минимума функции нескольких переменных

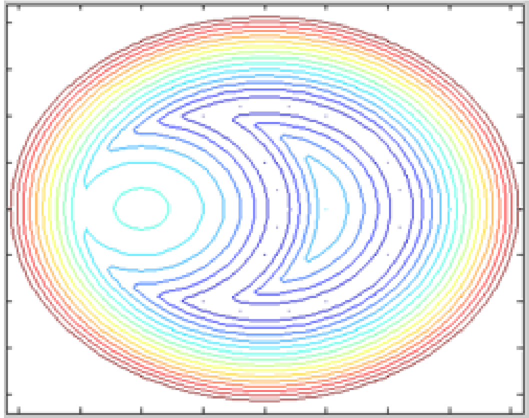
Будем рассматривать методы поиска минимума $\inf\left(f(\vec{r})\right)$ в многомерных задачах на примере функции двух переменных $f(x, y)$, так как эти методы легко аппроксимировать на случай трех и более измерений. Все эффективные методы поиска минимума сводятся к построению траекторий, вдоль которых функция убывает. Разные методы отличаются способами построения таких траекторий, так как метод, приспособленный к одному типу рельефа, может оказаться плохим для рельефа другого типа. Различают следующие типы рельефа:

1) Котловинный (гладкая функция)	2) Истинный овраг	3) Разрешимый овраг	4) Неупорядоченный
			

Программа Data_sheet4.m строит линии уровня поверхностей 4-х указанных типов.

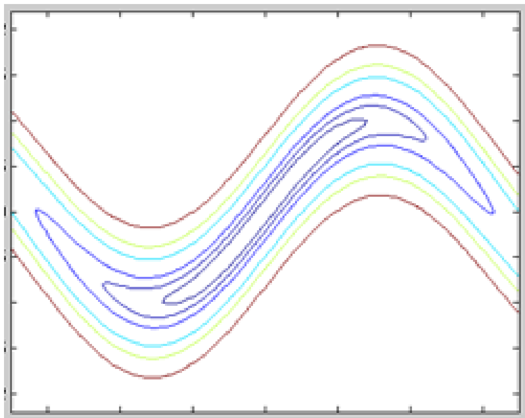
Программа Data_sheet4.m строит линии уровня поверхностей 4-х указанных типов.

При котловинном рельефе линии уровня похожи на эллипсы.



а

При овражном типе рельефа рассмотрим 2 случая. Если линии уровня кусочно-гладкие, как на рис.а, то выделим на них точки излома. Совокупность точек излома назовём **истинным оврагом** или **гребнем** в зависимости от того, растёт или падает значение функции в направлении угла между линией излома и линией уровня.



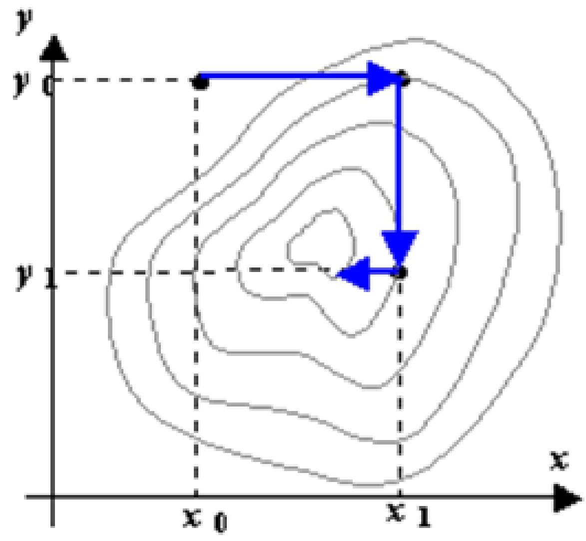
б

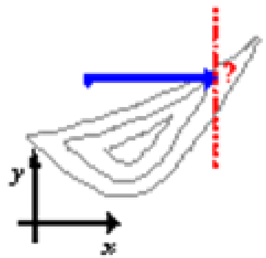
Если функция гладкая, то и линии уровня являются гладкими, однако на них могут быть участки с большой кривизной. Геометрическое место точек с максимальной кривизной называют **разрешимыми оврагами** или **гребнями**. На рис.б приведён пример длинного разрешимого оврага, дно которого имеет форму синуса. Наличие овражного рельефа указывает на то, что в модели не учтена некая скрытая связь между переменными.

Так, если в примере функции рис.б ввести замену переменных $\xi = x, \ \ \eta = y - \sin x$, то функция F предстанет в виде $F = 10\eta^2 + 0{,}1\xi^2$, и уже после замены переменных рельеф станет котловинным.

Метод покоординатного спуска

Пусть требуется найти минимум $f(x, y)$. Выберем нулевое приближение (x_0, y_0). Рассмотрим функцию одной переменной, зафиксировав вторую, $f(x, y_0)$ и найдем ее минимум, используя любой из рассмотренных выше способов. Пусть этот минимум оказался в точке (x_1, y_0). Теперь точно так же будем искать минимум функции другой переменной $f(x_1, y)$. Этот минимум окажется в точке (x_1, y_1). Одна итерация спусков завершена. Будем повторять циклы, постепенно приближаясь ко дну котловины, пока не выполнится условие $\max\left|\vec{r}_{k+1} - \vec{r}_k\right| < \xi$





Сходимость метода зависит от вида функции и выбора нулевого приближения. Вблизи невырожденного минимума гладкой функции спуск по координатам линейно сходится к минимуму. Если линии уровня образуют истинный овраг, возможен случай, когда спуск по одной координате приводит на дно оврага, а любое движение по следующей координате ведет на подъем. Процесс координатного спуска в данном случае не сходится к минимуму.

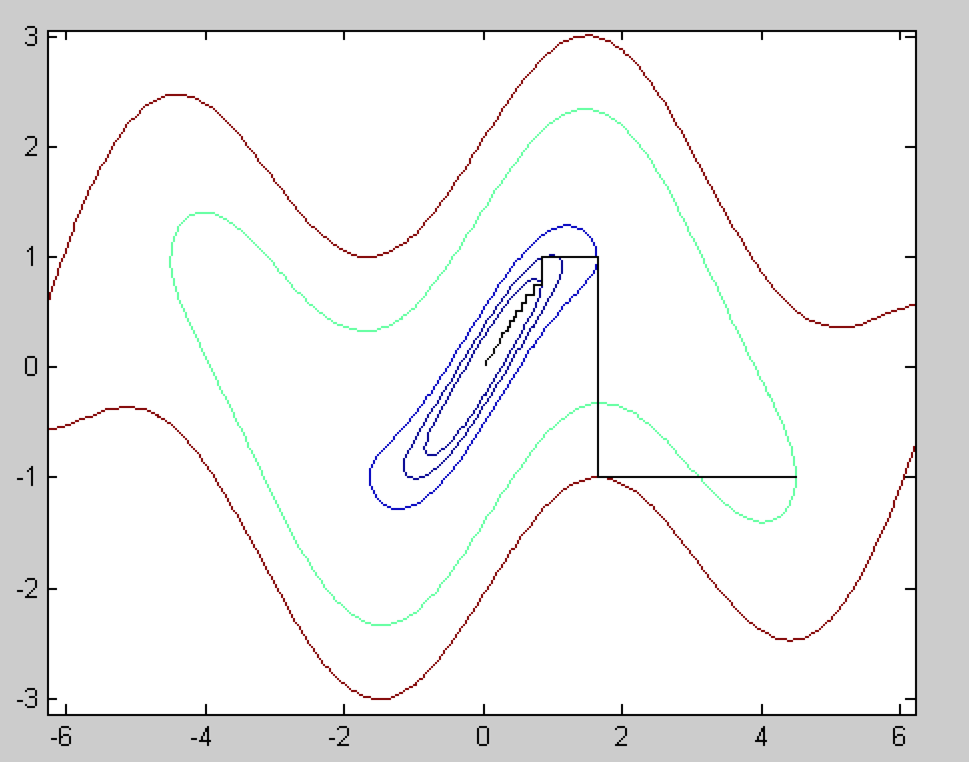
При попадании траектории спуска в разрешимый овраг сходимость становится чрезвычайно медленной. В физических задачах овражный рельеф указывает на то, что не учтена какая-то закономерность, определяющая связь между переменными в математической модели. Явный учет этой закономерности облегчает использование численных методов. Программа Data_sheet5.m иллюстрирует метод покоординатного спуска.

Data_sheet5.m

```
%Программа, иллюстрирующая поиск минимума
%функции F(x,y)=(y-sin(x))^2+0.1x^2
%методом покоординатного спуска
%очищаем рабочее пространство
clear all
%задаем точность близости частных производных
%функции F к нулю
eps=1e-3;
%определяем функцию и ее частные производные
F=@(x,y)(y-sin(x))^2+0.1*x^2;
Fx=@(x,y)-2*cos(x)*(y-sin(x))+0.2*x;
Fxx=@(x,y)2*y*sin(x)+2*cos(2*x)+0.2;
Fy=@(x,y)2*(y-sin(x));
Fyy=@(x,y)2;
%задаем начальное приближение
x(1)=4.5; y(1)=-1;
%задаем счетчик числа шагов в методе спуска и
%максимальное число шагов
k=1; iterm=200;
```

```
%организуем цикл покоординатного спуска
while ((abs(Fx(x(k),y(k)))>eps)|...
     (abs(Fy(x(k),y(k)))>eps))&(k<iterm)
   %цикл спуска по координате x осуществим с
   %помощью метода Ньютона
   xs=x(k);
   for i=1:2
      xs=xs-Fx(xs,y(k))/Fxx(xs,y(k));
   end
   k=k+1;
   x(k)=xs; y(k)=y(k-1);
   %цикл спуска по координате y осуществим с
   %помощью метода Ньютона
   ys=y(k);
   for i=1:2
      ys=ys-Fy(x(k),ys)/Fyy(x(k),ys);
   end
   k=k+1;
   x(k)=x(k-1); y(k)=ys;
end
```

```
%подготовительные мероприятия к построению
%линий уровня
[u v]=meshgrid(-2*pi:0.1:2*pi,-pi:0.1:pi);
Func=(v-sin(u)).^2+0.1*u.^2;
%определяем значения функции, линии уровня
%которых будут построены
for i=1:5
   s(i)=F(x(i),y(i));
end
%построение линий уровня
contour(u,v,Func,s);
hold on
%построение траектории спуска к минимуму
%функции F
line(x,y,'Color','black');
```

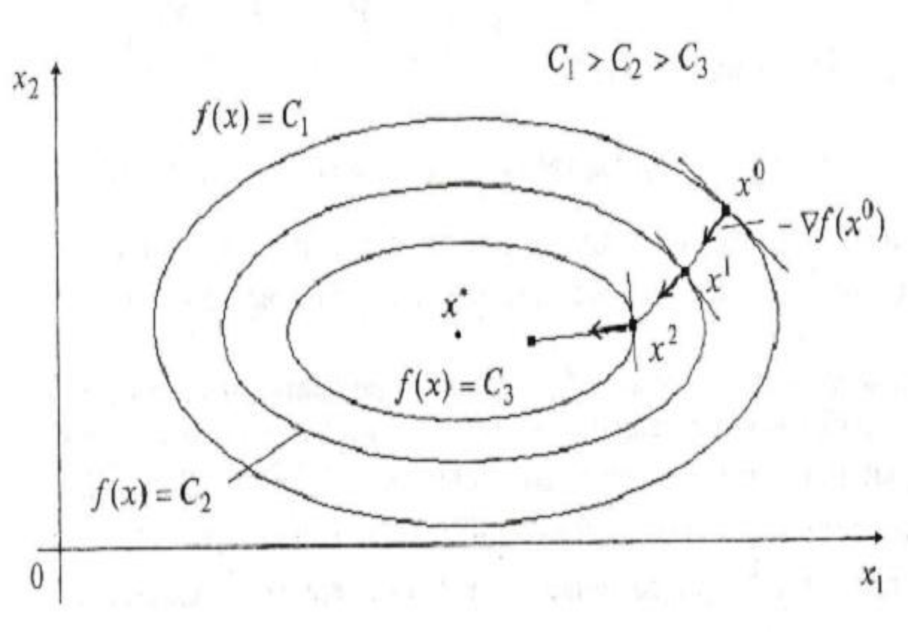


Метод градиентного спуска

Алгоритм

1. Задать – предельное число итераций. Найти градиент функции в произвольной точке

$$\nabla f(x) = \left\{\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n}\right\}^T$$



2. Положить *k = 0.*

3. Вычислить $\nabla f(x^k)$

4. Проверить выполнение условия окончания $\left\|\nabla f(x^k)\right\| < \varepsilon_1$

а) Если критерий выполнен, расчёт закончен $x^* = x^k$

б) Если критерий не выполнен, то перейти к шагу 5.

5. Проверить выполнение неравенства $k \geq M$

а) Если неравенство выполнено, расчёт закончен $x^* = x^k$

б) Если нет, то перейти к шагу 6.

Шаг 6. Задать величину шага t_k

Шаг 7. Вычислить $x^{k+1} = x^k - t_k \nabla f(x^k)$

Шаг 8. Проверить выполнение условия $f(x^{k+1}) - f(x^k) < 0$

$$(\textit{или } f(x^{k+1}) - f(x^k) < -\varepsilon \left\| \nabla f(x^k) \right\|^2)$$

а) Если условие выполнено, то перейти к шагу 9.

б) Если условие не выполнено, то положить $t_k = \frac{t_k}{2}$ и перейти к шагу 7.

Шаг 9. Проверить выполнение условий $\left\| x^{k+1} - x^k \right\| < \varepsilon_2$ и

$$\left| f(x^{k+1}) - f(x^k) \right| < \varepsilon_2$$

а) Если оба условия выполнены при текущем значении k и $k = k\text{-}1$, то расчёт окончен $x^* = x^{k+1}$

б) Если хотя бы одно из условий не выполнено, положить $k = k+1$ и перейти к шагу 3.

С помощью метода градиентного спуска минимум гладких функций в общем случае находится быстрее, чем при использовании координатного спуска. Однако нахождение градиента численными методами может свести на нет полученный выигрыш.

Сходимость плохая для функций с овражным рельефом, т.е. с точки зрения сходимости градиентный спуск не лучше спуска по координатам.

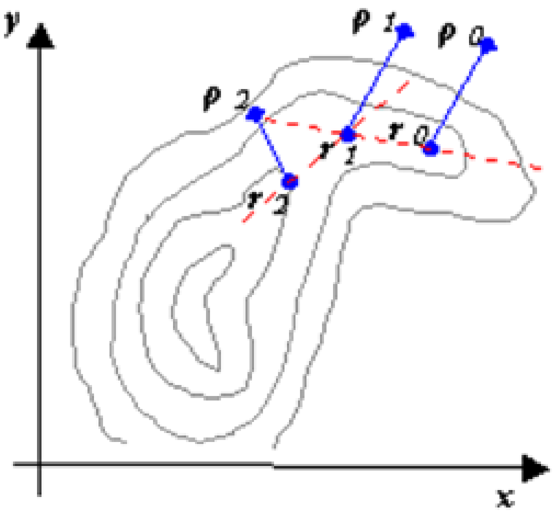
Каждый спуск заканчивается в точке, где линия градиента касательна к линии (поверхности) уровня. Это означает, что каждый следующий спуск должен быть перпендикулярен предыдущему. Таким образом, вместо поиска градиента в каждой новой точке можно сосчитать градиент в начальной точке, и *развернуть оси координат* так, чтобы одна их осей была параллельна градиенту, а затем осуществлять спуск координатным методом.

Программа Data_sheet6.m иллюстрирует градиентный метод.

В условиях неупорядоченного рельефа все методы спуска не дают способа поиска глобального минимума, т.к. из данного начального приближения сходятся к одному-единственному локальному минимуму. В этих условиях эффективен **метод случайного поиска**.

Метод оврагов

Ставится задача найти минимум $f(\vec{r})$ для овражной функции. Для этого выбираются две близкие точки $\vec{\rho}_0$ и $\vec{\rho}_1$, и осуществляется спуск из этих точек (любым методом), причем высокой точности сходимости не требуется. Конечные точки спуска $\vec{r}_0$ и $\vec{r}_1$ будут лежать вблизи дна оврага. Затем осуществляется движение вдоль прямой, соединяющей $\vec{r}_0$ и $\vec{r}_1$ в сторону уменьшения $f(\vec{r})$ (как бы вблизи дна оврага). Движение может быть осуществлено только на один шаг ~ h, направление выбирается из сравнения значения функции в точках $\vec{r}_0$ и $\vec{r}_1$. Таким образом, находится новая точка $\vec{\rho}_2 = \vec{r}_1 \pm (\vec{r}_1 - \vec{r}_0)h$. Так как возможно, что точка $\vec{\rho}_2$ уже лежит на склоне оврага, а не на дне, то из нее снова осуществляется спуск в новую точку $\vec{r}_2$. Затем намечается новый путь по дну оврага вдоль прямой, соединяющей $\vec{r}_2$ и $\vec{r}_1$. Если $f(\vec{r}_{k+1}) > f(\vec{r}_k)$ – процесс прекращается, а в качестве минимума в данном овраге используется значение $f(\vec{r}_k)$.



Метод оврагов рассчитан на то, чтобы пройти вдоль оврага и выйти в котловину около минимума. В этой котловине значения минимума лучше уточнять другими методами.

Проблемы поиска минимума в задачах с большим числом измерений

Пусть в n-мерном векторном пространстве задана скалярная функция $f(\vec{r})$. Наложим дополнительные условия $\varphi_i(\vec{r}) = 0$, $1 \le i \le m$; $\psi_j(\vec{r}) \ge 0$, $1 \le j \le p$. Условия типа равенств выделяют в пространстве некоторую $(n - m)$-мерную поверхность, а условия типа неравенств выделяют n-мерную область, ограниченную гиперповерхностями $\psi_j(\vec{r}) = 0$. Число таких условий может быть произвольным. Следовательно, задача **inf** $f(\vec{r})$ есть поиск минимума функции n переменных в некоторой $(n - m)$-мерной области E. Функция может достигать минимального значения как внутри области, так и на ее границе. Однако перейти к $(n - m)$-мерной системе координат практически никогда не удается, поэтому спуск приходится вести во всем n-мерном пространстве.

Даже если нулевое приближение лежит в области E, естественная траектория спуска сразу выходит из этой области. Для принудительного возврата процесса в область E, например, используется *метод штрафных функций*: к $f(\vec{r})$ прибавляются члены, равные нулю в E, и возрастающие при нарушении дополнительных условий (ограничений). Метод прост и универсален, однако считается недостаточно надежным.

Поиск минимума функций в MATLAB

В MATLAB поиск минимума функции одной переменной осуществляет функция: **[x, y]= fminbnd(name, a, b [, options])** для которой:

name - имя М-функции, вычисляющей значение f(x);

a, b - границы интервала, на котором осуществляется поиск минимума;

options - параметры, управляющие ходом решения;

x, y - координаты точки, в которой достигается минимум функции на заданном интервале. Функция f(x) может не являться унимодальной, тогда **fminbnd** найдёт один из локальных минимумов и не выдаст никаких сообщений о других экстремумах. Минимизируемая функция может быть негладкой и даже разрывной.

[x,fval,exitflag] = fminbnd(...)

[x,fval,exitflag,output] = fminbnd(...)

exitflag – условия прерывания процесса поиска;

output – информация об оптимизации.

```
[x,fval,exitflag] = fminbnd(@cos,3,4,optimset('TolX',1e-12,
'Display','off'))
```

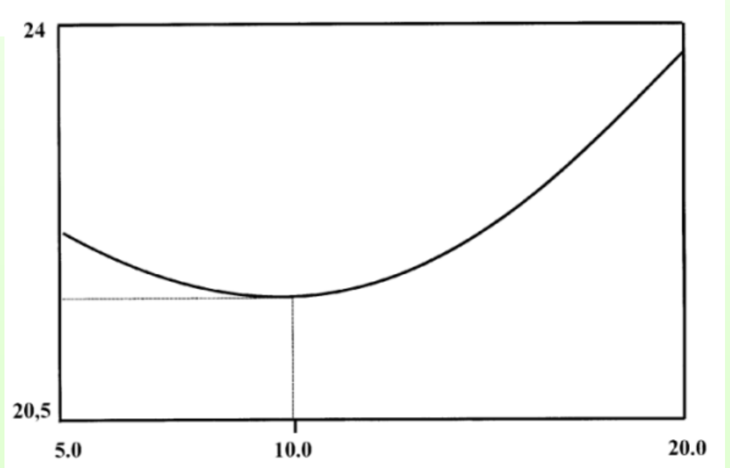
Функцию **fminbnd** можно использовать и для вычисления максимума. Для этого достаточно взять функцию name с противоположным знаком.

Пример: найти минимум функции *f(x),* на заданном интервале

$f(x) = 24 - 2x/3 + x^2/30$ на [5; 20].

Строим график этой функции, чтобы убедиться в наличии минимума на заданном интервале.

```
>> x = 5.0 : 0.001 : 20.0 ;    y = 24 – 2* x/3 + x.^2/30 ;
>> plot(x, y) ; grid on
```




```
>> [x, y] = fminbnd ( ' (24.0 – 2* x/3 + x.^2/30) ', 5.0, 20.0)
```

Результат поиска

```
x =
    10.0000
y =
    20.6667
```

Пример: найти максимум функции.

В М-файле с именем mf.m пишем:

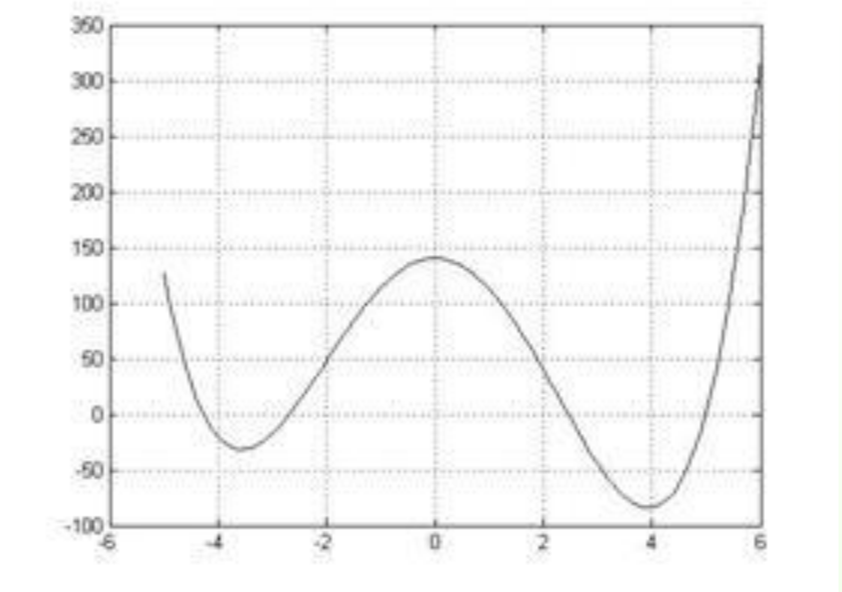
```
function y=mf(x)
y=x.^4-0.5*x.^3-28*x.^2+140;
end
```

Потом в командном окне пишем:

```
x=-5:0.1:6;
y=x.^4-0.5*x.^3-28*x.^2+140;
plot(x,y,'-k'), grid
%Максимум функции на интервале [-2 2]
y=-mf(x);
[x,y,]=fminbnd(@mf,-2,2)
```

Результат:

```
x =
3.7224e-008
y =
-140.0000
```



Для вычисление **экстремума функции многих** переменных MATLAB использует симплекс – метод Нелдера-Мида. Данный метод является одним из лучших методов поиска минимума функций многих переменных, где не вычисляются производные или градиент функции. Он сводится к построению симплекса в n-мерном пространстве, заданного $n + 1$ вершиной. В двухмерном пространстве симплекс является треугольником, а в трехмерном – пирамидой. На каждом шаге итераций выбирается новая точка решения внутри или вблизи симплекса. Она сравнивается с одной из вершин симплекса. Ближайшая к этой точке вершина симплекса заменяется этой точкой. Таким образом, симплекс перестраивается и позволяет найти новое, более точное положение точки решения. Алгоритм поиска повторяется, пока размеры симплекса по всем переменным не станут меньше заданной погрешности решения.

Вычисления реализует команда:

[x, z] = fminsearch(name, x0 [, options])

где: **name** - имя М-функции, вычисляющей значение $z=f(x_1,x_2,\ldots,x_n)$, зависящей от n переменных;

x0 – вектор из n элементов, содержащий координаты точки начального приближения;

options – параметры, управляющие ходом решения;

x - из n элементов, содержащий координаты точки, в которой достигается минимум функции;

z – значение функции в точке с координатами x.

[x,fval,exitflag] = fminsearch(...)

[x,fval,exitflag,output] = fminsearch(...)

exitflag – условия прерывания процесса поиска;

output – информация об оптимизации.

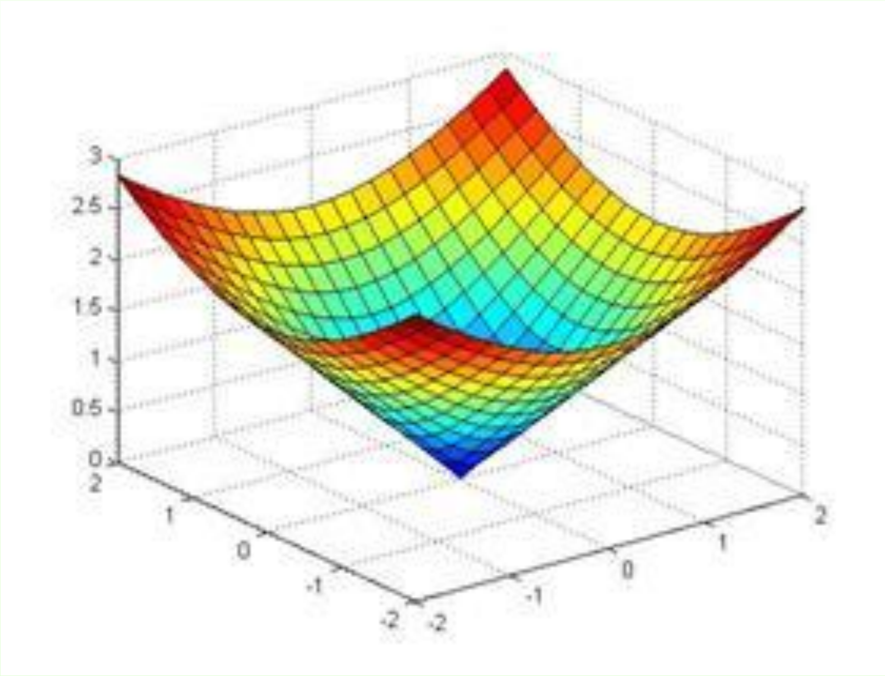
[x,fval] = fminsearch(banana, [-1.2, 1],optimset('TolX',1e-8));

Пример:

Найти минимум функции

[min.m]()

```
[z,f,exitflag,output] = fminsearch(@(x) sqrt(x(1)^2+x(2)^2), [2,2])
%Построение графика
[x y]=meshgrid(-2:0.2:2, -2:0.2:2);
z=sqrt(x.^2+y.^2);
surf(x,y,z);
```



Результат:

z =

1.0e-004 *

-0.4133 -0.1015

f =

4.2559e-005

Когда минимизируемая функция является достаточно гладкой и дважды дифференцируемой на заданном интервале, то для поиска её минимума можно воспользоваться функцией **fminunc**, реализующей метод наискорейшего спуска: **x= fminunc(@fun,x0)**

Для ускорения процесса поиска в функцию **fun** желательно включить формулы для вычисления градиента (это должно быть оговорено в **options**).

Пример. Пусть имеется функция $f=x_1^3+x_2^3-3x_1x_2$

Ее линия уровня при `f=0` представляет собой известную в геометрии кривую — *декартов лист*. Подключим к программе вычисления значения функции операторы для нахождения градиента `g` и гессиана `H` (см. текст функции `Descartes`). Сообщим об этом в списке управляющих параметров для функций минимизации:

```
options=optimset('Display','final','GradObj','on','Hessian','on');
```

Значение параметра `Display=final` означает, что все промежуточные выдачи, кроме заключительной, блокируются. Возьмем стартовую точку для поиска минимума `x0=[2; 2]`. Обращения к функции `fminunc` выполним в форме, позволяющей получить на выходе также значения градиента и гессиана.

```
[x,f1,e_flag,out,grad,hes]=fminunc(@Descartes,x0,options)
```

Построение линий уровня, фиксация начальной точки и найденного минимума (рис. 15.7) выполнены с помощью программы prog15_6.m:

```
function prog15_6
axes('Xlim',[-1.5 2.5],'Ylim',[-1.5 2.5]);
axis equal; grid off; hold on;
xlabel('x1'); ylabel('x2'); colormap copper
X0=-1.5:0.05:2.5;
[X Y]=meshgrid(X0);
s=size(X); Z=zeros(s);
for i = 1:s(1)
    for j = 1:s(2)
        Z(i,j) = Descartes([X(i,j); Y(i,j)]);
    end
end
```

```
V=-0.8:0.2:1;  contour(X,Y,Z,V);
options = ...
  optimset('Display','final','GradObj','on','Hessian','on');
x0=[2; 2];
line(x0(1),x0(2),'Marker','.','MarkerSize',10);
[x,f1,e_flag,out,grad,hes] = fminunc(@Descartes,x0,options)
line(x(1),x(2),'Marker','.','MarkerSize',20);
plot([x0(1),x(1)],[x0(2),x(2)],'k-');
```

```
function [f,g,H] = Descartes(x)
f = x(1)^3+x(2)^3-3*x(1)*x(2);
if nargout > 1
       g = [3*(x(1)^2-x(2)); 3*(x(2)^2-x(1))];
end
if nargout > 2
       H = [6*x(1) -3; -3 6*x(2)];
   end
end
```

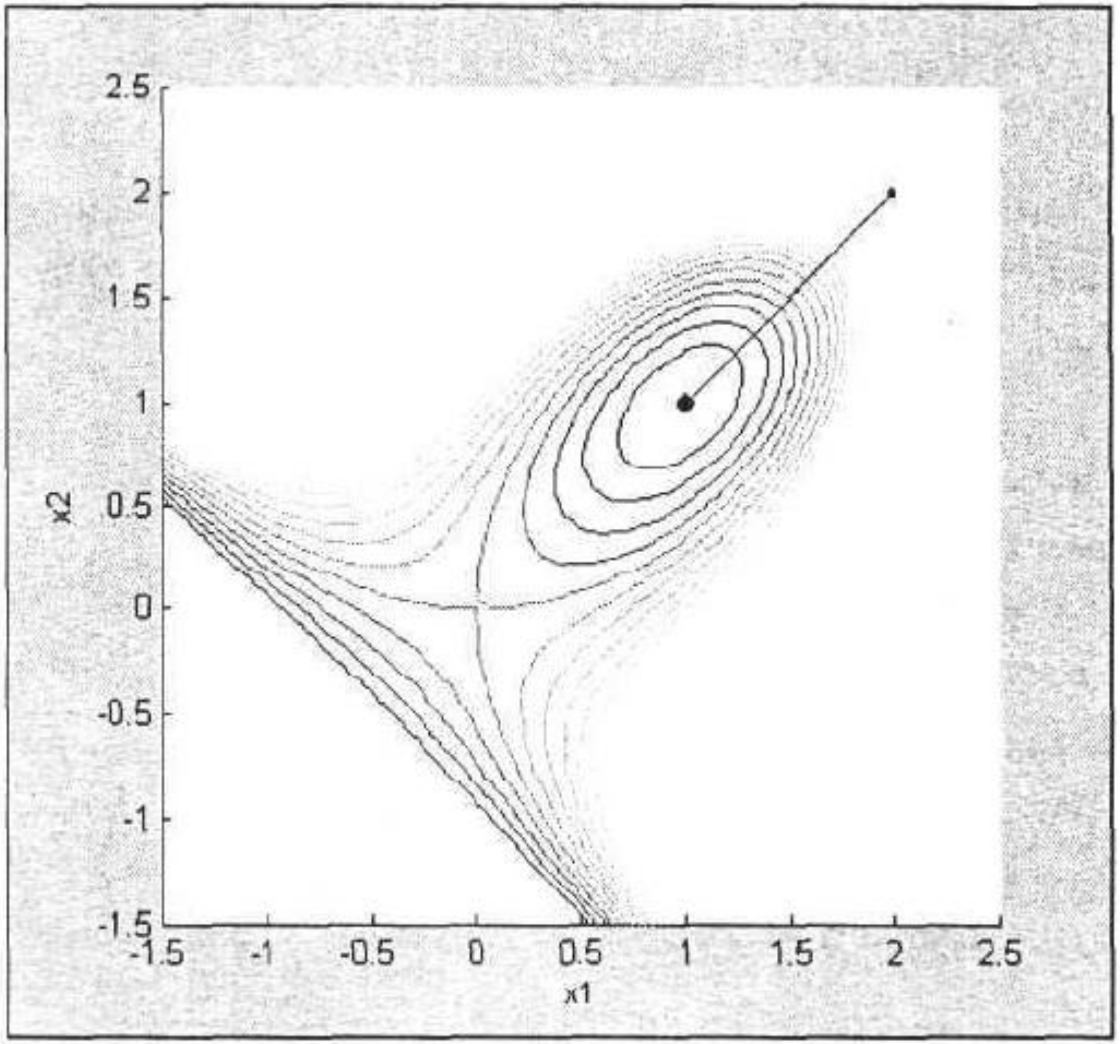


Рис. 15.7. Декартов лист

Результаты работы программы prog15_6.m приведены ниже:

```
>> Optimization terminated successfully:
 First-order optimality less than OPTIONS.TolFun, and no negative/zero
curvature detected
x =
    1.0000
    1.0000
 f1 =
     -1
 e_flag =
      1
 out =
        iterations: 6
         funcCount: 6
      cgiterations: 5
     firstorderopt: 6.9849e-010
         algorithm: 'large-scale: trust-region Newton'
```

```
grad =
  1.0e-009 *
     0.6985
     0.6985
hes =
     6.0000   -3.0000
    -3.0000    6.0000
```

В найденной точке (x=[1, 1]) градиент близок к нулю, а гессиан — положительно определен, что свидетельствует об успешном решении задачи.

Задания.

1. Вычислить минимум функции $f(x) = -x^{1/x}$, определив графически интервал его локализации. Вычисления провести с минимальным шагом по аргументу $1*10^{-5}$.

2. Вычислить минимум функции двух переменных

$x^4 + y^4 - 2x^2 + 4xy - 2y^2 + 1$ с точность $1*10^{-5}$.

Координаты начальной точки поиска [1.0, -1.0].

Оба задания выполнить:

1) используя специальные функции MATLAB; добавить в них вывод данных о процессе поиска exitflag,output; дополнить программы графическим выводом самих функций и выводом на этот же график процесса поиска экстремума;

2) использовать прилагающиеся программы по отдельным методам оптимизации для решения указанных задач; сравнить количество итераций, затраченное различными методами на поиск экстремумов.

3) попробовать найти все локальные минимумы в задании 2; попробовать найти локальный максимум на «дне чаши».

3.

Методом Ньютона найти точку минимума x^* и минимальное значение f^* функции $f(x) = (x-2)^4 - \ln x$ на отрезке $x \in [2; 3]$ с точностью $\varepsilon = |f'(x_k)| < 10^{-7}$.