Команды `do` и `od` – это *операторные скобки*, отмечающие начало и конец тела цикла (вместо команды `od` можно использовать более длинную команду `end do`). Продемонстрируем другой вариант использования оператора цикла:

```
> while b-a>0.00001 do x:=(a+b)/2:
if (df>0) then b:=(a+b)/2 else a:=(a+b)/2 fi od:
```

Кстати, для более детального *численного* анализа (в том числе при применении приближенных методов) может пригодиться переменная `Digits`, которая устанавливает количество выводимых цифр числа. По умолчанию этой переменной присваивается значение 10. Чтобы выводить больше цифр, нужно (лучше в начале программы) присвоить переменной `Digits` большее значение.

2. Функции нескольких переменных

Напомним необходимые и достаточные условия локального экстремума для функций нескольких переменных.

Теорема (необходимое условие экстремума первого порядка). *Пусть функция $f : \mathbb{R}^n \to \mathbb{R}$ в точке x_0 имеет локальный экстремум. Если в этой точке существуют частные производные $\frac{\partial f(x_0)}{\partial x_i}$, $i = 1, \ldots, n$, то все эти производные равны нулю.*

Если в точке x_0 все частные производные функции f равны нулю, точку называют критической или стационарной.

Теорема (условия экстремума второго порядка). *Пусть функция $f : \mathbb{R}^n \to \mathbb{R}$ определена, непрерывна и имеет непрерывные частные производные первого и второго порядка в некоторой окрестности точки x_0.*

1. Если в точке x_0 она имеет локальный минимум, то

$$\frac{\partial f(x_0)}{\partial x_i} = 0, \quad i = 1, \ldots, n, \tag{$*$}$$

и матрица частных производных второго порядка

$$\begin{pmatrix} \frac{\partial^2 f(x_0)}{\partial x_1^2} & \frac{\partial^2 f(x_0)}{\partial x_1 \partial x_2} & \ldots & \frac{\partial^2 f(x_0)}{\partial x_1 \partial x_n} \\ \ldots & \ldots & \ldots & \ldots \\ \frac{\partial^2 f(x_0)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x_0)}{\partial x_n \partial x_2} & \ldots & \frac{\partial^2 f(x_0)}{\partial x_n^2} \end{pmatrix} \tag{$**$}$$

неотрицательно определена.

Если в точке x_0 она имеет локальный максимум, то выполнены условия $(*)$ и матрица $(**)$ неположительно определена.

2. Если выполнены условия $(*)$ и матрица $(**)$ положительно определена, то функция f имеет в точке x_0 локальный минимум. Если выполнены условия $(*)$ и матрица $(**)$ отрицательно определена, то функция f имеет в точке x_0 локальный максимум.

Мы ограничимся рассмотрением функций двух переменных. Приведем пример исследования функции двух переменных с использованием системы MAPLE.

Зададим функцию. Это можно сделать так же, как на стр. 5:

```
> f:=x^4+y^4-2*x^2-2*y^2+4*x*y;
```

Как и ранее, для получения значения функции f в конкретной точке (например, (1,1)) нужно выполнить такие команды:

```
> x:=1; y:=1; f;
```

Однако в этом параграфе мы продемонстрируем использование другого способа задания функции – с помощью *функционального оператора*:

```
> f:=(x,y)->x^4+y^4-2*x^2-2*y^2+4*x*y;
```

При обращении к функции, заданной таким образом, нужно в скобках указывать список ее аргументов. Например, по команде

```
> f(1,1);
```

мы получим значение функции f в точке $x = 1$, $y = 1$.

Прежде всего нарисуем (трехмерный) график нашей функции:

```
> plot3d(f(x,y),x=-2.5..2.5,y=-2.5..2.5,axes=frame);
```

В этой команде указана опция `axes`, которая задает вид координатных осей. На самом деле, в команде `plot3d`, как и во многих других графических командах, можно указывать большое количество различных опций, которые задают вид графиков: цвет,

тип и толщину линий, угол поворота осей и др. Их подробное описание можно найти в Help.

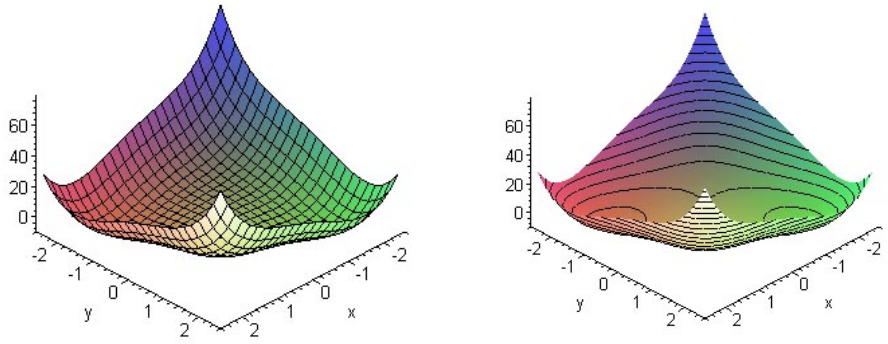

Рис. 2. Трехмерный график функции $f(x, y)$

Итак, мы получаем график, изображенный на рис. 2 слева. Заметим, что MAPLE позволяет поворачивать готовый график с помощью мыши, чтобы рассмотреть его с разных сторон.

На этой поверхности для наглядности нанесена координатная сетка, т. е. нарисованы линии $x = const$ и $y = const$. Другой способ визуализации поверхности – нанести на поверхность линии $z = const$ (они называются *линиями уровня* функции f). Получившийся график называется *контурным графиком*. Контурный график можно построить с помощью команды `contourplot3d`, которая на поверхности рисует линии уровня, получающиеся как пересечение поверхности и семейства *равноотстоящих* плоскостей, параллельных плоскости xOy. Для того, чтобы использовать это средство MAPLE, нужно вначале подключить пакет расширения `plots` с помощью команды `with`:

```
> with(plots):
> contourplot3d(f(x,y),x=-2.5..2.5,y=-2.5..2.5,
axes=frame,filled=true,contours=45);
```

Опция `contours` задает количество линий уровня функции на поверхности.

Другой способ получить контурный график – воспользовать-

ся командой `plot3d`, в которой опции `style` присвоено значение `patchcontour`:

```
> plot3d(f(x,y),x=-2.5..2.5,y=-2.5..2.5,axes=frame,
style=patchcontour,contours=45);
```

По этой команде получим график, изображенный на рис. 2 справа. На контурном графике чуть лучше виден характер поведения функции, но все равно для определения экстремумов удобнее использовать двумерные линии уровня функции. Их можно получить таким образом:

```
>contourplot(f(x,y),x=-2.5..2.5,y=-2.5..2.5,axes=frame,
filled=true,contours=25,coloring=[yellow,green]);
```

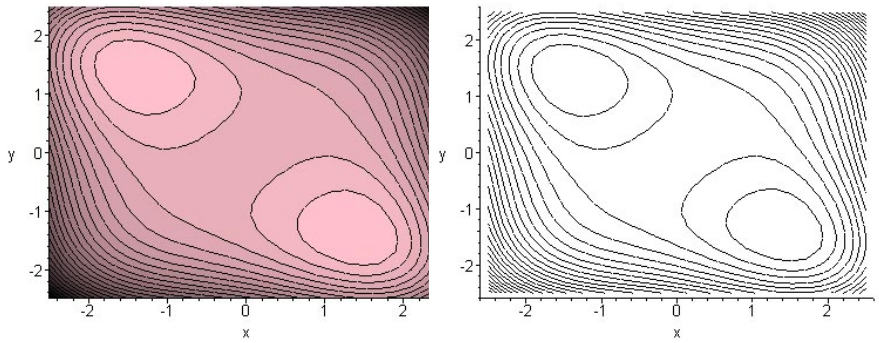

Рис. 3. Двумерные графики линий уровня функции $f(x, y)$

Получим картинку, изображенную на рис. 3 слева. В этой команде опция `filled` равна `true`. Это означает, что MAPLE автоматически закрашивает области между линиями уровня в разные цвета в зависимости от значения функции f на этих линиях уровня. Это делает график более выразительным и позволяет визуально оценить характер изменения значений функции. Если опцию `filled` положить равной `false` (или вообще не указывать), то получим «незакрашенный» график (как на рис. 3 справа). Напротив, если положить `filled=true`, то можно использовать опцию `coloring`, которая позволяет управлять цветами

закраски. Например, если указана `coloring=[yellow,green]`, то линии уровня, отвечающие наименьшему значению f, будут желтыми, линии уровня, отвечающие наибольшему значению f – зелеными, а все промежуточные линии уровня MAPLE закрасит в промежуточные цвета.

Глядя на эти рисунки, можно высказать предположение, что наша функция имеет два экстремума (они лежат внутри замкнутых кривых), причем из характера графика, полученного выше, следует, что это – точки минимума. Теперь можно, варьируя пределы изменения x и y, рассмотреть подробнее «подозрительные участки», или наоборот, построить линии уровня функции вне области, изображенной выше, чтобы убедиться в том, что там функция экстремумов не имеет.

Далее продемонстрируем, как можно решать задачу нахождения экстремумов нашей функции аналитически. Вначале применим необходимое условие первого порядка. Для нахождения критических точек функции зададим систему уравнений $\frac{\partial f(x,y)}{\partial x} = 0$, $\frac{\partial f(x,y)}{\partial y} = 0$:

```
> Sys_Eqn:=diff(f(x,y),x)=0,diff(f(x,y),y)=0;
```

Отметим, что уравнения системы перечисляются после знака `:=` через запятую. Получаем:

$$Sys\_Eqn := 4x^3 - 4x + 4y = 0,\ 4y^3 - 4y + 4x = 0.$$

Для решения этой системы применим команду `solve`:

```
> solve({Sys_Eqn});
```

Заметим, что в этой команде уравнения системы заключены в фигурные скобки, которые означают тип данных «множество».

По команде `solve` MAPLE пытается найти точные решения системы, если это возможно. Однако в данном случае MAPLE выдаст сообщение, содержащее, в частности, функцию *RootOf*. Это означает, что аналитические решения системы нельзя выразить в радикалах. В таком случае можно попытаться получить численные, приближенные значения решений с помощью

команды `evalf`. Эта команда служит для принудительного приближенного вычисления результата:

```
> evalf(%);
```

Обратим внимание на символ %, который стоит в качестве параметра. Он означает результат последнего **по времени** вычисления[2].

Две последние команды можно объединить в одну:

```
> evalf(solve({Sys_Eqn}));
```

Другой способ «бороться» с функцией *RootOf* в ответе – в исходную систему искусственно внести «неточность», чтобы MAPLE не пыталась найти точное решение. Это можно сделать, например, так: вместо какого-нибудь целого числа в записи уравнения системы, например, числа 0, написать число 0.0 (или просто 0. – «ноль с точкой»). MAPLE воспринимает это число как нецелое, и автоматически ищет приближенные решения такой системы.

Итак, мы получили такие (вещественные) корни: точку $(0, 0)$ (кратности 3) и точку $(-1.414213562, 1.414213562)$. Остается открытым вопрос: получены ли *все* решения системы? Решение систем нелинейных уравнений – очень сложная задача, и универсальные методы, которые применяются в системе MAPLE, могут выдать только часть решений. Для контроля применим графический метод: на плоскости (x, y) нарисуем кривые, отвечающие уравнениям нашей системы. Воспользуемся командой построения графиков неявных функций `implicitplot` (эта команда также находится в пакете `plots`):

```
> implicitplot({Sys_Eqn},x=-2.5..2.5,y=-2.5..2.5);
```

[2] Заметим, что пользователь может выполнять команды не в той последовательности, в которой они введены и изображены на экране. Поэтому конкретное значение выражения % зависит от того, какая команда была выполнена последней, а не от того, что написано в предыдущей строке.

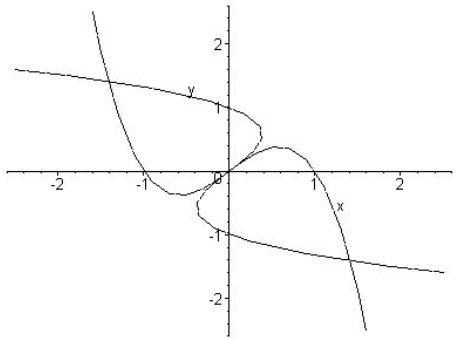

Рис. 4. Кривые $\frac{\partial f(x,y)}{\partial x} = 0$ и $\frac{\partial f(x,y)}{\partial y} = 0$

На рисунке мы видим две кривые, которые задаются уравнениями $4x^3 - 4x + 4y = 0$ и $4y^3 - 4y + 4x = 0$ нашей системы. Значит, решения нашей системы – это точки пересечения кривых. Итак, рассматриваемая система имеет по меньшей мере три решения: начало координат и еще две точки, симметричные относительно начала координат. Выше мы видели, что MAPLE по команде `evalf(%)` выдает только два из этих трех корней![3]

Когда корни локализованы, для нахождения их численных значений гораздо эффективнее вместо команды `solve` применять команду `fsolve`, в которой можно указать область для поиска решений:

```
> fsolve({Sys_Eqn},{x=0.1..2.5,y=-2.5..0.1});
```

Аналогично можно получить и остальные решения. Заметим, что команда `fsolve` не всегда находит решения в указанной области, даже если они там есть. В таких случаях помогает такой прием: попытаться изменить пределы варьирования переменных. Например, для нахождения решения $(0, 0)$ команда

```
> fsolve({Sys_Eqn},{x=-0.5..0.5,y=-0.5..0.5});
```

может не выдать решения[4] (тогда MAPLE в строке результата просто повторяет введенную команду), а по команде

[3] Это может зависеть от версии MAPLE.

[4] Это зависит от версии MAPLE.

```
> fsolve({Sys_Eqn},{x=-0.05..0.05,y=-0.05..0.05});
```

результат может быть успешно получен.

Итак, мы получили, что критическими точками нашей функции $f(x, y)$ являются точки

$$(0, 0),\ (1.414213562, -1.414213562),\ (-1.414213562, 1.414213562).$$

Так как $\sqrt{2} \approx 1.414213562$, то возникает предположение, что на самом деле три решения системы – это точки

$$(0, 0), \quad (\sqrt{2}, -\sqrt{2}), \quad (-\sqrt{2}, \sqrt{2}).$$

Чтобы это проверить, подставим указанные точки в систему, используя команду `subs`:

```
> subs(x=sqrt(2),y=-sqrt(2),Sys_Eqn);
```

Получим $0 = 0$, значит, действительно, $(\sqrt{2}, -\sqrt{2})$ – решение системы. Аналогично проверяется точка $(-\sqrt{2}, \sqrt{2})$.

И тем не менее, вопрос о том, найдены ли все решения системы `Sys_Eqn`, остается открытым! Для того, чтобы доказать, что мы нашли все решения системы, примем во внимание конкретный вид уравнений. В *данном конкретном* случае можно сложить уравнения системы, откуда сразу получим, что $x^3+y^3 = 0$, а значит, $x = -y$. Подставляя это выражение для x, например, в первое уравнение, получаем $-4y^3 + 8y = 0$, откуда $y = 0$, $y = \pm\sqrt{2}$. Итак, действительно, наша система имеет ровно три решения, которые мы уже нашли.

Далее применим условия экстремума второго порядка. Для этого построим матрицу вторых производных, используя тип данных `matrix`:

```
> My_matr:=matrix(2,2,
[[diff(f(x,y),[x,x]),diff(f(x,y),[x,y])],
[diff(f(x,y),[y,x]),diff(f(x,y),[y,y])]]);
```

Первые два параметра описания типа `matrix` (в данном случае они равны 2, 2) – это размеры матрицы (количество строк и столбцов), а третий параметр – это список строк.

Тип данных «список» в системе MAPLE задается таким образом: все элементы списка помещаются в квадратные скобки. Например, команда

```
> My_List:=[1,2,3];
```

задает список из трех элементов – чисел 1, 2 и 3, а команда

```
> My_List:=[[1,2,3],[4,5,6]];
```

задает список из двух элементов, причем каждый из этих элементов – это тоже список.

Значит, в описании матрицы `My_matr` третий параметр – это «список списков»: внешний список – это список строк, а каждая строка – это список элементов, входящих в эту строку. Обратите внимание на то, что в уже известной нам команде `diff` для нахождения производных второго порядка используется *список* переменных, по которым нужно производить дифференцирование.

Итак, мы определили переменную `My_matr`, которая имеет вид

$$My\_matr := \begin{bmatrix} 12x^2 - 4 & 4 \\ 4 & 12y^2 - 4 \end{bmatrix}.$$

Для того, чтобы определить, является ли матрица вторых производных положительно определенной или отрицательно определенной, воспользуемся критерием Сильвестра. Для этого найдем определитель нашей матрицы. Подключив пакет расширения `linalg`, используем функцию `det`:

```
> with(linalg):
> d:=det(My_matr);
```

Получим:

$$d := 144x^2y^2 - 48x^2 - 48y^2.$$

Теперь подставим по очереди все критические точки в главные угловые миноры нашей матрицы:

```
> x:=0:y:=0: print('x'=x,'y'=y,
'First_minor'=My_matr[1,1],'Second_minor'=d);
> x:=sqrt(2):y:=-sqrt(2): print('x'=x,'y'=y,
'First_minor'=My_matr[1,1],'Second_minor'=d);
> x:=-sqrt(2):y:=sqrt(2): print('x'=x,'y'=y,
'First_minor'=My_matr[1,1],'Second_minor'=d);
```

Это же можно сделать в цикле: зададим список `My_List` критических точек, в котором каждый элемент списка – список из двух координат точки, а потом выполним цикл по переменной `z` (она имеет тип «список»), которая пробегает список наших критических точек:

```
>My_List:=[[0,0],[sqrt(2),-sqrt(2)],[-sqrt(2),sqrt(2)]];
>for z in My_List do x:=z[1];y:=z[2];print('x'=x,'y'=y,
'First_minor'=My_matr[1,1],'Second_minor'=d) od:
```

Получаем, что матрицы вторых производных в точках $(\sqrt{2}, -\sqrt{2})$ и $(-\sqrt{2}, \sqrt{2})$ положительно определены. Следовательно, функция f имеет локальные минимумы в этих точках. Минимальные значения можно найти так:

```
> f(sqrt(2),-sqrt(2)); f(-sqrt(2),sqrt(2));
```

В точке $(0, 0)$ матрица вторых производных вырожденная. Значит, для того, чтобы определить, есть ли в этой точке локальный экстремум, необходимо провести дополнительные исследования. Покажем, как можно использовать MAPLE в такой ситуации.

Прежде всего, можно рассмотреть более подробно график функции f в окрестности точки $(0, 0)$, используя команды `plot3d`, `contourplot3d` и `contourplot`. Можно также попробовать нарисовать отдельно линию уровня функции, на которой лежит точка (0,0), т. е. кривую $f(x, y) = 0$. Это можно сделать с помощью команды `implicitplot`:

```
> implicitplot(f(x,y)=0,x=-3..3,y=-3..3,grid=[30,30]);
```

Эта команда работает следующим образом: вначале определяется знак $f(x,y)$ в точках сетки $n \times m$, где величины n и m определяются опцией `grid=[n,m]`. Если в соседних точках функция f имеет разные знаки, то между ними должна находиться точка кривой $f(x,y) = 0$. По определенному интерполяционному алгоритму MAPLE строит *отдельные точки* искомой кривой, а затем эти точки соединяются так, чтобы получилась гладкая кривая. Для кривых с особенностями эта команда может работать не совсем корректно, особенно в случае разрывных кривых и кривых с самопересечениями. Тогда можно попробовать увеличить число точек сетки (с помощью опции `grid`), однако при этом время выполнения команды заметно возрастет. Кроме того, можно использовать опцию `style=point`. Тогда MAPLE не будет соединять точки, а нарисует на их месте кружочки. В некоторых случаях такой рисунок будет более наглядным.

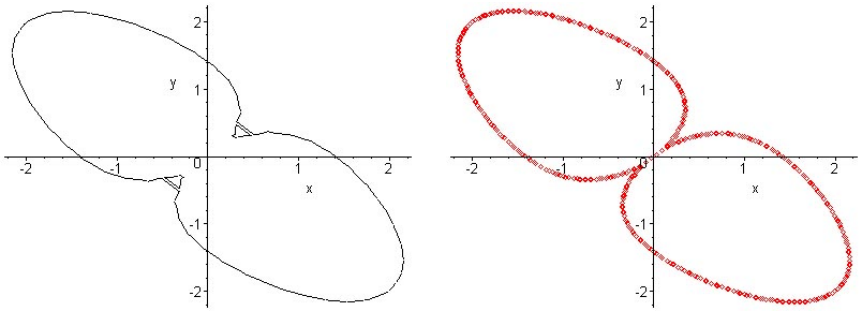

Рис. 5. Линия уровня $f(x,y) = 0$

На рис. 5 показаны результаты выполнения команды `implicitplot`: график слева соответствует команде

```
> implicitplot(f(x,y)=0,x=-3..3,y=-3..3,grid=[30,30]);
```

а график справа – команде

```
> implicitplot(f(x,y)=0,x=-3..3,y=-3..3,grid=[100,100],
style=point);
```

Итак, линия уровня $f(x,y)=0$ состоит из двух замкнутых кривых, имеющих общую точку (0,0). Так может выглядеть линия уровня, соответствующая седлу: в некоторых направлениях функция f растет, а в некоторых – убывает, т. е. (0,0) не является точкой экстремума. Докажем, что это действительно так.

Вернемся еще раз к рис. 3, на котором изображены линии уровня функции $f(x,y)$, и попробуем найти направления роста и убывания функции (из начала координат). Глядя на график, можно предположить, что вдоль прямых $y=x$ функция растет, а вдоль прямых $y=-x$ – убывает. Покажем это. Определим функции

```
> f1:=(x)->f(x,x);
> f2:=(x)->f(x,-x);
```

и нарисуем их графики в окрестности точки $x=0$ (или исследуем эти функции так, как рассказано в параграфе 1). Получаем, что в точке $x=0$ функция $f1$ имеет минимум, а функция $f2$ – максимум. Следовательно, точка (0,0) действительно не является экстремальной для исходной функции f.

Полный текст программы MAPLE см. в приложении, стр.43.

Замечание. В этом параграфе мы продемонстрировали возможности MAPLE для нахождения экстремумов *полиномиальной* функции двух переменных. В таком случае при применении метода множителей Лагранжа возникает система полиномиальных уравнений. Обычно MAPLE хорошо справляется с нахождением решений полиномиальных систем. Однако в более сложных случаях особое внимание приходится уделять локализации решений систем, чтобы в дальнейшем можно было корректно применять команду `fsolve` в ограниченных областях. Здесь особенно полезными оказываются возможности MAPLE по визуализации поверхностей.

Интересный класс поверхностей можно получить таким образом. Рассмотрим функцию h от двух переменных x и y, зависящую еще от трех параметров:

```
> h:=(x,y,a,b,c)->c*exp(-(x-a)^2-(y-b)^2);
```

и зададим функцию $f(x,y)$ как сумму нескольких таких слагаемых, отвечающих различным значениям параметров, например:

```
>f:=(x,y)->h(x,y,-1,-1,1)+h(x,y,0,0,-1.2)+h(x,y,2,-2,1);
```

На рис. 6 изображены ее график и линии уровня. Полученная «вулканическая» поверхность похожа на фантастический пейзаж.

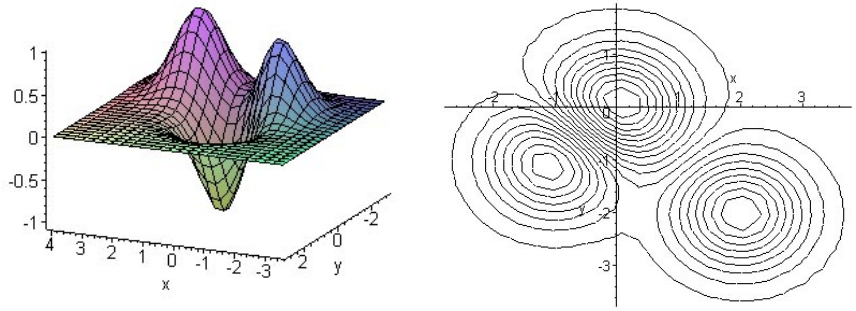

Рис. 6. «Вулканическая» поверхность

Замечание. Если точка экстремума локализована, то для ее нахождения можно применять численные методы, используя систему MAPLE. В качестве примера приведем одну возможную реализацию градиентного метода для нахождения минимума нашей функции f. Напомним, что идея градиентного метода состоит в следующем: чтобы найти очередное приближение, мы «шагаем» из текущей точки в направлении антиградиента функции f. При этом величина шага может выбираться по-разному. В нашей программе она выбирается так, чтобы значение функции в каждом следующем приближении было меньше, чем в предыдущем:

```
> restart;
> f:=(x,y)->x^4+y^4-2*x^2-2*y^2+4*x*y;
> fx:=diff(f(x,y),x); fy:=diff(f(x,y),y);
> x0:=2.0:y0:=-1.0:List[0]:=[x0,y0]:
> w:=1:j:=0:
```

```
> while w>0.00001 do x:=x0:y:=y0:a:=1:
for i from 1 to 20 do x1:=x0-a*fx:y1:=y0-a*fy:
if f(x1,y1)-f(x0,y0)<0 then break else a:=a/2:fi:od:
w:=sqrt((x1-x0)^2+(y1-y0)^2):x0:=x1:y0:=y1:j:=j+1:
List[j]:=[x0,y0]: od:
> j;x0;y0;
> with(plots):with(plottools):
> for k from 0 to j do
d[k]:=display(POINTS(List[k])):od:
> display(seq([d[k]],k=0..j),insequence=true);
> for k from 1 to j do
c[k]:=display(line(List[k-1],List[k])):od:
> x:='x':y:='y':
z:=contourplot(f(x,y),x=0.5..2,y=-1.6..-0.9,
contours=18,scaling=constrained):
> display(z,seq([c[k]],k=1..j));
```

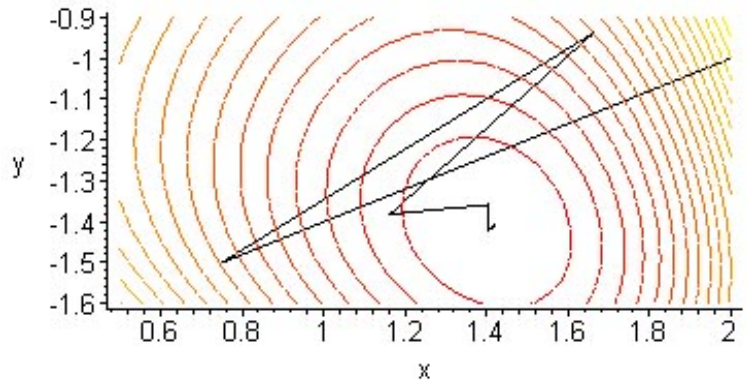

Рис. 7. Нахождение минимума градиентным методом

В этой программе (fx, fy) – это градиент функции, $(x0, y0)$ и $(x1, y1)$ – два последовательных приближения, j – номер итерации, w – точность, a – величина шага в направлении антиградиента. Список *List* создается для дальнейшей визуализации приближений. Команда

```
> display(seq([d[k]],k=0..j),insequence=true);
```

позволяет получить *анимацию* приближений: точки, приближающие решение, выводятся последовательно. Анимационный эффект достигается благодаря значению `true` опции `insequence` (по умолчанию эта опция имеет значение `false`). Команда

```
> display(z,seq([c[k]],k=1..j));
```

рисует одновременно все последовательные приближения, соединенные отрезками, на фоне линий уровня (см. рис. 7).

3. Пример: функция Розенброка

Как и в случае функции одной переменной, для полиномов от нескольких переменных аналитический метод нахождения экстремума *обычно* оказывается достаточно эффективным. Однако при визуальном способе локализации корней даже в этом случае могут возникнуть трудности. Классический пример – функция Розенброка:

$$f(x,y) = 100(y-x^2)^2 + (1-x)^2.$$

Эта функция имеет *овражный характер*: ее график напоминает пологий изогнутый овраг с почти плоским дном и крутыми стенками. Функция Розенброка используется как тестовая для проверки и сравнения эффективности различных *численных* методов нахождения экстремумов. Из вида функции Розенброка следует, что она имеет минимум в точке $x = 1$, $y = 1$, однако простого взгляда на график недостаточно, чтобы этот минимум локализовать.

Посмотрим, можно ли «увидеть» овраги функции Розенброка с помощью системы MAPLE. Сначала нарисуем ее график:

```
> f:=(x,y)->100*(y-x^2)^2+(1-x)^2;
> plot3d(f(x,y),x=-5..5,y=-12..12,axes=framed);
```

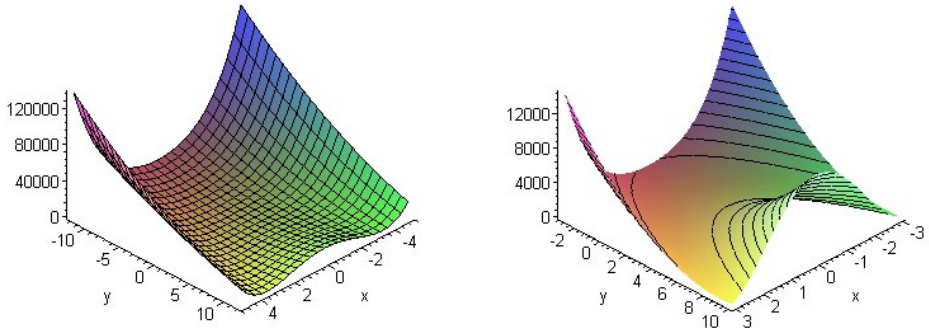

Рис. 8. График функции Розенброка

Полученный график (рис. 8 слева) показывает, что функция очень быстро возрастает, но визуально точку минимума определить не удается. Уменьшим пределы изменения аргументов и применим опцию `style=patchcontour`:

```
> plot3d(f(x,y),x=-3..3,y=-3..10,axes=framed,
style=patchcontour);
```

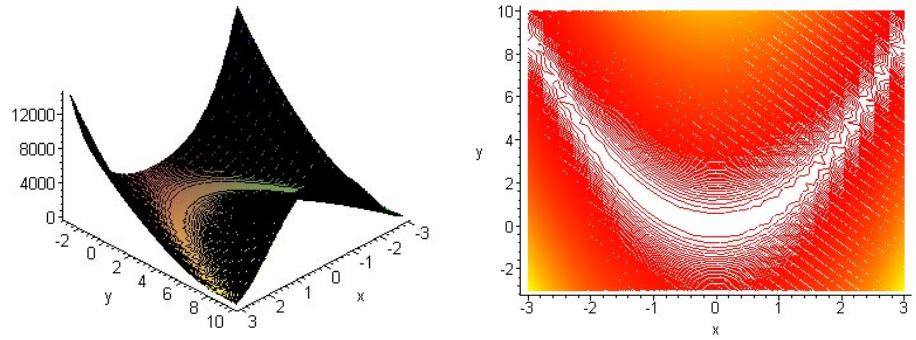

Рис. 9. Дно оврага (функция Розенброка)

Получили график, изображенный на рис. 8 справа: подозрительной является та область, где линии уровня не прорисованы, но *гарантировать*, что там есть минимум, нельзя. Попробуем увеличить количество линий уровня (применив опцию `contours`):

```
> plot3d(f(x,y),x=-3..3,y=-3..10,axes=framed,
style=patchcontour,contours=200);
```

Вся поверхность оказалась покрытой линиями уровня, и более светлой осталась плоская область на дне оврага (рис. 9 слева). Команда

```
> contourplot(f(x,y),x=-3..3,y=-3..10,
axes=framed,contours=400);
```

позволяет получить двумерное изображение дна оврага (рис. 9 справа). Чтобы получить представление о том, насколько «круты» стенки оврага, можно применить опцию `view`, которая ограничивает видимую часть графика по оси Oz (рис.10):

```
> plot3d(f(x,y),x=-2..2,y=-2..5,axes=framed,
style=patchcontour,view=0..100);
```

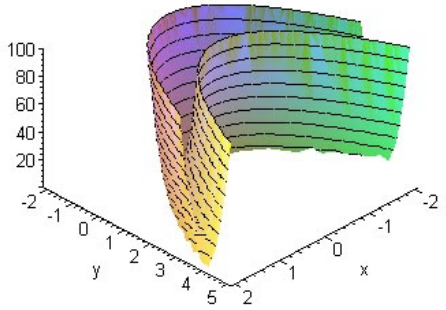

Рис. 10. Стенки оврага (функция Розенброка)

Построим теперь отдельно замкнутую линию уровня, внутри которой находится точка экстремума. Для этого воспользуемся командой `implicitplot`:

```
> implicitplot(f(x,y)=0.1,x=0.2..1.8,y=0.2..2.3,
grid=[150,150],color=black);
```

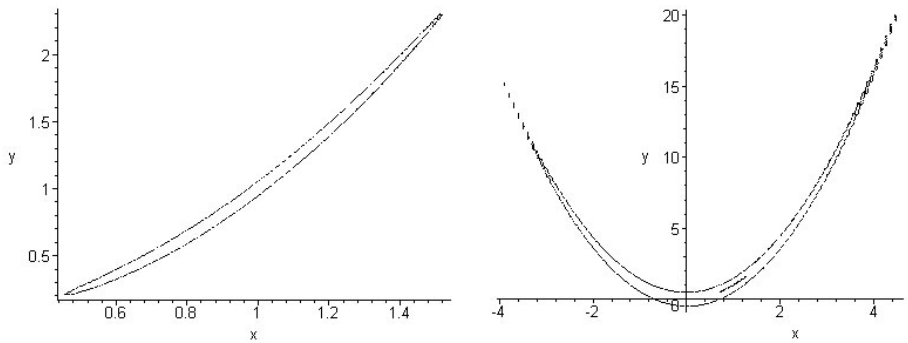

Рис. 11. Линии уровня функции Розенброка

Получили замкнутую кривую $f(x, y) = 0.1$, внутри которой и находится точка минимума (рис. 11 слева). Чтобы лучше представить себе вид дна оврага, можно нарисовать еще несколько линий уровня. Мы ограничимся двумя кривыми – $f(x, y) = 0.1$ и $f(x, y) = 25$:

```
> a1:=implicitplot(f(x,y)=0.1,x=0.2..1.8,y=0.2..2.3,
grid=[150,150],color=black):
> a2:=implicitplot(f(x,y)=25,x=-5..10,y=-1..20,
grid=[150,150],color=black):
> display({a1,a2});
```

В первых двух строчках создаются, но *не выводятся на печать* графические структуры (обратите внимание на двоеточие в конце этих команд). Команда `display` выводит обе кривые на одном графике. Теперь (рис. 11 справа) видно, что овраг изогнут, имеет очень крутые стенки, причем его концы очень узкие (видно, как некорректно работает на этих участках функция `implicitplot`).

Интересно применить программу приближенного нахождения минимума, приведенную в конце предыдущего параграфа, для функции Розенброка с различными начальными приближениями. Как правило, число итераций будет весьма велико, причем основное движение к точке минимума будет происходить по дну оврага с очень низкой скоростью.