5 Построение трехмерных графиков в Scilab

В настоящей главе будут рассмотрены основные возможности SCILAB по созданию трехмерной графики объемной и пространственной .При этом к трехмерным отнесем все графики, положение каждой точки которых задается тремя величинами.

В целом процесс построения графика функции вида $Z(x, y)$ можно разделить на три этапа:

1. Создание в области построения графика прямоугольной сетки. Для этого формируются прямые линии, параллельные координатным осям x_i и y_j, где $x_i = x_0 + ih$, $h = \frac{x_n - x_0}{n}$, $i = 0,1\ldots,n$ и $y_j = y_0 + jh$, $h = \frac{y_k - y_0}{k}$, $j = 0,1\ldots,k$.
2. Вычисление значений функции $z_{ij} = f(x_i, y_j)$ во всех узлах сетки.
3. Обращение к функции построения трехмерной графики.

5.1 Функция plot3d и plot3d1

В Scilab поверхность можно построить с помощью функций *plot3d* или *plot3d1*. Их отличие состоит в том, что *plot3d* строит поверхность и заливает ее одним цветом, а *plot3d1* поверхность, каждая ячейка которой имеет цвет, зависящий от значения функции в каждом соответствующем узле сетки (см. рис. 5.1).

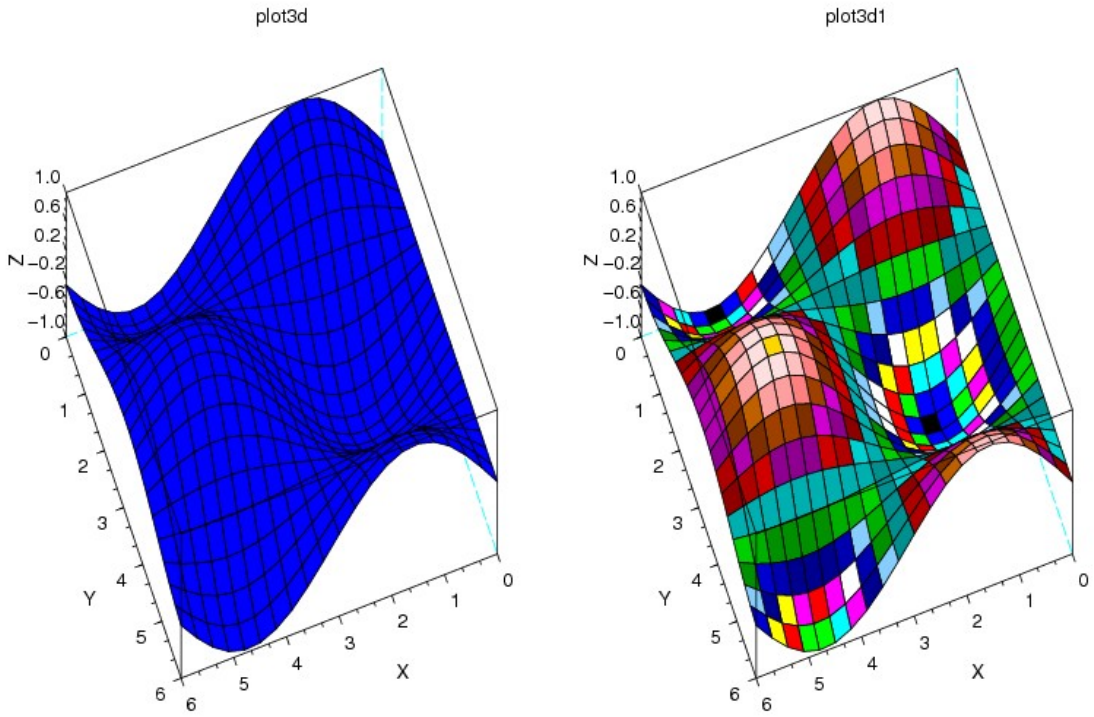

Рисунок 5.1. Отличие функций plot3d и plot3d

Обращение к функциям следующее:

```
plot3d(x,y,z,[theta,alpha,leg,flag,ebox][keyn=valuen]),
plot3d1(x,y,z,[theta,alpha,leg,flag,ebox][keyn=valuen]),
```

здесь *x* – вектор-столбец значений абсцисс ;

y – вектор-столбец значений ординат ;

z – матрица значений функции ;

theta, alpha – действительные числа , которые определяют в градусах сферические координаты угла зрения на график. Попросту говоря, это угол, под которым наблюдатель видит отображаемую поверхность;

leg – подписи координатных осей графика – символы, отделяемые знаком @. Например , 'X@Y@Z'.

flag – массив, состоящий из 3 целочисленных параметров *[mode,type,box]*. Здесь *mode* устанавливает цвет поверхности (см .табл .5.1).

Таблица 5.1. Значения параметра mode

Значение	Описание
>0	поверхность имеет цвет «mode», выводится прямоугольная сетка.
0	выводится прямоугольная сетка, заливка отсутствует (белый цвет).
<0	поверхность имеет цвет «mode», отсутствует прямоугольная сетка.

По умолчанию, равен 2 – цвет заливки синий ,прямоугольная сетка выводится .

type – позволяет управлять масштабом графика (см . табл . 5.2), по умолчанию имеет значение 2;

Таблица 5.2. Значения параметра type

Значение	Описание
0	применяется способ масштабирования, как у ранее созданной графики
1	границы графика указываются вручную с помощью параметра `ebox`
2	границы графика определяют исходные данные

box – определяет наличие рамки вокруг отображаемого графика (см . табл . 5.3). По умолчанию равен 4.

Таблица 5.3.Значения параметра box

Значение	Описание
0 и 1	нет рамки
2	только оси, находящиеся за поверхностью
3	выводится рамка и подписи осей
4	выводится рамка, оси и их подписи

ebox – определяет границы области , в которую будет выводиться поверхность , как вектор *[xmin,xmax,ymin,ymax,zmin,zmax]*. Этот параметр может использоваться только при значении параметра *type=1*.

keyn=valuen – последовательность значений свойств графика *key1=value1, key2=value2,...,keyn=valuen*, таких как – толщина линии , ее цвет , цвет заливки фона графического окна, наличие маркера и др. (см. параграф 4.6).

Таким образом, функции *plot3d (plot3d1)* в качестве параметров необходимо передать

прямоугольную сетку и матрицу значений в узлах сетки.

ЗАДАЧА 5.1

Построить график функции $Z=\sin(t)\cdot\cos(t)$ при помощи команды *plot3d*.

Создадим массив значений аргумента *t*. Вычислим значения функции и запишем их в массив Z.

Обратите внимание, что при обращении к функции *plot3d* в качестве параметров X и Y, задающих прямоугольную сетку, дважды указан параметр *t*, поскольку обе функции и *sin*, и *cos* зависят от одной переменной *t* (см. листинг 5.1, рис. 5.2).

```
t=[0:0.3:2*%pi]';
Z=sin(t)*cos(t');
plot3d(t,t,Z);
```

Листинг 5.1

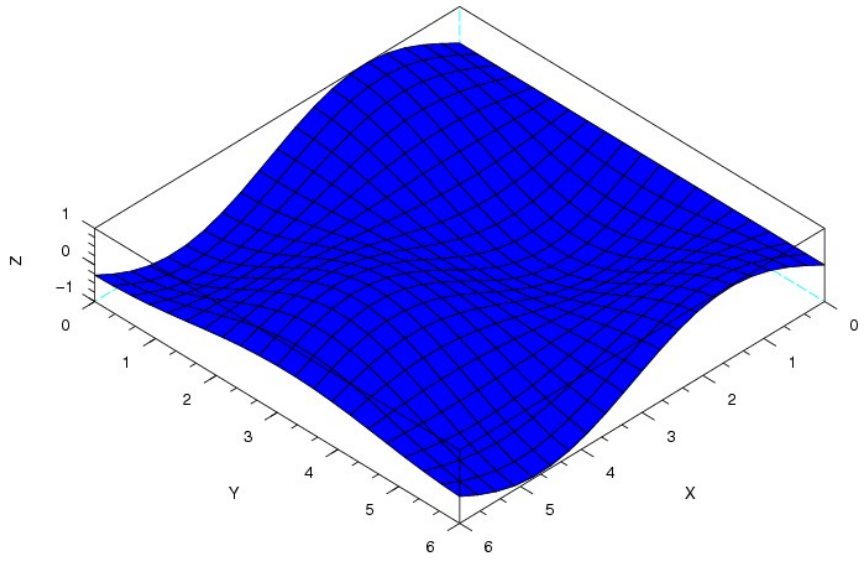

Рисунок 5.2. График функции $Z=\sin(t)\cdot\cos(t)$

Теперь немного усложним задачу. Построим поверхность, уравнение которой задается двумя независимыми переменными.

ЗАДАЧА 5.2

Построить график функции $Z=5y^2-x^2$ при помощи команды *plot3d1*.

Прежде всего зададим массивы X и Y.

Затем сформируем матрицу значений функции $Z(x_i, y_j)$, используя оператор цикла *for*. Здесь *i* параметр цикла, который будет перебирать все значения массива X, а *j* параметр цикла, который будет сопоставлять каждого значению массива X по очереди все значения массива Y.

Таким образом, сначала будут вычислены все значения функции Z при меняющемся Y (от первого до последнего значения в массиве) и первом значении массива X. Затем при втором значении массива X и т.д.

Напомним, здесь length определяет количество элементов массива X (Y) (см. главу 2).

Наконец, для построения поверхности обратимся к функции *plot3d1* (см. листинг 5.2, рис. 5.3).

```
x=[-2:0.1:2];
y=[-3:0.1:3];
```

```
for i=1:length(x)
for j=1:length(y)
z(i,j)=5*y(j)^2-x(i)^2;
end
end
plot3d1(x',y',z,-125,51);
```

Листинг 5.2

Как видно из примера, использование лишь функции *plot3d* для графического изображения показателей, зависящих от двух независимых переменных, достаточно сложно. В Scilab существует несколько команд, призванных облегчить процедуру создания прямоугольной сетки это *genfac3d, eval3dp*.

Простейшей из них по синтаксису является функция *genfac3d:*

```
[xx,yy,zz]=genfac3d(x,y,z)
```

Здесь *xx,yy,zz* результирующая матрица размером (4,n-1xm-1),где *xx(:,i), yy(:,i) и zz(:,i)* координаты каждой из ячеек прямоугольной сети ;

x вектор x-координат размера m;

y - вектор y-координат размера n;

z матрица размера (m,n)значений функции $Z(x_i, y_j)$.

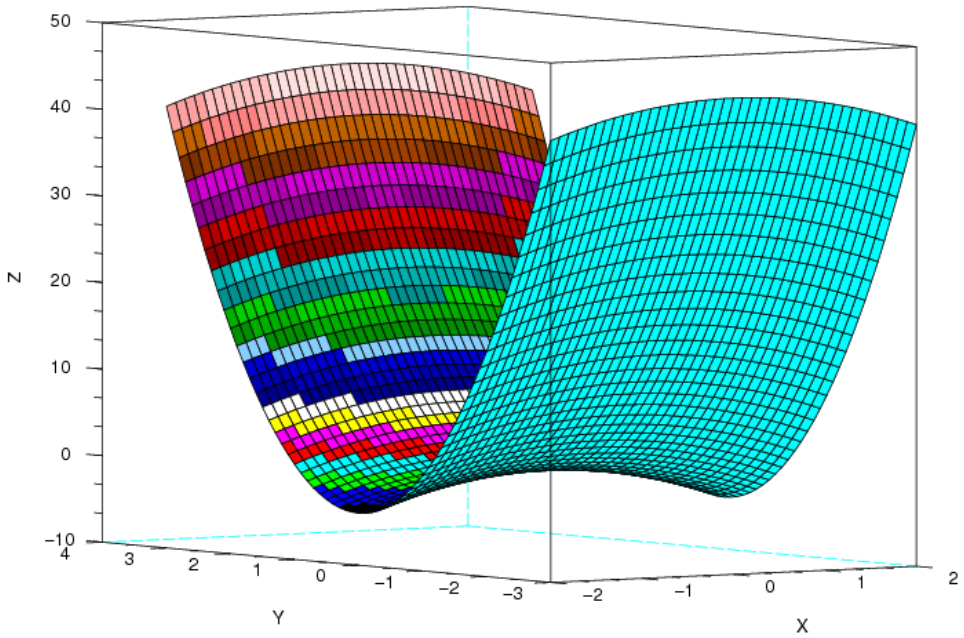

Рисунок 5.3. График функции $Z=5y^2-x^2$ *, построенный при помощи команды plot3d1*

ЗАДАЧА 5.3

Построить график функции $Z=\sin(t)\cdot\cos(t)$, используя команду *genfac3d.*

Определим массив параметра *t* и вычислим значения функции $Z=\sin(t)\cdot\cos(t)$. прямоугольную сетку создадим при помощи команды *genfac3d* (см. листинг 5.3).

Для формирования графика обратимся к функции *plot3d* (см. рис. 5.4).

```
t=[0:0.3:2*%pi]';
z=sin(t)*cos(t');
```

```
[xx,yy,zz]=genfac3d(t,t,z);
plot3d(xx,yy,zz);
```

Листинг 5.3

Недостатком команды *genfac3d* является то, что она все-таки не упрощает работу с функцией *plot3d*, если поверхность задается функцией от двух переменных. Тогда необходимо использовать команду *eval3dp*:

```
[Xf,Yf,Zf]=eval3dp(fun,p1,p2)
```

Xf,Yf,Zf результирующая матрица размером (4,n-1xm-1), где *xx(:,i), yy(:,i)* и *zz(:,i)* координаты каждой из ячеек прямоугольной сети;

fun функция ,определенная пользователем ,которая задает трехмерный график ;

p1 - вектор размера m;

p2 - вектор размера n.

Проиллюстрируем действие команды *eval3dp* следующим примером.

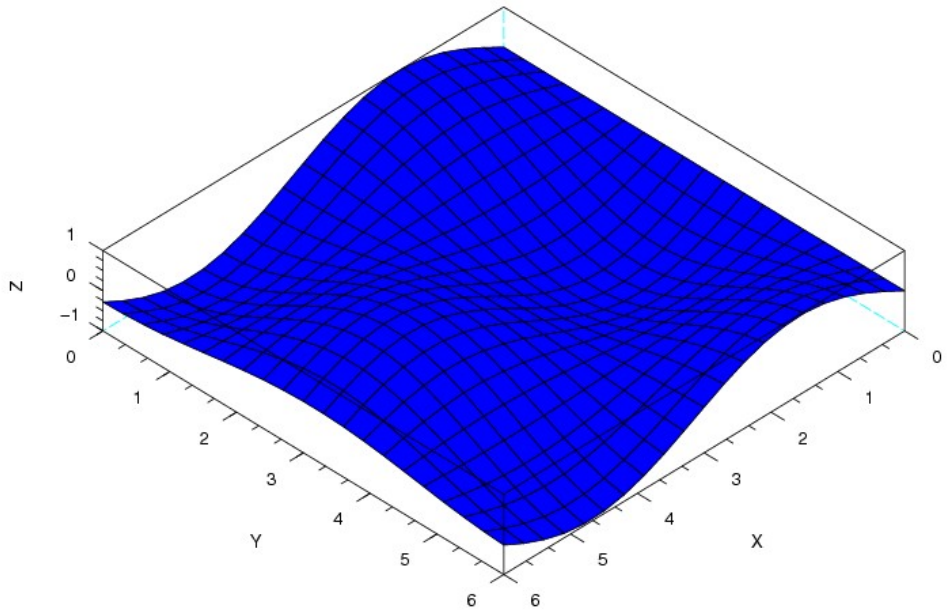

Рисунок 5.4. График функции $Z=\sin(t)\cdot\cos(t)$ *, построенный при помощи команды genfac3d*

ЗАДАЧА 5.4

Построить график поверхности, заданной следующей системой уравнений $\begin{cases} x=p1\cdot\sin(p1)\cdot\cos(p2) \\ y=p1\cdot\cos(p1)\cdot\cos(p2) \\ z=p1\cdot\sin(p2) \end{cases}$, используя команду *eval3dp*.

Прежде всего, определим массивы значений параметров *p1* и *p2*. Далее создадим функцию *scp*, которая задает график.

Напомним, что функции в Scilab создаются при помощи команды *deff*:

```
deff([s1,s2,...]=newfunction(e1,e2,...)'
```

где *s1,s2,...* список выходных переменных , которым будет присвоен конечный результат вычислений;

newfunction имя создаваемой функции ,оно будет использоваться для ее вызова ;

e1,e2,... входные параметры .

Обратите внимание, что команда *deff* записана в три строки только для удобства чтения листинга (см листинг 5.4).

Теперь сформируем прямоугольную сеть при помощи команды *eval3dp* и построим график, обратившись к функции *plot3d* (см. рис. 5.5).

```
p1=linspace(0,2*%pi,10);
p2=linspace(0,2*%pi,10);
deff("[x,y,z]=scp(p1,p2)",["x=p1.*sin(p1).*cos(p2)";
"y=p1.*cos(p1).*cos(p2)";
"z=p1.*sin(p2)"]);
[Xf,Yf,Zf]=eval3dp(scp,p1,p2);
plot3d(Xf,Yf,Zf);
```

Листинг 5.4

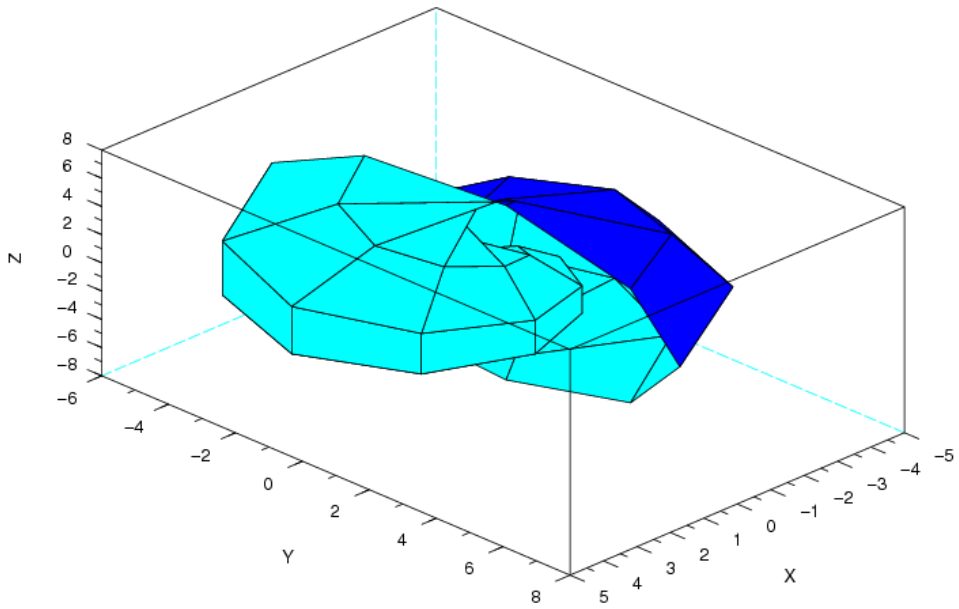

Рисунок 5.5. График, построенный при помощи команды eval3dp

В Scilab также существуют несколько других функций для построения поверхностей. Они имеют более простой синтаксис Matlab, однако, по мнению авторов, не всегда могут заменить функцию *plot3d*.

5.2 Функции meshgrid, surf и mesh

Для формирования прямоугольной сетки впервые в Scilab 4.0 появилась функция *meshgrid*. Обращение к ней имеет вид:

```
[X,Y[Z]]=meshgrid(x,y[z])
```

здесь *x,y [z]* массивы 2 (3)исходных параметров X, Y (Z),указываемые через запятую ;

X, Y [Z] матрицы в случае 2и массивы в случае 3входных величин .

После формирования сети вывести в нее графику можно с помощью функции *surf* либо *mesh*. Так же, как и в случае с функциями *plot3d* и *plot3d1*, *surf* строит поверхность, заливая каждую ячейку цветом, который зависит от конкретного значения функции в узле сетки, а *mesh* заливает ее одним цветом.

Таким образом, *mesh* является полным аналогом функции *surf* со значением параметров *Color mode=индекс белого цвета* в текущей палитре цветов и *Color flag=0*.

Обращение к функциям имеет вид:

```
surf([X,Y],Z,[color, keyn=valuen])
mesh([X,Y],Z,[color, ])
```

здесь *X,Y* массивы ,задающие прямоугольную сеть ;

Z матрица значений функции ;

color матрица действительных чисел ,устанавливающих цвет для каждого узла сети ;

keyn=valuen - последовательность значений свойств графика *key1=value1, key2=value2,...,keyn=valuen,* определяющих его внешний вид (см. параграф 4.6).

Конечно, в том случае, если прежде прямоугольная сеть была построена командой *meshgrid* необходимости указывать параметры *X, Y* нет. В самом простейшем случае к функции `surf` можно обратиться так *surf(z).*

ЗАДАЧА 5.5

Построить график функции $Z=5y^2-x^2$ с помощью команды *mesh.*

С помощью команды *meshgrid* создадим прямоугольную сетку. Здесь -2:2 определяет положение прямых, параллельных оси X, а -3:3 оси Y.

После формирования сетки вычислим значения функции Z во всех узлах и обратимся к функции *mesh* для построения графика (см. листинг 5.5, рис. 5.6).

```
[x y]=meshgrid(-2:2,-3:3);
z=5*y.^2-x.^2;
mesh(x,y,z);
```

Листинг 5.5

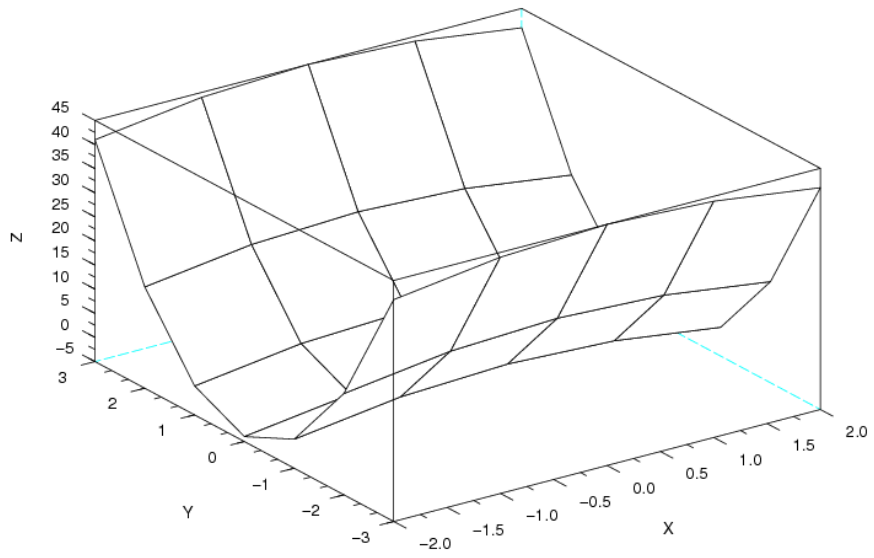

Рисунок 5.6. График функции $Z=5y^2-x^2$ *, построенный командой mesh*

Как видно из рисунка 5.6, сетка, построенная с шагом 1, слишком редкая, а вычисленных значений функции в узлах недостаточно для изображения плавного графика. Поэтому зачастую лучше самостоятельно указывать шаг формирования прямоугольной сети при вызове команды *meshgrid.*

ЗАДАЧА 5.6

Построить график функции $Z=5y^2-x^2$ с помощью команды *surf.*

Создадим с помощью команды *meshgrid* прямоугольную сеть, указывая шаг, через

который будут построены параллельные координатным осям линии 0,1 для обеих осей . В этом случае, сетка будет плотнее, а график более плавным ,чем в предыдущем примере .

Далее вычисляем значения функции *Z* и вызываем *surf* для построения поверхности (см. листинг 5.6, рис.).

```
[x y]=meshgrid(-2:0.1:2,-3:0.1:3);
z=5*y.^2-x.^2;
surf(x,y,z)
```

Листинг 5.6

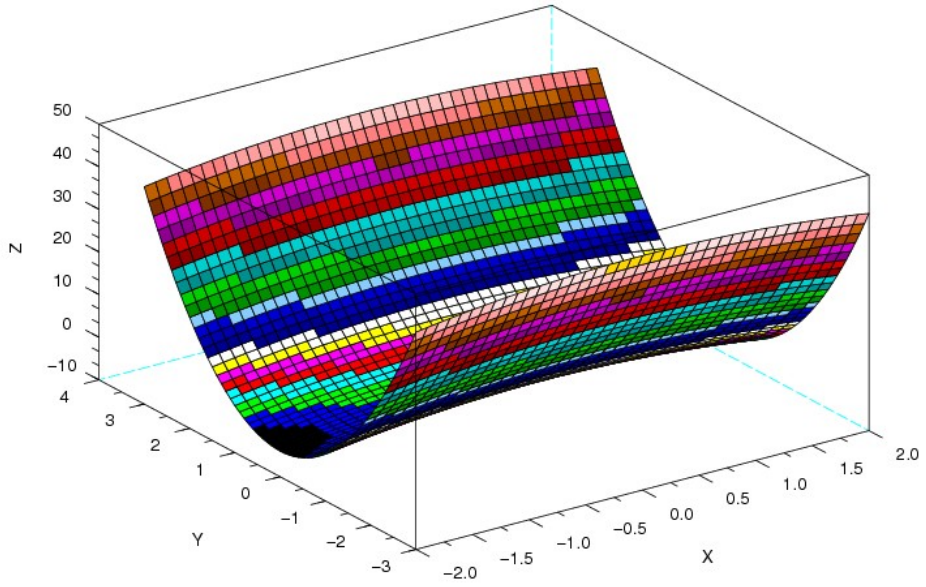

Рисунок 5.7. График функции $Z=5y^2-x^2$ *, построенный командой surf*

На первый взгляд может показаться, что для функции $Z=5y^2-x^2$ *plot3d1* и *surf* построили разные поверхности (см. рис. 5.7 и рис. 5.2). Однако это не так. Различие обусловлено использованием по умолчанию функцией *surf* режима масштабирования *Cube scaling* (см. рис. 5.8). Если его отключить, *surf* выведет изображение, идентичное, построенному с помощью функции *plot3d* (см. рис. 5.9).

В Scilab можно построить графики двух поверхностей в одной системе координат. Для этого следует использовать команду *mtlb_hold('on')*, которая блокирует создание нового графического окна при выполнении команд *surf* или *mesh*.

Проиллюстрируем это примером.

ЗАДАЧА 5.7

Построить график функции $\begin{cases} z(x,y)=(3x^2+4y^2)-1 \\ z1(x,y)=-(3x^2+4y^2)-1 \end{cases}$, используя команду *mtlb_hold('on')*.

Сформируем плотную прямоугольную сеть с помощью команды *meshgrid*.

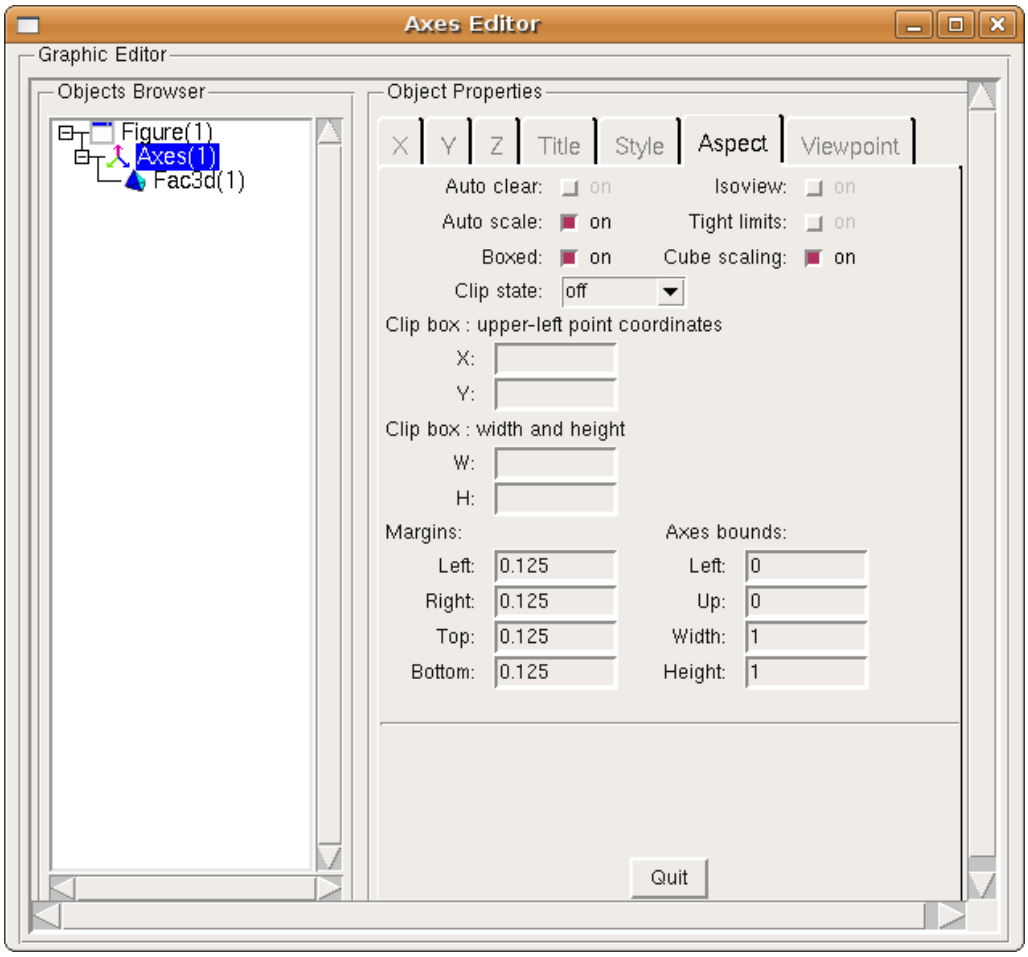

Рисунок 5.8. Режим Cube scaling окна форматирования Axes Editor

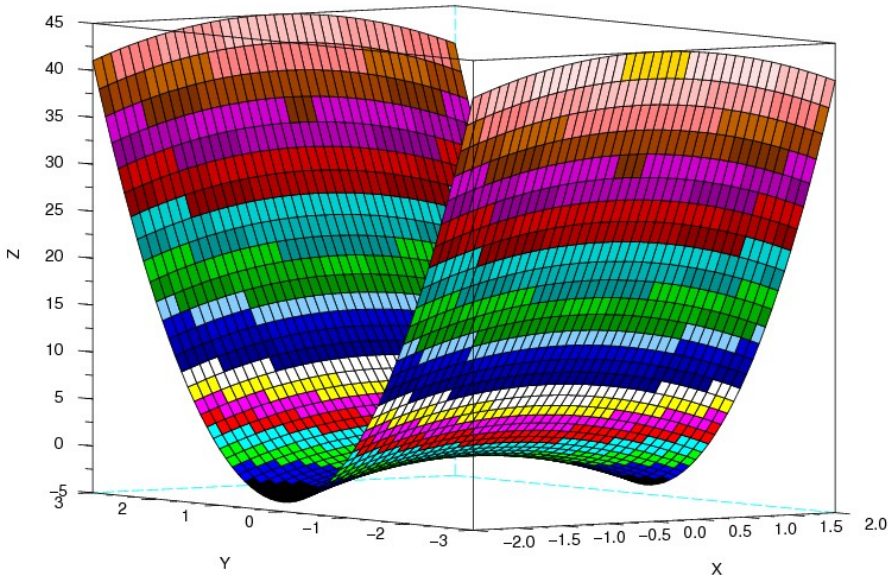

Рисунок 5.9. График функции $Z=5y^2-x^2$ *при отключенном режиме Cube scaling*

Пусть $z=+(3x^2+4y^2)-1$ и $z1=-(3x^2+4y^2)-1$. Вычислим значения функций во всех узлах сети.

Поверхность $z=+(3x^2+4y^2)-1$ построим с помощью команды *surf*, каждая ее ячейка будет залита цветом, зависящим от значения функции в узле сетки. Далее вызовем команду *mtlb_hold('on')*, которая заблокирует создание нового графического окна, и с помощью команды *mesh* построим поверхность $z1=-(3x^2+4y^2)-1$ в одних координатных осях с поверхностью *Z*, при этом будет выведена прямоугольная сетка, а ячейки залиты белым цветом (см. листинг 5.7, рис. 5.10).

```
[x y]=meshgrid(-2:0.2:2,-2:0.2:2);
z=3*x.^2+4*y.^2-1;
z1=-3*x.^2-4*y.^2-1;
surf(x,y,z);
mtlb_hold('on');
mesh(x,y,z1);
```

Листинг 5.7

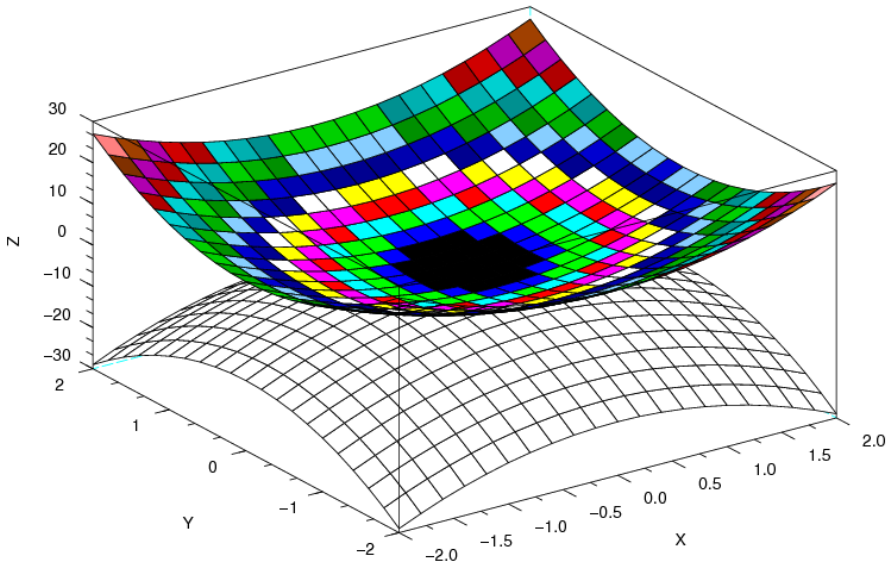

Рисунок 5.10. График функции $z(x,y)=\pm(3x^2+4y^2)-1$

5.3 Функции plot3d2 и plot3d3

Функции *plot3d2* и *plot3d3* являются аналогами функции *plot3d*, поэтому имеют такой же синтаксис:

```
plot3d2(x,y,z,[theta,alpha,leg,flag,ebox][keyn=valuen]),
plot3d3(x,y,z,[theta,alpha,leg,flag,ebox][keyn=valuen])
```

Эти функции предназначены для построения поверхности, которая задается набором граней. Т.е. если функция *plot3d* по входным данным сможет построить лишь отдельно стоящие друг от друга плоские грани, то *plot3d2 (plot3d3)* проинтерпретирует взаимное расположение этих граней в виде цельного геометрического тела.

Отличие функций *plot3d2* и *plot3d3* сходно с различием действия функций *plot3d* и *plot3d1*, а также *surf* и *mesh*. *Plot3d2* строит поверхность, при этом выводит сетку и заливает все ячейки одним из цветов, по умолчанию синим . *Plot3d* также выводит сетку, однако оставляет все ячейки без заливки (т.е. Белыми)

ЗАДАЧА 5.8

Построить сферу $\begin{cases} x(u,v)=\cos(u)\cos(v); \\ y(u,v)=\cos(u)\sin(v); \\ z(u,v)=\sin(u). \end{cases}$ при помощи функции *plot3d2*.

При построении графиков поверхностей, заданных параметрически x(u,v), y(u,v) и *z(u,v)* необходимо сформировать матрицы X, Y и Z одинакового размера. Для этого массивы u и v должны иметь одинаковый размер. После этого следует выделить два основных вида представления x, y и z в случае параметрического задания поверхностей.

Первый базируется натом, что x, y и z можно представить в виде $f(u)\cdot g(v)$, тогда соответствующие им матрицы X, Y и Z следует формировать в виде матричного умножения $f(u)$ на $g(v)$.

В противном случае - если x, y и z можно представит в виде $f(u)$ или $\cdot\ g(v)$ - то матрицы X, Y и Z следует записывать в виде $f(u)\cdot ones(size(v))$ или $g(v)\cdot ones(size(u))$ соответственно.

В этом примере также используется функция *linspace* (см. листинг 5.8). Эта функция, возвращает массив с линейным приращением значений в заданном диапазоне.

Например, *u=linspace(-%pi/2, %pi/2, 40)* значит, что параметр *u* линейно изменяется в диапазоне [-2π;2π]. Число 40 устанавливает количество значений, которое массив должен содержать 40,по умолчанию их 100.

Построенная функцией *plot3d2* сфера представлена на рис. 5.11.

```
u = linspace(-%pi/2,%pi/2,40);
v = linspace(0,2*%pi,20);
X = cos(u)'*cos(v);
Y = cos(u)'*sin(v);
Z = sin(u)'*ones(v);
plot3d2(X,Y,Z);
```

Листинг 5.8

Теперь посмотрим, как эту же задачу выполнит функция *plot3d*.

ЗАДАЧА 5.9

Построить сферу $\begin{cases} x(u,v)=\cos(u)\cos(v); \\ y(u,v)=\cos(u)\sin(v); \\ z(u,v)=\sin(u). \end{cases}$ с помощью функции *plot3d*.

Определим параметры u и v, вычислим значения функций *x, y, z,* как и в предыдущем примере (см. листинг 5.9). Однако для построения графика обратимся к функции *plot3d*. Получим следующее изображение (см. рис. 5.12).

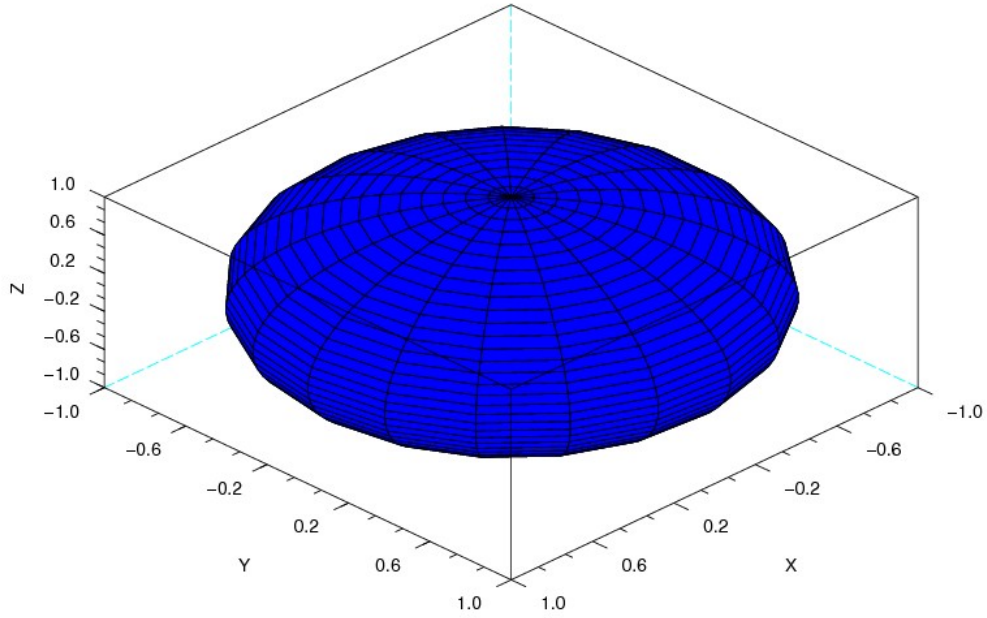

Рисунок 5.11. График сферы, построенный функцией plot3d2

```
u = linspace(-%pi/2,%pi/2,40);
v = linspace(0,2*%pi,20);
X = cos(u)'*cos(v);
Y = cos(u)'*sin(v);
Z = sin(u)'*ones(v);
plot3d(X,Y,Z);
```

Листинг 5.9

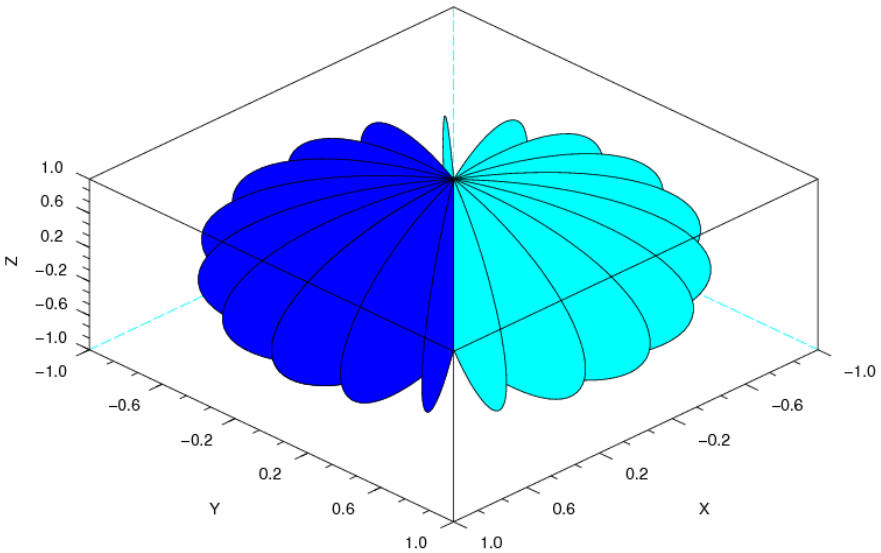

Рисунок 5.12. График, построенный функцией plot3d

Теперь проиллюстрируем действие функции *plot3d3* на этом же примере.

ЗАДАЧА 5.10

Построить сферу $\begin{cases} x(u,v)=\cos(u)\cos(v); \\ y(u,v)=\cos(u)\sin(v); \\ z(u,v)=\sin(u). \end{cases}$ при помощи функции *plot3d3*.

Определим диапазоны изменения параметров *u* и *v*, как и в предыдущих примерах. Уменьшим лишь количество значений для массива *u* с 40 до 20 так график будет менее загроможден.

Вычислим значения функций *x, y, z* и обратимся для изображения графика к команде *plot3d3* (см. листинг 5.10). Обратите внимание, что заливка ячеек полученной сферы отсутствует, она прозрачна (см. рис. 5.13).

```
u = linspace(-%pi/2,%pi/2,20);
v = linspace(0,2*%pi,20);
X = cos(u)'*cos(v); Y=cos(u)'*sin(v);
Z = sin(u)'*ones(v);
plot3d3(X,Y,Z);
```

Листинг 5.10

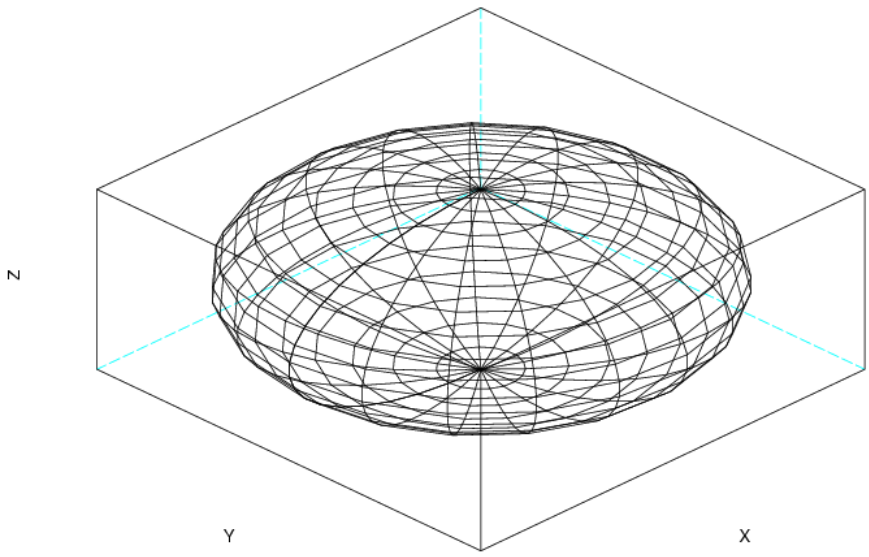

Рисунок 5.13. График сферы, построенный функцией plot3d3

5.4 Функции param3d и param3d1

Для построения параметрической кривой в Scilab существует команда *param3d*:
`param3d(x,y,z,[theta,alpha,leg,flag,ebox])`.
Проиллюстрируем возможности функции *param3d* следующими примерами.

ЗАДАЧА 5.11

Построить график линии, заданной параметрически $\begin{cases} y=\sin(t) \\ y1=\cos(t) \\ y2=\dfrac{t}{7} \end{cases}$.

Прежде всего, определим диапазон и шаг изменения параметра *t*.

Затем обратимся к функции *param3d*, передав ей математические выражения функций *y*, *y1* и *y2*, а также углы в градусах, под которыми наблюдатель будет видеть формируемую графику 45и 35 (см .листинг 5.11, рис. 5.14).

```
t=[0:0.1:10*%pi];
param3d(sin(t),cos(t),t/7,45,35);
```

Листинг 5.11

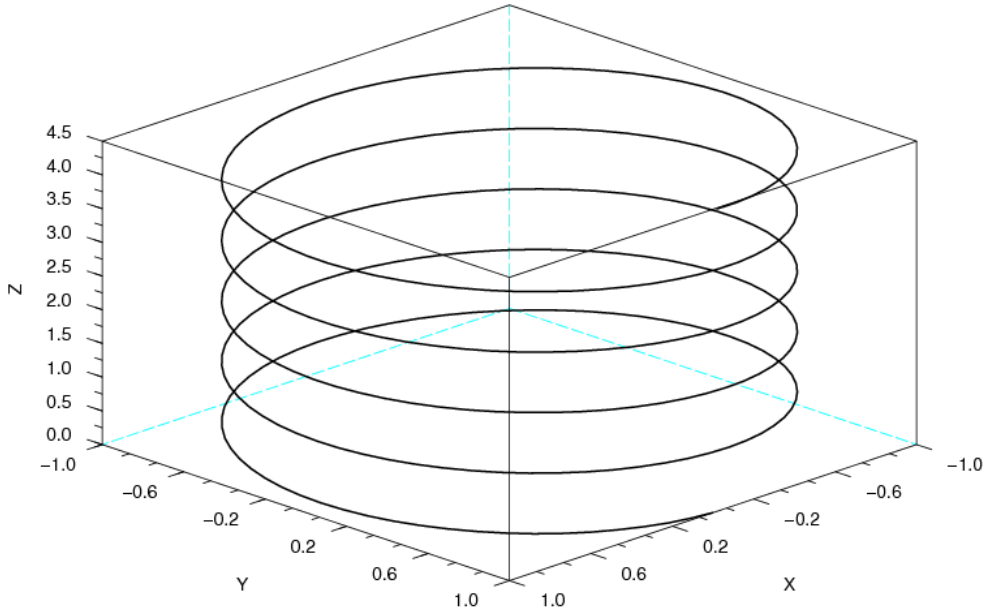

Рисунок 5.14. График параметрической кривой, построенный функцией param3d

ЗАДАЧА 5.12

Построить линию, заданную параметрически	$\begin{cases} x = t \cdot \sin(t); \\ y = t \cdot \cos(t); \\ z = \dfrac{t \cdot \lvert t \rvert}{(50\pi)}. \end{cases}$

Определив массив значений параметра *t*, вычислим значения X, Y и Z координат кривой.

Для построения графика используем команду *param3d*, установив углы обозрения наблюдателя 45и 60 (см .листинг 5.12, рис. 5.15).

```
t=-50*%pi:0.1:50*%pi;
x=t.*sin(t);
y=t.*cos(t);
z=t.*abs(t)/(50*%pi);
param3d(x,y,z,45,60);
```

Листинг 5.12

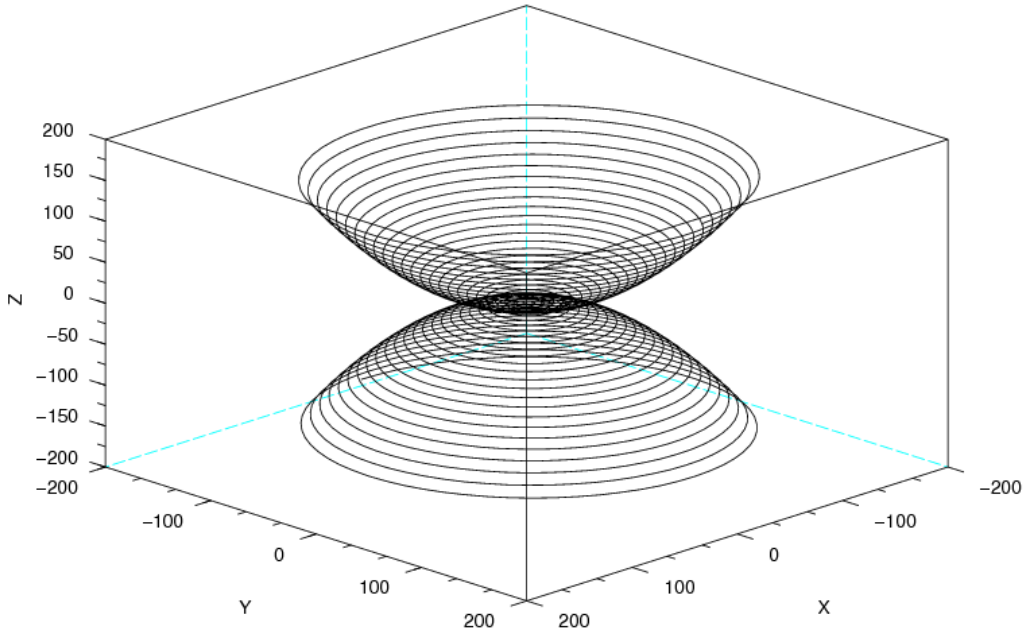

Рисунок 5.15. График параметрической линии, построенный функцией param3d

Для вывода нескольких параметрически заданных кривых в одних координатах в Scilab используется функция *param3d1*. Она имеет несколько отличный синтаксис:

```
param3d1(x,y,list(z,colors),[theta,alpha,leg,flag,ebox])
```

Здесь впервые появляется необходимость использования конструкции *list(z,colors)*, которая позволяет задавать не только Z-координату для каждой из кривых, но и устанавливать для них желаемый цвет. Рассмотрим это на примере.

ЗАДАЧА 5.13

Построить графики линий, заданных параметрически:	$\begin{cases}\sin(t);\\ \sin(2t);\\ t/10.\end{cases}$ и $\begin{cases}\cos(t);\\ \cos(2t);\\ \sin(t).\end{cases}$

Зададим массив значений параметра *t*.

Для построения графиков линий в одной системе координат обратимся к функции *param3d1*. В качестве параметров в первых квадратных скобках передадим ей X и Y координаты первой кривой, а во вторых второй . При помощи свойства *list* определяем Z-координаты и для первой кривой установим темно-синий цвет линии 9, а для второй красный 5. Числа 35 и 45 углы поворота наблюдателя. Параметр *"X@Y@Z"* отвечает за вывод подписей осей графика (см. листинг 5.13, рис. 5.16).

```
t=[0:0.1:5*%pi]';
param3d1([sin(t),sin(2*t)],
[cos(t),cos(2*t)],list([t/10,sin(t)],[9,5]),35,45,"X@Y@Z");
```

Листинг 5.13

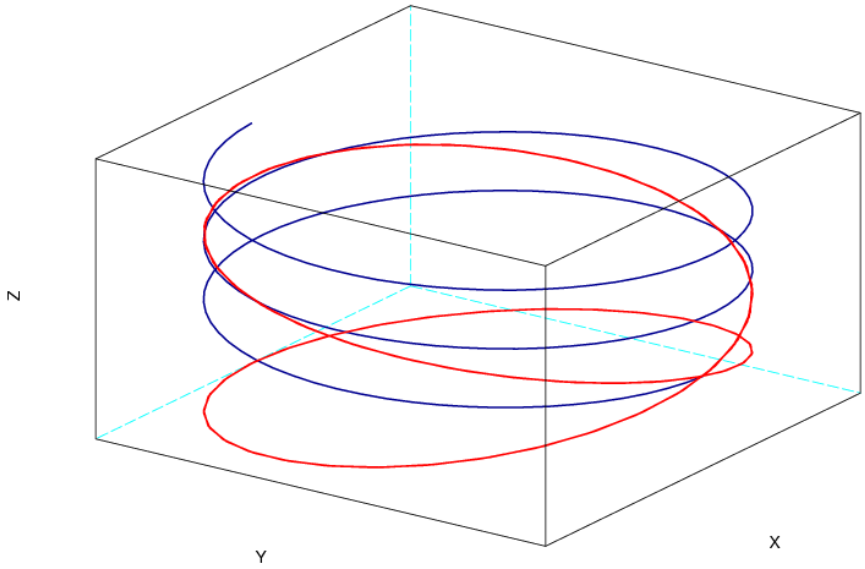

Рисунок 5.16. Графики параметрических кривых, построенные функцией param3d1

5.5 Функция contour

В Scilab кроме построения объемной графики также реализована возможность создания пространственных моделей объектов. На практике часто возникает необходимость построения карт в изолиниях значений показателя, где X, Y координаты задают положение конкретной изучаемой точки на плоскости, а Z-координата зафиксированную величину показателя в этой точке.

Точки с одинаковыми значениями показателя соединяют, так называемые, изолинии линии одинаковых уровней значений исследуемой величины.

Для построения изолиний Scilab существует функция *contour*. Обращение к ней имеет вид:

```
contour(x,y,z,nz[theta,alpha,leg,flag,ebox,zlev])
```

Здесь *x, y* массивы действительных чисел ;

z матрица действительных чисел значения функции ,описывающей поверхность $Z(x, y)$;

nz параметр , который устанавливает количество изолиний . Если целое число ,то в диапазоне между минимальным и максимальным значениями функции $Z(x, y)$ через равные интервалы будут проведены `nz` изолиний. Если же задать nz как массив, то изолинии будут проводиться через все указанные в массиве значения;

theta, alpha действительные числа , которые определяют в градусах сферические координаты угла обозрения наблюдателя. Попросту говоря, это угол, под которым наблюдатель видит отображаемую поверхность;

leg подписи координатных осей графика символы ,отделяемые знаком @Например , 'X@Y@Z'.

flag – массив, состоящий из 3 целочисленных параметров *[mode,type,box]*. Здесь *mode* устанавливает способ и место нанесения линий уровня (см. табл. 5.4).

Таблица 5.4. Значения параметра `mode`

Значение	Описание
0	Изолинии наносятся на поверхность

	$Z(x, y)$
1	Изолинии наносятся на поверхность и план, который задается уравнением $Z = zlev$
2	Изолинии наносятся на двумерный график

type позволяет управлять масштабом графика (см. табл. 5.2), по умолчанию имеет значение 2;

box определяет наличие рамки вокруг отображаемого графика (см. табл. 5.3). По умолчанию равен 4;

ebox определяет границы области, в которую будет выводиться поверхность, как вектор *[xmin,xmax,ymin,ymax,zmin,zmax]*. Этот параметр может использоваться только при значении параметра *type=1;*

zlev математическое выражение, которое задает план (горизонтальная проекция заданной поверхности) для построения изолиний, по умолчанию, совпадает с уравнением, описывающим плоскость, в этом случае может не указываться.

Следует отметить, что функции *contour* уравнение поверхности $Z(x, y)$ удобнее передавать в качестве параметра как функцию, определенную пользователем.

Напомним, что функции в Scilab создаются при помощи команды *deff*:

```
deff('[s1,s2,...]=newfunction(e1,e2,...)
```

где *s1,s2,...* список выходных параметров, то есть переменных, которым будет присвоен конечный результат вычислений;

newfunction имя создаваемой функции ,оно будет использоватся для ее вызова ;

e1,e2,... входные параметры .

Второй способ создания функции - это примение конструкции вида:

```
function <lhs_arguments>=<function_name><rhs_arguments>
<тело функции>
endfunction
```

где *lhs_arguments* список выходных параметров ;

function_name имя создаваемой функции ;

rhs_arguments входные параметры .

ЗАДАЧА 5.14

Построить линии уровня поверхности $Z = x \cdot \sin(x)^2 \cdot \cos(y)$ при помощи функции *contour.*

Введем параметр *t* и определим массив его значений. При помощи команды *function* создадим функцию *my_surface* с входными данными - *x, y* и выходными *z*. В теле функции вычислим значения математическое выражение, задающее поверхность.

Для построения изолиний обратимся к функции *contour* (см листинг 5.14, рис.5.17).

```
t=linspace(-%pi,%pi,30);
function z=my_surface(x,y)
z=x*sin(x)^2*cos(y)
endfunction
contour(t,t,my_surface,10);
```

Листинг 5.14

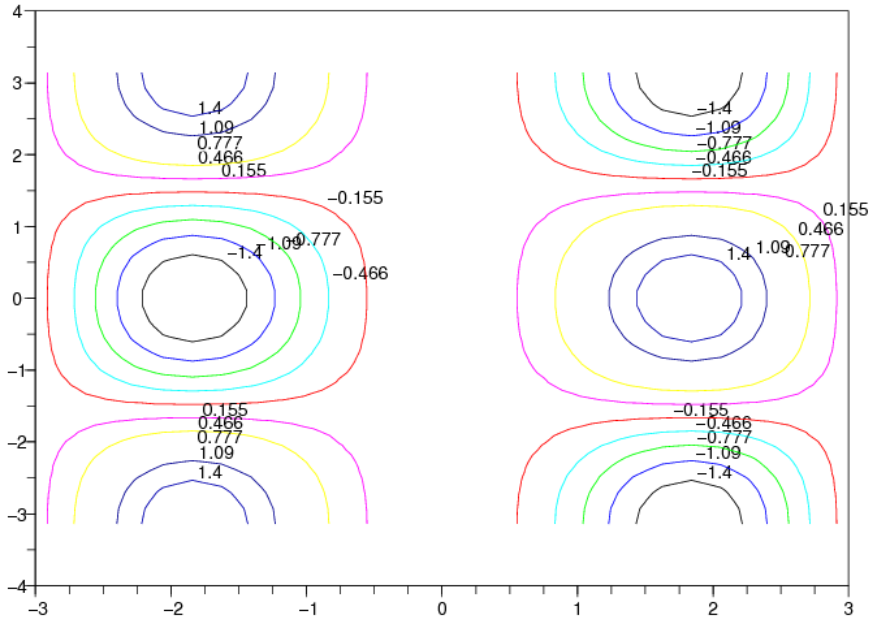

Рисунок 5.17. Изолинии поверхности $Z = x \cdot \sin(x)^2 \cdot \cos(y)$

Этот пример показывает, что выполнение функции *contour* приводит к формированию линий одинаковых значений показателя и проецированию их на горизонтальную плоскость. Очевидно, что такое представление данных малоинформативно. Гораздо наглядней изображение изолиний поверхности и собственно поверхности в одном графическом окне.

ЗАДАЧА 5.15

Построить поверхность $Z = \sin(x) \cdot \cos(y)$ и вывести изолинии в одном графическом окне.

Прежде всего, введем параметр *t* и сформируем массив его значений.

Создадим функцию *Surf*, обратившись к команде *deff*.

С помощью команды *rect* установим границы области построения графики в графическом окне для того, чтобы стало возможным совместить и поверхность, и спроецированные на горизонтальную плоскость изолинии поверхности.

Напомним, что при построении графика функции вида $Z(x, y)$ при помощи функции *plot3d* необходимо использовать оператор цикла *For*, формировать матрицу значений функции $z_{ij} = f(x_i, y_j)$. Чтобы избежать этого, воспользуемся командой *feval*.

Далее с помощью функции *plot3d* строим график поверхности $Z = \sin(x) \cdot \cos(y)$, устанавливая углы обзора наблюдателя, подписи для координатных осей. Определяем и массив *flag [2,1,4]*: 2 цвет графика синий , 1 границы области построения графики определяются вручную (далее указан параметр *rect*, заданный выше), 4 выводятся все оси и рамка вокруг графика.

Затем формируем изолинии, обратившись к функции *contour*, также устанавливаем углы обзора наблюдателя, подписи координатных осей, число формируемых изолиний 10 и значения массива *flag [1,1,4]*: 1 режим вывода изолиний на отдельно построенный план , который задается тем же уравнением , что и поверхность - $Z = \sin(x) \cdot \cos(y)$, 1 - границы области построения графики определяются вручную (далее указан параметр `rect`, заданный выше), 4 выводятся все оси и рамка вокруг графика . Число -5 устанавливает положение горизонтальной плоскости с изолиниями 5единиц ниже графика поверхности .

С помощью команды *xtitle* выведем подпись для графика (см. листинг 5.15, рис. 5.18)

```
t=%pi*(-10:10)/10;
deff('[z]=Surf(x,y)','z=sin(x)*cos(y)');
```

```
rect=[-%pi,%pi,-%pi,%pi,-5,1];
z=feval(t,t,Surf);
plot3d(t,t,z,35,45,'X@Y@Z',[2,1,4],rect);
contour(t,t,z,10,35,45,'X@Y@Z',[1,1,4],rect, -5);
xtitle('plot3d and contour');
```

Листинг 5.15

Однако и такое изображение поверхности и ее изолиний не всегда бывает удобным. Попробуем совместить график поверхности и ее линии уровня.

ЗАДАЧА 5.16

Совместить график поверхности $Z=\sin(x)\cdot\cos(y)$ с ее изолиниями.

Как и в предыдущем примере зададим массив значений параметра *t*, создадим функцию *Surf*, ограничим область для вывода графики внутри графического окна с помощью команды *rect*, а также вычислим значения функции $Z=\sin(x)\cdot\cos(y)$, выполнив команду *feval*.

При построении поверхности оставим все параметры без изменений, кроме углов обзора наблюдателя, установим 75 и 45, а также цвета заливки графика, установим значение параметра *mode* в массиве *flag* равным -19 коричневый цвет .

При обращении к функции *contour* для совмещения поверхности и ее изолиний удалим значение параметра *position* -5 и установим для режима *mode* в массиве *flag* равным 0 - изолинии наносятся непосредственно на поверхность $Z=\sin(x)\cdot\cos(y)$.

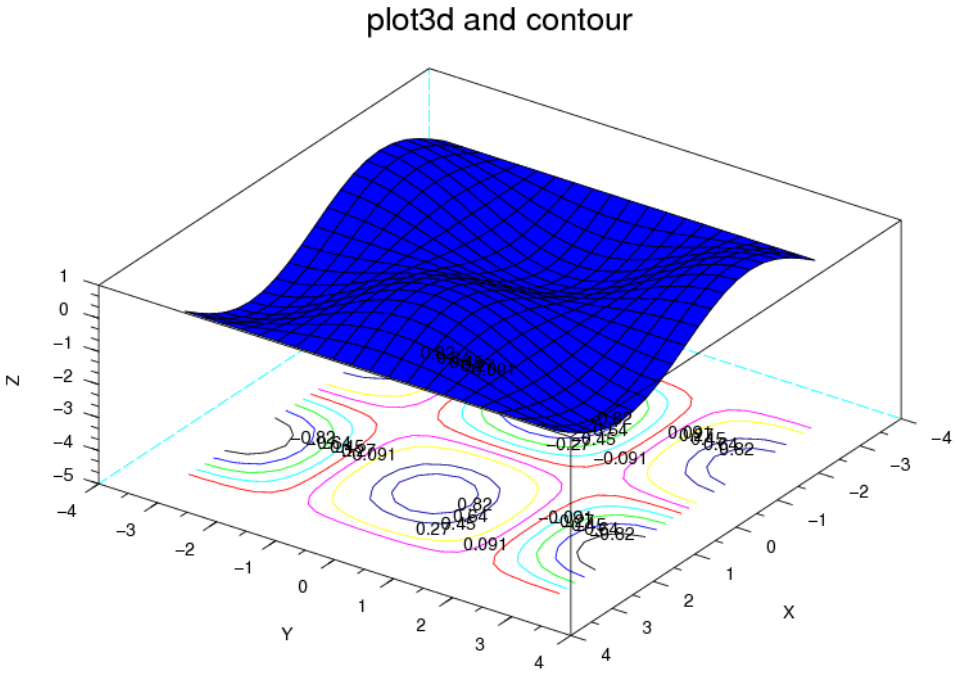

Рисунок 5.18. График поверхности $Z=\sin(x)\cdot\cos(y)$ *и ее изолинии в одном окне*

С помощью команды *xtitle* также выводим подпись для графика (см. листинг 5.16, рис. 5.19).

```
t=%pi*(-10:10)/10; deff('[z]=Surf(x,y)','z=sin(x)*cos(y)');
rect=[-%pi,%pi,-%pi,%pi,-1,1];
z=feval(t,t,Surf);
plot3d(t,t,z,35,45,'X@Y@Z',[-19,1,4],rect);
contour(t,t,z+0.1,10,35,45,'X@Y@Z',[0,1,4],rect);
```

```
xtitle('plot3d and contour');
```

Листинг 5.16

5.6 Функция contourf

В Scilab существует функция *contourf*, которая изображает поверхность на горизонтальной плоскости не просто и в виде изолиний, но и заливает интервалы между ними цветом, в зависимости от конкретного уровня значений показателя.

Обращение к функции имеет вид:

```
contourf (x,y,z,nz,[style,strf,leg,rect,nax])
```

Здесь *x, y* массивы действительных чисел ; *z* матрица действительных чисел значения функции, описывающей поверхность *Z(x,y)*;

nz параметр ,который устанавливает количество изолиний .Если *nz* целое число ,то в диапазоне между минимальным и максимальным значениями функции $Z(x, y)$ через равные интервалы будут проведены `nz` изолиний. Если же задать *nz* как массив, то изолинии будут проводиться через все указанные в этом массиве значения;

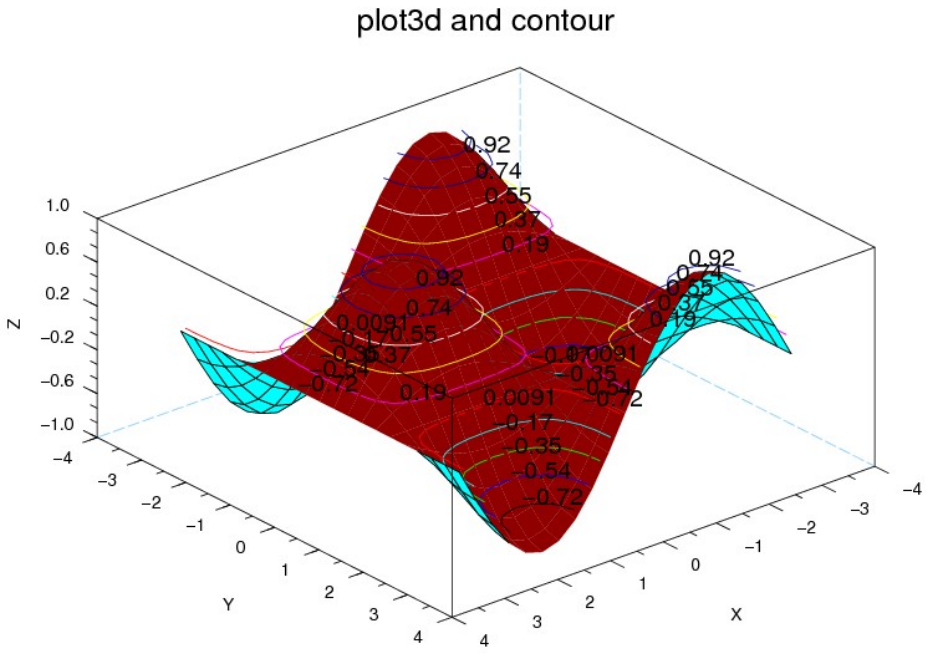

Рисунок 5.19. Совмещение поверхности и ее изолиний

style массив того же размера , что и *nz* устанавливает цвет для каждого интервала уровней значений;

strf строка ,состоящая из трех чисел -*"csa"*. Здесь *c (Captions)* устанавливает режим отображения подписей графика (см табл. 5.5); *s (Scaling)* режим масштабирования (см . табл. 5.6); *a (Axes)* определяет положение осей графика (см .табл .5.7).

Таблица 5.5. Значение параметра c (Captions) строки strf

Значение	Описание
0	нет подписей
1	отображаются подписи, заданные параметров *leg*

Таблица 5.6. Значение параметра s (Scaling) строки strf

Значение	Описание
0	масштабирование по умолчанию
1	устанавливается параметром *rect*
2	масштаб зависит от минимального и максимального значения входных данных
3	выводятся изометрические оси, исходя из значений параметра *rect*
4	выводятся изометрические оси, исходя из входных данных
5	расширение осей для наилучшего вида, исходя из значений параметра *rect*
6	расширение осей для наилучшего вида, исходя из входных данных

Таблица 5.7. Значение параметра a (Axes) строки strf

Значение	Описание
0	нет осей
1	выводятся оси, ось Y слева
2	выводится рамка вокруг графика без делений
3	выводятся оси, ось Y справа
4	оси центрируются в графической области окна
5	выводятся оси таким образом, чтоб они пересекались в точке (0;0).

leg легенда графика ,подпись каждой из кривых -символы ,отделяемые знаком @.По умолчанию, " ".

rect вектор *[xmin, ymin, xmax, ymax]*, который определяет границы изменения *x* и *y* координат графической области окна;

nax - это массив из четырех значений *[nx, Nx, ny, Ny]*, определяющий число основных и промежуточных делений координатных осей графика. Здесь *Nx (Ny)* число основных делений с подписями под осью X (Y); *nx (ny)* число промежуточных делений .

ЗАДАЧА 5.17

Построить изображение поверхности $Z=\sin(x)\cdot\cos(y)$ с помощью функции *contourf*.

Введем параметр *t* и создадим массив его значений, определим при помощи команды *deff* функцию *surf*.

Для наглядности приведем график поверхности $Z=\sin(x)\cdot\cos(y)$, построенный функцией *plot3d1*, и ее изображение на горизонтальной плоскости, сформированное функцией contourf в одном графическом окне. С этой целью обратимся к команде *subplot*, которой разобьем графическое окно на две области для вывода графики.

Используя *feval*, вычислим значения функции $Z=\sin(x)\cdot\cos(y)$ и построим ее график при помощи *plot3d1*, указав углы обозрения наблюдателя 80 и 15, а также вызвав

команду *xtitle* выведем подпись графика *'plot3d1"*.

Теперь сформируем проекцию поверхности на горизонтальную плоскость посредством функции *contourf*. В качестве параметров ей передаем X, Y и Z координаты, 10 число изолиний, 10:20 массив , определяющий цвет каждого интервала между изолиниями , а также значения строки *strf="121"*: 1 режим отображения подписей ; 2 выбор масштаба зависит от минимального и максимального значения входных данных; 1 - режим отображения координатных осей, ось Y находится слева.

Обратите внимание, в этой задаче мы впервые создали шкалу цвета. Ее использование часто облегчает чтение графики. Для ее вывода в Scilab существует команда *colorbar (n, m)*, здесь *n* минимальное значение диапазона , *m* максимальное значение .

Выведем и для этого графика подписи осей и графика в целом *'contourf"* при помощи команды *xtitle* (см. листинг 5.17, рис. 5.20)

```
t=-%pi:0.2:%pi;
deff('[z]=Surf(x,y)','z=sin(x)*cos(y)');
subplot(121);
z=feval(t,t,Surf);
plot3d1(t,t,z,80,15);
xtitle('plot3d1');
subplot(122);
contourf(t,t,z,10,10:20,strf="121");
colorbar(-%pi,%pi);
xtitle('contourf','X','Y');
```

Листинг 5.17

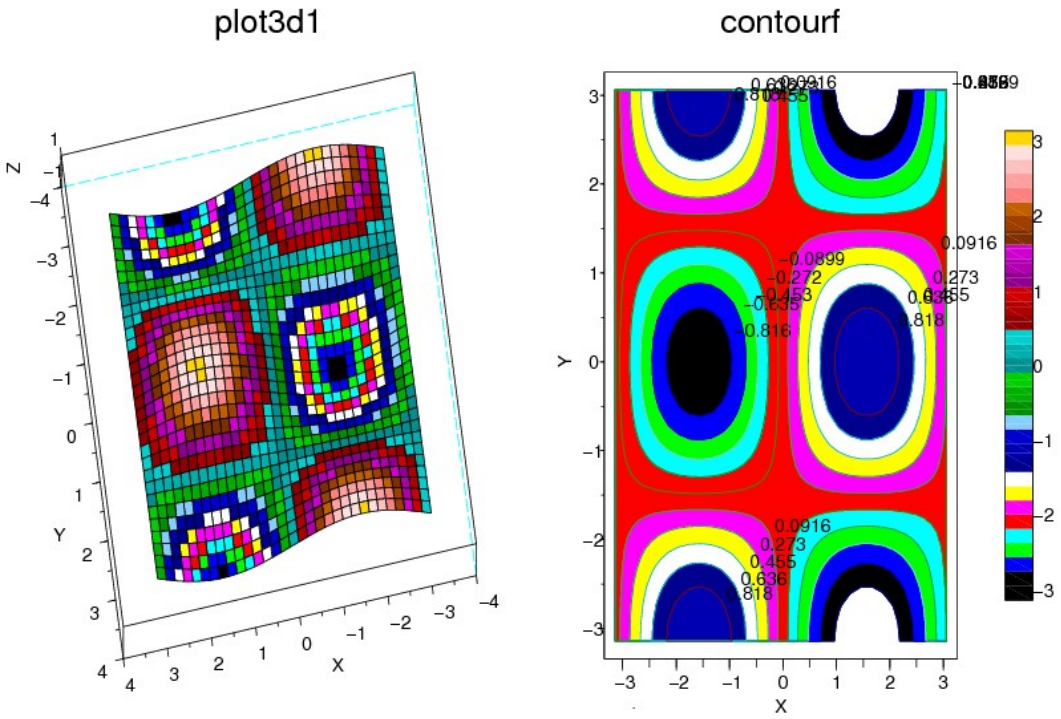

Рисунок 5.20. График функции $Z=\sin(x)\cdot\cos(y)$ *, построенный функцией plot3d1 и contourf.*

5.7 Функция hist3d

Для построения трехмерных гистограмм в Scilab используется функция *hist3d*:

```
hist3d(f,[theta,alpha,leg,flag,ebox])
```

Здесь f матрица (m:n),задающая гистограмму $f(i,j)=F(x(i),y(j))$.

Параметры *theta,alpha,leg,flag,ebox* управляют теми же свойствами, как и у функции *plot3d*.

ЗАДАЧА 5.18

Построить трехмерную гистограмму при помощи команды *hist3d*.

Для формирования матрицы входных данных воспользуемся командой *rand*. Напомним, чтобы создать матрицу размером *(m,n)*, необходимо использовать конструкцию *rand(m,n)* (см. листинг 5.18).

Полученная гистограмма изображена на рис. 5.21.

```
hist3d(9.7*rand(10,10),20,35);
```

Листинг 5.18

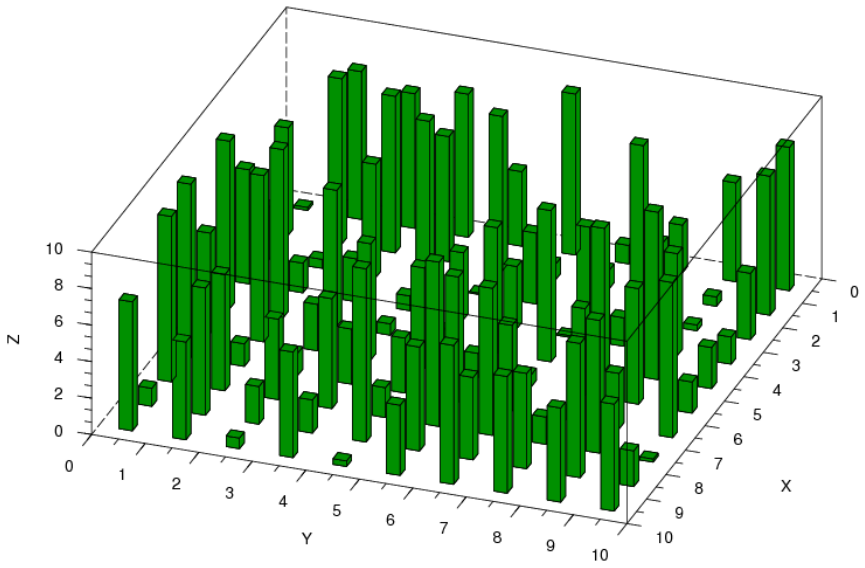

Рисунок 5.21. Трехмерная гистограмма, построенная функцией hist3d

5.8 Примеры построения некоторых трехмерных графиков в Scilab

В этом параграфе мы рассмотрим приемы построения некоторых нестандартных трехмерных графиков средствами Scilab.

Прежде всего, научимся вырезать из графики «ненужные» части.

ЗАДАЧА 5.19

Построить поверхность $Z=\sin(t)\cdot\cos(t)$, вырезать из графика области, где $\lvert Z\rvert>0.5$.

Сформируем массив значений параметра *t*, вычислим значения функции $Z=\sin(t)\cdot\cos(t)$ и запишем их в массив Z.

В массив Z1 при помощи команды *find* запишем индексы тех элементов массива Z, чье модальное значение больше 0,5.

Далее мы будем использовать функцию *%inf*. Она предназначена для определения бесконечных элементов массива, поэтому запись *z(z1)=%inf*z1* приведет к следующему: *%inf*z1* объявляет те элементы массива Z, чьи индексы содержатся в массиве Z1, бесконечным величинами.

При формировании прямоугольной сети для построения графика в узлах сети, смежных с бесконечными элементами массива Z, разумеется, все значения будут равны бесконечности, и функция *plot3d1* зальет соответствующие ячейки белым цветом. Таким образом, нам удастся создать эффект вырезания целых областей поверхности $Z=\sin(t)\cdot\cos(t)$ (см. листинг 5.19, рис. 5.22).

Обратите внимание, что на ось Z автоматически будут нанесены подписи отметок, выше и ниже которых значения элементов массива Z были объявлены бесконечными 0.5и -0.5.

```
t=linspace(-%pi,%pi,40);
z=sin(t)'*cos(t);
z1=find(abs(z)>0.5);
z(z1)=%inf*z1;
plot3d1(t,t,z);
```

Листинг 5.19

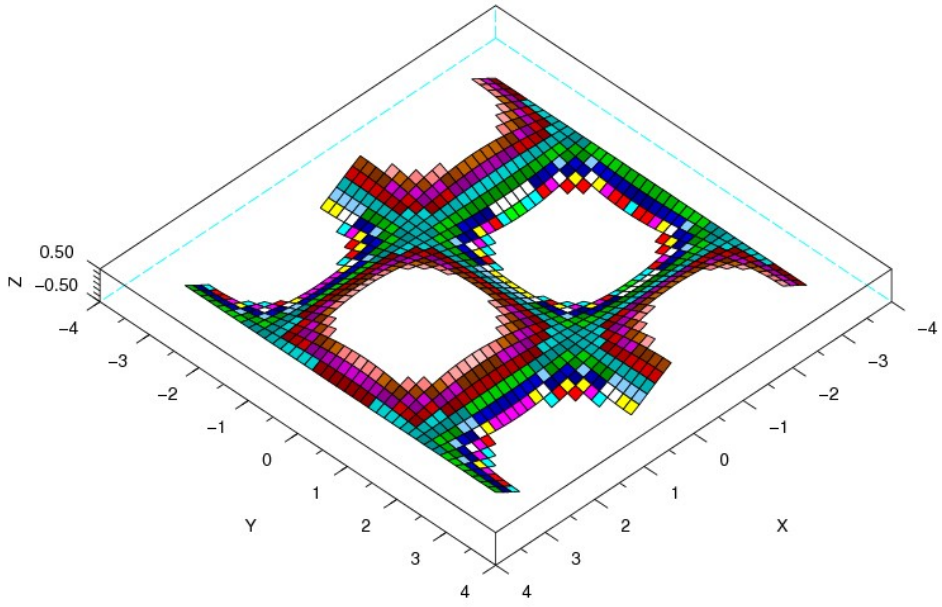

Рисунок 5.22. Пример «вырезания» областей из трехмерного графика

Теперь поставим перед собой другую задачу: построить геометрическое тело, полое внутри.

ЗАДАЧА 5.20

Построить график полой сферы.

Прежде всего, при помощи команды *deff* создадим функцию sph, которая задает сферу тремя параметрическими уравнениями X, Y, Z. Обратите внимание, что функция sph записана первыми тремя строками для удобства (см .листинг 5.20):

```
deff('[x,y,z]=sph(alp,tet)',['x=r*cos(alp).*cos(tet)
+orig(1)*ones(tet)';
'y=r*cos(alp).*sin(tet)+orig(2)*ones(tet)';
```

```
'z=r*sin(alp)+orig(3)*ones(tet)']);
```

Далее задаем значения параметра *r*, вектора-строки *orig*, массивов *x* и *y*.

Уже известным нам способом, при помощи функции *%inf*, объявляем бесконечными величинами элементы массива x с индексами (5:8) и (30:35).

Таким образом мы добьемся того, что в графическое окно будет выведена сфера, имеющая две «шляпки», сверху и снизу (см. рис. 5.23). Под ними элементы, объявленные бесконечными, образуют два «окошка», через которые мы сможем увидеть полость внутри сферы.

Саму же сферу построим при помощи команды *eval3dp* и функции *plot3d1*, для лучшего обзора всех деталей графика укажем углы поворота наблюдателя 35 и 15.

```
    deff('[x,y,z]=sph(alp,tet)',['x=r*cos(alp).*cos(tet)
+orig(1)*ones(tet)';
    'y=r*cos(alp).*sin(tet)+orig(2)*ones(tet)';
    'z=r*sin(alp)+orig(3)*ones(tet)']);
    r=1;orig=[0 0 0];
    x=linspace(-%pi/2,%pi/2,40);
    y=linspace(0,%pi*2,20);
    x(5:8)=%inf*ones(5:8);
    x(30:35)=%inf*ones(30:35);
    [x1,y1,z1]=eval3dp(sph,x,y);
    plot3d1(x1,y1,z1,35,15);
```

Листинг 5.20

Ранее мы уже познакомились с возможностями функции *plot3d2*. Теперь рассмотрим еще несколько примеров ее применения.

ЗАДАЧА 5.21

Построение ракушкообразного графика.

Такой график можно задать следующей системой уравнений:

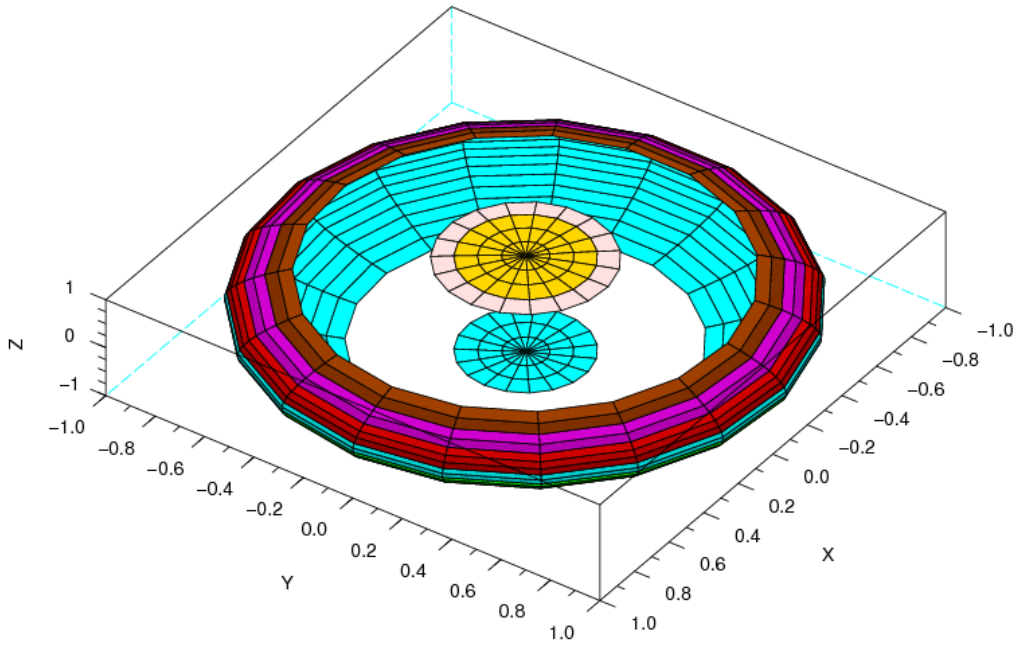

Рисунок 5.23. График полой сферы

$$\begin{cases} x = \cos(u)\cdot u\cdot\left(1+\cos\dfrac{(v)}{2}\right); \\ y = \dfrac{u}{2}\cdot\sin(v); \\ z = (\sin(u)\cdot u)\cdot\left(1+\cos\dfrac{(v)}{2}\right). \end{cases} \qquad (5.1)$$

Зададим массивы значений параметров *u*, *v*. Вычислим значения функций *x*, *y*, *z* (см. листинг 5.21). Обратившись к функции *plot3d2*, получим график, представленный на рис. 5.24.

```
u = linspace(0,2*%pi,40);
v = linspace(0,2*%pi,20);
x= (cos(u).*u)'*(1+cos(v)/2);
y= (u/2)'*sin(v);
z= (sin(u).*u)'*(1+cos(v)/2);
plot3d2(x,y,z);
```

Листинг 5.21

ЗАДАЧА 5.22

Построение ленты Мебиуса при помощи функции *plot3d2*.

Лента Мебиуса - простейшая односторонняя поверхность с краем. Попасть из одной точки этой поверхности в любую другую можно, не пересекая края.

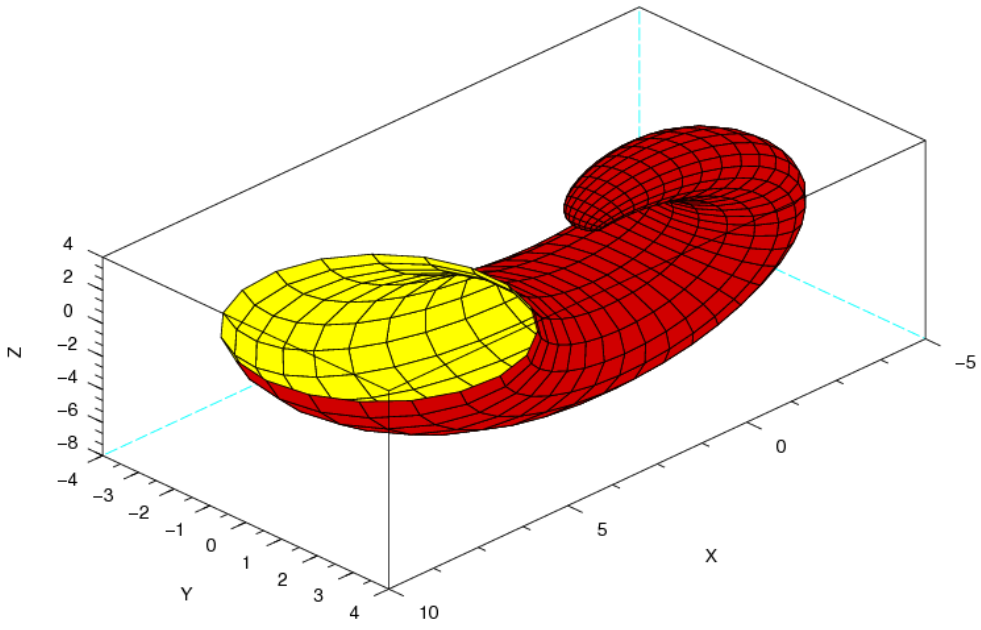

Рисунок 5.24. Ракушкообразный график, построенный функцией plot3d2

В общем параметрическом виде лента Мебиуса может быть представлена системой уравнений:

$$\begin{cases} x(u,v)=\left(1+\frac{v}{2}\cdot\cos\left(\frac{u}{2}\right)\right)\cdot\cos(u) \\ y(u,v)=\left(1+\frac{v}{2}\cdot\cos\left(\frac{u}{2}\right)\right)\cdot\sin(u) \\ z(u,v)=\frac{v}{2}\cdot\sin\left(\frac{u}{2}\right) \end{cases} \qquad (5.2)$$

Здесь *u* принадлежит интервалу [0;2π], а *v* - [-1;1]. Эти формулы задают ленту Мебиуса ширины 1, чей центральный круг имеет радиус 1, лежит в плоскости *xy* с центром в (0,0,0). Параметр *u* пробегает вдоль ленты, в то время как *v* задает расстояние от края.

На листинге 5.22 предложен один из способов построения ленты Мебиуса, а ее график представлен на рис. 5.25.

```
t=linspace(-1,1,20)';
x=linspace(0,%pi,40);
factor=2+t*cos(x);
X=factor*diag(cos(2*x));
Y=factor*diag(sin(2*x));
Z=t*sin(x);
plot3d2(X,Y,Z);
```

Листинг 5.22

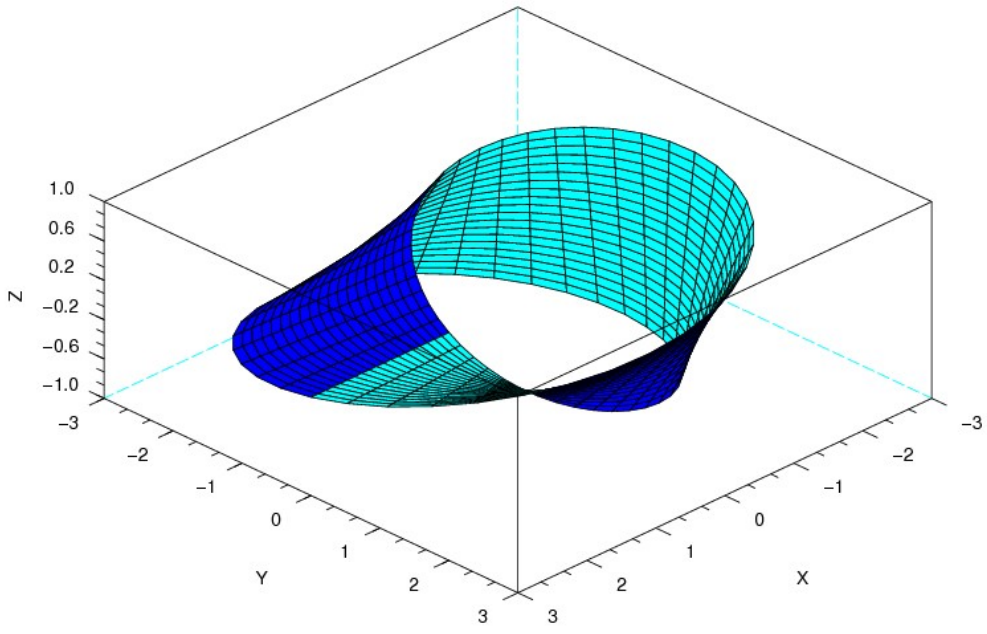

Рисунок 5.25. Лента Мебиуса

ЗАДАЧА 5.23

Построение тора с узкой и широкой стороной при помощи функции *plot3d2*.

Тор поверхность вращения форме бублика , получаемая вращением окружности вокруг оси, лежащей в плоскости окружности и её не пересекающей. Уравнение тора может быть задано параметрически в виде:

$$\begin{cases} x(u,v)=(R+r\cdot\cos(u)\cdot\cos(v)) \\ y(u,v)=(R+r\cdot\cos(u)\cdot\sin(v)) \\ z(u,v)=r\cdot\sin(u) \end{cases} \quad (5.3)$$

Здесь u, v принадлежат интервалу [0;2π), а также *R* - расстояние от центра окружности до оси вращения, *r* радиус окружности .

На листинге 5.23 приведен способ построения тора с узкой и широкой стороной подобно ленте Мебиуса, график тора изображен на рис. 5.26.

```
x=linspace(0,2*%pi,40);
y=linspace(0,2*%pi,20)';
fact=1.5+cos(y)*(cos(x)/2+0.6);
X=fact*diag(cos(x));
Y=fact*diag(sin(x));
Z=sin(y)*(cos(x)/2+0.6);
plot3d2(X,Y,Z,);
```

Листинг 5.23

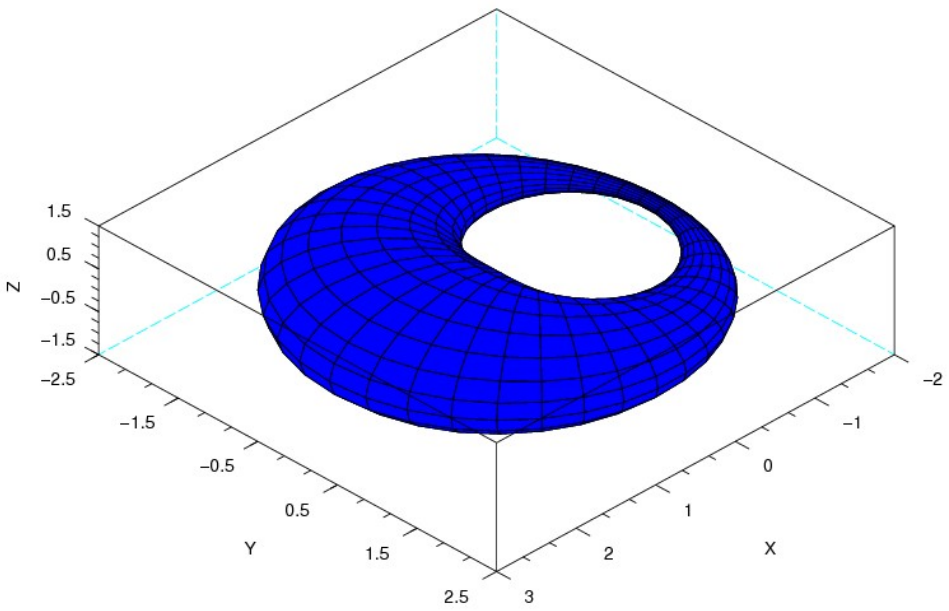

Рисунок 5.26. Тор с узкой и широкой стороной

ЗАДАЧА 5.24

Построение деформированного тора при помощи функции *plot3d2*.

Листинг 5.24 демострирует один из возможных способов построения вогнутого тора при помощи функции *plot3d2*, график тора представлен на рис. 5.27.

```
x=linspace(0,2*%pi,40);
y=linspace(0,2*%pi,20)';
factor=1.5+cos(y);
X=factor*cos(x);
Y=factor*sin(x);
Z=sin(y)*ones(x)+ ones(y)*cos(2*x);
plot3d2(X,Y,Z);
```

Листинг 5.24

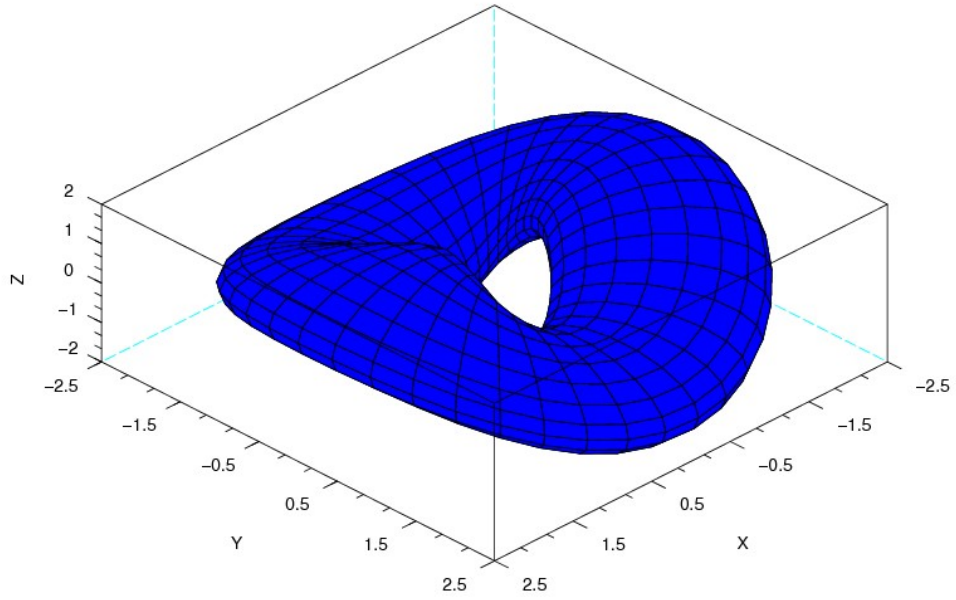

Рисунок 5.27. Деформированный тор