4. Построение графиков в **Octave**

В этой главе читатель научится строить графики в Octave. Первый параграф посвящен работе с двумерными графиками, также в нём описаны анимационные возможности пакета. Во втором параграфе рассмотрено создание различных трехмерных графиков. Завершается глава описанием возможностей графических пакетов расширений.

4.1 Построение двумерных графиков

Двумерным будем считать такой график, в котором положение точки определяется двумя величинами. Двумерные графики наиболее часто строят в декартовой и полярной системах координат.

4.1.1 Построение графиков в декартовой системе координат

Декартова или прямоугольная система координат, задается двумя перпендикулярными прямыми, называемыми осями координат. Горизонтальная прямая *X* – ось абсцисс, а вертикальная *Y* – ось ординат. Точку пересечения осей называют началом координат. Четыре угла, образованные осями координат, носят название координатных углов. Положение точки в прямоугольной системе координат определяется значением двух величин, называемых координатами точки. Если точка имеет координаты *x* и *y*, то *x* – абсцисса точки, *y* – ордината. Уравнение, связывающее координаты *x* и *y*, определяется как уравнение линии, если координаты любой точки этой линии удовлетворяют ему.

Величина *y* называется *функцией* переменной величины *x*, если каждому из тех значений, которые может принимать *x*, соответствует одно или несколько определенных значений *y*. При этом переменная величина *x* называется аргументом функции *y=f(x)*. Говорят также, что величина *y* зависит от величины *x*. Функция считается заданной, если для каждого значения аргумента существует соответствующее значение функции. Чаще всего используют следующие способы задания функций:

- *табличный* – числовые значения функции уже заданы и занесены в таблицу, недостаток заключается в том, что таблица может не содержать все нужные значения функции;
- *графический* – значения функции заданы при помощи линии (графика), у которой абсциссы изображают значения аргумента, а ординаты – соответствующие значения функции;
- *аналитический* – функция задается одной или несколькими формулами (уравнениями), при этом, если зависимость между *x* и *y* выражена уравнением, разрешенным относительно *y*, то говорят о явно заданной функции, в противном случае функция считается неявной.

Совокупность всех значений, которые может принимать в условиях поставленной задачи аргумент *x* функции *y=f(x)*, называется *областью определения* этой функции. Совокупность значений *y*, которые принимает функция *f(x)*, называется *множеством значений* функции.

Далее будем рассматривать построение графиков в прямоугольной системе координат на конкретных примерах.

ПРИМЕР 4.1. Построить график функции $y = \sin x + \frac{1}{3}\sin 3x + \frac{1}{5}\sin 5x$ на интервале [-10;10].

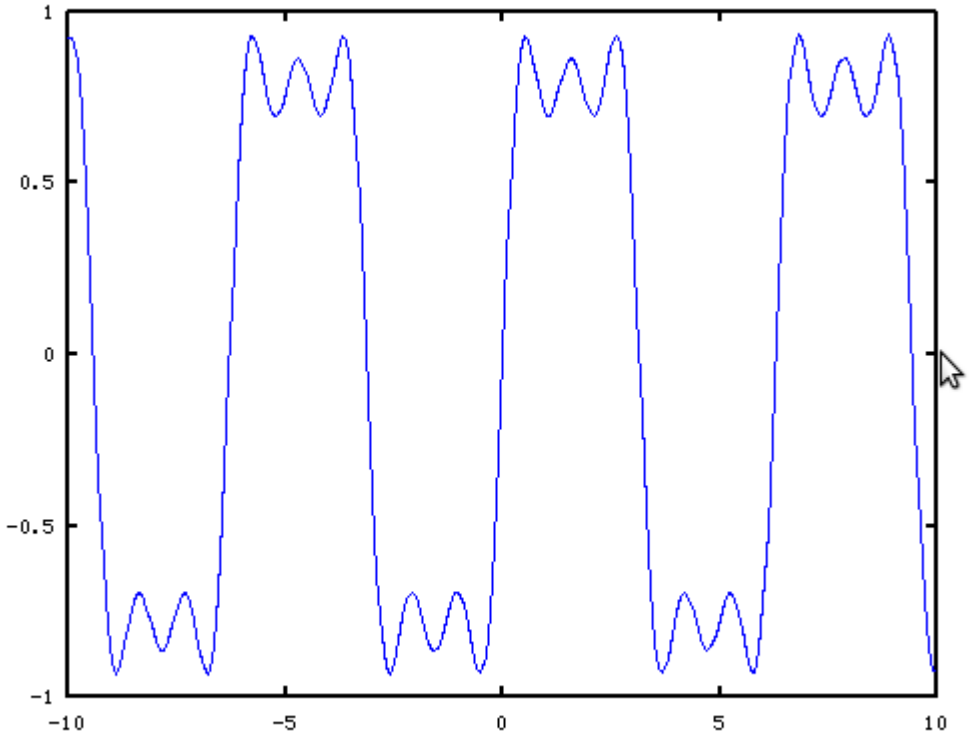
Для того, чтобы построить график функции *f(x)* необходимо сформировать два массива *x* и *y* одинаковой размерности, а затем обратиться к функции `plot`.

Решение этой задачи представлено на листинге 4.1.

```
x=-10:0.1:10; %Формирование массива x.
y=cos(x/2)+cos(5*x)/5; %Формирование массива y.
plot(x,y) %Построение графика функции.
```

Листинг 4.1

В результате обращения к функции `plot(x,y)` будет создано окно с именем *Figure* 1, в котором будет построен график функции $y=\sin(x)+\frac{1}{3}\sin(3x)+\frac{1}{5}\sin(5x)$ (см. рис. 4.1).



10.0819, 0.000299670

Рисунок *4.1:* График функции $y=\sin x+\frac{1}{3}\sin 3x+\frac{1}{5}\sin 5x$ на интервале *[-10;10]*

График формируется путем соединения соседних точек прямыми линиями. Чем больше будет интервал между соседними точками (чем меньше будет точек), тем больше будет заметно, что график представляет из себя ломанную.

Если повторно обратиться к функции `plot`, то в этом же окне будет стёрт первый график и нарисован второй. Для построения нескольких графиков в одной системе координат можно поступить одним из следующих способов:

1. Обратиться к функции `plot` следующим образом `plot(x1,y1,x2,y2,… xn,yn)`, где *x1*, *y1* – массивы абсцисс и ординат первого графика, *x2*, *y2* – массивы абсцисс и ординат второго графика, …, *xn*, *yn* – массивы абсцисс и ординат n-ого графика.
2. Каждый график изображать с помощью функции `plot(x,y)`, но перед обращением к функциям `plot(x2,y2)`, `plot(x3,y3)`, …, `plot(xn,yn)` вызвать команду `hold on`[1], которая блокирует режим очистки

1 Команда hold работает в режиме переключателя: hold on — блокирует режим очистки экрана, hold off — включает режим очистки экрана.

окна.

Рассмотрим построение нескольких графиков этими способамиобоими способами на примере решения следующей задачи.

ПРИМЕР 4.2. Построить графики функций $v=\sin x$, $w=\cos x$, $r=\sin\frac{x}{2}$, $p=\frac{3}{2}\cos x$ на интервале *[-4π; 4π]*.

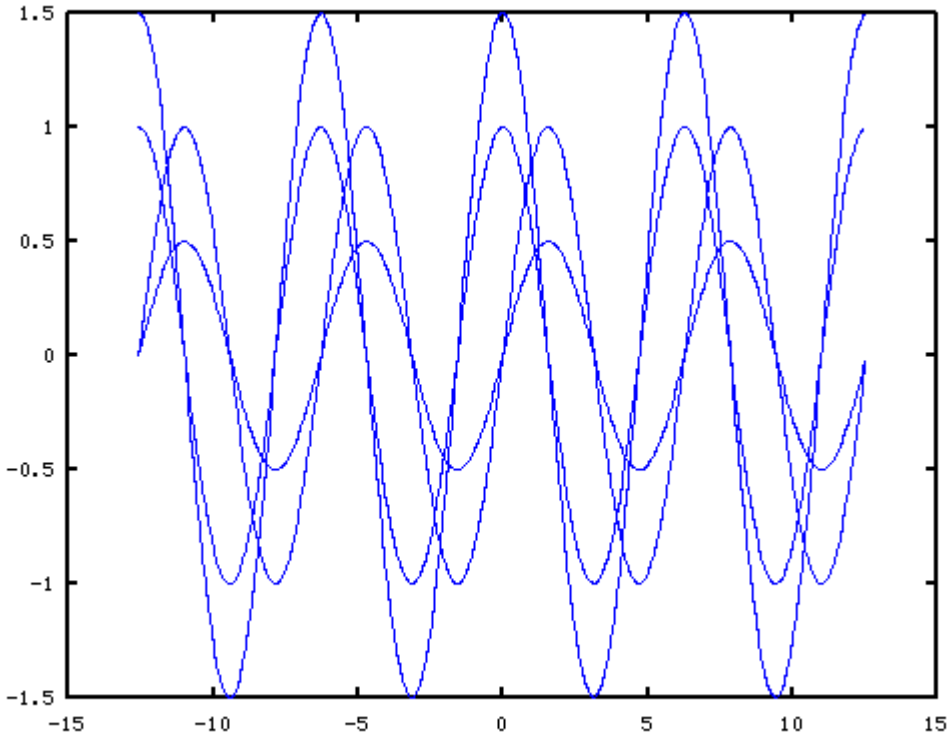
Построение графиков функций первый способом представлено на листинге 4.2, вторым – на листинге 4.3. Получившиеся графики функций представлены на рис. 4.2.

```
x=-4*pi:0.1:4*pi;
v=sin(x);w=cos(x);
r=sin(x)/2;p=1.5*cos(x);
plot(x,v,x,w,x,r,x,p);
```

Листинг 4.2.

```
x=-4*pi:0.1:4*pi;
v=sin(x);plot(x,v);
hold on;
v=cos(x);plot(x,v);
v=sin(x)/2;plot(x,v);
v=1.5*cos(x);plot(x,v);
```

Листинг 4.3



18.0246, 1.26266

Рисунок *4.2:* Графики функций $v=\sin x$, $w=\cos x$, $r=\sin\frac{x}{2}$, $p=\frac{3}{2}\cos x$

Обратите внимание, что при построении графиков первым способом Octave автоматически изменяет цвета изображаемых в одной системе координат графиков. Однако управлять цветом и видом каждого из изображаемых графиков может и пользователь, для чего необходимо воспользоваться полной формой функции `plot`:

```
plot(x1, y1, s1, x2, y2, s2, …, xn, yn, sn)
```

где `x1, x2, …, xn` – массивы абсцисс графиков; `y1, y2, …, yn` – массивы ординат графиков; `s1, s2, …, sn` – строка форматов, определяющая параметры линии и при необходимости, позволяющая вывести легенду.

В строке могут участвовать символы, отвечающие за тип линии, маркер, его размер, цвет линии и вывод легенды. Попробуем разобраться с этими символами. За сплошную линию отвечает символ «-». За маркеры отвечают следующие символы (см. табл. 4.1).

Цвет линии определяется буквой латинского алфавита (см. табл. 4.2), можно использовать и цифры, но на взгляд авторов использование букв более логично (их легче запомнить по английским названиям цветов).

Таблица *4.1:* Символы маркеров

Символ маркера	Изображение маркера
.	точка
*	✳
x	×
+	+
o	⊙
s	■
d	◆
v	▼
^	▲
<	▽
>	△
p	⊡
h	◇

Таблица *4.2:* Цвета линии

Символ	Цвет линии
y	желтый
m	розовый
c	голубой
r	красный
g	зеленый
b	синий
w	белый

При определении строки, отвечающей за вывод линии, следует учитывать следующее:

- не важен порядок символа цвета и символа маркера;
- если присутствует символ «-», то линия всегда будет сплошная, при этом, если

присутствует символ маркера, то все изображаемые точки еще будут помечатся маркером, если символа маркера – нет, то соседние точки просто будут соединяться линиями;

• если символ маркера отсутствует, то линия может быть, как сплошная, так и точечная; это зависит от наличия символа маркера, если символа маркера нет, то будет сплошная линия, иначе – точеная.

Если необходима легенда для графика, то ее следует включить в строку форматов, заключенную в символы «;». Например, команда

```
plot(x=-pi:0.1:pi,sin(x),"-k;sin(x);")
```

выведет на экран график функции *y=sin(x)* черного цвета на интервале [*-π;π*] с легендой «*sin(x)*» (см. рис. 4.3)

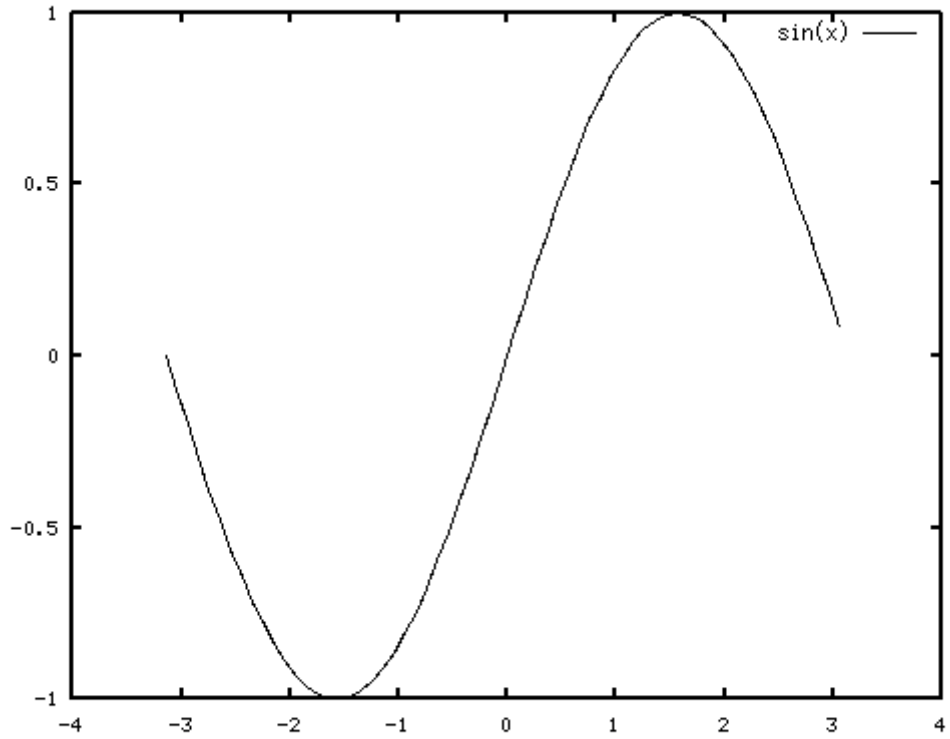


Рисунок *4.3:* Результаты функции работы функции *plot(x=-pi:0.1:pi,sin(x),"-k;sin(x);")*

Пользователь может управлять и величиной маркера, для этого после строки форматов следует указать имя параметра `"markersize"` (размер маркера) и через запятую величину — целое число), определяющее размер маркера на графике.

Например, команда

```
plot(x=-pi:0.1:pi,sin(x),"-ok;sin(x);", "markersize",4);
```

выведет на экран график, представленный на рис. 4.4.

Для того, чтобы чтобы вывести график в новом окне, перед функцией `plot`, следует вызвать функцию `figure()`.

Внимание!!! При работе с графиками в Octave **необходимо понимать следующее:** щелчок по кнопке закрытия окна с графиками приводит не к уничтожению (закрытию) окна, а к его скрытию. При повторном вызове команды рисования графиков происходит восстановление окна, в котором и изображаются графики. Корректное закрытие графического окна возможно осуществить в Octave можно осуществить только программно. Для этого необходимо осуществить открытие с помощью следующего вызова функции figure

```
h=figure();
```

Здесь `h` — переменная, в которой будет храниться дескриптор (номер) окна. Для дальнейших операций с окном надо будет использовать именно переменную, в которой храниться дескриптор.

Уничтожение (закрытие) окна осуществляется с помощью функции `delete(h)`, где `h` — имя дескриптора закрываемого окна.

В Octave есть функция `pause(n)`, которая приостанавливает выполнение программы на `n` секунд.

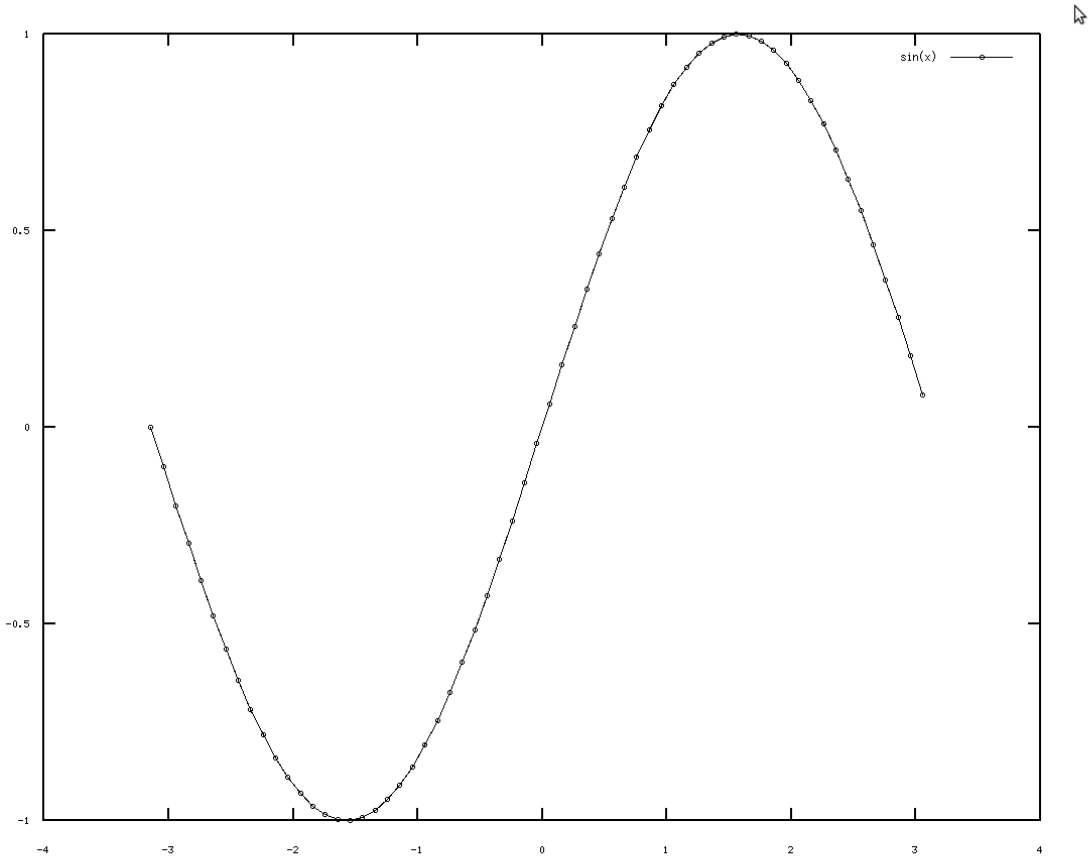


Рисунок *4.4:* Результаты работы функции *plot(x=-pi:0.1:pi,sin(x),"-ok;sin(x);", "markersize",4);*

Octave представляет дополнительные возможности для оформления графиков:

- команда `grid on` (`grid`) наносит сетку на график, `grid off` убирает сетку с графика;
- функция `axis[xmin, xmax, ymin, ymax]` выводит только часть графика, определяемую прямоугольной областью $x_{min} \le x \le x_{max}$, $y_{min} \le y < y_{max}$;
- функция `title('Заголовок')` предназначена для вывода заголовка графика;
- функции `xlabel('Подпись под осью x')`, `ylabel('Подпись под осью y')` служат для подписей осей *x* и *y* соответственно;
- функция `text(x,y,'текст')` выводит `текст` левее точки с координатами *(x,y)*;
- функция `legend('легенда1', 'легенда2', …, 'легендаn', m)` выводит легенды для каждого из графиков, параметр m определяет месторасположение легенды в графическом окне: 1 – в правом верхнем углу графика (значение по умолчанию); 2 – в

левом верхнем углу графика; 3 – в левом нижнем углу графика; 4 – в правом нижнем углу графика.

При выводе текста с помощью функций `xlabel, ylabel, title, text` можно выводить греческие буквы[2] (см. табл. 4.3), использовать символы верхнего и нижнего индекса. Для вывода текста в верхнем индексе используется символ «^», в нижнем — символ «_». Например, для вывода $e^{\cos(x)}$ необходимо будет ввести текст `e^{cos(x)}`, а для вывода x_{min} — текст `x_{min}`. При работе с текстом можно также использовать синтаксис TEX.

Таблица *4.3:* Греческие буквы

Команда	**Символ**	**Команда**	**Символ**
\alpha	α	\upsilon	υ
\beta	β	\phi	φ
\gamma	γ	\chi	χ
\delta	δ	\psi	ψ
\epsilon	ε	\omega	ω
\zeta	ζ	\Gamma	Γ
\eta	η	\Delta	Δ
\theta	θ	\Theta	Θ
\iota	ι	\Lambda	Λ
\kappa	κ	\Xi	Ξ
\lambda	λ	\Pi	Π
\mu	μ	\Sigma	Σ
\nu	ν	\Upsilon	Υ
\xi	ξ	\Phi	Φ
\pi	π	\Psi	Ψ
\rho	ρ	\Omega	Ω
\sigma	σ	\forall	∀
\varsigma	ς	\exists	∃
\tau	τ	\approx	≈
\int	∫	\in	Π
\wedge	^	\sim	~
\vee	v	\leq	≤
\pm	±	\leftrightarrow	↔
\geq	≥	\leftarrow	←
\inftyo	∞	\uparrow	↑
\partial	∂	\rightarrow	→

2 Для вывода греческих букв и кириллицы в вашей операционной системе должны быть установленны соответствующие шрифты.

Команда	**Символ**	**Команда**	**Символ**
\neq	≠	\downarrow	↓
\nabla	∆	\circ	º

После описания основных возможностей по оформлению графиков рассмотрим еще несколько примеров построения графиков.

ПРИМЕР 4.3. Последовательно вывести в графическое окно графики функций $y=\sin x$, $y=\sin\frac{3x}{4}$, $y=\cos x$, $y=\cos\frac{x}{3}$ с задержкой 5 секунд. Текст программы решения задачи на Octave с комментариями приведен на листинге 4.4.

```
% Создаём графическое окно с дескриптором okno1.
okno1=figure();
% Определяем аргумент (массив x) на интервале [-6π;6π].
x=-6*pi():pi()/50:6*pi();
%Вычисляем значение функции sin(x).
y=sin(x);
%Выводим график функции sin(x) чёрного цвета.
plot(x,y,'k');
%Выводим линии сетки.
grid on;
%Выводим заголовок графика.
title('Plot y=sin(x)');
%Приостанавливаем выполнение программы на 5 секунд.
pause(5);
y=sin(0.75*x);
%Выводим график функции sin(0.75x) голубого цвета.
plot(x,y,'b');
%Выводим линии сетки.
grid on;
%Выводим заголовок графика.
title('Plot y=sin(0.75x)');
%Приостанавливаем выполнение программы на 5 секунд.
pause(5);
y=cos(x);
%Выводим график функции cos(x) красного цвета.
plot(x,y,'r');
%Выводим линии сетки.
grid on;
%Выводим заголовок графика.
title('Plot y=cos(x)');
%Приостанавливаем выполнение программы на 5 секунд.
pause(5);
y=cos(x/3);
%Выводим график функции cos(x/3) зеленого цвета.
plot(x,y,'g');
%Выводим линии сетки.
grid on;
%Выводим заголовок графика.
title('Plot y=cos(x/3)');
%Приостанавливаем выполнение программы на 5 секунд.
pause(5);
```

```
%Закрываем окно с дескриптором okno1.
delete(okno);
```

Листинг 4.4

При запуске программы будет создано графическое окно, в котором будет выведен график функции *sin(x)* черного цвета с линиями сетки и заголовком, которое будет на экране в течении 5 секунд, после этого окно очиститься. Будет выведен график функции *sin(0.75x)* голубого цвета с линиями сетки и заголовком, которое будет на экране в течении 5 секунд. Далее аналогично будут с задержками 5 секунд выведены графики функций *y=cos(x)* и *y=cos(x/3)*. После этого окно автоматически закроется. Для понимания механизма работы с графическими окнами в Octave, авторы рекомендуют читателю при выводе графика *sin(x)* попытаться закрыть окно и посмотреть, что из этого получится.

ПРИМЕР 4.4. Построить графики функций $y=e^{\sin x}, u=e^{\cos\frac{x}{3}}, v=e^{\sin\frac{x}{2}}$ на интервале $[-3\pi;3\pi]$. Рассмотрим два варианта построения графиков (см. листинги 4.5, 4.6).

```
x=-3*pi():pi()/20:3*pi();% Формируем массив x.
y=exp(sin(x)); % Формируем массив y.
u=exp(cos(x/3));% Формируем массив u.
v=exp(sin(x/2));% Формируем массив v.
%Строим график функции y(x), сплошная чёрная линия,
% без маркера, качестве легенды выводим  e^{sin(x)} .
plot(x, y, "k;e^{sin(x)};")
%Блокируем режим очистки окна.
hold on;
%Строим график функции u(x), сплошная чёрная линия,
%с маркером треугольником, размер маркера — 4,
%качестве легенды выводим  e^{cos(x/3)} .
plot( x, u, "->k; e^{cos(x/3)};", "markersize", 4)
%Строим график функции v(x), сплошная чёрная линия,
%с маркером окружностью, размер маркера — 4,
%качестве легенды выводим  e^{sin(x/2)} .
plot( x,v,"-ok;e^{sin(x/2)};", "markersize",4);
```

Листинг 4.5

```
x=-3*pi():pi()/20:3*pi();
y=exp(sin(x));
u=exp(cos(x/3));
v=exp(sin(x/2));
%Отличие вывода трёх графиков состоит в том, вместо 3-х
%функций plot и двух hold on используется одна функция plot,
%в которой указаны теже параметры вывода графиков,
%что и на листинге 4.6
plot(x, y, "k;e^{sin(x)};", x, u, "->k; e^{cos(x/3)};",
"markersize", 4, x,v,"-ok;e^{sin(x/2)};", "markersize",4);
```

Листинг 4.6

Оба способа формируют один и тот же график (см. рис. 4.5).

ПРИМЕР 4.5. Построить график функции $y(x)=1-\frac{0.4}{x}+\frac{0.05}{x^2}$ на интервале *.[-2;2]*.

В связи с тем, что функция не определена в точке *x=0*, будем строить её, как графики двух функций *y(x)* на интервале *[-2;0)* и *y(x)* на интервале *(0;2]*. Текст программы на Octave с комментариями приведён на листинге 4.7, график функции — на рис. 4.6.

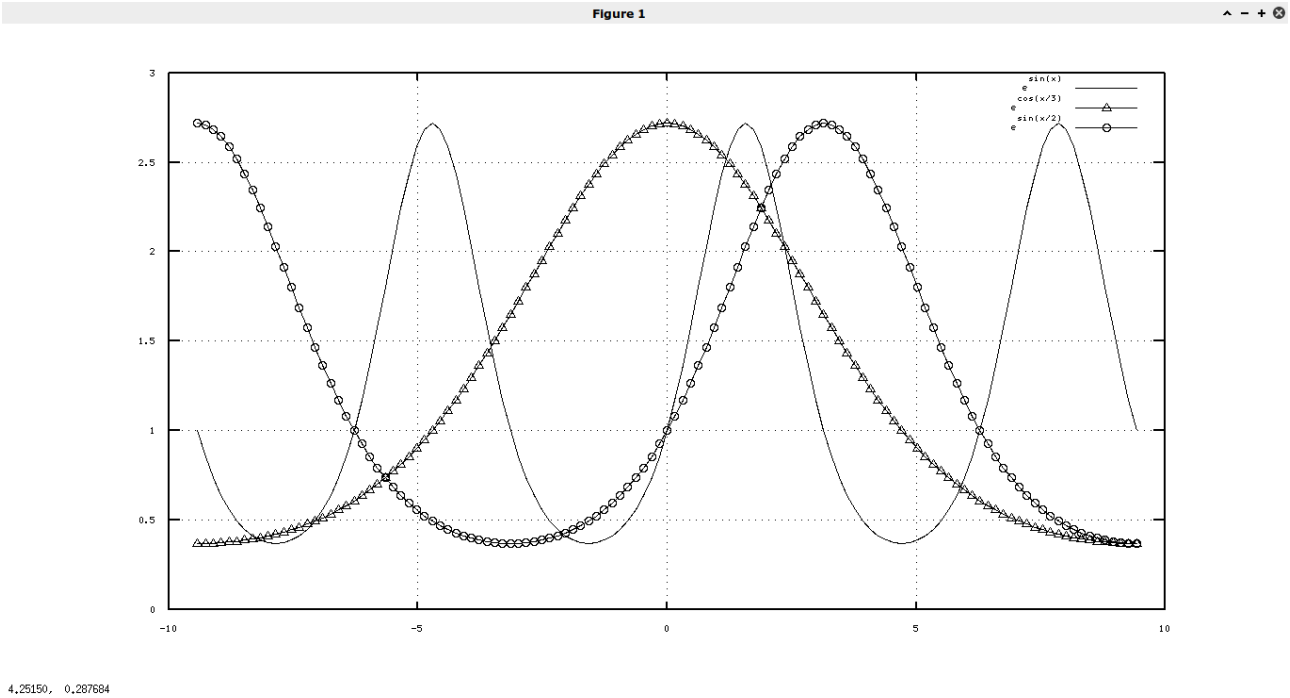


Рисунок *4.5:* Графики функций $y=e^{\sin x}, u=e^{\cos\frac{x}{3}}, v=e^{\sin\frac{x}{2}}$

```
a=1;d=-0.4;c=0.05;
% Определяем аргумент (массив x) на интервале [-2;-0.1].
x1=-2:0.01:-0.1;
% Определяем аргумент (массив x) на интервале [0.1;2].
x2=0.1:0.01:2;
%Вычисляем значение функции y(x) на интервале [-2;-0.1].
y1=a+b./x1+c./x1./x1;
%Вычисляем значение функции y(x) на интервале [0.1;2].
y2=a+b./x2+c./x2./x2;
% Строим график  y(x)=1-0.4/x+0.05/x^2  как график двух функций,
% на интервалах [-2;-0.1], [0.1;2], цвет графика чёрный,
% Легенда -  f(x)=a+b/x+c/x^2  .
plot(x1,y1,'k;f(x)=a+b/x+c/x^2;',x2,y2,'k');
title('y=f(x)');% Подпись над графиком.
xlabel('X'); % Подпись оси X.
ylabel('Y');% Подпись оси Y.
grid on; % Рисуем линии сетки.
```

Листинг 4.7

ПРИМЕР 4.6. Построить график функции $y(x)=\frac{1}{x^2-2x-3}$ на интервале [-5;7]. Уравнение $x^2-2x-3=0$ имеет корни *-1*, *3*. Поэтому наша функция *y(x)* будет иметь разрывы в этих точках *x=-1*, *x=3*. Будем строить её, как графики трёх функций, на трёх интервалах [*-5;-1.1*], [*0.9;2.9*], [*3.1;7*]. Листинг 4.8 демонстрирует решение примера 4.6. На рис. 4.7 изображен график функции $y(x)=\frac{1}{x^2-2x-3}$, который получился в результате работы программы, представленной на листинге 4.8.

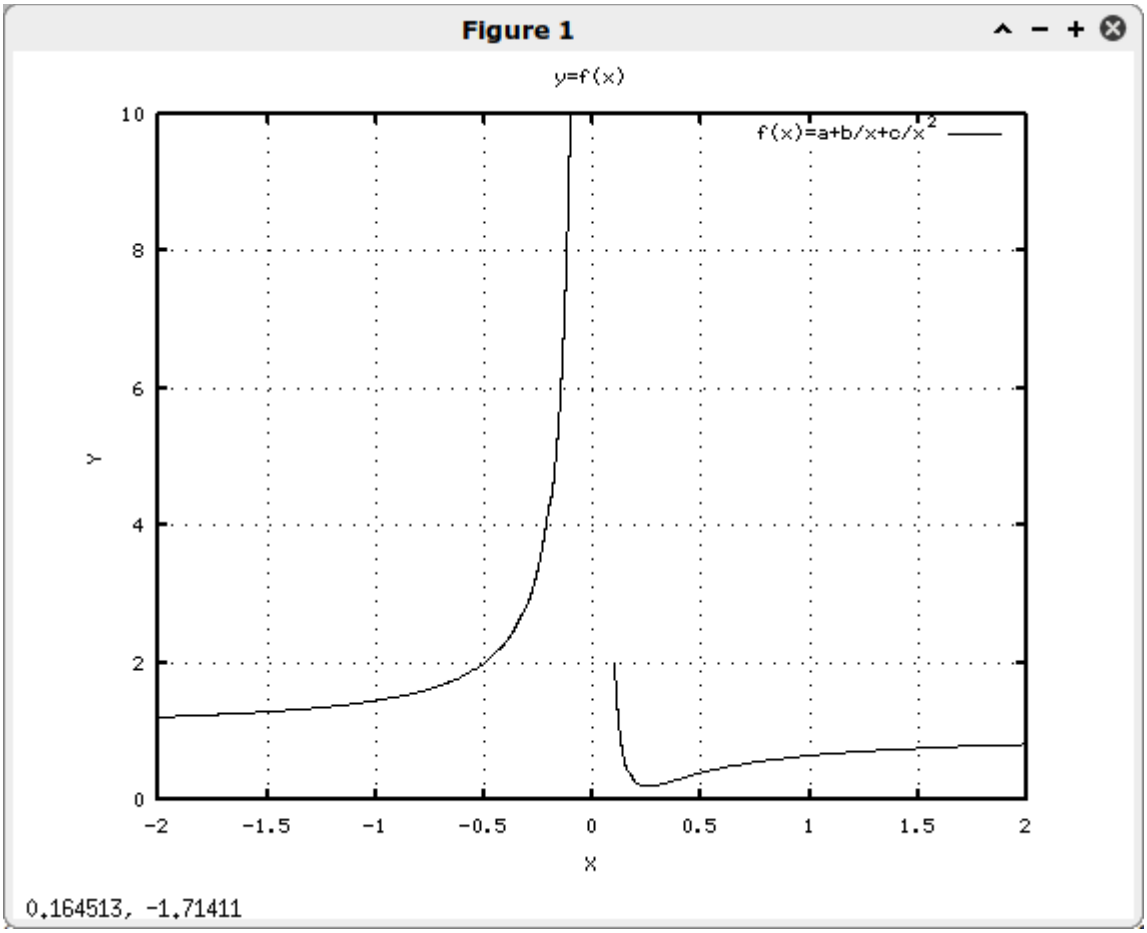


Рисунок *4.6:* График функции $y(x)=1-\frac{0.4}{x}+\frac{0.05}{x^2}$

```
%Определяем аргументы на интервалах [-5;-1.1], [0.9;2.9],
% [3.1;7].
x1=-5:0.01:-1.1;
x2=-0.9:0.01:2.9;
x3=3.1:0.01:7;
%Вычисляем значение  y(x) на соответствующих интервалах.
y1=1./(x1.*x1-2*x1-3);
y2=1./(x2.*x2-2*x2-3);
y3=1./(x3.*x3-2*x3-3);
%Строим график  y(x)=1/(x^2-2x-3)  черного цвета,
%как график 3-х функций.
plot(x1,y1,'k',x2,y2,'k', x3,y3,'k');
title('y=f(x)'); % Подпись над графиком.
xlabel('X'); % Подпись оси X.
ylabel('Y');.% Подпись оси Y.
legend('f(x)=1/(x^2-2x-3)',4);% Вывод легенды.
grid on; % Рисуем линии сетки.
```

Листинг 4.8

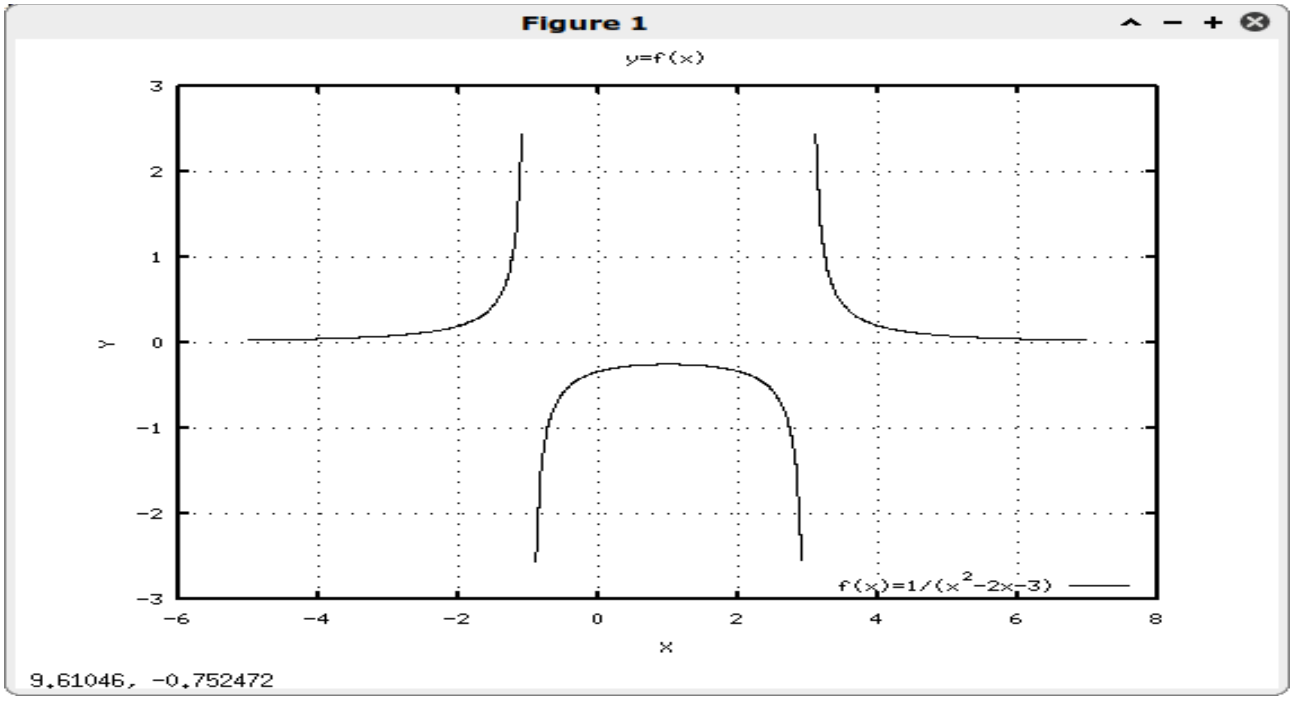


Рисунок *4.7:* График функции $y(x)=\frac{1}{x^2-2x-3}$

Еще одну интересную возможность построения двух графиков предоставляет функция `plotyy` , которая позволяет изображать на графике две оси ординат, что очень удобно при построении графиков разных порядков. К сожалению эта функция не позволяет явно задавать типы изображаемых линий. На каждой из осей ординат подписи значений выводятся тем же цветом, что и график функции. Использование функции `plotyy` рассмотрено ниже.

ПРИМЕР 4.7. Построить графики функции $y=x^3, y=\cos 2x$ на интервале *[-5;5]*.

Решение этой задачи представлено на листинге 4.9, полученный график – на рис. 4.8.

```
x=-5:0.1:5;
y=x.^3;
z=cos(2*x);
plotyy(x,y,x,z);
grid on;
title('Plot x^3, cos(x/2)');
xlabel('X');
ylabel('Y');
```

Листинг 4.9.

Octave предоставляет возможность построить несколько осей в графическом окне и вывести на каждую из них свои графики. Для этого следует использовать функцию `subplot`:

```
subplot(row, col, cur);
```

Параметры `row` и `col` определяют количество графиков по вертикали и горизонтали соответственно, `cur` определяет номер текущего графика. Повторное обращение к функции `subplot` с теми же значениями `row` и `col` позволяет просто изменяет номер текущего графика и может использоваться для переключения между графиками. Рассмотрим использование функции `subplot` при решении следующей задачи.

ПРИМЕР 4.8. Построить графики функции $y=2e^{-0.15x^2}, z=e^{0.7x-0.25x^2}, u=0.5e^{-0.33x}\sin\left(2x+\frac{\pi}{3}\right), k=3\sin(x-0.22x^2), v=\cos x, w=e^{\cos x}$ на

интервале [*-6;6*]. Решение задачи представлено на листинге 4.10, полученное графическое окно – на рис. 4.9.

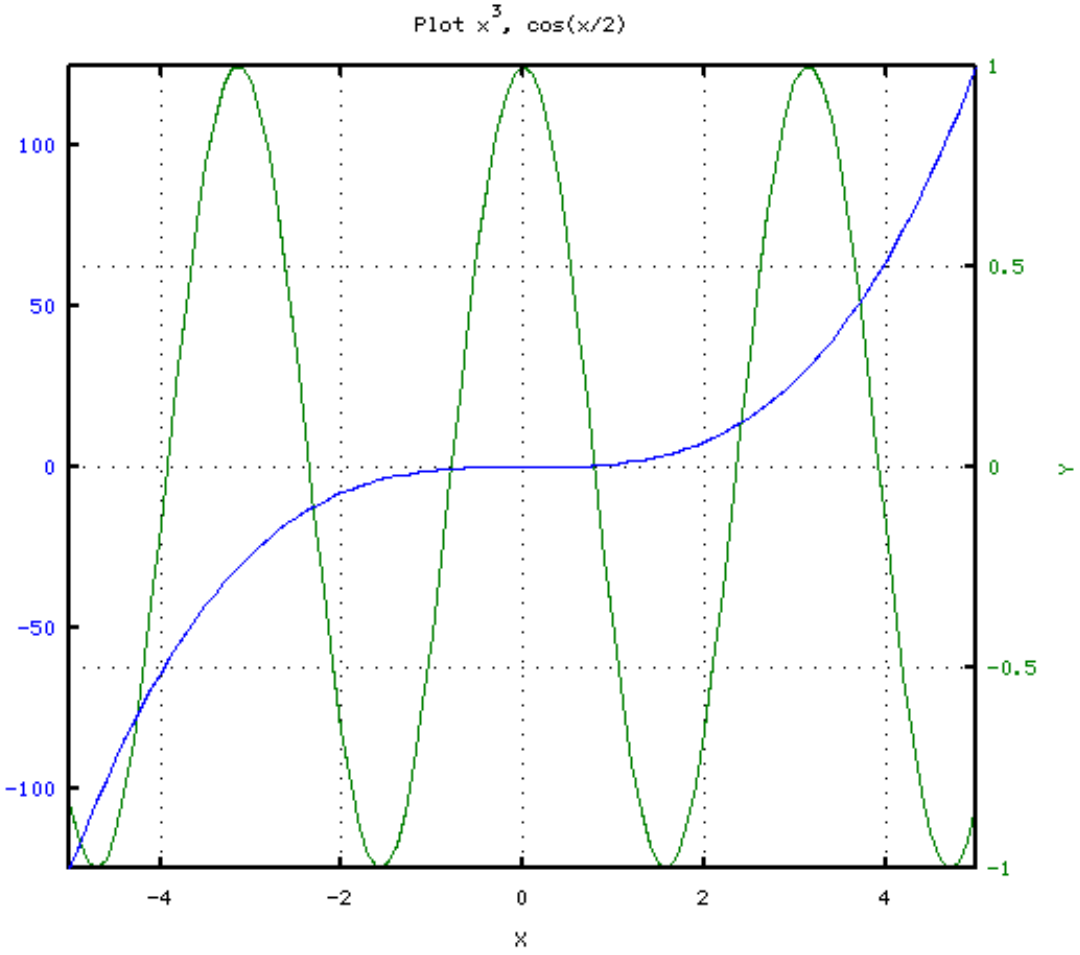


Рисунок *4.8:* Графики функции $y=x^3, y=\cos 2x$

```
%Формируем массивы x,y,z,u,k,v,w.
x=-6:0.2:6;
y=2*exp(-0.15*x.^2);
z=exp (0.7*x-0.25*x.^2)
u=0.5*exp (-0.33*x).*sin (2*x+pi ()/3);
k=3*sin (x-0.22.*x.^2);
v=cos(x);
w=exp (cos (x));
% Делим графическое окно на 6 частей и объявляем первый график
% текущим.
subplot(3,2,1);
% Строим график y(x) с линиями сетки и подписями.
plot(x,y);
grid on;
title('Plot y=2e^{-0.15x^2}');
xlabel('x');
ylabel('y');
% Второй график объявляем текущим.
subplot(3,2,2);
```

```
% Строим график z(x) с линиями сетки и подписями.
plot(x,z);
grid on;
title('Plot z=cos^2(x)');
xlabel('x');
ylabel('z');
% Третий график объявляем текущим.
subplot(3,2,3);
% Строим график u(x) с линиями сетки и подписями.
plot(x,u);
grid on;
title('Plot u=0.5e^{-0.33x}sin (2x+pi/3)');
xlabel('x');
ylabel('u');
% Четвертый график объявляем текущим.
subplot(3,2,4);
% Строим график k(x) с линиями сетки и подписями.
plot(x,k);
grid on;
title('Plot k=3sin (x-0.22x^2)');
xlabel('x');
ylabel('k');
% Пятый график объявляем текущим.
subplot(3,2,5);
% Строим график v(x) с линиями сетки и подписями.
plot(x,v);
grid on;
title('Plot v=cos(x)');
xlabel('x');
ylabel('v');
% Шестой график объявляем текущим.
subplot(3,2,6);
% Строим график w(x) с линиями сетки и подписями.
plot(x,w);
grid on;
title('Plot w=e^{cos (x)}');
xlabel('x');
ylabel('w');
```

Листинг 4.10

Для построения графика функции можно использовать функцию `fplot` следующей структуры

```
fplot(@f,[xmin, xmax],s).
```

Здесь `f` — имя функции (стандартной функции Octave или функции, определенной пользователем), `[xmin, xmax]` — интервал, на котором будет строиться график, `s` — строка формата, определяющая только параметры линии (но не легенду). Легенда графика формируется автоматически функцией `fplot` без использования функции `legend`. Не позволяет формировать легенду и третий параметр функции `fplot`, в отличии от функции `plot`.

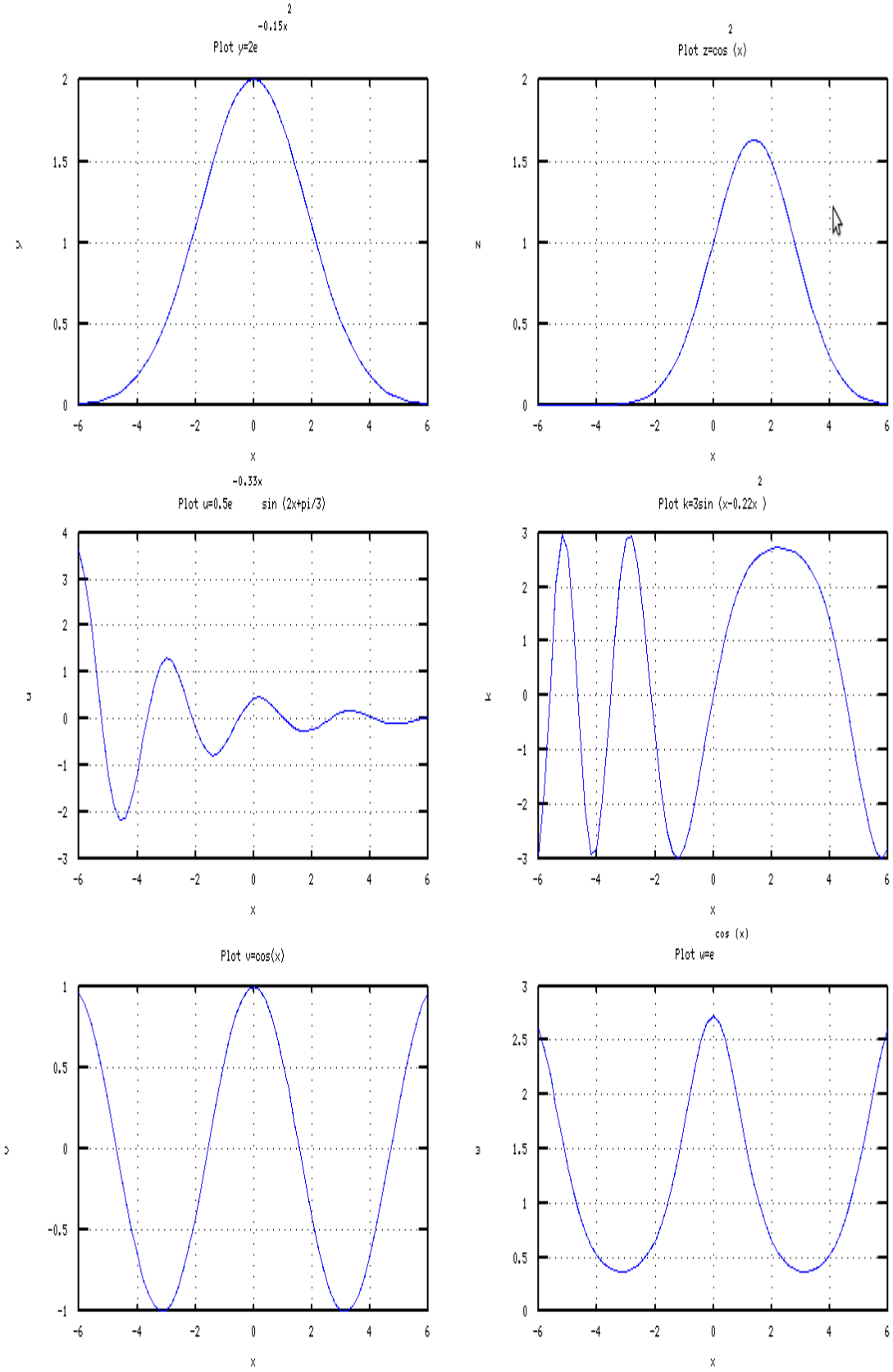


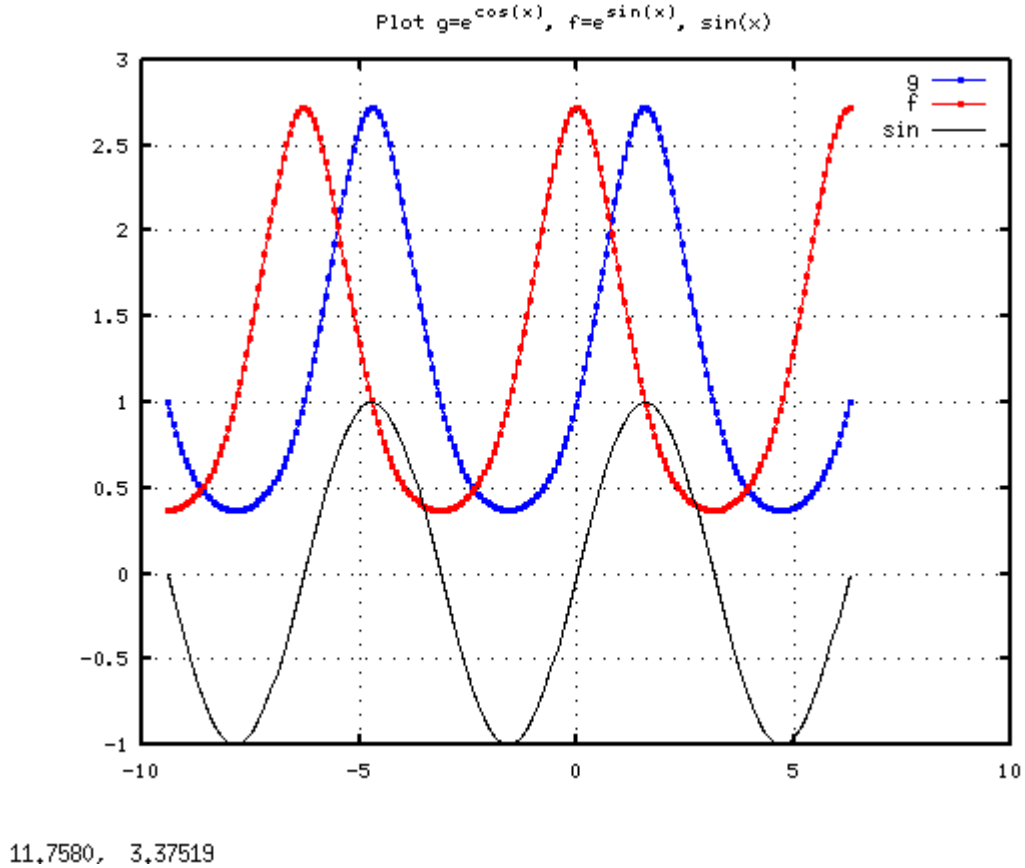
Рисунок *4.9:* Графики функций *y(x), z(x), u(x), k(x), v(x), w(x)*

ПРИМЕР 4.9. Используя функцию `fplot`, построить графики функций $e^{\sin x}, e^{\cos x}, \sin x$ на интервале [*-3π;2π*]. Текст программы на языке Octave с комментариями

приведен на листинге 4.11. Полученные графики можно увидеть на рис. 4.10

```
% Определяем функцию f=exp(cos(x))
function y=f(x)
y=exp(cos(x));
end
% Определяем функцию g=exp(cos(x))
function z=g(x)
z=exp(sin(x));
end
%Строим график g=exp(cos(x)) на интервале [-3π;2π] голубого
% цвета с маркером .
fplot (@g, [-3*pi(), 2*pi()],'-pb');
hold on;
%Строим график f=exp(sin(x)) на интервале [-3π;2π] красного
% цвета с маркером .
fplot (@f, [-3*pi(), 2*pi()],'-or');
%Строим график sin(x) на интервале [-3π;2π] чёрного цвета
fplot (@sin, [-3*pi(), 2*pi()],'k');
%Подписываем график.
title('Plot y=e^{cos(x)}, z=e^{cos(x)}, sin(x)');
% Проводим линии сетки
grid on;
```

Листинг 4.11



11.7580, 3.37519

Рисунок *4.10:* Графики функций $e^{\sin x}, e^{\cos x}, \sin x$

4.1.2 Построение графиков в полярной системе координат

Полярная система координат состоит из заданной фиксированной точки O, называемой полюсом, концентрических окружностей с центром в полюсе и лучей, выходящих из точки O, один из которых, OX, называют полярной осью. Положение любой точки M в полярных координатах можно задать положительным числом ρ=|OM| (полярный радиус) и числом φ, равным величине угла XOM (полярный угол). Числа ρ и φ называют полярными координатами точки M и обозначают M(ρ, φ).

Для формирования графика в полярной системе координат необходимо сформировать массивы значений полярного угла и полярного радиуса и обратиться к функции `polar`

```
polar(phi, ro, s)
```

где `phi` – массив полярного угла; `ro` – массив полярного радиуса; `s` – строка, состоящая из трех символов, которые определяют цвет линии, тип маркера и тип линии (см. табл. 4.1, 4.2).

В качестве примера использования функции `polar` рассмотрим решение следующей задачи.

ПРИМЕР 4.10. Построить график лемнискаты.

Уравнение лемнискаты в полярных координатах имеет вид: $\rho = a\sqrt{2\cos 2\varphi}$, функция ρ определена при $-\frac{\pi}{2} \le \varphi \le \frac{\pi}{2}$ ($\cos 2\varphi \ge 0$). Поэтому листинг для изображения графика лемнискаты будет таким.

```
% Определяем массив полярного угла fi.
fi=-pi/4:pi/200:pi/4;
%Определяем массив положительных значений полярного радиуса ro
лемнискаты.
ro=3*sqrt(2*cos(2*fi));
% Рисуем правую часть лемнискаты.
polar(fi,ro,'r');
%Блокируем режим очистки окна.
hold on;
% Рисуем левую часть лемнискаты.
polar(fi,-ro,'r');
grid on;
```

Листинг 4.12

Полученный график изображен на рис. 4.11.

ПРИМЕР 4.11. Построить графики архимедовой спирали, гиперболической спирали и логарифмической спирали в полярных координатах.

Уравнение архимедовой спирали в полярных координатах имеет вид: $\rho = a\varphi$, гиперболической – $\rho = \frac{a}{\varphi}$. Соотношение $\rho = a e^{k\varphi}, k = ctg\,\alpha$ является уравнением логарифмической спирали в полярных координатах. Частным случаем логарифмической спирали ($\alpha = \frac{\pi}{2}, k = 0$) является уравнение окружности ($\rho = a$).

На листинге 4.13 приведён текст, программы, позволяющей построить в одном графическом окне четыре оси координат, в каждом из которых построить свой график архимедову, гиперболическую и логарифмическую спирали, а также окружность.

График представлен на рис. 4.12.

```
h=figure()
clear all;
%Формируем массивы fi1,fi2,fi3,fi4,ro1,ro2,ro3,ro4.
fi1=0:pi/20:6*pi;
```

```
fi2=pi/3:pi/200:6*pi;
fi3=0:pi/20:4*pi;
fi4=-pi:pi/20:pi
ro1=4*fi1;
r2=0.5./fi2;
ro3=4*exp(0.2*fi3);
for i = 1:41
ro4(i)=4;
endfor
% Делим графическое окно на 4 части и объявляем первый график
% текущим.
subplot(2,2,1);
% Строим график архимедовой спирали.
polar(fi1,ro1);
% Подписываем график.
title('Graph of Archimedean spiral');
% Второй график объявляем текущим.
subplot(2,2,2);
% Строим график гиперболической спирали.
polar(fi2,r2);
% Подписываем график.
title('Graph of the hyperbolic spiral');
% Третий график объявляем текущим.
subplot(2,2,3);
% Строим график логарифмической спирали.
polar(fi3,ro3);
% Подписываем график.
title('Graph of the logarithmic spiral');
% Четвертый график объявляем текущим.
subplot(2,2,4);
% Строим график окружности.
polar(fi4,ro4);
% Подписываем график.
title('Graph the circle');
```

Листинг 4.13

4.1.3 Построение графиков, заданных в параметрической форме

Задание функции $y(x)$ с помощью равенств $x=f(t)$ и $y=g(t)$, называют параметрическим, а вспомогательную величину t – параметром. Построение графика функции, заданной параметрически, можно осуществлять следующим образом:

1.Определить массив *t*.

2.Определить массивы $x=f(t)$ и $y=g(t)$.

3.Построить график функции $y(x)$ с помощью функции *plot(x,y)*.

В качестве примера рассмотрим построение график эпициклоиды и астроиды.

ПРИМЕР 4.12. Построить график эпициклоиды. Уравнение эпициклоиды в параметрической форме имеет вид $x=4\cos t-\cos 4t$, $y=4\sin t-\sin 4t$, $t\in[0;2\pi]$. На листинге 4.14 представлен текст программы для изображения графика эпициклоиды, а на рис. 4.13 – сам график.

```
t=0:pi/50:2*pi;
x=4*cos(t)-cos(4*t);
y=4*sin(t)-sin(4*t);
plot(x,y);
grid on;
```
Листинг 4.14

ПРИМЕР 4.13. Построить график астроиды. Уравнение астроиды в параметрической форме имеет вид $x=3\cos^3 t, y=3\sin^3 t, t\in[0;2\pi]$. На листинге 4.15 представлен текст программы для изображения графика астроиды, а на рис. 4.14 – сам график.

```
t=0:pi/50:2*pi;
x=3*cos(t).^3;
y=3*sin(t).^3;
plot(x,y);
grid on;
```
Листинг 4.15

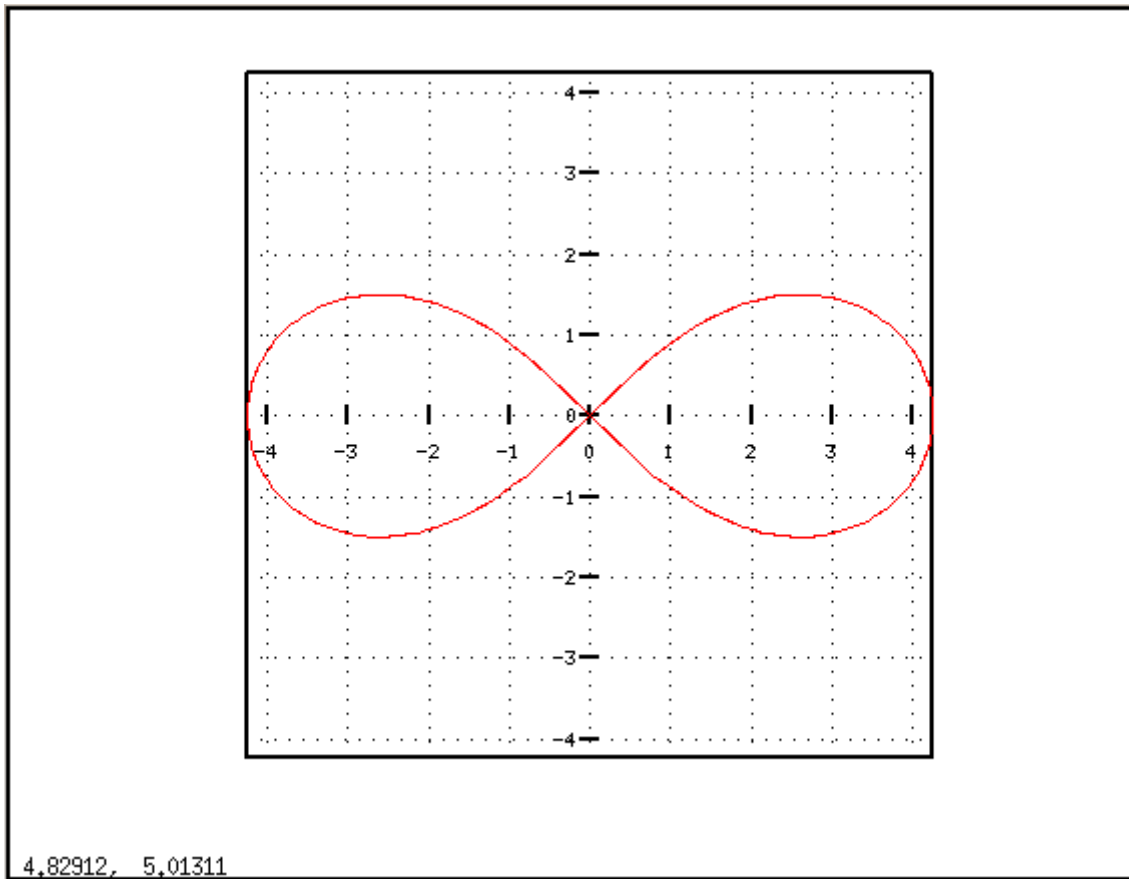


Рисунок *4.11* График лемнискаты в полярных координатах

Функция `bar` предназначена для построения гистограммы. Функция `bar(y)` выводит элементы массива `y` в виде гистограммы, в качестве массива `x` выступает массив номеров элементов массива `y`. Функция `bar(x,y)` выводит гистограмму элементов массива `y` в виде столбцов в позициях, определяемых массивом `x`, элементы которого должны быть упорядочены в порядке возрастания.

Рассмотрим несколько примеров.

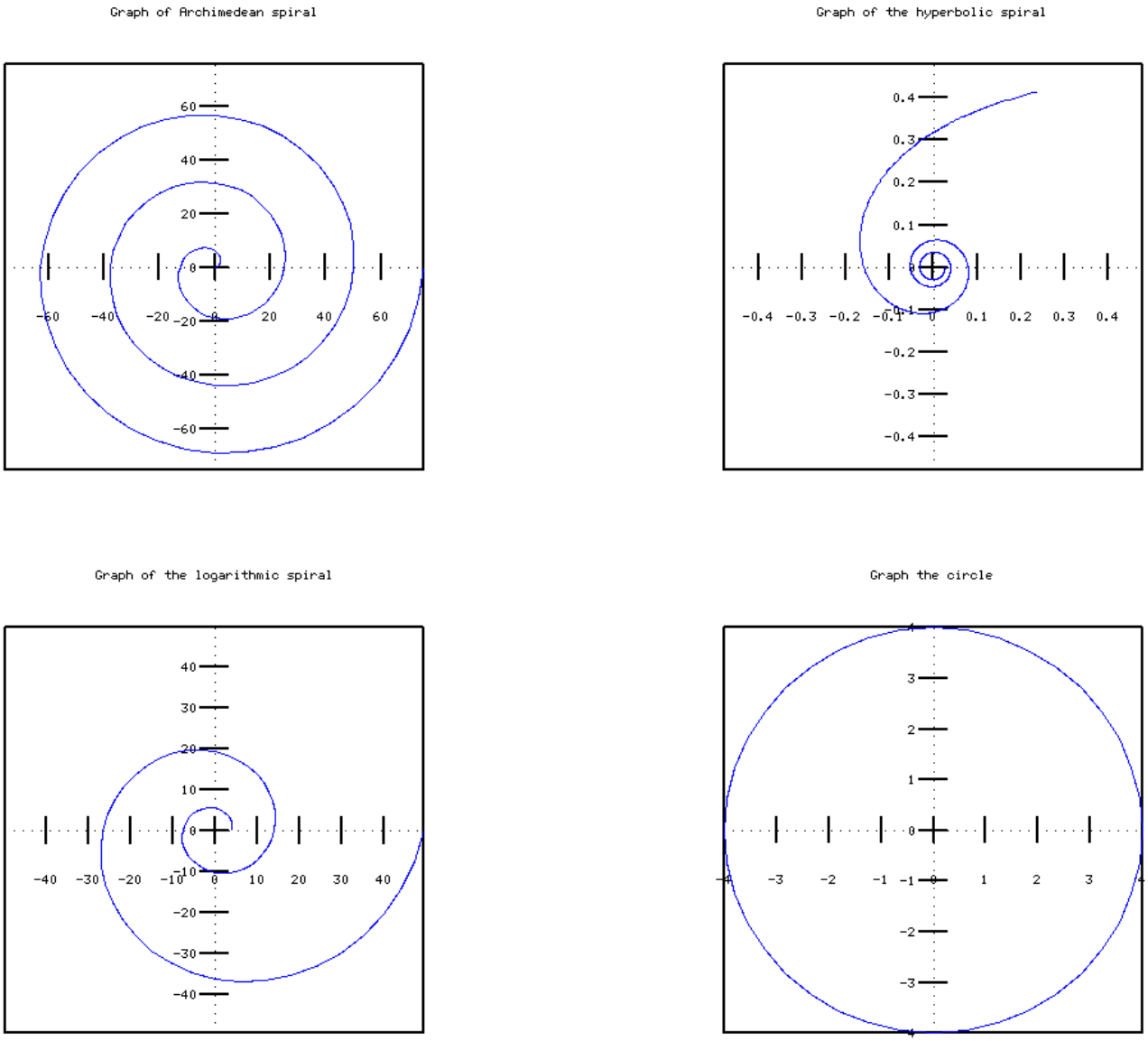


Рисунок *4.12* Графики архимедовой , гиперболической и логарифмической спирали, окружности в полярных координатах

Фрагмент

```
y=[5; 6;7; 8; 9; 8 ;7;6;4;3];
bar(y);
```

строит гистограмму, представленную на рис. 4.15.

Фрагмент

```
x1=[-2,-1,0,1,2,3,4];
y1=exp(sin(x1));
bar(x1,y1);
```

строит гистограмму, представленную на рис. 4.16.

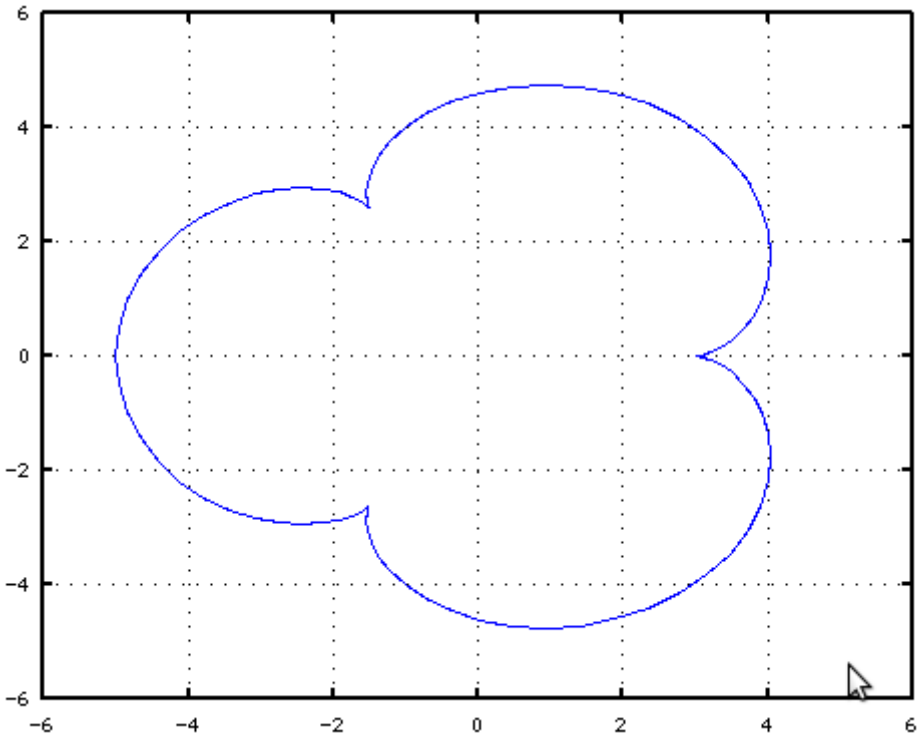


Рисунок *4.13* Г*рафик эпициклоиды*

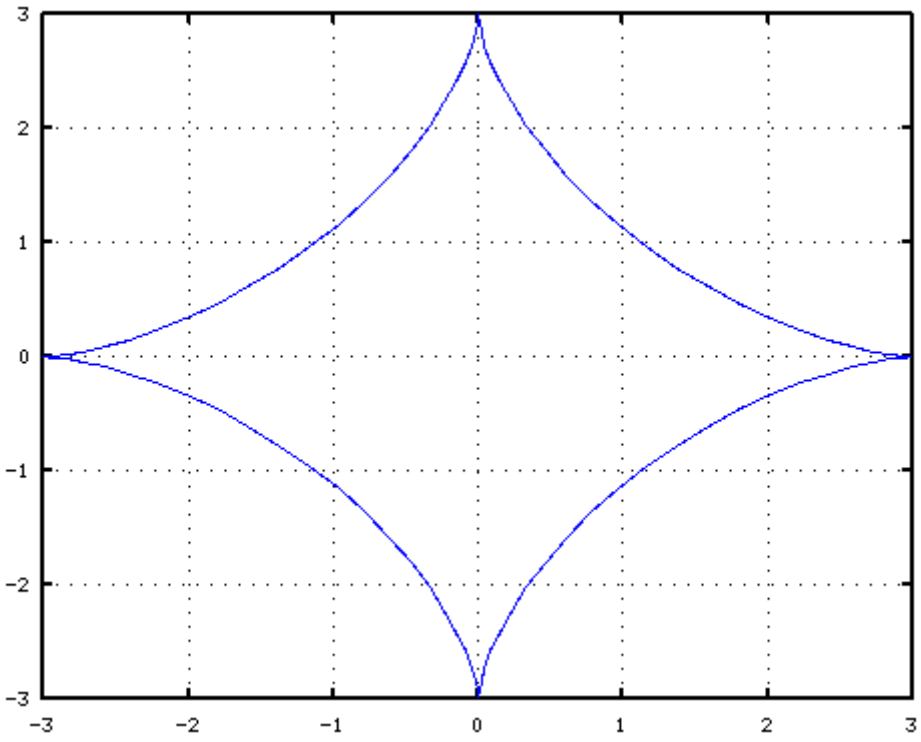


Рисунок *4.14* Г*рафик астроиды*

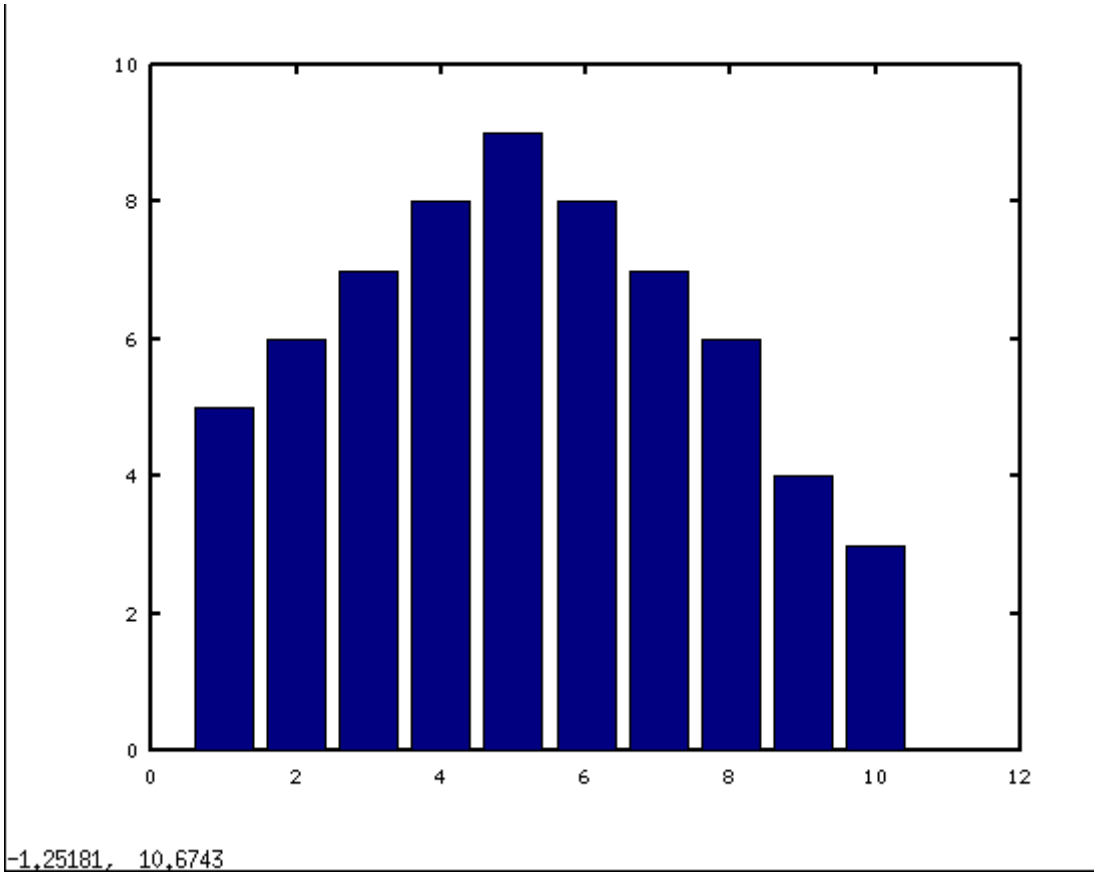


Рисунок *4.15* Пример использования функции *bar(y)*

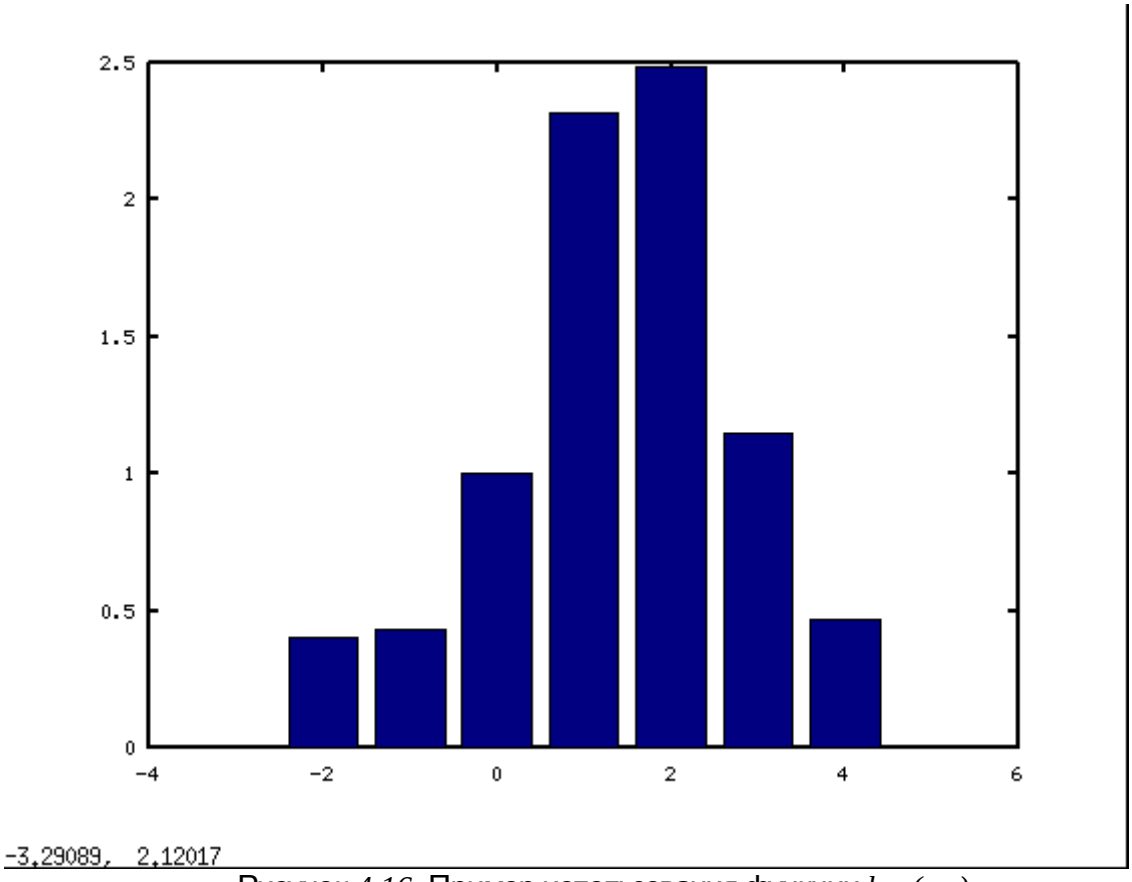


Рисунок *4.16.* Пример использования функции *bar(x,y)*

4.2 Построение трёхмерных графиков

График поверхности (трехмерный или 3D-график) - это график, положение точки в котором определяется значениями трех координат.

4.2.1 Построение графиков поверхностей

Дадим определение прямоугольной (или декартовой) системы координат в пространстве. *Прямоугольная система координат в пространстве* состоит из заданной фиксированной точки **O** пространства, называемой *началом координат*, и трех перпендикулярных прямых пространства **OX**, **OY** и **OZ**, не лежащих в одной плоскости и пересекающихся в начале координат, их называют *координатными осями* (**OX** – *ось абсцисс*, **OY** – *ось ординат*, **OZ** – *ось аппликат*). Положение точки *M* в пространственной системе координат определяется значением трех координат и обозначается *M(x,y,z)*. Три плоскости, содержащие пары координатных осей, называются *координатными плоскостями* **XY**, **XZ** и **YZ**.

Величина *z* называется *функцией двух величин x* и *y,* если каждой паре чисел, которые могут быть значениями переменных *x* и *y*, соответствует одно или несколько определенных значений величины *z*. При этом переменные *x* и *y* называют *аргументами функции z(x,y)*. Пары тех чисел, которые могут быть значениями аргументов *x, y* функции *z(x,y)*, в совокупности составляют *область определения* этой функции.

Для построения графика двух переменных *z=f(x,y)* необходимо выполнить следующие действия.

1. Сформировать в области построения графика прямоугольную сетку, проводя прямые, параллельные осям $y=y_j$ и $x=x_i$, где $x_i=x_0+ih$, $h=\frac{x_n-x_0}{n}$, $i=0,1,2,\dots,n$, $y_j=y_0+j\Delta$, $\Delta=\frac{y_k-y_0}{k}$, $j=0,1,2,\dots,k$.
2. Вычислить значения $z_{i,j}=f(x_i,y_j)$ во всех узлах сетки.
3. Обратиться к функции построения поверхности, передавая ей в качестве параметров сетку и матрицу $Z=\{z_{i,j}\}$ значений в узлах сетки.

Для формирования прямоугольной сетки в Octave есть функция `meshgrid`. Рассмотрим построение 3-х мерного графика на следующем примере.

ПРИМЕР 4.14. Построить график функции $z(x,y)=3x^2-2\sin^2 y$, $x\in[-2,2]$, $y\in[-3,3]$.

Для формирования сетки воспользуемся функцией `meshgrid`.

```
[x y]=meshgrid(-2:2,-3:3)
x =
-2    -1    0    1    2
-2    -1    0    1    2
-2    -1    0    1    2
-2    -1    0    1    2
-2    -1    0    1    2
-2    -1    0    1    2
-2    -1    0    1    2
y =
-3    -3    -3    -3    -3
-2    -2    -2    -2    -2
-1    -1    -1    -1    -1
0     0     0     0     0
1     1     1     1     1
2     2     2     2     2
```

```
3     3     3     3     3
```

После формирования сетки вычислим значение функции во всех узловых точках

```
>> z=3*x.*x-2*sin(y).^2
z =
   11.96017    2.96017   -0.03983    2.96017   11.96017
   10.34636    1.34636   -1.65364    1.34636   10.34636
   10.58385    1.58385   -1.41615    1.58385   10.58385
   12.00000    3.00000    0.00000    3.00000   12.00000
   10.58385    1.58385   -1.41615    1.58385   10.58385
   10.34636    1.34636   -1.65364    1.34636   10.34636
   11.96017    2.96017   -0.03983    2.96017   11.96017
```

Для построения каркасного графика следует обратиться к функции `mesh`.

```
mesh(x,y,z);
```

После это будет создано графическое окно с трехмерным графиком (см. рис. 3.20).

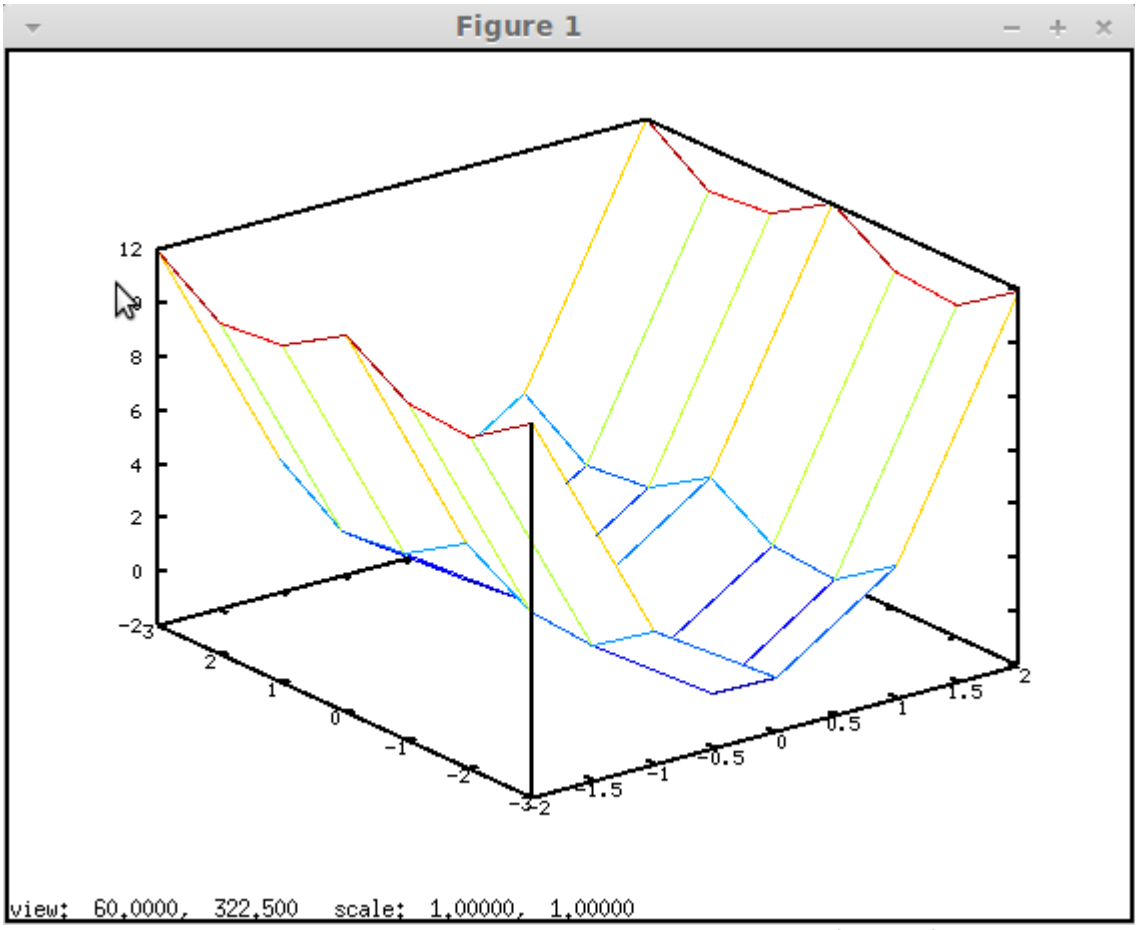


Рисунок *4.17*. График функции $z(x,y)=3x^2-2\sin^2 y$

Как видно, полученный график получился грубым, для получения менее грубого графика следует сетку можно сделать более плотной (см. листинг 4.16 и рис. 4.18).

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

Листинг 4.16

Любой трёхмерный график можно вращать, используя мышку.

Для построения поверхностей, кроме функции `mesh` построения каркасного графика, есть функция `surf`, которая строит каркасную поверхность, заливая ее каждую клетку цветом, который зависит от значения функции в узлах сетки.

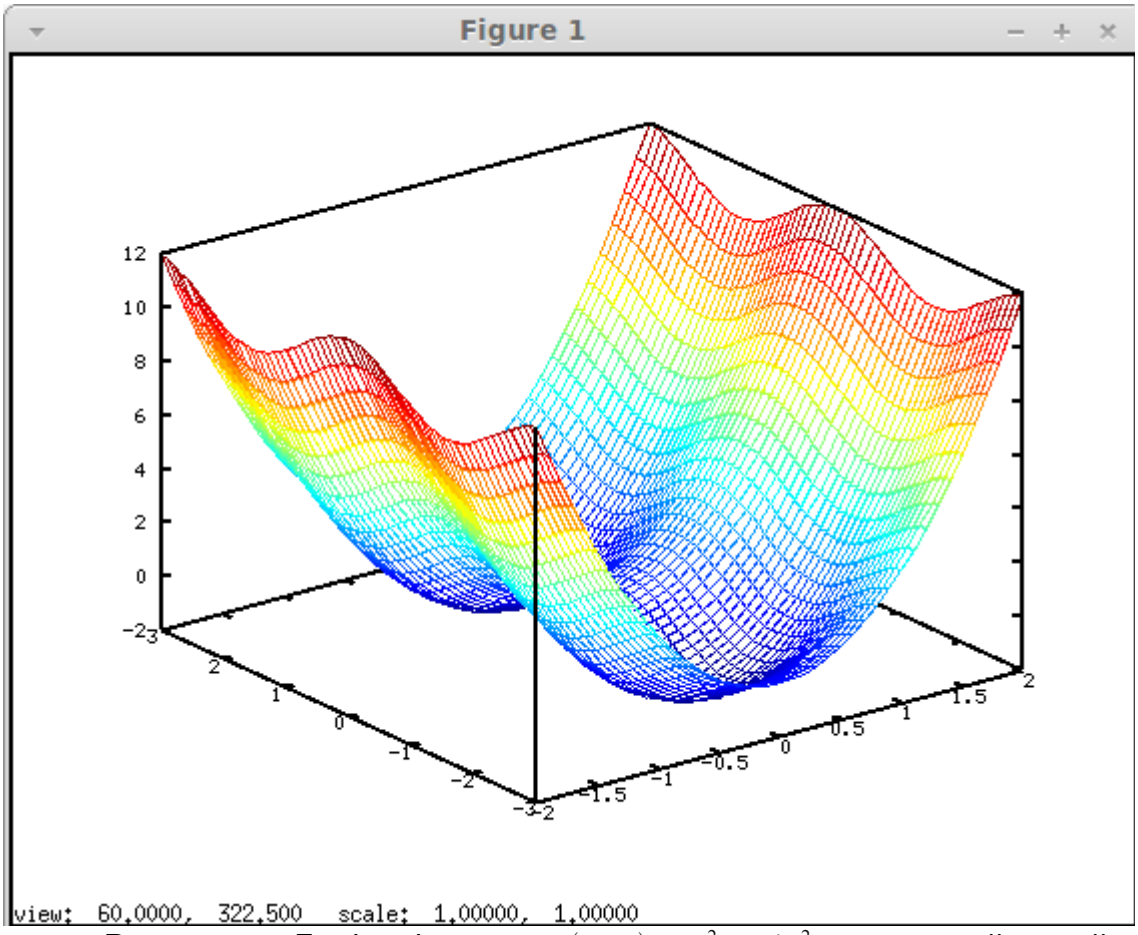


Рисунок *4.18*. График функции $z(x,y)=3x^2-2\sin^2 y$ с плотной сеткой

ПРИМЕР 4.15. С использованием функции `surf` построить график функции $z(x,y)=\sqrt{\sin^2 x+\cos^2 y}$.

На листинге 4.17 представлено решение задачи, а на рис. 4.19 изображён получившийся график.

```
[x y]=meshgrid(-2:0.2:2,0:0.2:4);
z=sqrt(sin(x).^2+cos(y).^2);
surf(x,y,z);
```

Листинг 4.17

В Octave можно построить графики двух поверхностей в одной системе координат, для этого, как и для плоских графиков следует использовать команду `hold on`, которая блокирует создание второго нового окна при выполнении команд `surf` или `mesh`.

ПРИМЕР 4.16. Построить график функций $z(x,y)=\pm(2x^2+3y^4)-1$.

Решение задачи с использованием функции `surf` представлено на листинге 4.18, полученный график изображен на рис. 4.20.

```
h=figure();
[x y]=meshgrid(-2:0.1:2,-3:0.1:3);
z=2*x.^2+3*y.^4-1;
z1=-2*x.^2-3*y.^4-1;
surf(x,y,z);
hold on
surf(x,y,z1);
```

Листинг 4.18

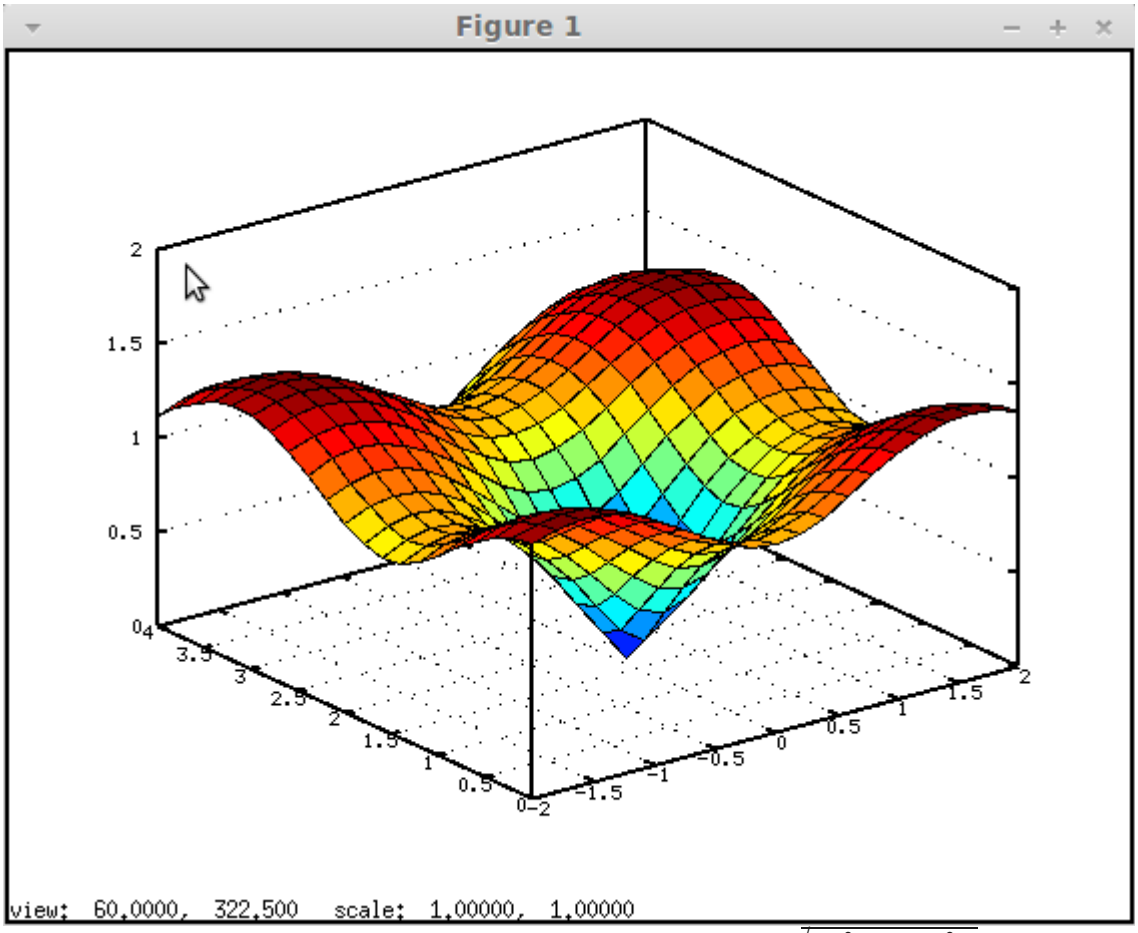


Рисунок *4.19*. График функции $z(x, y) = \sqrt{\sin^2 x + \cos^2 y}$

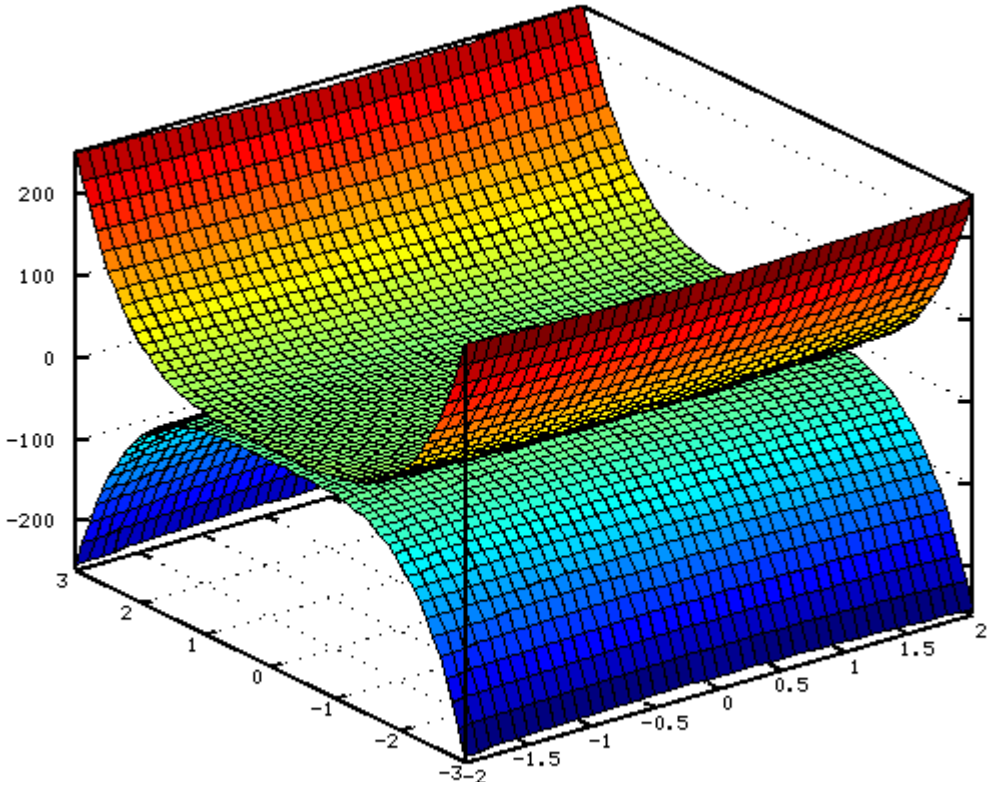


Рисунок *4.20*. Изображение двух поверхностей в одной системе координат с использованием функции `surf`

Построение поверхности с помощью функции `mesh` можно осуществить аналогично (см. листинг 4.19), графики функций – можно увидеть на рис.4.21.

```
h=figure();
[x y]=meshgrid(-2:0.1:2,-3:0.1:3);
z=2*x.^2+3*y.^4-1;
z1=-2*x.^2-3*y.^4-1;
mesh(x,y,z);
hold on
mesh(x,y,z1);
```

Листинг 4.19.

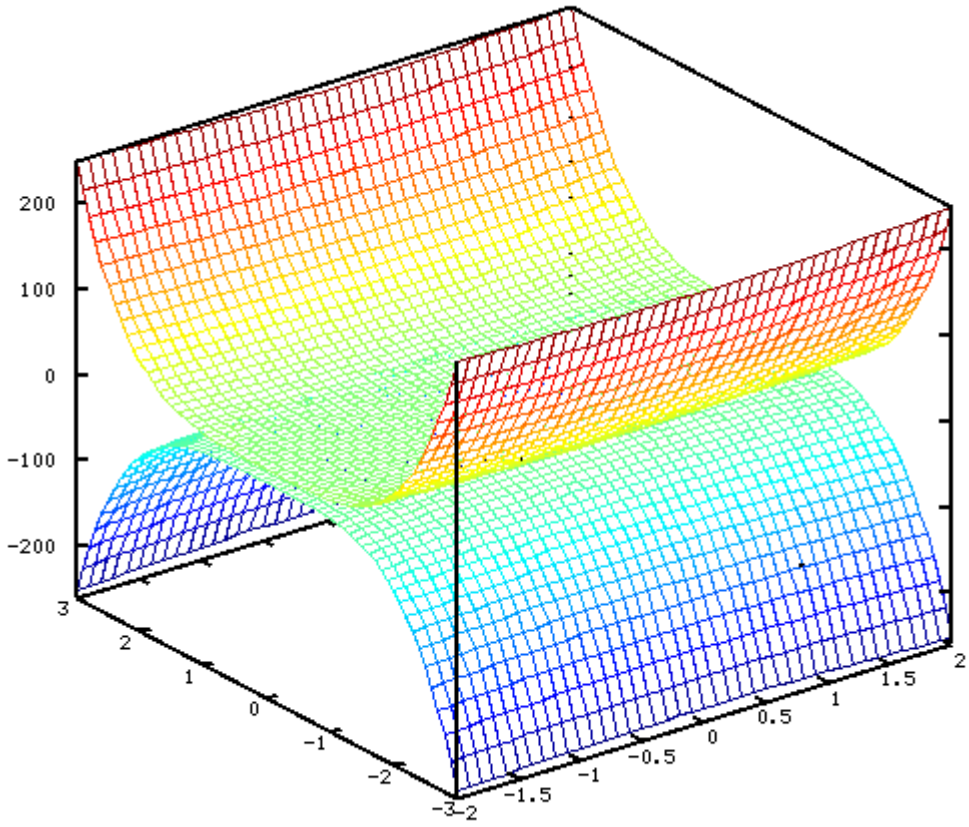


Рисунок *4.21*. Изображение двух поверхностей в одной системе координат с использованием функции `mesh`

4.2.2 Построение графиков поверхностей, заданных параметрически

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

1. Если *x*, *y* и *z* представимы в виде *f(u)*·*g(v)* , то соответствующие им матрицы *X*, *Y* и *Z* следует формировать в виде матричного умножения *f(u)* на *g(v)*.
2. Если *x*, *y* и *z* представимы в виде *f(u)* или *g(v)*, то в этом случае матрицы *X*, *Y* и *Z* следует записывать в виде *f(u)*·*ones(size(v))* или *g(v)*·*ones(size(u))* соответственно.

Рассмотрим несколько задач построения графиков поверхностей заданных

параметрически.

ПРИМЕР 4.17. Построить поверхность однополостного гиперболоида, уравнение которого задано в параметрическом виде *x(u,v)=ch(u)cos(v), y(u,v)= ch(u)sin(v), z(u,v)=sh(u),* $u \in [0,\pi], v \in [0,2\pi]$.

На листинге 4.20 представлено решение этой задачи, согласно описанному выше алгоритму. График однополостного гиперболоида представлен на рис.4.22.

```
clear all;
h=3.14/50;
% Формируем вектор-столбец u.
u=[0:h:3.14]';
% Формируем вектор-строку v.
% Обратите внимание, u – столбец, v – строка с одинаковым
количеством
% элементов.
v=[0:2*h:6.28];
% Формируем матрицу X как матричное произведение ch(u)*cos(v).
X=cosh(u)*cos(v);
% Формируем матрицу Y как матричное произведение ch(u)*sin(v).
Y=cosh(u)*sin(v);
% Формируем матрицу Z как матричное произведение столбца sh(u)
на строку
% ones(size(v)).
Z=sinh(u)*ones(size(v));
% Формируем график поверхности.
surf(X,Y,Z);
grid on;
% Подписываем график и оси.
title('Plank hyperboloid');
xlabel('X');
ylabel('Y');
zlabel('Z');
```

Листинг 4.20.

Рассмотрим несколько способов построения сферы в Octave.

ПРИМЕР 4.18. Построить поверхность сферы с центром $(x_{0,} y_{0,} z_0)$ и радиусом R. $\left((x-x_0)^2+(y-y_0)^2+(z-z_0)^2=R^2\right)$. Уравнение сферы можно записать в параметрическом виде $x(u,v)=x_0+R\sin(u)\cos(v), y(u,v)=y_0+R\sin(u)\sin(v), z(u,v)=z_0+R\cos(u)$, $u \in [0,2\pi]$, $v \in [0,\pi]$.

Методика построения сферы подобна построению однополостному гиперболоиду, описанному в примере 4.17. На листинге 4.21 представлен текст программы построения сферы с центром в точке (*1,1,1*) и радиусом *R=4*, а на рис. 4.23 изображена сфера.

```
clear all;
h=pi/30;
% Формируем вектор-столбец u.
u=[-0:h:pi]';
% Формируем вектор-строку v.
v=[0:2*h:2*pi];
% Формируем матрицу X, используя матричное произведение
% sin(u)*cos(v).
x=1+4*sin(u)*cos(v);
% Формируем матрицу Y, используя произведение
```

```
% sin(u)*sin(v).
y=1+4*sin(u)*sin(v);
% Формируем матрицу Z, используя произведение столбца
% cos(u) на строку ones(size(v)).
z=1+4*cos(u)*ones(size(v));
% Формируем график поверхности.
surf(x,y,z);
grid on;
% Подписываем график и оси.
title('SPHERE');
xlabel('X');
ylabel('Y');
zlabel('Z');
```

Листинг 4.21.

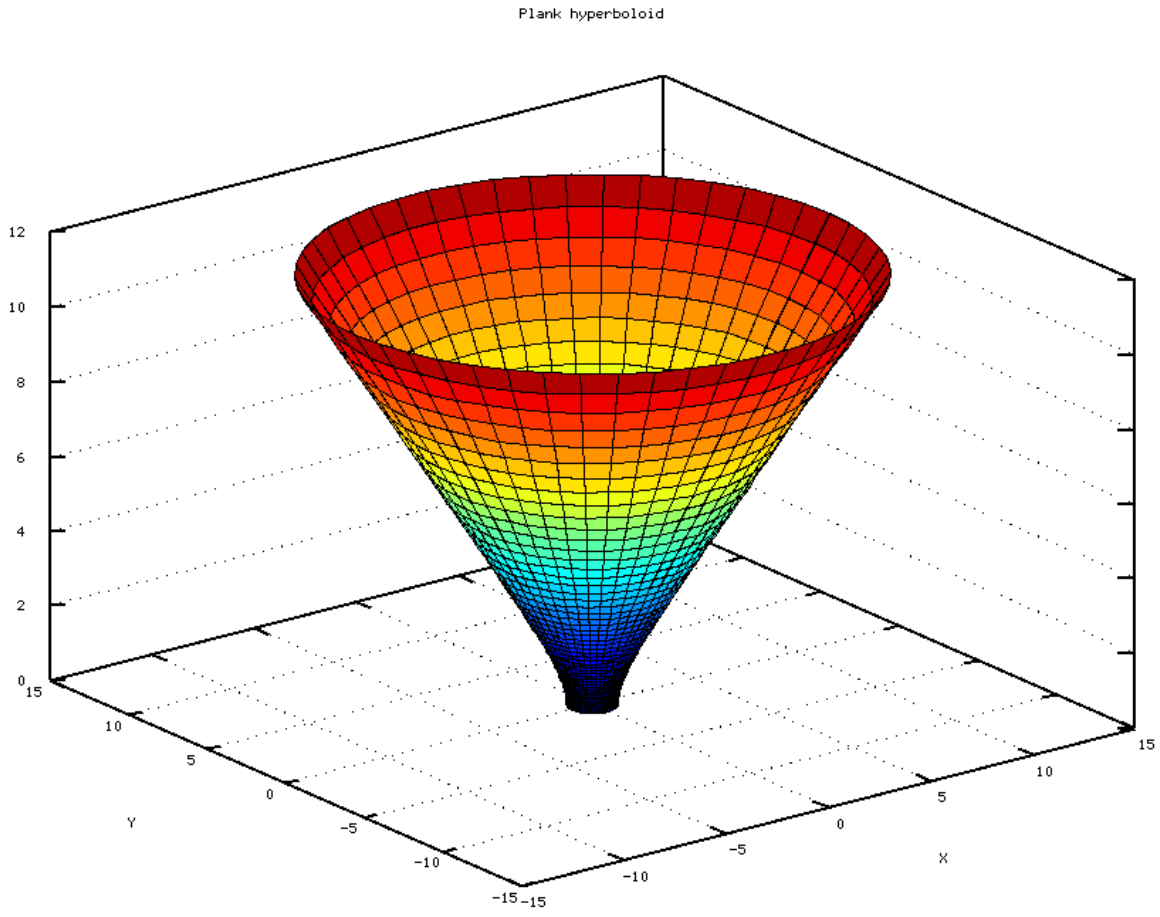


Рисунок *4.22*. График однополостного гиперболоида

Octave содержит встроенную функцию `[X,Y,Z]=sphere(n)`, позволяющую формировать матрицы (`X`, `Y`, `Z`) размерности `n+1` для построения сферы единичного радиуса (см. пример 4.18) с центром в начале координат.

Для построения сферы единичного радиуса с центром в начале координат достаточно двух команд `[X,Y,Z]=sphere(n);surf(X,Y,Z)`. Чем n больше, тем более «округлой» будет сфера. На рис. 4.24 изображена сфера единичного радиуса в центре в начале координат при `n=25`.

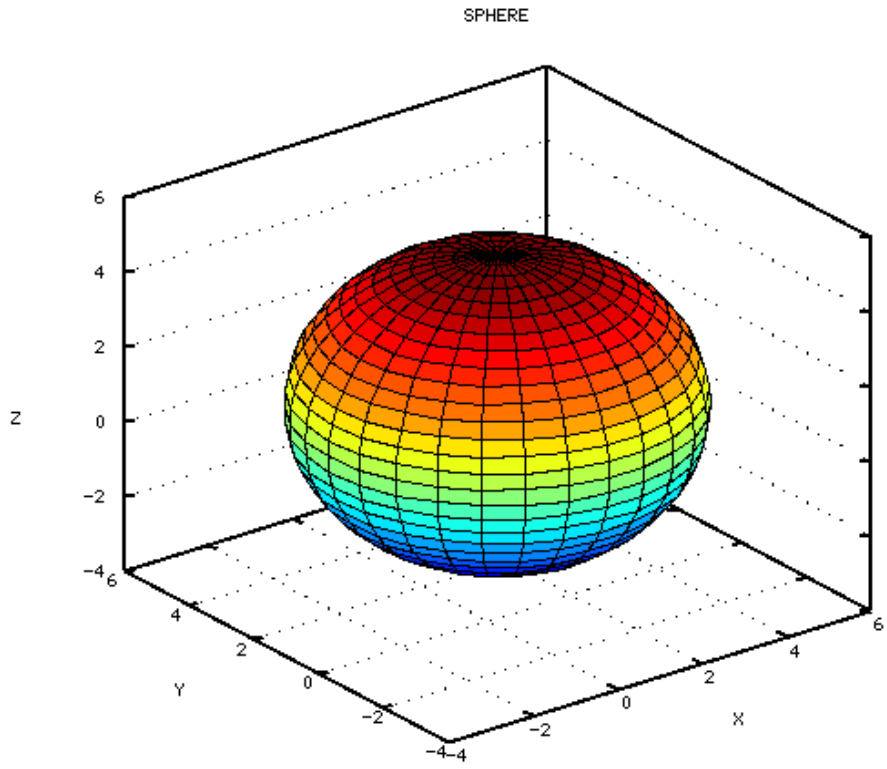


Рисунок *4.23* График сферы, построенный с использованием функции `surf`

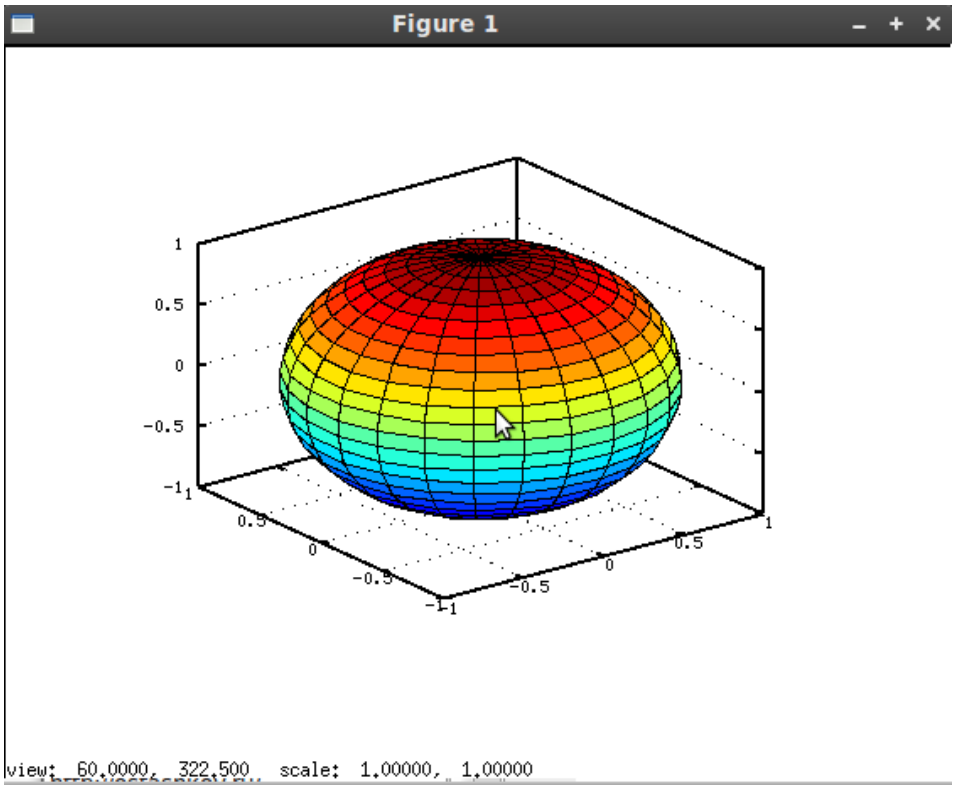


Рисунок *4.24*. График сферы единичного радиуса, построенный с использованием функции `sphere(25)`

Встроенную функцию `sphere(n)` можно использовать и для построения сферы с центром (x_0, y_0, z_0) и радиусом R. На листинге 4.22 приведено решение примера 4.18 с помощью функции `sphere(n)`.

```
clear all;
% Определяем центр и радиус сферы.
x0=2;y0=-2;z0=5;R=10
% Формируем матрицы X, Y, Z для построения сферы единичного
%радиуса с центром в начале координат, используя функцию
% sphere(n).
% cos(u) на строку ones(size(v)).
[X,Y,Z]=sphere(25);
% Пересчитываем матрицы X,Y,Z для сферы с центром x0, y0, z0 и
% радиусом R.
X=x0+R*X;
Y=y0+R*Y;
Z=z0+R*Z;
% Изображаем сферу
surf(X,Y,Z)
```

Листинг 4.22.

В результате работы программы будет построена сфера, представленная на рис. 4.25.

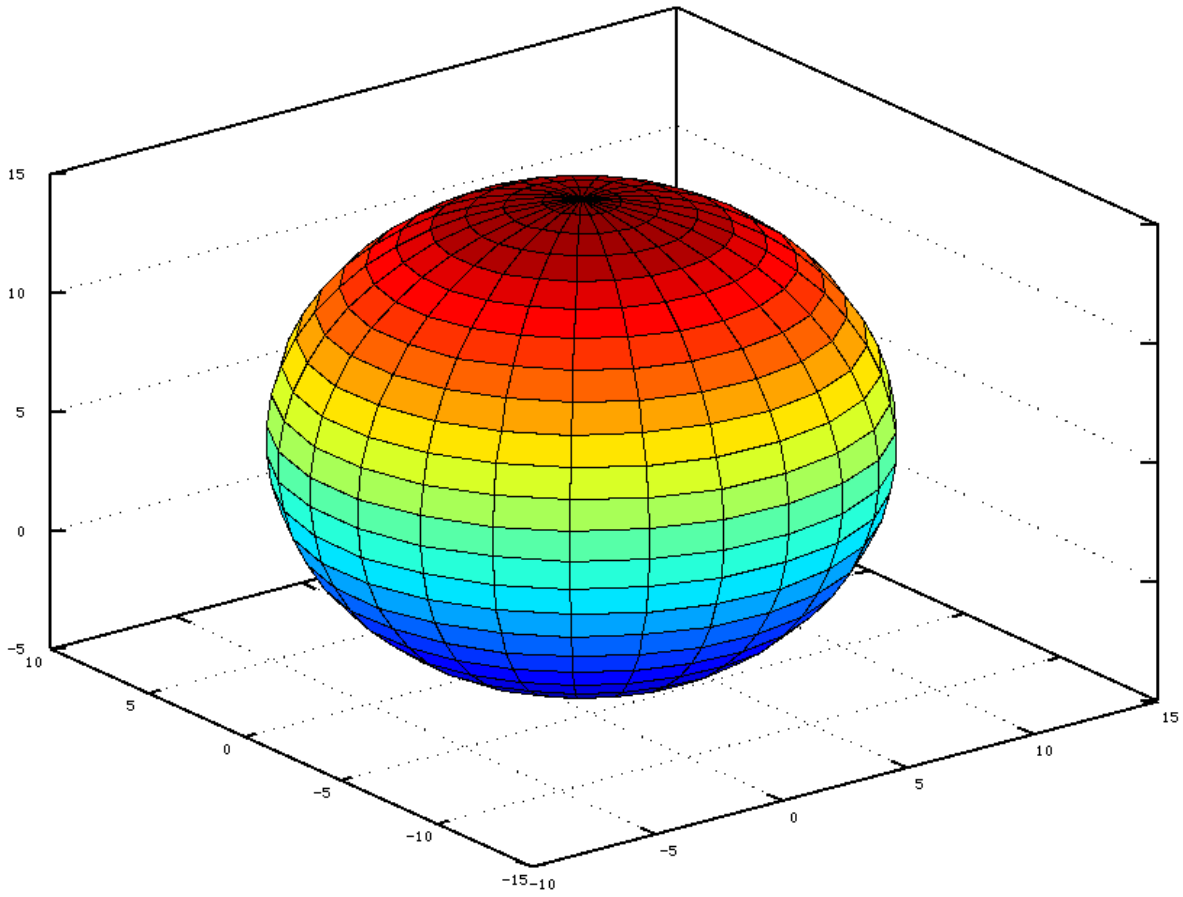


Рисунок *4.25*. Сфера, построенная с помощью программы, представленной на листинге *4.22*

Сфера является частным случае более общей фигуры эллипсоида. Рассмотрим два способа построения эллипсоида.

ПРИМЕР 4.19. Построить поверхность эллипсоида, уравнение которой задано в параметрическом виде $x(u,v)=x_0+a\sin(u)\cos(v),$ $y(u,v)=y_0+b\sin(u)\sin(v),$ $z(u,v)=z_0+c\cos(u)$. Здесь a, b, c – полуоси эллипсоида, x_0 , y_0 , z_0 – центр эллипсоида.

Методика построения эллипсоида подобна тому, как ранее были построены однополостный гиперболоид (пример 4.17) и сфера (пример 4.18). Для этого необходимо сформировать матрицы *X*, *Y* и *Z*, после чего вызвать функцию `surf`. Как это сделать показано на листинге 4.23.

```
clear all;
h=pi/30;
% Формируем вектор-столбец u.
u=[-0:h:pi]';
% Формируем вектор-строку v.
v=[0:2*h:2*pi];
% Формируем матрицу X, используя матричное произведение
% sin(u)*cos(v).
a=3;b=7;c=1;
x0=y0=z0=10;
x=x0+a*sin(u)*cos(v);
% Формируем матрицу Y, используя произведение
% sin(u)*sin(v).
y=y0+b*sin(u)*sin(v);
% Формируем матрицу Z, используя произведение столбца
% cos(u) на строку ones(size(v)).
z=z0+c*cos(u)*ones(size(v));
% Формируем эллипсоид.
surf(x,y,z);
grid on;
% Подписываем график и оси.
title('ELLIPSOID');
xlabel('X');
ylabel('Y');
zlabel('Z');
```

Листинг 4.23.

Эллипсоид с центром в точке (10, 10,10) и полуосями `a=3, b=7,c=1` представлен на рис. 4.23.

Однако, для построения эллипсоида в Octave существует функция `[X, Y, Z] = ellipsoid(xc,yc, zc, xr, yr, zr, n)`, которая позволяет автоматически сформировать матрицы `X`, `Y`, `Z`.

В функции `ellipsoid`

- `X`, `Y`, `Z` – формируемые для построения поверхности матрицы размерности n+1;
- `xc`, `yc`, `zc` – координаты центра эллипсоида;
- `xr`, `yr`, `zr` – полуоси эллипсоида.

Для построения эллипсоида, представленного на рис. 4.23 достаточно ввести команды

```
a=3;b=7;c=1;
```

```
x0=y0=z0=10;
[X, Y, Z]=ellipsoid (x0,y0, z0, a, b, c, 30);
surf(x,y,z);
grid on;
title('ELLIPSOID');
xlabel('X');
ylabel('Y');
zlabel('Z');
```

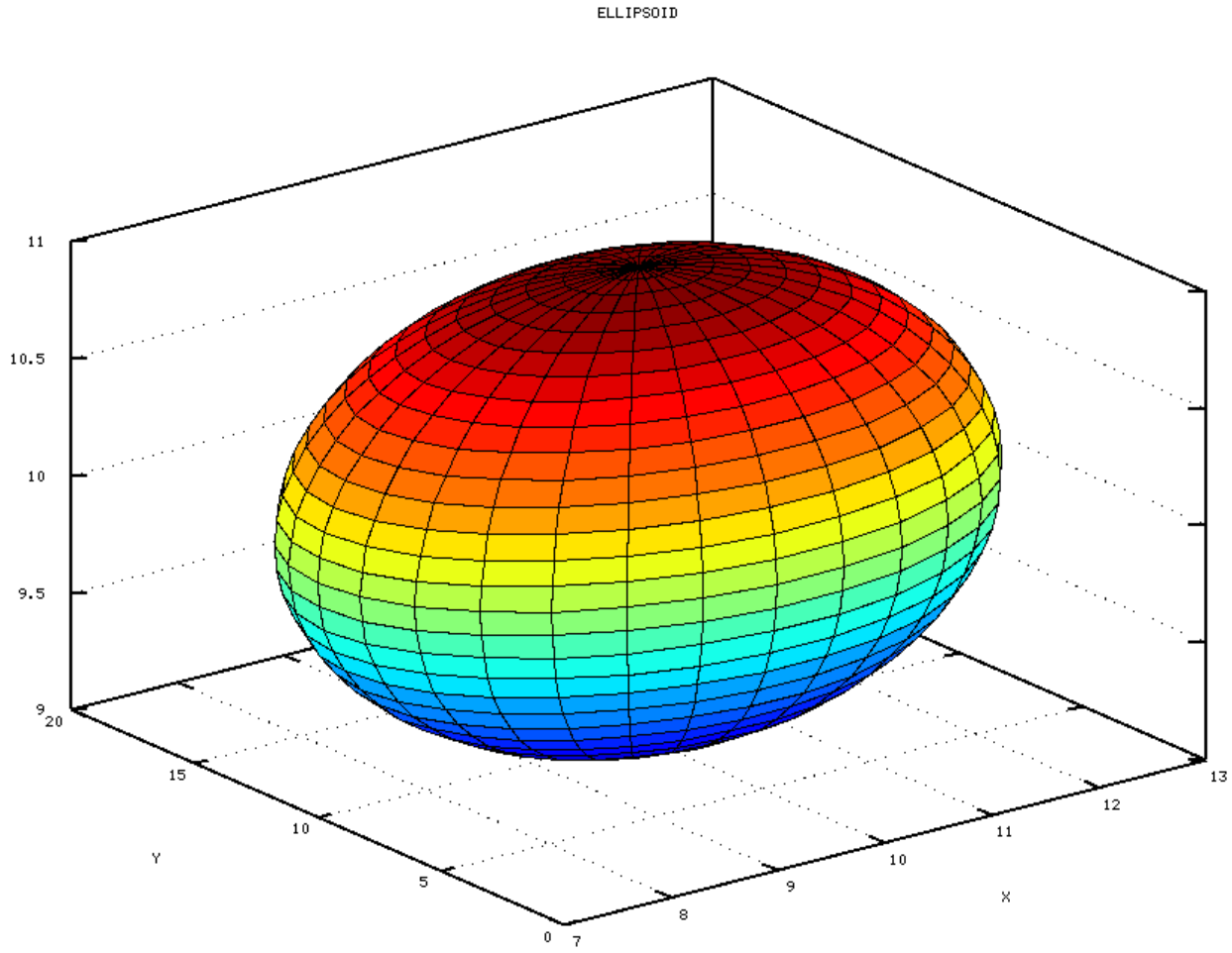


Рисунок *4.26*. Эллипсоид с центром в точке *(10, 10,10)* и полуосями *a=3, b=7,c=1*

Для построения цилиндров и круговых конусов можно использовать функцию `cylinder` для формирования матриц `X`, `Y`, `Z`. Затем строим саму поверхность (цилиндр, конус) с помощью функции `surf`.

Познакомимся с функцией `cylinder` подробнее. Обращение к функции имеет вид.

```
[X, Y, Z] = cylinder(r,n);
```

Здесь

`r` – массив радиусов; если мы строим цилиндр, `r` – массив, состоящий из двух одинаковых значений, функция требует, как минимум два значения и для построения цилиндра это будут радиус верхнего и нижнего основания; при построении конуса `r` является массивом радиусов горизонтальных сечений кругового конуса;

`X`, `Y`, `Z` – формируемые для построения поверхности (конуса, цилиндра) матрицы размерности `n+1`.

Рассмотрим несколько примеров.

ПРИМЕР 4.20. Построить цилиндр радиуса *R=4* и высотой *h=1*.
Текст программы приведен на листинге 4.24, график – на рисунке 4.27.

```
clear all
%Формирование матриц x, y, z.
[x, y, z] = cylinder ([4,4],25);
grid on;
% Построение цилиндра.
surf(x, y, z);
title ("Cylinder")
```

Листинг 4.24.

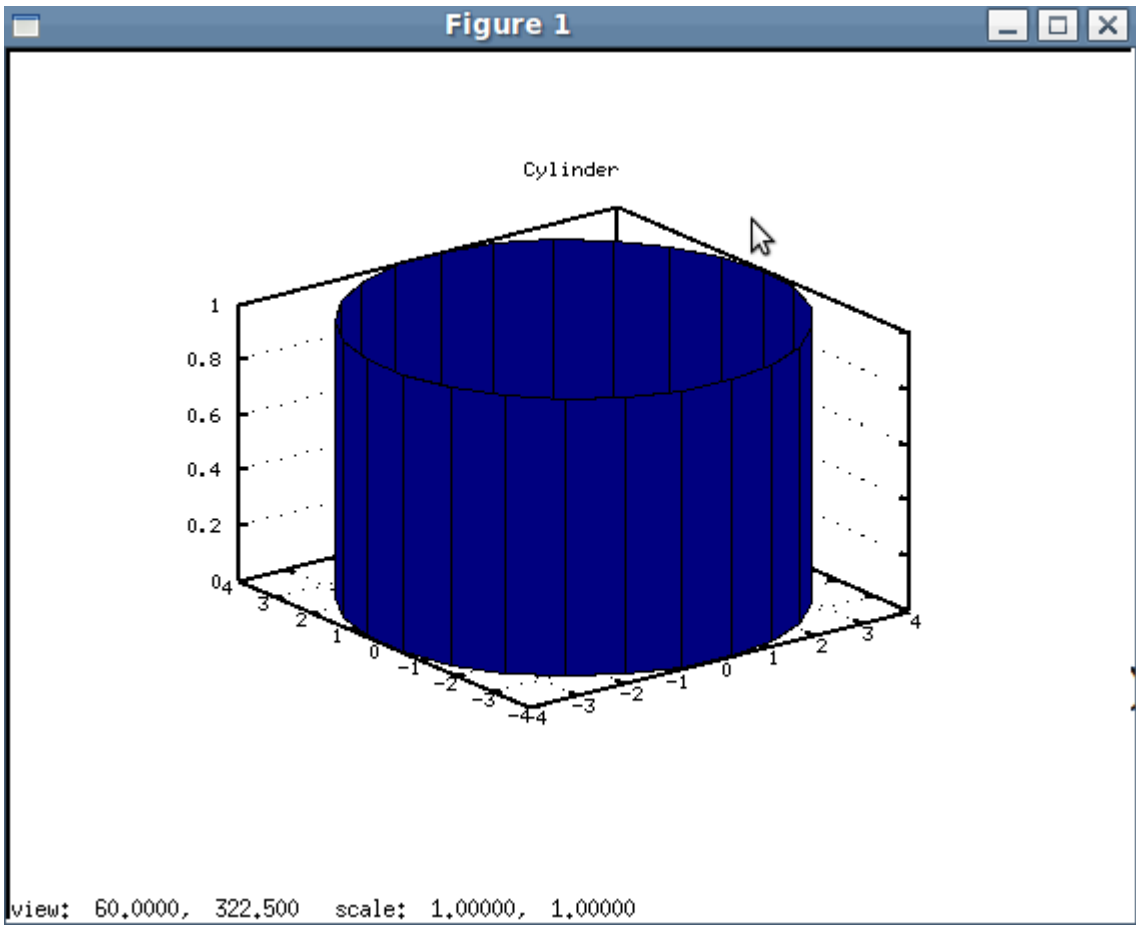


Рисунок *4.27. Цилиндр радиуса R=4 и высотой h=1*

ПРИМЕР 4.21. Построить цилиндр радиуса 4 и высотой 20.
Текст программы приведен на листинге 4.25, график – на рисунке 4.28.

```
clear all
%Формирование матриц x, y, z.
[x, y, z] = cylinder ([4,4],25);
grid on;
% Построение цилиндра с учётом высоты h=20.
surf (x, y, 20*z);
title ("Cylinder")
```

Листинг 4.25.

ПРИМЕР 4.22. Примеры круговых конусов.
Рассмотрим несколько примеров.

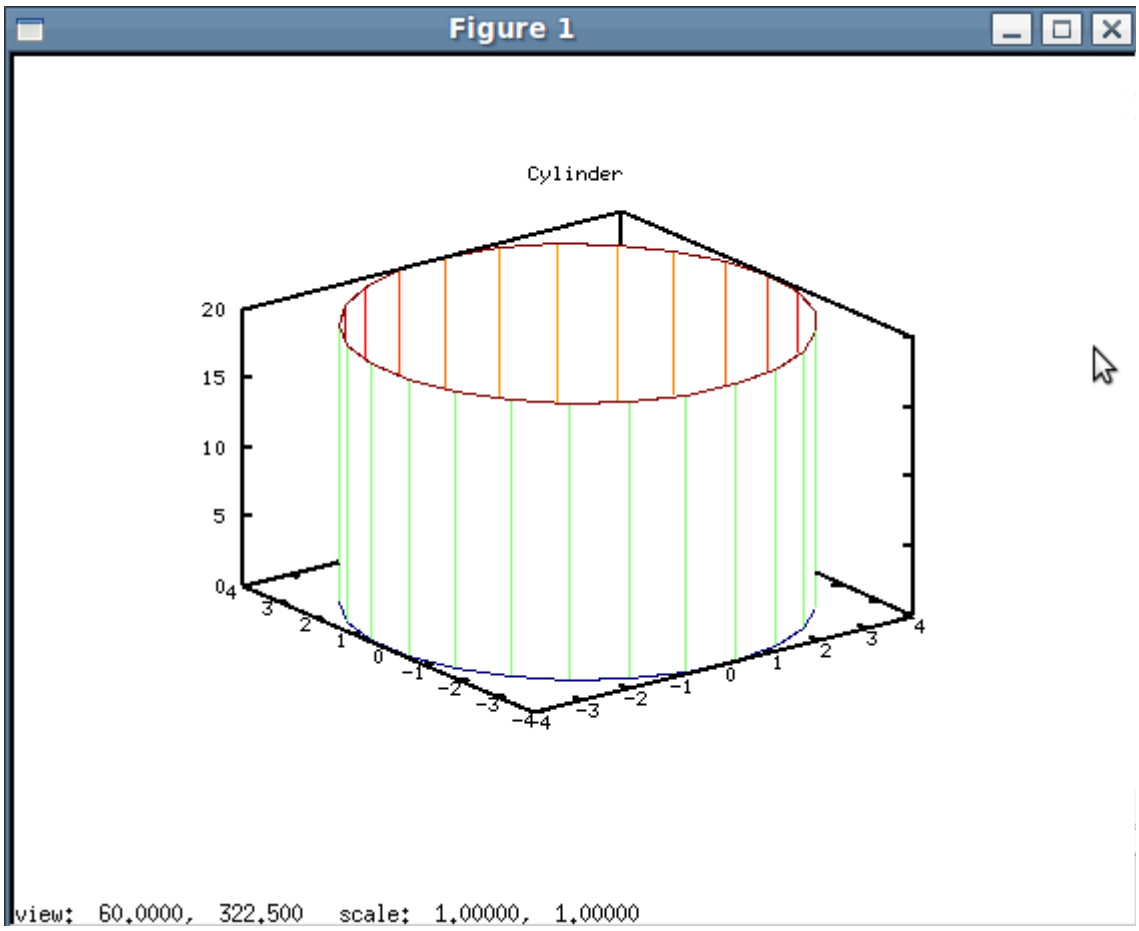


Рисунок *4.28. Цилиндр радиуса R=4 и высотой h=20*

Усеченный круговой конус, представленный на рис. 4.29, генерируется программой на листинге 4.26.

```
clear all
[x, y, z] = cylinder (2:1:10,25);
grid on;
surf (x, y, z);
title ("Cone")
xlabel('X');
ylabel('Y');
zlabel('Z');
```

Листинг 4.26.

Круговой конус, представленный на рис. 4.30, генерируется программой на листинге 4.27.

```
clear all
[x, y, z] = cylinder ([5,4,3,2,1,0,1,2,3,4,5],25);
grid on;
mesh(x, y, z);
title ("Cone")
xlabel('X');
ylabel('Y');
zlabel('Z');
```

Листинг 4.27.

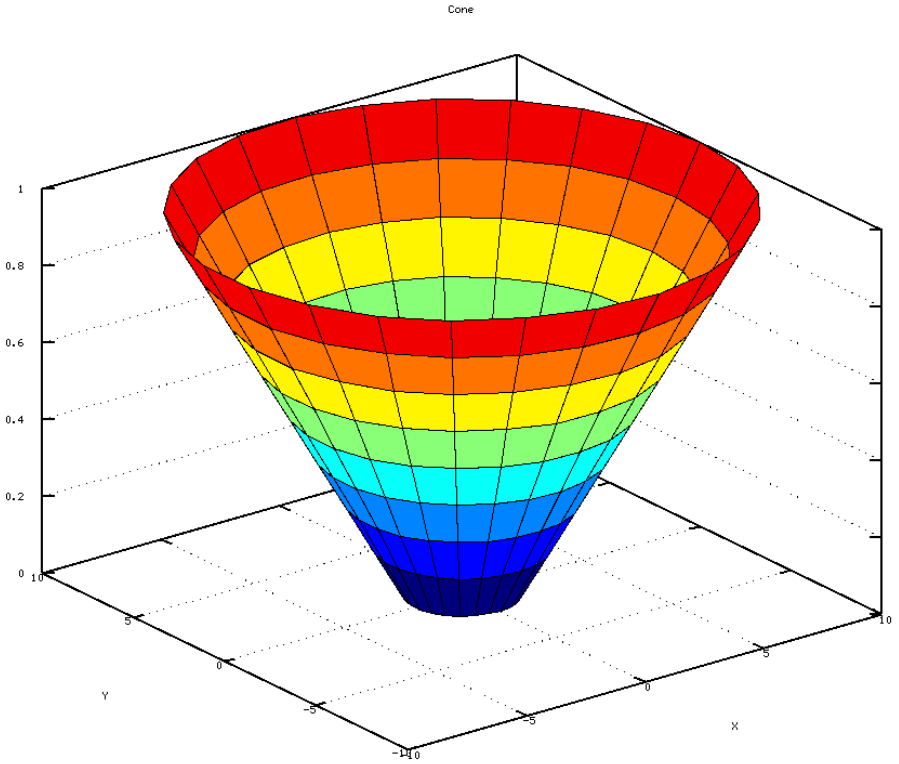


Рисунок *4.29* Усеченный круговой конус к листингу *4.26*

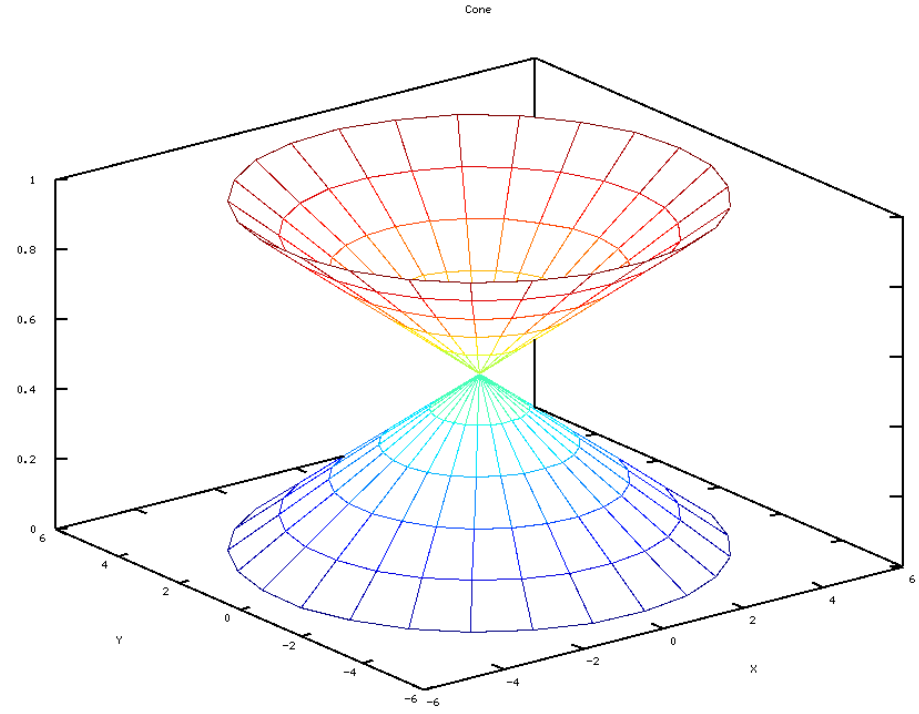


Рисунок *4.30*. Круговой конус к листингу *4.26*

В завершении приведен листинг 4.28, который генерирует поверхность, представленную на рис. 4.31.

```
clear all
```

```
[x, y, z] = cylinder([1,3,5,7,6,4],25);
surf(x, y, z);
title ("Surface")
xlabel('X');
ylabel('Y');
zlabel('Z');
```

Листинг 4.28.

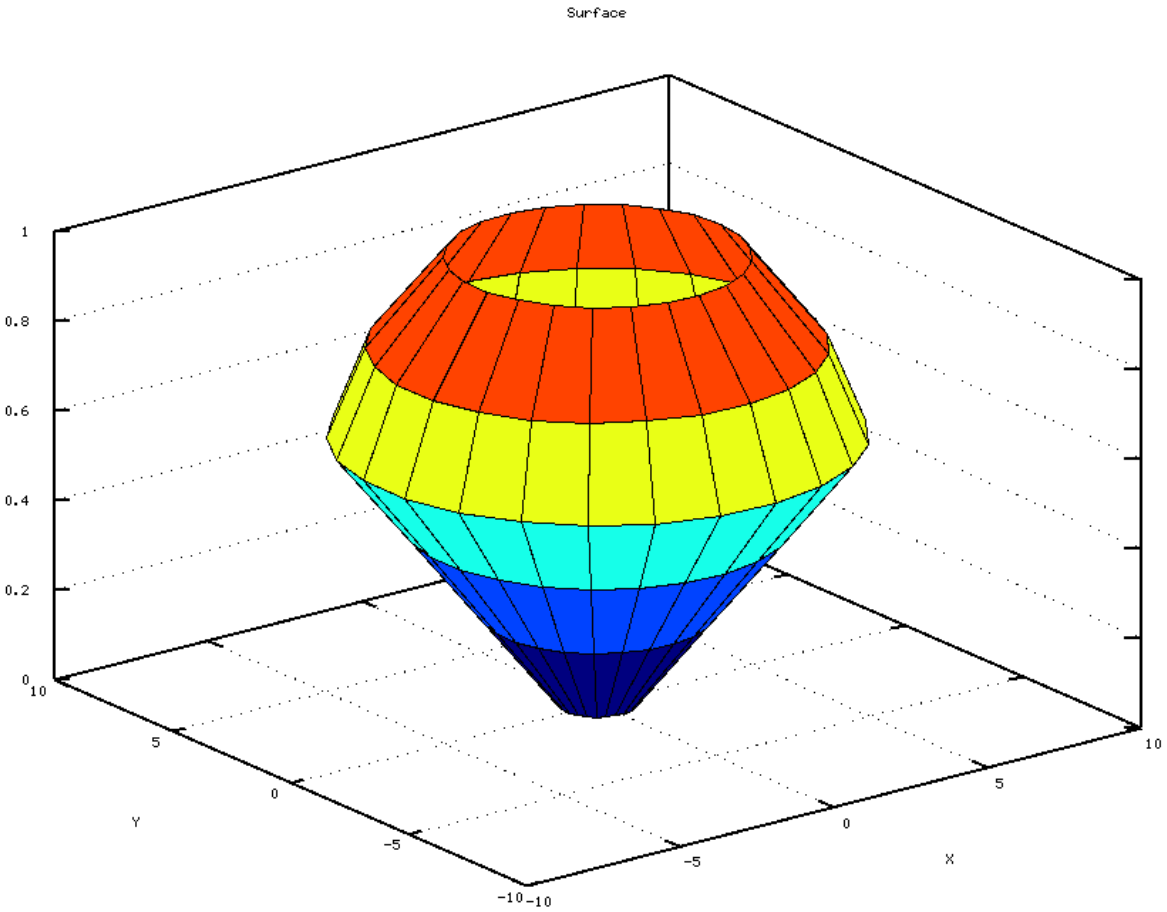


Рисунок *4.31*. Поверхность к листингу *4.28*

4.3 Анимация

При изучении движения точки на плоскости Octave позволит построить график движения и проследить за движением. Построить анимационный ролик можно с помощью функции `comet(x,y)`, которая позволит увидеть движение точки вдоль кривой y(x) на плоскости.

Для движения точки на плоскости вдоль синусоиды достаточно ввести команды

```
x=0:pi/30:6*pi;
y=sin(x);
comet(x,y);
```

Листинг 4.29

Процесс движения точки вдоль эллипса представлен на рис. 4.32, окончательный вид траектории движения точки вдоль эллипса показан на рис. 4.33.

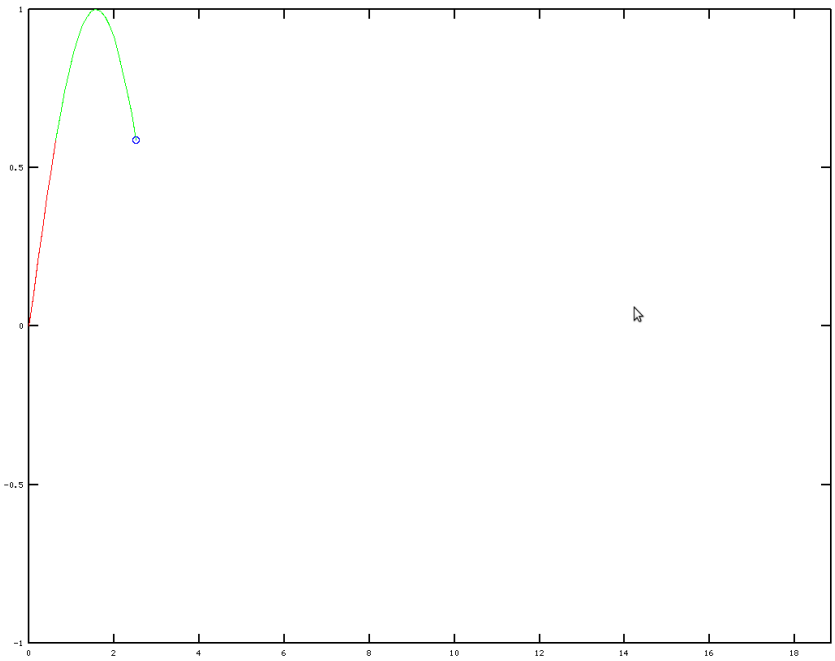


Рисунок *4.32*. Движение точки вдоль синусоиды

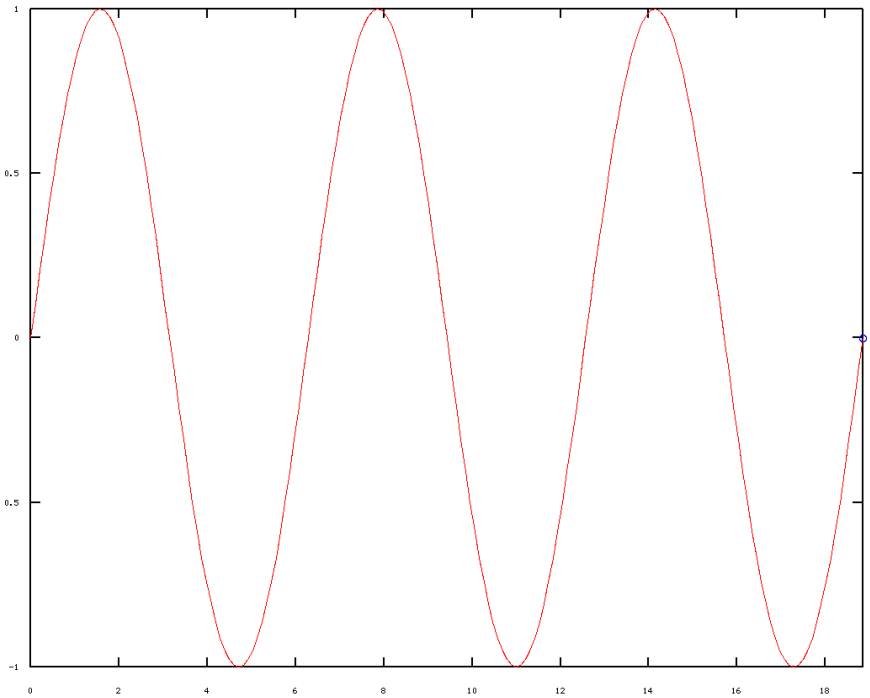


Рисунок *4.33*. *Окончательный вид траектории движения точки*

4.4 Графические объекты *Octave*

Встроенный язык Octave – объектно-ориентированный язык программирования. Все объекты находятся в определенной иерархии по отношению друг к другу. Рассмотрим основные графические объекты для работы с графикой и общие принципы работы с объектами на примере построения графика функции $x(t)=\sin(t)$ на интервале $[-3\pi;3\pi]$. Построим график функции x(t) (см. листинг. 4.30). Окно с графиком синуса на интервале $[-3\pi;3\pi]$ представлено на рис. 4.34.

```
t=-3*pi:pi/100:3*pi;
x=sin(t);
plot(t,x);
```

Листинг 4.30.

В результате работы функции `plot` были созданы три графических объекта:

- графическое окно **Figure 1**;
- линия графика `sin(t)`;
- оси.

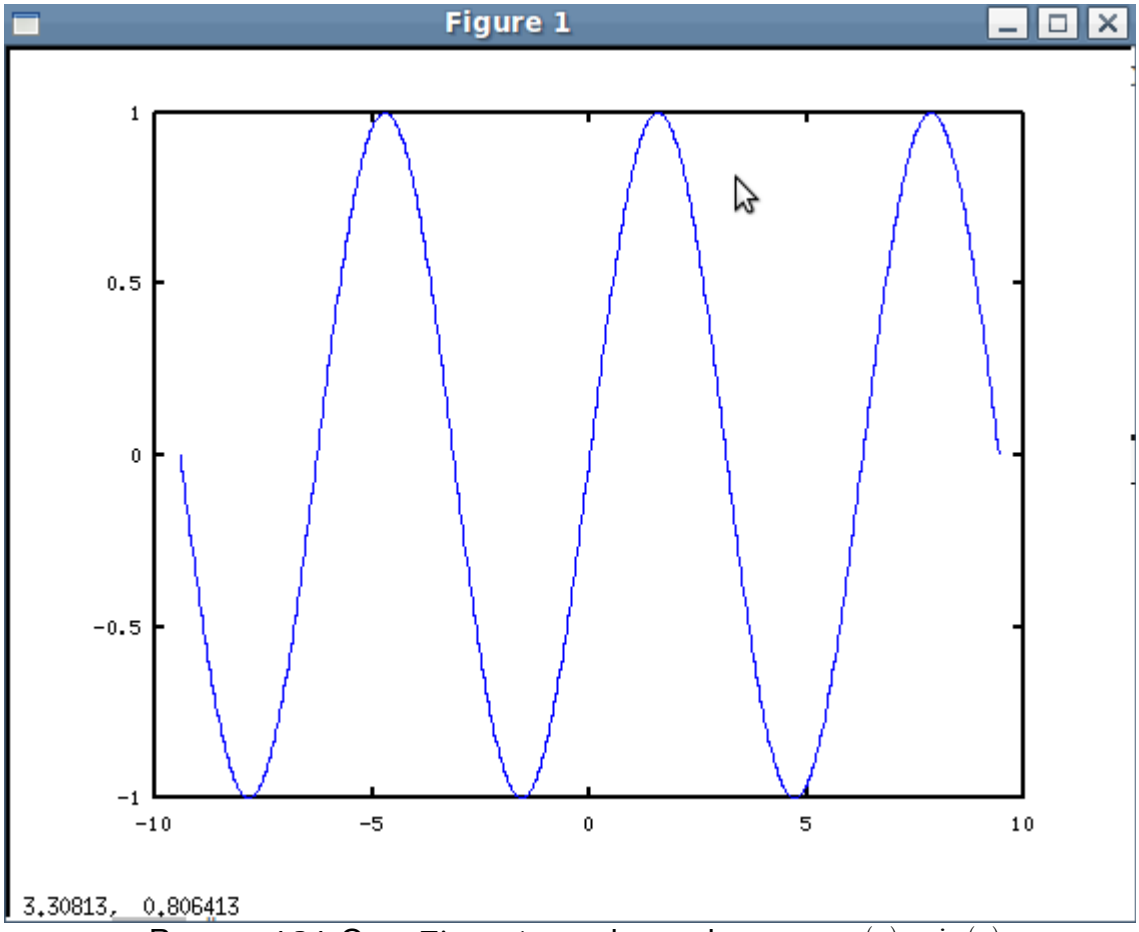


Рисунок *4.34*. Окно *Figure 1* с графиком функции $x(t)=\sin(t)$ на интервале $[-3\pi;3\pi]$

Для получения указателей на объекты. При работе с переменными, в которых хранятся объекты, пользователь оперирует ими, как обычными переменными. Однако, реально это указатели – адреса в памяти, в которых хранятся объекты, в Octave в качестве указателя используется номер объекта.

В яыке Octave есть три функции:

- `gcf()`[3] возвращает указатель на текущее графическое окно;
- `gca()` возвращает указатель на текущие оси;
- `gco()` возвращает указатель на текущий графический объект.

4.4.1 Свойства графических объекта

Для установки свойств объектов служит функция `set`

```
set(h, 'Свойство1', Значение1, 'Свойство2', Значение2, 'Свойство3', Значение3, …)
```

3 Синтаксис Octave допускает обращение и без скобок (gcf — это верно).

Здесь

- `h` – указатель на объект, свойства которого будут устанавливаться (изменяться);
- `'Свойство1', 'Свойство2', 'Свойство3', …` – имена свойств, которые будут изменяться;
- `Значение1, Значение2, Значение3, …` – новые значения свойств.

В простейшем виде функция `set` имеет вид:

```
set(h, 'Свойство', Значение)
```

Для получения свойства объекта служит функция `get`

```
get(h, 'Свойство');
```

Функция возвращает значения `Свойства` объекта с указателем `h`.

Если к функции `get` обратиться с одним параметром `h`, то функция вернет значения всех свойств объекта в виде `Свойство = Значение`.

4.4.2 Работа с графическим окном

Как уже рассматривалось ранее, для создания графического служит функция `figure`(), которая создает пустое графическое окно (см. рис. 4.35) и возвращает указатель на него.

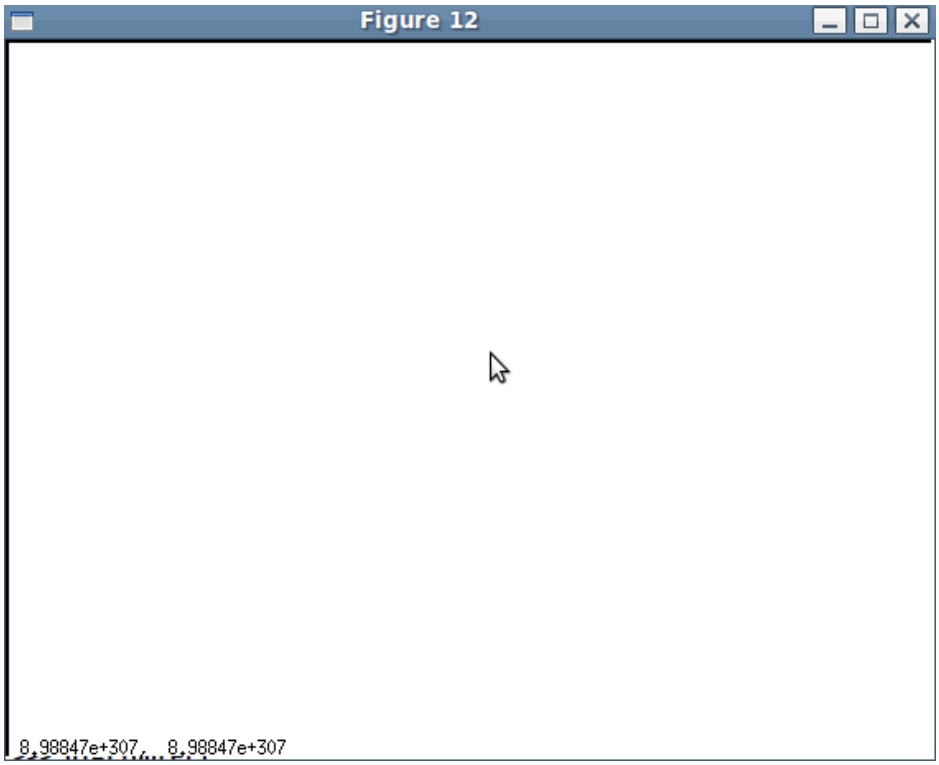


Рисунок *4.35*. Окно, созданное с помощью функции *figure()*

Например:

```
>>> g=figure()
g = 12
```

Если есть несколько окон, то окно с указателем `g` выдвигается на передний план и становится текущим.

Как при создании графиков с определенными свойствами с помощью функции `plot`, при создании графических окон, осей и других объектов можно сразу определять некоторые свойства создаваемых объектов. Обращение к функции создания окна с определёнными свойствами имеет вид

```
figure('Свойство1', Значение1, 'Свойство2', Значение2, 'Свойство3', Значение3, …);
```

Для удаления (закрытия) окна с указателем `h` служит функция `delete(h)` .

Доступ к имени окна осуществляется с помощью свойства `name`, которое определяет строку, которая будет дописана к имени окна после стандартного имени окна `Figure 1`, `Figure 2`, … ;

Например,

```
h=figure();
set(h,'name','New Window')
```

В результате появится окно, представленное на рис. 4.36.

Если свойству `numbertitle` присвоить значение '`off`', то это позволит отказаться от текущей нумерации окон `Figure 1`, `Figure 2`, … (см. листинг 4.31 и рис. 4.37)

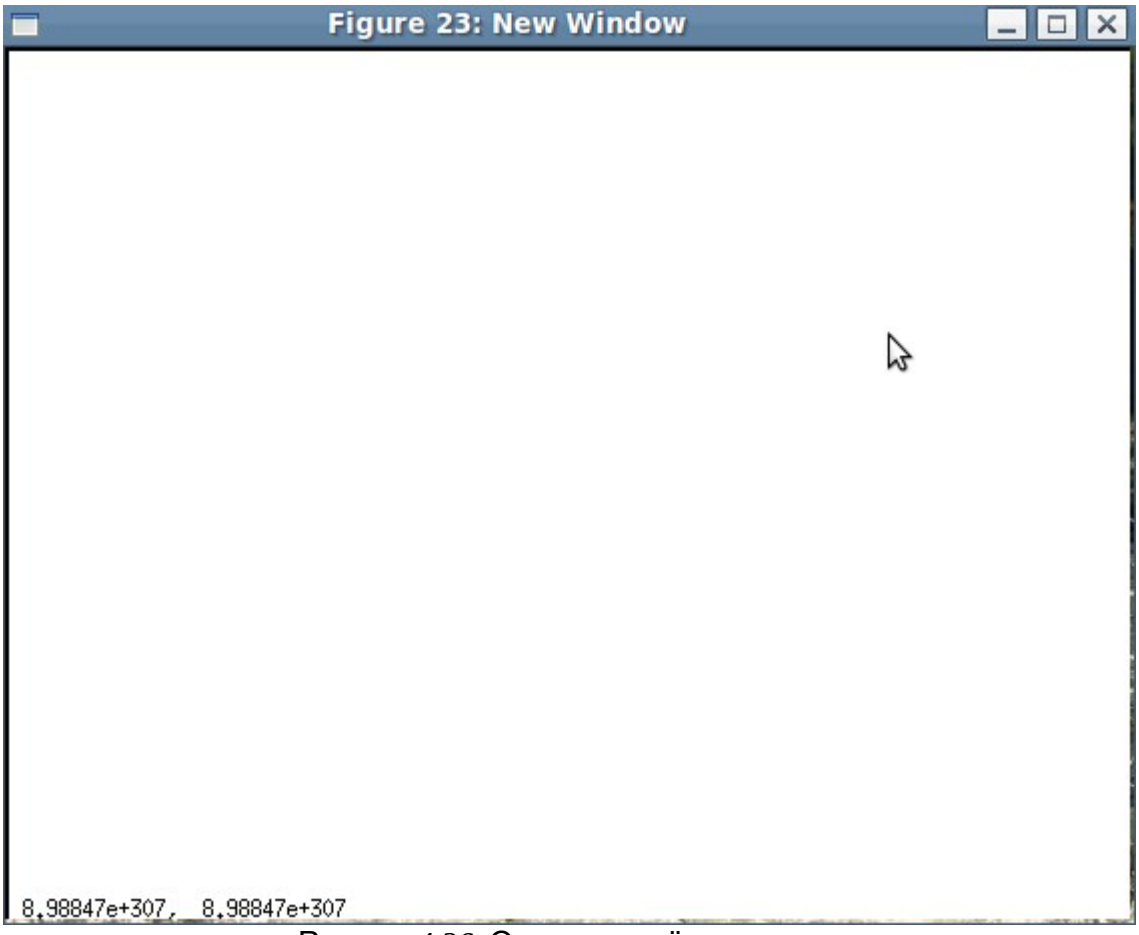


Рисунок *4.36*. Окно с изменённым заголовком

```
h=figure();
set(h,'numbertitle','off')
set(h,'name','New Window')
```

Листинг 4.31

Как и у многих рассматриваемых графических объектов у окна есть свойство `Position`, определяющее расположение объекта. `Position` – массив из четырех элементов [`xleft ybottom width height`], `xleft`, `ybottom` определяют координаты левого нижнего угла экрана, относительно левого нижнего угла монитора `weight` – ширина, `height` – высота графического окна в пикселях.

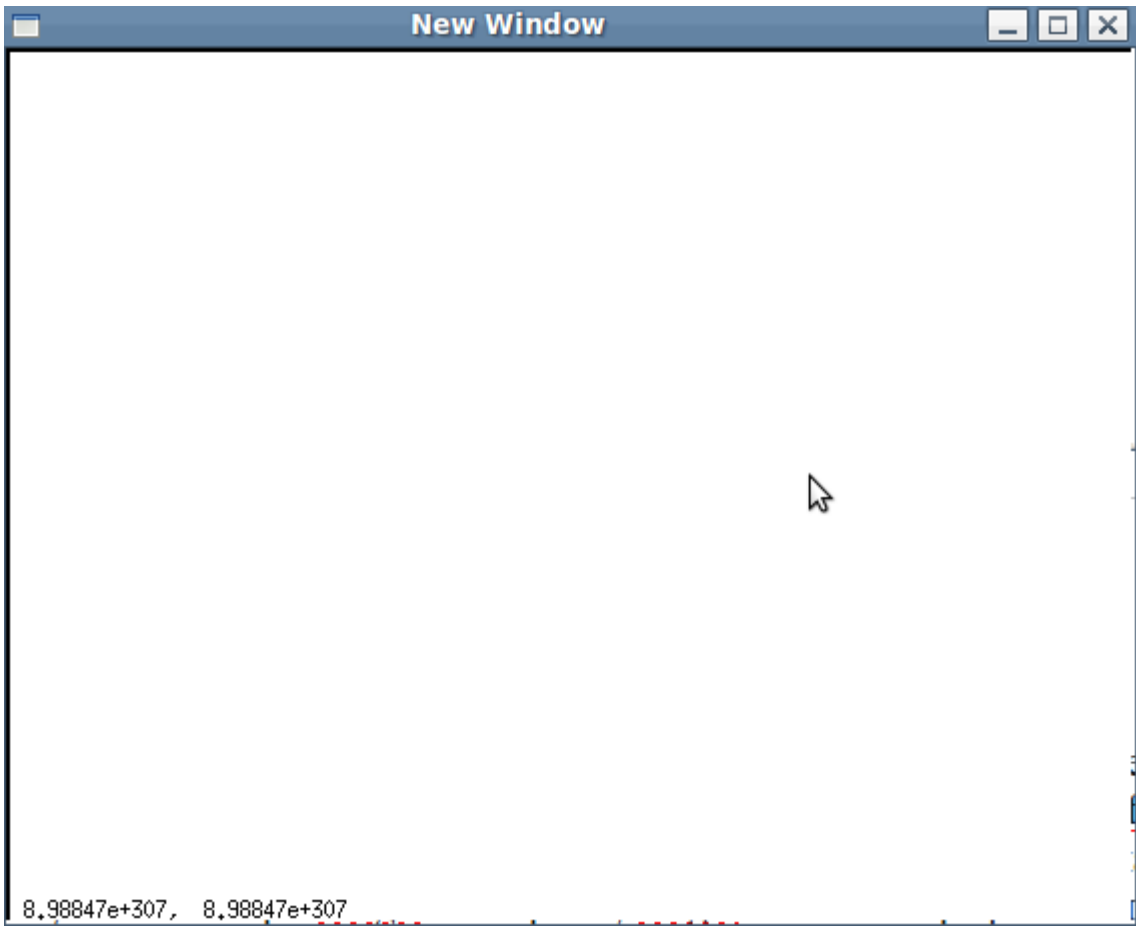


Рисунок *4.37*. Окно после выключения стандартной нумерации окон

Создадим окно с именем «Our Window» шириной и высотой 400 пикселей, с левым нижним углом с координатами (75, 90) (см. листинг 4.32)[4].

```
h=figure('position',[75 90 400 400])
set(h,'numbertitle','off')
set(h,'name','New Window')
```

Листинг 4.32

Для создания осей в текущем окне служит функция `axes`, которая возвращает указатель на созданные оси.

4.4.3 Свойства осей графика

Для получения всех свойств осей построенного с помощью листинга 4.30 графика пользователь может ввести команду

```
get(gca)
```

На листинге 4.33 приведены свойства осей графика, представленного на рис. 4.29.

```
>>>ans =
{
beingdeleted = off
busyaction = queue
buttondownfcn = [](0x0)
children = -6.1976
clipping = on
```

4 У авторов при использовании GNU Octave, version 3.2.3 после изменения свойства position у окна с помощью функции set не происходило перерисовывания окна на новом месте; будем надеяться, что в последующих версиях эта проблема будет устранена.

```
createfcn = [](0x0)
deletefcn = [](0x0)
handlevisibility = on
hittest = on
interruptible = on
parent = 1
selected = off
selectionhighlight = on
tag =
type = axes
userdata = [](0x0)
visible = on
__modified__ = on
uicontextmenu = [](0x0)
position =
0.13000 0.11000 0.77500 0.81500
box = on
key = off
keybox = off
keyreverse = off
keypos = 1
colororder =
0.00000 0.00000 1.00000
0.00000 0.50000 0.00000
1.00000 0.00000 0.00000
0.00000 0.75000 0.75000
0.75000 0.00000 0.75000
0.75000 0.75000 0.00000
0.25000 0.25000 0.25000
dataaspectratio =
20 2 1
dataaspectratiomode = auto
layer = bottom
xlim =
-10 10
ylim =
-1 1
zlim =
0 1
clim =
0 1
alim =
0 1
xlimmode = auto
ylimmode = auto
zlimmode = auto
climmode = auto
alimmode = auto
xlabel = -5.3352
ylabel = -4.7682
zlabel = -3.2778
```

```
    title = -2.5540
    xgrid = off
    ygrid = off
    zgrid = off
    xminorgrid = off
    yminorgrid = off
    zminorgrid = off
    xtick =
    -10 -5 0 5 10
    ytick =
    -1.00000 -0.50000 0.00000 0.50000 1.00000
    ztick = [](0x0)
    xtickmode = auto
    ytickmode = auto
    ztickmode = auto
    xminortick = off
    yminortick = off
    zminortick = off
    xticklabel =
    yticklabel =
    zticklabel =
    xticklabelmode = auto
    yticklabelmode = auto
    zticklabelmode = auto
    interpreter = none
    color =
    1 1 1
    xcolor =
    0 0 0
    ycolor =
    0 0 0
    zcolor =
    0 0 0
    xscale = linear
    yscale = linear
    zscale = linear
    xdir = normal
    ydir = normal
    zdir = normal
    yaxislocation = left
    xaxislocation = bottom
    view =
    0 90
    nextplot = replace
    outerposition =
    0 0 1 1
    activepositionproperty = outerposition
    ambientlightcolor =
    1 1 1
    cameraposition =
    0.00000 0.00000 9.16025
```

```
cameratarget =
0.00000 0.00000 0.50000
cameraupvector =
-0 2 0
cameraviewangle = 6.6086
camerapositionmode = auto
cameratargetmode = auto
cameraupvectormode = auto
cameraviewanglemode = auto
currentpoint =
0 0 0
0 0 0
drawmode = normal
fontangle = normal
fontname = *
fontsize = 12
fontunits = points
fontweight = normal
gridlinestyle = :
linestyleorder = -
linewidth = 0.50000
minorgridlinestyle = :
plotboxaspectratio =
1 1 1
plotboxaspectratiomode = auto
projection = orthographic
tickdir = in
tickdirmode = auto
ticklength =
0.010000 0.025000
tightinset =
0 0 0 0
units = normalized
```

Листинг 4.33.

Рассмотрим наиболее часто используемые свойства осей:

• `box` определяет, заключать оси в прямоугольную рамку '`on`' (значение по умолчанию) или нет '`off`';

• `color` должен определять цвет фона графика, цвет задается в формате **RGB** `[r g b]`, где `r`, `g`, `b` - яркость красного, зеленого и синего цветов соответственно, которая меняется от 0 до 1 (см. табл. 4.4) или один из предопределенных цветов.

Таблица *4.4:* Наиболее распространенные цвета

Цвет	**Цвет в формате RGB**
Черный	[0 0 0]
Синий	[0 0 1]
Темно-синий	[0 0 128/255]
Зеленый	[0 1 0]
Темно-зеленый	[0 128/255 0]

Голубой	[0 1 1]
Темно-голубой	[0 128/255 128/255]
Красный	[1 0 0]
Темно-красный	[128/255 0 0]
Пурпурный	[1 0 1]
Темно-пурпурный	[128/255 0 128/255]
Желтый	[1 1 0]
Темно-желтый	[128/255 128/255 0]
Темно-серый	[128/255 128/255 128/255]
Светло-серый	[192/255 192/255 192/255]
Белый	[1 1 1]

• `fontangle` позволит установить наклон шрифта разметки осей '`italic`' или не использовать наклон '`normal`' (значение по умолчанию);

• `fontname` определяет название шрифта, используемого при подписи осей (например, '`Arial`');

• `fontsize` определяет размер шрифта в пунктах;

• `fontweight` определяет толщину шрифта, наиболее часто используемые значения '`normal`' (по умолчанию) и '`bold`';

• `gridlinestyle` позволяет изменять стиль линий сетки, значения стиля линий подробно рассмотрены при описании функции `plot;`

• `linewidth` определяет толщину линий осей, значение по умолчанию 0.5;

• `visible` – видимость осей: '`on`' (значение по умолчанию) – оси видимы, '`off`' – оси невидимы;

• `xcolor`, `ycolor`, `zcolor` определяет цвет соответствующей оси в формате **RGB**;

• `xdir`, `ydir`, `zdir` определяет направление соответствующей оси: нормальное '`normal`' (значение по умолчанию) или обратное '`reverse`';

• `xgrid`, `ygrid`, `zgrid` определяет наличие '`on`' или отсутствие '`off`' (значение по умолчанию) сетки, перпендикулярной оси;

• `xaxislocation` определяет расположение оси X: сверху – '`top`' или снизу – '`bottom`' (значение по умолчанию);

• `yaxislocation` определяет расположение оси Y: справа – '`right`' или слева – '`left`' (значение по умолчанию);

• `xlim`, `ylim`, `zlim` задают пределы изменения переменных *x*, *y* и *z* в виде массива из двух значений;

• `xscale`, `yscale` и `zscale` определяют масштаб соответствующих осей: линейный '`linear`' (значение по умолчанию) или логарифмический '`log`';

• `xtick`, `ytick`, `ztick` – вектора, определяющие координаты разметки соответствующих осей.

На листинге 4.34 представлены команды, изменяющие внешний вид осей графика, изображенного на рис. 4.29. График функции $x=\sin(t)$ на интервале $[-6\pi;6\pi]$ после их применения представлен на рис. 5.2.

```
h=figure();
t=-3*pi:pi/100:3*pi;
x=sin(t);
```

```
plot(t,x);
% Убираем прямоугольную сетку вокруг оси.
set(gca,'box','off');
% Определяем шрифта.
set(gca,'fontname','Arial');
% Определяем размер шрифта 20.
set(gca,'fontsize',20);
% Включаем линии сетки, перпендикулярные OX и OY.
set(gca,'xgrid','on');
set(gca,'ygrid','on');
% Устанавливаем координаты линий сетки, перпендикулярной OX.
set(gca,'xtick',[-3 -1 0 1 2] );
```

Листинг 4.34.

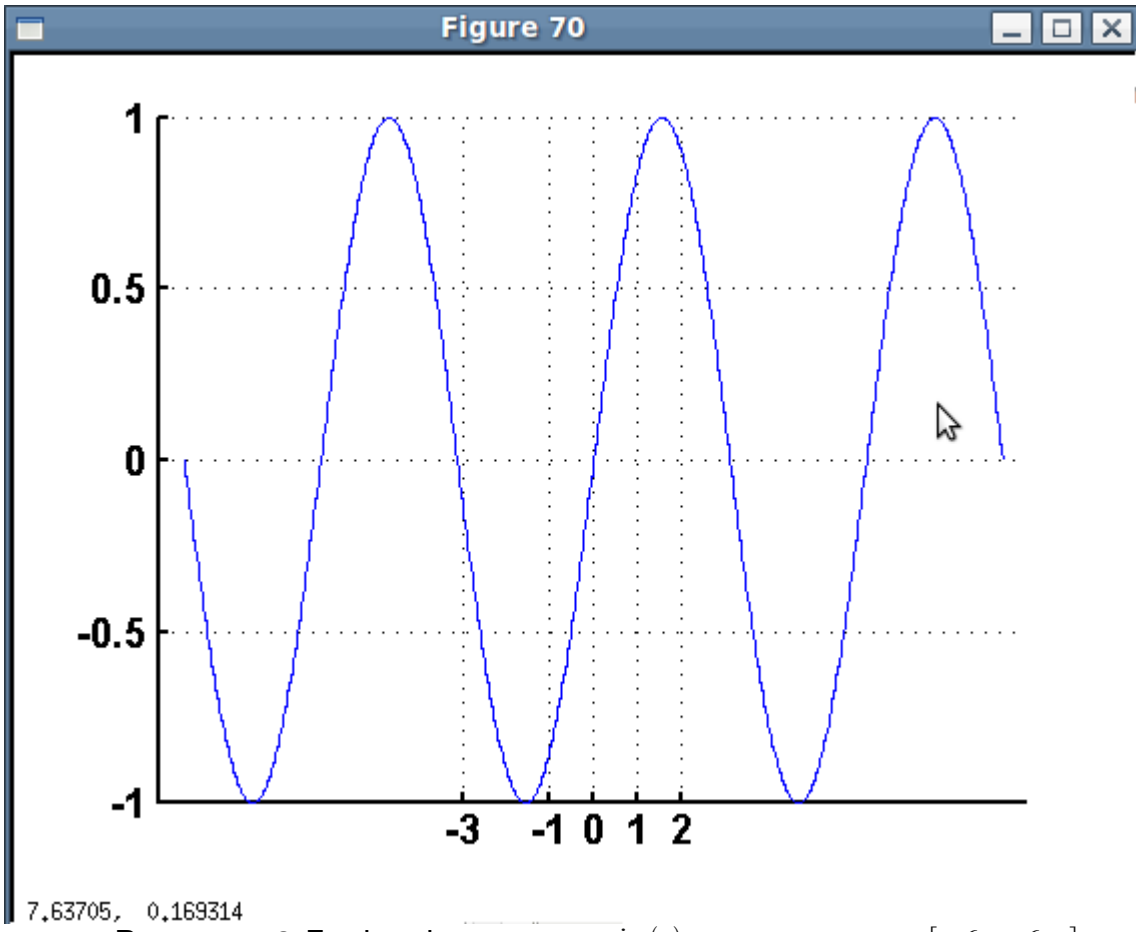


Рисунок *4.38.* График функции $x=\sin(t)$ на интервале $[-6\pi;6\pi]$ после выполнения команд из листинга *4.34*

Обращение к функции создания осей с определёнными свойствами имеет вид:

```
axes('Свойство1', Значение1, 'Свойство2', Значение2, 'Свойство3', Значение3, …);
```

С помощью функции `set` можно также изменять свойства линий, которые формируется с помощью подробно рассмотренной ранее функции `plot`.

Рассмотрим наиболее часто используемые свойства линий:

- `color` определяет цвет текущей линии в формате RGB или с помощью предопределенного цвета;
- `linestyle` устанавливает стиль линий;
- `linewidth` определяет толщину линии в пунктах;

• `marker` устанавливает тип маркера для изображения точек на графике.
• `markersize` определяет размер маркера в пунктах.

4.4.4 Удаление и очистка объектов

Для того, чтобы удалить объект в графическом окне, необходимо вызвать функцию
`delete(h)`

где `h` – указатель на удаляемый объект (указатель на линию, оси и т.д.). Следует понимать, что удаление осей приведет к исчезновению всех объектов, которые располагались на осях.

Очистка текущих осей осуществляется функцией `cla`, очистка текущего окна – функцией `clf`.

Рассмотрим описанные возможности работы с окнами на нескольких примерах. Авторы рекомендуют читателю внимательно изучить примеры 4.23, 4.24, в которых собраны стандартные приемы работы с окнами, линиями графиков и осями и их свойствами.

4.4.5 Примеры

ПРИМЕР 4.23. Написать программу создания графиков функций $x(t)=e^{\cos(t)}, y=e^{\sin(t)}, z=\cos(t^2)$ на интервале [-5;5]. Графики функций $x(t)=e^{\cos(t)}, y=e^{\sin(t)}$ изобразить в графическом окне с именем **WINDOW1** красным и синим цветом, а график функции $z=\cos(t^2)$ – в окне с именем **WINDOW2** зеленым. В обоих окнах вывести линии сетки.

На листинге 4.35 приведено решение этой задачи с подробными комментариями.

```
t=-5:0.1:5;
x=exp(cos(t));
y=exp(sin(t));
z=cos(t.^2);
% Создаем первое графическое окно, указатель, на которое будет
%храниться в переменной hfig1.
hfig1=figure;
% Создаем второе графическое окно, указатель, на которое будет
% храниться в переменной hfig2.
hfig2=figure;
% Объявляем первое графическое окно текущим.
figure(hfig1);
% Выводим в нем оси, указатель на на которые будет храниться в
% переменной hAxes1.
hAxes1=axes;
% Выводим в этом окне график функций x(t) и y(t), указатель на
% который записываем в переменную h_gr1. В h_gr(1) будет
% храниться первая линия - x(t), в h_gr(2) будет храниться
% вторая линия - y(t).
h_gr1=plot(t,x,t,y);
% Объявляем второе графическое окно текущим.
figure(hfig2);
% Выводим в нем оси, указатель на на которые будет храниться в
% переменной hAxes2.
hAxes2=axes;
% Выводим в этом окне график функции z(t), уазатель на который
% записываем в переменную h_gr2.
```

```
h_gr2=plot(t,z);
% Объявляем первое графическое окно текущим
figure(hfig1);
% Далее устанавливаем свойства осей и графиков в первом окне.
% Отказываемся от стандартной нумерации окон для первого окна.
set(hfig1,'numbertitle','off');
% Устанавливаем новое имя первого графического окна.
set(hfig1,'name','WINDOW1');
% Включаем отображение линий сетки, перпендикулярной оси OX
% для осей hAxes1.
set(hAxes1,'xgrid','on');
% Включаем отображение линий сетки, перпендикулярной оси OY
% для осе. hAxes1.
set(hAxes1,'ygrid','on');
% Устанавливаем красный цвет первой линии в первом графическом
% окне.
set(h_gr1(1),'color','r');
% Устанавливаем синий цвет второй линии в первом графическом
% окне.
set(h_gr1(2),'color','b');
% Объявляем второе графическое окно текущим
figure(hfig2);
% Далее устанавливаем свойства осей и графиков во втором окне.
% Отказываемся от стандартной нумерации окон для второго окна.
set(hfig2,'numbertitle','off');
% Устанавливаем новое имя второго графического окна.
set(hfig2,'name','WINDOW2');
% Включаем отображение линий сетки, перпендикулярной оси OX
% для осей hAxes2.
set(hAxes2,'xgrid','on');
% Включаем отображение линий сетки, перпендикулярной оси OY
% для осей hAxes2.
set(hAxes2,'ygrid','on');
% Устанавливаем зеленый цвет первой линии во втором
% графическом окне.
set(h_gr2,'color','g');
% Эта функция может быть и такой.
% set(h_gr2(1),'Color','g');
```

Листинг 4.35.

На рис. 4.39 и 4.40 представлены созданные с помощью программы, приведенной в листинге 4.35, окна с графиками функций $x(t)=e^{\cos(t)}, y=e^{\sin(t)}$ и $z=\cos(t^2)$ соответственно.

При программировании работы с графическими окнами, вызов функции `plot` может осуществляться со всеми параметрами, рассмотренными в этой главе. Для того, что бы добавить новый график в текущие оси необходимо перед вызовом функции plot выполнить команду `hold on`. При добавлении графика в текущие оси с помощью функции plot необходимо самостоятельно устанавливать цвет и тип графика.

ПРИМЕР 4.24. Изобразить функций $x_i=\alpha_i e^{\sin(t)}, y_i=\sin(\alpha_i t), z_i=\cos(\alpha_i t), v_i=\sin(2\alpha_i t)+\cos(3\alpha_i t)$ на интервале [-5; 5], если коэффициенты $\alpha_i=0.5, 0.6, 0.73, 0.79$ хранятся в текстовом файле **gr.txt** (см. рис. 4.41).

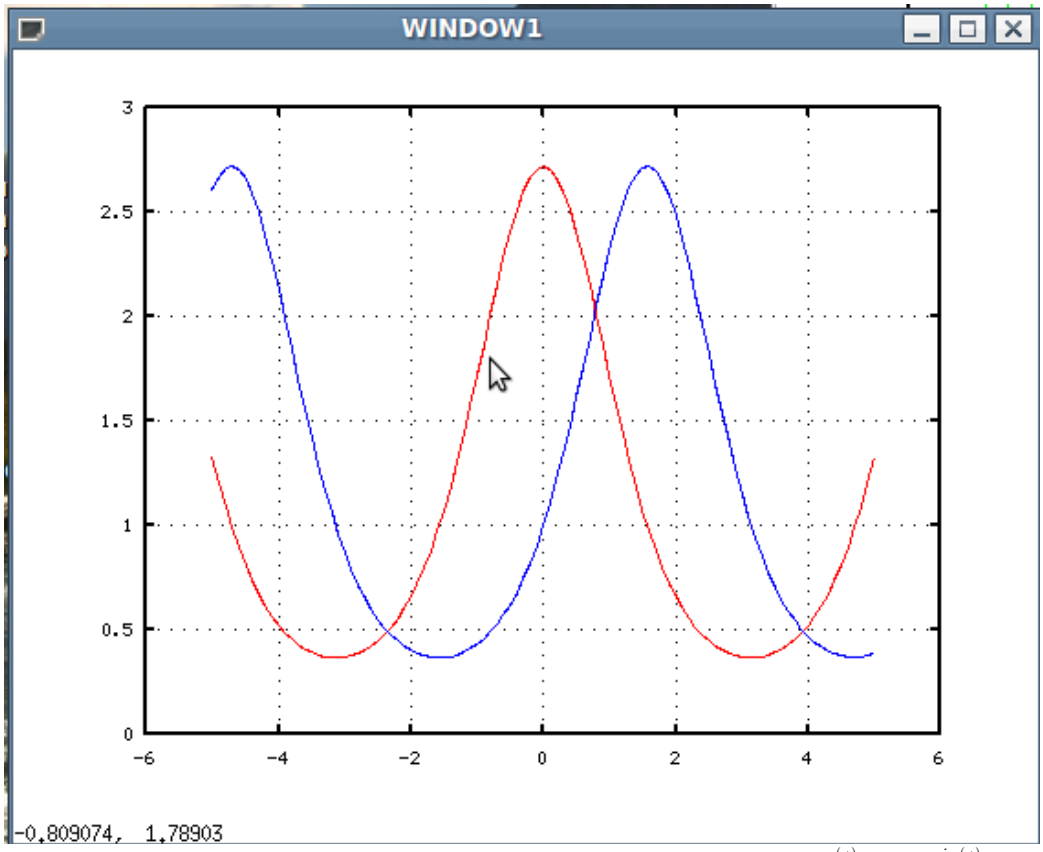


Рисунок *4.39*. Окно с графиками функций $x(t)=e^{\cos(t)}$, $y=e^{\sin(t)}$

Рисунок *4.40*. Окно с графиком функции $z=\cos(t^2)$

При решении этой задачи необходимо будет построить четыре множества графиков x_i, y_i, z_i и v_i. Каждое множество будем изображать в своих осях. На листинге

4.36 приведена программа решения примера 4.24, а на рис. 4.42 представлено полученное в результате графическое окно.

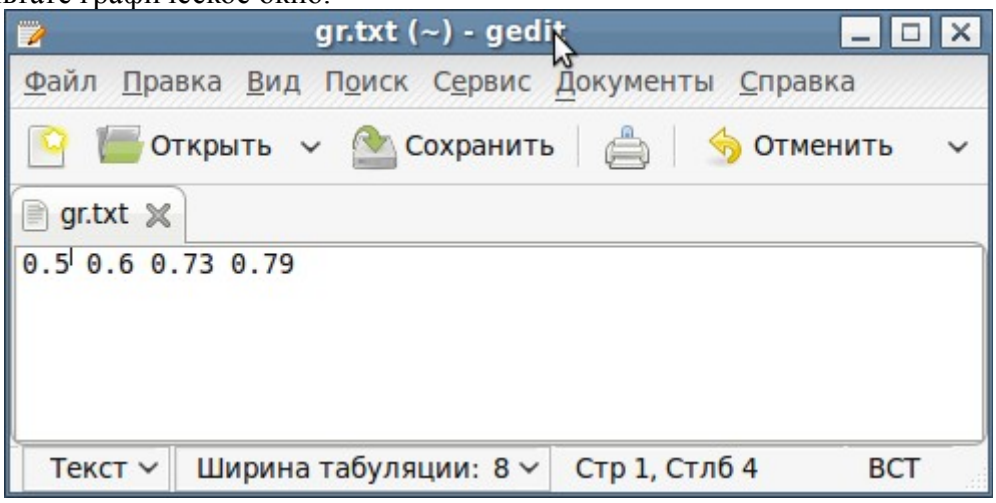


Рисунок *4.41*. Содержимое файла ***gr.txt***

```
% Открываем текстовый файл gr.txt в режиме чтения.
f=fopen('gr.txt','rt');
% Считываем из него данные в массив alf.
alf=fscanf(f,'%f',4);
% Формируем массивы t, x, y, z, v
t=-5:0.1:5;
x=alf*exp(sin(t));
y=sin(alf*t);
z=cos(alf*t);
v=sin(2*alf*t)+cos(3*alf*t);
% Создаем графическое окно
hfig1=figure;
% Устанавливаем новое имя первого графического окна.
set(hfig1,'numbertitle','off');
set(hfig1,'name','Plots');
% Выводим в нем оси, указатель на которые будет храниться в
% переменной haxes1. Оси будут располагаться в левом нижнем
% углу графического окна.
haxes1=axes('position',[0.05 0.05 0.4 0.4]);
% Выводим множество графиков xi(t)
plot(t,x);
% Выводим линии сетки на осях.
set(haxes1,'xgrid','on','ygrid','on');
% Выводим в графическом окне оси, указатель на которые будет
% храниться в переменной haxes2. Оси будут располагаться в
% правом нижнем углу графического окна.
haxes2=axes('position',[0.5 0.05 0.4 0.4]);
% Выводим множество графиков yi(t).
plot(t,y);
% Выводим линии сетки на осях.
set(haxes2,'xgrid','on','ygrid','on');
% Выводим в графическом окне оси, указатель на которые будет
% храниться в переменной haxes3. Оси будут располагаться в
% левом верхнем углу графического окна.
haxes3=axes('position',[0.05 0.5 0.4 0.4]);
```

```
% Выводим множество графиков zi(t)
plot(t,z);
% Выводим линии сетки на осях.
set(haxes3,'xgrid','on','ygrid','on');
% Выводим в графическом окне оси, указатель на которые будет
% храниться в переменной haxes3. Оси будут располагаться в
% правом верхнем углу графического окна.
haxes4=axes('position',[0.5 0.5 0.4 0.4]);
% Выводим множество графиков zi(t).
plot(t,v);
% Выводим линии сетки на осях.
set(haxes4,'xgrid','on','ygrid','on');
```

Листинг 4.36.

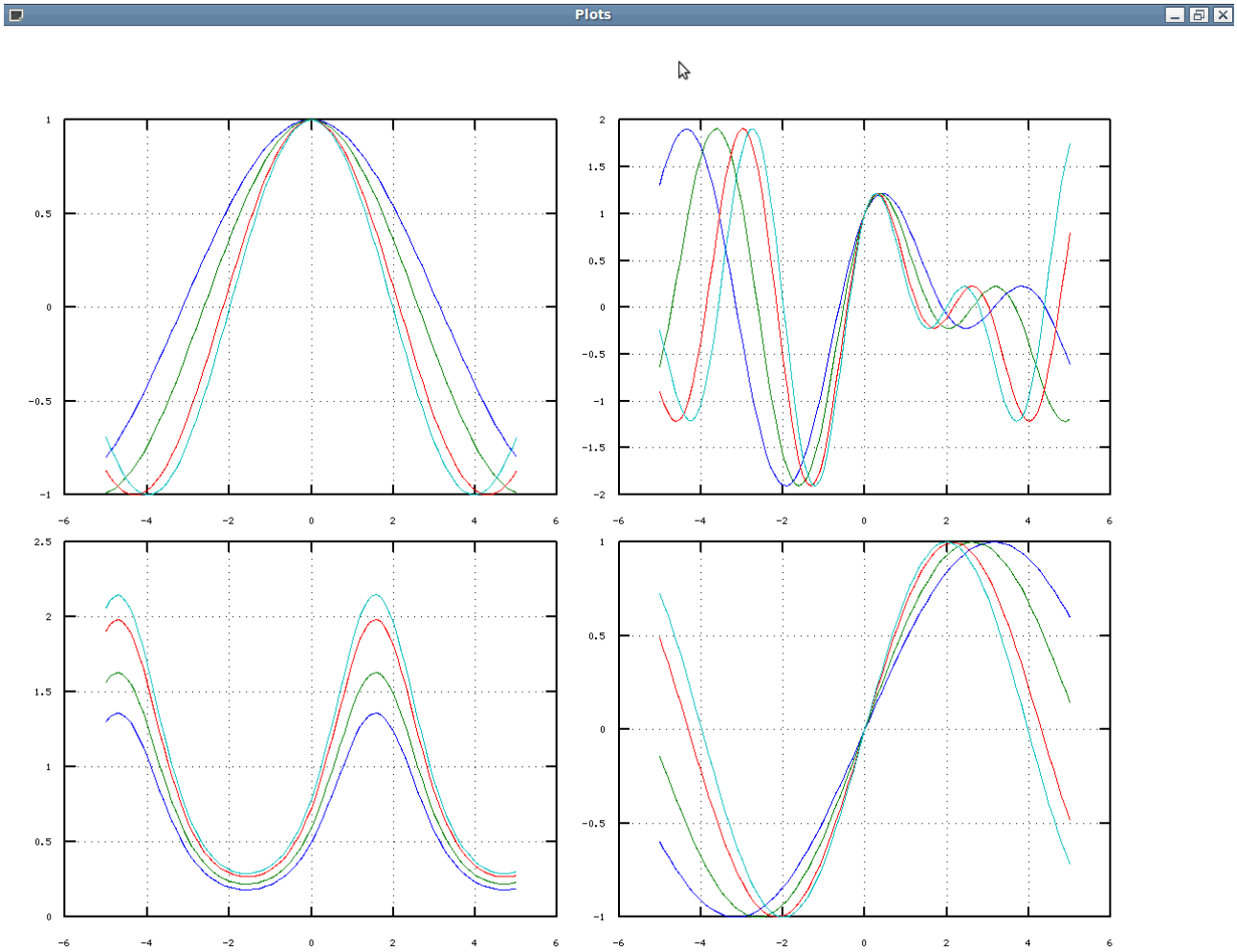


Рисунок *4.42*. Окно с графиками функций

$x_i = \alpha_i e^{\sin(t)}$, $y_i = \sin(\alpha_i t)$, $z_i = \cos(\alpha_i t)$, $v_i = \sin(2\alpha_i t) + \cos(3\alpha_i t)$

На графиках не хватает текстовой информации, которая бы поясняла выведенные графики.

Для вывода текста можно использовать следующие функции:

- `title('Заголовок')` предназначена для вывода заголовка графика;
- `xlabel('Подпись')` служит для вывода текста под осью **OX**;
- `ylabel('Подпись')` предназначена для вывода названия оси **OY**;
- `text(x,y,'Text')` служит для вывода текста в точке с координатами `(x,y)`; координаты `(x,y)` задаются в системе координат графика, нет необходимости

пересчитывать их в "экранную" систему координат.

На листинге 4.37 представлена программа построения графика $x=\sin(t)$ на интервале $[-2\pi;2\pi]$ вместе с заголовками, подписями осей и примером использования функции text. Полученный в результате работы программы график представлен на рис. 5.9.

```
t=-2*pi:pi/50:2*pi;
x=sin(t);
plot(t,x);
xlabel('t');
ylabel('x');
title('Plot function x=sin(t)');
text(-1,-0.8,'<- Point (-1,-0.8)');
```

Листинг 4.37.

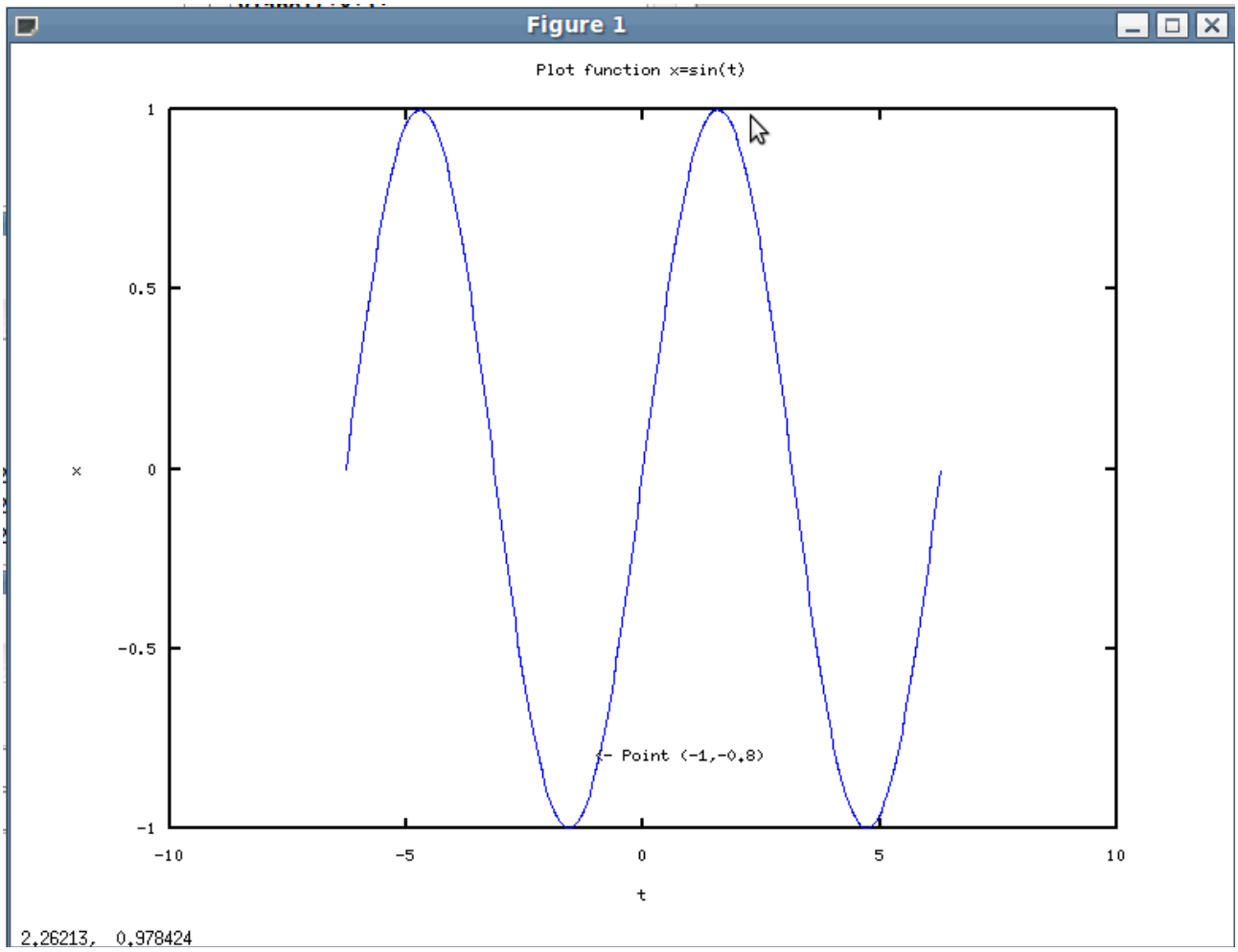


Рисунок *4.43*. График функции *x=sin(t)* с подписями

Все рассмотренные функции вывода текста формирует указатель на созданный текстовый объект, у которого с помощью функции `set` можно установить соответствующие свойства, наиболее часто встречающиеся из которых приведены ниже:

- `color` определяет цвет шрифта;
- `backroundcolor` позволяет определить цвет фона;
- `fontangle` позволяет установить наклон шрифта;
- `fontname` определяет название шрифта;
- `fontsize` определяет размер шрифта в пунктах;
- `fontweight` определяет толщину шрифта;
- `linestyle` позволяет изменять стиль прямоугольной рамки;

• `linewidth` определяет толщину линий прямоугольной рамки.

На этом мы заканчивает краткое знакомство с графическими объектами языка Octave и предлагаем читателю самостоятельно поэкспериментировать с описанными свойствами графических объектов при написании собственных программ.

В четвертой главе мы рассмотрели основные возможности Octave для построения двух- и трёхмерных графиков, а также средства языка программирования для работы с графиками.