МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Брянский государственный технический университет

Утверждаю
Ректор университета
_____А.В. Лагерев
“__”_____ 2011 г.

ИНФОРМАТИКА

НАЗНАЧЕНИЕ ДЕСКРИПТОРНОЙ ГРАФИКИ В MATLAB

Методические указания к выполнению
лабораторной работы № 2
для студентов очной формы обучения
специальностей
140400 – "Электроэнергетика и электроника "
210100 – "Электроника и наноэлектроника"
210400 – "Радиотехника"

БРЯНСК 2011

УДК 519.682(076)

Информатика.Назначение дескрипторной графики в MATLAB [такст] +[электронный ресурс]: методические указания к выполнению лабораторной работы № 1 для студентов очной формы обучения специальностей 180400- «Электропривод и автоматика промышленных установок и технологических комплексов»; 210106-«Промышленная электроника»; 20010-«Микроэлектроника и твердотельная электроника»; 21030-«Радиоэлектронные системы». – Брянск: БГТУ, 2011 – 16 с.

Разработал:
В.В.Симкин
доцент, канд. техн. наук

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол № 7 от 03.2011)

1. ЦЕЛЬ РАБОТЫ

Цель работы: изучить методы управления элементами графиков: удаления поверхности, изменения цвета и толщины линий, добавления стрелки и поясняющих надписей и т.д. Получить навыки использования дескрипторной графики и низкоуровневых графических функций для обеспечения полного контроля над элементами графиков.

Продолжительность работы – 4 часа.

Последовательность выполнения лабораторной работы:

1) изучение теоретических вопросов;
2) выполнение практических заданий;
3) защита лабораторной работы.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Назначение дескрипторной графики

Дескрипторная графика используется для создания собственных приложений. Ее понимание обязательно для эффективного написания приложений с графическим интерфейсом пользователя. Кроме того, большинство высокоуровневых графических функций допускают обращение к ним с использованием низкоуровневых свойств того графического объекта, который они создают, например:

```
x = 0:0.2:10;

y = cos(x);

plot(x, y, 'LineWidth', 2, 'Marker', 'o' , 'MarkerSize', 10)
```

В данном примере при построении графика функцией plot заданы следующие свойства линии:

LineWidth - толщина линии 2пт.;

Marker - тип маркера (кружок);

MarkerSize - размер маркера 10пт.

2.2. Иерархия графических объектов

Зададимся вопросом, что происходит при выполнении следующих команд:

```
x = 0:0.2:10;

y = cos(x);

plot(x, y);
```

Строится график функции, но при рассмотрении дескрипторной графики нам потребуются другой взгляд на этот процесс и соответствующая терминология. Если не было открыто графическое окно, то высокоуровневая графическая функция plot() создала ряд *графических объектов*: сначала графическое окно, затем оси и, наконец, линию. Все графические объекты MATLAB выстроены в определенную иерархию, оси являются *потомком* графического окна и не могут существовать сами по себе. В свою очередь, графическое окно - *предок* для осей. Аналогичным образом дело обстоит с линией. Она является потомком осей, а оси - ее предком. Одновременно может существовать несколько графических окон, каждое из них может содержать и несколько потомков (осей), а каждые оси по нескольку потомков (линий, поверхностей и других графических объектов), например:

```
x = 0:0.1:5;

f = exp(-x).*sin(x);

g = exp(-x).*sin(2*x);

subplot(2, 1, 1);

plot(x, f, x, g);

subplot(2, 1, 2);

mesh(rand(10));
```

Графических объектов достаточно много, их иерархическая структура представлена на рис. 1 для MATLAB версии 7.

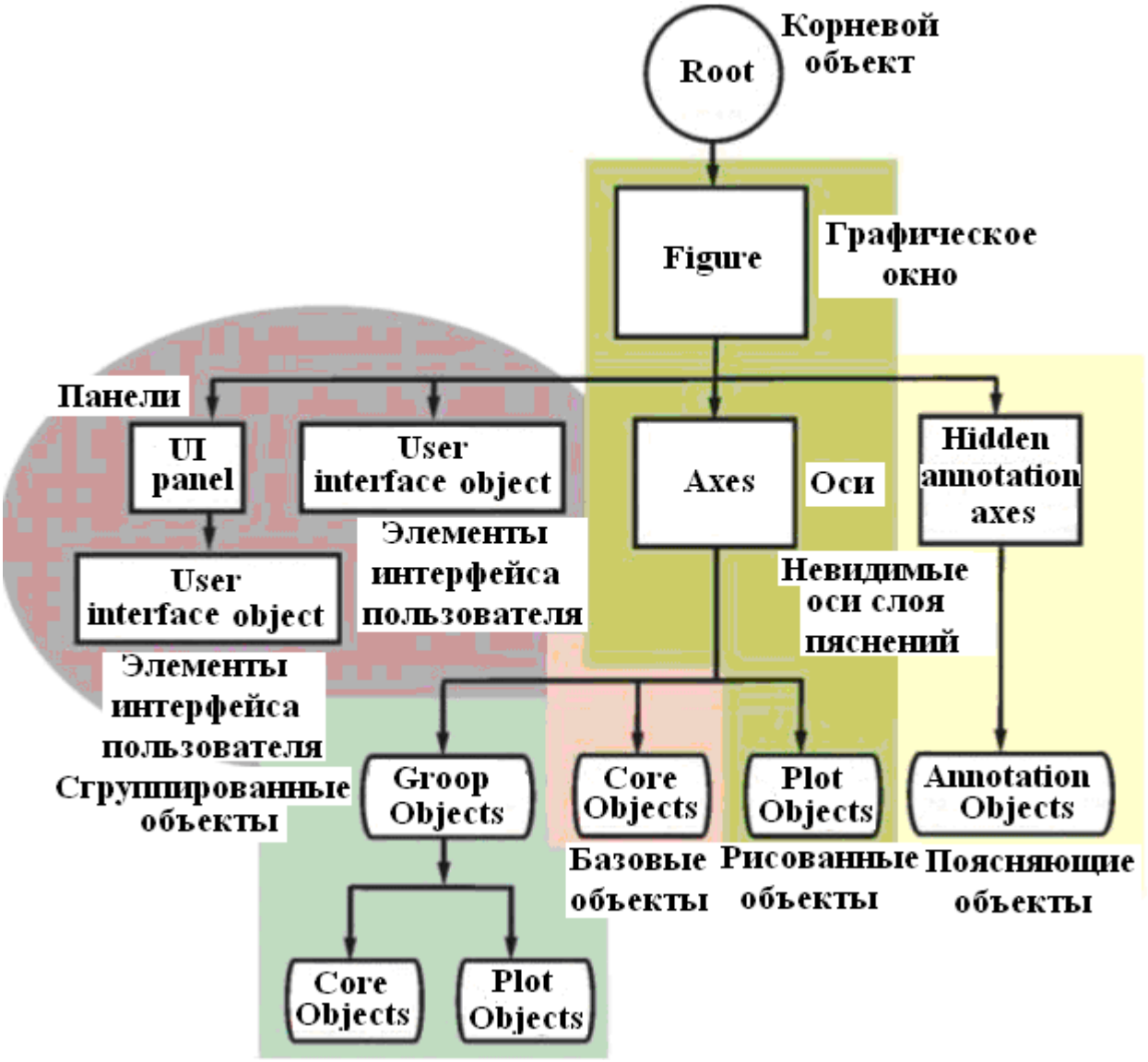

Рис. 1. Иерархическая структура графических объектов

Рассмотрим приемы работы с ними в соответствии с цветовой заливкой на рис. 1. Сначала - оси (Axes), графические окна (Figure) и рисованные объекты (Plot Objects), поскольку, как правило, они создаются в результате работы высокоуровневых графических функций plot, bar, surf и др. Затем мы обратимся к свойствам поясняющих объектов (Annotation Objects), которые являются потомками специальных невидимых осей (Hidden Annotation Axes), служащих для размещения текстовых примечаний, стрелок, текстовых выносок и других объектов. После этого мы займемся базовыми объектами, включающими линии, полигональные объекты и ряд других. Рисованные и ба-

зовые объекты могут быть сгруппированы для удобства работы с ними и выполнения однотипных операций. Этот вопрос мы обсудим при конструировании сгруппированных объектов (Group Objects). Потомки графического окна - панели и элементы пользовательского интерфейса - нужны при создании приложений с графическим интерфейсом пользователя.

2.3. Текущий графический объект и указатели на объекты

Предположим, что открыто несколько графических окон, содержащих одну или несколько пар осей. На какую из них произойдет вывод графика, например, функцией plot? Оказывается, что на *текущие оси*, т.е. в последние созданные или те, которые были сделаны текущими при помощи щелчка мышью по ним. Если ваше приложение осуществляет вывод на разные оси, то необходим способ, который позволяет делать оси (да и любые объекты: графические окна, линии и поверхности) текущими в ходе работы приложения. Для создания графического окна служит функция figure. Вызов ее с выходным аргументом приводит не только к появлению графического окна, но и записи в него *указателя* на созданный объект - графическое окно. Создадим два графических окна, записав указатели на них в переменные hF1 и hF2:

```
hF1 = figure;

hF2 = figure;
```

Теперь для того, чтобы в любом месте приложения сделать текущим графическое окно с указателем hF1, достаточно обратиться к функции figure со входным аргументом - указателем на окно:

```
figure(hF1);
```

При этом графическое окно не только станет текущим, но и расположится поверх остальных окон. Предположим, что в некотором месте алгоритма требуется вывести графики в первое графическое окно с указателем h1. Создадим в первом графическом окне оси при помощи функции axes, запомнив указатель на оси в переменной hA1 (вызов большинства функций высокоуровневой графики с выходным

аргументом обеспечивает запись в него указателя на созданный графический объект):

hA1 = axes;

Построим теперь на этих осях графики двух функций красным и зеленым цветами, обратившись к plot() с выходным аргументом, в который будет записано значение указателя на линию:

x = -3:0.1:3;

f = sin(x.^2);

g = sin(x).^2;

hL1 = plot(x, f, 'r');

hold on;

hL2 = plot(x, g, 'g');

После выполнения приложением некоторых действий может оказаться, что график первой функции не нужен и его следует удалить. Для этого достаточно воспользоваться функцией delete(), которая удаляет графический объект с заданным указателем:

delete(hL1);

График первой функции удалился. Для дальнейшего построения графиков на осях с указателем hA1 в любом месте программы достаточно сделать их текущими

axes(hA1);

Очевидный вывод состоит в том, что очень полезно при создании графического объекта сохранять указатели на него в переменных - появляется возможность манипулировать объектами. Если в нашем примере удалить оси, то удалится и оставшаяся на них линия, поскольку в иерархии объектов линия является потомком осей и не может существовать отдельно от них:

delete(hA1);

Аналогичным образом удаление графического окна вызовет исчезновение всех объектов, лежащих ниже в иерархии: осей, размещенных в этом окне, и всех принадлежащих им поверхностей и линий. Разумеется, объекты можно не только удалять при помощи delete(), но и копировать, или осуществлять поиск одного или нескольких объектов с нужными свойствами. Мы разобрали, как делать объект текущим, зная указатель на него. Обратная задача - получение указателя на текущий объект - решается с привлечением функции gco (сокращение от get current object):

hCO=gco;

Эта функция позволяет определить, какой объект сделан текущим, скажем, щелчком мыши. Для получения указателя на текущие оси и графическое окно служат две функции: gca (сокращение от get current axes) и gcf (сокращение от get current figure), соответственно. Их можно использовать, например, для установки значений свойствам только что созданных осей или графического окна. Обратимся теперь к основным свойствам графических объектов и их использованию для организации графического вывода.

2.4. Доступ к значениям свойств графических объектов

Для задания значений свойствам графических объектов служит функция set, которая вызывается от указателя на объект и пары 'НазваниеСвойства' - значение:

set(h, 'НазваниеСвойства', значение);

или от нескольких пар для установки значений ряду свойств:

set(h, 'НазваниеСвойства1', значение1, 'НазваниеСвойства2', значение2, :);

Приведем простой пример: требуется установить определенную толщину (5пт) линии графика, построенного в некотором месте программы при помощи функции plot():

hL = plot(x,y);

Знание указателя на линию hL и название соответствующего свойства LineWidth позволяет легко решить эту задачу:

set(hL, 'LineWidth',5) ;

Нередко возникает и обратная задача - получить значение того или иного свойства графического объекта. Для этих целей предназначена функция get, которая вызывается от указателя на интересующий объект и названия свойства. Ее выходным аргументом является значение данного свойства. Предположим, что в предыдущем примере необходимо не просто установить заданную толщину линии, а увеличить ее на 2пт. Для этого выясняется текущая толщина, увеличивается на 2пт и задается в качестве нового значения свойства LineWidth:

W = get(hL, 'LineWidth');

set(hL, 'LineWidth', W+2);

Графические объекты обладают достаточно большими наборами свойств, описание свойств всех графических объектов будет размещаться в разделе "Справочник свойств графических объектов". Обсудим использование ряда свойств на примерах.

2.5. Свойства осей

Предположим, приложение должно разместить в графическом окне три пары осей так, как показано на рис. 2. Создадим графическое окно, сохранив указатель на него в переменной hF:

hF = figure;

Затем первую пару осей

hA1 = axes;

Расположим оси с указателем hA1 вверху окна, прибегнув к их свойству Position. Его значением является вектор из четырех чисел [x y width height], где

x - абсцисса левого нижнего угла осей;

y - ордината левого нижнего угла осей;

width - ширина осей;

height - высота осей.

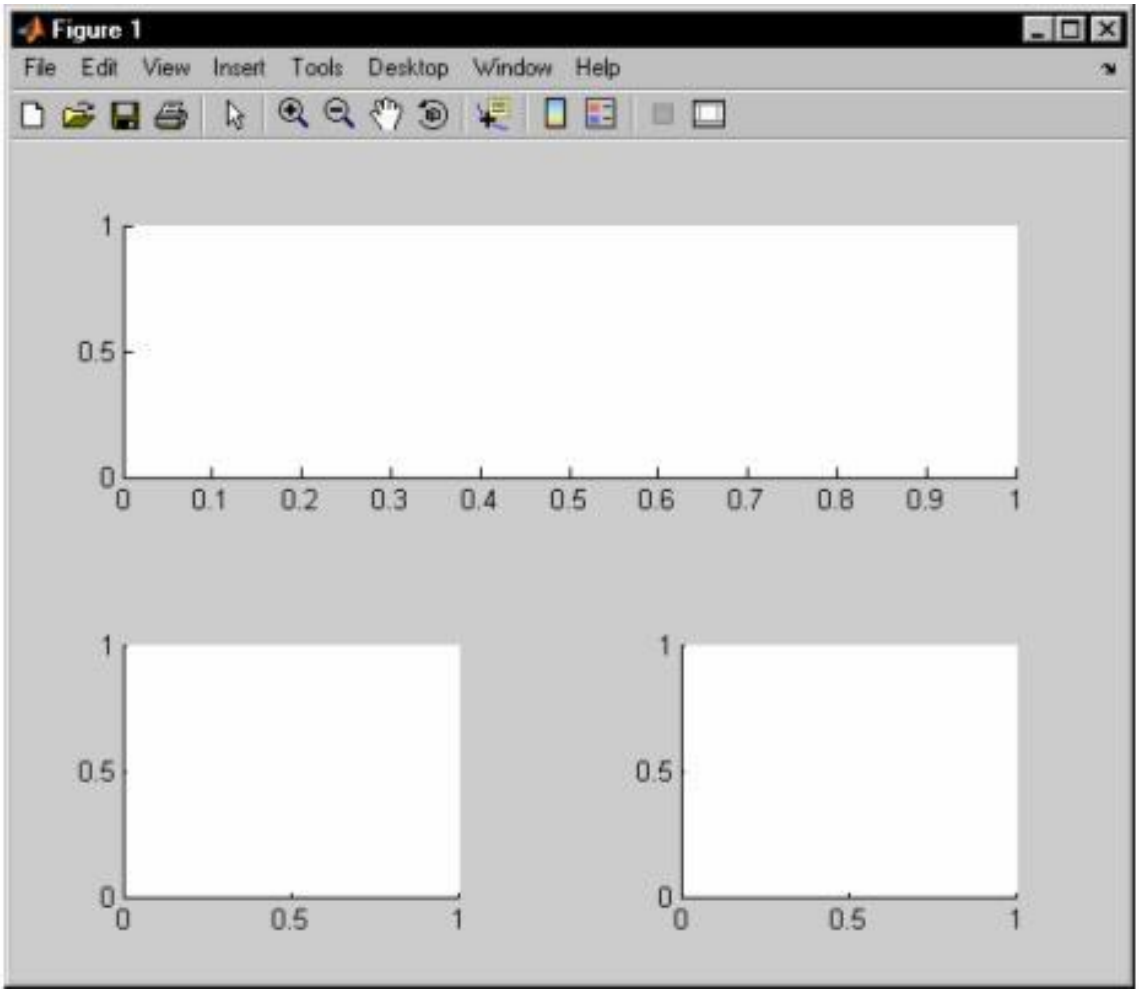

Рис. 2. Графическое окно с тремя парами осей

Эти величины задаются в системе координат графического окна с началом в его левом нижнем углу. Единицы измерений по умолчанию являются нормализованными, т. е. как высота, так и ширина графического окна полагаются равными единице (можно выбрать и другие единицы измерений). Изменим теперь значение свойства Position осей с указателем hA:

set(hA1, 'Position', [0.1 0.6 0.8 0.3]);

Аналогичным образом создадим еще две пары осей и установим их свойство Position в подходящие значения:

hA2 = axes;

set(hA2, 'Position', [0.1 0.1 0.3 0.3]);

hA3 = axes;

set(hA3, 'Position', [0.6 0.1 0.3 0.3]);

В результате получаем графическое окно, приведенное на рис. 2, и указатели hF, hA1, hA2, hA3 на все созданные графические объекты. Эти указатели следует использовать перед выводом на оси, делая нужную пару осей текущей, например:

axes(hA2);

plot(x, y);

MATLAB допускает достаточно гибкое управление положением осей. Так, привлечение свойства OuterPosition позволяет избежать выхода заголовка и подписей к осям за пределы графического окна. Его значением является вектор из четырех элементов [x y width height], смысл которых такой же, как и у свойства Position. Отличие поясняет рис. 3.

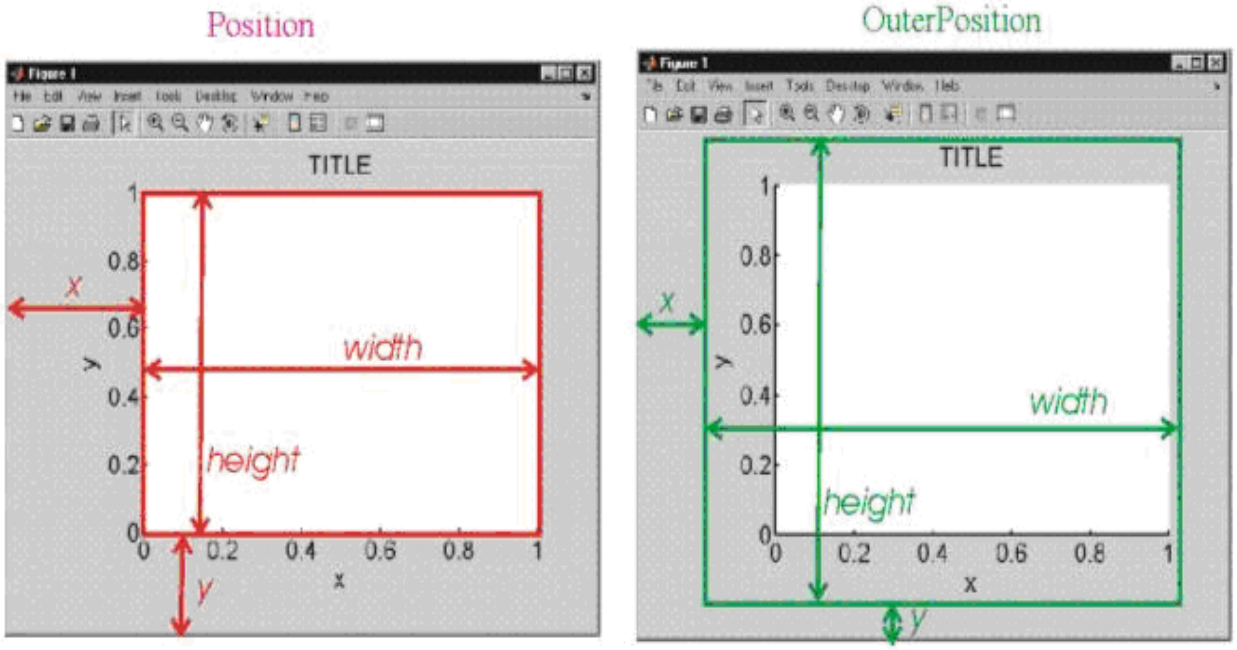

Рис .3. Отличие Position от OuterPosition

Задание пределов трехмерных осей ничем не отличается от случая двумерных, координаты нижнего левого угла, ширина и высота указываются для прямоугольника, ограничивающего их проекцию на графическое окно. При изменении размеров графического окна размеры осей изменяются автоматически. Способ изменения размеров осей может быть основан либо на значении свойства Position, либо OuterPosition - в зависимости от значения свойства ActivePositionProperty: 'outerposition' (по умолчанию) либо 'position'. Для быстрого набора приведенных команд в командной строке MATLAB 7 можно использовать клавишу Tab. Например, если напечатано set(hA,'P') и нажата клавиша Tab, появляется всплывающая подсказка с одним из возможных вариантов продолжения команды, в которой стрелками или мышью выбирается нужная строка. Масштабы осей несколько отличаются, но иногда удобно задать равные масштабы (например, при рисовании круга). За соотношение масштабов осей отвечают свойство DataAspectRatio и сопутствующее ему DataAspectRatioMode. Если значением DataAspectRatioMode является 'auto', то соотношение выбирается автоматически так, чтобы оси занимали бóльшую площадь. Установка свойства DataAspectRatioMode в 'manual' означает, что соотношение масштабов будет определяться значением DataAspectRatio. Требуемое соотношение масштабов по осям x, y и z определяется вектором из трех элементов, который задается в качестве значения свойства DataAspectRatio (при этом свойство DataAspectRatioMode автоматически устанавливается в 'manual'). При желании, можно вернуться к автоматическому выбору масштаба, установив DataAspectRatioMode в 'auto'. Мы обсудили сейчас характерную ситуацию для тех свойств, которые подбираются автоматически для получения лучшего вида графика с учетом отображаемых данных. Эти свойства снабжены сопутствующими данными, их имена оканчиваются словом Mode. Таких свойств достаточно много, они отвечают за управление камерой для обзора трехмерных объектов, сетку, разметку координатных осей, их пределы. Для задания соотношений размеров осей предназначены свойства PlotBoxAspectRatio и PlotBoxAspectRatioMode. Значением свойства PlotBoxAspectRatio должен быть вектор из трех элементов с относительными размерами осей, а PlotBoxAspectRatioMode принимает значение 'auto' либо 'manual'. Очевидно, что не всегда значения свойств Position (или

OuterPosition), DataAspectRatio, PlotBoxAspectRatio и пределы осей будут согласованы. В разделе "Справочник свойств графических объектов" в подразделе "Свойства осей" приведено взаимное согласование этих и других свойств, отвечающих за размеры и положение осей. Кроме того, если ширина и высота, указанные в Position или OuterPosition, не согласуются с PlotBoxAspectRatio, то оси занимают максимальную площадь в выделенном для них прямоугольнике с учетом заданных относительных размеров. Названия ряда свойств осей начинаются с одной из букв X, Y, Z. Эти свойства служат для задания вида каждой оси по отдельности. Наличие сетки на осях определяется значениями свойств XGrid, YGrid и ZGrid. Эти свойства могут принимать только два значения: 'on' или 'off' (установлено по умолчанию). Отобразим в нашем примере (см. рис. 2) на верхних осях линии сетки, перпендикулярные оси *y*:

set(hA1, 'YGrid', 'on');

Для отображения линий сетки по обеим направлениям в каждой из нижних пар осей последовательно вызваем set:

set(hA2, 'XGrid', 'on', 'YGrid', 'on');

set(hA3, 'XGrid', 'on', 'YGrid', 'on');

Можно было бы воспользоваться тем, что допускается задание вектора указателей на объекты в качестве первого входного аргумента функции set, и применить только одно обращение:

set([hA2 hA3], 'XGrid', 'on', 'YGrid', 'on');

Линии сетки совпадают с координатами разметки соответствующей оси, которая выбирается автоматически. Для задания координат разметки осей *x*, *y* и *z* служат соответственно свойства XTick, YTick и ZTick, значениями которых является вектор возрастающих значений координат или пустой массив [ ], если требуется скрыть разметку:

set(hA1, 'XTick', 0:0.05:1);

set(hA1, 'YTick', []);

Как только координаты разметки оси *x* заданы, автоматический режим их выбора выключается. При этом свойство XTickMode, сопутствующее XTick, принимает значение 'manual'. Для перехода к автоматической разметки осей, следует установить свойству XTickMode значение 'auto' при помощи функции set. Аналогичные свойства YTickMode и ZTickMode связаны со свойствами YTick и ZTick. Числовые подписи к разметке можно заменить на текстовые, задействуя свойства XTickLabel, YTickLabel, ZTickLabel, XTickLabelMode, YTickLabelMode и ZTickLabelMode. Линии сетки не обязательно должны отображаться штриховыми линиями. Тип линии зависит от значения свойства GridLineStyle: '-' (сплошная), '--' (штриховая), ':' (пунктирная), '-.' (штрихпунктирная) или 'none' (отсутствие линий). Кроме основной сетки, можно нанести вспомогательную, например так, как на рис. 4.

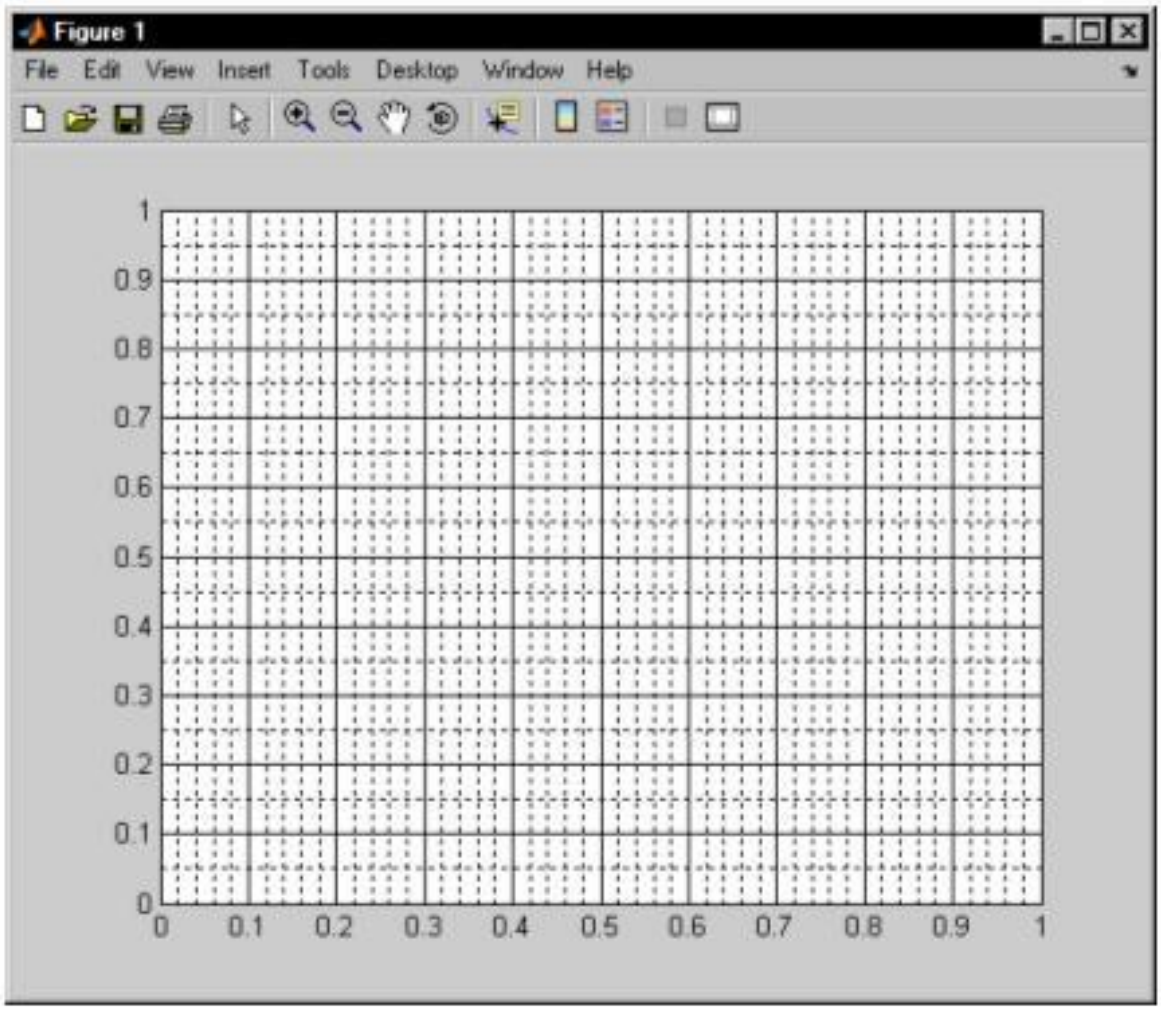

Рис. 4. Основная и вспомогательная сетки

Ее вид определяется свойствами, названия которых отличаются словом Minor от названия соответствующих свойств основной сетки:

```
hF = figure;

hA = axes;

set(hA, 'XGrid', 'on', 'YGrid', 'on', 'GridLineStyle', '-');

set(hA, 'XMinorGrid', 'on', 'YMinorGrid', 'on');
```

Пределы осей выбираются автоматически в зависимости от границ значений визуализируемых данных, если соответствующее свойство XLimMode, YLimMode, или ZLimMode имеет значение 'auto', либо задаются свойствами XLim, YLim, ZLim. Значениями этих свойств должен быть вектор из двух элементов с пределами соответствующей оси. Выше было замечено, что расположение осей определяется рядом свойств, включая их относительные масштабы, относительные размеры и пределы. Для понимания этого достаточно выполнить следующие команды и следить за состоянием осей:

```
t = 0:0.01:2*pi;

x = sin(t);

y = cos(t);

hA = axes;

plot(x, y);

set(hA, 'DataAspectRatio', [1 1 1]);

set(hA, 'PlotBoxAspectRatio', [1 2 1]);

set(hA, 'XLim', [-0.8 0.8]);

set(hA, 'YLim', [-0.9 0.9]);
```

Бывают случаи, когда требуется осуществить вывод, например поверхности, в графическое окно и не отображать оси так, как на рис. 5.

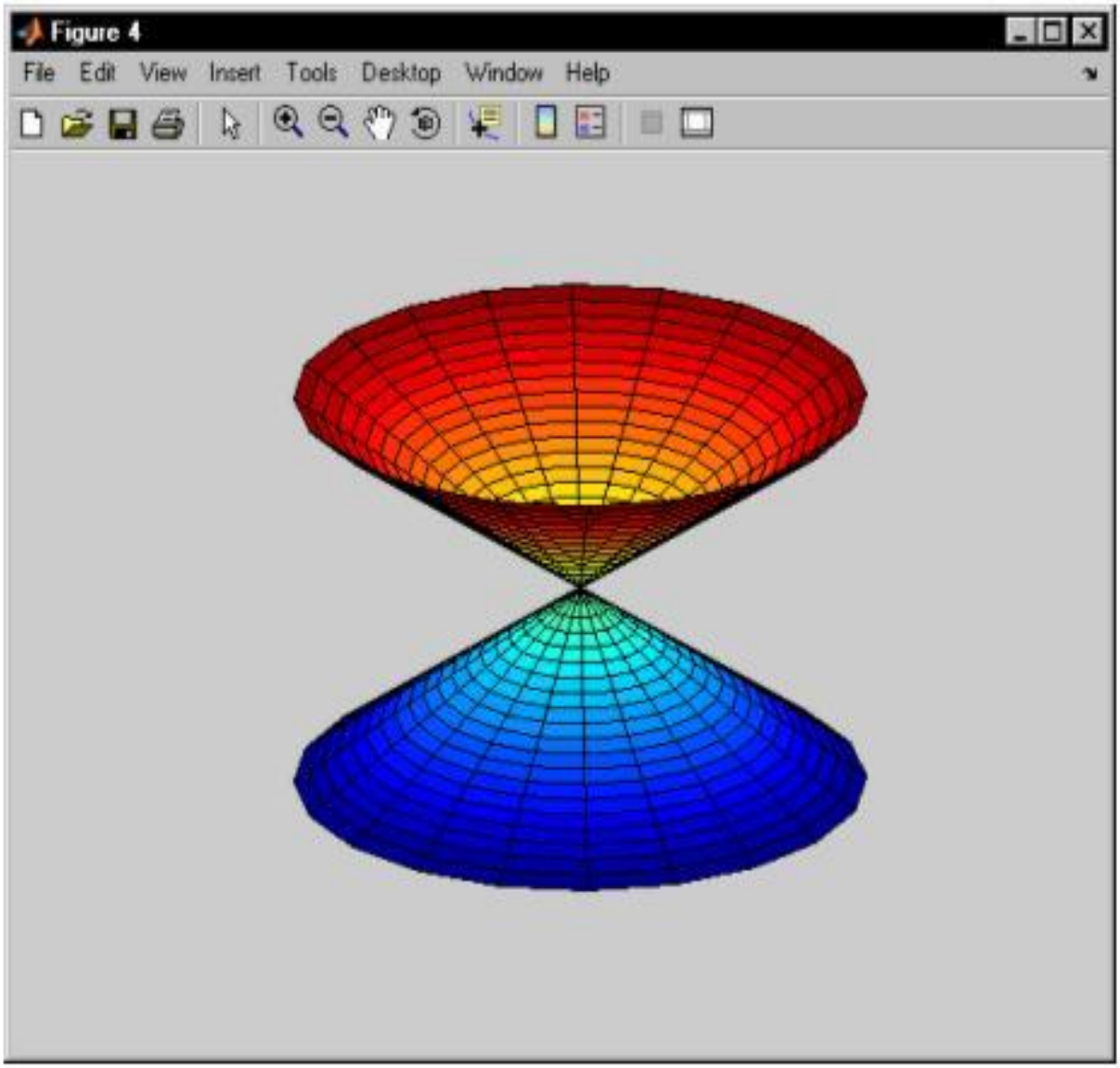

Рис. 5. Использование невидимых осей для вывода в графическое окно

Согласно иерархии объектов, поверхности являются потомками осей и не могут существовать без них. Но оси можно сделать невидимыми, установив их свойство Visible в значение 'off':

```
u = (-2*pi:0.1*pi:2*pi)';

v = -2*pi:0.1*pi:2*pi;

X = 0.3*u*cos(v);

Y = 0.3*u*sin(v);

Z = 0.3*u*ones(size(v));
```

hF = figure;

hA = axes;

surf(X, Y, Z);

set(hA, 'Visible', 'off');

Одновременное задание размеров невидимых осей, совпадающих с размерами графического окна, дает возможность выполнять вывод в любое место графического окна. Если требуется создать оси определенного вида, то пары 'Название Свойства' - значение можно указывать в качестве входных аргументов функции axes, но указатель на оси лучше сохранить - он может понадобиться впоследствии для изменения значений свойств осей:

hA = axes('XGrid', 'on', 'GridLineStyle', '-', 'XMinorGrid', 'on');

Сгруппированные по назначению свойства осей приведены в разделе "Справочник свойств графических объектов". Они предоставляют полный доступ к свойствам осей и позволяют изменять их вид по своему усмотрению (рис 6).

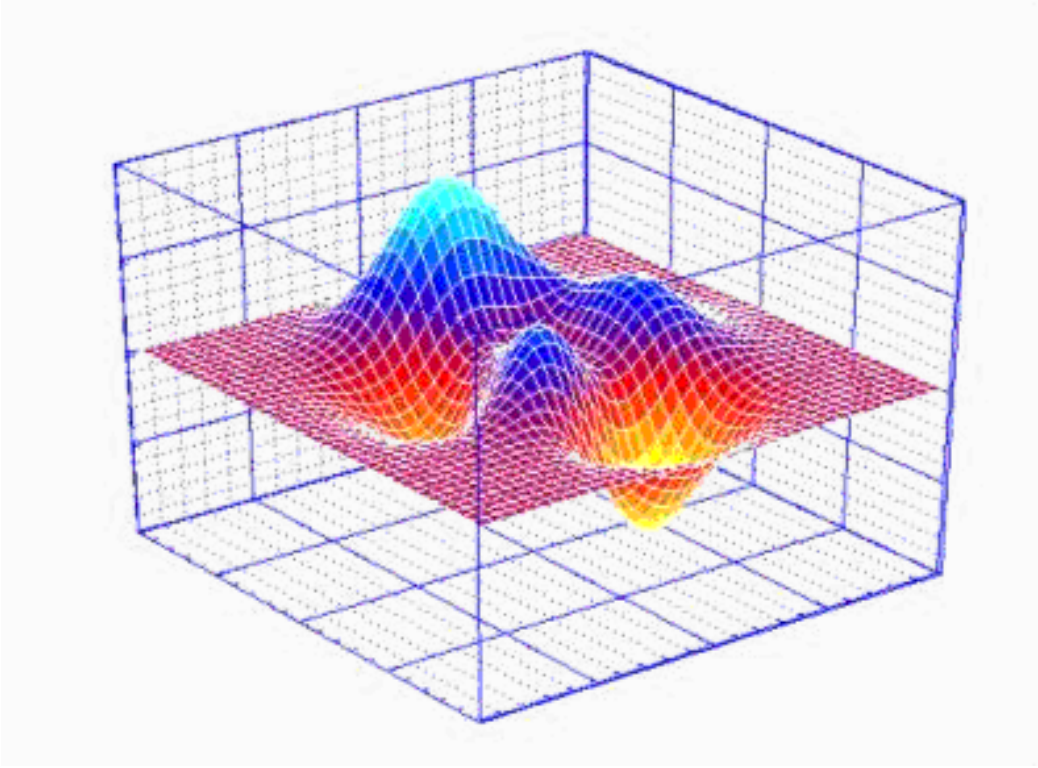

Рис. 6. Пример изменения свойств осей

3. ВЫПОЛНЕНИЕ РАБОТЫ

1. Получите у преподавателя задание на построение графика функции.

2.Постройте график функции, используя функцию plot(x, y, 'LineWidth', 2, 'Marker', 'o' , 'MarkerSize', 10).

3.Постройте график функции, используя функцию subplot(2, 1, 2).

4.Создайте два графических окна, записав указатели на них в переменные hF1 и hF2.

5.Сделайте текущими графические окна с указателями hF1 и hF2

6. Выведите график функции в первое и второе графические окна с указателями на оси в переменной hA1 и hA2.

7. Постройте теперь на этих осях графики функции красным и зеленым цветами, обратившись к функции plot() с выходным аргументом.

8. Удалите графический объект с первым указателем.

9. Восстановите графический объект с первым указателем.

10. Получите указатели на текущий объект, используя функции gco (сокращение от get current object).

11. Получите указатели на текущие оси и графические окна, используя две функции: gca (сокращение от get current axes) и gcf (сокращение от get current figure).

12. Используя функцию set(), установите толщину линий первого графика 5 пт, а второго графика 8 пт.

13. Используя функцию get(), получите значение толщины первого графического объекта и цвета второго графического объекта.

14.Отобразите линии сетки для первого графика красным, а для второго - черным цветом, используя только одно обращение.

15.Сделайте оси графиков невидимыми.

4. ВАРИАНТЫ ЗАДАНИЙ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Построить графики функций, используя дескрипторную графику. Вариант задания выдается преподавателем.

1 вариант

$$a = -1{,}25;\ c = 0{,}05;\ d = 2{,}5;\ i = 5;\ x = \pm 1{,}35$$

$$y = \frac{\sqrt{|c-d| + (a+c)^2}}{\sin 2i} + 10^{-3}\, e^{ix} - \frac{|c-d| + a^2}{\sqrt[3]{(a+c)^2}}$$

2 вариант

$$k = 2;\ x = \pm 2{,}5;\ c = 0{,}31;\ a = 0{,}93;\ b = 5{,}61$$

$$y = \frac{\ln|kx|}{\sin 7} - \sqrt{|x - a^2|} - \frac{10^4 a - b}{\cos kx} + \sqrt[3]{x - a^2} + c^3 x$$

3 вариант

$$k = -2;\ a = 3{,}5;\ b = 0{,}35;\ x = \pm 1{,}523$$

$$y = 10^4 \frac{ax}{b^2} - \left|\frac{a-b}{kx}\right| + \frac{\ln 3}{\sqrt[3]{ax + b^2}} - e^{-kx}$$

4 вариант

$$a = 3{,}5;\ b = 0{,}8;\ k = -2{,}3;\ x = \pm 2{,}75$$

$$y = \frac{1}{7} - \cos\left(\sqrt{x^2 + b} + k\right) + \frac{e^{\frac{k}{x}} + \frac{a}{b}}{\sqrt[3]{308 + k}} + \frac{|a - b|}{\operatorname{tg}\frac{k}{a}}$$

5 вариант

$$a = 10;\ b = 5{,}43;\ c = 0{,}26;\ x = \pm 0{,}55$$

$$y = \frac{cx^2 + (abc)^3}{\cos cx} + \sqrt[4]{\frac{c+1}{x+b}} + \left|e^{cx-a}\right|$$

6 вариант

$$a = 1{,}2;\ k = 0{,}5;\ b = 0{,}1;\ x = \pm 4{,}75$$

$$y = \sqrt[3]{(a^2 + x)x^2} - \frac{1}{\sqrt{\ln(b + x)}} + \sin\left(k + \frac{x^3}{a}\right)$$

7 вариант

$$a = 3{,}27;\ b = 0{,}89;\ i = 0{,}5;\ x = \pm 1.5$$

$$y = \frac{\sqrt{17x}}{a\,e^{bx}} - \left(\frac{xi}{9}\right)^5 e^{a+b} + \operatorname{tg} i\,\frac{\ln(a+b)}{ix^2}$$

8 вариант

$$a = 7{,}83;\ b = 3{,}25;\ k = 1{,}5;\ x = \pm 1$$

$$y = \left|\frac{\sin k^2 x}{a^2 + 3b^2}\right| - \sqrt[5]{b + kx} + \frac{a\left(a^2 - b\right)}{\mathrm{e}^{2x+b}}$$

9 вариант

$$k = -2;\ a = 3{,}5;\ b = 0{,}35;\ x = \pm 1{,}523$$

$$y = 10^4 \frac{ax}{b^2} - \left|\frac{a-b}{kx}\right| + \frac{\ln 3}{\sqrt[3]{ax + b^2}} - \mathrm{e}^{-kx}$$

10 вариант

$$a = 1{,}7;\ b = -1{,}25;\ c = -0{,}3;\ x = \pm 2{,}5;\ k = 3$$

$$y = \sqrt{\frac{abc}{2{,}4}} - \frac{0{,}7abc}{\sin 7} + 10^4 \sqrt[5]{\left|\cos kx\right|} - \frac{\left|b - a\right|}{kx}$$

5. СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Мартынов,Н.Н. MATLAB 5.x. Вычисления, визуализация, программирование/Н.Н. Мартынов, А. П. Иванов – М.:КУДИЦ-ОБРАЗ, 2000. - 336 с.
2. Дьяконов, В. П. MATLAB 6 : учеб. курс / В. П. Дьяконов. − СПб. : Питер, 2001. − 592 с.
3. Степанов, А. Н. Информатика : учебник для вузов / А. Н. Степанов. − 4-е изд.-СПб. : Питер, 2005. − 684 с.

Информатика. Назначение дескрипторной графики в MATLAB: методические указания к выполнению лабораторной работы № 2 для студентов очной формы обучения специальностей 180400- «Электропривод и автоматика промышленных установок и технологических комплексов»; 210106-«Промышленная электроника»; 20010-«Микроэлектроника и твердотельная электроника»; 21030-«Радиоэлектронные системы».

СИМКИН ВАСИЛИЙ ВАСИЛЬЕВИЧ

Научный редактор А.К.Буйвал
Редактор издательства Л.И. Афонина
Компьютерный набор В.В.Симкин

Темплан 2011 г., п.127

Подписано в печать __.__.07. Формат Бумага офсетная. Офсетная печать. Усл.печ.л. 1,39 Уч.-изд.л.1,39 Тираж 30 экз. Заказ Бесплатно____

Брянский государственный технический университет.
241035, Брянск, бульвар 50-летия Октября, 7, БГТУ. 58-82-49.
Лаборатория оперативной полиграфии БГТУ, ул. Институтская, 16.