

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ  
И ИНФОРМАТИКИ  
Кафедра компьютерных технологий и систем**

**В. Б. Таранчук**

**ВВЕДЕНИЕ В ГРАФИКУ  
СИСТЕМЫ *MATHEMATICA***

**Учебные материалы для студентов  
факультета прикладной математики и информатики**

**МИНСК  
2017**

УДК 004.92:004.42(075.8)+519.67(075.8)

ББК 32.973-018.3я73-1+22.183.4

Т19

Утверждено на заседании  
кафедры компьютерных технологий и систем  
24 октября 2017 г., протокол № 3

Р е ц е н з е н т  
доктор физико-математических наук *H. H. Гринчик*

**Таранчук, В. Б.**

Т19      Введение в графику системы *Mathematica* : учеб. материалы для студентов фак. прикладной математики и информатики / В. Б. Таранчук. – Минск : БГУ, 2017. – 53 с.

В учебных материалах дан обзор графических функций системы, изложены основные сервисные функции, опции и директивы, используемые при программировании графики на языке Wolfram, размещении и оформлении иллюстраций. Примерами поясняются сервисные средства, возможности и конструкции языка системы *Mathematica*, приёмы выполнения функций и оформления формируемых изображений. В тексте также приведены практические задания, включены вопросы и задачи для самостоятельного изучения и выполнения.

Предназначено для студентов факультета прикладной математики и информатики.

УДК 004.92:004.42(075.8)+519.67(075.8))

ББК 32.973-018.3я73-1+22.183.4

## **ПРЕДИСЛОВИЕ**

В учебных материалах изложена четвертая часть дисциплины специализации «Технологии интерактивной визуализации». Эта дисциплина знакомит студентов с методами и инструментами создания в Wolfram *Mathematica* интерактивных программных модулей с возможностями: символьных вычислений, графического представления математических преобразований и расчётов, визуализации функций и данных, экспорта формируемых динамических изображений.

Основная цель дисциплины – подготовка студентов к практической работе по использованию современных информационных технологий для решения задач обработки и визуализации результатов компьютерного моделирования.

Основными задачами дисциплины являются:

- дать характеристику современного состояния, классификацию систем компьютерной математики;
- ознакомить с основами функционального программирования;
- сформировать практические навыки выполнения символьных вычислений с помощью систем компьютерной алгебры;
- сформировать практические навыки визуализации результатов компьютерного моделирования.

В результате изучения дисциплины студенты должны  
знать:

- основные правила и приёмы работы с системой компьютерной алгебры Wolfram *Mathematica*; правила работы с интерактивной справочной системой;
- основные возможности программирования и отладки блокнотов системы *Mathematica*;
- основные функции работы со списками;
- функции преобразования и упрощения математических выражений;
- основные требования, правила иллюстрирования графиками и диаграммами функциональных зависимостей и табличных данных;  
уметь:

- получать подсказки функций, опций, директив в системе *Mathematica*;
- вводить и редактировать в *Mathematica* тексты с математической нотацией, готовить документы разного назначения, стилей и форматов;
- составлять и форматировать таблицы, создавать и сопровождать базы данных, выполнять импорт, экспорт, обработку и архивирование наборов экспериментальных данных;

- иллюстрировать результаты наблюдений и расчётов графиками и диаграммами, оформлять их, создавать динамические иллюстрации; владеть:
- средствами разработки интерактивных программных компонент системы *Mathematica*;
- навыками создания в *Mathematica* свободно распространяемых программных модулей.

В учебных материалах дан обзор графических функций системы, изложены основные сервисные функции, опции и директивы, используемые при программировании графики на языке Wolfram, размещении и оформлении иллюстраций. Примерами поясняются сервисные средства, возможности и конструкции языка системы *Mathematica*, приёмы выполнения функций и оформления формируемых изображений. В тексте также приведены практические задания, включены вопросы и задачи для самостоятельного изучения и выполнения.

Советы и критические замечания по изданию просьба направлять на [taranchuk@bsu.by](mailto:taranchuk@bsu.by).

## СПИСОК ОСНОВНЫХ СОКРАЩЕНИЙ

БД – база данных

ИТ – информационные технологии

ОС – операционная система

ПО – программное обеспечение

СКА – система компьютерной алгебры

СКМ – система компьютерной математики

NB (notebook) – блокнот, основной документ системы *Mathematica*



## Введение в графику системы *Mathematica*

Таранчук Валерий Борисович

БГУ,

факультет прикладной математики и информатики

Учебные материалы, инструкции и рекомендации пользователям системы компьютерной алгебры *Mathematica*, обучающие примеры и упражнения (оригинал документа создан и предоставляется студентам в формате NB)

### ► Содержание лекций и занятий 1 - 9

Уважаемые читатели. В сгруппированной секции выше (и везде далее в подобных) при работе с оригинальным блокнотом, раскрывая группу секций, вы получаете дополнительные материалы для самостоятельного изучения.

### ▼ Содержание лекции и занятия 10

- ✓ О графических объектах системы *Mathematica*
- ✓ Типы (категории) графических объектов *Mathematica*
- ✓ Контролируемая самостоятельная работа

#### О графических объектах системы *Mathematica*

Графика системы *Mathematica* всегда считалась для многих пакетов и систем эталоном и во многом способствовала ее высокой репутации мирового лидера среди подобных программных приложений. Графические возможности достигаются, как обилием встроенных функций, так и средствами их модификации с помощью директив, опций, примитивов, языка программирования. *Mathematica* позволяет строить любые виды математических графиков, причем обычного пользователя в большинстве случаев удовлетворяют графики, параметры которых задаются по умолчанию. В системе поддерживаются работы практически со всеми форматами графики, их импорт и экспорт.

Концептуально графики в *Mathematica* являются объектами, которые создаются (возвращаются) соответствующими графическими функциями. Они охватывают построение всех типов математических графиков, иллюстраций деловой графики – гистограмм, двумерных и трехмерных секторных и столбчатых, пузырьковых диаграмм, специфических графиков для таких областей, как финансы и статистика, теория графов, управляющие системы. Достигается многообразие за счет применения функций, опций и директив. Графическим объектам можно присваивать имена, а затем оперировать как любыми объектами.

Программирование графики в *Mathematica* относится к функциональному типу. Почти каждый ввод в системе является функцией

(командой), каждая имеет определённый набор атрибутов и может иметь набор опций, директив. Атрибуты важны для эффективной работы системы и могут использоваться при программировании. Пользователи *Mathematica* могут применять опции для управления работой отдельных функций. Опции влияют на результат выполнения функции, различные варианты опций могут дать совершенно разные по виду результаты.

*Mathematica* содержит множество средств для визуализации функций и данных. Система автоматизирует процесс подбора и указания разных деталей построения иллюстраций. Применение опций, число которых очень большое, позволяет оформлять графические образы в любом масштабе, виде, дизайне. У каждой графической функции опция всегда имеет значение по умолчанию, которое определяет результат применения функции, если не указано другое. Несмотря на то, что используемые по умолчанию параметры визуализации тщательно подобраны, чтобы наилучшим образом (с точки зрения разработчиков) подходить для большинства типичных случаев, *Mathematica* обеспечивает пользователей возможностями индивидуальной настройки.

Более того, как уже отмечалось ранее, пользователь может в Out-секции блокнота *Mathematica* изменить размер графического изображения, 3D объекты можно вращать и перемещать. Отдельно будет обстоятельный разговор об интерактивных графических объектах, точнее – об инструментах, позволяющих “оживить”, сделать иллюстрации управляемыми, интерактивными с разными альтернативами не только оформления, но и изменения параметров визуализации.

### Типы (категории) графических объектов *Mathematica*

Используя оригинальные алгоритмы, разработанные в компании, отложены и включены в систему многочисленные функции, которые автоматизируют процесс создания содержательных и эстетически оформленяемых иллюстраций функций, данных (структурированных и неструктурных), причём не только для точек, линий и поверхностей, но и для графов и сетей. Каждое графическое изображение в системе *Mathematica* называется графическим объектом. Условно (к настоящему моменту нет общепринятой терминологии) формируемые системой графические объекты можно разделить на следующие категории:

#### Графики аналитически задаваемых функций одной переменной:

- Plot – график аналитически заданной функции  $f(x)$  (и везде ниже – или нескольких функций), обе шкалы линейные;
- LogPlot –  $\text{LogPlot}[f, \{x, x_{\min}, x_{\max}\}]$  генерирует график аналитически заданной функции  $f(x)$  от  $x_{\min}$  до  $x_{\max}$  в логарифмической шкале (по оси ординат шкала логарифмическая, по оси абсцисс – линейная);

- `LogLinearPlot` – `LogLinearPlot[f, {x, xmin, xmax}]` генерирует график аналитически заданной функции  $f(x)$  от  $x_{min}$  до  $x_{max}$  в логарифмически линейной шкале (по оси ординат шкала линейная, по оси абсцисс логарифмическая);
- `LogLogPlot` – `LogLogPlot[f, {x, xmin, xmax}]` генерирует график аналитически заданной функции  $f(x)$  от  $x_{min}$  до  $x_{max}$  в лог-лог шкале (логарифмический по обеим осям);
- `PolarPlot` – `PolarPlot[r, {\theta, \theta_{min}, \theta_{max}}]` генерирует график кривой радиуса  $r$  в зависимости от угла  $\theta$  в полярной системе координат – положение конца радиус-вектора при изменении угла в заданном диапазоне (полярная диаграмма);
- `ParametricPlot` – график кривой (а также поверхности), заданной параметрически, `ParametricPlot[{fx, fy}, {u, umin, umax}]` генерирует график кривой с  $x$  и  $y$  координатами  $f_x$  и  $f_y$  как функций  $u$ .

Примеры на рисунках 10.1 и 10.2 иллюстрируют графики функций, заданных аналитически (без комментариев так как будет обстоятельное изучение):

```
f1[x_] := Cos[x^2] / E^(0.2 * x^2) + 0.2 * x
f2[x_] := 0.2 * x
f3[x_] := (-E^(-0.2 * x^2)) * Cos[5 * x^2] - 0.1 * x
Plot[{f1[x], f2[x], f3[x]}, {x, -5, 5}]
```

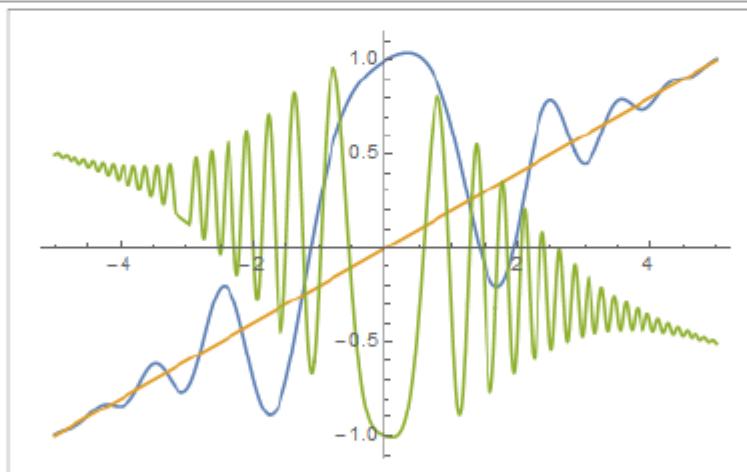


Рис. 10.1. Пример вывода с установками по умолчанию графиков функций, заданных аналитически

Пример выше – коды и результат, как система покажет графики трех функций с установками по умолчанию. А на следующем рисунке 10.2 выводятся графики этих же функций, оформленные пользователем – добавлены заголовок и легенда, назначены тип, цвет и толщины линий кривых графиков, выводится сетка указанного стиля, по другому оформлены вид осей и подписей (код в сгруппированной секции, все шаги оформления рассматриваются позже).



Рис. 10.2. Пример вывода с пользовательскими установками графиков функций, заданных аналитически

#### Тестовые задания для самоконтроля

*Внимание.* Код в большинстве секций не оптимальный, а рабочий и максимально простой, чтобы всем было понятно, что делается. Так как многие работают с конспектами в формате PDF, большинство предназначенных для выполнения секций даются в формате InputForm.

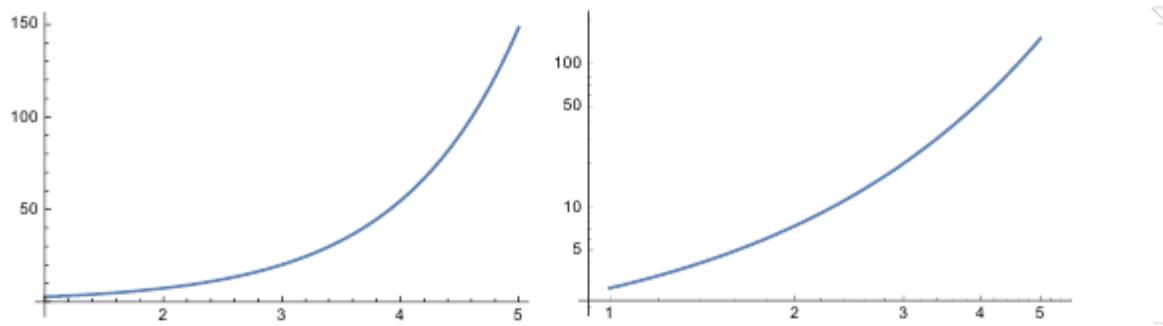
*Подсказки:* Графики в упражнениях ниже выводятся с использованием функции Row, которая обеспечивает (если помещаются) вывод в ряд перечисляемых графических объектов. Разделительный пробел между графическими объектами обеспечивается заданием значения отступа в аргументе Spacer. В примерах ниже графических объектов по 2, один из них можно вывести отдельно и независимо. Например, результатом выполнения исполняемой секции с текстом

`Plot[E^x, {x, 1, 5}, ImageSize -> 270]`

будет график, как в фрагменте слева. Функцию формирования второго надо определить (из числа перечисленных выше).

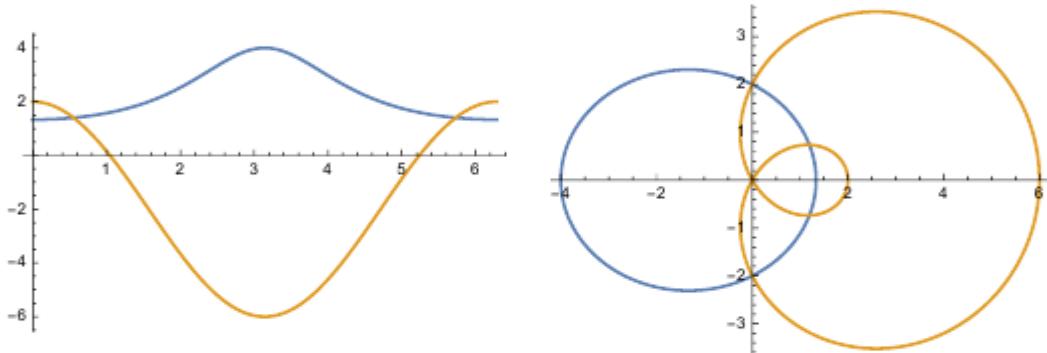
*Упражнение 10.1a.* Что следует вписать вместо Хaaaa для получения графика справа?

```
Row[ {Plot[E^x, {x, 1, 5}, ImageSize -> 270],  
Хaaaa[E^x, {x, 1, 5}, ImageSize -> 270]}, Spacer[5] ]
```



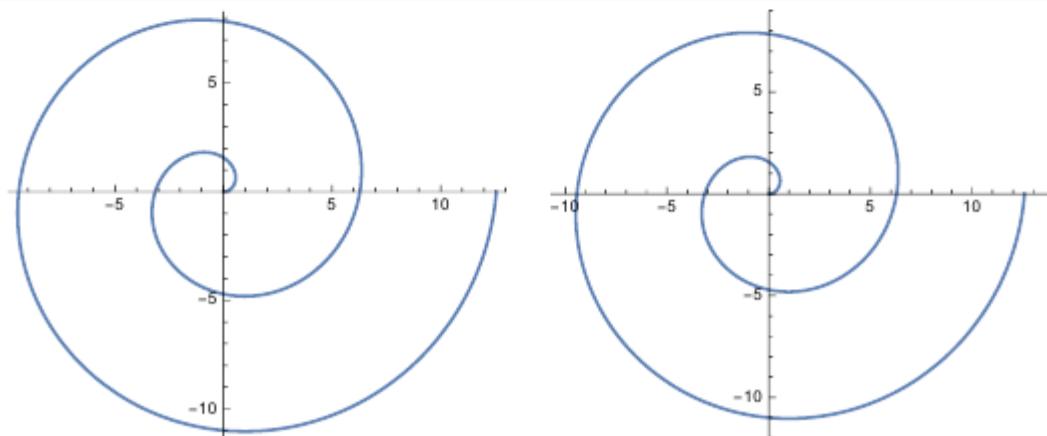
*Упражнение 10.1b.* Что следует вписать вместо Xbbbb для получения графика справа?

```
fPlt1 = {4 / (2 + Cos[t]), 4 * Cos[t] - 2};
g1 = Plot[fPlt1, {t, 0, 2 * Pi}, ImageSize -> 260];
g2 = Xbbbb[fPlt1, {t, 0, 2 * Pi}, ImageSize -> 260];
Row[{g1, g2}, Spacer[15]]
```



*Упражнение 10.1c.* Что следует вписать вместо Xcccc для получения графика справа?

```
Row[{PolarPlot[θ, {θ, 0, 4 * Pi}, ImageSize -> 260],
  Xcccc[{θ * Cos[θ], θ * Sin[θ]}, {θ, 0, 4 * Pi},
  ImageSize -> 260]}, Spacer[15]]
```



**Визуализация массивов одномерных данных:**

Ниже приведены функции визуализации данных по точкам, с одной переменной, по номерам ряда данных (если есть только набор), по координатам (если заданы пары значений):

- `ListPlot` – диаграмма разброса данных; `ListPlot[{ $y_1, y_2, \dots$ }]` выводит график списка величин  $y_1, y_2, \dots$ , координаты которых задаются значениями номеров 1, 2, ...;
- `ListLinePlot` – линейный график по точкам списка данных, `ListLinePlot[{ $y_1, y_2, \dots$ }]` выводит график, на котором соединяет отрезками последовательные точки с координатами (значение, номер);
- `ListStepPlot` – ступенчатый график по списку значений, `ListStepPlot[{ $y_1, y_2, \dots$ }]` строит график, на котором соединяет ступеньками последовательные точки с координатами (значение, номер);
- `ListLogPlot` – диаграмма разброса данных в лог-шкале (по оси ординат шкала логарифмическая, по оси абсцисс линейная);
- `ListLogLinearPlot` – диаграмма разброса данных в лог-линейной шкале (по оси ординат шкала линейная, по оси абсцисс логарифмическая);
- `ListLogLogPlot` – диаграмма разброса данных в лог-лог шкале (логарифмический по обеим осям);
- `ListPolarPlot` – диаграмма разброса данных в полярных координатах;
- `DiscretePlot` – генерирует график последовательности, `DiscretePlot[ $expr, \{n, n_{max}\}$ ]` генерирует график определяемых по формуле  $expr$  значений  $n$  для последовательности от 1 до  $n_{max}$ ;
- `NumberLinePlot` – диаграмма на числовой оси, `NumberLinePlot[{ $v_1, v_2, \dots$ }]` выводит значения  $v_i$  на числовой линии;
- `DateListPlot[{ $\{date_1, v_1\}, \{date_2, v_2\}, \dots$ }`] генерирует точки со значениями  $v_i$  на заданные даты;
- `DateListStepPlot[{ $\{date_1, v_1\}, \{date_2, v_2\}, \dots$ }`] генерирует график ступеньками по точкам со значениями  $v_i$  и заданными датами;
- `DateListLogPlot[{ $\{date_1, v_1\}, \{date_2, v_2\}, \dots$ }`] генерирует график по точкам со значениями  $v_i$  и заданными датами (по оси ординат шкала логарифмическая);
- `QuantilePlot` – квантильный Q-Q график (для сравнения совокупности данных с согласованным нормальным распределением, оценки качества подгонки к наблюдаемым данным);
- `ProbabilityPlot` – график плотности распределения, `ProbabilityPlot[ $list$ ]` генерирует график CDF (cumulative distribution function – функция распределения) случайной величины  $list$ ;
- `ProbabilityScalePlot[{ $x_1, x_2, \dots$ }]` генерирует график плотности распределения (в процентах) по набору  $x_i$ .

Графики на рисунках ниже иллюстрируют табличные данные, а именно: рассчитанные на отрезке  $[-5,5]$  с шагом 0.25 значения функции  $f_3$ . Примеры визуализации данных с использованием функций ListLinePlot и ListStepPlot показаны на рисунках 10.3 и 10.4:

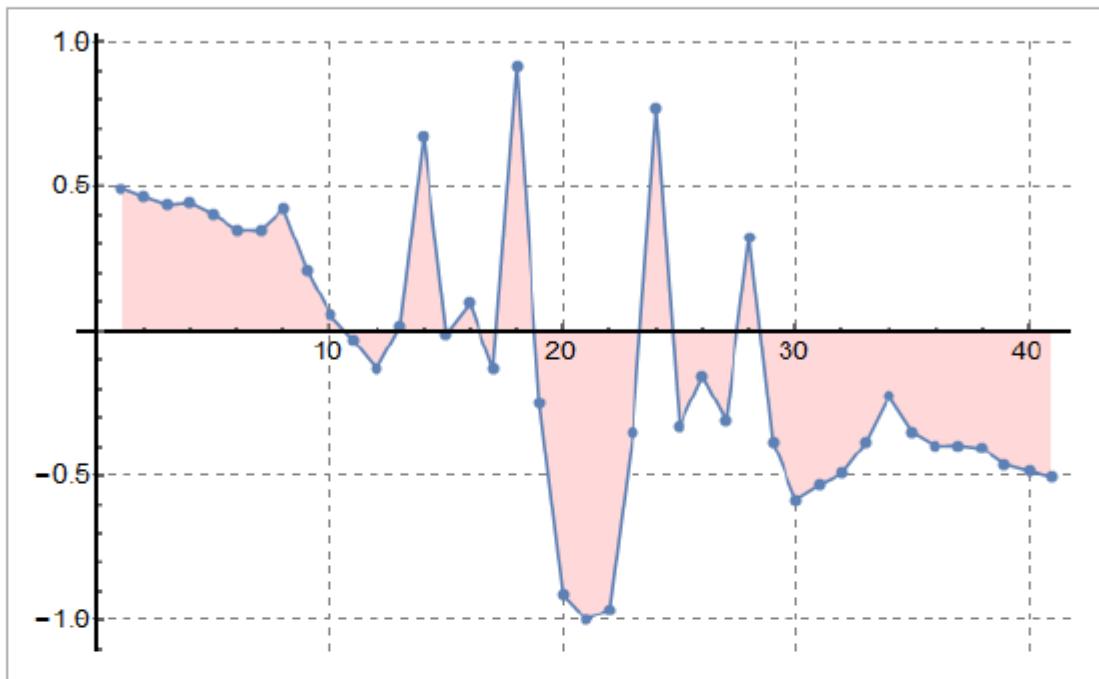


Рис. 10.3. Пример вывода табличных данных с использованием ListLinePlot

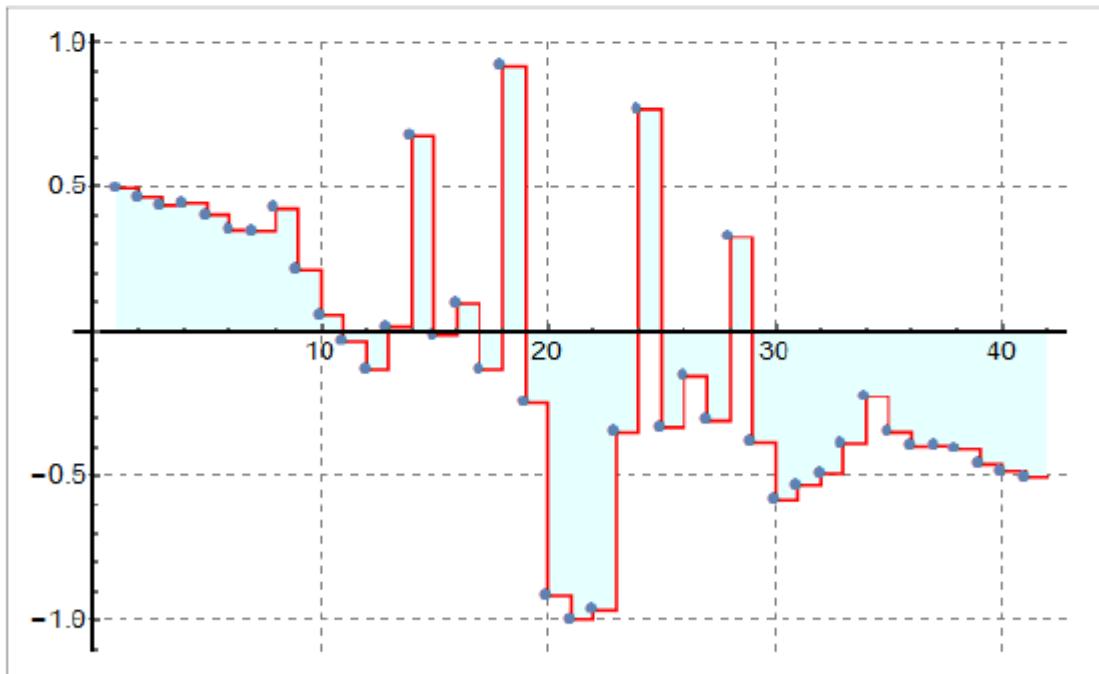
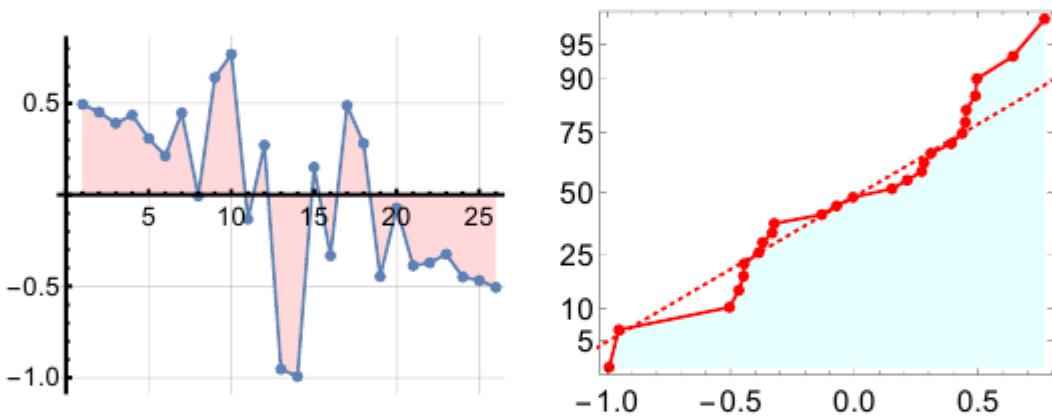


Рис. 10.4. Пример вывода табличных данных с использованием ListStepPlot

Тестовые задания для самоконтроля

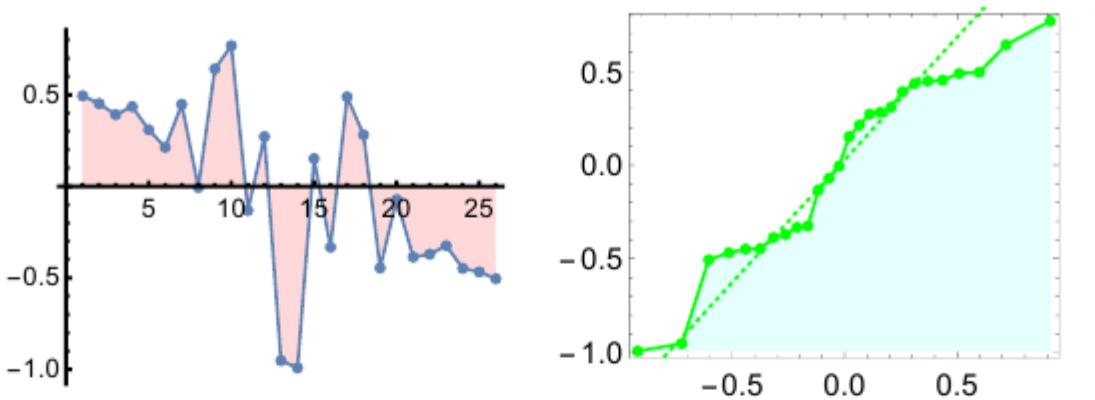
Упражнение 10.2a. Что следует вписать вместо Yaaaa для получения графика справа

```
f3[x_] := (-E^(-0.2*x^2)) * Cos[5*x^2] - 0.1*x
list3 = Table[f3[x], {x, -5, 5, 0.4}];
grListLP1 = ListLinePlot[list3, BaseStyle -> 16,
  GridLines -> {{5, 10, 20, 25}, {-1, -0.5, 0.5}},
  Mesh -> Full, Joined -> True,
  AxesStyle -> Directive[Black, Thick, 14],
  Filling -> Axis, FillingStyle -> LightRed,
  AspectRatio -> 0.8, ImageSize -> 260];
grListPSP = Yaaaa[list3, BaseStyle -> 16,
  Mesh -> Full, Joined -> True, PlotStyle -> Red,
  AxesStyle -> Directive[Black, Thick, 14],
  Filling -> Axis, FillingStyle -> LightCyan,
  AspectRatio -> 0.8, ImageSize -> 260];
Row[{grListLP1, grListPSP}, Spacer[20]]
```



Упражнение 10.2b. Что следует вписать вместо Ybbbb для получения графика справа

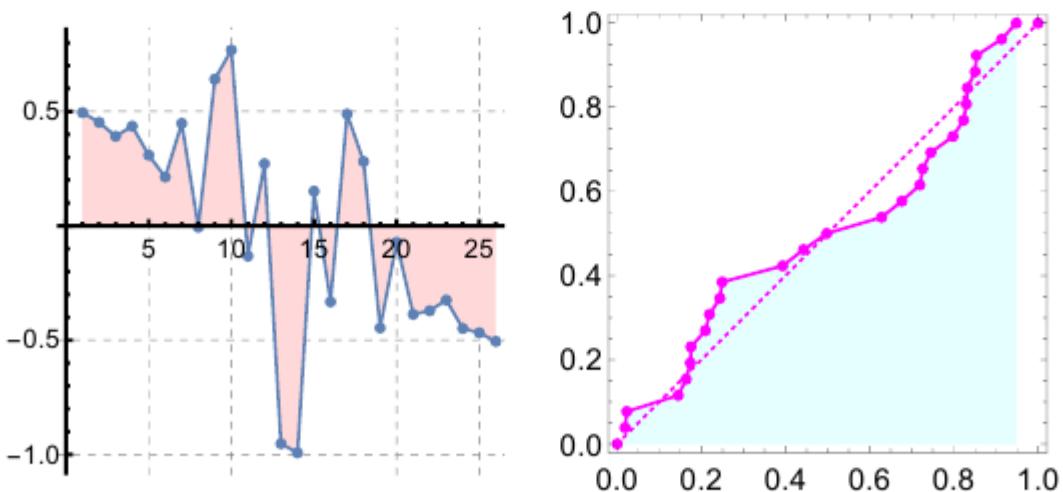
```
f3[x_] := (-E^(-0.2*x^2)) * Cos[5*x^2] - 0.1*x
list3 = Table[f3[x], {x, -5, 5, 0.4}];
grListLP1 = ListLinePlot[list3,
  BaseStyle -> 16, Mesh -> Full, Joined -> True,
  AxesStyle -> Directive[Black, Thick, 14],
  Filling -> Axis, FillingStyle -> LightRed,
  AspectRatio -> 0.8, ImageSize -> 260];
grListQP = Ybbbb[list3, BaseStyle -> 16,
  Mesh -> Full, Joined -> True, PlotStyle -> Green,
  AxesStyle -> Directive[Black, Thick, 14],
  Filling -> Axis, FillingStyle -> LightCyan,
  AspectRatio -> 0.8, ImageSize -> 260];
Row[{grListLP1, grListQP}, Spacer[20]]
```



*Упражнение 10.2с.* Что следует вписать вместо Ycccc для получения графика справа

```
f3[x_] := (-E^(-0.2 + x^2)) * Cos[5 + x^2] - 0.1*x
list3 = Table[f3[x], {x, -5, 5, 0.4}];
grListLP1 = ListLinePlot[list3,
  BaseStyle -> 16, Mesh -> Full, Joined -> True,
  AxesStyle -> Directive[Black, Thick, 14],
  GridLines -> {{5, 10, 20, 25}, {-1, -0.5, 0.5}},
  GridLinesStyle -> Directive[Gray, Dashed],
  (* директивой задается цвет и тип линий сетки *)
  Filling -> Axis, FillingStyle -> LightRed,
  AspectRatio -> 1., ImageSize -> 260];

grListPP = Ycccc[list3,
  BaseStyle -> 16, Mesh -> Full, Joined -> True,
  PlotStyle -> Magenta,
  AxesStyle -> Directive[Black, Thick, 14],
  Filling -> Axis, FillingStyle -> LightCyan,
  AspectRatio -> 1., ImageSize -> 260];
Row[{grListLP1, grListPP}, Spacer[20]]
```



**Гистограммы, столбиковые, круговые, секторные диаграммы:**

Визуализация данных разного вида диаграммами – это всегда графическое построение, которое предназначено для анализа имеющихся данных. Фактически при визуализации мы сталкиваемся с ограничением, которое накладывает восприятие, заключающееся в том, что размерность графических изображений не может быть больше двух, графические средства двумерные – лист бумаги или экран монитора. Количество визуальных образов, которыми могут представляться данные, ограничиваются только человеческой фантазией. Основное требование к ним – это наглядность и удобство анализа данных, которые они представляют. Методы визуализации могут быть как самые простые (линейные графики, диаграммы, гистограммы и т.п.), так и более сложные, основанные на математическом аппарате. Кроме того, при визуализации могут использоваться комбинации различных методов. Приведем функции системы *Mathematica*, которые условно можно отнести к изображениям типа диаграмм:

- `Histogram` – гистограмма; `Histogram[{x1, x2, ...}, bspec, hspec]` выводит гистограмму по списку значений с указанными спецификациями; `HistogramList` – гистограммные данные;
- `SmoothHistogram` – сглаженная гистограмма, выводит распределение `dfun`; вызов функции – `SmoothHistogram[{x1, x2, ...}, espec, dfun]`;
- `PairedHistogram` – спаренная гистограмма, вид графика, когда выводятся два столбчатых графика, имеющие одну координатную ось; как правило, состоит из двух гистограмм, расположенных в зеркальном отражении; вызов функции – `PairedHistogram[{x1, x2, ...}, {y1, y2, ...}]` для списков  $x_i$  и  $y_i$ ;
- `SmoothKernelDistribution` – распределение, интерполированное по методу ядерного сглаживания; вызов функции применительно к списку  $x_i$  – `SmoothKernelDistribution[{x1, x2, ...}]`;
- `RectangleChart` – прямоугольная диаграмма; при вызове `RectangleChart[{{x1, y1}, {x2, y2}, ...}]` выводит прямоугольные столбики ширины  $x_i$  и высоты  $y_i$ ;
- `BarChart` – столбиковая диаграмма; при вызове `BarChart[{y1, y2, ...}]` выводит прямоугольные столбики высоты  $y_i$ ;
- `PieChart` – круговая диаграмма; при вызове `PieChart[{y1, y2, ...}]` выводит сектора в круге пропорциональные  $y_i$ ;
- `SectorChart` – секторная диаграмма; при задании списков  $x_i$  и  $y_i$  вызове `SectorChart[{{x1, y1}, {x1, y2}, ...}]` выводит сектора радиуса  $y_i$  пропорциональные  $x_i$ ;

- `BubbleChart` – пузырьковая диаграмма; при вызове `BubbleChart[{{x1, y1, z1}, {x2, y2, z2}, ...}]` выводит пузырьки размера  $z_i$  с центрами в  $(x_i, y_i)$ ;
- `DistributionChart` – диаграмма вида скрипка; выводит при вызове `DistributionChart[{data1, data2, ...}]` диаграммы формы скрипки размера  $data_i$ .

Примеры работы с `PieChart`, иллюстрация на рисунке 10.5 двух вариантов оформления круговой диаграммы для списка  $\{2,4,3,4\}$ :

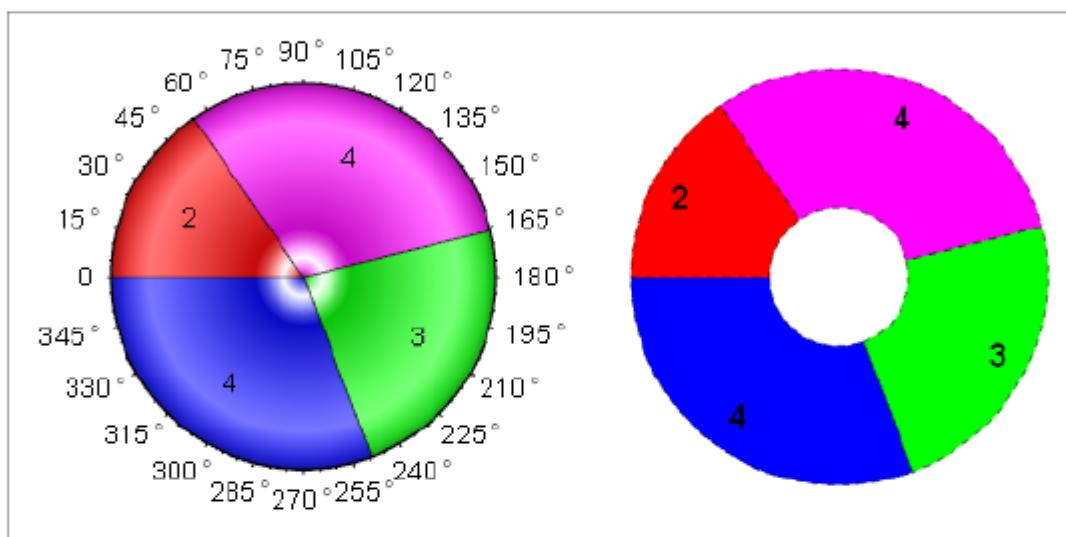


Рис. 10.5. Пример вывода табличных данных с использованием `PieChart`

#### Тестовые задания для самоконтроля

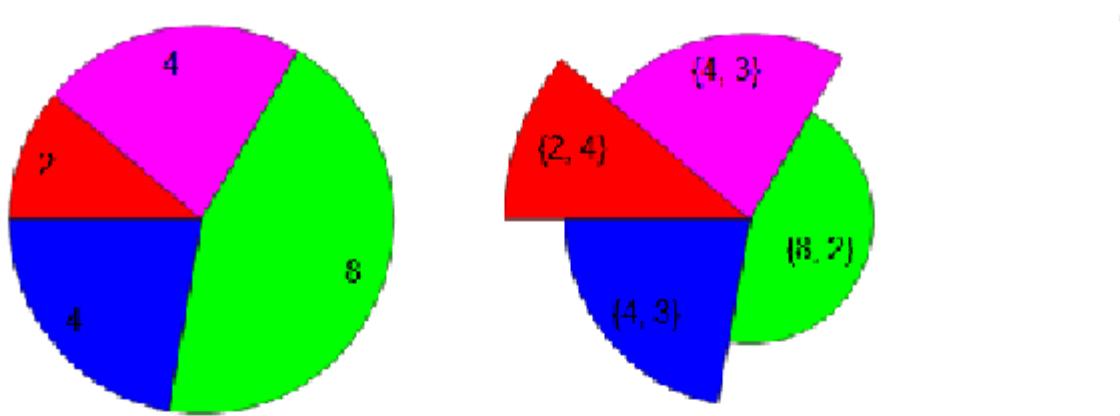
*Упражнение 10.1d.* Что следует вписать вместо Xdddd для получения графика справа?

```
lstCh1 = {2, 4, 8, 4};
lstCh2 = {{2, 4}, {4, 3}, {8, 2}, {4, 3}};
(* исходные данные для визуализации *)

grCh1 = PieChart[lstCh1, BaseStyle -> 16, ImageSize -> 250,
  ChartStyle -> {Red, Magenta, Green, Blue},
  ChartLabels -> Placed[lstCh1, "RadialOuter"]];
(* опцией Placed указан способ размещения подписей *)

grCh2 = Xdddd[lstCh2, BaseStyle -> 16, ImageSize -> 320,
  ChartStyle -> {Red, Magenta, Green, Blue},
  ChartLabels -> Placed[lstCh2, "RadialOuter"]];

Row[{grCh1, grCh2}]
```



Графики по точкам (с временной переменной):

- DateListPlot – график от календарного времени;
- DateListStepPlot – ступенчатый график от календарного времени;
- DateListLogPlot – график от времени в логарифмическом масштабе;
- DateHistogram – временная гистограмма;
- Databin – накопитель данных;
- TimelinePlot – хронологическая диаграмма;
- TimeSeries – временной ряд;
- EventSeries – временной ряд событий;
- HistoricalPeriodData – данные об исторических периодах.

Пример вывода с DateListPlot во временном интервале с 1/01/2017 по 30/09/2017 показан на рисунке 10.6.

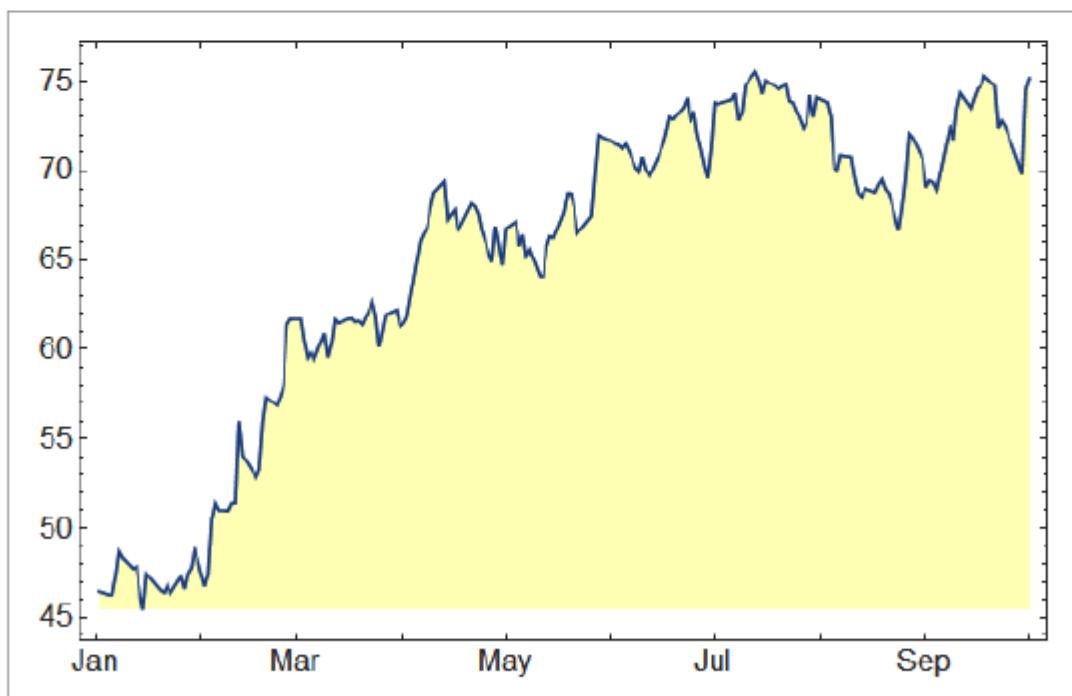


Рис. 10.6. Пример вывода данных с использованием DateListPlot

Для формирования графика выше сделан запрос на сервер Wolfram списка дат и стоимостей акций компании EPAM Systems на момент закрытия Нью-Йоркской фондовой биржи.

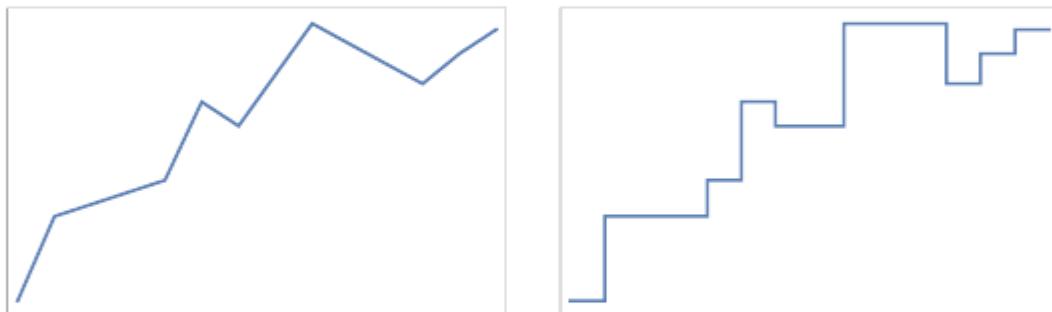
### Тестовые задания для самоконтроля

*Упражнение 10.1e.* Что следует вписать вместо Xeeee для получения графика справа?

```
fData0 = {{ {2016, 11, 17}, 63},
    {{2016, 11, 18}, 64.4}, {{2016, 11, 21}, 65.0},
    {{2016, 11, 22}, 66.3}, {{2016, 11, 23}, 65.9},
    {{2016, 11, 25}, 67.6}, {{2016, 11, 28}, 66.6},
    {{2016, 11, 29}, 67.1}, {{2016, 11, 30}, 67.5}};

grDLP = DateListPlot[fData0, BaseStyle -> 14, ImageSize -> 260];
grDLSP = Xeeee[fData0, BaseStyle -> 14, ImageSize -> 260];

Row[{grDLP, grDLSP}, Spacer[20]]
```



### Финансовые диаграммы:

- **TradingChart** – торговый график, генерирует диаграмму, показывающую цены (минимальную, максимальную, при открытии и закрытии) объемы продаж на каждую дату диапазона;
- **InteractiveTradingChart** – трейдинговая интерактивная диаграмма (“живой” график изменения цен с возможностью указания дополнительных индикаторов анализа, работы в выбранном временном интервале, в том числе, с диапазонами неделя или месяц);
- **LineBreakChart** – диаграмма трендовых линий;
- **BoxWhiskerChart** – диаграмма ящик с усами, блочная диаграмма, представляющая средство для изображения пяти статистических показателей (от прямоугольника отходят «усы», равные по длине одному из показателей разброса или точности);
- **KagiChart** – диаграмма Каги, представляет собой ступенчатый график состоящий из линий разной толщины или разного цвета;

- CandlestickChart – диаграмма японские свечи, чаще всего используется для отображения биржевых, а именно значений высокой цены, низкой цены, цены открытия и цены закрытия;
- RenkoChart – график Ренко, применяемый в техническом анализе вид графика котировок объекта торговли (товара, ценной бумаги, валюты), который отображает изменения цены в форме диагональных рядов;
- PointFigureChart – пункто-цифровой график.

Пример вывода с TradingChart показан на рисунке 10.7. Используя функцию FinancialData, сделан запрос на сервер Wolfram списка цен и объемов продаж акций компании EPAM Systems на Нью-Йоркской фондовой бирже во временном интервале с 20/04/2017 по 30/08/2017. Выведена диаграмма, иллюстрирующая на каждую дату объем продаж, стоимости акций (минимальную, максимальную, при открытии и закрытии):

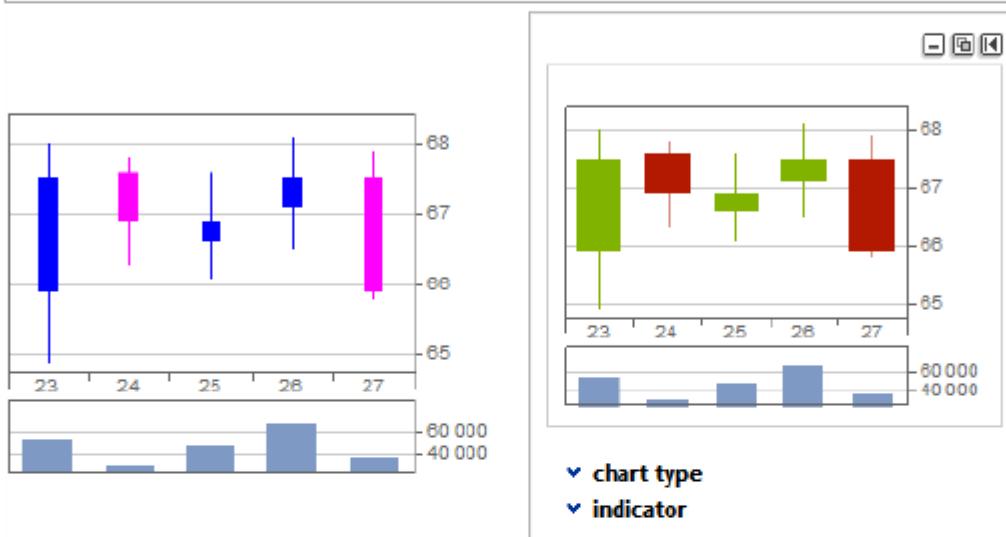


Рис. 10.7. Пример вывода данных с использованием TradingChart

#### Тестовые задания для самоконтроля

Упражнение 10.1f. Что следует вписать вместо Xffff для получения графика справа?

```
fDataOHLCV = {{{2017, 10, 23}, {65.9, 68., 64.9, 67.5, 53950}},  
 {{2017, 10, 24}, {67.6, 67.8, 66.3, 66.9, 31740}}},  
 {{2017, 10, 25}, {66.6, 67.6, 66.1, 66.9, 48200}}},  
 {{2017, 10, 26}, {67.1, 68.1, 66.5, 67.5, 66750}}},  
 {{2017, 10, 27}, {67.5, 67.9, 65.8, 65.9, 38080}}}};  
 grTrCh = TradingChart[fDataOHLCV, AspectRatio -> 7/10,  
 ImageSize -> 260, TrendStyle -> {Blue, Magenta}];  
 grITrCh = Xffff[fDataOHLCV, AspectRatio -> 7/10, ImageSize -> 220];  
 Row[{grTrCh, grITrCh}, Spacer[10]]
```



Для формирования графиков выше использовались данные, полученные по соответствующему запросу с сервера Wolfram.

#### Специальные функции графики пакетов-приложений:

- **ErrorListPlot** – генерирует график с нанесением погрешностей (пакет **ErrorBarPlots**); используется для иллюстрации данных, которые содержат погрешности – прямое отображение погрешностей в поле графика, а именно: у проставленной точки строят вертикальный отрезок, его длина в выбранном масштабе показывает погрешность.
- **ErrorBar** – генерирует график с нанесением погрешностей в положительном и отрицательном направлениях (пакет **ErrorBarPlots**), вокруг точки строят два отрезка, параллельные осям абсцисс и ординат, в выбранном масштабе длина каждого отрезка должна равняться погрешности величины, откладываемой по параллельной оси.
- **ParetoPlot** – генерирует график Парето, графическое отражение закона Парето, кумулятивной зависимости распределения определённых ресурсов или результатов от большой совокупности (выборки) причин (пакет **StatisticalPlots**); тип графика, в котором строятся полосы в нисходящем порядке, начиная слева; позволяет наглядно представить вклад отдельных факторов в общий результат; используется и помогает выделить “жизненно важное меньшинство” по сравнению с

“незначительным большинством”.

Пример вывода с ErrorListPlot показан на рисунке 10.8:

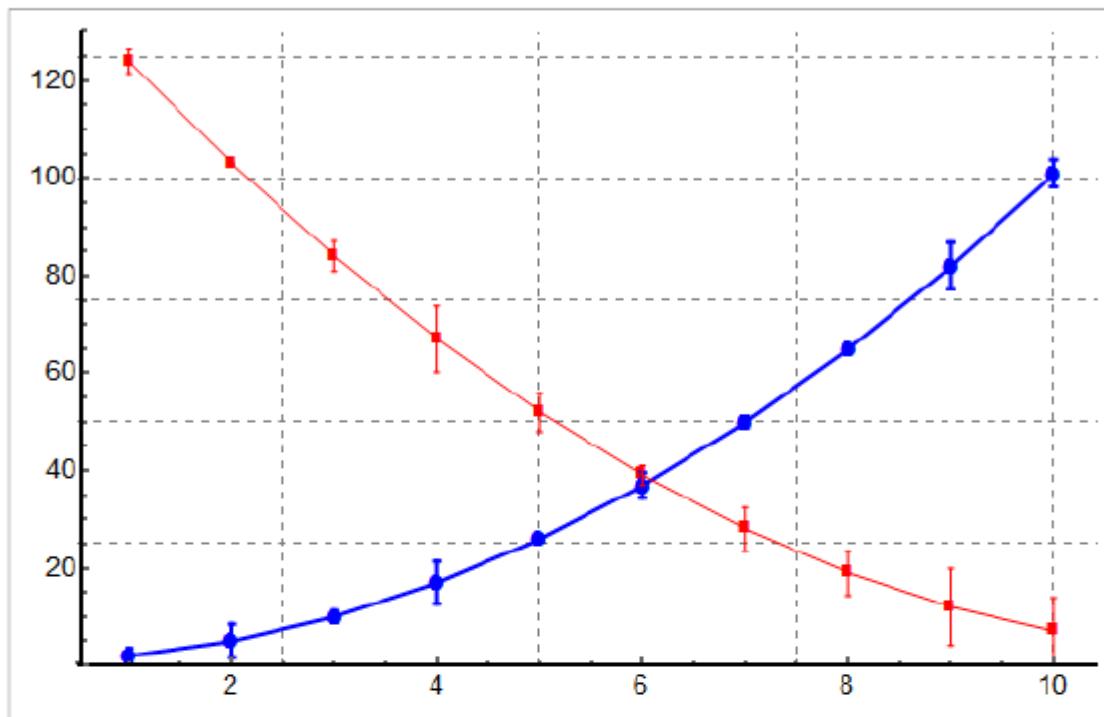


Рис. 10.8. Пример вывода данных с использованием ErrorListPlot

**Замечания:** Формирование и визуализация кривых Безье, Эрмита, сплайнов – предмет отдельного изучения.

Средства 2D графики аналитически задаваемых функций:

- ContourPlot – контурный график на плоскости, который показывает линии равного уровня (изолинии); контурные графики (или графики в горизонталях) наиболее часто используются в топографии для представления на плоскости объемного рельефа местности;
- DensityPlot – плотностный график (карта зон), значение функции в каждой точке отображается при помощи окрашивания в определённый цвет, можно задавать интервалы и соответствующие им цвета, можно применять плавные цветовые переходы (градиентная заливка);
- RegionPlot – визуализация геометрической фигуры на плоскости;
- ParametricPlot – используется для рисования на плоскости графиков функций, заданных параметрически;
- StreamPlot – линии тока на плоскости (диаграмма потоков); для построения линий тока используются тангенциальные компоненты векторных переменных; в гидромеханике – линия, направление касательной к которой в каждой точке совпадает с направлением скорости частицы жидкости в этой точке (в каждый момент времени частица движется вдоль линии тока);

- StreamDensityPlot – линии тока на плоскости с фоном плотности функции (плотностная диаграмма потоков);
- VectorPlot – векторное поле на плоскости, графическое изображение при помощи направленных отрезков – векторов;
- VectorDensityPlot – векторно-плотностная диаграмма;
- LineIntegralConvolutionPlot – диаграмма потоков над изображением по LIC-методу (Line Integral Convolution – линейная интегральная свертка).

Пример формирования и вывода изолиний функции  $f_{XY}(x, y) = -e^{-(x/2-2)^2-(y-2)^2} + 2e^{-(x/2-3)^2-(y/3-1)^2} + 2$  показан на рисунке 10.9. Отметим, что этот графический объект и все приведенные выше включают динамическую интерактивность. В частности, на скриншоте видно, что под указателем вблизи изолинии подсвечивается значение уровня поверхности. В приведенном примере уровни задавались списком, хотя в большинстве случаев достаточно задавать просто число изолиний, тогда уровни определяются с одним шагом от минимального до максимального значений функции в области определения.

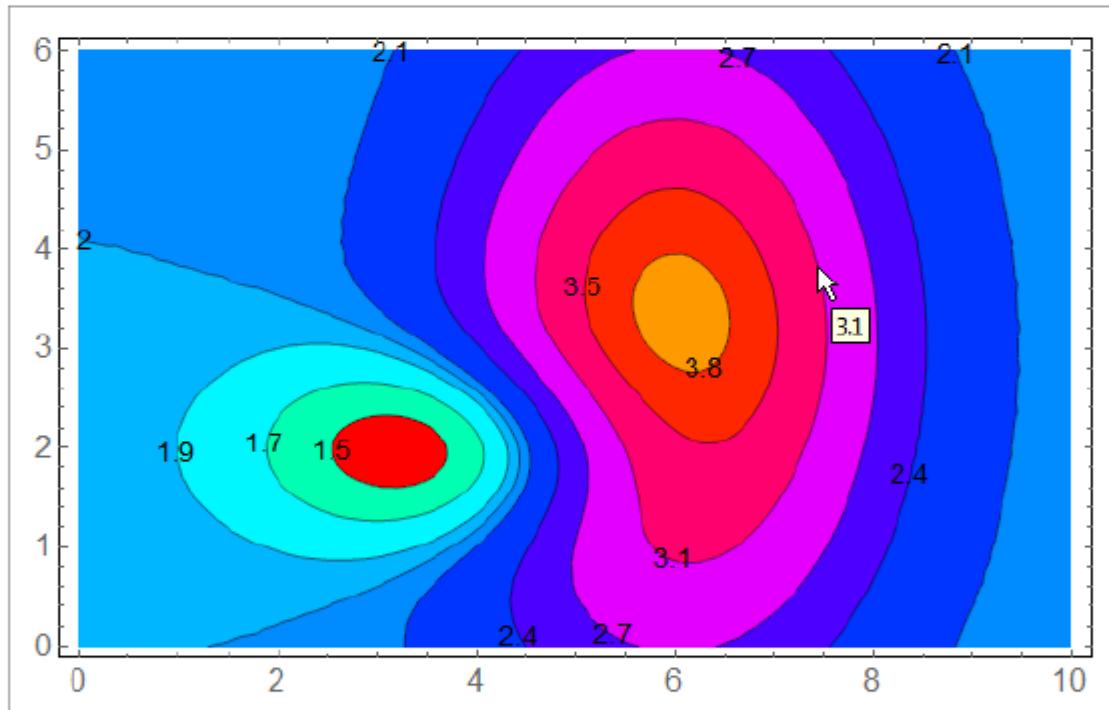


Рис. 10.9. Пример формирования и вывода изолиний с использованием функции ContourPlot

#### Тестовые задания для самоконтроля

*Упражнение 10.1g.* Что следует вписать вместо Xgggg для получения графика справа?

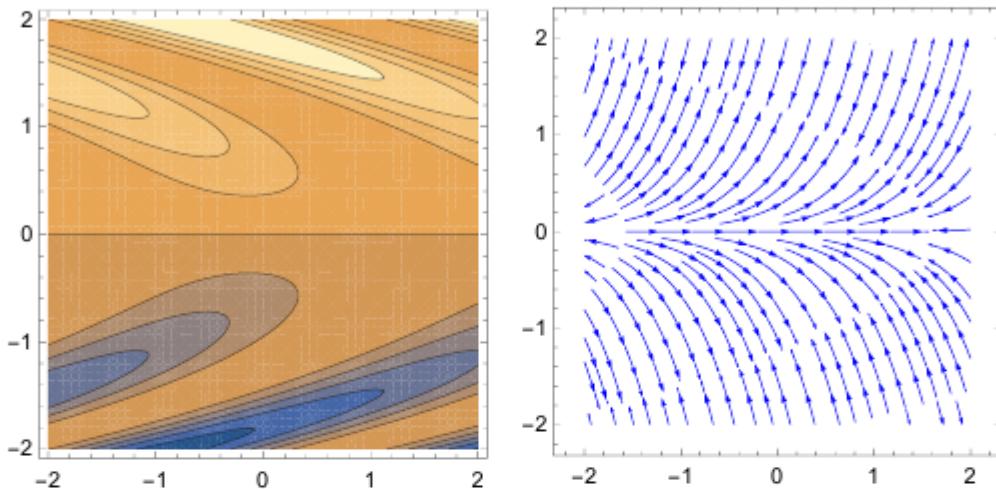
```

grfCP1 = ContourPlot[{Cos[x + y^2] * 2 y Cos[x + y^2]}, {x, -2, 2},
{y, -2, 2}, Contours -> 10, BaseStyle -> {12}, ImageSize -> 250];

grStrPl1 =
Xgggg[{Cos[x + y^2], 2 y Cos[x + y^2]}, {x, -2, 2}, {y, -2, 2},
StreamStyle -> Blue, BaseStyle -> {12}, ImageSize -> 250];

Row[{grfCP1, grStrPl1}, Spacer[10]]

```



#### Функции интерполяции и визуализации 2D графики массивов данных:

- ListContourPlot – контурный график по массиву значений;
- ListDensityPlot – плотностный график по массиву значений;
- ListVectorPlot – векторная диаграмма, формируемое по данным векторное поле на плоскости;
- ListVectorDensityPlot – плотностно-векторная диаграмма по данным;
- ListStreamPlot – диаграмма интенсивности потоков, формируемая по таблицам данных; выполняется аппроксимация, а затем построение линий тока на плоскости;
- ListStreamDensityPlot – плотностная диаграмма интенсивности потоков по данным;
- ListLineIntegralConvolutionPlot – диаграмма массива векторов над изображением по LIC-методу (Line Integral Convolution), повышение выразительности визуальных образов двумерных и трехмерных векторных полей, обеспечиваемое использованием текстур.

Пример аппроксимации набора дискретных данных, формирования и вывода изолиний сеточной функции показан на рисунке 10.10. Данные примера – таблица рассчитанных с шагом 0.5 значений использованной в примере выше функции  $f_{XY}(x,y)$ . Отдельно отметим, что порядок аппроксимации InterpolationOrder принят по умолчанию.

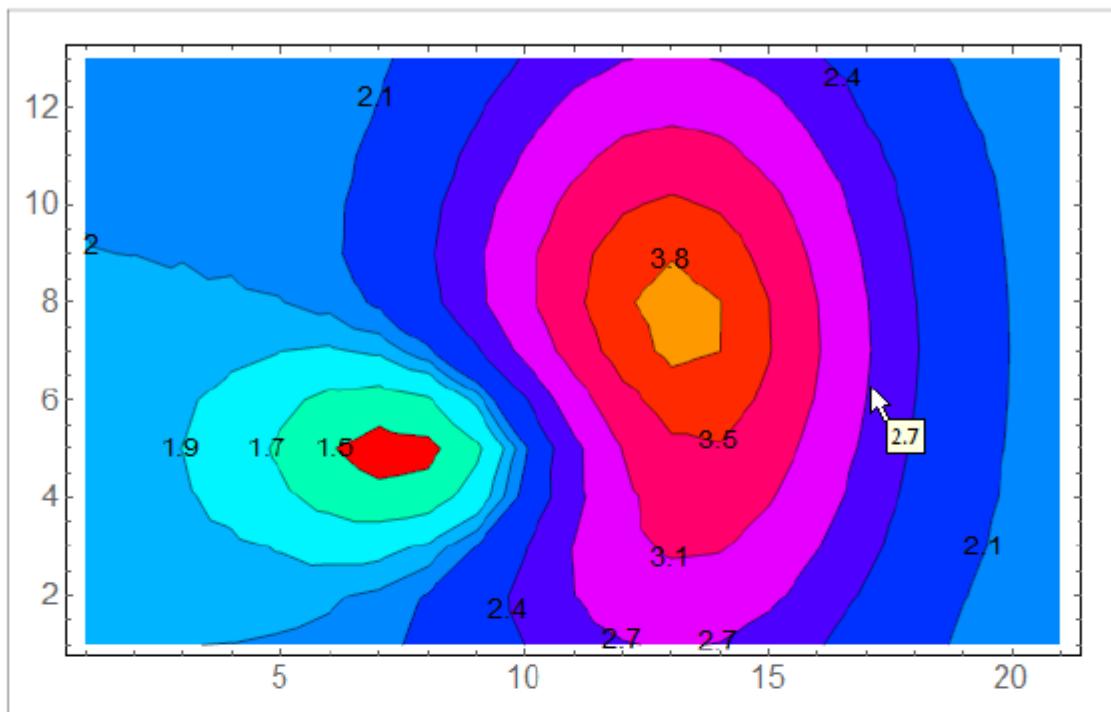


Рис. 10.10. Пример аппроксимации набора данных, формирования и вывода изолиний с использованием функции ListContourPlot

#### Функции формирования и вывода объектов 2D графики:

- **Graphics** – генерирует примитивы 2D графики (отрезки, полигоны, разные треугольники, прямоугольники, параллелограммы, ромбы, окружности, дуги, сегменты, В-сплайны, Безье-кривые,
- **GraphPlot** – визуализация графа; строит граф по матрице вершинной смежности, учитывается кратность ребра;
- **TreePlot** – генерирует графическое дерево графа на плоскости;
- **TreeForm** – изображение выражений в виде дерева с разными уровнями по глубинам (древовидные графы);
- **ArrayPlot** – изображение массива; генерирует изображение, в котором значениям массива соответствуют раскрашенные квадраты;
- **MatrixPlot** – визуализация матрицы; функция обеспечивает построение квадрата из клеток, цвет которых соответствует значению соответствующего элемента матрицы;
- **ReliefPlot** – рельефная диаграмма, используется для построения реалистических (с разным разрешением и разной окраской) графиков рельефа, который задается ординатами точек массива;
- **GeoGraphics** – гео-графика, представление 2D географических изображений.

Пример использования функции GraphPlot для визуализации графа показан на рисунке 10.11.

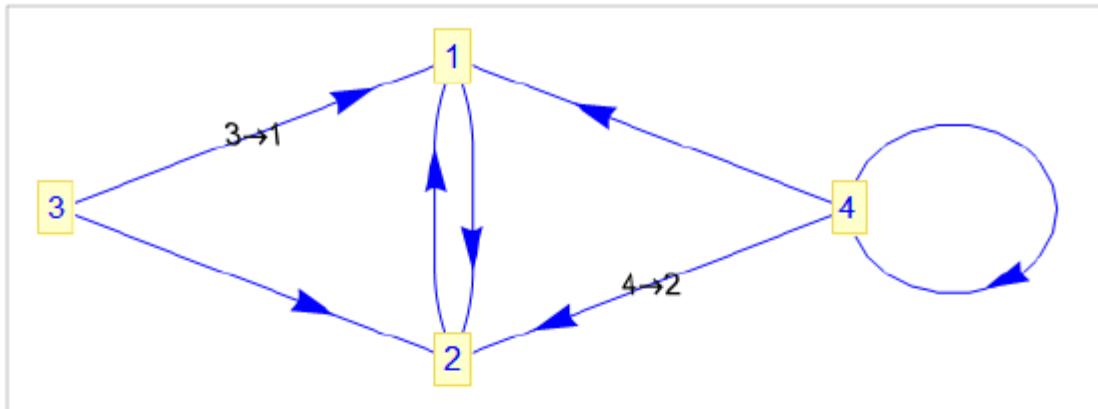


Рис. 10.11. Пример вывода графа

#### Тестовые задания для самоконтроля

*Упражнение 10.1h.* Что следует вписать вместо Xhhhh для получения графика внизу?

```

fTrFrm = {1 → 2, 2 → 1, {3 → 1, "3→1"},  

          3 → 2, 4 → 1, {4 → 2, "4→2"}, 4 → 4};  
  

GraphPlot[fTrFrm, VertexLabeling → True,  

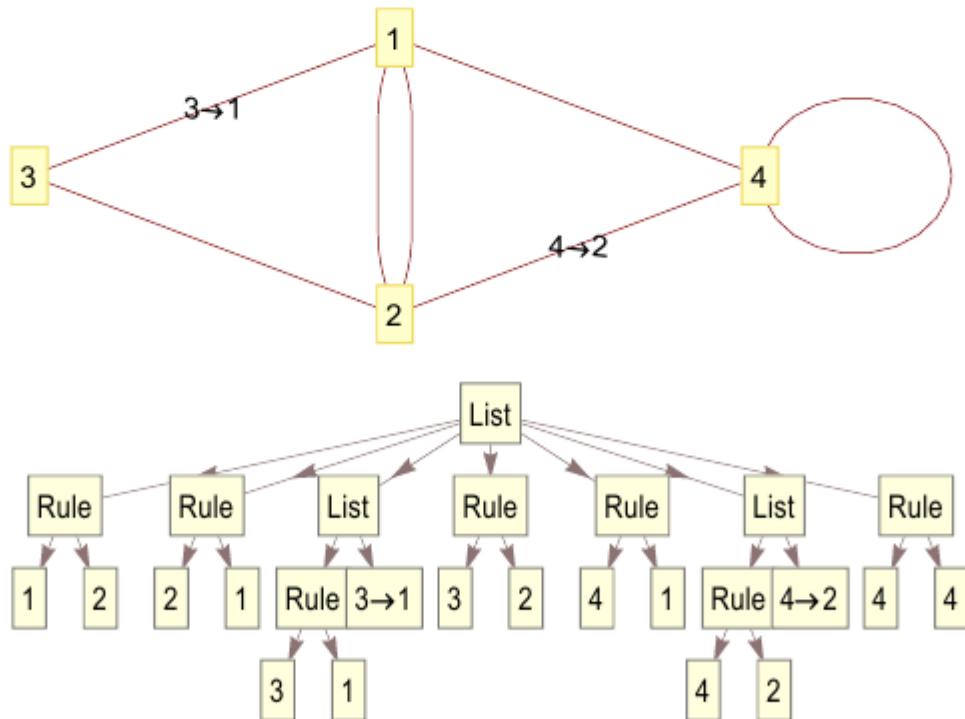
          BaseStyle → 16, AspectRatio → 3 / 10, ImageSize → 600]  

Xhhhh[fTrFrm, DirectedEdges → True, BaseStyle → 16,  

       VertexLabeling → True, AspectRatio → 3 / 10,  

       ImageSize → 600]

```



Средства 3D графики аналитически задаваемых функций:

- Plot3D – 3D визуализация функции двух переменных, заданной явно аналитически в декартовых координатах; с графиком можно работать интерактивно, обеспечиваются: изменение масштаба, повороты и перемещения (интерактивность поддерживается для всех перечисленных ниже функций выводимых 3D изображений); как и для всех случаев ниже, возможна визуализация нескольких функций, обязательно задать область определения;
- ContourPlot3D – контурный график явно заданной в декартовых координатах функции в пространстве; трехмерный контурный график включает также расположенные в пространстве линии равного уровня, показывающие границы слоев трехмерной фигуры в секущих плоскостях, расположенных параллельно опорной плоскости фигуры (расположенные в пространстве линии равного уровня, полученные при расслоении трехмерной фигуры рядом секущих плоскостей, расположенных параллельно опорной плоскости фигуры);
- SliceContourPlot3D – пространственный контурный график функции на срезах;
- RegionPlot3D – визуализация геометрической фигуры в пространстве;
- DensityPlot3D – пространственный график, иллюстрирующий распределение в пространстве заданной функции от двух переменных на основе непрерывной или специально задаваемой цветовой схемы;
- SliceDensityPlot3D – пространственный плотностный график функции на срезах (сечениях);
- ParametricPlot3D – график параметрически заданной аналитическими выражениями в трехмерном пространстве двухмерной поверхности или одномерной кривой;
- VectorPlot3D – векторное поле в пространстве (совокупность векторов); должно быть определено, как вектор-функция трех координат;
- SliceVectorPlot3D – генерируется пространственный векторный график функции на плоскостях-срезах (только в указанных плоскостях, в остальном пространстве вектора не отрисовываются);
- RevolutionPlot3D – график поверхности вращения (трехмерные объекты, полученные вращением кривых);
- SphericalPlot3D – функция строит график трехмерной поверхности, заданной в сферических координатах.

Пример использования Plot3D для визуализации выше приведенной функции  $f_{XY}$  показан на рисунках 10.12 и 10.13. Результаты получены с применением разных цветовых схем для раскраски, приведены в разных ракурсах осмотра поверхности; в обоих случаях дополнительно (пунктирные кривые) выводятся линии уровня (обеспечивает опция MeshFunctions).

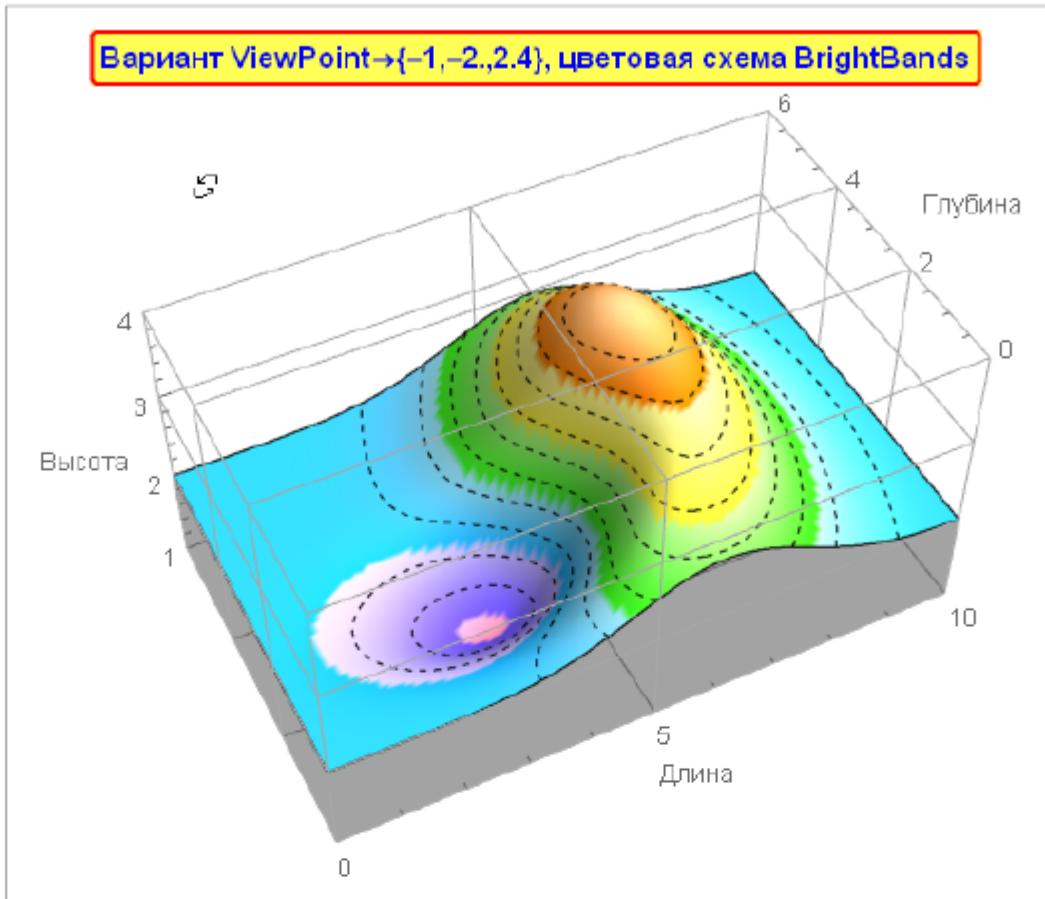


Рис. 10.12. Пример вывода с использованием Plot3D

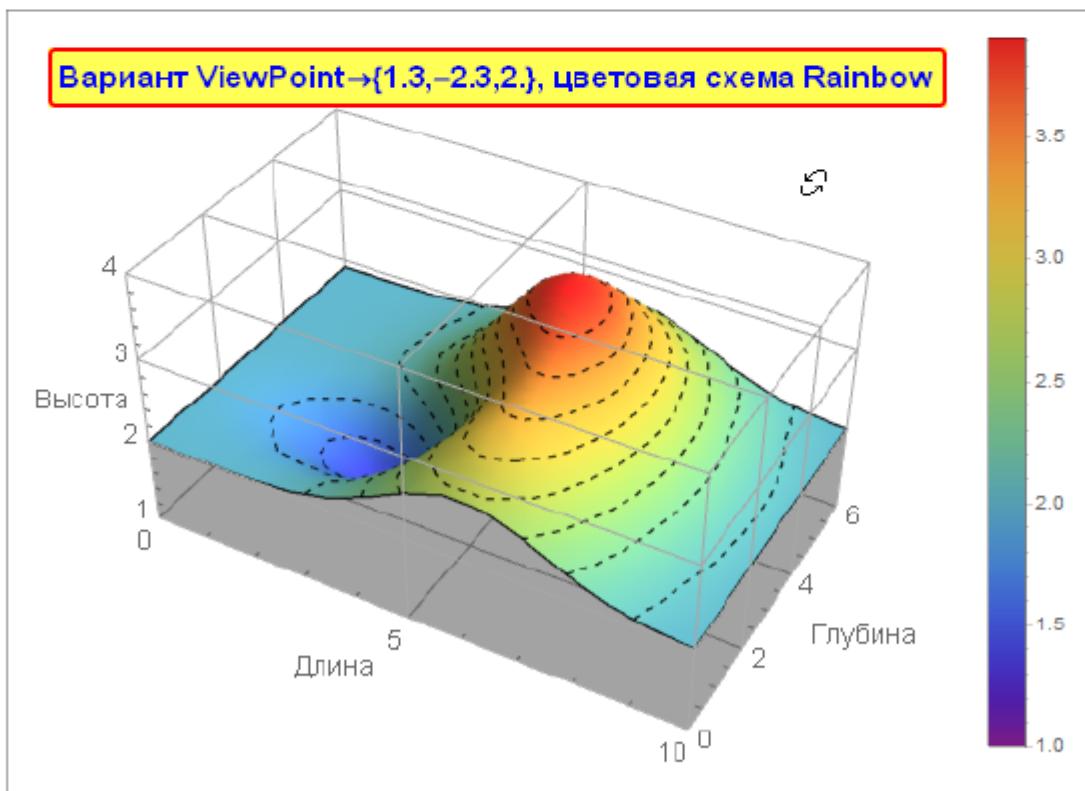


Рис. 10.13. Пример визуализации с выводом легенды

### Функции интерполяции и визуализации 3D графики массивов данных:

- ListPlot3D – трехмерная диаграмма разброса данных, представление в пространстве поверхности по массиву значений высот в точках;
- ListContourPlot3D – контурный 3D-график по массиву значений;
- ListSurfacePlot3D – 3D-график поверхности, восстановленной по списку точек;
- ListPointPlot3D – 3D-диаграмма разброса данных, в пространстве генерируются изображения точек с тремя заданными координатами;
- ListDensityPlot3D – пространственный плотностный график по данным;
- ListSliceContourPlot3D – 3D контурный график данных на срезах;
- ListSliceDensityPlot3D – 3D плотностный график данных на срезах;
- ListVectorPlot3D – векторная 3D-диаграмма по данным;
- DiscretePlot3D – график дискретной функции двух переменных;
- RectangleChart3D – трёхмерная столбиковая/брюсковая диаграмма;
- BarChart3D – трёхмерная столбиковая диаграмма;
- Histogram3D – гистограмма 3D;
- SmoothHistogram3D – сглаженная гистограмма в пространстве;
- SectorChart3D – секторная диаграмма в пространстве;
- PieChart3D – пространственная круговая диаграмма.

Пример использования DiscretePlot3D для визуализации наборов данных показан на рисунке 10.14. Результаты визуализируют табличные данные, сформированные расчетом  $\{\text{Sin}[t+u], \text{Cos}[t+u]\}$  в прямоугольнике  $0 \leq t \leq 2\pi$ ,  $0 \leq u \leq 2\pi$  с шагом  $\pi/15$ .

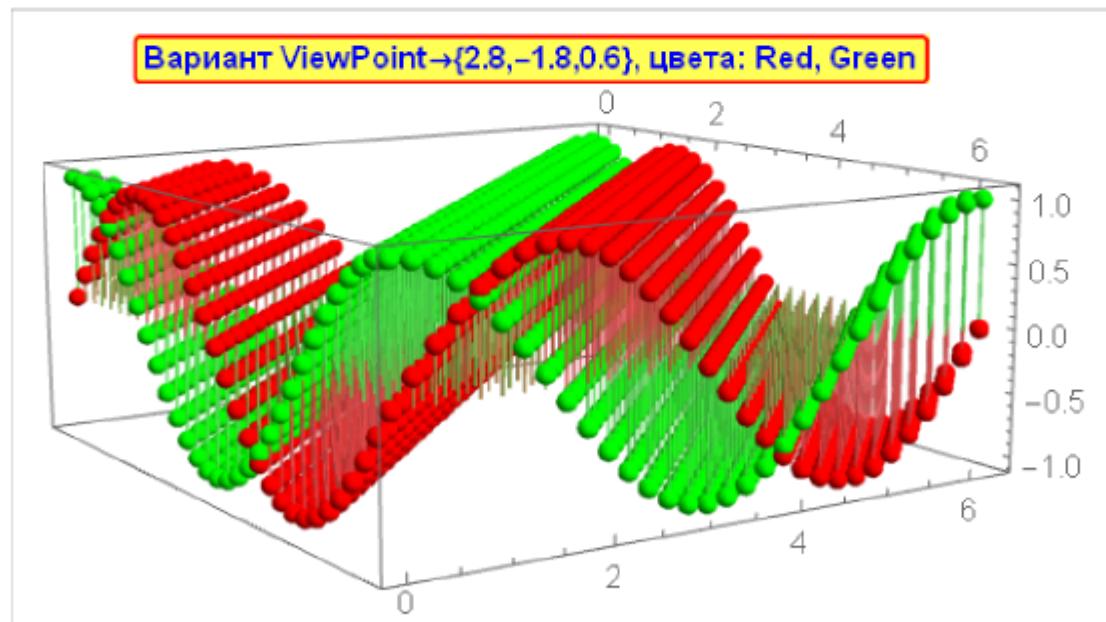


Рис. 10.14. Пример вывода с использованием DiscretePlot3D

Функции сбора, формирования и вывода объектов 3D графики, географические визуализации и вычисления:

- GeoGraphics – географические информационные системы (в сочетании с GeoRange, GeoProjection, GeoPosition, GeoBackground, GeoPath, GeoDistance, GeoDisplacement, GeoModel, GeoScaleBar, GeoZoomLevel, GeologicalPeriodData, EarthquakeData, GeoBounds, CountryData, CityData, GeoStyling, ListPlot3D, ReliefPlot);
- GeoListPlot – обеспечивает вывод на географической картооснове указанных объектов (границы, города, пути и другое);
- GeoRegionValuePlot – картограмма с раскраской географических областей указанными цветами;
- GeoElevationData – данные о высоте над уровнем моря (в сочетании с \$GeoLocation, GeoPosition, GeoZoomLevel, GeoGraphics, ListPlot3D, ReliefPlot);
- MountainData – данные о высоте гор (в сочетании с GeoGraphics).

Примеры использования GeoGraphics, \$GeoLocation, GeoPath для визуализации географических данных показаны на рисунках 10.15 и 10.16.



Рис. 10.15. Пример вывода с использованием GeoGraphics, \$GeoLocation

Результаты получены следующими запросами и функциями:

- используя \$GeoLocation, определены координаты места, откуда сделан запрос;
- функция GeoGraphics с настройками Frame True, GeoProjection Equirectangular, GeoZoomLevel = 11 выводит карту, показанную на рисунке 10.15;

- функция GeoPath в сочетании с Arrow обеспечивает отрисовку карты и направления \$GeoLocation → Warsaw (city); показаны на рисунке 10.16 (запрос сделан с компьютера, находящегося в Минске).



Рис. 10.16. Пример вывода с использованием GeoGraphics, GeoPath

Дополнительные детали, подробности, иллюстрации можно прочитать в перечисленных ниже источниках.

#### Рекомендуемая литература

1. WOLFRAM MATHEMATICA. Наиболее полная система для современных технических вычислений в мире [Электронный ресурс]. URL: <http://www.wolfram.com/mathematica>
2. WOLFRAM MATHEMATICA. Новое в системе Mathematica 11 [Электронный ресурс]. URL: <http://www.wolfram.com/mathematica/new-in-11/>
3. WOLFRAM MATHEMATICA. Новые функциональные возможности в Mathematica 10 [Электронный ресурс]. URL: <http://www.wolfram.com/mathematica/new-in-10/>



## Введение в графику системы *Mathematica*

Таранчук Валерий Борисович

БГУ,

факультет прикладной математики и информатики

Учебные материалы, инструкции и рекомендации пользователям системы компьютерной алгебры *Mathematica*, обучающие примеры и упражнения (оригинал документа создан и предоставляется студентам в формате NB)

### ► Содержание лекций и занятий 1 - 10

Уважаемые читатели. В сгруппированной секции выше (и везде далее в подобных) при работе с оригинальным блокнотом, раскрывая группу секций, вы получаете дополнительные материалы для самостоятельного изучения.

### ▼ Содержание лекции и занятия 11

- ✓ *Mathematica*. Общие правила работы с графикой
- ✓ О выводе и манипуляциях с объектами графики
- ✓ Тестовые задания для самоконтроля
- ✓ Примеры применения функций Show, GraphicsRow
- ✓ Тестовые задания для самоконтроля
- ✓ Цвет и его задание в системе *Mathematica*
- ✓ Тестовые задания для самоконтроля

### ▼ *Mathematica*. Общие правила работы с графикой

Программирование графики в *Mathematica* относится к функциональному типу. Каждая функция имеет определённый набор атрибутов и может иметь набор опций, директив. Пользователи системы могут применять опции для управления работой отдельных функций. Опции влияют на результат выполнения функции, различные варианты опций могут дать совершенно разные по виду результаты. Опция всегда имеет значение по умолчанию, которое определяет действие функции, если не указано другое. Перечни опций графических функций приводятся в системе помощи, этих опций очень много, причём при их перечислении у конкретных функций отмечаются индивидуальные, а фактически можно применять значительно больше, в частности почти все опции функции Plot.

Поскольку графики в системе *Mathematica* являются объектами, то они могут быть значениями переменных. *Mathematica* допускает следующие стандартные для ее языка конструкции, иллюстрации которых приведём на примере функции Plot:

- Plot[Sin[x],{x,0,20}] – построение графика синусоиды (немедленный вывод);

- `gr1:=Plot [Sin [x], {x, 0, 20}]` – формирование объекта (графика синусоиды) с отложенным выводом;
- `gr2=Plot [Sin [x], {x, 0, 20}]` – задание объекта с немедленным выводом.

Далее для иллюстраций будут использоваться введенные ранее выражения:

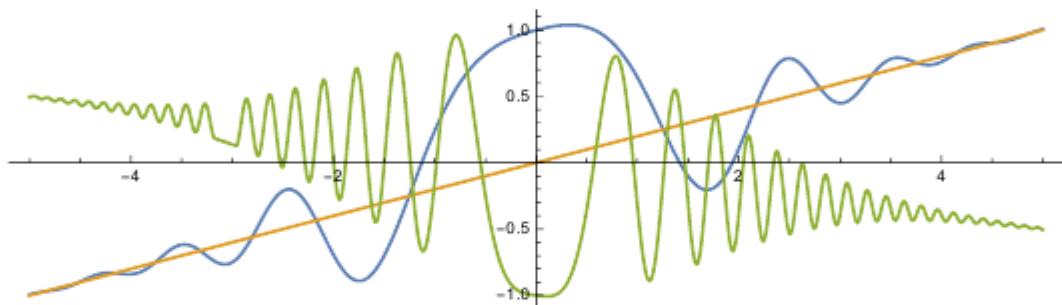
```
f1[x_] := Cos[x^2] / E^(0.2 * x^2) + 0.2 * x
f2[x_] := 0.2 * x
f3[x_] := (-E^(-0.2 * x^2)) * Cos[5 * x^2] - 0.1 * x
```

дополнительно определим объект:

```
grf1D := Plot[{f1[x], f2[x], f3[x]}, {x, -5, 5}, AspectRatio -> 2/7, ImageSize -> 550]
```

После того, как графический объект определен, его можно использовать, работая через имя. В примере ниже выведем этот график:

```
grf1D
```



Иногда отдельные детали построения графиков в системе оказываются для пользователя неожиданными и не вполне понятными. Причина – множество опций, которые могут использоваться в графиках, причем в самых различных сочетаниях. Поэтому полезно знать, как можно получить информацию о свойствах графических объектов, опциях конкретного результата, в том числе в случаях, когда нет кода, а есть только объект вывода. Отметим, что часто даже небольшая модификация опций (например, замена цвета или типа линий) делает график полностью удовлетворяющим требованиям пользователя. Информацию об опциях графического объекта дают, например, приведенные ниже функции. Продолжим работу с объектом. В следующей секции записана функция вывода основных опций графика grf1D:

```
Options[grf1D]
```

```
{DisplayFunction -> Identity, AspectRatio -> 2/7,
Axes -> {True, True}, AxesLabel -> {None, None},
```

Продолжение на следующей странице

### Продолжение. Начало на предыдущей странице

```
AxesOrigin -> {0, 0}, DisplayFunction :> Identity,
Frame -> {{False, False}, {False, False}},
FrameLabel -> {{None, None}, {None, None}},
FrameTicks -> {{Automatic, Automatic}, {Automatic, Automatic}},
GridLines -> {None, None},
GridLineStyle -> Directive[GrayLevel[0.5, 0.4]],
ImageSize -> 550, Method -> {"DefaultBoundaryStyle" -> Automatic,
"DefaultMeshStyle" -> AbsolutePointSize[6],
"ScalingFunctions" -> None},
PlotRange -> {{-5, 5}, {-1.00678, 1.03866}},
PlotRangeClipping -> True, PlotRangePadding ->
{{Scaled[0.02], Scaled[0.02]}, {Scaled[0.05], Scaled[0.05]}},
Ticks -> {Automatic, Automatic}}
```

Обратите внимание, что сказано “записана функция вывода основных опций”. Полный список можно получить, выполняя `AbsoluteOptions`:

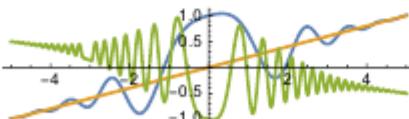
```
AbsoluteOptions[grf1D]
```

Важно заметить также, что аргументами функций `Options` и `AbsoluteOptions` могут быть сами объекты (не имя, а копия графического изображения системы *Mathematica*), причем можно уточнить какую именно опцию или директиву следует вывести. Примеры даны ниже. В следующей секции записан запрос об области вывода и размере изображения объекта `grf1D` (перечислены в списке `{PlotRange, ImageSize}`), при этом, конечно же, объект предварительно должен быть сформирован:

```
Options[grf1D, {PlotRange, AspectRatio}]
```

```
{PlotRange -> {{-5, 5}, {-1.00678, 1.03866}}, AspectRatio ->  $\frac{2}{7}$ }
```

В секции ниже дан такой же запрос, но аргументом функции `Options` является не имя объекта, а само изображение, которое вставлено на место аргумента из буфера обмена, куда было скопировано из Out-секции после выполнения `grf1D`:

```
Options[, {PlotRange, AspectRatio}]
```

```
{PlotRange -> {{-5, 5}, {-1.00678, 1.03866}}, AspectRatio ->  $\frac{2}{7}$ }
```

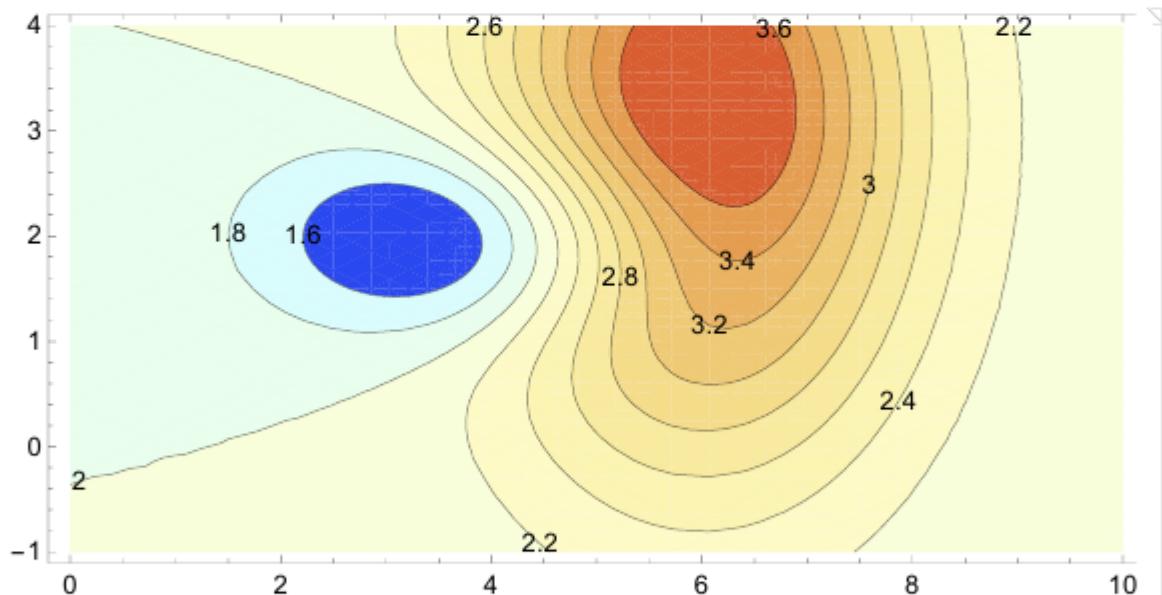
Отмеченное выше относительно функции `Options` действительно для всех видов графики (одномерной, многомерной, анимационной). Для примера приведем код формирования иллюстрации и вывода основных

опций двумерной графики. В качестве исходного рассматривается функция двух переменных  $fXY(x, y)$ , задаваемая выражением  $-e^{-(x/2-2)^2-(y-2)^2} + 2 e^{-(x/2-3)^2-(y/3-1)^2} + 2$ . В следующем примере функцией 2D графики ContourPlot в прямоугольной области  $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$  выводятся 11 изолиний, фон определён стандартной цветовой схемой LightTemperatureMap (обеспечивает опция ColorFunction→"LightTemperatureMap"), базовый стиль подписей – 14 пт (BaseStyle→14), соотношение масштабов при выводе изображения задается отношением  $(y_{\max}-y_{\min})/(x_{\max}-x_{\min})$ :

```
xmin := 0; xmax := 10;
ymin := -1; ymax := 4;
aspRat = (ymax - ymin) / (xmax - xmin);

fXY[x_, y_] := 2 - E^(-(x/2 - 2)^2 - (y - 2)^2) +
  2 E^(-(x/2 - 3)^2 - (y/3 - 1)^2);
```

```
grf2D = ContourPlot[fXY[x, y], {x, xmin, xmax},
{y, ymin, ymax}, Contours → 11, ContourLabels → All,
ColorFunction → "LightTemperatureMap",
BaseStyle → 14, AspectRatio → aspRat, ImageSize → 590]
```



Замечание. После включения в ядро системы нескольких функций и опций, которые первоначально были в пакетах / приложениях (в частности функции вывода легенды) не всегда функция Options работает правильно. Например, для рассмотренного на предыдущей лекции примера график трех указанных функций выводился с дополнениями легендой. Если опция PlotLegends включена, функция Options возвращает пустой список, приходится использовать AbsoluteOptions.

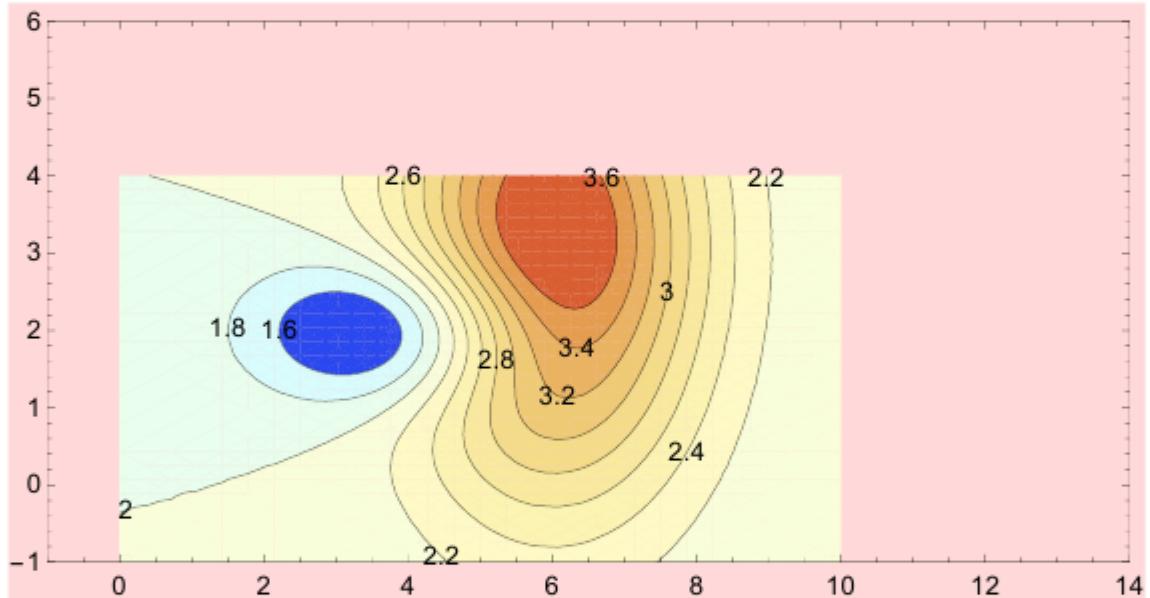
**Тестовые задания для самоконтроля**

*Внимание.* Код в большинстве секций не оптимальный, а рабочий и максимально простой, чтобы всем было понятно, что делается.

*Подсказки:* Аналитическое выражение, переменные, как в примере выше. Символов в ответе – 16.

*Упражнение 11.2a.* Что следует вписать вместо Yaaaa для получения показанного ниже графика

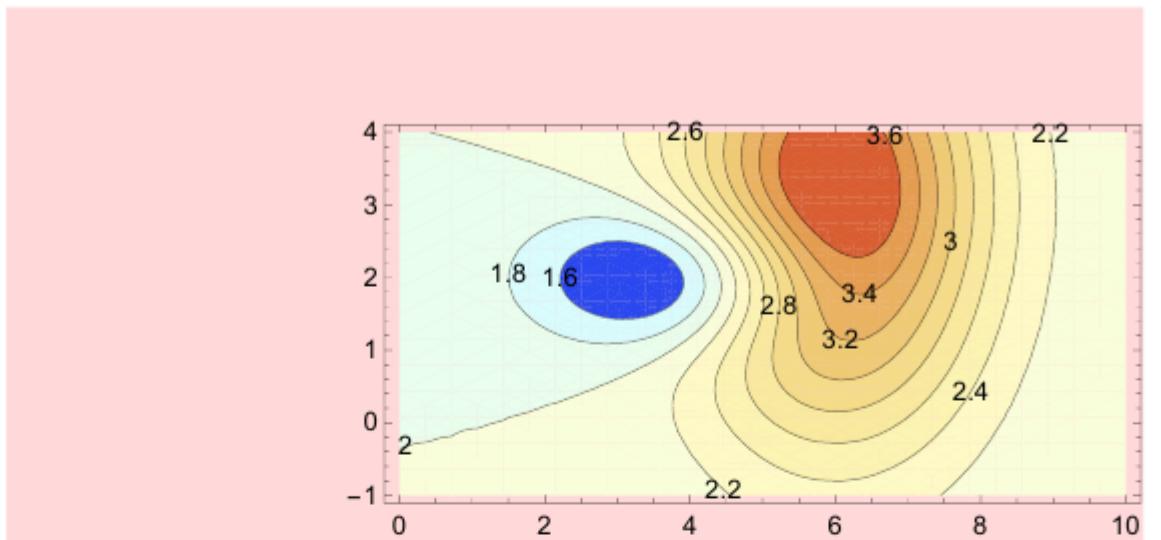
```
grf2Db = ContourPlot[fXY[x, y] {x, xmin, xmax} {y, ymin, ymax},
Contours -> 11, ContourLabels -> All,
ColorFunction -> "LightTemperatureMap",
BaseStyle -> 14, AspectRatio -> aspRat, ImageSize -> 590,
Yaaaa -> {{1, 4}, {0, 2}}, Background -> LightRed]
```



*Упражнение 11.2b.* Что следует вписать вместо Ybbbb для получения показанного ниже графика

*Подсказки:* Аналитическое выражение, переменные, как в примере выше. Символов в ответе – 10.

```
grf2Dc =
ContourPlot[fXY[x, y], {x, xmin, xmax} {y, ymin, ymax},
Contours -> 11, ContourLabels -> All,
ColorFunction -> "LightTemperatureMap",
BaseStyle -> 14, AspectRatio -> aspRat, ImageSize -> 590,
Ybbbb -> {{0.3, 1}, {0, 0.8}}, Background -> LightRed]
```



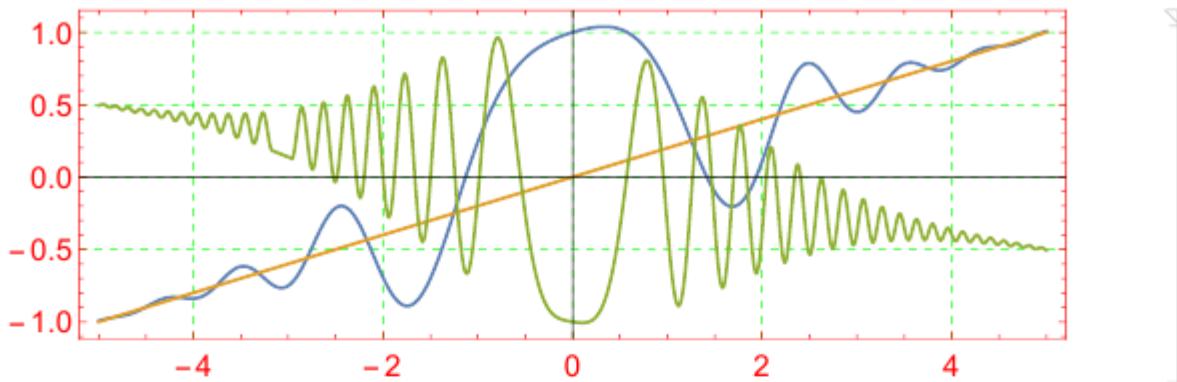
### О выводе и манипуляциях с объектами графики

*Mathematica* содержит много инструментов и поддерживает разные манипуляции с графическими объектами, перестроение и комбинирование графиков. При этом в первую очередь можно выделить следующие:

- изменение вида выводимого ранее сформированного графического объекта без повтора его формирования;
- компоновка и размещение в окне блокнота нескольких ранее сформированных графических объектов, причём они могут быть совершенно разных категорий, в частности:
  - вывод в одно окно нескольких разных, но одной размерности;
  - вывод объектов списком в последовательно размещаемые окна;
  - формирование окна, содержащего несколько других окон с графикой.

Поясним и приведем примеры изменения при выводе вида ранее сформированного графического объекта. Ниже – исходный код и результат, как система покажет график grf1D с следующими изменениями (сам объект повторно не формируется): соотношение линейных размеров осей ординат и абсцисс задано равным 1/3 (`AspectRatio→1/3`), задан вывод и цвет сетки зеленый (`GridLinesStyle→Directive[Green,Dashed]`), назначено окаймление поля графики красной рамкой (`Frame→True,FrameStyle→Red`), задан размер шрифта сопровождающих подписей – 16 (`BaseStyle →16`), изменен размер окна графика (`ImageSize→550`):

```
grf1Dn = Show[grf1D,
  AspectRatio -> 1 / 3, BaseStyle -> 16,
  GridLines -> Automatic,
  GridLinesStyle -> Directive[Green, Dashed] ,
  Frame -> True, FrameStyle -> Red,
  ImageSize -> 550]
```

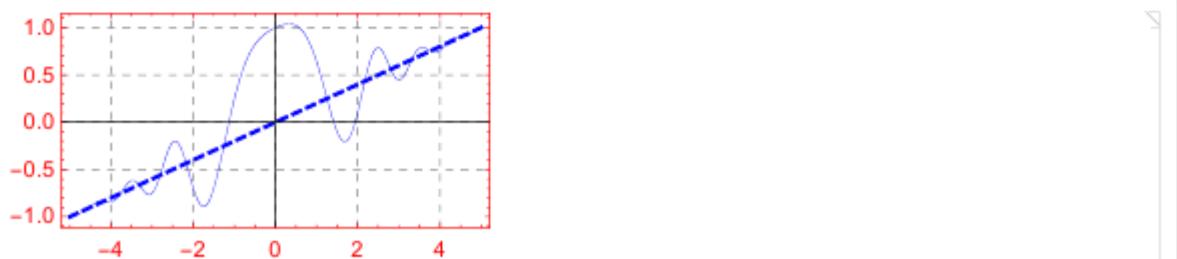


При построении графиков часто требуется изменение их вида, масштаба, отдельных атрибутов кривых. Этого можно достичь повторением вычислений, но тогда скорость работы с системой заметно снижается, что особенно проявляется при построении многомерных графиков. Для эффективного использования ресурсов системы следует использовать функции перестроения и вывода графиков, учитывающие, что узловые точки уже рассчитаны, основные опции уже заданы (по умолчанию). Например, можно использовать функцию-директиву `Show` в одном из следующих вариантов: `Show[plot]` – вывод графика `plot`, `Show[plot, option -> value]` – вывод графика `plot` с задаваемой уточняющей опцией, `Show[plot1, plot2,...]` – вывод нескольких графиков с наложением их друг на друга (графические объекты совмещаются путем послойного наложения либо путем взаимного внедрения в указываемом порядке).

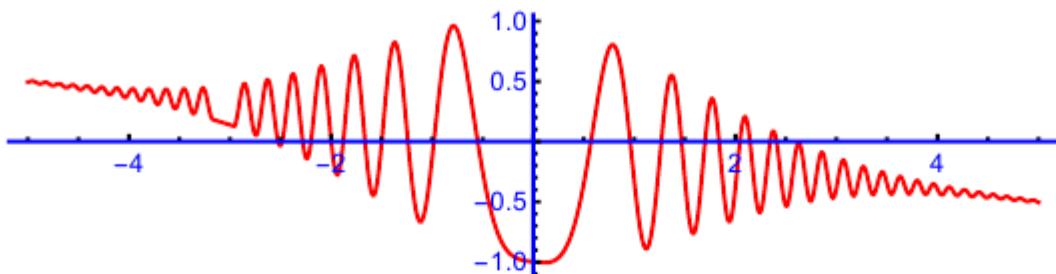
Рассмотрим примеры компоновки и размещения в одном окне секции нескольких графических объектов. Для упражнений используем сформированный объект одномерной графики `grf1D` и его части, которые специально оформим по разному и выведем в окна разных размеров:

Пример изменения стиля выводимого изображения (без повторного расчета) показан ниже с объектом двумерной графики `grf2D`.

```
grf1Da = Plot[{f1[x], f2[x]}, {x, -5, 5},
  PlotStyle -> {{Blue, Thin}, {Blue, Thick, Dashed}},
  AspectRatio -> 1/2, BaseStyle -> 12,
  Frame -> True, FrameStyle -> Red,
  GridLines -> Automatic,
  GridLinesStyle -> Directive[Gray, Dashed],
  ImageSize -> 250]
```

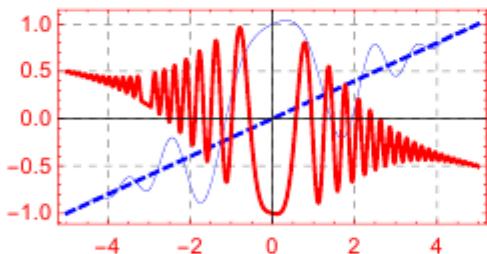


```
grf1Db = Plot[f3[x], {x, -5, 5}, PlotStyle ->
{Red, Thick}, AxesStyle -> Directive[Blue, Thick],
AspectRatio -> 1/4, BaseStyle -> 14, ImageSize -> 550]
```

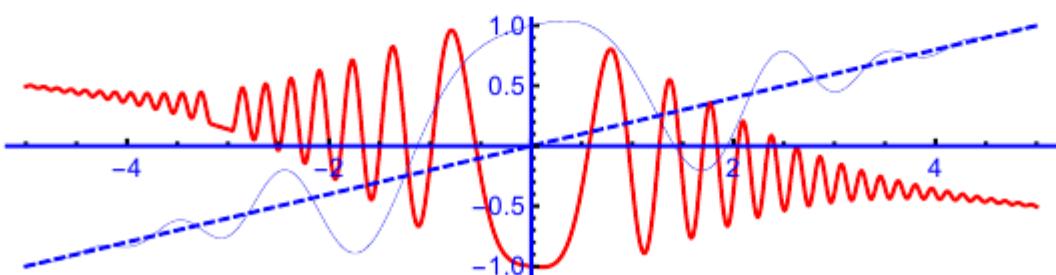


Два приведенных ниже примера `Show[grf1Da,grf1Db]` и `Show[grf1Db,grf1Da]` иллюстрируют, в том числе, приоритеты опций (стиля, размера) в зависимости от порядка перечисления выводимых объектов. Важно обратить внимание, какие атрибуты применяются к выводимому итоговому изображению. В секции ввода, когда при перечислении объектов вывода на первом месте (первый аргумент функции `Show`) стоит объект `grf1Da`, размер графика и основное оформление (окаймляющая рамка) определены установками этого объекта. В примере, когда на первом месте стоит объект `grf1Db`, приоритетными являются, опять-таки, опции первого аргумента `Show`, но это уже другие назначения (размер окна, оси координат и их подписи).

```
grf1Dc = Show[grf1Da, grf1Db]
```



```
grf1Dd = Show[grf1Db, grf1Da]
```



Подобный пример изменения стиля выводимого изображения (без повторного расчета) показан ниже с объектом двумерной графики `grf2D`. В оригинал объекта двумерной графики `grf2D` при выводе изменены

формат базового стиля подписей и область вывода (PlotRange -> {{1, 8}, {-1, 4}}, BaseStyle -> 20).

Во втором примере этой серии ниже (Show[grf2D, grf1Dd]) в одно окно выводятся несколько разных графиков: двумерный график изолиний (оригинальный, а не измененный), комбинированный (сводный) объект двух одномерных графиков – пример иллюстрирует возможности функции Show при выводе объектов, сформированных разными графическими функциями (функцией двумерной графики ContourPlot и одномерной Plot).

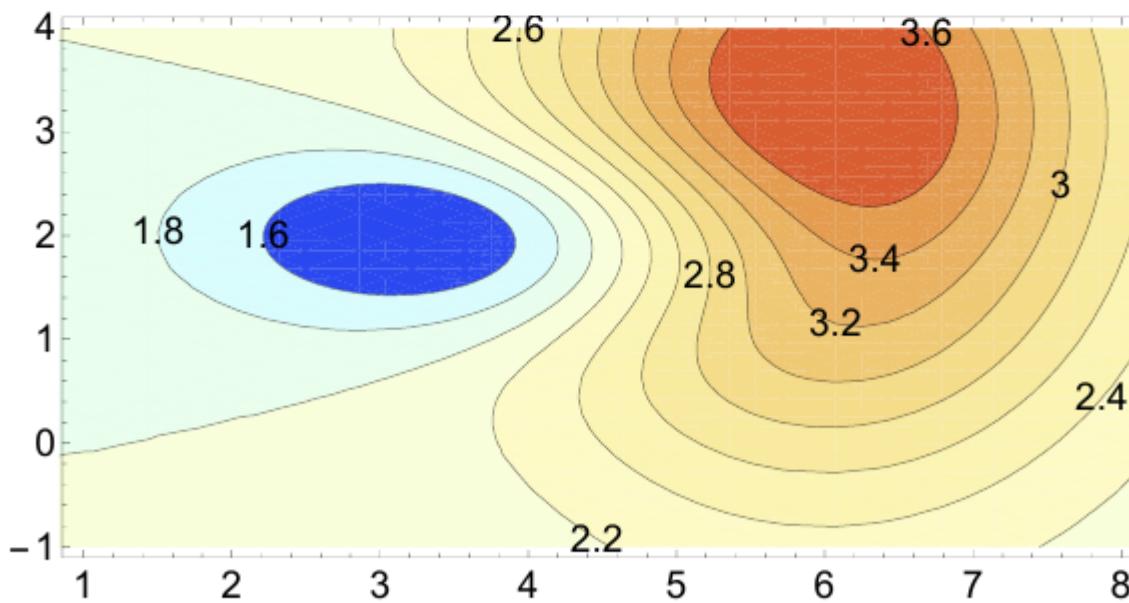
Полезно также обратить внимание на возможность присваивания графиков функций переменным (в наших примерах – grf1D\* и grf2D) в качестве значений. Такие переменные становятся графическими объектами, используемыми функцией Show для вывода на экран.

```
xmin := 0; xmax := 10;
ymin := -1; ymax := 4;
aspRat = (ymax - ymin) / (xmax - xmin);

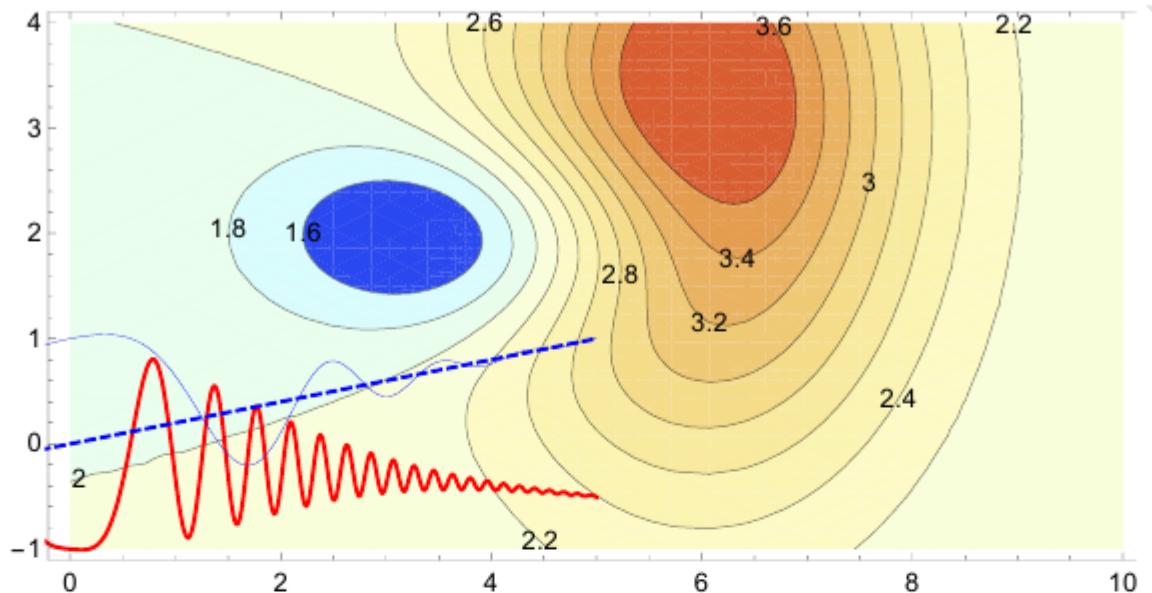
fXY[x_, y_] := 2 - E^(- (x / 2 - 2)^2 - (y - 2)^2) +
  2 E^(- (x / 2 - 3)^2 - (y / 3 - 1)^2);

grf2D =
  ContourPlot[fXY[x, y], {x, xmin, xmax}, {y, ymin, ymax},
    Contours -> 11, ContourLabels -> All,
    ColorFunction -> "LightTemperatureMap",
    BaseStyle -> 14,
    AspectRatio -> aspRat, ImageSize -> 590]
```

```
Show[grf2D, PlotRange -> {{1, 8}, {-1, 4}}, BaseStyle -> 20]
```



Show[grf2D, grf1Dd]



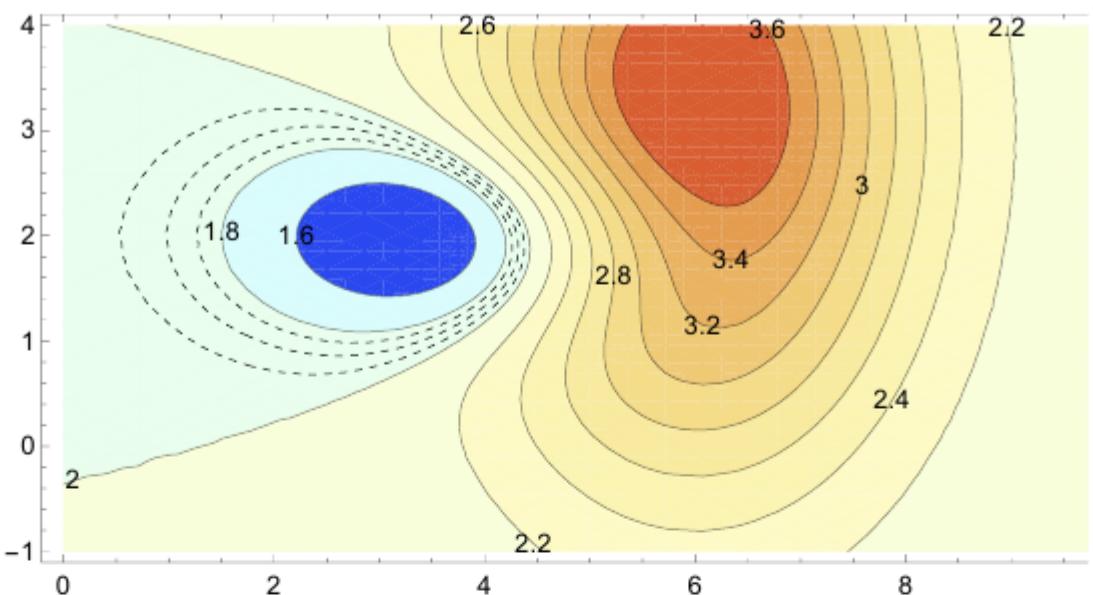
Завершая часть изложения по основам применения функции Show, отметим, что дополнительные возможности и особенности ее настроек будут даны еще много раз, в частности, в примерах объединения на одном графике нескольких графиков различных функций, скаттерограмм экспериментальных точек и графиков теоретической зависимости. Приведенные выше примеры акцентируют внимание, что при использовании Show надо побеспокоиться о выравнивании масштабов графиков, налагаемых друг на друга.

#### Тестовые задания для самоконтроля

*Упражнение 11.1с.* Что следует вписать вместо Ycccc для получения показанного ниже графика

*Подсказки:* Аналитическое выражение, переменные, как в примере выше. Чтобы были видны дополнительные изолинии (выводятся пунктиром), отключена заливка площади между изолиниями (оцвечивание контуров). Если заливку не отключать, график дополнительный перекрывает график-оригинал. Символов в ответе – 14.

```
grf2Dc =
ContourPlot[fXY[x, y], {x, xmin, xmax}, {y, ymin, ymax},
Contours -> {1.85, 1.9, 1.95}, ContourStyle -> Dashed,
Ycccc -> None,
AspectRatio -> aspRat, ImageSize -> 590];
Show[grf2D, grf2Dc]
```

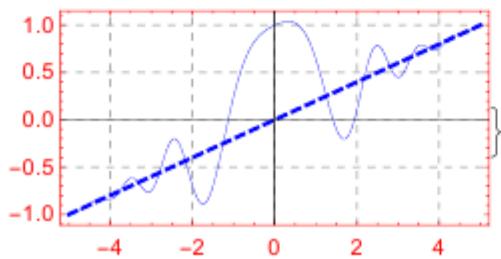
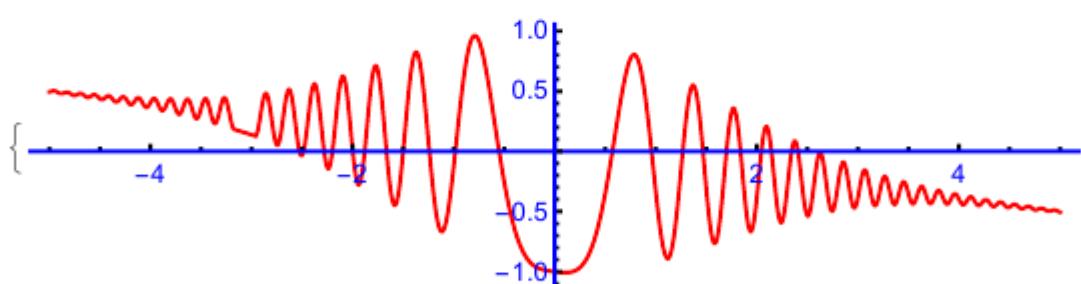


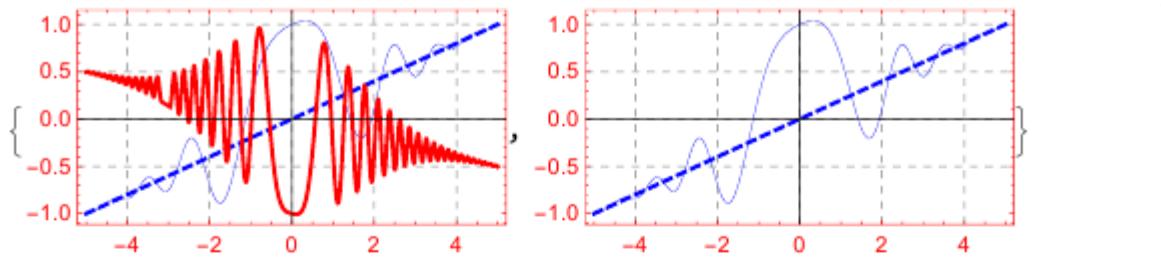
Рассмотрим примеры вывода графических объектов в несколько окон и соответствующие настройки вида.

Следующие секции ввода и результатов – иллюстрации простейшего вывода элементов списка, когда элементами являются графические объекты. Выводятся уже рассмотренные выше графики.

Следует обратить внимание, что размещение объектов осуществляется в последовательно открываемые окна в один или два ряда, что определяется шириной активного окна блокнота. Также заметим, что при таком выводе видим всегда сопутствующие спискам фигурные скобки и запятые.

```
{grf1Db, grf1Da}  
{grf1Dc, grf1Da}
```



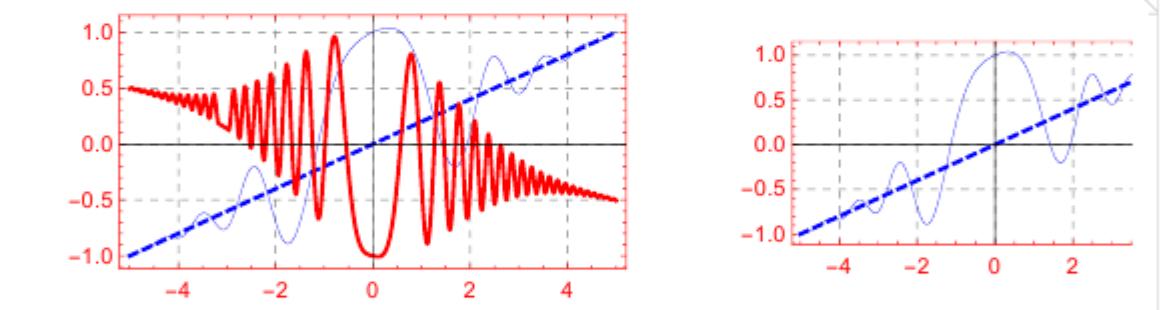


В секции выше вывод стандартный – открывающая и закрывающая фигурные скобки, разделителем объектов является запятая, где объекты не поместились в одну строку, второй график выводится под первым.

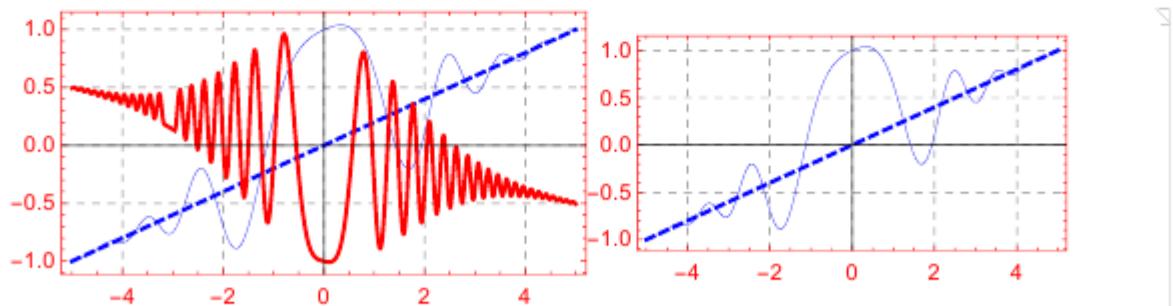
Следующие три пары секций иллюстрируют возможности применения функций `GraphicsRow` и `Row`. Заметим, что `GraphicsRow` (функция “строка графики”), вообще говоря, специализирована для вывода графических объектов в строку. Но надо заметить, что работает она (особенно, если графика дается с легендой) не всегда корректно, а настройки оставляют желать лучшего.

В приведенном ниже примере установленное по умолчанию расстояние (отступ) перед объектами оказывается неоправданно большим, и график “убегает” за границу поля, соответственно в окне блокнота включается и возможен скроллинг (горизонтальная прокрутка).

```
GraphicsRow [{Show[grf1Dc, ImageSize → 300], grf1Da}]
```



```
Row [{Show[grf1Dc, ImageSize → 300], grf1Da}]
```

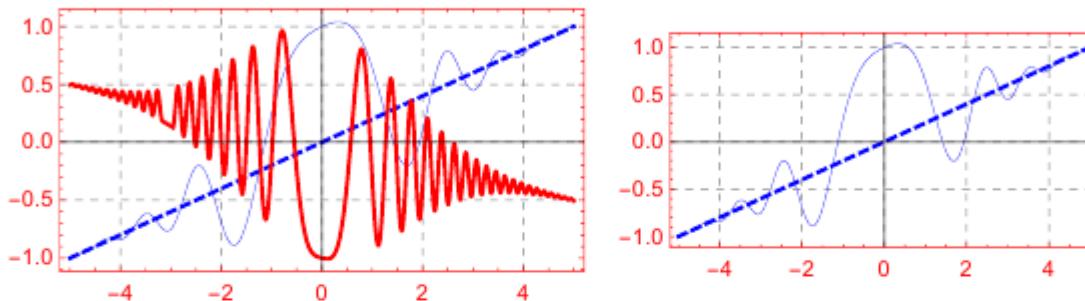


Наоборот, `Row` – базовая функция вывода элементов в ряд/ряды (таблицу) даёт желаемый результат. В частности, в `Row` удобно использовать опцию `Spacer` для задания промежутка между выводимыми объектами, а также

много других настроек отрисовки внешних и внутренних разделительных линий разметки таблицы (Alignment, Frame, FrameStyle, FrameMargins, ImageMargins, ImageSize, RoundingRadius):

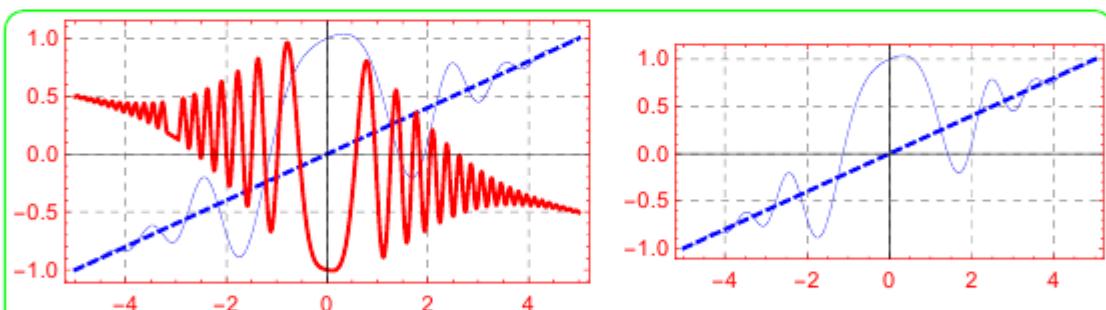
В результате вывода выше два графика позиционированы без дополнения интервала между ними, но при необходимости, используя опцию Spacer, требуемый отступ можно назначить, что иллюстрирует следующий пример:

```
Row [ {Show[grf1Dc, ImageSize → 300], grf1Da}, Spacer[10] ]
```



Для демонстрации дополнительных настроек Row отметим, что в примере ниже включена опция вывода окаймляющей рамки, для нее назначен стиль линии (FrameStyle→Green) и указано скругление углов (RoundingRadius→10).

```
Row [ {Show[grf1Dc, ImageSize → 300], grf1Da}, Spacer[10],  
Frame → True, FrameStyle → Green, RoundingRadius → 10]
```



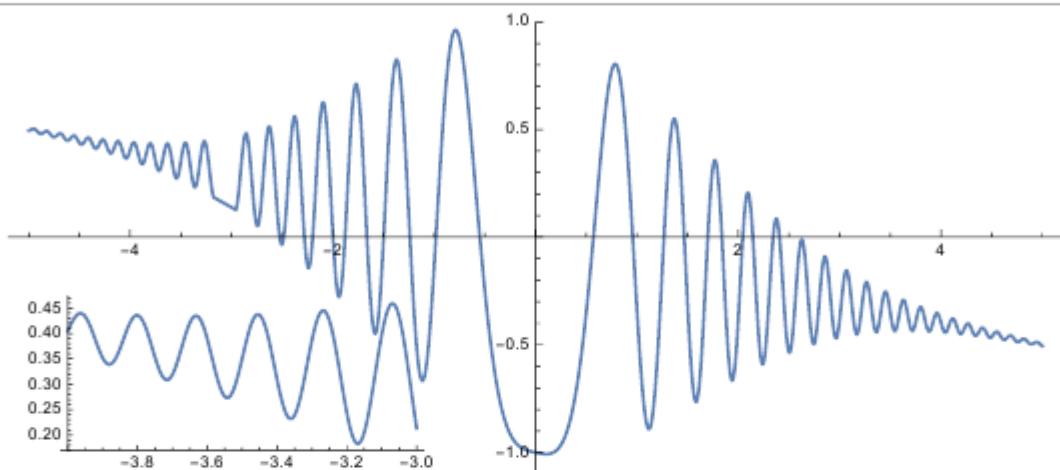
Также нужно упомянуть функцию GraphicsGrid (двумерная таблица графических объектов), которую по сути полностью заменяет Row, примеры применения будут даны в следующих темах.

#### Тестовые задания для самоконтроля

*Упражнение 11.1d.* Что следует вписать вместо Ydddd для получения показанных ниже графиков

*Подсказки:* В левом нижнем углу добавлен фрагмент изучаемого графика (вставка, выкопировка), который обеспечивает детализацию вида (увеличение в 4 раза). Символов в ответе – 5.

```
f3[x_] := (-E^(-0.2*x^2)) * Cos[5*x^2] - 0.1*x
grf1DdFrgm = Plot[f3[x], {x, -4, -3}, AspectRatio -> 3/7];
Plot[f3[x], {x, -5, 5}, AspectRatio -> 3/7, Epilog ->
Yddd[grf1DdFrgm, Scaled[{0.05, .05}], Scaled[{0, 0}], 4],
PlotRange -> {-1.1, 1}, PlotRangeClipping -> False,
ImageSize -> 550]
```



### Цвет и его задание в системе *Mathematica*

При формировании графических изображений в любой программной среде очень важной составляющей является работа с цветом и ее результаты. В частности, важно, какие и как компьютерные модели использовать, на сколько эффективны инструменты визуализации цветных изображений на экране и твердых копиях.

Система *Mathematica* поддерживает все общеизвестные компьютерные модели цвета, обеспечивает пользователей возможностями задавать цвета в различных цветовых пространствах, а также содержит большое число наборов предопределенных цветов и эстетически привлекательных цветовых гамм. Детали применения конкретных опций и директив приводятся ниже в примерах применения функций графики. Здесь отметим общее для всех, то, что можно применять практически к любым формируемым в системе объектам. Отдельно отметим, что язык системы *Mathematica* предоставляет уникальный и разносторонний подход цветового оформления и реализации прозрачности в графике, во всех формах визуализации (отображения).

При использовании графических функций системы цвета можно назначать именами. Наиболее часто используются следующие: Black (черный), White (белый), Red (красный), Green (зеленый), Blue (синий), Gray (серый), Cyan (сине-зелёный, цвет морской волны), Magenta (сиреневый), Yellow (желтый), Brown (коричневый), Orange (оранжевый), Pink (розовый), Purple (фиолетовый), LightRed (ярко-красный), LightGreen (светло-зеленый), LightBlue (ярко-синий), LightGray (светло-серый),

LightCyan (светло-голубой), LightMagenta (малиновый), LightYellow (светло-желтый), LightBrown (светло-коричневый), LightOrange (светло-оранжевый).

Производные цвета – цвета получаемые при использовании Lighter, Darker, Blend, ColorNegate (Colors). А именно:

- Lighter, Darker – делают цвет-аргумент более светлым, темным;
- Blend – обеспечивает смешивание цветов в указанных пропорциях;
- ColorNegate – инвертированный цвет (негатив).

Основные цветовые модели в системе поддерживаются графическими директивами RGBColor (аддитивная схема формирования цвета), CMYKColor (субтрактивная схема формирования цвета), Hue (интуитивные цветовые модели HSB, HLS, HSV), GrayLevel (схема оттенков серого). Аппаратно независимые модели цвета поддерживают: LABColor (цвет в координатах Lab), LCHColor, LUVColor.

Цветовые схемы (Color Schemes) – *Mathematica* предоставляет пользователям комплект разнообразных тщательно выбранных цветовых схем, которые могут использоваться/подключаться в функциях графики и значительно улучшать зрелищность визуализации. Например, можно использовать любую градиентную цветовую гамму из следующего списка: AlpineColors, ArmyColors, AtlanticColors, AuroraColors, AvocadoColors, BeachColors, CandyColors, CMYKColors, DeepSeaColors, FallColors, FruitPunchColors, IslandColors, BrassTones, BrownCyanTones, CherryTones, CoffeeTones, FuchsiaTones, GrayTones, GrayYellowTones, DarkTerrain, GreenBrownTerrain, BrightBands, Aquamarine, BlueGreenYellow, DarkRainbow, LightTemperatureMap, LakeColors, MintColors, NeonColors, PearlColors, PlumColors, RoseColors, SolarColors, SouthwestColors, StarryNightColors, SunsetColors, ThermometerColors, WatermelonColors, GreenPinkTones, PigeonTones, RedBlueTones, RustTones, SiennaTones, ValentineTones, LightTerrain, SandyTerrain, DarkBands, Pastel, Rainbow, TemperatureMap.

Приведем несколько простых иллюстраций.

На рисунке 11.1 в приведенном примере результатом выполнения In-секции и клика по кнопке 12 является визуализация 12 секторов. Код, который обеспечивает вывод и интерактивное задание указываемого пользователем числа секторов цветового круга:

```
Manipulate[
 Module[{t = 0}, Graphics[
  Table[{Hue[i/nColors], Disk[{0, 0}, 1, {Pi*i/(nColors/2),
    Pi*(i + 1)/(nColors/2)}]}, {i, 0, (nColors - 1)}],
  ImageSize -> 510, AspectRatio -> Automatic]],
 {nColors, {6, 12, 24, 48}}, LabelStyle -> 16]
```

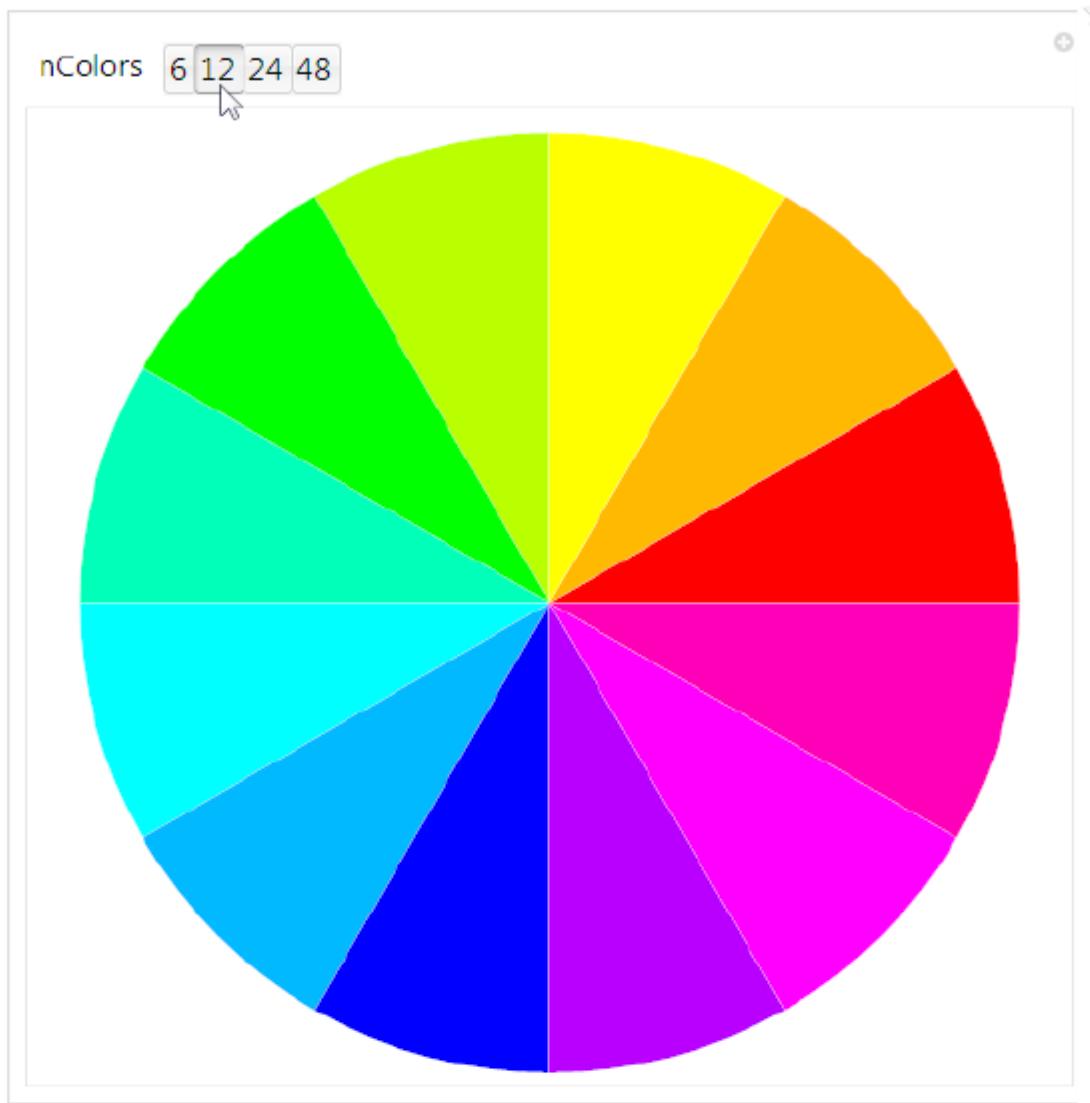


Рис. 11.1. Пример вывода 12 секторов модели цветового круга

Напомним, что, как и во всех других примерах настоящего учебного пособия, код не оптимальный, а рабочий и максимально простой, чтобы всем было понятно, что делается. Используемые в коде функции уже упоминались, графические примитивы (Disk), опции и директивы подробно будут описаны в следующих частях, особенности настройки функции Manipulate будут предметом обстоятельного рассмотрения.

Дополнительно на рисунке 11.2 приведены два результата вывода секторов модели цветового круга с размером окна ImageSize→250. В левом окне сформированы 6 цветов (3 основных цвета аддитивной и 3 основных цвета субтрактивной цветовых моделей). А именно, цвета: Red (красный), Yellow (желтый), Green (зеленый), Cyan (сине-зелёный), Blue (синий), Magenta (сиреневый). Заметим, что, как правило, графические программы позволяют комбинировать требуемый цвет из 256 оттенков красного, 256 оттенков зелёного и 256 оттенков синего. В твердых копиях, полученных

при печати, свет отражается от листа бумаги, поэтому в издательской продукции для графических изображений используется система субтрактивных цветов. И в ее основные цвета являются комбинациями цветов аддитивной системы: Yellow = Red + Green, Cyan = Green + Blue, Magenta = Blue + Red.

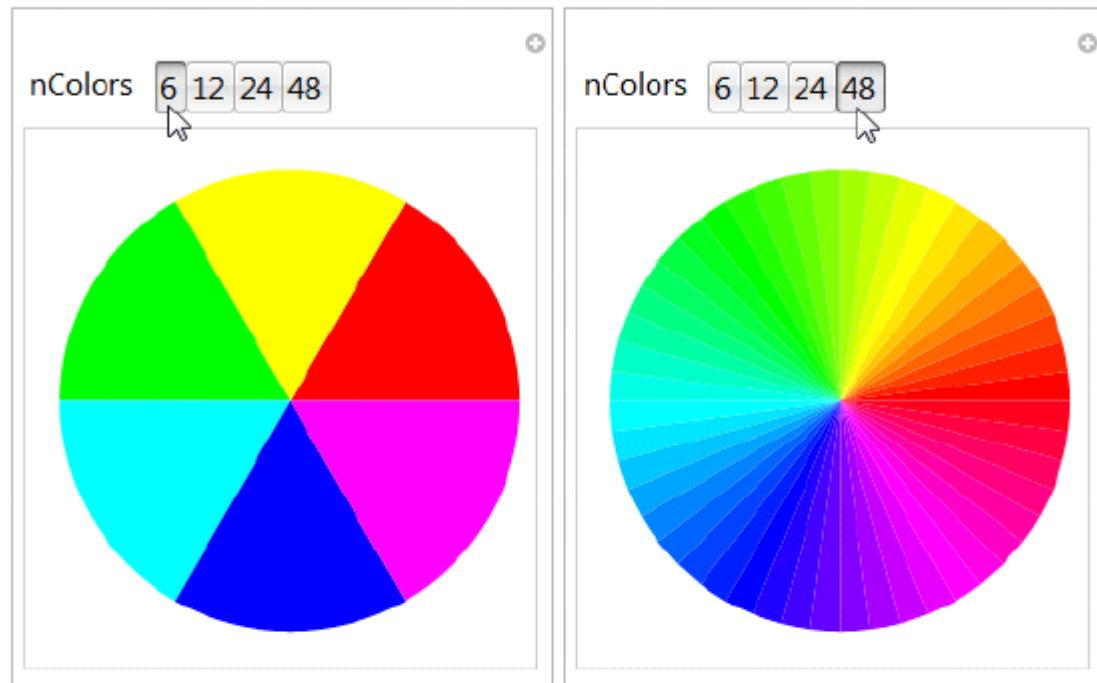
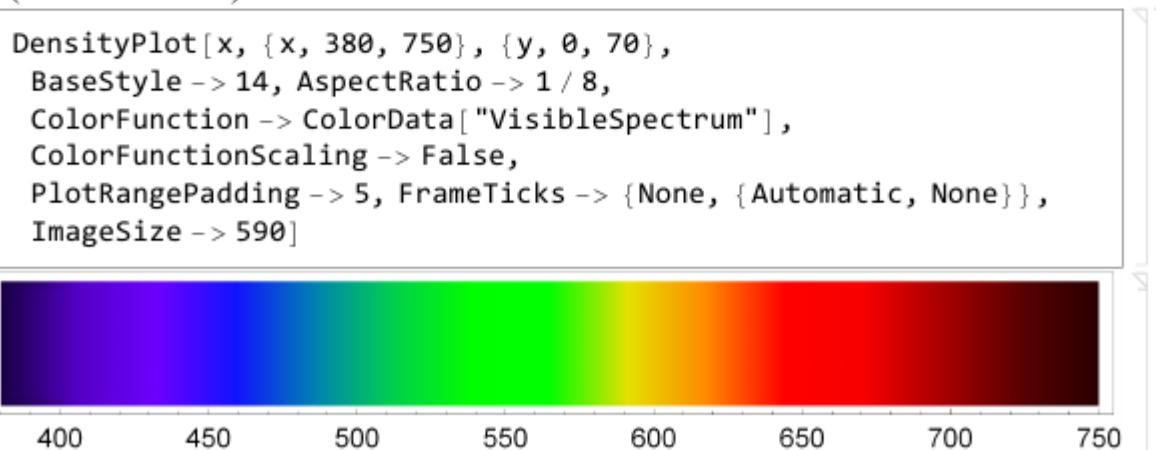


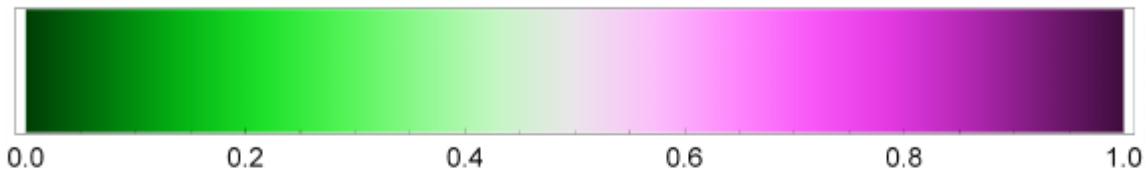
Рис. 11.2. Пример вывода 6 и 48 секторов модели цветового круга

Примеры ниже иллюстрируют возможности использования типовых схем непрерывных световых интервалов. Наиболее часто используются: "CMYKColors", "Pastel", "Rainbow", "TemperatureMap", "DeepSeaColors", "FallColors", "LakeColors", "MintColors", "NeonColors", "PearlColors", "SolarColors", "SunsetColors", "ThermometerColors". Ниже в примере используется VisibleSpectrum выводится видимый спектр, в примере GreenPinkTones – один из вариантов типовой градиентной цветовой гаммы (Color Schemes):

```
DensityPlot[x, {x, 380, 750}, {y, 0, 70},
BaseStyle -> 14, AspectRatio -> 1/8,
ColorFunction -> ColorData["VisibleSpectrum"],
ColorFunctionScaling -> False,
PlotRangePadding -> 5, FrameTicks -> {None, {Automatic, None}},
ImageSize -> 590]
```

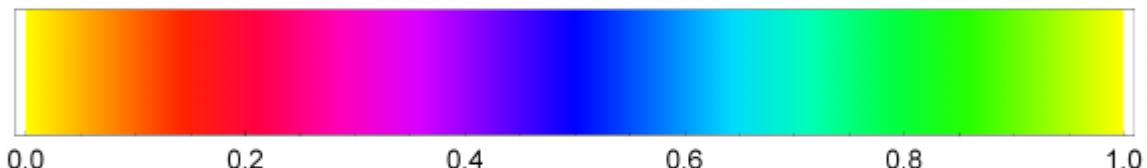


```
DensityPlot[x, {x, 0, 1}, {y, 0, 1}, BaseStyle -> 14,
ColorFunction -> ColorData["GreenPinkTones"],
AspectRatio -> 1/9, PlotRangePadding -> 0.01,
FrameTicks -> {None, {Automatic, None}}, ImageSize -> 590]
```

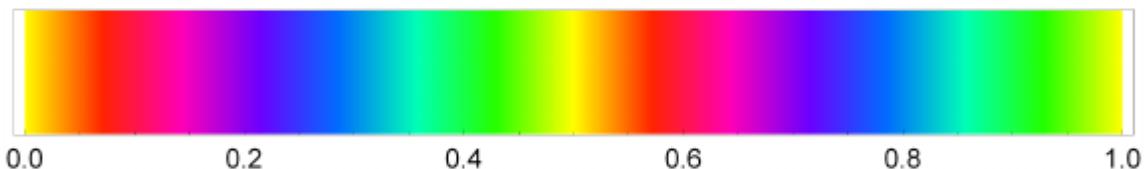


Примеры задания непрерывных цветовых схем пользователем, варианты: цвета всего цветового круга, они же дважды, цвета половины круга:

```
DensityPlot[x, {x, 0, 1}, {y, 0, 1},
BaseStyle -> 14, ColorFunction -> (Hue[60/360 - #] &),
AspectRatio -> 1/9, PlotRangePadding -> 0.01,
FrameTicks -> {None, {Automatic, None}}, ImageSize -> 590]
```



```
DensityPlot[x, {x, 0, 1}, {y, 0, 1}, BaseStyle -> 14,
ColorFunction -> (Hue[60/360 - 2 * #] &),
AspectRatio -> 1/9, PlotRangePadding -> 0.01,
FrameTicks -> {None, {Automatic, None}}, ImageSize -> 590]
```



```
DensityPlot[x, {x, 0, 1}, {y, 0, 1}, BaseStyle -> 14,
ColorFunction -> (Hue[60/360 - #/2] &),
AspectRatio -> 1/9, PlotRangePadding -> 0.01,
FrameTicks -> {None, {Automatic, None}}, ImageSize -> 590]
```



Примеры задания непрерывных цветовых схем пользователем в варианте оттенки конкретного цвета:

```
DensityPlot[x, {x, 0, 1}, {y, 0, 1}, BaseStyle -> 14,
ColorFunction -> (Lighter[Purple, 0.9 * #] &),
AspectRatio -> 1 / 15, PlotRangePadding -> 0.01,
FrameTicks -> {None, {Automatic, None}}, ImageSize -> 590]
```



```
DensityPlot[x, {x, 0, 1}, {y, 0, 1}, BaseStyle -> 14,
ColorFunction -> (Lighter[Purple, (1 - 0.9 * #)] &),
AspectRatio -> 1 / 15, PlotRangePadding -> 0.01,
FrameTicks -> {None, {Automatic, None}}, ImageSize -> 590]
```



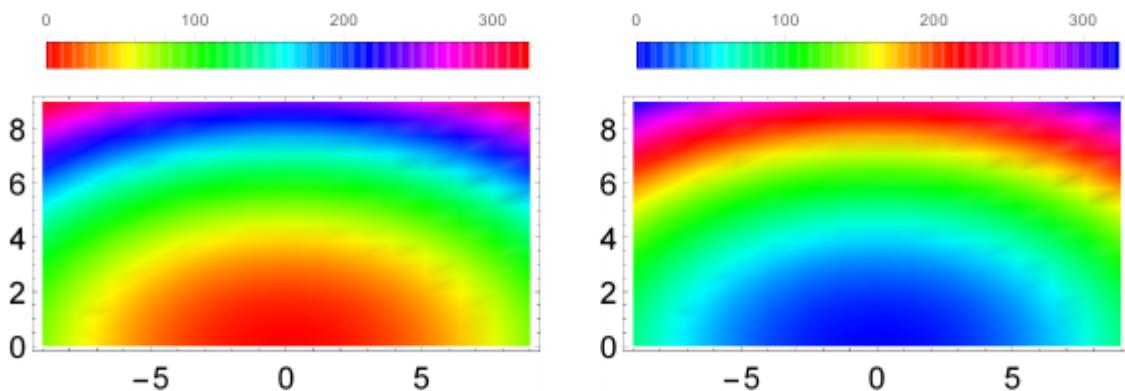
### Тестовые задания для самоконтроля

*Упражнение 11.1e.* Что следует вписать вместо Yeeee для получения показанного ниже графика справа

*Подсказки:* Символов в ответе – 13, в том числе есть 2/3.

```
grf2Dd = DensityPlot[x^2 + 3 * y^2, {x, -9, 9}, {y, 0, 9},
BaseStyle -> 16, AspectRatio -> 1 / 2,
PlotLegends -> Placed[Automatic, Above],
ColorFunction -> Hue, ImageSize -> 270];
grf2De = DensityPlot[x^2 + 3 * y^2, {x, -9, 9}, {y, 0, 9},
BaseStyle -> 16, AspectRatio -> 1 / 2,
PlotLegends -> Placed[Automatic, Above],
ColorFunction -> Yeeee, ImageSize -> 270];
```

```
Row [{grf2Dd, grf2De}, Spacer[5]]
```



Инструменты системы *Mathematica* предоставляют при работе с цветом возможности использовать специальные графические директивы и опции:

- EdgeForm, FaceForm – стиль ребер, граней; директивы, которые позволяют назначить стиль отрисовки рёбер, граней лицевых поверхностей трехмерных полигонов; позволяют указать соответствующие графические директивы отрисовки;
- Opacity – позволяет назначить уровень прозрачности двумерных и трехмерных графических объектов, делать указываемые выводимые объекты частично прозрачными, чтобы иметь возможность сквозь них наблюдать другие, когда требуется обзор нескольких разных компонентов графика;
- Specularity – зеркальность; директива, которая позволяет задавать показатель зеркального отражения, параметр светоотражающей способности поверхности; важно для достижения фотorealизма; полезна для управления рассеянным отражением света на матовых поверхностях и зеркального отражения на зеркально подобных поверхностях;
- Glow – свечение; директива, которая позволяет выводить обработанный цвет трехмерной поверхности, указать, как расцвечивать 3D поверхности независимо от моделей освещения и отражения, по существу имитируя свечение поверхности заданным цветом;
- Lighting – опция 3D графики, которая используется для установки следует ли дополнительно моделировать освещение в трехмерных изображениях; позволяет управлять рассеянным отражением света на матовых и зеркальным отражением на зеркально подобных поверхностях; может сочетаться с другими функциями и опциями для получения качественной графики индивидуального стиля, при управлении окончательным видом 3D поверхности.

Рассмотренные ниже примеры поясняют несколько основных возможностей использования инструментов управления освещением при формировании и выводе изображений 3D объектов. Подробно соответствующие опции и директивы рассматриваются в темах визуализации функций двух переменных и многомерных данных.

Приведены код и два результата вывода изображений поверхности, которую иллюстрировали изолиниями на рисунке 10.9 и в вариантах 3D визуализации с разными ракурсами осмотра на рисунках 10.12 и 10.13. В данном случае в примерах на обоих рисунках приведены результаты, когда включены опции: "сетки на гранях" (FaceGrids->All), "сетка" (Mesh->8), "функция построения сеточных линий" (MeshFunctions->{#3&}) – интервальная заливка цветом по третьей координате, по уровням высоты поверхности), "закраска между сеточными кривыми" (MeshShading) цветами, которые перечислены в списке ({Brown, Red, Pink, Magenta, LightRed, Orange, LightBrown, Yellow, LightMagenta}). Также включена

заливка цветом по периметру от границ поверхности до горизонтальной плоскости нулевого уровня (Filling->Bottom).

Объект grf3Da выводится с установками освещения, задаваемыми по умолчанию.

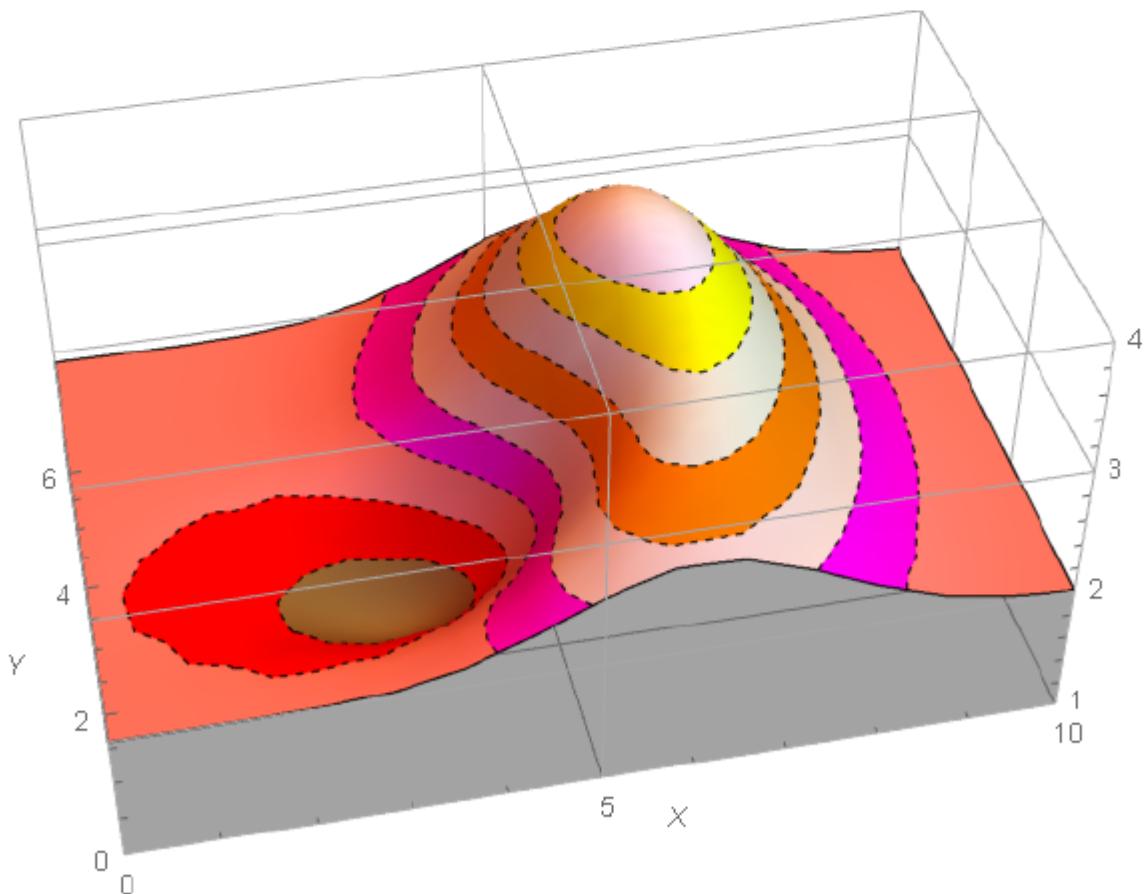
```

xmin := 0; xmax := 10;
ymin := 0; ymax := 6;
zWINmin = 1; zWINmax = 4; zScale = 5;
zMeshShdng = {Brown, Red, Pink, Magenta,
    LightRed, Orange, LightBrown, Yellow, LightMagenta};

fXY[x_, y_] := 2 - E^(-(x/2 - 2)^2 - (y - 2)^2) +
    2 E^(-(x/2 - 3)^2 - (y/3 - 1)^2);

grf3Da = Plot3D[fXY[x, y],
    {x, xmin, xmax}, {y, ymin, ymax},
    PlotRange -> {{xmin, xmax}, {ymin, ymax}, {zWINmin, zWINmax}},
    AxesLabel -> {X, Y, Z}, BaseStyle -> {Black, 14},
    FaceGrids -> All, Filling -> Bottom,
    BoxRatios -> {xmax - xmin, ymax - ymin, zScale},
    MeshFunctions -> {#3 &}, Mesh -> 8,
    MeshStyle -> {Dashed, Black}, MeshShading -> zMeshShdng,
    ViewPoint -> {-0.5, -2.3, 2.}, ImageSize -> 600]

```

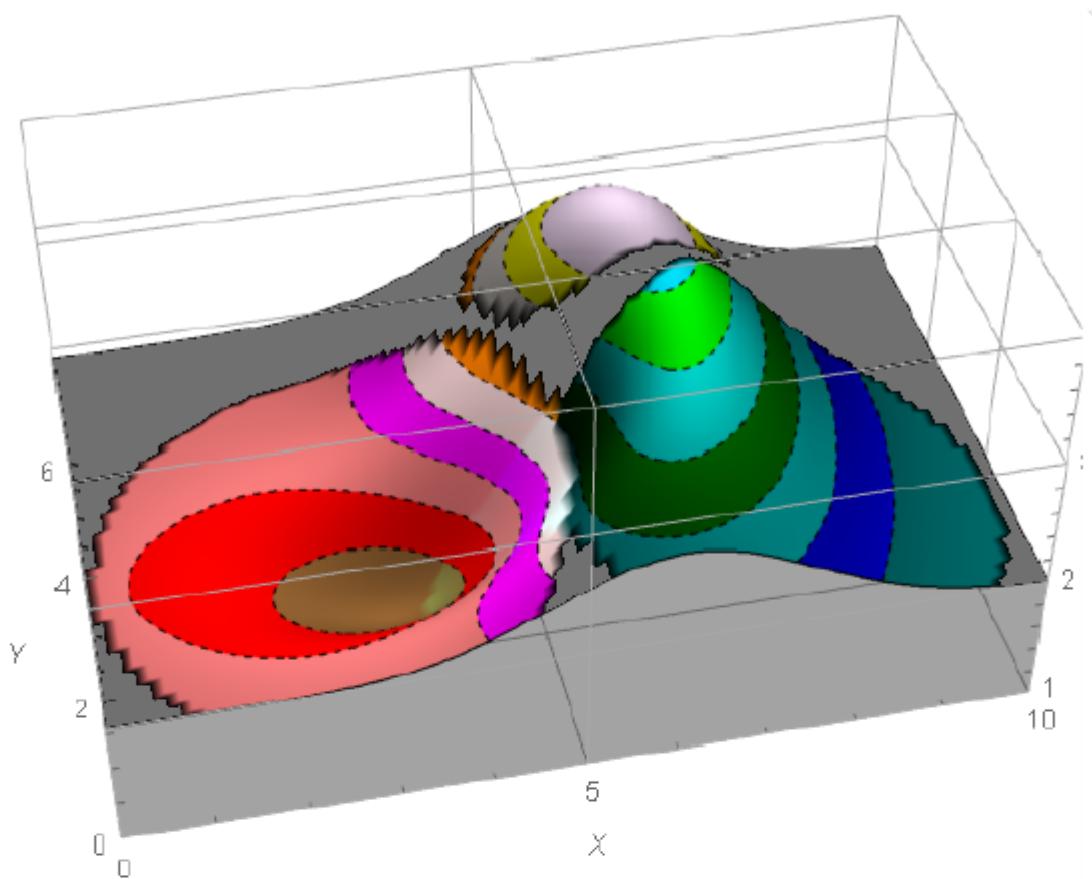


В изображении grf3Db иллюстрируется один из базовых вариантов освещения – источник типа прожектор, при задании разных параметров света и места положения источника.

Заметим, что средствами *Mathematica* поддерживается применение моделей освещения: рассеянный свет (*Ambient*, используется для равномерного освещения поверхностей всех объектов и создания общего светового фона сцены), направленный свет (*Directional*, испускает лучи в определенном направлении, равномерно освещая сцену), точечный источник света (*Point*, распространяет лучи во всех направлениях из определенной указываемой точки пространства; аналогом в реальном мире является обычная электрическая лампочка или свеча), источники света типа прожектор (*Spot*, используются для имитации освещения с помощью прожектора; включает несколько настроек, отдельные поясняются в приведенном примере).

В рассматриваемом примере визуализации эффектов освещения разными источниками разных частей поверхности задаются параметры трех размещенных в разных точках источников типа прожектор. Два источника размещены примерно над точками локальных экстремумов поверхности (условно – яма, холм). Для всех имитируется, что они светят сфокусированными пучками вертикально вниз, свет – конусообразный луч, сгенерированный от места расположения источника света, который является верхушкой конуса, в заданное направление. Два источника светят белым цветом, один зелёным, угол сектора задается так, чтобы пятно света было на определенном участке. Углы подобраны, чтобы наложение было минимальным. Обратите внимание, что четко просматривается разница в видах зон одного цвета в оригинале (Yellow, LightMagenta в куполе холма) и их разная окраска при освещении прожекторами белого и зеленого света.

```
grf3Db =
Plot3D[fXY[x, y], {x, xmin, xmax}, {y, ymin, ymax},
PlotRange -> {{xmin, xmax}, {ymin, ymax}, {zWINmin, zWINmax}},
BaseStyle -> {Black, 14},
AxesLabel -> {X, Y, Z}, FaceGrids -> All,
Filling -> Bottom,
BoxRatios -> {xmax - xmin, ymax - ymin, zScale},
MeshFunctions -> {#3 &}, Mesh -> 8,
MeshStyle -> {Dashed, Black}, MeshShading -> zMeshShdng,
PlotPoints -> 80,
Lighting -> {
  {"Spot", White, {{3, 2, 5}, {3, 2, 0}}, Pi/6},
  {"Spot", White, {{6, 4, 8}, {6, 4, 2}}, Pi/18},
  {"Spot", Cyan, {{7, 1.5, 5}, {7, 1.5, 0}}, Pi/6}},
ViewPoint -> {-0.5, -2.3, 2.}, ImageSize -> 600]
```



Примеры применения изложенного материала при решении практических и научных задач можно прочитать, в частности, в статьях, приведенных ниже.

#### Рекомендуемая литература

1. Таранчук, В.Б. О создании интерактивных образовательных ресурсов с использованием технологий Wolfram / В.Б. Таранчук // Информатизация образования: – 2014. – № 1 (73). – С. 78–89.
2. Таранчук, В.Б. Функции и инструменты подготовки в системе Mathematica интерактивных графических приложений / В.Б. Таранчук, В.А. Кулинкович // Вестн. Института современных знаний: – 2015. – № 2 (63). – С. 75–82.
3. Таранчук, В.Б. Особенности функционального программирования интерактивных графических приложений / В.Б. Таранчук // Вестник Самарского государственного университета. Естественнонаучная серия, раздел Математика: – 2015. – № 6 (128). – С. 178–189.
4. Taranchuk, V.B. Development of interactive teaching materials for computer mechanics. / V.B. Taranchuk, M.A. Zhuravkov // Vestnik BGU. Ser. 1, Fiz. Mat. Inform. – 2016. – No. 3. – P. 97–107 (in Engl.)

## ОГЛАВЛЕНИЕ

<b>ПРЕДИСЛОВИЕ .....</b>	<b>3</b>
<b>СПИСОК ОСНОВНЫХ СОКРАЩЕНИЙ .....</b>	<b>4</b>
<b>Содержание лекции и занятия 10, учебные материалы .....</b>	<b>5</b>
О графических объектах системы <i>Mathematica</i> .....	5
Типы (категории) графических объектов <i>Mathematica</i> .....	6
Графики аналитически задаваемых функций одной переменной .....	6
Тестовые задания для самоконтроля .....	8
Визуализация массивов одномерных данных.....	10
Тестовые задания для самоконтроля .....	11
Гистограммы, столбиковые, круговые, секторные диаграммы .....	14
Тестовые задания для самоконтроля .....	15
Графики по точкам (с временной переменной).....	16
Тестовые задания для самоконтроля .....	17
Финансовые диаграммы.....	17
Тестовые задания для самоконтроля .....	18
Специальные функции графики пакетов-приложений.....	19
Средства 2D графики аналитически задаваемых функций .....	20
Тестовые задания для самоконтроля .....	21
Функции интерполяции и визуализации 2D графики массивов данных .....	22
Функции формирования и вывода объектов 2D графики .....	23
Тестовые задания для самоконтроля .....	24
Средства 3D графики аналитически задаваемых функций .....	25
Функции интерполяции и визуализации 3D графики массивов данных .....	27
Функции сбора, формирования и вывода объектов 3D графики, географические визуализации и вычисления .....	28
Рекомендуемая литература .....	29
<b>Содержание лекции и занятия 11, учебные материалы .....</b>	<b>30</b>
<i>Mathematica</i> . Общие правила работы с графикой .....	30
Тестовые задания для самоконтроля .....	34
О выводе и манипуляциях с объектами графики .....	35
Тестовые задания для самоконтроля .....	39
Примеры применения функций Show, GraphicsRow .....	40
Тестовые задания для самоконтроля .....	42
Цвет и его задание в системе <i>Mathematica</i> .....	43
Тестовые задания для самоконтроля .....	48
Специальные графические директивы и опции (Opacity, Lighting) .....	49
Рекомендуемая литература .....	52

Учебное издание

**Таранчук Валерий Борисович**

**ВВЕДЕНИЕ В ГРАФИКУ  
СИСТЕМЫ *MATHEMATICA***

**Учебные материалы для студентов  
факультета прикладной математики  
и информатики**

В авторской редакции

Ответственный за выпуск *В. Б. Таранчук*

Подписано в печать 27.11.2017. Формат 60×84/16. Бумага офсетная.  
Усл. печ. л. 3,25 Уч.-изд. л. 2,6. Тираж 50 экз. Заказ

Белорусский государственный университет.

Свидетельство о государственной регистрации издателя, изготовителя,  
распространителя печатных изданий № 1/270 от 03.04.2014.

Пр. Независимости, 4, 220030, Минск.

Отпечатано на копировально-множительной технике  
факультета прикладной математики и информатики  
Белорусского государственного университета.  
Пр. Независимости, 4, 220030, Минск.