Программирование графики (окончание)

Учебная практика

мехмат, III курс, кафедра ИВЭ

Определение параметров команд и стили

Места определения параметров команд

Параметры могут указываться:

- ✔ Непосредственно в командах.

Пример

```
\draw [->] (0, 0) -- (1, 0);
```

- ✔ В областях, в частности, в окружении `tikzpicture`.

Пример

```
{ [->] %
  \draw (0, 0) -- (1, 0);
  \draw (0, 1) -- (1, 1);
} %
```

- ✔ В *стилях*.

Стиль

Стиль: именованный набор настроек.

⟨*имя_стиля*⟩**/.style** = ⟨*параметры*⟩

Определяет или переопределяет стиль с заданным именем.

⟨*имя_стиля*⟩**/.append style** = ⟨*параметры*⟩

Дополняет стиль с заданным именем.

⟨*имя_стиля*⟩

Исполняет настройки, записанные в заданном стиле.

Замечание: при помощи переопределений и дополнений стилей можно включать/отключать цвета, эффекты теней и т. д. в разных версиях документа, создаваемого на основе пакета `beamer` (презентация, статья, раздаточный материал и т. д.) △

Пример определения стиля


```
\usepackage {tikz}

\usetikzlibrary {shadows}

\tikzset %
{ %
  box/.style = %
  { %
    shape = rectangle, %
    draw, %
    % ...
  } %
} %
```

```
\mode <beamer> %
{ %
  \tikzset %
  { %
    box/.append style = %
    { %
      draw = structure, %
      fill = %
        structure!30!white, %
      drop shadow %
    } %
  } %
} %
```

Пример использования стиля


```
\begin {frame} {Рисунок}
  \begin {figure}
    \begin {tikzpicture}
      \node [box] (A) {Объект};
      % ...
    \end {tikzpicture}
    \caption {диаграмма}
  \end {figure}
\end {frame}
```

Предопределённые стили

Таблица 1: основные предопределённые стили

Стиль	Исполняется в начале каждой команды/окружения/ключа
every picture	tikz, tikzpicture
every scope	scope
every path	path
every node	node
every edge	edge
every matrix	matrix
every cell	Ячейка матрицы
every on chain	on chain
every join	join
	...

Пример использования предопределённых стилей


```
\tikzset %
{ %
  every picture/.style = %
  { %
    thick, %
  } %
} %
```

```
\begin {tikzpicture} %
  [ %
    every node/.style = %
    { %
      draw, %
      font = \large %
    } %
  ] %
  \node %
  [ %
    fill = gray, %
    font = \small %
  ] %
    {Text};
\end {tikzpicture}
```

Стиль вершин матрицы

nodes = ⟨*параметры*⟩

Дополняет стиль всех вершин, находящихся в текущей матрице. Сокращённая запись для `every node/.append style` = ⟨*параметры*⟩.

cells = ⟨*параметры*⟩

Аналогично, для ячеек: `every cell/.append style` = ⟨*параметры*⟩.

row ⟨*номер*⟩ = ⟨*параметры*⟩

column ⟨*номер*⟩ = ⟨*параметры*⟩

row ⟨*строка*⟩ **column** ⟨*столбец*⟩ = ⟨*параметры*⟩

Аналогично, для строки/столбца/ячейки с заданным номером.

Стили ячеек матрицы

every odd/even row/column

Устанавливают стиль для ячеек нечётной/чётной строки/столбца.

Пример

```
\matrix [matrix of math nodes, %
  every even row/.append style =
    {nodes = {fill = orange}}, %
  every even column/.append style =
    {text = green!50!black}] %
{ %
  a_{1, 1} & a_{1, 2} & %
  a_{1, 3} & a_{1, 4} \\
  % ...
};%
```

$$\begin{matrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{matrix}$$

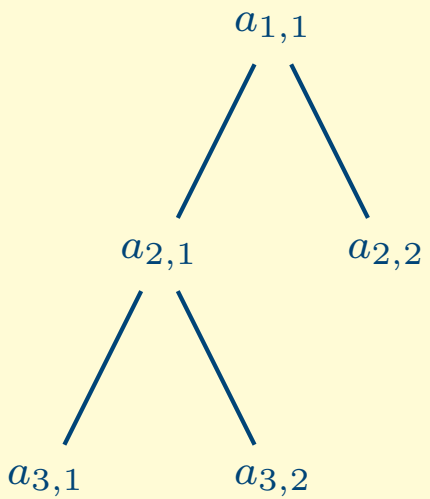
Отображение древовидных структур

Определение деревьев в коде

- ✔ При помощи синтаксиса вершины можно определить подчинённую ей иерархию вершин.
- ✔ Пакет `tikz-qtree` предоставляет более компактный по сравнению с TikZ синтаксис для определения деревьев.

Пример

```
\node {$a_{1, 1}$} %
  child %
  { %
    node {$a_{2, 1}$} %
      child { node {$a_{3, 1}$} } %
      child { node {$a_{3, 2}$} } %
  } %
  child { node {$a_{2, 2}$} };%
```



Синтаксис операции пути `child`

child [⟨*параметры*⟩] **foreach** ⟨*переменные*⟩ **in** {⟨*значения*⟩} {⟨*дочерний_путь*⟩}

Операция должна следовать вдоль пути непосредственно за законченной операцией **node** или **child**. Выполняет отображение всех иерархически подчинённых путей, соединяя их дугами. При выводе каждого дочернего пути начало координат устанавливается в вычисленную позицию этого пути. Если дочерний путь пропущен, или в нём отсутствует вершина, в соответствующую позицию автоматически вставляется пустая вершина формы **coordinate**.

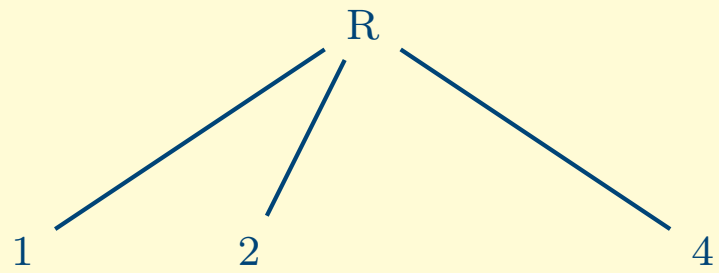
Пропуск дочерних вершин

missing = true, false (по умолчанию: **false**)

Указывается в параметрах к child. Означает, что соответствующая дочерняя вершина будет учтена при вычислении позиции остальных, но операции отображения её самой исполнены не будут.

Пример

```
\node {R} %
  child { node {1} } %
  child { node {2} } %
  child [missing] %
    { node {3} } %
  child { node {4} };
```



Расположение вершин дерева

level distance = ⟨*длина*⟩ (по умолчанию: 15mm)

Определяет расстояние от родителя до прямой, на которой располагаются дочерние узлы.

sibling distance = ⟨*длина*⟩ (по умолчанию: 15mm)

Определяет расстояние между якорями дочерних узлов.

growth parent anchor = ⟨*имя_якоря*⟩ (по умолчанию: **center**)

Определяет якорь родительской вершины, относительно которого будет вычислены позиции дочерних.

grow = ⟨*направление*⟩ (по умолчанию: **down**)

Определяет направление «роста» дерева. Может быть целым числом (градусы) или таким значением, как **up**, **south west** и т. д.

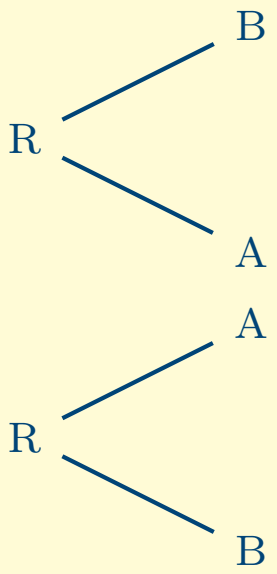
Расположение вершин дерева (окончание)

grow' = ⟨*направление*⟩ (по умолчанию: **down**)

Аналогично, только дочерние вершины располагаются в обратном порядке.

Пример

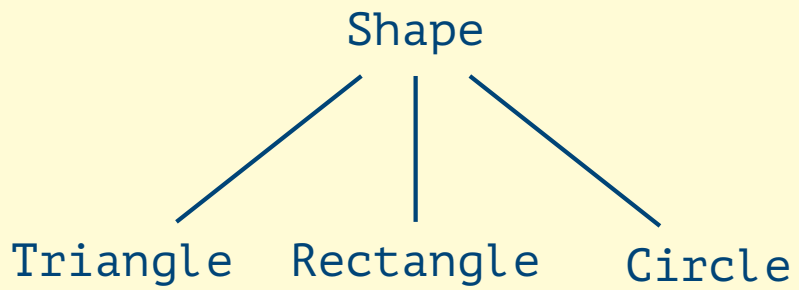
```
\node (A) {R} %
  [grow = right] %
  child { node {A} } %
  child { node {B} };

\node [below = 15mm of A] {R} %
  [grow' = right] %
  child { node {A} } %
  child { node {B} };
```



Пример определения дочерних узлов при помощи операции foreach

Пример

```
\node [font = \ttfamily] {Shape} %
  [font = \ttfamily, %
  level distance = 4em, %
  sibling distance = width("Rectangle") + 1em] %
  child foreach \name in {Triangle, Rectangle, Circle} %
  { %
    node {\name} %
  };%
```



Замечания к примеру

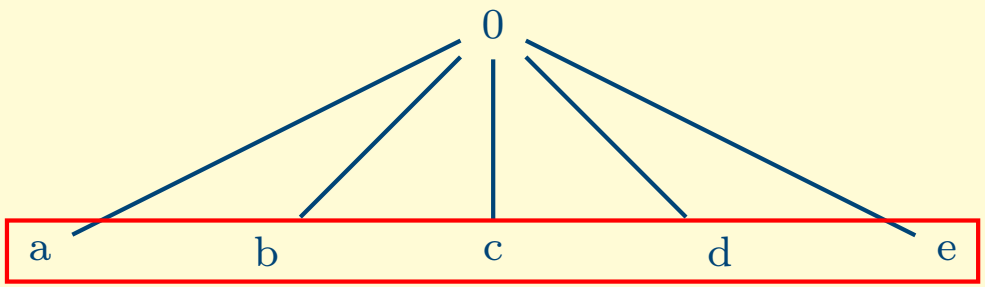
- ✔ Функции `width("str")`, `height("str")` и `depth("str")` возвращают, соответственно, ширину, высоту и глубину (высота от базовой линии текста до его нижнего края) строки `str`, если бы она была набрана текущим шрифтом. Функции являются частью математического модуля Ti*k*Z (см. далее).
- ✔ Настройки перед операцией `child` относятся к дочерней вершине и всем подчинённым ей.
- ✔ Вместо функции роста деревьев по умолчанию (использующей настройки `level distance` и т. д.) можно определять собственные функции. Некоторые дополнительные готовые функции роста определены в библиотеке Ti*k*Z `trees`.

Имена дочерних вершин

Замечание: для всех дочерних вершин, имена которым не были определены пользователем, автоматически назначаются имена вида: ⟨*имя_родителя*⟩-⟨*номер*⟩ (рекурсивно для нижних уровней: ⟨*имя_родителя*⟩-⟨*номер*$_1$⟩-⟨*номер*$_2$⟩ и т. д.) △

Пример

```
\node (T) {0} %
  child foreach \i in {a, ..., e} {node {\i}};
\draw [red] (T-1.north west) rectangle (T-5.south east);
```



Операция рисования дуги от родительской к дочерней вершине

edge from parent [⟨*параметры*⟩]

Определяет параметры отображения дуги. Эта операция должна встречаться только внутри дочернего пути в конце, возможно, после описания дочерней вершины. Возможно указание после неё операций **node**, которые будут добавлены вдоль дуги. Если эта операция пропущена, будет автоматически добавлена дуга, использующая следующий стиль:

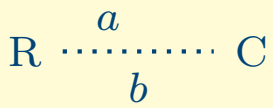
edge from parent (стиль, по умолчанию: `draw`)

Исполняется перед рисованием дуги по умолчанию.

Пример расположения вершин вдоль дуги от родительской к дочерней вершине

Пример

```
\node {R} %
  [grow = right] %
  child %
  { %
    node {C} %
    edge from parent [dotted] %
      node [pos = .3, above] {$a$} %
      node [below] {$b$} %
  };
```



Операция рисования дуги от родительской к дочерней вершине (продолжение)

edge from parent path = ⟨*путь*⟩ (по умолчанию: см. ниже)

Определяет операции рисования дуги. По умолчанию содержит код:

```
(\tikzparentnode \tikzparentanchor) -- %
  (\tikzchildnode \tikzchildanchor)
```

Здесь вместо макросов `\tikzparentnode` и т. д. при рисовании будут подставлены имя родительской вершины, её якорь (при необходимости с точкой в начале) и имя с якорем дочерней. Таким образом, по умолчанию будет нарисована прямая линия.

Замечание: далее будут приведены готовые операции рисования дуг из библиотеки TikZ `trees`. △

Операция рисования дуги от родительской к дочерней вершине (окончание)

parent anchor = ⟨*имя_якоря*⟩ (по умолчанию: **border**)

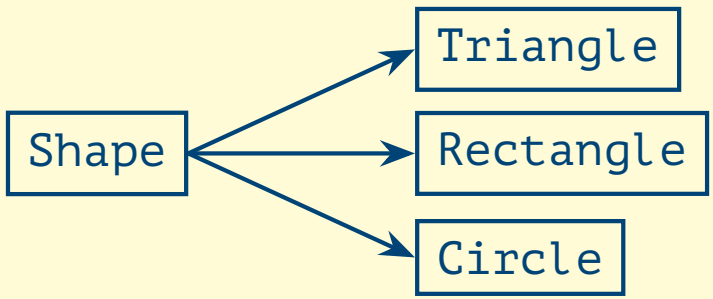
child anchor = ⟨*имя_якоря*⟩ (по умолчанию: **border**)

Определяют якоря родительской и дочерней вершины, между которыми рисуется дуга. Значение **border** приводит к передаче пустой строки вместо макроса `\tikzparentanchor` или `\tikzchildanchor`, в результате чего дуга будет начинаться/заканчиваться на границе вершины.

Пример определения вида дуг между родительскими и дочерними вершинами

Пример

```
\tikzset { %
  every node/.style = {draw, font = \ttfamily}, %
  edge from parent/.append style = {-Stealth} } %
\node {Shape} %
  [sibling distance = 1.8em, %
  grow' = right, growth parent anchor = east, %
  parent anchor = east, child anchor = west] %
  child foreach \name in {Triangle, Rectangle, Circle} %
    { node [anchor = west] {\name} };
```

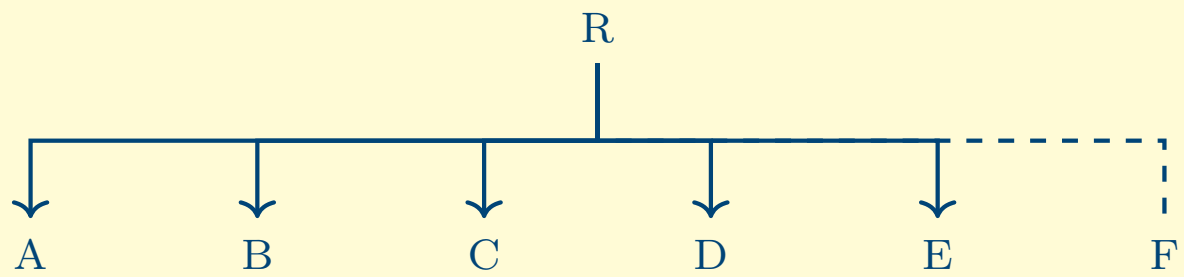


Предопределённые пути для рисования дуг библиотеки TikZ `trees`

`edge from parent fork down/right/left/up` (стиль)

Пример

```
\node {R} %
  [->, edge from parent fork down]
  child foreach \i in {A, ..., E} { node {\i} } %
  child { node {F} edge from parent [-, dashed] };
```



Стили, относящиеся к деревьям

every child (стиль, по умолчанию: ⟨*пусто*⟩)

Исполняется перед отображением пути каждой дочерней вершины, аналогично указываемым параметрам операции `child`.

every child node (стиль, по умолчанию: ⟨*пусто*⟩)

Исполняется перед отображением каждой дочерней вершины, в дополнение к стилю `every node`.

level = ⟨*номер*⟩ (стиль, по умолчанию: ⟨*пусто*⟩)

Исполняется перед отображением дочерних вершин одной родительской, ⟨*номер*⟩ определяет слой дерева. При переопределении стиля этот номер доступен через параметр «#1».

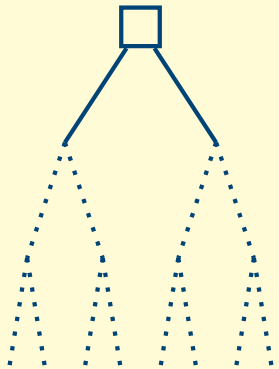
Стили, относящиеся к деревьям

level ⟨*номер*⟩ (стиль, по умолчанию: ⟨*пусто*⟩)

Исполняется перед отображением дочерних вершин заданного уровня, в дополнение к стилю `level`.

Пример

```
\tikzset { %
  level/.style = {sibling distance = 20mm / 2^#1},
  level 2/.style = {dotted} } %
\node [draw] {} %
  [level distance = 2em] %
  child foreach \i in {1, 2} %
    { child foreach \j in {1, 2} %
      { child foreach \k in {1, 2} } %
    };
```



Вычисления координат

Обзор средств определения координат

Для определения сложных правил расположения графических объектов пакет Ti*k*Z предоставляет следующие возможности:

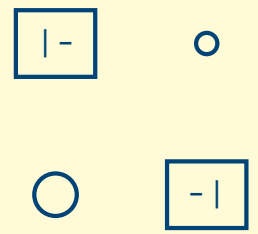
- ✔ операции «`|-`» и «`-|`» для нахождения точек пересечения горизонтальной и вертикальной прямых, проходящих через точки с заданными координатами.
- ✔ встроенные средства преобразования координат при помощи параметров (сдвиг, поворот, и т. д.);
- ✔ библиотеку Ti*k*Z `calc` для вычисления координат;
- ✔ операцию пути `let`;
- ✔ математический модуль Ti*k*Z.

Все перечисленные модули можно использовать как по отдельности, так и совместно друг с другом.

Нахождение координат пересечения горизонтальной и вертикальной прямой

Пример

```
\draw (0, 0) circle [radius = 4pt] %
  coordinate (A);
\draw (1, 1) circle [radius = 2pt];
\node [draw] at (A |- 1, 1) {\texttt {|-}};
\node [draw] at (A -| 1, 1) {\texttt {-|}};
```



Замечание: операции над координатами «|-» и «-|» не следует путать с рассмотренными ранее одноимёнными операциями путей, которые добавляли к пути пары отрезков. △

Математический модуль

Математический модуль (слой, уровень) Ti*k*Z:

предоставляет возможности вычисления по формулам, записанным на интуитивно понятном языке, в вещественной арифметике над скалярами и длинами. Так как реализован на математических возможностях TEX, значения должны находиться в диапазоне $\pm 16\,383{,}999\,99$. Может использоваться как неявно в высокоуровневых конструкциях Ti*k*Z, когда в аргументах команд вместо значений указываются выражения, так и отдельно от PGF при помощи команд **\pgfmathparse** и т. д.

Замечание: ранее рассмотренный пример изображения дерева использовал математический модуль для определения расстояния между вершинами в зависимости от уровня (с операциями деления и возведения в степень). △

Синтаксис выражений, поддерживаемых математическим модулем TikZ

Математический модуль поддерживает следующие элементы выражений в инфиксной записи:

- ✔ десятичные вещественные числа в экспоненциальной записи;
- ✔ целые числа в двоичной, восьмеричной и шестнадцатеричной записи;
- ✔ регистры и размерности TEX;
- ✔ текстовые строки в двойных кавычках;
- ✔ арифметические операции и функции;
- ✔ скобки.

Операции и функции, поддерживаемые математическим модулем

Таблица 2: основные операции математического модуля

-x	x + y	x - y	x * y	x / y	x ^ y
x == y	x != y	x > y	x < y	x >= y	x <= y
!x	x && y	x || y	x ? y : z		
xr (преобразование радианов к градусам)					

Операции и функции, поддерживаемые математическим модулем (окончание)

Таблица 3: основные функции математического модуля

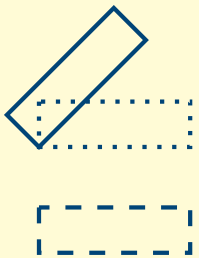
sqrt(x)	pow(x, y)	e	exp(x)
ln(x)	log2(x)	log10(x, y)	abs(x)
mod(x, y)	round(x)	floor(x)	ceil(x)
sin(x)	cos(x)	tan(x)	pi
asin(x)	acos(x)	atan(x)	atan2(x, y)
rnd ($\in [0, 1]$)		random(x, y) ($\in [x, y]$)	
veclen(x, y) (длина двумерного вектора)			
width("s")	height("s")	depth("s")	

Встроенные средства TikZ для преобразования координат

Средства преобразования координат: аналогично библиотеке OpenGl, позволяют определять последовательность преобразований для используемых координат (и только для них, но не для текста, толщины линий и т. п.) На координаты, расположенные внутри последовательно вложенных областей с различными матрицами преобразований, действует общая матрица, равная произведению всех матриц областей.

Пример

```
\draw [dashed] (0, 0) rectangle (1, 0.3);
{ [yshift = 0.7cm] %
  \draw [dotted] (0, 0) rectangle (1, 0.3);
  \draw [rotate = 45] (0, 0) rectangle (1, 0.3);
} %
```



Основные параметры преобразований координат

xshift = ⟨*длина*⟩

yshift = ⟨*длина*⟩

Сдвигает координаты на заданную длину вдоль, соответственно, оси x и y.

shift = {⟨*координата*⟩}

Сдвигает координаты на величины, равные x- и y-компонентам заданной координаты, одновременно вдоль осей x и y.

Основные параметры преобразований координат (продолжение)

scale = ⟨*множитель*⟩

Умножает все координаты на заданный множитель.

xscale = ⟨*множитель*⟩

yscale = ⟨*множитель*⟩

Умножает только, соответственно, x- и y-компоненты координат на заданный множитель

scale around = {⟨*множитель*⟩ : ⟨*координата*⟩}

Выполняет масштабирование координат. В отличие от `scale` центром масштабирования является не начало координат, а заданная координата.

Основные параметры преобразований координат (окончание)

rotate = ⟨*угол*⟩

Выполняет поворот координат против часовой стрелки на заданный угол в градусах относительно начала координат.

rotate around = {⟨*угол*⟩ : ⟨*координата*⟩}

Поворачивает координаты относительно заданной точки.

rotate around x = ⟨*угол*⟩

rotate around y = ⟨*угол*⟩

rotate around z = ⟨*угол*⟩

Поворачивают трёхмерные координаты на заданный угол вокруг оси x, y и z соответственно.

Вычисления координат (окончание)

Выражения библиотеки `calc`

После подключении библиотеки Ti*k*Z `calc` становятся доступны выражения, заключённые в символы «`($...$)`», при помощи которых можно выполнять следующие операции с координатами:

- ✔ сумма и разность координат (то есть смещение первой координаты на величину второй);
- ✔ умножение координаты на число (масштабирование относительно начала координат);
- ✔ Нахождение координат точки на прямой, проходящей через две другие точки с заданными координатами, которая либо находится на заданном расстоянии от одной из них, либо делит отрезок в заданной пропорции, либо является проекцией на прямую третьей точки.

Синтаксис выражений пакета `calc`

- ✔ Синтаксис выражения координат:

  ([⟨*параметры*⟩]**$**⟨*координатное_выражение*⟩**$)**

- ✔ ⟨*параметры*⟩ могут содержать ранее рассмотренные настройки преобразования координат, такие как `xshift`.
- ✔ ⟨*координатное_выражение*⟩ имеет следующий синтаксис:

  ⟨*компонент*⟩ { ⟨*операция*⟩ ⟨*компонент*⟩ }

- ✔ ⟨*операция*⟩ может быть «+» или «-».
- ✔ ⟨*компонент*⟩ имеет следующий синтаксис:

  ⟨*множитель*⟩ * ⟨*координата*$_1$⟩ ⟨*модификаторы*⟩

Синтаксис выражений пакета `calc` (продолжение)

- ✔ ⟨*множитель*⟩ может представлять собой произвольное выражение, поддерживаемое математическим модулем Ti*k*Z. Считается, что множитель заканчивается до следующего вхождения символов «*(» (без пробела). Так как сам множитель может содержать скобки, для разрешения неоднозначностей возможно поместить его внутрь фигурных скобок.
- ✔ ⟨*модификаторы*⟩ имеют следующий синтаксис:

  !⟨*параметр*⟩**!**⟨*угол*⟩**:**⟨*координата*$_2$⟩

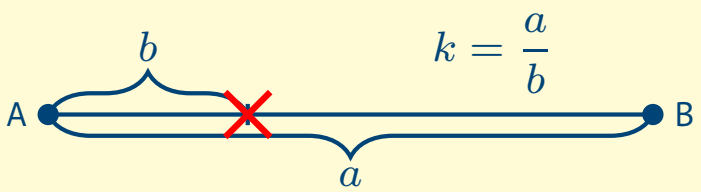
- ✔ Необязательный аргумент ⟨*угол*⟩ может задавать угол поворота. Он также может задаваться выражением, поддерживаемым математическим модулем Ti*k*Z.
- ✔ Все координаты должны заключаться в круглые скобки.

Синтаксис выражений пакета `calc` (продолжение)

- ✔ В зависимости от типа модификатора ⟨*параметр*⟩ может представлять собой выражение множителя, длины или третьей координаты.

Множитель

```
($(A) ! k ! (B)$)
```



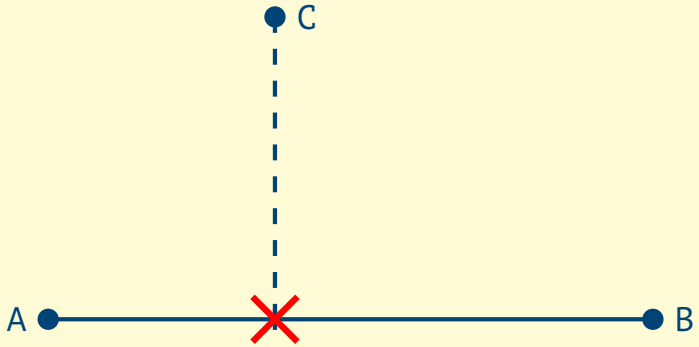
Длина

```
($(A) ! d ! (B)$)
```

Синтаксис выражений пакета `calc` (продолжение)

Координата

```
($(A) ! (C) ! (B)$)
```

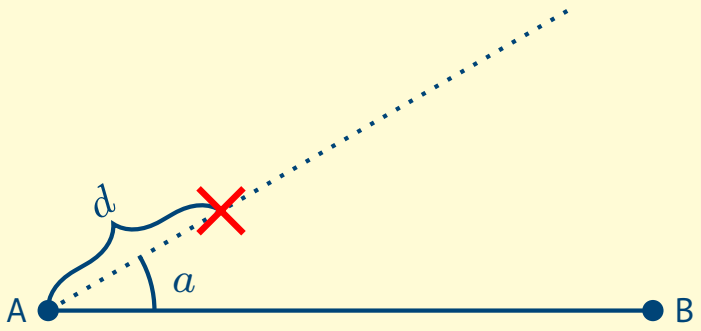


Синтаксис выражений пакета `calc` (окончание)

✔ Если в модификаторах присутствует угол, на него поворачивается прямая, на которой расположена искомая точка, относительно первой из двух заданных точек.

+ угол

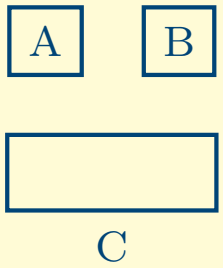
```
($(A) ! d ! a : (B)$)
```



Пример вычислений координат

Пример

```
\node [draw] (A) {A};
\node [draw, right = 1em of A] (B) {B};
\coordinate (D) at %
  ([rotate = -90]$(B.west) - (A.east)$);
\draw %
  ($(A.south west) + (D)$) %
    coordinate (E) %
      rectangle %
      ($(B.south east) + (D) - (0, .5cm)$) %
      coordinate (F);
\coordinate (G) at (E |- F);
\node [anchor = north] at ($(G)!.5!(F)$) {C};
```



Операция пути `let`

- ✔ Синтаксис операции пути `let`:

  let ⟨*присваивание*⟩ { , ⟨*присваивание*⟩ } **in** ⟨*тело*⟩

- ✔ ⟨*присваивание*⟩ может выполняться с вычислением скаляра (длины) или координаты. В зависимости от этого оно может иметь одну из двух форм:

  \n⟨*скалярный_регистр*⟩ **=** {⟨*выражение*⟩}

  \p⟨*координатный_регистр*⟩ **=** {⟨*выражение*⟩}

- ✔ Имя регистра может быть либо целым неотрицательным числом, либо строкой в фигурных скобках.

Операция пути `let` (продолжение)

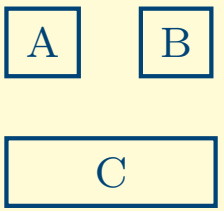
- ✔ {⟨*выражение*⟩} должно поддерживаться математическим модулем Ti*k*Z или быть выражением координат соответственно.
- ✔ ⟨*тело*⟩ может быть определением пути Ti*k*Z.
- ✔ Скалярное присваивание определяет макрос \n{⟨*имя_регистра*⟩}, которое хранит результат вычисления своего выражения.
- ✔ Координатное присваивание определяет макросы \x{⟨*имя_регистра*⟩} и \y{⟨*имя_регистра*⟩}, которые хранят, соответственно, x- и y-координаты результата вычисления своего выражения, а также макрос \p{⟨*имя_регистра*⟩}, который хранит обе координаты через запятую.

Операция пути `let` (окончание)

✔ Все вышеперечисленные макросы определены только в последующих присваиваниях и в теле операции `let`. Обращение к ним заменяется хранящимся в них значении.

Пример

```
\node [draw] (A) {A};
\node [draw, right = 1em of A] (B) {B};
\coordinate (D) at %
  ([rotate = -90]$(B.west) - (A.east)$);
\path %
  let \p1 = ($(B.east) - (A.west)$), %
      \n1 = {veclen(\x1, \y1)} in %
    node at ($(A.south west) + (D)$) %
      [draw, anchor = north west, %
      minimum width = \n1] {C};
```



Визуализация данных в виде графиков

Обзор средств рисования графиков

В Ti*k*Z доступны следующие средства рисования графиков:

- ✔ ранее рассмотренные низкоуровневые операции с путями;
- ✔ библиотека Ti*k*Z `datavisualization`;
- ✔ внешний пакет LaTeX `pgfplots`, использующий Ti*k*Z (рассматривается далее).

Основные возможности пакета `pgfplots`:

- ✔ графики: в 2 и 3 измерениях; в декартовых и полярных координатах, обычные, логарифмические и полулогарифмические;
- ✔ внешний вид отображения графиков: линии, точки, кусочно-постоянные прямые, стобликовые диаграммы, сетки, поверхности и т. д.;
- ✔ дополнительные данные: обозначения осей, отметки значений, легенды и т. д.

Основные этапы работы с пакетом pgfplots

1. Подключение пакета:

```
\usepackage {pgfplots}
```

2. Необязательно: загрузка в преамбуле вспомогательных библиотек `pgfplots` командой `\usepgfplotslibrary`.
3. Установка в любом месте глобальных настроек при помощи команды `\pgfplotsset`. Как минимум рекомендуется устанавливать параметр совместимости с используемой версией пакета:

```
\pgfplotsset { compat = 1.13, ... }
```

4. Использование команд и окружений пакта внутри команды TikZ `\tikz` или окружения `tikzpicture`.

Окружения пакета

\begin{**axis**} [⟨*параметры*⟩]
⟨*содержимое_окружения*⟩
\end{**axis**}

Отображает оси координат и предназначена для помещения внутрь команд вывода графиков, легенд и т. д.

Замечание: вспомогательные окружения **semilogxaxis**, **semilogyaxis** и **loglogaxis** для полу- и логарифмических осей являются сокращённой записью использования окружения `axis` со следующими аргументами:

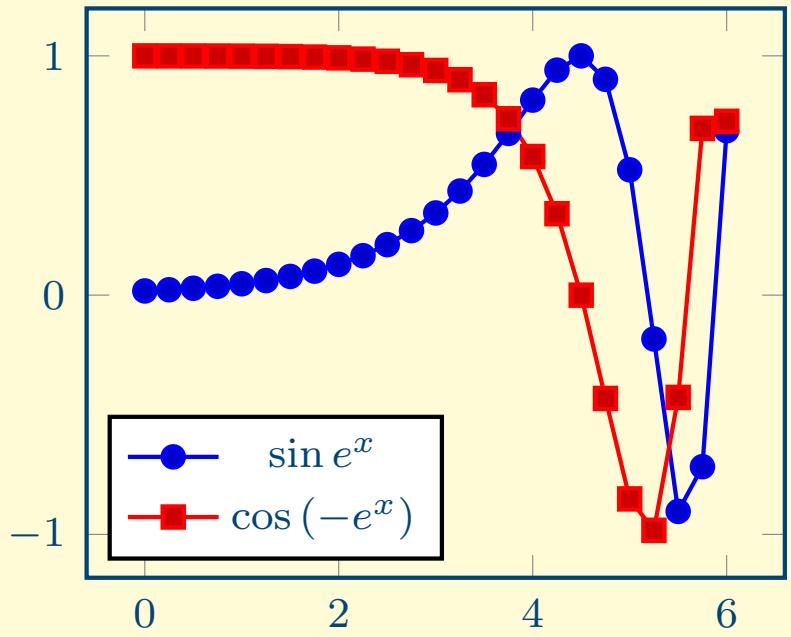
```
\begin {axis} [xmode = normal | log, ymode = normal | log, ...]
  % ...
\end {axis}
```

△

Пример вывода графиков

Пример

```
\begin {tikzpicture}
  \begin {axis} %
    [ %
      width = 62mm, %
      domain = 0 : 6, %
      legend pos = south west
    ] %
    \addplot {sin(exp(x))};
    \addplot {cos(-exp(x))};
    \legend %
      {$\sin e^x$,%
      $\cos \del {-e^x}$}
  \end {axis}
\end {tikzpicture}
```



Команды пакета pgfplots

\addplot [⟨*параметры*⟩] ⟨*входные_данные*⟩ ⟨*завершающие_команды_пути*⟩

Выводит график. Различные варианты указания входных данных будут рассмотрены далее. ⟨*параметры*⟩ задают внешний вид. Если они пропущены, они загружаются из следующего элемента *циклического списка*, благодаря которому разные графики на одних осях выглядят по разному. Текущие настройки графика сохраняются для дальнейшего отображения в легенде. ⟨*завершающие_команды_пути*⟩ могут содержать команды пути TikZ, исполняемые в конце вывода графика.

Команды пакета pgfplots

Замечания:

- ✔ Команда **\addplot+** отличается от \addplot только тем, что её параметры всегда добавляются к параметрам из циклического списка.
- ✔ Аналогично, в пакете определены команды **\addplot3** и **\addplot3+** для построения трёхмерных графиков. △

\legend {⟨*список*⟩}

Определяет список подписей к элементам легенды, разделённый запятыми. Также включает её отображение.

Замечание: рекомендуется не разделять элементы списка легенды запятыми с пробелами, так как пробелы могут попасть в их надписи. △

Определения источника данных из выражения

expression {⟨*выражение*⟩} (входные данные)

Определяет входные данные в форме выражения, которое затем используется для вычисления значений функции математическим модулем Ti*k*Z. ⟨*выражение*⟩ может зависеть от переменных «x» и «y» (для трёхмерных графиков).

{⟨*выражение*⟩} (входные данные)

Сокращённая запись для «expression {⟨*выражение*⟩}».

(⟨*выражение_x*⟩, ⟨*выражение_y*⟩) (входные данные)

(⟨*выражение_x*⟩, ⟨*выражение_y*⟩, ⟨*выражение_z*⟩)

Аналогично, определяют координаты параметрическим способом.

Определения источника данных из выражения (продолжение)

domain = $\langle x_{\min} \rangle$: $\langle x_{\max} \rangle$ (по умолчанию: -5 : 5)

y domain = $\langle y_{\min} \rangle$: $\langle y_{\max} \rangle$ (по умолчанию: -5 : 5)

Ограничивают диапазоны выводимых значений по осям x и y.

samples = {⟨*количество*⟩} (по умолчанию: 25)

samples y = {⟨*количество*⟩} (по умолчанию: 25)

Определяет количество значений переменной x (y для трёхмерного случая), в которых будут вычислены значения координат.

samples at = {⟨*список_значений*⟩}

Определяет значения переменной в явном виде.

Определения источника данных из выражения (окончание)

variable = {⟨*имя_переменной*⟩} (по умолчанию: x)

variable y = {⟨*имя_переменной*⟩} (по умолчанию: y)

Определяют используемые в выражениях имена переменных вместо «x» и «y».

Замечание: по умолчанию библиотека pgfplots использует собственный модуль, реализующий вычисления над вещественными числами в двойной точности, вместо рассмотренного ранее математического модуля TikZ, точность которого гораздо ниже. △

Определения источника данных из списка

coordinates {⟨*список_координат*⟩} (входные данные)

Определяет входные данные в виде явного списка координат в следующем формате:

- ✔ Координаты разделяются пробелами.
- ✔ Каждая координата представляет список значений через запятые, ограниченный круглыми скобками.
- ✔ Значения кроме чисел могут содержать любые выражения, поддерживаемые математическим модулем Ti*k*Z. Если выражение само содержит круглые скобки, его необходимо заключить в фигурные скобки.

Определения источника данных из списка (окончание)

- ✔ После координаты может следовать другая, отделяемая от первой символами «+-». В этом случае вторая координата определяет значения погрешностей для первой, которые отображаются на графиках при включении соответствующей настройки.
- ✔ После координаты (и, возможно, её погрешности) могут быть указаны произвольные метаданные в квадратных скобках, связанные с этой координатой. Способ их отображения зависит от настроек, например, в виде цвета (см. далее).
- ✔ Пустые строки означают разрывы функции.

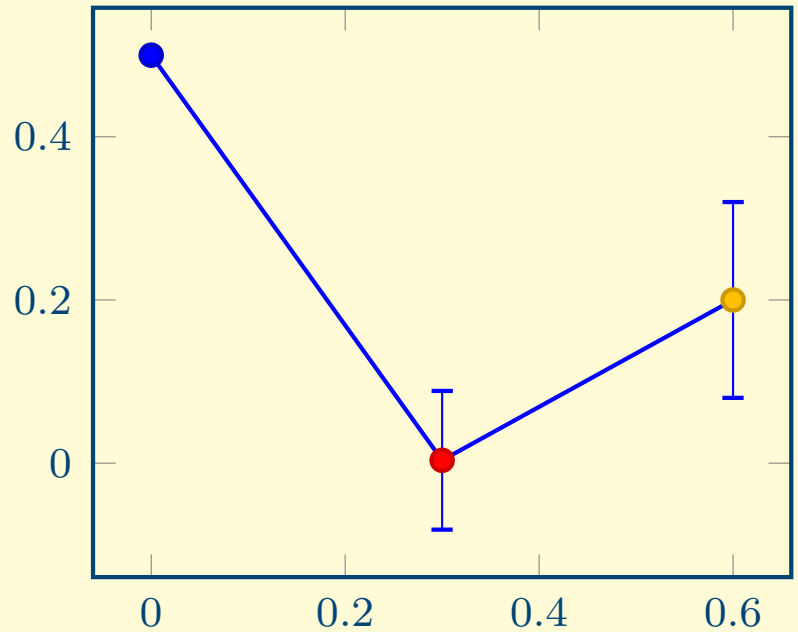
math parser = true, false (по умолчанию: **true**)

Включает/отключает синтаксический анализатор математических выражений для координат.

Пример вывода графика по координатам

Пример

```
\begin {axis} [width = 62mm]
  \addplot+ %
    [error bars/y dir = both,
    error bars/y explicit, %
    scatter, %
    point meta = explicit] %
    coordinates %
    { %
      (0,   0.5)   [1] %
      (0.3, {sin(0.2)}) +- %
      (0.1, 0.085) [3] %
      (0.6, 0.2)       +- %
      (1e-1, 0.12) [2] %
    };
\end {axis}
```



Определения источника данных из таблицы

table [⟨*выбор_колонок*⟩] {⟨*данные*⟩} (входные данные)

Определяет входные данные в табличном виде из текстового файла с заданным именем через параметр ⟨*данные*⟩ или непосредственно передаваемые через этот параметр в следующем формате:

- ✔ По умолчанию строки разделяются переводами строки, ячейки — пробелами.
- ✔ Строки-комментарии (по умолчанию начинаются с «#» и «%») игнорируются.
- ✔ Первая строка-не-комментарий анализируется на содержание названий колонок. По этим именам можно в дальнейшем на них ссылаться. Иначе строка считается первой строкой числовых данных.

Определения источника данных из таблицы (продолжение)

- ✔ Таблица может иметь несколько колонок. По умолчанию первая используется в качестве x-координат, вторая — y. При помощи параметров в ⟨*выбор_колонок*⟩ (см. далее) можно выбрать колонки для значений координат x, y, z, ошибок по каждой координате и метаданных.

table/x = {⟨*имя_колонки*⟩}

table/x index = {⟨*номер_колонки*⟩}

Позволяет указывать колонку для координаты x по имени или номеру (начиная с 0). Для этой и последующих настроек существуют аналогичные для координат **y**, **z**, **x**/**y**/**z error** (**plus**/**minus**) и **meta**.

Определения источника данных из таблицы (продолжение)

table/x expr = {⟨*выражение*⟩}

Позволяет использовать для значений координаты x математическое выражение вместо данных из какой-либо колонки. В выражении могут использоваться следующие макросы:

Таблица 4: макросы для использования в выражениях

Макрос	Возвращаемое значение
\thisrow {⟨*имя*⟩}	Значение колонки с заданным именем в текущей строке
\thisrowno {⟨№⟩}	Аналогично, для номера колонки
\coordindex	Номер текущей строки данных, начиная с 0

Определения источника данных из таблицы (окончание)

table/row sep = newline, \\ (по умолчанию: **newline**)

Устанавливает в качестве разделителя строк таблицы символ перевода строки или «\\».

table/col sep = tab, comma, semicolon, ... (по ум.: **space**)

Устанавливает в качестве разделителя строк таблицы заданные символы.

table/skip first n = {⟨*количество*⟩} (по умолчанию: 0)

Позволяет игнорировать первые строки файла в заданном количестве.

Визуализация данных в виде графиков (окончание)

Типы двумерных графиков

`sharp plot`

Ломаная линия между вершинами (по умолчанию).

`smooth`

Плавная интерполированная линия.

`const plot mark left`

Соединяет вершины последовательностью горизонтальных и вертикальных отрезков (соединяемые точки расположены на левых краях «ступеней»). Синоним: **`const plot`**.

`const plot mark right`

Аналогично, на правых краях.

Типы двумерных графиков (продолжение)

`const plot mark mid`

Аналогично, посередине.

`jump mark left`

`jump mark right`

`jump mark mid`

Аналогично, но без вертикальных отрезков.

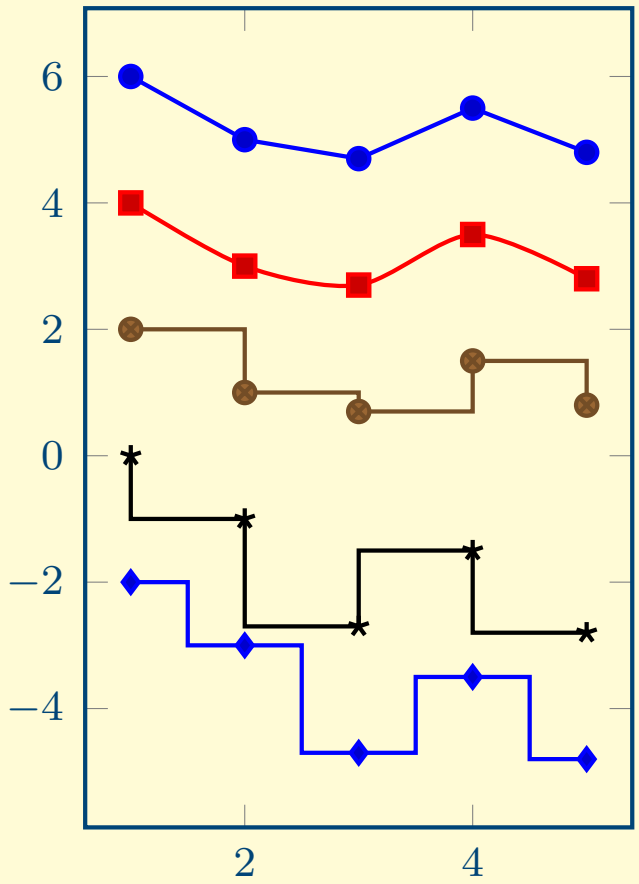
`only marks`

Рисует только отметки в координатах.

Примеры типов графиков

Пример

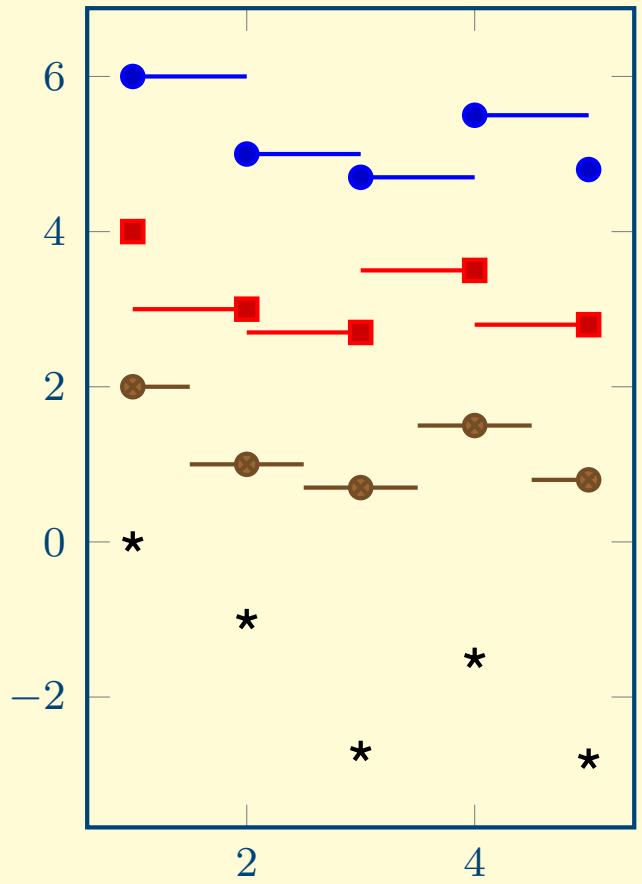
```
\begin {axis} %
  [width = 52mm, height = 70mm]
  \addplot+ table {data1.txt};
  \addplot+ [smooth] %
    table {data2.txt};
  \addplot+ [const plot mark left] %
    table {data3.txt};
  \addplot+ [const plot mark right] %
    table {data4.txt};
  \addplot+ [const plot mark mid] %
    table {data5.txt};
\end {axis}
```



Примеры типов графиков (продолжение)

Пример

```
\begin {axis} %
  [width = 52mm, height = 70mm]
  \addplot+ [jump mark left] %
    table {data1.txt};
  \addplot+ [jump mark right] %
    table {data2.txt};
  \addplot+ [jump mark mid] %
    table {data3.txt};
  \addplot+ [only marks] %
    table {data4.txt};
\end {axis}
```



Типы двумерных графиков (продолжение)

ybar interval

Рисует прямоугольники от оси $\overrightarrow{OX}$ вверх между x-координатами соседних точек и высоты, определяемой y-координатой левой точки (таким образом, значение y последней точки не используется).

xbar interval

Аналогично `ybar interval`, прямоугольники рисуются от оси $\overrightarrow{OY}$ вправо.

ybar

Столбики между осью $\overrightarrow{OX}$ и заданными координатами.

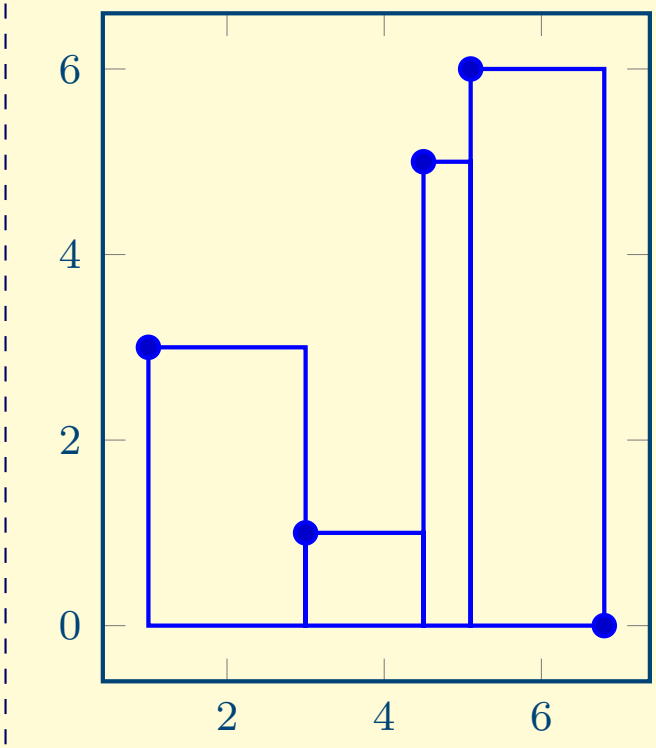
xbar

Аналогично, между осью $\overrightarrow{OY}$ и заданными координатами.

Примеры типов графиков (продолжение)

Пример

```
\begin {axis} %
  [width = 52mm, height = 60mm]
  \addplot+ [ybar interval] %
    table {data6.txt};
\end {axis}
```



Типы двумерных графиков (окончание)

scatter

То же, что и `only marks`, но дополнительно меняет внешний вид отметок графика (по умолчанию цвет) в зависимости от метаданных, привязанных к координатам. Может быть также использован совместно с другими типами графиков. Источник метаданных и влияние на внешний вид определяются настройками, которые приведены ниже.

scatter src = none, ⟨*выражение*⟩, ... (по ум.: none)

Определяет источник метаданных (см. табл. 5, 6). Является синонимом **point meta**. Ранее приводился пример использования ключа.

Использование метаданных

Таблица 5: возможные значения ключа `scatter src`

Значение	Смысл
none	Отключает использование метаданных.
x, **y**, **z**	Использует соответствующую координату в качестве метаданных.
f(x)	Использует координату **y** при выводе двумерного графика и **z** для трёхмерного.
explicit	Указывает, что метаданные должны задаваться в явном виде вместе с координатами. Их формат подробно описан вместе с командами `\plot coordinates` и `\plot table`.

Использование метаданных (окончание)

Таблица 6: возможные значения ключа `scatter src` (окончание)

Значение	Смысл
`explicit symbolic`	Аналогично **`explicit`**, но метаданные представляются в виде произвольного текста. Используется при некоторых настройках отображения, например, для вывода метаданных возле отметок (см. `nodes near coords`).
⟨*выражение*⟩	Определяет метаданные как результат вычисления выражения PGF. Может использовать переменные x, y, z, макросы табл. 4 и т. д.

Вывод подписей возле отметок на графиках

nodes near coords* = {⟨*содержимое*⟩} (по ум.: см. ниже)

Модификация настройки `scatter`, которая размещает вершины со значениями метаданных возле отметок графиков. ⟨*содержимое*⟩ определяет способ вывода, по умолчанию оно равно `\pgfmathprintnumber {\pgfplotspointmeta}` (вывод метаданных в виде числа). Если настройка `point meta` не указана, по умолчанию считается, что задано значение `f(x)`. Вариант «со звёздочкой» используется для комбинирования стиля с уже установленными.

nodes near coords style = {⟨*настройки_стиля*⟩}

Добавляет настройки к стилю вершин для вывода метаданных.

Настройки осей и графиков

xtick = \empty, data, ⟨*список_значений*⟩ (по умолчанию: {})

Определяет расположение отметок вдоль оси $\overrightarrow{OX}$. Пустой список (по умолчанию) означает автоматическое расположение, **\empty** подавляет их использование, ⟨*список_значений*⟩ задаёт явно x-координаты (в формате команды PGF \foreach, включая возможное многоточие). Аналогично, определены ключи **ytick** и **ztick**.

xtick distance = ⟨*число_или_пусто*⟩ (по умолчанию: пусто)

Определяет расстояние между отметками при их автоматическом расположении. Пустое значение означает автоматический выбор. Аналогично, **ytick distance** и **ztick distance**.

Настройки осей и графиков (продолжение)

xticklabels = ⟨*список_меток*⟩ (по умолчанию: {})

Определяет обозначения возле отметок вдоль оси $\overrightarrow{OX}$. Аналогично, **yticklabels** и **zticklabels**.

x tick label as interval = `true, false` (по ум.: `false`)

Устанавливает обозначения между соседними отметками в виде интервалов. Аналогично, **y tick label as interval** и **z tick label as interval**.

tick style = ⟨*стиль_TikZ*⟩

Используется для рисования отметок в дополнение к **every tick**.

Настройки осей и графиков (продолжение)

`symbolic x coords` = ⟨*список_строк*⟩

Определяет символические обозначения для координат. Каждая строка сопоставляется с очередным целым числом, начиная с 0. Эти значения можно затем использовать во входных данных вместо чисел. Подход аналогичен использованию `xticklabels`, только проще (но менее гибко). Аналогично, **`symbolic y coords`** и **`symbolic z coords`**.

`enlarge x limits` = `{auto, true, false, ...}` (по ум.: `auto`)

Позволяет увеличивать размеры оси $\overrightarrow{OX}$ по отношению к ширине графика. Значения в списке указываются из табл. 7, 8. Аналогично, **`enlarge y limits`** и **`enlarge z limits`**.

Настройки осей и графиков (окончание)

Таблица 7: возможные значения ключа `enlarge x limits`

Значение	Смысл
true	Увеличивает левую и правую границы.
false	Использует плотно прилегающие к графику границы.
lower	Увеличивает левую границу, правая прилегает плотно.
upper	Увеличивает правую границу.
auto	Увеличивает границы только если они были автоматически определены (для трёхмерных графиков работает только для оси $\overrightarrow{OZ}$).

Настройки осей и графиков (окончание)

Таблица 8: возможные значения ключа `enlarge x limits`

Значение	Смысл
value=⟨*коэф.*⟩	Определяет коэффициент, на который умножается размер оси при увеличении. При отсутствии других значений возможна сокращённая запись: `enlarge x limits` = ⟨*коэф.*⟩
abs value=⟨*коэф.*⟩	Определяет коэффициент, на который умножается размер оси при увеличении.

Пример столбиковой диаграммы

Пример

```
\begin {axis} %
  [width = 50mm, height = 70mm, %
  ybar, nodes near coords, %
  symbolic x coords = {Yes, No}, %
  enlarge x limits = {abs = 10mm}, %
  enlarge y limits = {abs = 5mm}, %
  xtick = data, ytick = \empty, %
  legend pos = north west]
  \addplot+ coordinates %
    {(Yes, 20) (No, 80)};
  \addplot+ coordinates %
    {(Yes, 37) (No, 63)};
  \legend {Men,Women};
\end {axis}
```