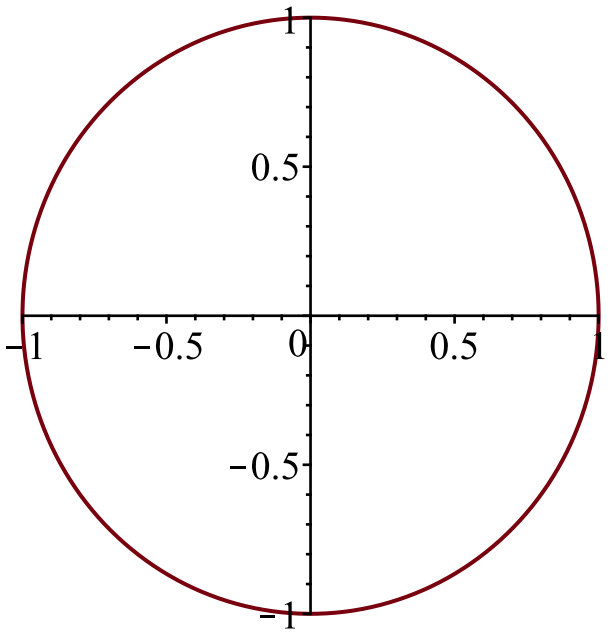
Графики параметрически заданных функций

В тех случаях, когда одному значению аргумента x может соответствовать несколько значений функции $y(x)$, стоит обратить внимание на возможность построения кривых, заданных в параметрической форме.

Для построения таких кривых приходится несколько изменить синтаксис команды **plot**. Первый аргумент этой команды превращается в список из трёх элементов. Два первых элемента списка представляют собой выражения, зависящие от некоторого параметра, например, *t*. Эти выражения определяют абсциссу и ординату точки кривой при заданном значении параметра. Последний элемент списка является уравнением, в левой части которого указано имя параметра, а в правой − диапазон его изменения.

Именно таким способом проще всего построить окружность.

```
> plot([cos(t),sin(t),t=0..2*Pi]);
```



Границы диапазона могут задаваться с помощью объявленных ранее переменных и тоже представляют собой выражения.

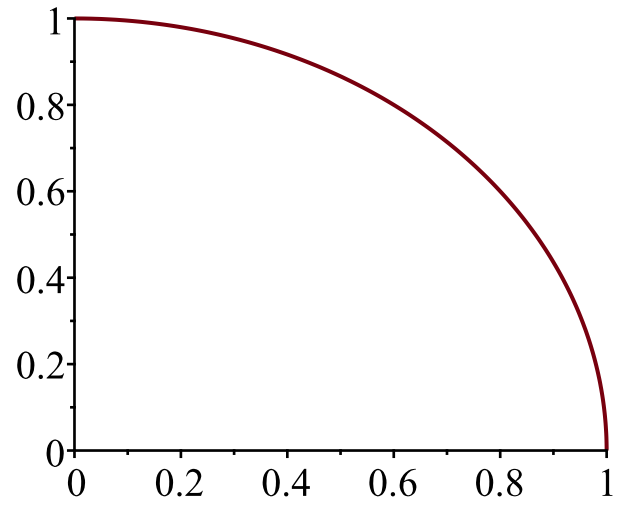
```
> Tmin:=0;Tmax:=Pi/2;
```

$$Tmin := 0$$

$$Tmax := \frac{1}{2}\,\pi$$

Четверть окружности строится следующим образом (из-за автомасштабирования она будет больше похожа на четверть эллипса)

```
> plot([cos(t),sin(t),t=Tmin..Tmax]);
```



Данный способ построения кривых является совершенно естественным при решении

задач по классической механике. Если параметр t имеет смысл времени, а два первых элемента списка задают закон движения материальной точки на плоскости, то изображаемая кривая будет являться траекторией движения этой точки.

Иногда закон движения удобно задавать в функциональной форме.

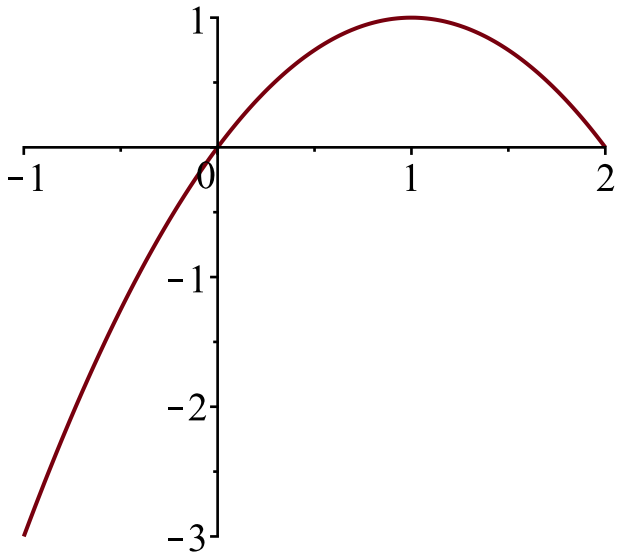
```
> X:=t->1-t;Y:=t->1-t^2;
```

$$X := t \to 1 - t$$

$$Y := t \to 1 - t^2$$

Тогда можно не только опустить зависимость от параметра в двух первых элементах списка, но и отбросить имя самого параметра в последнем элементе. Достаточно указать только сам диапазон.

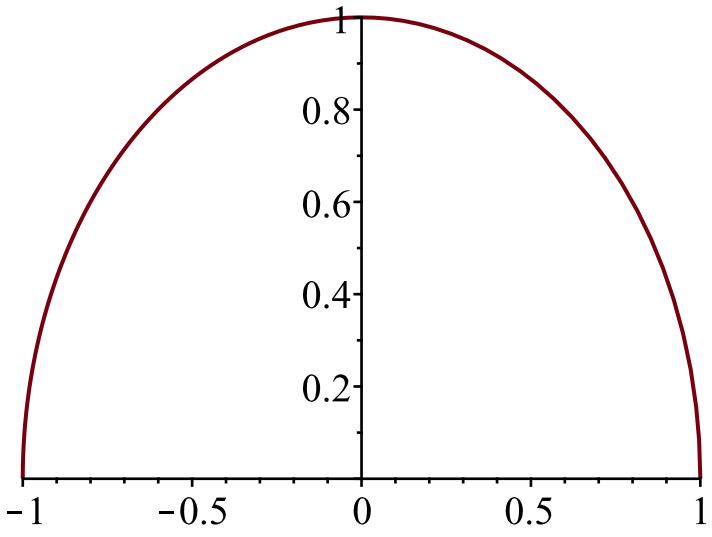
```
> plot([X,Y,-1..2]);
```



Границы диапазона могут обращаться в ∞. Такая возможность может оказаться весьма кстати в тех ситуациях, когда необходимо гарантировать замкнутость отображаемой кривой.

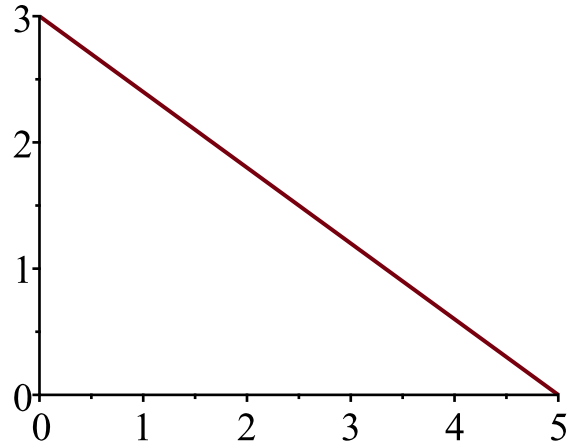
Вот таким нетрадиционным способом тоже можно нарисовать полуокружность.

```
> plot([(1-t^2)/(1+t^2),2*t/(1+t^2),t=0..infinity]);
```



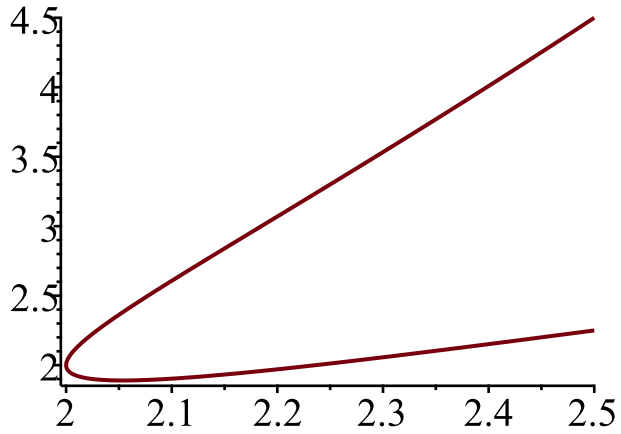
Сложный вид зависимостей $x(t)$ и $y(t)$ вовсе не обязан приводить к замысловатым кривым. Так, например, можно рисовать отрезки, соединяющие точки на координатных осях.

```
> plot([5*cos(t)^2,3*sin(t)^2,t=0..Pi/2]);
```



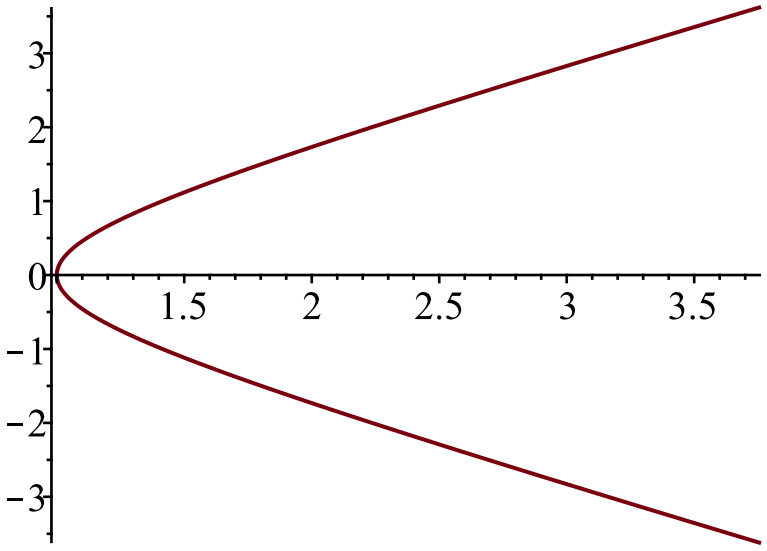
Верно и обратное. Простые, на первый взгляд, зависимости способны порождать кривые, которые трудно представить без помощи команды **plot**.

```
> plot([t+1/t,t+1/t^2,t=1/2..2]);
```



В то же время все известные из курса аналитической геометрии кривые второго порядка без труда описываются в параметрической форме. В частности, для рисования ветви гиперболы достаточно выполнить следующую команду

```
> plot([cosh(t),sinh(t),t=-2..2]);
```

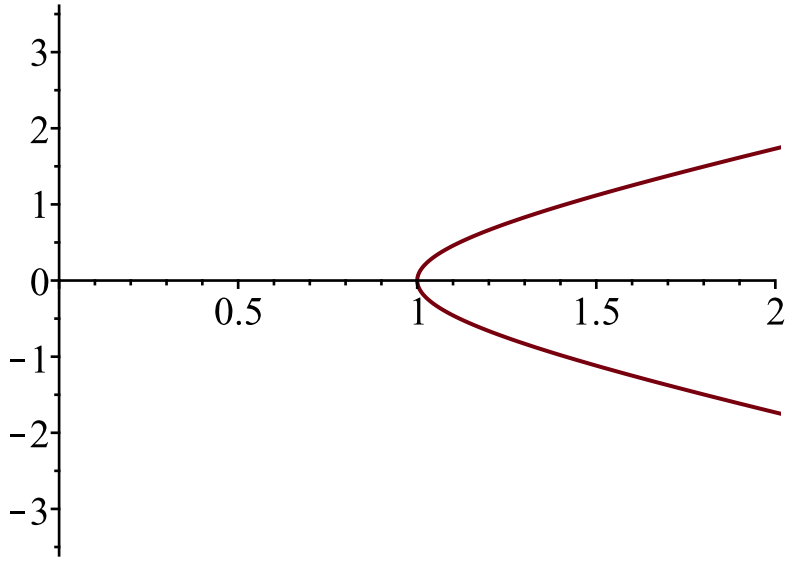


При построении графиков явно заданных функций диапазон изменения аргумента *x* известен заранее. Для функций, заданных в параметрическом виде, этот диапазон вычисляется в ходе построения самого графика, что приводит к автомасштабированию изображения не только по оси ординат (об этом уже шла речь в предыдущих разделах), но и по оси абсцисс.

Если мы хотим отобразить кривую только в некотором заданном диапазоне аргумента *x*,

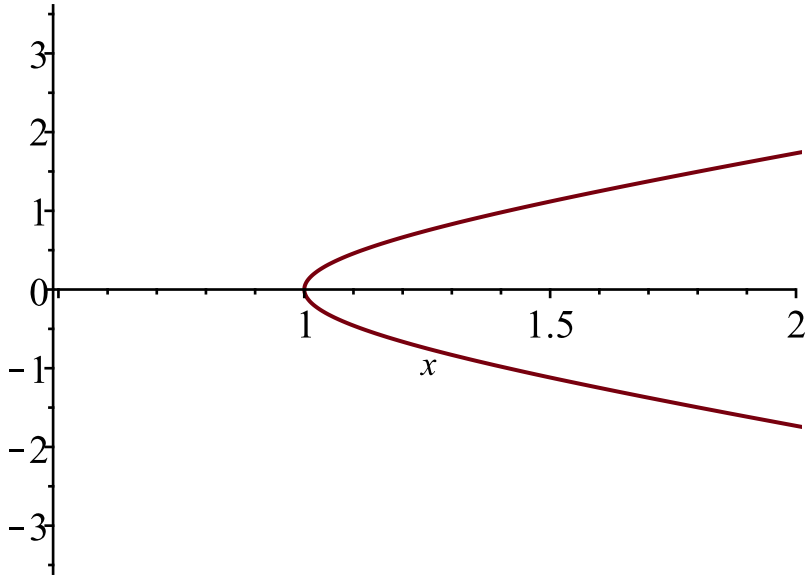
то этот диапазон следует указать вторым аргументом команды **plot**, то есть сразу же после списка.

```
> plot([cosh(t),sinh(t),t=-2..2],0..2);
```



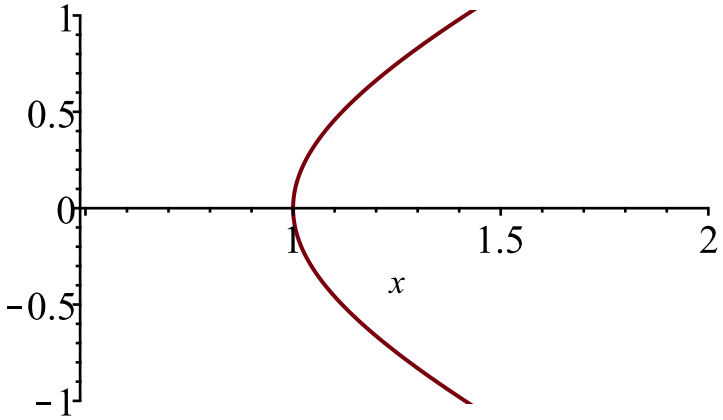
Несмотря на то, что об этом ничего не сказано в документации, второй параметр можно записывать и в традиционном для команды **plot** виде, то есть в виде уравнения. Имя этой переменной станет названием оси абсцисс.

```
> plot([cosh(t),sinh(t),t=-2..2],x=1/2..2);
```



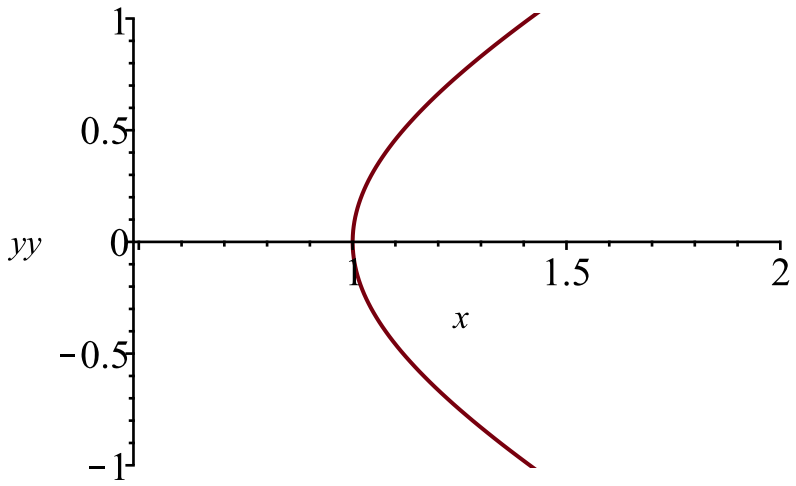
Ограничить область, в которой отображается кривая, можно не только по оси абсцисс, но и по оси ординат. Для этого, также как и в случае явно заданных функций, третьим параметром **plot** следует указать ещё один диапазон.

```
> plot([cosh(t),sinh(t),t=-2..2],x=1/2..2,-1..1);
```



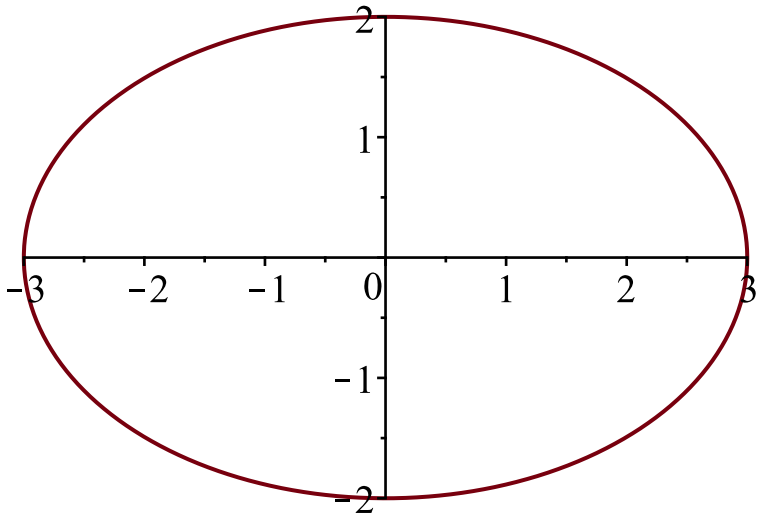
Продолжая аналогии, отметим, что данный параметр также допускает преобразование в уравнение. Имя переменной, стоящей в левой части этого уравнения, станет названием оси ординат.

```
> plot([cosh(t),sinh(t),t=-2..2],x=1/2..2,yy=-1..1);
```



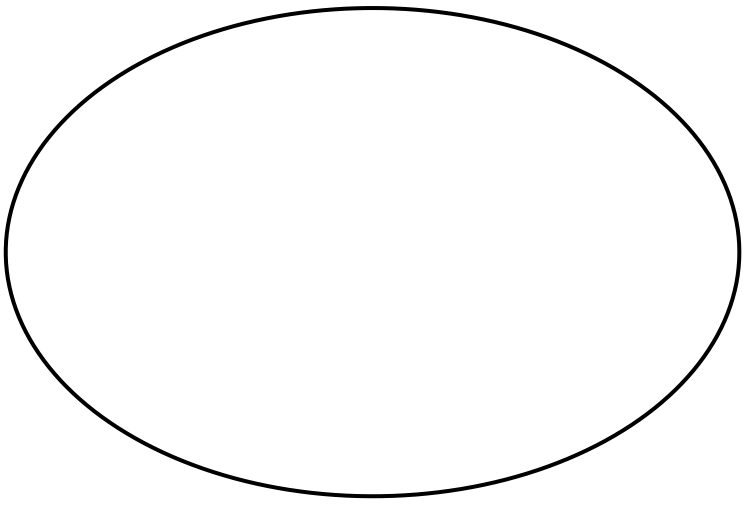
После того как все необходимые диапазоны добавлены, в конец списка аргументов команды **plot** можно добавить любые другие изученные ранее опции. Например, для корректного отображения геометрических фигур рекомендуется использовать опцию *scaling*.

```
> plot([3*cos(t),2*sin(t),t=0..2*Pi],scaling=constrained);
```



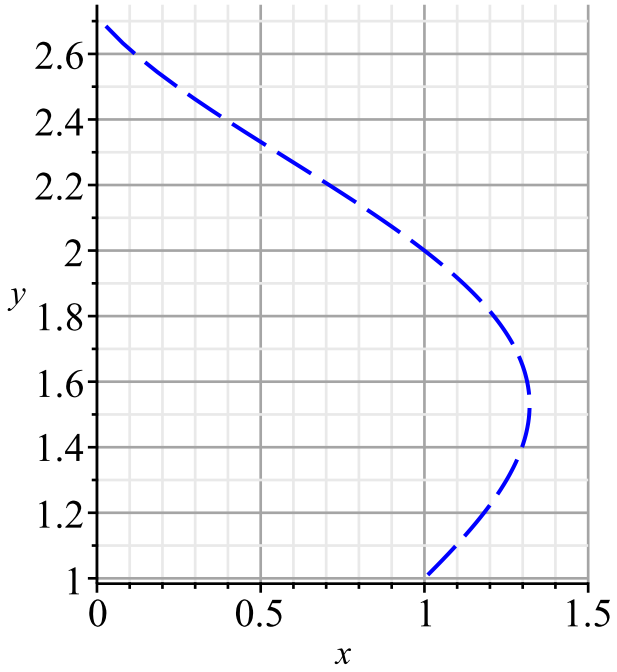
При рисовании геометрических фигур координатные оси, как правило, не нужны, а для печати цвет лучше сделать чёрным.

```
> plot([3*cos(t),2*sin(t),t=0..2*Pi],scaling=constrained,
      axes=none,color=black);
```



Пример с большим количеством настроек

```
> plot([t^(1/(1+t)),(1+t)^(1/t),t=1/40..infinity],x=0..3/2,y=1..2.75,
      scaling=constrained,color=blue,linestyle=dash,gridlines);
```



Опираясь на возможность построения графиков функций, заданных в параметрическом виде, можно рисовать кривые, которые заданы в полярных координатах.

В качестве примера изобразим спираль Архимеда $\rho = \phi$. Вспоминая, что $x = \rho \cos(\phi)$, $y = \rho \sin(\phi)$, записываем команду

```
> plot([phi*cos(phi),phi*sin(phi),phi=0..3*Pi],scaling=constrained,
      color=coral,axes=none);
```