

Graphing in L^AT_EX using PGF and TikZ

Lindsey-Kay Lauderdale Mathew R. Gluck

Department of Mathematics, University of Florida

PGF and TikZ are packages in L^AT_EX that allow for programming figures into documents. PGF is an acronym for ‘Portable Graphics Format’ and TikZ is a recursive acronym for ‘TikZ ist kein Zeichenprogramm’. With these packages, we get all the advantages of the L^AT_EX-approach to typesetting. As with typesetting in L^AT_EX however, programming graphics in PGF/TikZ has a steep learning curve. Fortunately, there is a 726 page user-friendly PGF/TikZ manual [1] to explain everything to us! The goal of this workshop is to explore some of the basic features of the PGF/TikZ packages in L^AT_EX while programming some simple graphics.

Using the PGF/TikZ packages in a L^AT_EX document requires loading the PGF package somewhere in the preamble:

```
\usepackage{pgfplots}
```

This package has many libraries. We will use the following code to load the three libraries which will be used in the problems below.

```
\usepgfplotslibrary{polar}
\usepgfplotslibrary{shapes.geometric}
\usetikzlibrary{calc}
```

The first allows for polar graphs, the second is used to insert basic shapes into a document and the third will help us label nodes in any shapes that we manually draw. Before we start with some examples, let’s create a macro with some axes options. Suppose we want all functions graphed in the xy -plane to have the following specifications:

- (i) The x -axis and the y -axis in the middle of our graph. Code:

```
axis x line=middle, axis y line=middle
```

- (ii) The x -axis and the y -axis labeled. Code:

```
xlabel={ $x$ }, ylabel={ $y$ }
```

- (iii) The scale on the axes to be equal. Code:

```
axis equal
```

Henceforth, this group of specifications will be called ‘my style’. They can be seen in any axis environment in which ‘my style’ option has been specified. To define these specifications and name them ‘my style’, place the following code in the preamble of your L^AT_EX document.

```
\pgfplotsset{my style/.append style={axis x line=middle, axis y line=
middle, xlabel={\$x\$}, ylabel={\$y\$}, axis equal }}
```

To typeset a graphics in \LaTeX , we need to tell \LaTeX that we want to typeset graphics. This will be done in the ‘tikzpicture’ environment. Among the graphics we will be programming are plots of functions. To plot a function in the xy -plane, we will use the ‘axis’ environment inside the ‘tikzpicture’ environment. The basic structure of the code used to achieve this is as follows:

```
\begin{tikzpicture}
  \begin{axis}[axes options]
    ...
  \end{axis}
\end{tikzpicture}
```

Every command inside these environments will end with a semicolon. Now we can start with some examples. These examples will not demonstrate all of the capabilities of PGF/TikZ, but they will introduce us to some of the features that we are most likely to use.

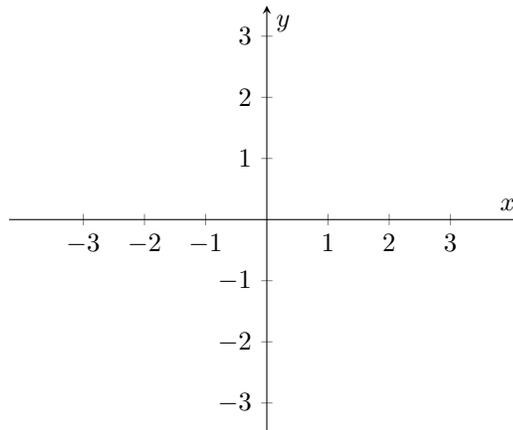
Suppose we want to recreate the axis in the following exam question from a calculus 2 course:

Problem 1. Consider the region \mathcal{R} bounded by $y = \sqrt{x - 1}$ and $y = x - 1$.

- (a) Sketch the region \mathcal{R} .
- (b) Come up with an expression (involving one or more integrals as appropriate) for determining the volume of the solid obtained by rotating \mathcal{R} about the line

$$x = -2.$$

Then, calculate the volume by evaluating this expression.



Because we are drawing the xy -plane without plotting any functions, we need to specify the minimum and maximum values for both x and y . We also need to specify where the tick marks on the axes should be placed. The code for these specifications will be placed after ‘my style’ in the axes options. Note that when written after the ‘my style’ options, any axis options that conflict with axis options defined in ‘my style’ will take precedence over the conflicted ‘my style’ options. The following code yields the desired graph:

```
\begin{tikzpicture}
  \begin{axis}[my style, xtick={-3,-2,...,3}, ytick={-3,-2,...,3},
    xmin=-3, xmax=3, ymin=-3, ymax=3]

  \end{axis}
\end{tikzpicture}
```

Next, we'll plot both an xy -axis and a function on the plane determined by the axis. Consider the graph in the following precalculus exam problem.

Problem 2. Use the graph of the function $f(x) = |x|$ to select the equation of the function whose graph is shown.

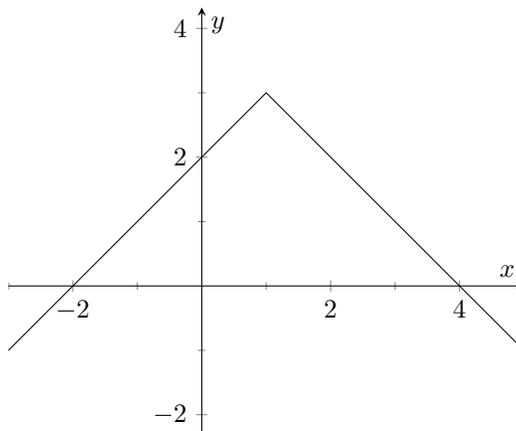
(a) $y = -|x - 1|$

(b) $y = -|x - 1| + 3$

(c) $y = |x - 1| + 3$

(d) $y = -|x + 1| + 3$

(e) $y = |x + 1| + 3$



Start with the same basic code structure for plotting functions mentioned above.

```
\begin{tikzpicture}
  \begin{axis}[my style]
    ...
  \end{axis}
\end{tikzpicture}
```

Will make two changes to this code. Namely, we will adjust the placement of the tick marks on the axes and we will plot the desired function. By default, tick marks are placed at the even-integer values on the axes. Here we want one tick mark between each of the even-valued tick marks. We accomplish this by placing the following code in the axis options:

```
minor tick num=1
```

Our last step is to program the plot of the function into the code above. The structure of the syntax used for this is

```
\addplot[plot options]{function};
```

In this example, the function to be plotted is $y = -|x - 1| + 3$. The domain over which to plot the function is placed in the plot options. The code that produces the desired graph is

```
\begin{tikzpicture}
  \begin{axis}[my style, minor tick num=1]

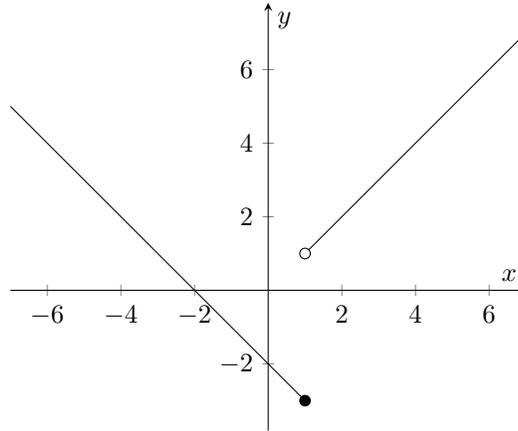
    \addplot[domain=-3:5]{-abs(x-1)+3};

  \end{axis}
\end{tikzpicture}
```

In the next example, we explain how to plot a simple piecewise-defined function. Consider the graph of the function in the following precalculus problem.

Problem 3. The graph of f is given below. Use the graph of f to find $2f(1) - f(4)$.

- (a) -10
- (b) -2
- (c) 0
- (d) 2
- (e) 10



We wish to program the graph of the function

$$f(x) = \begin{cases} -x - 2 & \text{if } -\infty < x \leq 1 \\ x & \text{if } 1 < x < \infty. \end{cases}$$

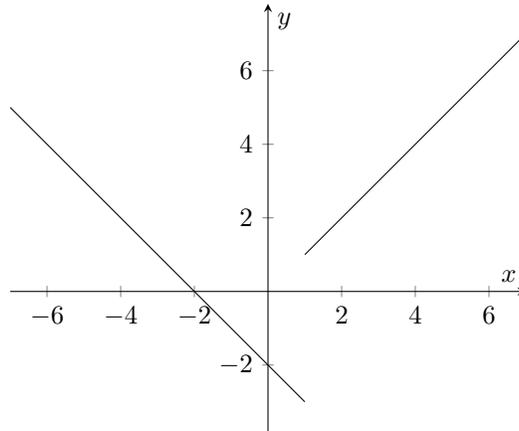
To program this graph, start with a similar code as in Problem 2. There will be two changes made to the code. First, we must plot two separate lines, one for each of the subdomains $(-\infty, 1]$ and $(1, \infty)$. Second, we place a solid dot and an open dot at the appropriate positions of the graph so that the value of $f(1)$ is apparent from the graph of the function. The code that achieves the plots of the two lines is as follows:

```
\begin{tikzpicture}
  \begin{axis}[my style]

    \addplot[domain=-7:1] {-x-2};
    \addplot[domain=1:7] {x};

  \end{axis}
\end{tikzpicture}
```

The above code yields the graph



It remains to make the value of $f(1)$ apparent on the graph. To do this, the ‘addplot’ command is used. The following code

```
\addplot[mark=*] coordinates {(1,-3)};
```

adds a closed circle to the graph at $(1, -3)$, while the code

```
\addplot[mark=*,fill=white] coordinates {(1,1)};
```

adds an open circle to the graph at $(1, 1)$. Combining the three pieces of code above, we have the following code which produces the desired graph.

```
\begin{tikzpicture}
  \begin{axis}[my style]

    \addplot[domain=-7:1] {-x-2};
    \addplot[domain=1:7] {x};

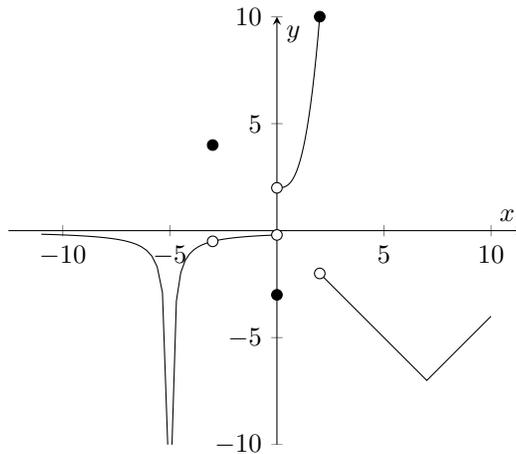
    \addplot[mark=*] coordinates {(1,-3)};
    \addplot[mark=*,fill=white] coordinates {(1,1)};

  \end{axis}
\end{tikzpicture}
```

In the next example, we plot a more intricate piecewise-defined function. Consider the graph of the function given in the following calculus 1 problem.

Problem 4. The graph of a function f is shown below. Find all values of a for which $\lim_{x \rightarrow a} f(x)$ exists but f is discontinuous at $x = a$.

- (a) -3
- (b) -1
- (c) 0
- (d) 2
- (e) 7



To program this graph, we will start with a similar code as in Problem 3 and make the following changes:

- (i) Changing the piecewise functions and their respective domains.
- (ii) Add the following code which plots multiple open/closed circles:

`only marks`

This will code allows us to plot multiple circles in the same line of code and ensures that we are only plotting the open/closed circles (not also the lines connecting them). This is a plot option and should be placed in the square brackets after the `addplot` command.

These two changes yield the complete code:

```

\begin{tikzpicture}
  \begin{axis}

    \addplot[domain=-11:-5.1] {1/(x+5)};
    \addplot[domain=-4.9:0] {-1/(x+5)};
    \addplot[domain=0:2] {x^3+2};
    \addplot[domain=2:10] {abs(x-7)-7};

    \addplot[mark=*,only marks] coordinates {(-3,4)(0,-3)(2,10)};
    \addplot[mark=*,fill=white,only marks] coordinates {(-3,-5)
      (0,-.2)(0,2)(2,-2)};

  \end{axis}
\end{tikzpicture}

```

The PNG and TikZ packages also support polar graphs. To graph in the polar coordinate system, we will use the ‘`polaraxis`’ environment. Just like in the xy -plane, we define a style which we would like all of our polar graphs to have. For this workshop, we’ll label our

angular axis every $\frac{\pi}{6}$ radians and we'll increase the thickness of the curves in our polar plots from the default curve thickness. This style will be called 'my style polar' and is defined by placing the following code in the preamble of our L^AT_EX document.

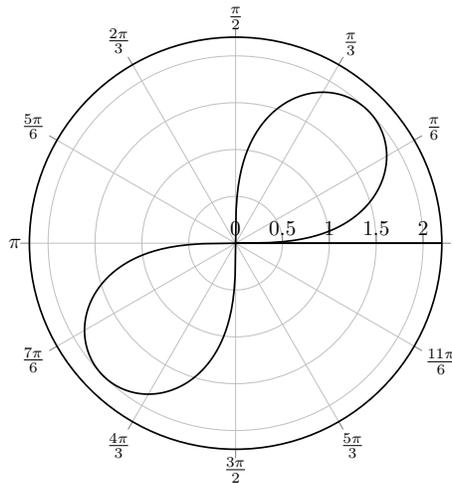
```
\pgfplotsset{my style polar/.append style={xticklabels={, ,
  $\frac{\pi}{6}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{2\pi}{3}$,
  $\frac{5\pi}{6}$, $\pi$, $\frac{7\pi}{6}$, $\frac{4\pi}{3}$,
  $\frac{3\pi}{2}$, $\frac{5\pi}{3}$, $\frac{11\pi}{6}$,}, thick }}
```

Moreover, similarly to the cartesian plots constructed above, we have the following basic code structure for creating polar plots which includes the 'my style polar' polar axis options:

```
\begin{tikzpicture}
  \begin{polaraxis}[my style polar]
    ...
  \end{polaraxis}
\end{tikzpicture}
```

To practice plotting in polar coordinates, consider the graph in the following calculus 2 problem.

Problem 5. Find the area of the region enclosed by the polar curve $r^2 = 4 \sin(2\theta)$.



To plot this graph, start with the basic code listed above for polar plots and simply input the code that plots the graph of the desired curve. In this case, there are a few technicalities to worry about. First, we need to increase the number of samples so the graph of the curve appears smooth. Second, since the curve does not define r as a function of θ , we need to plot $r = 2\sqrt{\sin(2\theta)}$ on the appropriate domain (we must exclude all values of theta for which $\sin(2\theta) < 0$). To increase the number of samples used in plotting our curve, insert the following code in the 'addplot' options:

```
samples=100
```

The code that produces the desired graph is as follows:

```

\begin{tikzpicture}
  \begin{polaraxis}[my style polar]

    \addplot[domain=0:90, samples=100]{2*sqrt(sin(2*x))};
    \addplot[domain=180:270, samples=100]{2*sqrt(sin(2*x))};

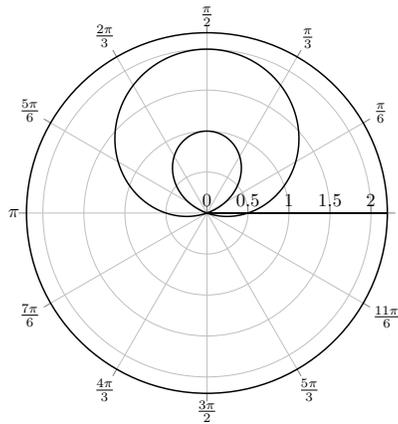
  \end{polaraxis}
\end{tikzpicture}

```

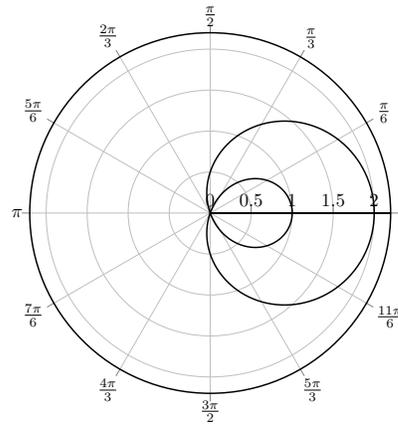
In our next example, we discuss a method of organizing polar graphs in an array in a visually appealing way. To accomplish this we use the same ‘tabular’ environment with a few minor adjustments. Consider the following problem from a multiple-choice calculus 2 exam.

Problem 6. Which of the following graphs best represents the graph of $r = 2 \cos(2\theta)$?

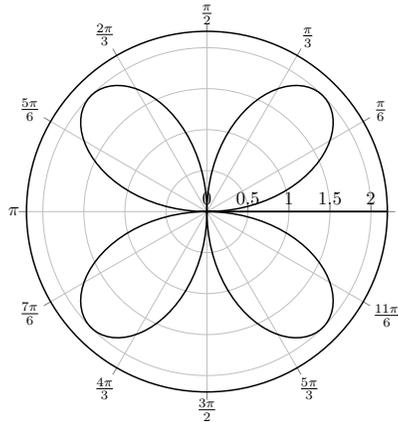
(a)



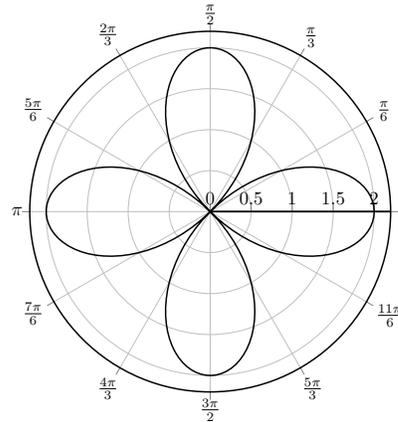
(b)



(c)



(d)



To program this problem, note that the code for each multiple choice option is similar to the code in Problem 5. We'll only discuss the arrangement of the multiple-choice options into an array. As mentioned above, this is achieved via the 'tabular' environment. We need to make a few adjustments to the default 'tabular' environment. First, the width of the columns in the tabular environment is determined automatically by L^AT_EX. Since the entries in our array are graphs, we want to manually specify the width of the columns in our array to be 6cm. Moreover, we'll place the graph in each array entry at the top of the array cell via the 'p' command. These specifications are reflected in the following code:

```

\begin{tabular}{p{6cm} p{6cm}}
  (a)

  &

  (b)

  \\

  (c)

  &

  (d)
\end{tabular}

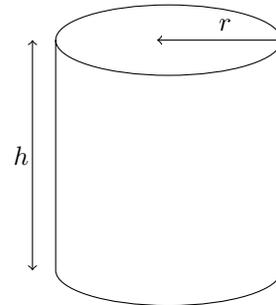
```

The PGF and TikZ packages also support simple shapes. In the next example we give an example of how to manipulate one of the built-in shapes offered by PGF/TikZ. Note that the shape library must be loaded in the preamble to use the built-in shapes. To load the shape library, simply type

```
\usepgflibrary{shapes.geometric}
```

into the preamble of your document. Consider the cylinder drawn in the following calculus 1 problem.

Problem 7. A cylinder is being flattened so that its volume does not change. If the height h of the cylinder is changing at 0.2 inches per second, find the rate of change of the radius r when $r = 3$ inches and $h = 4$ inches.



To program the code for a cylinder we will use the node command inside the TikZ environment:

```
\begin{tikzpicture}
  \node[draw, cylinder]
\end{tikzpicture}
```

This code produces this cylinder:



This undesirable cylinder can be modified by using the node options below.

(i) We can rotate the cylinder by specifying the degree of rotation. Code:

```
rotate=90
```

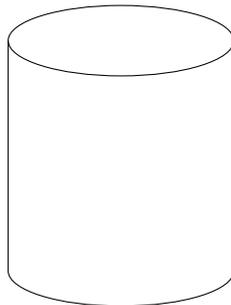
(ii) We can tilt the cylinder by changing the aspect ratio. Code:

```
shape aspect = 4
```

(iii) We can specify an minimum height and a minimum width. Code:

```
minimum height=4cm, minimum width=3cm
```

These node options should be placed in the square brackets after `\node` and will produce a much more desirable cylinder:



It remains to label the cylinder. There are many different ways of doing this. One way is to use the over 20 locations on the cylinder that PGF has already predefined (see page 434 of [1]). In the code of the cylinder below, these predefined locations are used to label the height h and the radius r of the cylinder. They are also used in the placement of the arrows seen in the picture. The following code produces the desired output:

```
\begin{tikzpicture}
  \node [draw, cylinder, shape aspect=4, rotate=90, minimum height=4cm,
    minimum width=3cm] (c) {};

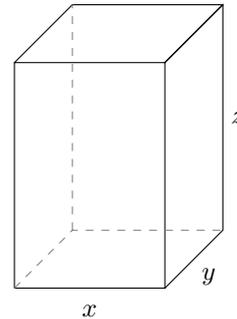
  \coordinate(htop) at ($(c.before top)!-1*.1!(c.after top)$);
  \coordinate(hbot) at ($(c.after bottom)!-1*.1!(c.before bottom)$);
  \coordinate(hlab) at ($(htop)!.5!(hbot)+(c.north)!.9!(c.center)$);
  \draw[<->] (hbot)--(htop);
  \path (hlab) node {$h$};      %Modify height label here

  \coordinate (center) at ($(c.before top)!0.5!(c.after top)$);
  \coordinate (rlab) at ($(center)!0.5!(c.after top)$);
  \coordinate (rtop) at ($(center)!-1*.1!(c.after top)$);
  \draw[<->] (rtop) -- (c.after top);
  \path (rlab) node[above] {$r$};      %Modify radius label here

\end{tikzpicture}
```

In our final example, we'll manually draw a shape. This may be necessary if one wishes to include in their document a shape that is not one of the predefined shapes offered by PGF. It is not necessary to use the predefined L^AT_EX shapes. In the next example, we will see a box which is drawn using coordinates in \mathbb{R}^3 .

Problem 8. A shipping crate with a square base is to be constructed so that it has a volume of 250 cubic meters. The material for the top and bottom cost \$2 per square meter and the material for the sides costs \$1 per square meter. What dimensions will minimize the cost of construction?



To recreate this graphic, we will use the path and draw commands. The path command will allow us to define the eight vertices of the box and the draw command will connect them for us. The code of the box is as follows:

```

\begin{tikzpicture}

  \pgfmathsetmacro{\x}{1}           %Modify length of box here
  \pgfmathsetmacro{\y}{1}           %Modify width of box here
  \pgfmathsetmacro{\z}{1.5}        %Modify height of box here

  \path (0,0,\y) coordinate (A)     (\x,0,\y) coordinate (B)
        (\x,0,0) coordinate (C)     (0,0,0) coordinate (D)
        (0,\z,\y) coordinate (E)     (\x,\z,\y) coordinate (F)
        (\x,\z,0) coordinate (G)     (0,\z,0) coordinate (H);

  \draw (A) -- (B) -- (C) -- (G) -- (F) -- (B)
        (A) -- (E) -- (F) -- (G) -- (H) -- (E);
  \draw[gray,dashed] (A) -- (D) -- (C) (D) -- (H);

\end{tikzpicture}

```

The sides of the box can be labeled using similar techniques to Problem 7. Note that the box can be resized by changing the values for x , y and z in the code above.

References

- [1] Tantau,Till, *The TikZ and PGF Packages (Manual for version 2.10)*, Institut für Theoretische Informatik, Universität zu Lübeck, 2007.