Creating Figures With TikZ

Me

April 2, 2023

Contents

1 Introduction to TikZ

- https://www.overleaf.com/learn/latex/TikZ_package
- https://ctan.org/pkg/pgf?lang=en

2 Drawing Lines

First we have to include the TikZ package in the preamble with \usepackage{tikz}.
The figure we want to draw has to be enclosed in the "tikzpicture" env.
In the figure 1 below we are using the following code:

```
\begin{tikzpicture}
    \draw (0, 0) -- (2, 0);
    \draw (2, 0) -- (1, 1.73205);
\end{tikzpicture} % \label{tikz:1st_figure}

To draw a line, where:
    (0, 0) => the starting point of the line,
    -- => the sign that the line is going from to,
    (2, 0) => the ending point of the line
    ; => to finish the TikZ input command


All the lengths are calculated in centimeters.

Instead of creating new \draw commands for each line we can also draw
them by simply adding another "-- (x, y)". Like this:

    \draw (0, 0) -- (2, 0) -- (1, 1.73205) -- (0, 0);

We can also use a "cycle" Instead of the "(0, 0)" point ro specify that
we are going right back to the beginning, like this:

    \draw (0, 0) -- (2, 0) -- (1, 1.73205) -- cycle;

It is preffered method of creating a closed shape as using the precise
point can create some random artefacts.
```

We can also change the size of the figure using the "scale" option, like:

```
    \begin{tikzpicture}[scale = 3]
```

We can also specify which axis we want to scale with: xscale / yscale.

There is also an option to make the lines thicker with option used in
\draw command:

```
    \draw[ultra thick]
```

Or change the line style:

```
    \draw[dashed] / [dotted]
```

If we want to draw more lines with the same style we can just separate the commands

```
    \draw[dashed] (1, 0) -- (1, 1.73205) (0, 0) -- (1.5, 0.866)
        (2, 0) -- (0.5, 0.866);
```

We can also make the LaTeX calculate the lenghts for us by using
"{sqrt(3)}" Instead of the "1.73205".

To center the TikZ picture we use the "figure" env to wrap it in
and with that we can also use the [h] option to change its placement.

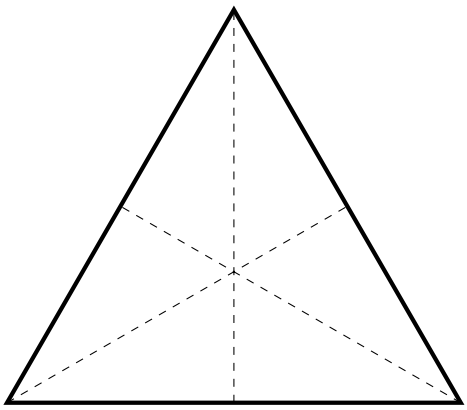

Figure 1: This is my first TikZ figure and it is beautiful.

3 Drawing Lines Exercise

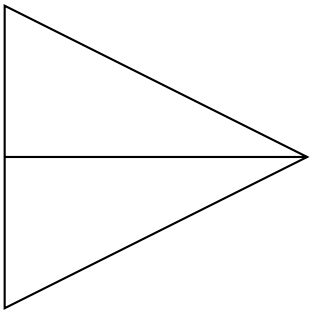

Figure 2: This is the 2nd TikZ figure!

Code used:

```
\begin{figure}[h]
    \centering
    \begin{tikzpicture}[scale=2]
        \draw[thick] (0, 0) -- (0, 2) -- (2, 1) -- cycle (0, 1) -- (2, 1);
    \end{tikzpicture}
    \caption{This is the 2nd TikZ figure!}
\end{figure}
```

4 Basic Shapes

```
To write the rectangle we specify the lower left corner and upper right
corner and connect them with "rectangle" instead of the "--" as we did before.
Like this:
    \draw[thick] (-3, -3) rectangle (3, 3);

We can also draw circles, where we specify the center point of the circle
and then connect it with options using "circle":
    \draw[thick] (0, 0) circle [radius = 3];
The option bracket can take more parameters than just a radius. We can also specify
    \draw[thick, red] (0, 0) circle [radius = 3];

To fill in the shape we can do it in 2 different ways - option or command:
    \draw[fill, green] (0, 0) circle [radius = 0.1];
    OR
    \fill[blue] (0, 0) circle [radius = 0.1];

We can specify the density of the dots / dashes with the "densely" or "loosely":
    \draw[ultra thick, blue, loosely dotted] (0, 0) -- (3, 0);

We can place text in the figures:
    \node[above] at (0, 0.1) {The center.};
    where:
        above => option specifying the placement relative to the point and
                    below, left, right can also be used,
        at (0, 0) => the point at which we want the text to be,
        {Text} => the caption we want to be displayed.

To use a caption which is long and needs the line break ("\\"),
we have to add another option, which is "align":
    \node[below, align = left] at (1.5, 0) {The radius of \\
                                            the circle is 3.};
```

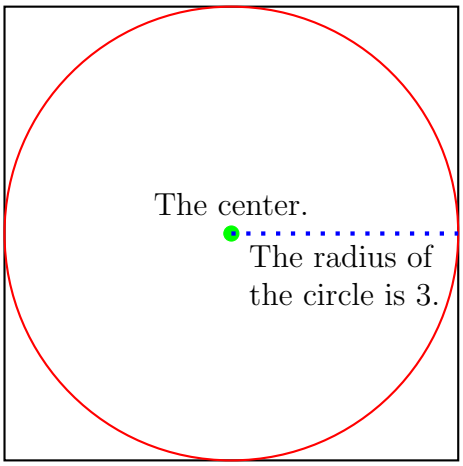

Figure 3: Here comes the 3rd figure. This one is more complicated!

5 Basic Shapes Exercise

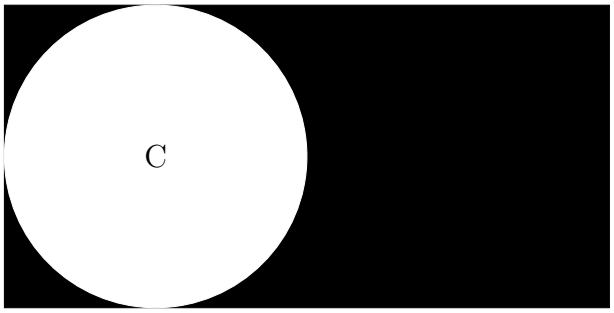

Figure 4: This is the 4th figure. Yay!

6 Plotting Functions

```
To draw a plot, we first have to draw the axis:
    \draw[->] (-0.1, 0) -- (10, 0); => the X axis
    \draw[->] (0, -0.1) -- (0, 5); => the Y axis
        The "->" option creates the arrow head of the axis

When we have the axis we can create the grid:
    \draw[gray, ultra thin] (-0.1, -0.1) grid (9.9, 4.9);
The 9.9 & 4.9 makes the grid without the last lines on right and at the top

Now to plot a Function we use:
    \draw[] plot (\x, \x/2);
BUT this will not work as we think it should. We have to specify the domain
of the function to make the plot fit in the grid we created:
    \draw[domain=0:9.9] plot (\x, \x/2);
BUT (2nd!) if we have more Functions with the same domain we can specify it
in the tikzpicture option.

Now that we have the Functions ploted we can label them. We do that by modifying
the draw command to include the node (same as we did with multiple lines):
    \draw[orange, thick] plot (\x, \x/2) node[above] {$f(x) = \frac{x}{2}$};
```

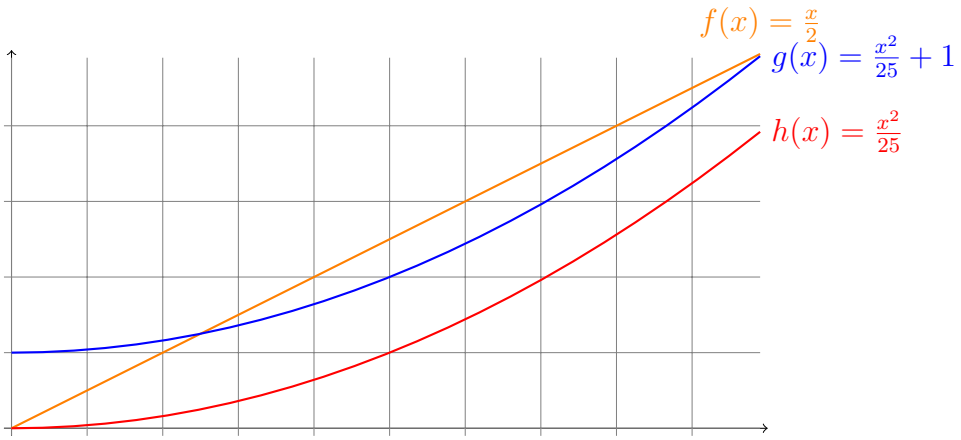

Figure 5: This is getting serious!

7 Plotting Functions Exercise

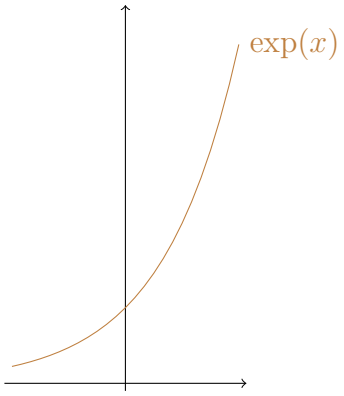

Figure 6: The exponential function denoted exp(\x) plotted from -1.5 to 1.5. We have used the color brown to color the graph

8 Plotting Curves

To note that the argument x of the cos function is in radians instead of the degrees we write it as `cos(\x r)`. The code:

```
\draw[->] (0, -2.7) -- (0, 2.7);
\draw[->] (-4.5, 0) -- (1.1, 0);
\draw[gray, ultra thin] (-4.5, -2.6) grid (0.9, 2.5);

\draw[domain=0:2*pi, smooth, pink, thick] plot ({2*(1-cos(\x r))
    * cos(\x r)}, {2*(1-cos(\x r)) * sin(\x r)});
% 2*pi/3
\draw[loosely dotted, thick] (0, 0) -- ({2*(1-cos(2*pi/3 r))
    * cos(2*pi/3 r)}, {2*(1-cos(2*pi/3 r)) * sin(2*pi/3 r)})
    node[above] {$\frac{3}{2} (-1, \sqrt{3})$};
\draw[fill] ({2*(1-cos(2*pi/3 r)) * cos(2*pi/3 r)},
    {2*(1-cos(2*pi/3 r)) * sin(2*pi/3 r)}) circle [radius = 0.05];
\draw[thick, dotted] ({2*(1-cos(2*pi/3 r)) * cos(2*pi/3 r)},
    {2*(1-cos(2*pi/3 r)) * sin(2*pi/3 r)})
    -- (-3/2, 0) node[below] {$-\frac{3}{2}$};
```

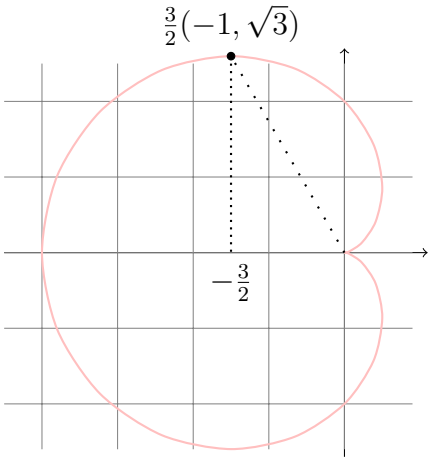

Figure 7: The cardioid $(2(1 - \cos(\theta))\cos(\theta), 2(1 - \cos(\theta))\sin(\theta))$.

9 For-Loops

Instead of using the:

```
    \draw[ultra thin] (1, 0) -- (1, -0.1) node[below] {$1$};
```

For each point by hand, we can use a for loop:

```
    \foreach \x in {1, ..., 6}{
        \draw[ultra thin] (\x, 0) -- (\x, -0.1) node[below] {$\x$};
    }
```

Full code:

```
\begin{tikzpicture}[domain = 0:2*pi, scale = 2]
    \draw[<->] (2*pi+0.1, 0) -- (0, 0) -- (0, 1.1);

    \draw[thick, teal, smooth] plot (\x, {cos(\x r)}) node[right] {$\cos(x)$};
    \draw[thick, purple, smooth] plot (\x, {sin(\x r)}) node[above right] {$\sin(x)

    \foreach \x in {1, ..., 6}{
        \draw[ultra thin] (\x, 0) -- (\x, -0.1) node[below] {$\x$};
    }

    \draw[ultra thin] (0, 1) -- (-0.1, 1) node[left] {$1$};
\end{tikzpicture}
```

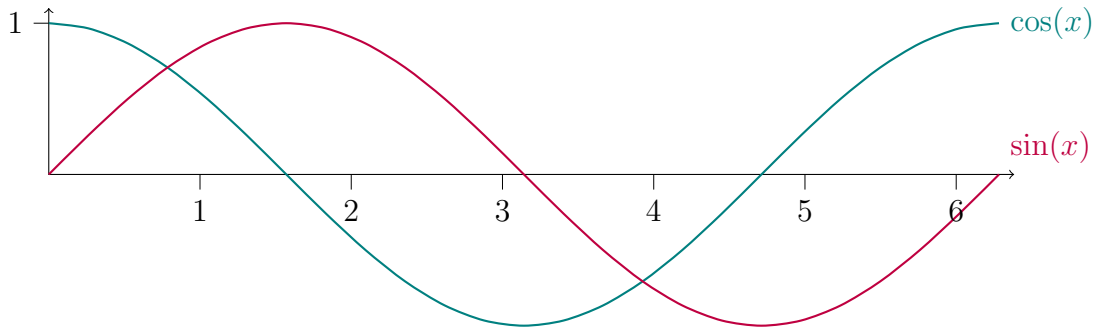

Figure 8: Automatic labeling of the points with a for loop.

10 For-Loops Exercise

The code:

```
\begin{tikzpicture}
    % the axis
    \draw[<->] (10.2, 0) -- (0, 0) -- (0, 10.2);

    % labeling the axis
    \foreach \x in {1, ..., 10}{
        % x:
        \draw[ultra thin] (\x, 0) -- (\x, -0.1) node[below] {$\x$};

        % y:
        \draw[ultra thin] (0, \x) -- (-0.1, \x) node[left] {$\x$};
    }
\end{tikzpicture}
```

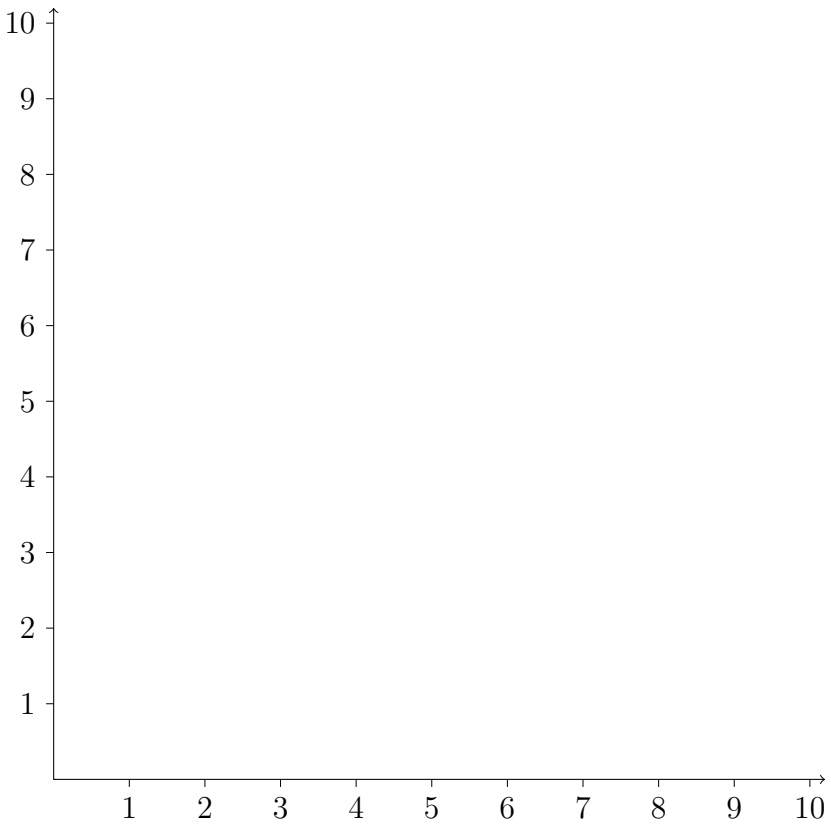

Figure 9: This is a for loop exercise plot

11 BONUS: Drawing 3D Images with TikZ

We can specify the points as coordinates and refer to them by their name.

We can also use a "calc" package to enable the vector calculations:
    \usetikzlibrary{calc}
The calculations have to be in math env => $$.

We can use something similar to "+=" when drawing the lines with the "--+":
    \draw[] (A) --+ (H);

The code:

```
\begin{tikzpicture}
    \coordinate (A) at ($2*(-1, 0, 0)$);
    \coordinate (B) at ($2*(1, 0, 0)$);
    \coordinate (C) at ($2*(0, 0, {sqrt(3)})$);
    \coordinate (H) at ($2*(0, 2, 0)$);

    % BASE
    \draw[] (B) -- (C) -- (A);
    \draw[dashed] (A) -- (B);

    % TOP
    \draw[] ($(B) + (H)$) -- ($(C) + (H)$) -- ($(A) + (H)$) -- cycle;

    % conecting the base and the top
    \draw[] (A) --+ (H);
    \draw[] (B) --+ (H);
    \draw[] (C) --+ (H);
\end{tikzpicture}
```

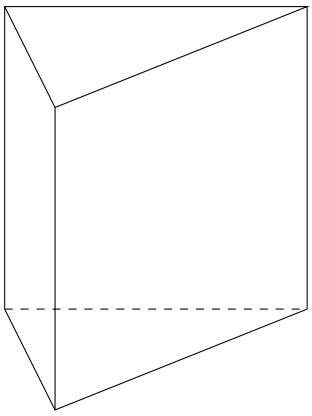

Figure 10: This is the final figure and it is in 3D!

11.1 Bonus 3D exercise

The code:

```
\begin{tikzpicture}
    \coordinate (A) at (0, 0, 0);
    \coordinate (B) at (2, 0, 0);
    \coordinate (C) at (3, 1, 1);
    \coordinate (D) at (1, 1, 1);

    \coordinate (H) at (0, 3, 0);

    \draw[] (A) -- (B) -- (C);
    \draw[dashed] (C) -- (D) -- (A);

    \draw[] ($(A) + (H)$) -- ($(B) + (H)$) -- ($(C) + (H)$) -- ($(D) + (H)$) -- cyc

    \draw[] (A) --+ (H);
    \draw[] (B) --+ (H);
    \draw[] (C) --+ (H);
    \draw[dashed] (D) --+ (H);
\end{tikzpicture}
```

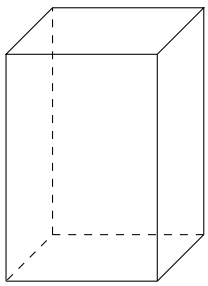

Figure 11: This is the last figure of the course. Such a beauty