Graphics in MathBook XML

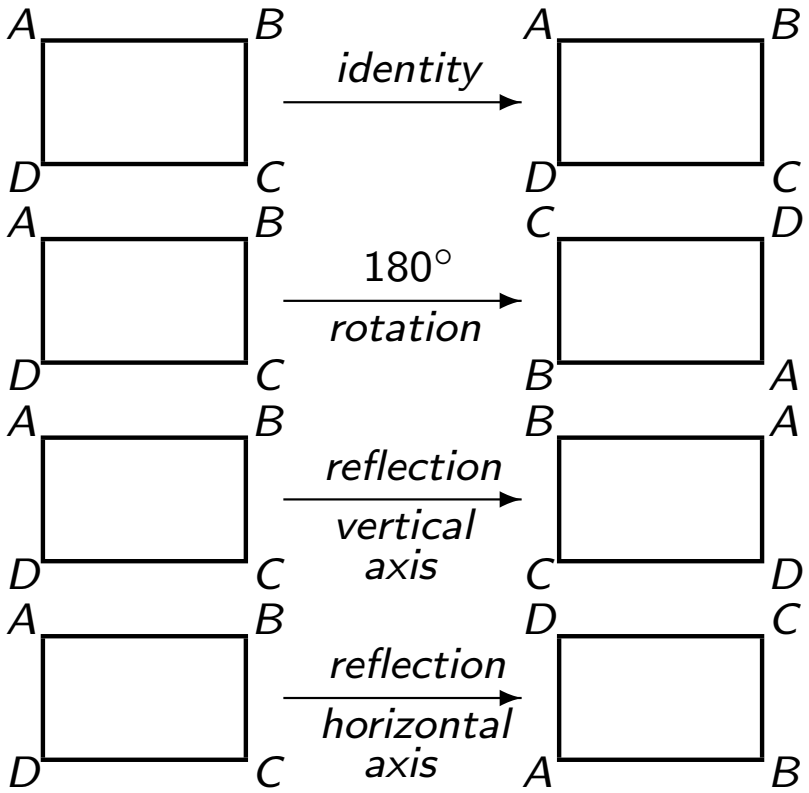
Thomas W. Judson
Stephen F. Austin State University

April 27, 2016

The LaTeX picture environment

- In the beginning graphics was created with the LaTeX picture environment or imported from a third party graphics package such as Adobe Illustrator.
- With the LaTeX picture environment, graphics are part of the LaTeX source.
- Third party programs produced EPS files that could be inserted into a LaTeX document.

Picture Environment Example



The Code for the Picture Environment

```
\begin{picture}(20,20)
\thicklines
\multiput(1,1)(0,5){4}{\framebox(5,3){ }}
\thinlines
\put(7, 2.5){\vector(1,0){6}}
\put(8.1,3){\scriptsize \it reflection}
\put(7.9,1.5){\scriptsize \it horizontal}
\put(9,.5){\scriptsize \it axis}
\put(7, 7.5){\vector(1,0){6}}
\put(8,8){\scriptsize \it reflection}
\put(8.25,6.5){\scriptsize \it vertical}
...
\end{picture}
```

Graphics Today

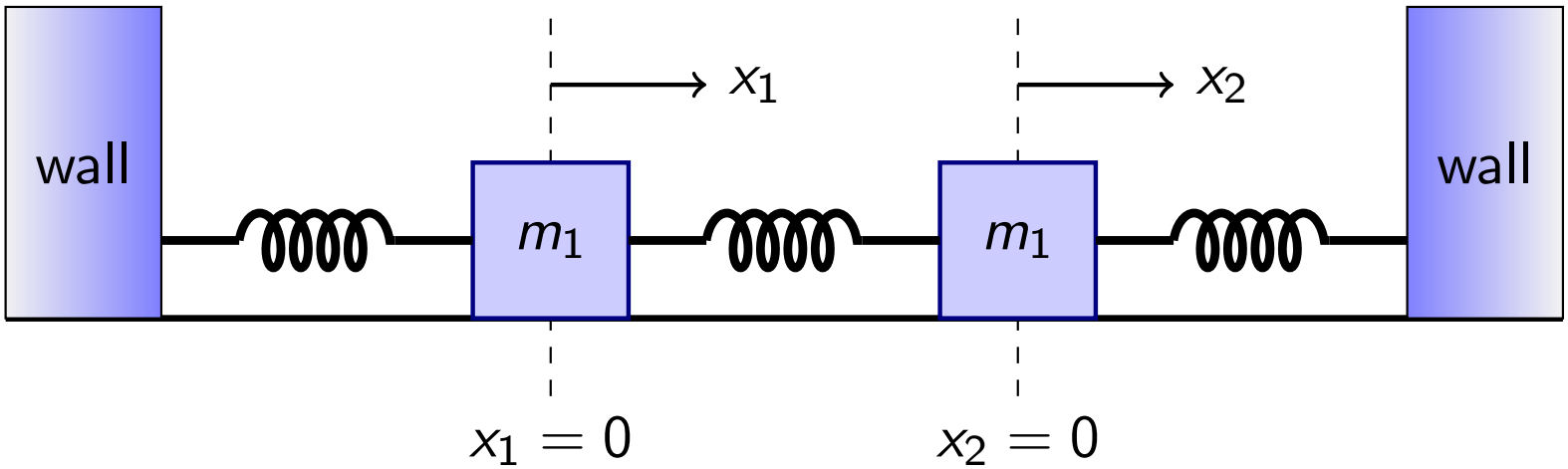
- Graphics formats include PDF, JPG, SVG, PNG, GIF, PS, EPS to name a few.
- TikZ and Asymptote can create graphics inside LaTeX
- Sage to insert graphics in LaTeX or MBX.
- Third party software packages such as Sage, xfig, Inkscape, Adobe Illustrator can be used to create a graphics file, which can be inserted into LaTeX or MBX.

TikZ

PGF/TikZ (orginally developed by Till Tantau) is a tandem of languages for producing vector graphics. PGF is a lower-level language, while TikZ is a set of higher-level macros that use PGF.

- `The code \tikz \draw (0pt,0pt) -- (20pt,6pt); yields and \tikz \fill[orange] (0,0) circle (1ex); provides inline graphics.`
- The code  yields and  provides inline graphics.
- Use `\begin{tikzpicture}...\end{tikzpicture}` for larger pictures.
- Many software packages (Inkscape, GeoGebra, MATLAB, R) will export graphics to TikZ.

A Double Spring-Mass System



The Code

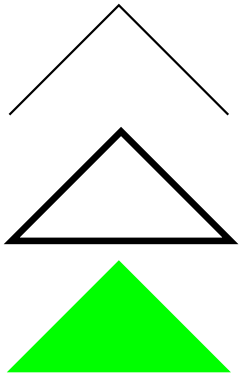
```
\begin{tikzpicture}[scale=0.5]
\draw[very thick] (-2,0) -- (18,0);
\draw[dashed] (5,-1) node[below] {$x_1 = 0$} -- (5,4);
\draw[dashed] (11,-1)node[below] {$x_2 = 0$}   -- (11,4);
\filldraw[fill=blue!20,draw=blue!50!black, thick] (4,0) -- (6,0) -
\draw[ultra thick] (0,1) -- (1,1) (3,1) -- (4,1);
\draw[snake=coil,segment length=5pt,segment amplitude=5pt,ultra th
\filldraw[fill=blue!20,draw=blue!50!black, thick] (10,0) -- (12,0)
\draw[ultra thick] (6,1) -- (7,1) (9,1) -- (10,1);
\draw[snake=coil,segment length=5pt,segment amplitude=5pt,ultra th
...
\end{tikzpicture}
```

The TikZ Coordinate System

- A TikZ picture is laid out on a grid.
- The coordinate system starts at the lower left hand corner of the canvas.
- The canvas is made large enough to hold the picture.
- The unit length is 1cm (other units possible).
- The basic element are paths and nodes.

Paths

A path is a series of straight and curved line segments connecting coordinate points.



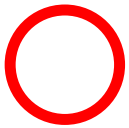
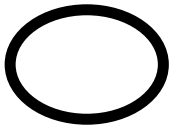
```
\draw (1,1) -- (2,2) -- (3,1);
```

```
\draw[very thick] (1,1) -- (2,2) -- (3,1) -- cycle;
```

```
\filldraw[green] (1,1) -- (2,2) -- (3,1) -- cycle;
```

Simple Shapes

It is possible to draw simple shapes such as rectangles, circles, and ellipses.

- 
  ```
  \draw (0, 0) rectangle (2, 1);
  ```
- 
  ```
  \draw[color=red] (0, 0) circle (.5);
  ```
- 
  ```
  \draw (0, 0) ellipse (.7 and 0.5);
  ```

Polar Coordinates and Curved Lines

- Polar coordinates are useful for drawing arcs, etc.

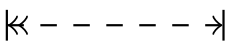
```
\draw[color=green] (40:1) arc (40:160:1);
```

- Curved lines can be drawn with Bézier curves

```
\draw (0, 0) .. controls(0.25, 0.5) and (0.75, -0.5) .. (1,0);
```

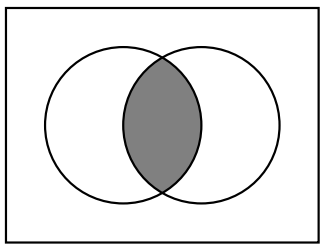
Arrows and Lines

We can draw arrows and lines in various styles and thicknesses.

- `\draw[->] (0, 0) -- (2,0);`
-  `\draw[dashed,thin,|<<->|] (0, 0) -- (2,0);`
-  `\draw[dotted,thick,o->|] (0, 0) -- (2,0);`
- `draw[densely dotted,very thick] (0, 0) -- (2,0);`
-  `\draw[loosely dashed,ultra thick] (0, 0) -- (2,0);`

Clipping and Scope

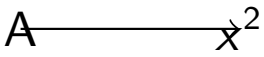
- After a \clip command, all subsequent drawings are clipped, only the parts inside the clipping region are drawn.
- Use the \scope environment to restrict the effect of clipping:



```
\begin{tikzpicture}[scale=0.5]
\draw (-2, 1.5) rectangle (2,
\begin{scope}
\clip (-0.5, 0) circle (1);
\clip ( 0.5, 0) circle (1);
\fill[color=gray] (-2,1.5)
    rectangle (2,-1.5);
\end{scope}
\draw (-0.5, 0) circle (1);
\draw ( 0.5, 0) circle (1);
\end{tikzpicture}
```

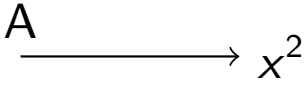
Nodes

- Nodes are added to paths after the path is drawn.




```
\draw[->] (0, 0) node {A} -- (2,0) node {$x^2$};
```

- Nodes have options

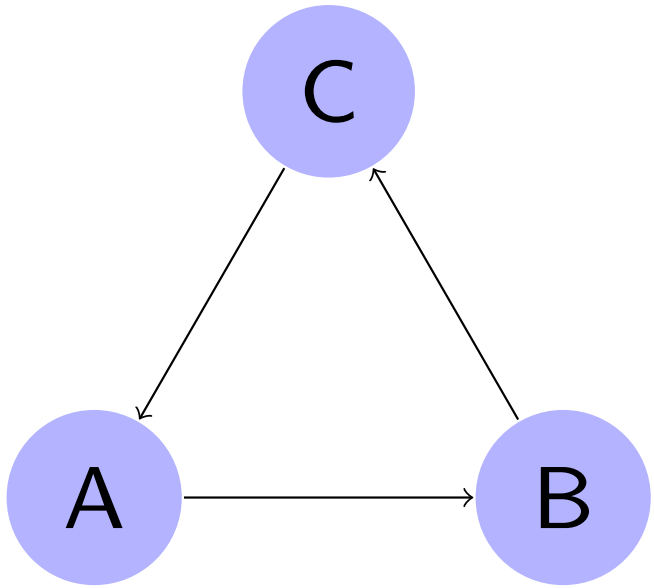



```
\draw[->] (0, 0) node[above] {A} -- (2,0) node[right] {$x^2$};
```

- Nodes can named for later reference and placed at a point independent of a path.

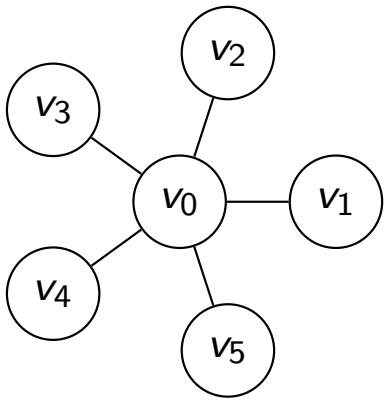
Naming and Navigating Nodes

```
\begin{tikzpicture}[scale=2,
   transform shape]
\tikzstyle{every node}
   = [circle,fill=blue!30]
\node (a) at (0, 0) {A};
\node (b) at +(0: 1.5) {B};
\node (c) at +(60: 1.5) {C};
B \foreach \from/\to
      in {a/b, b/c, c/a}
\draw [->] (\from) -- (\to);
\end{tikzpicture}
```



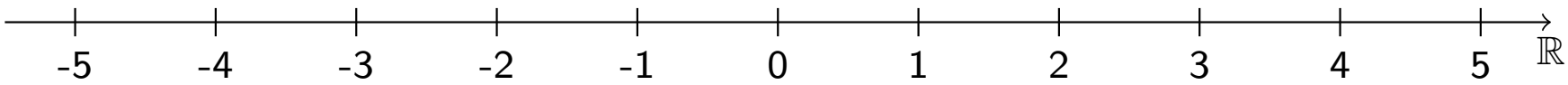
Computations in TikZ

```
\begin{tikzpicture}
  \tikzstyle{every node}=[draw,shape=circle];
  \node (v0) at (0:0) {$v_0$};
  \node (v1) at (   0:1) {$v_1$};
  \node (v2) at (  72:1) {$v_2$};
  \node (v3) at (2*72:1) {$v_3$};
  \node (v4) at (3*72:1) {$v_4$};
  \node (v5) at (4*72:1) {$v_5$};
  \draw (v0) -- (v1)
        (v0) -- (v2)
        (v0) -- (v3)
        (v0) -- (v4)
        (v0) -- (v5);
\end{tikzpicture}
```



Loops in TikZ

```
\begin{tikzpicture}
\draw[->] (-5.5,0) -- (5.5,0) node [below] {$\mathbb{R}$};
      \foreach \x in {-5,...,5}
          \draw (\x, 0.1) -- (\x, -0.1) node [below] {\x};
\end{tikzpicture}
```

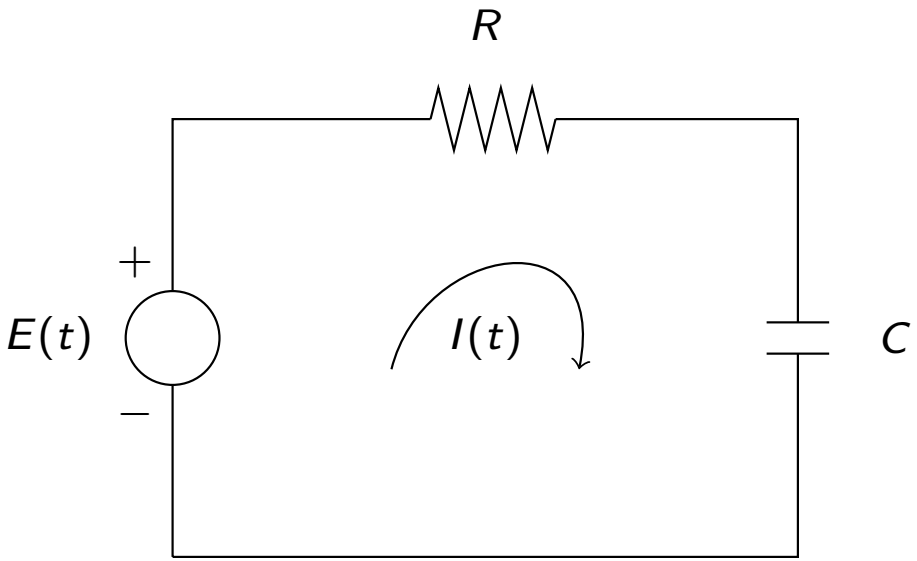


TikZ into MBX

The MBX source

```
<figure xml:id="figure-firstlook03-rc-circuit">
   <image xml:id="firstlook03-rc-circuit">
      <latex-image-code><![CDATA[
         \begin{tikzpicture}[scale=0.4]
         ...
         \end{tikzpicture}]]>
      </latex-image-code>
   </image>
   <caption>An RC circuit</caption>
</figure>
```

will produce the following graphic in the appropriate form (PDF for LaTeX and SVG for HTML).

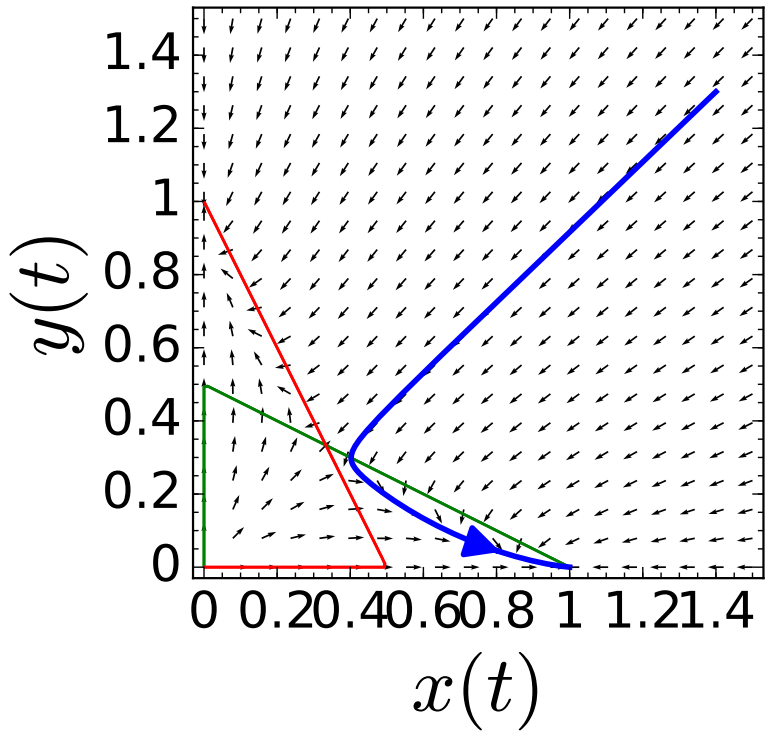


Sage Plots into MBX

The MBX source

```
<figure xml:id="figure-systems02-nullclines-competing-species-1">
   <image xml:id="systems02-nullclines-competing-species-1">
      <sageplot>
         a = 2
         xmin = 0
         ...
         vector_field = plot_vector_field( (f/n,g/n), (x,xmin,xmax
         xnullcline = implicit_plot(f, (x,xmin,xmax), (y,ymin,ymax
         ...
         result = vector_field + solution + starting_point + Q_arr
         result
      </sageplot>
   </image>
   <caption>Nullclines for the case <m>\beta = 2</m></caption>
</figure>
```

will produce the following graphic in the appropriate form (PDF for LaTeX and SVG for HTML).



Tips, Tricks, and Cautions

- Come up with an intelligent labelling scheme.
- To prevent conflicts with MBX reserved characters, use

```
<![CDATA[\begin{tikzpicture}
...
\end{tikzpicture}]]>
```

- Dont change units without a reason, use the `scale` and `transform shape` options of `tikzpicture`
- Be careful with XeLaTeX. It does not always work well with TikZ.
- Try out your plots in SageMathCloud first.

Resources

- Lots of examples at `http://www.texample.net/tikz/`
- Many of these examples came from
  `http://altermundus.fr/pages/downloads/remember_beamer.pdf`,
  `http://www.statistiker-wg.de/pgf/tutorials.htm`,
  `http://www.tug.org/pracjourn/2007-1/mertz/`,
  `http://tex.stackexchange.com`,
  and
  `http://www.math.uni-leipzig.de/~hellmund/LaTeX/pgf-tut.pdf`
- Till Tantau's PGF/TikZ manual
  `http://mirror.unl.edu/ctan/graphics/pgf/base/doc/pgfmanual.pdf`

Thanks for Listening

Thomas W. Judson, Associate Professor
Department of Mathematics and Statistics
Stephen F. Austin State University

EMAIL: judsontw@sfasu.edu

Slides available at http://faculty.sfasu.edu/judsontw/