Advanced LaTeX

Dan Parker* and David Schwein†

Spring 2015

Welcome to the second of the Brown Science Center's LaTeX Workshops! This workshop covers advanced material in LaTeX: complicated mathematical expressions, custom commands, tables, and bibliographies. We'll also cover individual packages that come in handy for various fields: the `siunitx` package for formatting quantities with units, the `mhchem` package for displaying chemical equations, the `amsthm` package for formatting theorems and proofs, and – last, but certainly not least – the magnificent Ti*k*Z package for making graphics.

1 Advanced Mathematics

Let's start off with some new mathematics commands. These are classified as *advanced* not because they're more difficult, but because they're more obscure. You'll need the `amsmath` and `amssymb` packages to use the commands in this section. Load them by writing

```
\usepackage{amsmath}
\usepackage{amssymb}
```

somewhere in the preamble of your document.

1.1 Aligned Objects: Matrices and `cases`

Last time we saw that LaTeX has environments for making aligned equations, the `align` and `align*` environments. In this section we'll describe two new aligned math environments, one for matrices and the other for case-by-case definitions.

$$\begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix} \qquad f(x) = \begin{cases} 1, & \text{if } x \in \mathbb{Q}, \\ 0, & \text{if } x \in \mathbb{R} \setminus \mathbb{Q}. \end{cases}$$

*danielericparker@gmail.com
†david_schwein@brown.edu

1.1.1 Matrices

Here's an example of a matrix.

```
\[
\begin{pmatrix}
a & b\\
c & d
\end{pmatrix}
\]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

What does this syntax mean? Every time LaTeX sees `&` it advances to the next column, and every time LaTeX sees `\\` it advances to a new row. The environment `pmatrix` makes matrices enclosed in parentheses. Other matrix environments include `matrix` (no parentheses), `bmatrix` (for `[ ]`), and `vmatrix` (for `| |`).

1.1.2 Dots

Typesetting an $n \times n$ matrix requires vertical, horizontal, and downward-sloped ellipses, as in the example at the beginning of this section.

```
\ldots   \cdots   \vdots   \ddots
```

$$\ldots \quad \cdots \quad \vdots \quad \ddots$$

The distinction between `\ldots` and `\cdots` is subtle: `\ldots` aligns the ellipsis with the bottom of the text while `\cdots` centers the ellipsis. Typographical style considerations dictate which of the two commands to use. For matrix entries and binary operations, use `\cdots`. For lists, use `\ldots`.

```
$x_1 + x_2 + \cdots + x_n$      $x_1,x_2,\ldots,x_n$
```

$x_1 + x_2 + \cdots + x_n$ $\qquad$ $x_1, x_2, \ldots, x_n$

1.1.3 `cases`

Here's an example of a case-by-case definition.

```
\[
|x| =
\begin{cases}
\phantom{-}x,
 & \text{if } x\geq 0,\\
-x, & \text{if } x<0.
\end{cases}
\]
```

$$|x| = \begin{cases} x, & \text{if } x \geq 0, \\ -x, & \text{if } x < 0. \end{cases}$$

If you understand matrices you should understand the `cases` environment: syntactically, the `cases` environment is a matrix that has exactly two rows.

A few other commands here require explanation. The command `\phantom{-}` adds an invisible $-$ to make the spacing come out right. (Try removing `\phantom{-}` and see what comes out.) The command `\text{...}` typesets its argument as regular text. Why did we add a space after `if`?

1.2 Accents and Font Styles in Math Mode

Some formulas require accents or special font styles. For example, mathematicians often indicate that the variable v stands for a vector by bolding it – like $\mathbf{v}$ – or by putting an arrow over it – like $\vec{v}$. The table below gives common accents and styles.

`\mathbf{v}`	`\mathbb C`	`\mathcal S`	`\mathfrak B`
$\mathbf{v}$	$\mathbb{C}$	$\mathcal{S}$	$\mathfrak{B}$
`\dot y`	`\ddot y`	`\overline{a+b}`	`\vec v`
$\dot{y}$	$\ddot{y}$	$\overline{a+b}$	$\vec{v}$
`\hat x`	`\widehat{x+y}`	`\tilde x`	`\widetilde{x+y}`
$\hat{x}$	$\widehat{x+y}$	$\tilde{x}$	$\widetilde{x+y}$

Unfortunately, `mathbf` does not work with Greek characters. To make bold Greek characters, load the `bm` (bold math) package and use the newly provided command `\bm{...}`.

1.3 Spacing

LaTeX allows fine control over spacing in math mode.

```
|\!|  |\,|  |\>|  |\;|  |\quad|  |\qquad|  |\phantom{a+b}|
```

The command `\!` adds *negative* space. Use it if LaTeX adds too much space to your formula and you want to tighten it.

`x^2/2`	$x^2/2$
`x^2\!/2`	$x^2/2$

The command `\phantom{ }` adds as much white space as its argument takes up. We already encountered it in the section on the `cases` environment.

Most of the spacing changes you'll make are subtle, yet mark the difference between nice math and ugly math. The more time you spend typing and reading math, the better an eye you'll develop for these sorts of spacing issues. In the meantime, we recommend placing `\,` before dx in integrals. Compare an integral with `\,` to one without it.

`\int_0^1 f(x)g(x)dx`	$\int_0^1 f(x)g(x)dx$
`\int_0^1 f(x)g(x)\,dx`	$\int_0^1 f(x)g(x)\,dx$

1.4 Equation Numbering

It's often useful to number equations that are particularly important so that they may be referred to later.

```
\begin{equation}
x^n + y^n \neq z^n
\label{fermat}
\end{equation}
```

$$x^n + y^n \neq z^n \quad (1)$$

The command `\label{fermat}` tells LaTeX that the name of the equation is `fermat`. We can reference it in the text using the command `\eqref`, which is like `\ref` but with added parentheses.

```
I proved \eqref{fermat},
but the margin
was too small
to contain it.
```

I proved (1), but the margin was too small to contain it.

To number a series of aligned equations, use the `align` environment. If you do not want to number a specific line in a series of aligned equations, place the command `\nonumber` on that line.

```
\begin{align}
z^n &= x^n + y^n
\nonumber\\
&\neq (x+y)^n
\end{align}
```

$$\begin{aligned} z^n &= x^n + y^n \\ &\neq (x+y)^n \end{aligned} \quad (2)$$

1.5 Delimiters

Some expressions are so large that placing them in standard-size parentheses would look awful.

```
\[
(\int_0^1 f(x)\,dx)^2
\]
```

$$(\int_0^1 f(x)\,dx)^2$$

To make the sizes come out right, use `\left` before the left parenthesis and `\right` before the right parenthesis.

```
\[
\left(\int_0^1 f(x)\,dx\right)^2
\]
```

$$\left(\int_0^1 f(x)\,dx\right)^2$$

The `\left` and `\right` commands also work for other delimiters, including the invisible delimiter `.` (a period).

```
\[
\left\|\frac{x}{y}\right\|
+ \left.\frac{z}{z+1}\right|_{z=i}
\]
```

$$\left\|\frac{x}{y}\right\| + \left.\frac{z}{z+1}\right|_{z=i}$$

1.6 Making Math Operators

LaTeX comes with predefined commands for the most commonly used math operators, such as log and sin. The reason we need special commands for these functions is that LaTeX italicizes text in math mode: $sin(x)$ is incorrect. But sometimes you may need to use an operator that has not already been defined. The proper way to do this is with the `\operatorname` command.

```
The subspace $A = \operatorname{span}\{x_1,x_2\}$
```

The subspace $A = \operatorname{span}\{x_1, x_2\}$

2 Making Custom Commands

In long documents, you may find yourself typing similar sequences of text over and over again. LaTeX provides a command that *defines new commands*. This is one of LaTeX's most useful features, but because custom commands are the most conceptually difficult subject this workshop will cover, they call for special explanation.

2.1 Custom Commands without Arguments

Let's look at a basic example and go through it very carefully to see how it works.

```
\newcommand{\ftc}{Fundamental Theorem of Calculus}
\ftc \ftc .

We can then apply the \ftc{} and conclude that...
```

Fundamental Theorem of CalculusFundamental Theorem of Calculus.
We can then apply the Fundamental Theorem of Calculus and conclude that...

What's going on here? The first line defines a new command whose name is `\ftc`; when the command is used, LaTeX prints "Fundamental Theorem of Calculus". The `{}` appearing after `\ftc` adds an interword space; as the example above shows, without `{}`, LaTeXignores space following the command. This particular "feature" of LaTeX, gobbling up space after commands, is a universal bugbear among LaTeX users.[1]

Conceptually speaking, typing the command `\ftc` tells LaTeX to copy the text `Fundamental Theorem of Calculus` to memory and, whenever it sees `\ftc`, paste in `Fundamental Theorem of Calculus`.

You can use `\newcommand` anywhere in the document, but the command it defines will only work *after* the place where `\newcommand` appears. It's therefore good style to place any `\newcommand` in the preamble of the document.

[1] Another way to fix this problem is with the `xspace` package. See the package documentation at `ctan.org/pkg/xspace`.

Also, `\newcommand` cannot make commands that already exist: for instance, `\newcommand{\textit}{hi}` will result in an error message. There are also subtle rules regarding command names: in general, command names should contain only letters.

2.2 Custom Commands with Arguments

The commands we defined in the last section did not take any arguments. In this section, we'll see how to define commands that have multiple arguments, like regular LaTeX commands.

Suppose we were typing up a math problem which involved taking many partial derivatives, such as

$$\frac{\partial f}{\partial x} - i\frac{\partial f}{\partial y}$$

It would be useful to have a command that automatically makes partial derivatives, instead of typing out `\frac{\partial ...}{\partial ...}` many times. To make commands taking arguments, `\newcommand` has an optional argument specifying the number of arguments taken by the new command.

```
\newcommand\pd[2]{\frac{\partial #1}{\partial #2}}
\[
\pd{f}{x}
\]
\[
\pd{\Lambda}{\nu} + \pd{\Omega}{\mu}
\]
```

$$\frac{\partial f}{\partial x}$$

$$\frac{\partial \Lambda}{\partial \nu} + \frac{\partial \Omega}{\partial \mu}$$

Every time LaTeXsees a `#1`, it replaces it with the first argument; every time LaTeXsees a `#2`, it replaces it with the second argument. For instance, the following command repeats its argument three times.

```
\newcommand{\rep}[1]{#1 #1 #1}
\rep{a} \rep{eggplant}
```

a a a eggplant eggplant eggplant

3 Tables

Making a table in LaTeX is very similar to making a matrix, with a few slight modifications. There are a few ways to make a table in LaTeX; the one we'll give uses the package `booktabs`, which makes tables of high typographical quality.[2]

[2]Specifically, tables should have as few lines as possible. The `booktabs` package only has three lines in each table.

```
\usepackage{booktabs}
...
\begin{center}
\begin{tabular}{ l r }
\toprule
City & Population \\
\midrule
Providence & 178,024 \\
Warwick    &  82,672 \\
Cranston   &  80,387 \\
Pawtucket  &  71,148 \\
\bottomrule
\end{tabular}
\end{center}
```

City	Population
Providence	178,024
Warwick	82,672
Cranston	80,387
Pawtucket	71,148

Our table consists of two environments: the `center` environment, which centers the table on the page, and the `tabular` environment, which makes the table. The `tabular` environment takes a second argument, in our case `l r`, which tells LaTeX how many columns the table will have and how to align them. The letter `l` is for left-alignment, the letter `r` is for right-alignment, and the letter `c`, which we did not use, is for centering. Finally, the commands `\toprule`, `\midrule`, and `\bottomrule` tell LaTeX where to place the horizontal lines separating the descriptors and the columns.

To make a table with a caption and a table number, use the `table` environment instead of the `center` environment. The `table` environment is completely analogous to the `figure` environment, which we discussed last time.

Tables can be very complicated, and for this reason there are many LaTeX packages that provide special table functionality, such as the `colortbl` environment for colored tables. See the LaTeX wikibook for a complete list.

4 Miscellany

4.1 Adjusting Margin Sizes

The default margins in LaTeX documents are quite wide, and for good reason: studies have shown that text is easier to read when it has fewer letters per line, which is why newspaper columns are so narrow. Nonetheless, many LaTeX users prefer to shrink the margins. If you are one of these people, we recommend using the `geometry` package with the `margin` option.

```
\usepackage[margin=2cm]{geometry}
```

4.2 Optional Arguments for `\documentclass`

When we told you about the `\documentclass` command, we left out functionality: `\documentclass` takes optional arguments. For example, writing `\documentclass[twocolumn,12pt]{article}` at the beginning of your TeX

file will typeset the document in two columns at 12-point font. See the LaTeX wikibook chapter on document structure for a complete list of optional parameters.

5 Bibliography Management and BibTeX

BibTeX is an extension to LaTeX for managing references, made to be easy to use and highly convenient. Citations and references are generally bothersome to do by hand, for two reasons.

1. Everyone wants a very specific, but slightly different format for citations (e.g. MLA, Chicago).
2. Citations are often numbered by order of appearance, so any time you insert a new citation, all the numbers change.

BibTeX has two elegant solutions to these problems:

1. Citation formatting is automatic.
2. Citation numbering is automatic.

This is one of the best examples of the LaTeX philosophy; BibTeX does all the formatting for you, so you can focus on content.

There are three steps to using BibTeX: creating a bibliography file, citing with the `\cite` command, and typesetting for BibTeX.

5.1 Creating a Bibiography File

BibTeX requires an external bibliography. There are several external tools to create these quickly, but it's useful to understand how they work before taking shortcuts. This takes the form of a file in the same folder as your `.tex` file that has the extension `.bib`. It's customary to call it `references.bib` or `citations.bib`, or something along those lines. The `.bib` files contains the information about each thing you want to cite.

BibTeX supports many types of bibliographic entries, including `book`, (journal) `article`, (conference) `proceedings` and many more. Let's look at a few examples:

```
@book{rudin1964principles,
  title={Principles of mathematical analysis},
  author={Walter Rudin},
  volume={3},
  year={1964},
  publisher={McGraw-Hill New York}
}

@article{bardeen1957theory,
```

```
  title={Theory of superconductivity},
  author={Bardeen, John and Cooper, Leon N and Schrieffer, J Robert},
  journal={Physical Review},
  volume={108},
  number={5},
  pages={1175},
  year={1957},
  publisher={APS}
}
```

Each entry starts with `@` and the type of the entry. Next comes the name of the specific entry, a unique identifier used to cite that entry. For the second entry above, the name is `bardeen1957theory`. After that, there are several fields that give the information needed to construct the reference. For `book`, the required fields are `title, author, publisher, year`. For `article`, it's also necessary to include `journal` and highly recommended to provide `year, volume, page`, so that a reader could easily find the article in question. A list of entry types and what information is required for each one is available at the LaTeX wikibook. It is absolutely necessary to include a comma after every field.

5.1.1 External Bibliography Management Systems

No one wants to bother typing all of this for each citation, but luckily, there are easier ways.

Google Scholar If you click "Cite" in Google Scholar you have the option to receive a citation for that work in several formats, one of which is BibTeX. These days, most journals have all their articles online and have a similar "generate citation" button for each article.

JabRef A (free, open source) program for managing BibTeX files. JabRef Works on Windows, Macs, and Linux, and has a nice interface for adding entries. It also gives a way to search online journal databases and turn the results into bibliography entries.

BibDesk A tool similar to JabRef, but slightly more polished. Bibdesk only works on Macs, and is included with TeXShop.

5.2 Citing in the Text

Citing references is very easy. Let's look at an example.

```
A standard textbook on analysis is \cite{rudin1964principles}.
The theory of low-temperature
superconductivity was invented in 1957 \cite{bardeen1957theory}.

\bibliographystyle{plain}
\bibliography{references}
```

A standard textbook on analysis is [2]. The theory of low-temperature superconductivity was invented in 1957 [1].

References

[1] John Bardeen, Leon N Cooper, and J Robert Schrieffer. Theory of superconductivity. *Physical Review*, 108(5):1175, 1957.

[2] Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1964.

To cite a bibliographic entry, use the `\cite` command with the name of that entry. The command `\bibliographystyle{plain}` says how the citations should be formatted. In addition to the `plain` style, which sorts references by author and use square brackets with numbers, there are many other styles available, such as `unsrt`, which lists references in order of appearance, or `alpha`, which gives citations like `[Rud64]`. The `natbib` package gives many more citation formats, including `apa`. For MLA-style citations, try the `biblatex-mla` package.

The command `\bibliography{references}` tells LaTeX which `.bib` file to use. Make sure to place the `.bib` file in the same folder as the `.tex` file.

5.3 Running BibTex

Since BibTeX has an external file, you need to typeset a little differently. After any changes to the BibTeX file, you must go through the following sequence:

1. Typset LaTeX normally.
2. Typeset BibTeX. This is normally a separate button or menu option.
3. Typset LaTeX normally.
4. Typset LaTeX normally *again*.

For the computer-savvy, there are ways to automatically perform all four typesets in a single command. This involves writing some sort of script.

5.4 URLs and DOIs In Citations

Almost all journals are online now, so it's usually more useful to give the DOI or a link to an article than the journal issues. To do this, load the `hyperref` and `doi` packages, and add the `url` and `doi` packages to your bibtex entries. As an example:

```
@article{bardeen1957theory,
  title={Theory of superconductivity},
  author={Bardeen, John and Cooper, Leon N and Schrieffer, J Robert},
  journal={Physical Review},
  volume={108},
  number={5},
  pages={1175},
  year={1957},
  publisher={APS}
  doi = {10.1103/PhysRev.108.1175},
  url = {http://link.aps.org/doi/10.1103/PhysRev.108.1175}
}
```

6 The `siunitx` Package

Units are something of a pain to typeset in normal LaTeX. Suppose we had measured the frequency of an oscillation to be 3.0×10^{-6}Hz. In math mode, we would write `$3.0 \times 10^{-6} \text{Hz}$`. Not only is this long, but the spacing between the number and the unit is terrible! The `siunitx` package offers a better alternative.

`\SI{3e-6}{\hertz}`	$3 \times 10^{-6}\,\mathrm{Hz}$

As its name suggests, the `siunitx` package supports all SI units. It also supports standard prefixes, such as giga- and femto-.

`\SI{3}{\giga\hertz}`	$3\,\mathrm{GHz}$
`\SI{3}{GHz}`	$3\,\mathrm{GHz}$
`\SI{19}{\meter\per\second}`	$19\,\mathrm{m\,s^{-1}}$
`\SI[per-mode=fraction]{19}{\meter\per\second}`	$19\,\frac{\mathrm{m}}{\mathrm{s}}$
`\SI{54.1}{kg.m/s^2}`	$54.1\,\mathrm{kg\,m/s^2}$
`\SI{32932e-3213}{\micro F}`	$32\,932 \times 10^{-3213}\,\mu\mathrm{F}$
`\SI{3.2}{kg/\nano\meter\squared}`	$3.2\,\mathrm{kg/nm^2}$
`\SI{8.9}{\kilo\gram\tothe{12}}`	$8.9\,\mathrm{kg^{12}}$
`\SI{27}{\degreeCelsius}`	$27\,^\circ\mathrm{C}$

By default, reciprocals of units are displayed as negative powers; to display them as fractions, import the package with the command

`\usepackage[per-mode=fraction]{siunitx}`.

The `siunitx` package contains several other useful commands: `\num` works just like the first argument of `\SI` by itself, and `\si` works like the second part.

`\num{3.0}`	3.0
`\num{3.0e5}`	3.0×10^{5}
`\num{3+2i}`	$3 + 2\mathrm{i}$
`\si{\kilo\gram\per\tesla}`	$\mathrm{kg\,T^{-1}}$
`\si{\lumen\per\radian}`	$\mathrm{lm\,rad^{-1}}$

Additionally, there is a command for typing numbers with the same units. Numbers are separated by semi-colons, and commands and an "and" are inserted automatically.

```
\SIlist{1;2.3;-534;7.3e-5}{\joule\per\second}
```
$1\,\mathrm{J\,s^{-1}}$, $2.3\,\mathrm{J\,s^{-1}}$, $-534\,\mathrm{J\,s^{-1}}$ and $7.3 \times 10^{-5}\,\mathrm{J\,s^{-1}}$

For more information on this package, check out its manual on CTAN at `ctan.org/pkg/siunitx`.

7 The `mhchem` Package

LaTeX's math mode is meant to write mathematics and is thus rather ill-suited for typing chemical equations, mostly because atomic symbols should not be italicized. The `mhchem` package provides the new command `\ce`, for typing chemical equations.

`\ce{H2SO4}`	`\ce{AgCl2-}`	`\ce{CrO4^2-}`	`\ce{^{235}_{92}U}`
$\mathrm{H_2SO_4}$	$\mathrm{AgCl_2^-}$	$\mathrm{CrO_4^{2-}}$	$\mathrm{^{235}_{92}U}$

We can also use `\ce` to make chemical equations.

```
\ce{CO2+C -> 2CO}\\[0.4em]
\ce{H+ + OH- <=>> H20}
```
$\mathrm{CO_2^+ C} \longrightarrow \mathrm{2\,CO}$
$\mathrm{H^+ + OH^-} \rightleftharpoons \mathrm{H_{20}}$

If you want to learn more, head to the manual at `ctan.org/pkg/mhchem`.

8 The `amsthm` Package

Professional mathematics publications often separate theorems, definitions, and other important information from the text, like so.

Theorem 1 (Wiles)**.** *If $n \geq 3$, the equation $x^n + y^n = z^n$ has no nontrivial integer solutions.*

Doing this in LaTeX requires the `amsthm` package, which gives us commands to define environments for theorems, propositions, and the like.

To make a theorem, you have to first tell `amsthm` to define a new environment. The command to use is `\newtheorem`, which takes two arguments. The first is

the name of the new environment, and the second is what should be displayed. For instance, placing `\newtheorem{thm}{Theorem}` in the preamble allows us to make a theorem using the newly defined `theorem` environment.

```
\newtheorem{thm}{Theorem}
...
\begin{thm}
Every finite $p$-group
has nontrivial center.
\end{thm}
```

Theorem 2. *Every finite p-group has nontrivial center.*

To prevent LaTeX from numbering the environment, use the `\newtheorem*` command.

```
\newtheorem*{prop}{Proposition}
...
\begin{prop}
Every finite $p$-group
has nontrivial center.
\end{prop}
```

Proposition. *Every finite p-group has nontrivial center.*

Every environment created in this way takes an optional argument, which indicates text to be places in parentheses.

```
\begin{thm}[Feit-Thompson]
Every group
of odd order
is solvable.
\end{thm}
```

Theorem 3 (Feit-Thompson)**.** *Every group of odd order is solvable.*

8.1 Proofs

The `amsthm` package comes with one predefined environment, the `proof` environment, which does just what you expect.

```
\begin{proof}
A trivial exercise
for the reader.
\end{proof}
```

Proof. A trivial exercise for the reader. □

If you end a proof with a displayed equation, the Halmos box[3] may not go in the right position. To place it in a specific spot, use the command `\qedhere`.

```
\begin{proof}
Just observe that
\[
1 + 1 = 2.
\]
\end{proof}
```

Proof. Just observe that

$$1 + 1 = 2.$$

□

[3]The Halmos box is the symbol □.

```
\begin{proof}
Just observe that
\[
1 + 1 = 2. \qedhere
\]
\end{proof}
```

Proof. Just observe that

$$1 + 1 = 2. \qquad \square$$

8.2 Theorem Styles

There are three possible styles for environment defined using `amsthm`: `plain` (the default), `definition`, and `remark`. To specify a style, group `\newtheorem` commands into blocks according to the style you would like them to have, then place `\theoremstyle{<style>}` before each block. For instance, part of your preamble might look like

```
\theoremstyle{theorem}
\newtheorem{theorem}{Theorem}
\newtheorem{proposition}{Proposition}
\newtheorem{lemma}{Lemma}

\theoremstyle{definition}
\newtheorem{definition}{Definition}
\newtheorem{exercise}{Exercise}

\theoremstyle{remark}
\newtheorem*{remark}{Remark}
```

There are also ways to define your own style. See the package documentation for more information.

9 The TikZ Package

Ti*k*Z is a package for drawing diagrams within LaTeX.[4] It's extremely powerful, capable of producing about any figure you could want.[5] Our goal is to introduce some of the basics of Ti*k*Z, so that you can get started making figures. For a more comprehensive introduction, see the superb package documentation at CTAN.[6]

9.1 Getting Started

First, include the package: `\usepackage{tikz}`. Figures in Ti*k*Z are made in the environment `tikzpicture`; this is the only environment we'll use.

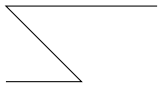
[4] This section is based on notes by Spencer Gordon.

[5] To get an idea just how powerful Ti*k*Z is, see the Ti*k*Z examples page at `texample.net/tikz/examples/`.

[6] `ctan.org/pkg/pgf`

One thing that makes Ti*k*Z a little confusing at first is that it has an entirely different syntax from LaTeX. Here's a simple example of Ti*k*Z's syntax.

```
\begin{tikzpicture}
\draw (0, 0) -- (1, 0) --
      (0, 1) -- (1, 1) -- (2, 1);
\end{tikzpicture}
```



You can read the command above as telling Ti*k*Z to draw a line from $(0, 0)$ to $(1, 0)$, then $(1, 0)$ to $(0, 1)$, and so on. The coordinates here are standard Cartesian xy coordinates. Every Ti*k*Z figure comes with an invisible coordinate grid.

A Ti*k*Z command begins with a LaTeX command, like `\draw`, `\node`, or `\fill`, followed by a sequence of instructions, and terminated with a semicolon. Let me say that again, because it constitutes the most common mistake beginning Ti*k*Z users make:

Every Ti*k*Z drawing command ends with a semicolon.

Ti*k*Z has a shorthand for creating simple pictures, which we'll use in these notes to save space. The command `\tikz ...` is short for the more verbose `\begin{tikzpicture}...\end{tikzpicture}`, provided that we only give `\tikz` a single command. For instance, the following commands will produce the same output:

```
\begin{tikzpicture}
\draw (0, 0) -- (1, 0);
\end{tikzpicture}
```

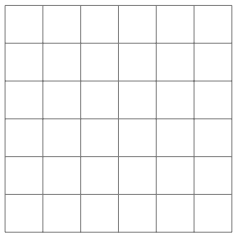
```
\tikz \draw (0, 0) -- (1, 0);
```

The default coordinate system has the positive x-axis pointing right, the positive y-axis pointing up, and one unit corresponding to one centimeter. To work with different units, you can simply add the units to the coordinates, like `(2in,0.3pt)`. But it's probably a better idea to resize the figure once it's finished; we'll see how to do this in Section 9.3.

9.2 Styles

If you don't yet have experience with computer graphics programming, it may be difficult to visualize the coordinates needed to create diagrams. Ti*k*Z has a built-in command to ease the transition.

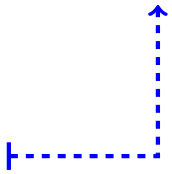
```
\tikz \draw[help lines, step=0.5]
        (0, 0) grid (3, 3);
```



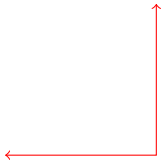
There are two differences between this `\draw` command and the previous one. First, we're using the `grid` instruction to draw a grid with corners at $(0, 0)$ and $(3, 3)$, as opposed to the `--` instruction we used earlier.

Second, the `\draw` command now takes optional arguments, enclosed in `[...]`. The argument `help lines` tells Ti*k*Z to color the lines gray, and the argument `step=0.5` tells Ti*k*Z to divide the grid in 0.5 cm increments. These are examples of a *style*: options given to Ti*k*Z which control parameters for drawing, such as width or color. The following figures, which show the same path in different styles, should give you a sense of the possibilities.

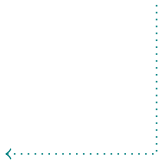
```
\tikz \draw[color=blue, ultra thick,
        |->, dashed]
        (0, 0) -- (2, 0) -- (2, 2);
```



```
\tikz \draw[color=red, thin, <->]
        (0, 0) -- (2, 0) -- (2, 2);
```



```
\tikz \draw[color=teal, semithick,
        <-, dotted]
        (0, 0) -- (2, 0) -- (2, 2);
```

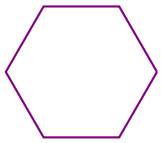


9.3 Coordinate Systems

There are two alternatives to the absolute Cartesian coordinate system we presented before.

First, polar coordinates.

```
\tikz \draw[color=violet, thick]
        (0:1) -- (60:1) --
        (120:1) -- (180:1) --
        (240:1) -- (300:1) -- (0:1);
```



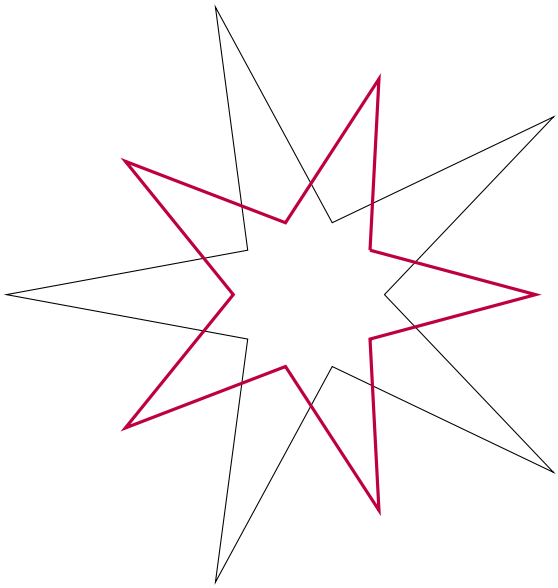
Polar coordinates are specified with the syntax `(angle:radius)`, and can be freely mixed with cartesian coordinates.

Second, relative coordinates. Suppose I wanted to draw a square 3 units above a triangle. Instead of adding 3 to every y-coordinate in the square, I can position one corner of the square, then tell Ti*k*Z to position the other corners relative to that corner.

```
\begin{tikzpicture}[scale=0.5]
\draw (0, 0) -- (0, 1) --
      (1, 0) -- (0, 0);
\draw (0, 2) -- ++(1, 0) --
    ++(0, 1) -- ++(-1, 0) --
    ++(0, -1);
\end{tikzpicture}
```

The optional argument `scale=0.5` tells Ti*k*Z to scale the figure by 50%.

As an exercise, recreate the following figure.



9.4 Curves and Colors

Now that you have a decent grasp on line drawings, we'll advance to more sophisticated functionality.

We can draw a curved path as follows.

```
\tikz \draw (0, 0) to [out=135, in=100] (1, 1);
```

The `out` option specifies the angle at which the curve should leave the starting point and the `in` option specifies the angle at which the curve should arrive at the end point.

In addition to the very general `in`/`out` syntax, Ti*k*Z provides shortcuts for common shapes and patterns. We've already seen one, the `grid` shape. We can draw a circle as follows.

```
\tikz \draw[orange] (1, 1) circle (0.5);
```

The first point is the center, and the second number in parentheses is the radius. We can quickly draw a rectangle as follows.

```
\tikz \draw[brown] (0, 0) rectangle (2, 1);
```

Here the first point is one corner of the rectangle and the second point is the opposite corner.

To color the inside of one of these shapes, use the `\fill` command.

```
\tikz \draw[blue,fill=green] (1, 1) circle (0.5);
```

9.5 Worked Example: Bullseyes

Rather than go through any more isolated pieces of functionality, we're going to work through a larger example and introduce new concepts as needed.

Here's the diagram I want to create.

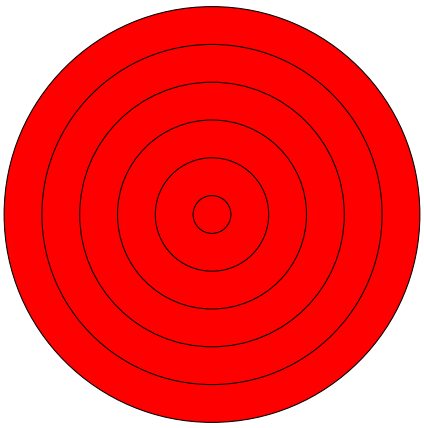
It seems clear that writing this using the `\draw ... circle ...;` command would be really annoying, and if I wanted to adjust anything, I'd have to make a lot of changes. Ti*k*Z offers a feature to perform repetitive tasks, which will be familiar to those of you with some programming experience: for loops. Here's the code I used to make the bullseye.

```
\begin{tikzpicture}[scale=0.5]
\foreach \r in {5,4,...,0}{
  \draw[fill=red,draw=black] (0,0) circle[radius=\r+0.5];
  \draw[fill=white,draw=black] (0,0) circle[radius=\r];
};
\end{tikzpicture}
```

The interesting part is the loop `\foreach \r in {5,4,...,0}`. It tells Ti*k*Z to take the code below the loop, make one copy for each item in the list $5, 4, \ldots, 0$ and replace `\r` with the current item in the list each time. We used the value of `\r` in our for-loop to adjust the radius of each circle that we draw, and that we do simple arithmetic to make the red circle bigger than the white circle.

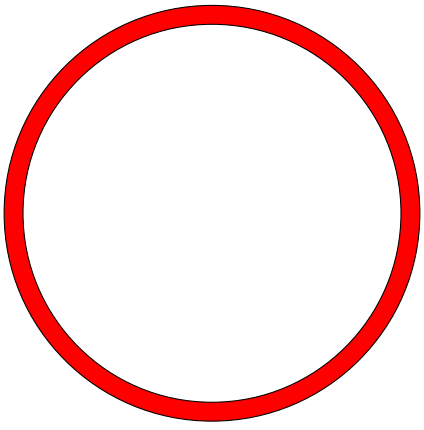
In Ti*k*Z, every new part of a drawing goes on top of everything else, so you need to pay attention to the order in which you draw things to ensure that they don't cover each other incorrectly. The following two examples illustrate this convention.

What would happen if we switched the order in which we drew the two circles? Here's the answer:



The red circles are drawn after the white circles and cover them up.

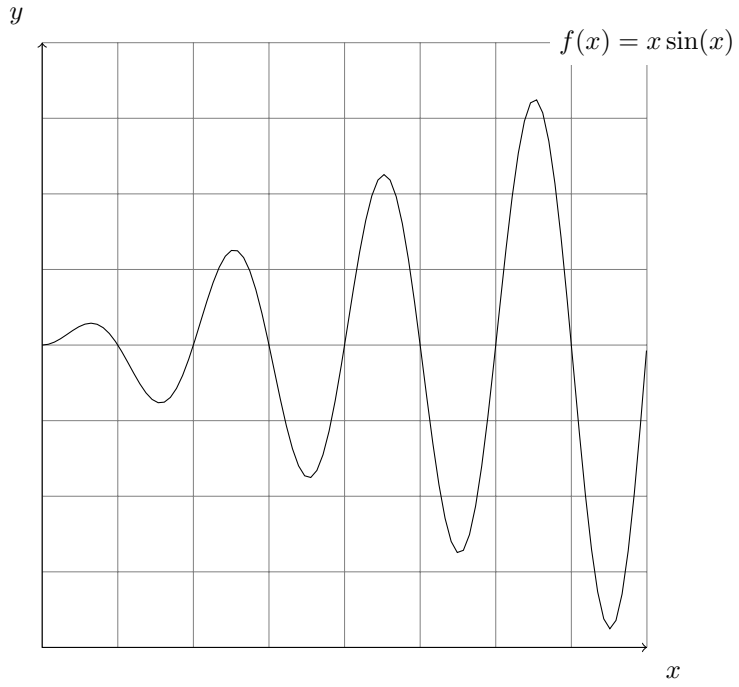
What would happen if we went back to the original order for drawing the circles, but instead of counting down, counted up? Here's the answer:



The last two circles cover all of the previous circles, so they can't be seen.

9.6 Worked Examples: Function Graphs

Here's what we want to produce.



We'll make this diagram with a few commands. First, use

```
\node[label=above left:$y$]  (topLeft)  at (0, 2) {};
\node[label=below right:$x$] (botRight) at (2, 0) {};
```

This defines two new coordinates, which can be referred to by `(topLeft)` and `(botRight)` respectively, and adds labels to the image with positions relative to the coordinates. The `{}` contains any text associated with the node. In this case, we don't want the nodes themselves to have text.

Generally, nodes are like coordinates in that they have a location and can be referred to later. Nodes can also have their own appearance and text content, although we're not using either of those features in this case.

To label a node, use the `label` option and provide a location relative to the node, followed by the label text. I find it confusing to talk about node labels, since nodes usually contain text themselves, and are often used as labels. Remember that labels are always associated with a node, while nodes can be inserted pretty much anywhere in a path.

Next, we'll draw the grid using the `grid` command, specifying that lines be drawn gray and very thin.

```
\draw[step=0.25,gray,very thin] (0,0) grid (2,2);
```

Draw the axes with a single path, starting from the center of the `topLeft` node, continuing to the origin, and finishing at the center of the `botRight` node.

```
\draw[<->] (topLeft.center) -- (0, 0) -- (botRight.center);
```

The following command actually draws the graph. For this to work, you'll need to add the line `\usetikzlibrary{calc}` to the preamble. It tells Ti*k*Z to load the `calc` library.[7]

```
\draw[domain=0:2,fill=none,samples=100,draw=black]
  plot (\x, {\x*sin(2*360*\x)/2 + 1});
```

This command uses the `plot` path instruction, which plots a function. We specify the function with `(\x, {\x*sin(2*360*\x)/2 + 1})`. The first part gives the independent variable, in our case `\x`, and the second part describes the function to be plotted. The function must be surrounded by curly braces. To see which functions are built-in to Ti*k*Z, consult the documentation. We use the `domain` option to specify the domain of coordinates over which we are plotting the function.

Ti*k*Z plots a function by taking samples of the function at various points, interpolating the missing values, and then trying to smooth out the curve. The more complex your function, the more samples you'll need to produce a graph that doesn't look jagged. In this case, we use the `samples` option to tell Ti*k*Z to use 100 samples.

Finally, we want to label the upper right corner to indicate the function that was plotted.

```
\node[fill=white] at (2, 2) {$f(x) = x \sin(x)$};
```

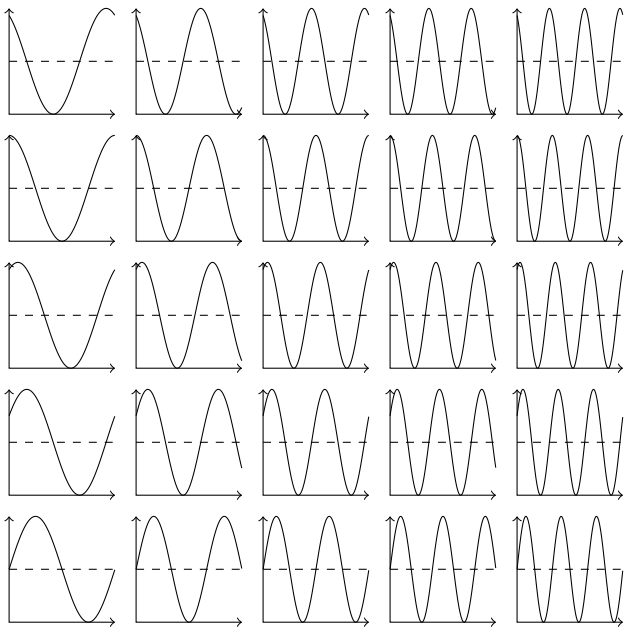
The complete code for the figure is as follows:

```
\begin{tikzpicture}[scale=4]
\node[label=above left:$y$] (topLeft) at (0, 2) {};
\node[label=below right:$x$] (botRight) at (2, 0) {};
\draw[step=0.25,gray,very thin] (0,0) grid (2,2);
\draw[<->] (topLeft.center) -- (0, 0) -- (botRight.center);

\draw[domain=0:2,fill=none,samples=100,draw=black]
  plot (\x, {\x*sin(2*360*\x)/2 + 1});
\node[fill=white] at (2, 2) {$f(x) = x \sin(x)$};
\end{tikzpicture}
```

As an exercise, make the following figure. The optional argument `shift={<x>,<y>}` to the `\draw` command may come in handy.

[7] A Ti*k*Z library is like a package for Ti*k*Z: it gives Ti*k*Z extra functionality.



10 Additional Resources

This set of notes is deliberately brief, which is good for getting across the important ideas but not good for communicating subtleties. If you would like to know more, the internet has extensive resources for learning and mastering LaTeX; the following are a few of our favorites.

- The Wikibooks LaTeX page (`en.wikibooks.org/wiki/LaTeX`). A compendium of information about LaTeX. If you want to know how to do something with LaTeX, this website can probably tell you how.
- Detexify (`detexify.kirelabs.org`). The LaTeX names for mathematical symbols can be hard to remember or look up because there are so many of them. Detexify lets you draw a symbol on the computer screen, then guesses which symbol you drew and tells you the LaTeX command for it.
- TeX Stack Exchange (`tex.stackexchange.com`). A question-and-answer site about TeX, LaTeX, and friends. The users of the site – among them people who designed LaTeX – are pros, and can answer about any question you have. Save it for the stumpers, though: Stack Exchange discourages questions whose answer can be easily looked up.
- The Comprehensive TeX Archive Network (CTAN) (`ctan.org`) Members of the LaTeX community have written over 4500 packages that add extra functionality to LaTeX. CTAN houses these packages and their documentation. When you have a question about a package, the documentation is often the best place to start.