Math II: Bold, Spacing, Matrices and Arrays

Introduction

Today, we will do some more work on math typesetting. In conjunction with the previous tutorial on math, by the end of the session, you will be in pretty good shape to write almost anything that your work will require.

- as before, open WinEdt and write the following:

```
\documentclass{article}
\usepackage{amsmath, amssymb}
\begin{document}

\end{document}
```

We will be needing the packages we loaded again today.

Bold Math

Sometimes you will want to write math symbols in bold. If so, you use the `\boldsymbol` command when in *math mode*. For example

```
$\boldsymbol{A\pi}-8$
```

gives us: $\boldsymbol{A\pi} - 8$. Note how only the A and the π are emboldened, so be careful about where you put the curly braces. Not also, that the emboldened A is not an emboldened text-mode A, which would be coded as this, `\textbf{A}` and written by LaTeX as this: **A**.

It is worth noting that there is another way to write your math output in bold, so long as don't want to write greek letters (and some other characters) in bold. If it is a simple expression like $\mathbf{x}$ or $y = \mathbf{7g} + 67$, you can use the `\mathbf` command. This works very similarly to the `\boldsymbol` command, so I produced the in-line math above by writing (in math mode) `\mathbf{x}` and `y=\mathbf{7g}+67`. As good practice, use `\boldsymbol`.

Spacing

LaTeX does spacing within formulas automatically and, like its other spacing, generally gets it right.

- sometimes though, it is useful to know how to insert some extra space. To get a 'thick space' (which is still pretty thin!) between two symbols, use `\;` You can use the space commands *multiple times*: this tends to be the only way you will notice a difference. So, `x+\sin\;\;\;\;\;\;\theta` gives you $x + \sin \quad \theta$.

Fraction-like environments

There are actually a few different ways to write fractions in LaTeX . The easiest is just to use `/` as in `$x/y$` which yields x/y. A much better way is to make use of the `\frac{}{}` command. Generally, the form of this command is:

```
\frac{numerator}{denominator}
```

As an example, here is `\frac{dy}{dx}`

$$\frac{dy}{dx}.$$

You can also use fractions in subscripts or superscripts, so `x^\frac{1}{4z}` gives you

$$x^{\frac{1}{4z}}.$$

- one of the packages we have loaded today gives you the ability to write binomial-like expressions too. So, `\binom{n}{i}` gives you

$$\binom{n}{i}$$

Equations and Equation Numbers

To produce a single displayed math formula with a numerical label, you need to use the `equation` environment. As an example,

```
\begin{equation}
(x+2)(x-1)=x^2+x-2
\end{equation}
```

gives you

$$(x+2)(x-1) = x^2 + x - 2 \tag{1}$$

There are few of things to notice here:

- first, the label (the number of the equation) is generated **automatically**. If you don't want a number, type `\begin{equation*}` and `\end{equation*}` and it won't label it. However, if you don't use a `*` next time you want an equation, it will number on from (1). You can have an unlabelled equation at any points in your article with respect to numbered ones. LaTeX has no problems renumbering equations if you want to chop and change them around in the article you are writing.

- second, I didn't have to use the dollar signs or the `\[ \]` to tell LaTeX I wanted to write math: just being in the equation environment is good enough
- third, the default is that the equation is centered. Some people—Kevin Clarke being their selected representative—don't like this effect, and put the command `[fleqn]` between `\documentclass` and `{article}` at the start of their LaTeX file to flush equations to the left of the page.
- you can refer to your equations using a command that doesn't involve writing `(1)` or `(5)` or whatever. Instead, you alter the code to write your equation by adding a `\label{}` command. For example:

  ```
  \begin{equation}
  x+y=98 \label{A}
  \end{equation}
  ```

  yields

  $$x + y = 98 \tag{2}$$

  I can refer to that equation by typing `(\ref{A})`, which gives (2). This is very helpful if you have lots of equations and think you may be shifting them around in the text (such that their numbers change).
- A useful extra tool here is to write a tilde before the parentheses and reference when using the word 'equation'. For example, writing

  ```
  equation~(\ref{A})
  ```

  with a tilde when you want to write 'equation (2)' means that LaTeX will never break a line between the word `equation` and the number you have given it. If you don't use this command, it can look rather strange.
- something else to note is that you should **not** leave a space (i.e. a blank line) between your equation and the `\begin{equation}` or `\end{equation}` commands. LaTeX will stall over it, thinking you have forgotten to write some math there (the same goes for the `eqnarray` environment we discuss below).

Equation Arrays

Sometimes you want equation arrays with rows of numbers and operators. An example is:

$$\begin{aligned} x &= v + 6b - f && (3)\\ &= (p+q)(p-q) && (4)\\ &= p^2 - q^2 && (5) \end{aligned}$$

which was produced using the `eqnarray` environment:

```
\begin{eqnarray}
x&=&v+6b-f\\
&=&(p+q)(p-q)\\
&=&p^2-q^2
\end{eqnarray}
```

Once again, note:

- the numbers appear on all lines automatically... and they follow on from equation (1), above.
- LaTeX knows we are in math mode
- you can stop it numbering in the same way as above: that is, using `\begin{eqnarray*}` and `\end{eqnarray*}`
- to start a new line of the array, use the `\\` command. Generally speaking, don't use it for the last line of the array.

It is important to note the use of `&` in the code above. These are called 'span marks' and must occur either side of the operator around which you want the equations to array. So, for example,

```
\begin{eqnarray}
x=v &+& 6b-f\\
&=& (p+q)(p-q)\\
=p^2 &-& q^2
\end{eqnarray}
```

gives you:

$$\begin{aligned} x = v &+ 6b - f && (6)\\ &= (p+q)(p-q) && (7)\\ = p^2 &- q^2 && (8) \end{aligned}$$

- sometimes, we don't want to label all the line of our equation array. This is easily done by adding the `\notag` command at the end of the input for that line, *before* the line end command, `\\`. For example,

  ```
  \begin{eqnarray}
  x&=&y + a\notag\\
  &=&92-b^2\\
  &=& (p+q)(p-q)\notag\\
  &=&z - 5
  \end{eqnarray}
  ```

  gives the following effect:

$$\begin{aligned} x &= y + a && \\ &= 92 - b^2 && (9)\\ &= (p+q)(p-q) && \\ &= z - 5 && (10) \end{aligned}$$

Matrices and Arrays

The LaTeX `array` environment is very similar to the `tabular` environment that is used in text mode. The main difference is that it writes in math mode, and that the box produced in the `array` environment has an axis, that tells LaTeX about the relative position of the entries. Generally, the format of the `array` environment is:

```
\begin{array}[position of axis]{justification of entries}
array entries
\end{array}
```

It is hard to think of really helpful uses of a arrays outside of definition by cases, which we mention below.

Matrices

Since we are using the `amsmath` package, there is a simple way to make matrices. The command you need is `\bmatrix` and you use it as follows:

```
\[\begin{bmatrix}
1&0&0\\
0&1&0\\
0&0&1\\
\end{bmatrix}\]
```

to produce:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Producing vectors is easy too, just type

```
\[\begin{bmatrix}
y_1\\
y_2\\
y_3\\
\vdots\\
y_n
\end{bmatrix}\]
```

to get:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

- note the use of the `\vdots` command to produce three vertical dots in that cell. You get horizontal dots by using `\hdots` and diagonal ones by using `\ddots`. So, we get

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

by using:

```
\[\begin{bmatrix}
a_{11}&a_{12}&\cdots &a_{1n} \\
a_{21}&a_{22}&\cdots &a_{2n} \\
\vdots & \vdots & \ddots & \vdots\\
a_{n1}&a_{n2}&\cdots &a_{nn}
\end{bmatrix}\]
```

There is one easier way to write a matrix (in the form of an array). Enter math mode by using `\[`, and then go up to `Insert` on the tool bar. Click it and then select `Matrix (n x m)`. LaTeX will offer you a box where you can give the dimensions. If you then press `OK`, it will immediately write down the matrix with a `*` to mark each entry **you** need to write. Then end by coming out of math mode, i.e. typing `\]`. You may need to change the brackets to square ones in the formula, depending on how you want your matrix to look.

Definition by Cases

The `array` environment can be used to typeset a definition by cases. From Wacklerly *et al.* (267):

$$f_y(y) = \begin{cases} 2y, & 0 \leq y \leq 1 \\ 0, & \text{elsewhere} \end{cases}$$

which was produced using

```
\[f_y(y)=\left\{\begin{array}{cl}
2y,&0\leq y \leq 1\\
0,& \mbox{elsewhere}\end{array}\right.\]
```

- the `{cl}` tells LaTeX to put the first column of stuff (i.e. the $2y$ and the 0) to the `c`enter, while the second column should be to the `l`eft
- notice the use of the span mark `&` to tell LaTeX there is a column divide.

- Notice the use of the `\mbox{ }` command here. This command allows you to write text in text mode whilst inside the math mode, such that it looks like normal text. If we didn't use the `\mbox` command, we would get a result like this:

$$f_y(y) = \begin{cases} 2y, & 0 \leq y \leq 1 \\ 0, & elsewhere \end{cases}$$