Computer Exercises 2010

Computer Aided Mathematics

Per-Anders Svensson

Linnæus University, School of Computer Science, Physics, and Mathematics

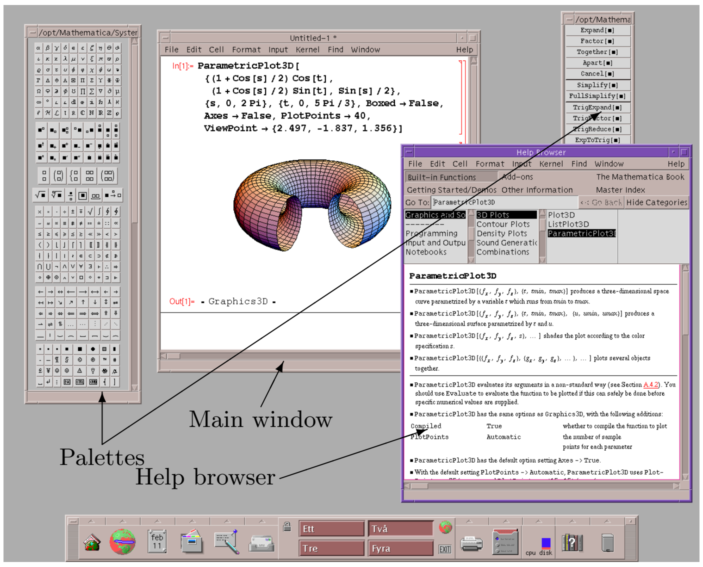
1 *Mathematica*

1.1 Introduction

Mathematica is a computer program that can be used for the most various kinds of mathematical calculations. This program is foremost adapted for symbolical manipulations, such as simplification of mathematical expressions, to explicitly compute the derivative of a function, and to find the exakt roots of a polynomial equation. Other computer programs, such as Matlab, use numerical methods and will thereby mostly return approximative values of a calculation. On the other hand, it is not possible to solve all mathematical problems exactly, and due to this *Mathematica* also can be used for numerical calculations.

Exercise 1: Start *Mathematica*. If you succeed, at least one window should be opened on the screen, namely the main window. In Figure 1 you can also see other windows or palettes such as a palette for algebraic manipulations (such as `Expand`, `Factor`, and `Together`), a palette for basic mathematical inputs (such as powers, square roots, and fractions), and a help browser. By default, only the main window opens when you start *Mathematica*.

The main window is just a part of *Mathematica*; a graphical interface towards the other part, the so-called *kernel*. We use this graphical interface to communicate with the kernel.



```
In[1]:= ParametricPlot3D[
          {(1 + Cos[s] / 2) Cos[t],
           (1 + Cos[s] / 2) Sin[t], Sin[s] / 2},
          {s, 0, 2 Pi}, {t, 0, 5 Pi / 3}, Boxed → False,
          Axes → False, PlotPoints → 40,
          ViewPoint → {2.497, -1.837, 1.356}]

Out[1]= - Graphics3D -
```

ParametricPlot3D

- `ParametricPlot3D[{`f_x, f_y, f_z`}, {`t, $tmin$, $tmax$`}]` produces a three-dimensional space curve parametrized by a variable t which runs from $tmin$ to $tmax$.
- `ParametricPlot3D[{`f_x, f_y, f_z`}, {`t, $tmin$, $tmax$`}, {`u, $umin$, $umax$`}]` produces a three-dimensional surface parametrized by t and u.
- `ParametricPlot3D[{`f_x, f_y, f_z, s`}, ... ]` shades the plot according to the color specification s.
- `ParametricPlot3D[{{`f_x, f_y, f_z`}, {`g_x, g_y, g_z`}, ... }, ... ]` plots several objects together.
- `ParametricPlot3D` evaluates its arguments in a non-standard way (see Section A.4.2). You should use `Evaluate` to evaluate the function to be plotted if this can safely be done before specific numerical values are supplied.
- `ParametricPlot3D` has the same options as `Graphics3D`, with the following additions:

`Compiled`	`True`	whether to compile the function to plot
`PlotPoints`	`Automatic`	the number of sample points for each parameter

- `ParametricPlot3D` has the default option setting `Axes -> True`.
- With the default setting `PlotPoints -> Automatic`, `ParametricPlot3D` uses `Plot-`

Figure 1: A screenshot of *Mathematica*.

Exercise 2: Make the main window active by clicking on it. Type `1 + 1` and press SHIFT+ENTER, i.e. first press SHIFT and then ENTER, keeping SHIFT pressed. (In this text a font with a fixed width, as in `Sin[x]`, is used to show commands that are supposed to be typed by hand into *Mathematica*.)

Note that when you start typing, a cursor shaped as a vertical bar shows up, and that the input is enclosed by a square bracket to the right of the main window. This is a so-called *cell*. If the cursor is within a certain cell when you press SHIFT+ENTER, the contents of the cell is sent to the kernel who will try to execute it. The text "`In[1] :=`", that is written on the screen after SHIFT+ENTER has been pressed, signifies that `1 + 1` has been sent to the kernel. The kernel executes the command, and types the resultat in a new cell, marked with `Out[1] =`. The digit `1` within the square brackets in "`In[1] :=`" and "`Out[1] =`" tells us that this the first calculation that the kernel has been executed and stored in its memory. Each time you quit *Mathematica*, we will also quit the kernel.

Exercise 3: You can always visit older input cells by placing the cursor inside it, do some modifications and/or corrections of the cell's input and then send its contents to the kernel once more. Try this by placing the cursor inside the input cell that contains `1 + 1`, change the input of that cell to `1 + 2`, and then press SHIFT+ENTER.

Exercise 4: The contents of the main window is possible to save to a file on e.g. the hard disk. To do this, choose File—Save in the pull-down menu, and save your computations to a file that you name `intro-mathematica.nb`. The extension `nb` is an abbreviation for "notebook".

Exercise 5: Move the cursor downwards in the main window, by pressing the *down-arrow* key (on the numerical keypad), until the cursor turns into a horisontal line (unless it is not already a horisontal line, of course). Note that when the cursor is a horisontal line, a new input cell will automatically be created, once you start to enter new text. Try to compute $2 \cdot 2$ in this new cell. The multiplication sign can be typed either as an asterisk (`*`) or by pressing the space bar.

Exercise 6: We have already got acquainted with how addition and multiplication are typed into *Mathematica*. When comes to the two remaining fundamental rules of arithmetics, subtraction and division, we use (of course) `-` and `/`, respectively. Try to compute 1/7, i.e. type `1/7` inside a new input cell and press SHIFT+ENTER. Note that the kernel returns 1/7. How come? We want *Mathematica* to compute 1/7 and it answers 1/7. Is there some kind of bug in the program? No, what *Mathematica* actually does, is to return the *exact* value of the calculation "1 divided by 7". *Mathematica* is constructed by mathematicians to be used by mathematicians, and therefore it tries to return exact values as long as possible. On the other hand, pocket calculators are notorious "miscalculators"; they return numerical approximations of computations, with sometimes can lead to misleading results, as we will see in an example, later on. If we however want *Mathematica* to behave like a pocket calculator and return a numerical approximation of 1/7, we could write `1./7.`, `1/7.`, or `1./7` and press SHIFT+ENTER. Here the point is to signify that the calculation should be performed numerically instead of exact.

Exercise 7: We use round parentheses (`()`) to control the order of calculations, in the exactly the same way as we are used to. Curly brackets (`{}`) and square brackets (`[]`) can however *not* be used instead of round parentheses in mathematical expressions, since they are interpreted in a certain way by *Mathematica*. Use *Mathematica* to compute the following expressions.

(a) $24 \cdot 73 + 91$	(b) $24(73 + 91)$	(c) $(45 - 2)/(23 - 67)$
(d) $12121212/343434$	(e) $(-4) \cdot (-5)$	(f) $2025/(54 - 9)$

Exercise 8: If we want *Mathematica* to compute powers a^b, we insert a circumflex (`^`) between the base and the exponent, as in `a^b`. So if we want to compute 2^3, for instance, we type `2^3` followed by SHIFT+ENTER. Compute
(a) $(-3)^8$ (b) -3^8 (c) 5^{-4} (d) 15^{2^7+33}

Exercise 9: There are several well-known mathematical constants predefined in *Mathematica* such as e.g. $\pi = 3.1415\ldots$, $e = 2.7182\ldots$, and ∞ (the infinity). Type `Pi` and press SHIFT+ENTER. Where you surprised by the answer? If you were, take one more look at Exercise 6!

To obtain an approximation of π with 200 significant digits, type `N[Pi, 200]` and press SHIFT+ENTER. The constants e and ∞ are called `E` and `Infinity`, respectively, in *Mathematica*.

Exercise 10: (This exercise can be skipped, if you are not familiar to complex numbers.) The imaginary unit, usually denoted i, is written `I` in *Mathematica*. Compute
(a) i^2 (b) $(2+i)(5-3i)$ (c) $(7-i)/(3-2i)$

1.2 Variables and predefined functions

Exercise 11: When you deal with extensive computations, it is often convenient to compute and save partial results, which then can be referred to later on, when they are needed. To achieve this in *Mathematica*, we may use variables to store the value of a computation. Try by typing

```
a = 3^8
```

and then press SHIFT+ENTER. Here `a` is the name of our variable, and we assign the result of the computation 3^8, i.e. 6561, to this variable. The equality sign is interpreted as an assignment, and the expression to the left of it, i.e. `a`, is assigned the value of the expression to the right of it, i.e. 3^8. To refer to the saved value of this computation, we can simply type `a`. Try to compute

$$\frac{a+5}{a^2-1}.$$

Note that the above computation does not change the value assigned to the variable `a`. Confirm this by typing `a` inside a new input cell and press SHIFT+ENTER.

A variable name may contain as many letters as you wish, as well as digits. The name cannot however start with a digit, and should moreover preferably not start with a capital letter, since predefined commands in *Mathematica* always do this.

Exercise 12: In the previous exercise we assigned a value to the variable `a`, to be used in later computations. Sometimes the result of a computation could be quite long. For instance, it could be a number with many digits. Since it perhaps is nothing more but a partial result, we might not be interested in the details anyway. Therefore there is a possibility to suppress the result of a calculation from being printed on the screen. Type

```
b = 2^a;
```

Do not forget the semicolon! Note that when you press SHIFT+ENTER, nothing is printed on the screen. But the kernel has performed the calculation anyway, and assigned its value to the variable `b`. Thanks to the semicolon, the result of computation is not shown on the screen. To see the value assigned to `b`, type `b` in a new input cell and press SHIFT+ENTER. A pretty large number, eh?

Exercise 13: Several mathematical functions are defined within *Mathematica*, such as e.g. $\sqrt{x}$ (*the square root of* x). Each such function has a unique name that starts with a capital letter. To compute $\sqrt{122}$, we send

```
Sqrt[122]
```

to the kernel. This time you should not be surprised by the answer! To obtain a numerical approximation of $\sqrt{122}$, type

```
Sqrt[122.]
```

Here the square brackets are used to signify that the expression surrounded by them is the input to the function `Sqrt`.

Exercise 14: As you probably already know, $\sin(30°) = 1/2$. Let us see if *Mathematica* knows this equality. Compute

```
Sin[30]
```

But *Mathematica* returns $\sin(30)$. Should it not reply with $1/2$, which is after all an exact value? Is this a bug? So it seems, but the thing is that degrees are rarely used (by mathematicians) to measure angles. Instead, radians are far more used. Try to compute

```
Cos[Pi]
```

If we still want to use degrees to measure angles, we must explain this to *Mathematica*. To do this in the above example, when it comes to computing $\sin(30°)$, we type

```
Sin[30 Degree]
```

Compute $\cos(30°)$ in the same way.

1.3 Algebra

Exercise 15: The command `Solve` can be used to solve equations. Suppose we want to solve the equation $2x - 3 = 5$. I *Mathematica* we then type

```
Solve[2 x - 3 == 5, x]
```

Note that we write `==` instead of `=` (a single equality sign is used for assignments, which is not the case here; we do not want to assign the value `5` to the expression `2 x - 3`). The command `Solve` takes two arguments, separated by a comma sign. The first argument is the equation we want to solve (in this case `2 x - 3 == 5`). The second argumentet tells *Mathematica* that `x` is the unknown. Use *Mathematica* to solve the equation $3x^2 - 14x + 5 = 0$.

Exercise 16: *Mathematica* contains several different commands for performing algebraical manipulations of mathematical expressions. Most of those commands are self-explained; in the table below, u denotes an expression.

`Simplify[`u`]`	– simplifies u	`Expand[`u`]`	– expands u
`Factor[`u`]`	– factors u	`Together[`u`]`	– writes u as a fraction

Suppose we want to simplify the expression $x(x+2)-(x+1)(x+3)$. This is performed in *Mathematica* by the command

```
Simplify[x (x + 2) - (x + 1)(x + 3)]
```

Use the commands above to

(a) Expand $(x-3)(x-4)$ and $(x-4)^3(7-x)^3$

(b) Factor $x^5+4x^4+4x^3$

(c) Write

$$\frac{2x-1}{x+3}-\frac{4-x}{x-2} \quad\text{and}\quad \frac{x+\dfrac{1}{x}}{1-\dfrac{1}{x^2+1}}-\frac{3-x}{2+\dfrac{x-2}{x+1}}$$

as fractions.

Exercise 17: Use *Mathematica* to simplify the expression

$$\sqrt{2+\sqrt{3}}\sqrt{2+\sqrt{2+\sqrt{3}}}\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{3}}}}\sqrt{2-\sqrt{2+\sqrt{2+\sqrt{3}}}}.$$

Hint: An expression like $\sqrt{2+\sqrt{3}}$ is typed `Sqrt[2 + Sqrt[3]]` in *Mathematica*. You could also use the palette BasicMathInput in the meny Palettes (see also Figure 1);

Exercise 18: In this exercise we will try to illustrate what troubles rounding errors could cause, if we are not careful. As we saw in Exercise 13, a numerical approximation of $\sqrt{122}$ is 11.0454. Do the following assignments:

```
a = 11.05;
b = 11.0454;
```

Then `a` as well as `b` contains a value, almost equal to the exact value of $\sqrt{122}$. Now consider the expression

$$\frac{1}{(\sqrt{122}-11)^2}. \tag{1}$$

Find a numerical approximation of (1) by typing the following code into *Mathematica*:

```
N[1/(Sqrt[122] - 11)^2]
```

Then compute (1) again, but replace $\sqrt{122}$ first by `a`, then by `b`. What do you say about the accuracy of the results?

Let us simplify the expression (1) by rewriting it as follows:

$$\frac{1}{(\sqrt{122}-11)^2}=\frac{(\sqrt{122}+11)^2}{(\sqrt{122}-11)^2(\sqrt{122}+11)^2}=243+22\sqrt{122}.$$

Use *Mathematica* to confirm that the last equality above is valid, by executing

```
Simplify[
  Expand[
    (Sqrt[122] + 11)^2/((Sqrt[122] - 11)^2 (Sqrt[122] + 11)^2)
  ]
]
```

Now try to compute (1) by replacing $\sqrt{122}$ by `a` and `b`, respectively, in the expression $243 + 22\sqrt{122}$. Was the accuracy better or worse this time?

We could of course also expand the denominator of (1):

```
Expand[(Sqrt[122] - 11)^2]
```

which returns $243 - 22\sqrt{122}$. Compute (1) once more, replacing $\sqrt{122}$ by `a` and `b`, respectively, in the expression $1/(243 - 22\sqrt{122})$. How accurate was the results this time, and how to you interpret the examples in this exercise?

1.4 Graphics

Exercise 19: You can reach the built-in help browser of *Mathematica* in several ways. Suppose for instance that we want to know more about the predefined command `Plot`. To obtain a short description, we may type `?Plot` in a cell of its own, followed by SHIFT+ENTER. For a more complete description of the command, along with examples of how to use it, type `Plot` in a new cell, and then press `F1` on the keyboard.

Exercise 20: To plot the graph of a function (in one variable) use can use the command `Plot`. Suppose we wish to plot the straight line $y = 2x + 1$ for $-5 \leq x \leq 5$. In *Mathematica* we then type

```
Plot[2 x + 1, {x, -5, 5}]
```

Plot the graph of $y = x^3 - 4x^2 + 4x$ for $-1 \leq x \leq 4$.

You have now had a short introduction to what you can to, using *Mathematica*. As you may expect, we have however only barely scratched the surface...

2 Typesetting

LaTeX is a typesetting program made for producing professional documents with a relatively small effort. Compared to the most common word processors, such as Microsoft Word, you "program" how the document should be look like, by certain control words. The idea behind this is that you in the first place should fokus on the contents of what you are writing; the layout comes second. Among mathematicians, computer scientists, scientists, and one and another economist, the computer program LaTeX is the most common tool for producing texts, such as research articles, theses, essays, and books.

In this assignment you will, step by step, construct a document, using LaTeX. The code that you are supposed to write, will always be set apart from a paragraph, and displayed using `this font`. To make it easier for you to find where each new piece of code is to be inserted, previously written code will be typed in a grayish shade (and thus will not have to be copied again). Be careful when you copy the code, especially with certain characters such as *backslash* (`\`) and *curly brackets* (`{}`), since these cannot be replaced by other characters.

2.1 Installation

All software that are needed are freeware and can be downloaded from the Internet. On the computers of the department (School of Computer Science, Physics, and Mathematics), LaTeX should already be installed, so if you are using one of these computers, you may skip this entire section.

It is not possible to describe in detail how you do to install LaTeX on your own computer, since the this procedure depends too much on the operating system you are using, and of which distribution of LaTeX you wish to install. Read the installation guide for the distribution of your choice. The following distributions of LaTeX are recommended:

- MiKTeX – Windows `http://www.miktex.org/`
- TeX Live – Linux `http://www.tug.org/texlive/`
- MacTeX – Mac OSX `http://tug.org/mactex/`

You also need a text editor, in which you type your LaTeX code. This program should by default save a text as an ordinary text file. Thus a word processor is not a good choice. Some advisable programs are

- TeXnicCenter – Windows `http://www.texniccenter.org/`
- GNU Emacs `http://www.gnu.org/software/emacs/`
- Kile – Linux `http://kile.sourceforge.net/`
- TeXShop – Mac OSX `http://www.uoregon.edu/~koch/texshop/`

To be able to read PDF files and work with PostScript files you also need

- Adobe Reader `http://adobe.se/products/acrobat/readstep2.html`
- Ghostscript `http://www.cs.wisc.edu/~ghost/index.html`

On the site where Ghostscript can be downloaded, you can also find a program that can read PostScript files, namely Ghostview (the Windows version of this program is called GSview). Note that all the above software are free. Hence you can legally download and use them, with some reservations. (For more information, see the license for each software).

2.2 Get started with LaTeX

During an ordinary LaTeX session, you usually shift between several programs, which is one of the reasons that LaTeX feels like if it is harder to learn, than it actually is. Below we will present an easy guide, in order to show how a simple document is constructed, and how the organization of the work during a LaTeX session could look like.

Exercise 1: Begin by starting a text editor, such as TeXnicCenter. Open a new document in this editor and copy the following code:

```
\documentclass[a4paper]{article}
\begin{document}
Hello world!
\end{document}
```

In TeXnicCenter it may look like something similar to Figure 2. Be sure to copy the code exactly as it is shown above.

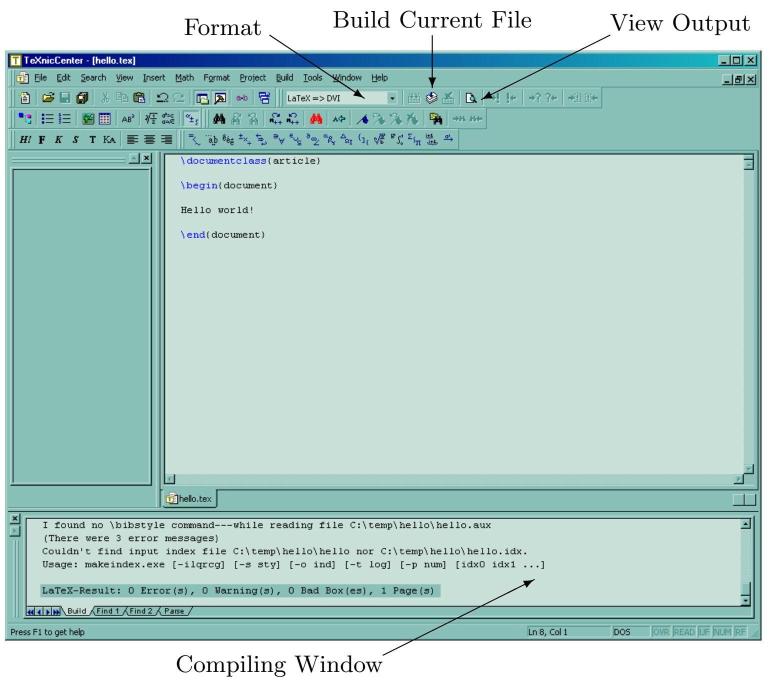
Create a new folder in your home directory and save the code above in a file called `hello.tex` in this new folder. The file extension `tex` is recommended. Since a document created by LaTeX will consist of several different files, it is suitable to sort one's files in such a way, that each document has its own folder. For the time of being the folder we just created only contains one file, namely `hello.tex`.

The next step is to create a file that will show how the document actually will look like. To do this, we need to *compile* the text file, using LaTeX. In TeXnicCenter this is done by clicking on the button "Build Current File" (shortcut: Ctrl+F7). During the compilation, several messages will be printed after one another in the compiling window. If you have typed the code above correctly, the last line in the compiling window of TeXnicCenter should be read:

```
LaTeX-result:  0 Error(s), 0 Warning(s), 0 Bad Box(es), 1 Page(s)
```

If one or more errors occurred, check if you copied the code correctly, correct the errors (if any), save the file, and compile it once again.

Exercise 2: Assuming that all went well in the previous exercise, it is now time for us to take a look at the result of the compilation. Before we do this, we observe that, in TeXnicCenter, `LaTeX => DVI` is shown in the pull-down meny "Format" (see Figure 2). This means that whenever the file `hello.tex` is compiled, LaTeX will generate the output in file by the name `hello.dvi`. In fact two more files will also be created: `hello.aux` and `hello.log`. The first one of these two files will in general contain information that LaTeX need for showing cross references in a correct way. The other file will contain all messages that was printed in the compiling window during the compilation. Open the folder in which you saved `hello.tex` and verify that all those files are there.


Format
Build Current File
View Output
TeXnicCenter - [hello.tex]
File Edit Search View Insert Math Format Project Build Tools Window Help
LaTeX => DVI
hello.tex
Build Find 1 Find 2 Parse
Press F1 to get help
Ln 8, Col 1
DOS
Compiling Window

```
\documentclass{article}

\begin{document}

Hello world!

\end{document}
```

```
I found no \bibstyle command---while reading file C:\temp\hello\hello.aux
(There were 3 error messages)
Couldn't find input index file C:\temp\hello\hello nor C:\temp\hello\hello.idx.
Usage: makeindex.exe [-ilqrcg] [-s sty] [-o ind] [-t log] [-p num] [idx0 idx1 ...]

LaTeX-Result: 0 Error(s), 0 Warning(s), 0 Bad Box(es), 1 Page(s)
```

Figure 2: TEXnicCenter.

Exercise 3: As we already have mentioned, we are interested of the how the document looks like after the compilation. Go back to TEXnicCenter and click on the button "View Output" (shortcut: F5). Then the file `hello.dvi` will be opened by another program, the DVI-viewer YAP. Note that besides the text "Hello world!" a page number was automatically added at the bottom of the page.

The document on the screen can be zoomed in or out by using the plus and minus keys on the numerical keypad. You can also zoom in certain parts of the document on the screen by using the mouse as a magnifying glass. To do this the magnifying glass button (the rightmost button on the icon bar at the top of the YAP window) must be pressed.

Exercise 4: In Windows you can now print your document from the DVI-viewer. However, if you wish to send your document to someone who does not use LATEX, he or she will probably not be able to read it, since the file format DVI is uncommon outside the world of LATEX. This can be solved (in TEXnicCenter) by choosing `LaTeX => PDF` in the pull-down meny "Format" (see Figure 2). If you then compile the document, the result will be a PDF-file instead of a DVI-file, and such files are far more widely accepted. Try to generate a PDF-file from your document and then open it.

3 Basic typesetting using LATEX

In the remaining part of this assignment you will put together a somewhat larger document. By following the instructions in the exercises that follow, you will construct this document in small steps. The intention is that you copy exactly each portion of code that is described in each exercise, save the file, compile it, and look at the result of the compilation. Doing so, it will be more easy to see what effect each one of the small code portions has, than it would be if you write too much code before looking at the compiled document.

Exercise 5: Create a new file in TEXnicCenter using File—New... in the pull-down menu, and save it as `intro.tex`.

Exercise 6: Type in the following code. Check carefully that you use the correct kind of parentheses and that you only use small letters in the names of the commands.

```
\documentclass[11pt,a4paper]{article}

\usepackage{amsmath,amssymb}

\begin{document}

\end{document}
```

As you can see, we have inserted empty lines in the code. These are not necessary, but it will make the code easier to read.

The behavior LATEX is controlled by commands. A command always starts with a backslash (\), followed by one or more letters. One such command in the text above is `\usepackage`. The way in which this command is used here, is to load two extra packages amsmath and amssymb into LATEX. If your document will contain mathematics (which will be the case here), these packages will be useful, since they contain useful commands for typing mathematical formulae and equations.

You do not need to compile your file until the next exercise, since compiling it in this very moment would not generate any document – we have not yet typed any of the text that will be displayed in the document (which is done between `\begin{document}` and `\end{document}`).

Exercise 7: We start by inserting the title and author of the document.

```
\begin{document}

\title{A Laboratory Lesson in Mathematical Typesetting}
\author{}
\maketitle
\end{document}
```

Write your own name in between the curly brackets that follow `\title`. Compile your file and watch the resultat. Note that the title, your name and today's date are formatted.

Exercise 8: Some editorial structures (such as e.g. tables and figures) are so extensive that they in LATEX are inserted by something called *environments*. An environment starts with `\begin{`*env*`}` and ends with `\end{`*env*`}`, where *env* is the name of the environment. You have already seen an example of an environment, namely `document`. Another example of an environment is `abstract`, which can be used for inserting abstracts:

```
\maketitle

\begin{abstract}
  An abstract should give a short and interesting description
  of the contents of the document.
\end{abstract}
\end{document}
```

Above we have made an indentation of two spaces on the rows inside the environment. This is only for making the code more easy to read, and will not affect the looks of the compiled document.

Exercise 9: It is easy to insert headings and running text.

```
\end{abstract}

\section{Introduction}

When you use \LaTeX there is no risk that a heading ends up at
the bottom of a page and the text on the following one. Also
observe that the first paragraph of a chapter or a section is
never indented. If you type two or more spaces after one
another like this:                      it will only be
regarded as one.
\end{document}
```

When you examine the resultat of a compilation, you will perhaps notice that a space is missing after `\LaTeX`. This can be solved inserting a backslash in the code like this:

```
When you use \LaTeX\ there is no risk that a heading ends up at
```

We continue by inserting more headings and text. Type in some rows at a time of the code below, save, compile, and compare the result with the corresponding code.

```
regarded as one.

A new paragraph is indicated by an empty row in the code and by
a new row with indentation in the typesetted document. Two or
more empty rows after one another will be regarded as one. As
you can see, the first row of this paragraph is indented.
The
final
result
does
not
depend
on
where
you
insert
new
lines
```

```
in
the
code.
% This sentence will not be included in the document.
```

Word-wrapping in the code is interpreted by LaTeX as spaces. All text to the right of a percent sign on a row, will be ignored by LaTeX. If you wish to write percent signs in your text, type `\%`.

```
% This sentence will not be included in the document.

\subsection{An intermediate heading}

A new section. Notice that \LaTeX\ controls how headings are
enumerated. Usually all enumerated headings will show up
automatically in the table of contents.

\subsubsection{A subheading}

The body text is typesetted with a straight right margin, and
since the hyphenation works well in \LaTeX, there will be no
large gaps between the words on a row.

\subsubsection{Yet another subheading}

Footnotes should only be used in exceptional cases, even if
they are easy to include.\footnote{This is a footnote.}

\subsection{Next intermediate heading}

Here we have moved up one level in the heading hierarchy.
Notice that the heading is enumerated correctly by \LaTeX.

\subsubsection{The third subheading}

If you want to emphasize something, like a word or a phrase,
you could write it in \emph{italics} or \textbf{boldface}.
Other available fonts are \textsl{slanted}, \textsc{small
caps}, \textsf{sans serif}, and \texttt{typewriter}. It is
also possible to modify the size of the text:
\begin{center}
  {\Large This} sentence {\LARGE looks}
  {\tiny very} {\large strange}{\Huge !}
\end{center}
\end{document}
```

The code above should be self-explained to the most, but do not hesitate to ask the instructor if there are some parts of it that you do not understand.

Exercise 10: You can add a table of contents to your document, by using the command `\tableofcontents`. Insert this command in your code, right after the abstract (that was added to the document in Exercise 8).

```
\end{abstract}

\tableofcontents

\section{Introduktion}
```

Compile and study the result. Note that the table of contents is empty; only the heading ”Contents” is displayed. For all headings to be included in the table of contents, you have to compile the document twice. This also holds for new headings; for them to show up in the table of contents, you have to compile twice.

Exercise 11: Go to the end of your file and add the following text:

```
\end{center}

\section*{Headings without numbers}

If you wish to supress the enumeration of a section, you
add an asterisk directly after the section command. This
also works for intermidiate headings and subheadings.
Headings without a number will not show up in the table of
contents.

\section{More examples}

We continue this laboration by looking at more editorial
structures and how one includes them in a \LaTeX\ document.
\end{document}
```

Compile your file twice and study the result, especially how the table of contents is updated.

Exercise 12: Let us insert a bulleted list:

```
structures and how one includes them in a \LaTeX\ document.

\subsection{Lists}

A simple bulleted list:
\begin{itemize}
\item First item
\item Second item
\item Third item
\end{itemize}
\end{document}
```

By default, a dot is used as an ornament for each entry in the list. If we replace `itemize` by `enumerate` we obtain an enumerated list. It is also possible to insert lists within lists, as shown in the next example.

```
\end{itemize}
An enumerated list with three sublists:
\begin{enumerate}
```

```
\item First item
\item The second item contains a sublist
  \begin{enumerate}
  \item The first item of the sublist contains in turn yet
    another sublist
    \begin{enumerate}
    \item First item of the subsublist
    \item Second item of the subsublist
    \end{enumerate}
  \item Second item of the sublist
  \end{enumerate}
\item The third item consists of a longer text that will not
  end until later. Namely after the next sentence. This is
  the last sentence of this item.
\end{enumerate}
It is possible to nest lists in at most four levels.
\end{document}
```

Notice how the way in which the items are enumerated changes between a lists and its sublists. Something similar happens if one nests several `itemize` environments. Experiment by adding a dotted sublist to the dotted list above.

Exercise 13: Now we come to the main strength of LaTeX, namely mathematical typesetting.

```
It is possible to nest lists in at most four levels.

\subsection{Formulae}

A mathematical equation within a paragraph must be put
between two dollar signs. An interesting equality is
$e^{i\pi} + 1 = 0$. Any formula is read from the left to the
right. The previous formula reads ``e raised to i pi, plus
one, equals zero''.
\end{document}
```

The special character circumflex (`^`) raises the character to the right of it, compared to the character to its left. If we wish to raise more than one character, they must be put within two curly brackets. You should **not** use `"` for quotation marks. Instead, use two grave accents (` `` `) for opening quotation marks and two vertical quotes (`''`) for closing quotation marks.

Also include the following code:

```
one, equals zero''.
A few examples more: $f_n = f_{n - 1} + f_{n - 2}$ and
$a^{(p - 1)/2} \equiv 1 \pmod{p}$.
\end{document}
```

Underscore (`_`) works in a similar way as circumflex, but instead of raising a character (or a group of characters within curly brackets), it lowers it.

Exercise 14: In the previous exercise we learned how to include equations within a paragraph. If a equation is large, it is convenient to set it apart from the rest of the paragraph; we then say that we *display* the equation. There are several environments for displaying equations, and we will here study the most common ones. We will show how to insert the equation

$$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}.$$

The three dots are a so-called ellipsis and is obtained by the command `\dots`. To obtain a fraction we use the command `\frac{`*numerator*`}{`*denominator*`}`. Copy the following code.

```
$a^{(p - 1)/2} \equiv 1 \pmod{p}$.

Equations, that are important or take too much space within
a paragraph, should be displayed. Often you wish to refer to
a equation later (or earlier) in the text. We can tell
\LaTeX\ to number such an equation in the right margin:
\begin{equation}
  \label{eq:summa}
  1 + 2 + 3 + \dots + n = \frac{n(n + 1)}{2}.
\end{equation}
The equation~\eqref{eq:summa} is well-known to mathematicians
and is applied every now and then.
\end{document}
```

Compile your file at least twice and check the result. In the compiled document you will see that the equation has been numbered by an (1) to the right of it, and that `\eqref{eq:summa}` in the code has been replaced by (1). This is an example of a so-called *cross reference*, which LaTeX deals with automatically. We use `\label` to put a label on the numbered equation, and by `\eqref` we will refer to the equation having this very label. The swung dash (`~`) is a special character which prohibits a line break to occur on that position in the compiled document.

The advantage of using automatical cross references is realized when one inserts a new displayed equation before the previous one. We will illustrate this by inserting the formula

$$\Lambda(n)\log n + \sum_{d \mid n} \Lambda(d)\Lambda\left(\frac{n}{d}\right) = \sum_{d \mid n} \mu(d)\log^2 \frac{n}{d}.$$

To make sure that parentheses have the correct size, depending of the size of what they enclose, you use `\left` and `\right`, where each command is followed by an (opening or closing) parenthesis. When you copy the code below, notice how the text before and after the equation is directly connected to the `equation` environment.

```
$a^{(p - 1)/2} \equiv 1 \pmod{p}$.

The equality
\begin{equation}
  \label{eq:Selberg}
  \Lambda(n) \log n
  +
  \sum_{d \mid n} \Lambda(d) \Lambda\left(\frac{n}{d}\right)
  =
  \sum_{d \mid n} \mu(d) \log^2 \frac{n}{d}
\end{equation}
occurs in analytical number theory and is called Selberg's
identity. You can use~\eqref{eq:Selberg} to prove the prime
number theorem.

Equations, that are important or take too much space within
```

Compile your file twice and look at the result. The displayed equation that we included at first, now has the number (2) and the cross reference to this equation has been updated accordingly. It is easy to see that there is a risk of forgetting to update some cross references, if you were to change them all by your own.

Exercise 15: Add the following code at the end of your file.

```
and is applied every now and then.
Here comes yet another displayed equation, but this one has
no number assigned to it:
\[
  \frac{\sin m x}{\sin x}
  = (-4)^{(m - 1)/2} \prod_{j = 1}^{(m - 1)/2}
  \left(\sin^2 x - \sin^2 \frac{2 \pi j}{m}\right).
\]
\end{document}
```

Sometimes you need to display equations that are too large to fit on a single row. You then need to insert line breaks within the equation. There are several different environments for doing this.

```
\]
Sometimes one single row is not enough for typesetting
displayed equations. The first example below shows how to
make each row centered on its own:
\begin{gather}
  x_{n + 1} \equiv x_n \pmod{p^n}, \\
  \prod_{p \leq \infty} |x|_p = 1,
\end{gather}
\end{document}
```

The double backslash (\\) signifies a line break. Here comes another example:

```
\end{gather}
and the second example shows how to align the rows
vertically to a fixed point (right before the equality sign
in this case):
\begin{align*}
  \sum_{k = 1}^{n + 1} f_k & = \sum_{k = 1}^n f_k + f_{n + 1}
  = f_{n + 2} - 1 + f_{n + 1} \\
  & = f_{n + 2} + f_{n + 1} - 2 = f_{n + 3} - 1.
\end{align*}
Note that if a sentence ends with an equation, a point
should be added at the end.
```

```
\end{document}
```

The ampersand (`&`) is used as an adjustment point. If you omit the asterisk (`*`) after `align`, both rows of the equation will be numbered.

Exercise 16: In this exercise we will typeset the following table:

		World record	
Name	**Country**	**Event**	**Result**
Therese Alshammar	Sweden	50 m butterfly	25.07
Wilson Kipketer	Denmark	800 m	1:41.11
Jan Železný	Czech Republic	Javelin	98.48
Sergey Bubka	Ukraine	Pole vault	6.14

We save the first row (containing the word "World record") for later, and take a look at the five other rows. Looking closely, we see that all the five rows have four columns, where the first three are left-aligned, and the fourth is right-aligned. This can be abbreviated `lllr`, where `l` means "left" and `r` means "right". The environment `tabular` is used to construct tables. To begin with, we only show how the code will look for the four bottom rows of the table:

```
should be added at the end.

\subsection{Tables}

The following table was valid as of August 6th, 2010.
\begin{center}
  \begin{tabular}{lllr}
    \hline
    \textbf{Name}  & \textbf{Country} &
    \textbf{Event} & \textbf{Result}  \\
    \hline
    Therese Alshammar & Sweden         & 50 m butterfly
      &   25.07 \\
    Wilson Kipketer   & Denmark        & 800 m
      & 1:41.11 \\
    Jan \u{Z}elezn\'y & Czech Republic & Javelin
      &   98.48 \\
    Sergey Bubka      & Ukraine        & Pole vault
      &    6.14 \\
    \hline
  \end{tabular}
\end{center}

\end{document}
```

The command `\hline` inserts a horisontal line all across the table. The ampersand (`&`) shifts to the next column and double backslash (`\\`) starts a new row of the table. The spaces that we have included in the code are not necessary for the final result, but they make the code easier to read.

What is left for us to do, is to include the first row of the table. We see that the headline "World record" is supposed to cover the third and fourth column, and in the same time be centered over them. Moreover a horisontal line covering the third and fourth column is to be inserted. Here comes the code that fulfills these requirements:

```
\hline
& & \multicolumn{2}{c}{\textbf{World record}} \\
\cline{3-4}
\textbf{Name}  & \textbf{Country} &
```

Thereby we are done with the table and also with the laboration. On the following pages is it shown how the result of your document is supposed to look like, if you have copied the code literally.

A Laboratory Lesson in Mathematical Typesetting

Per-Anders Svensson

August 6, 2010

Abstract

An abstract should give a short and interesting description of the contents of the document.

Contents

1 Introduction

When you use LaTeX there is no risk that a heading ends up at the bottom of a page and the text on the following one. Also observe that the first paragraph of a chapter or a section is never indented. If you type two or more spaces after one another like this: it will only be regarded as one.

A new paragraph is indicated by an empty row in the code and by a new row with indentation in the typesetted document. Two or more empty rows after one another will be regarded as one. As you can see, the first row of this paragraph is indented. The final result does not depend on where you insert new lines in the code.

1.1 An intermediate heading

A new section. Notice that LaTeX controls how headings are enumerated. Usually all enumerated headings will show up automatically in the table of contents.

1.1.1 A subheading

The body text is typesetted with a straight right margin, and since the hyphenation works well in LaTeX, there will be no large gaps between the words on a row.

1.1.2 Yet another subheading

Footnotes should only be used in exceptional cases, even if they are easy to include.[1]

1.2 Next intermediate heading

Here we have moved up one level in the heading hierarchy. Notice that the heading is enumerated correctly by LaTeX.

1.2.1 The third subheading

If you want to emphasize something, like a word or a phrase, you could write it in *italics* or **boldface**. Other available fonts are *slanted*, SMALL CAPS, sans serif, and `typewriter`. It is also possible to modify the size of the text:

This sentence looks very strange!

Headings without numbers

If you wish to supress the enumeration of a section, you add an asterisk directly after the section command. This also works for intermidiate headings and subheadings. Headings without a number will not show up in the table of contents.

2 More examples

We continue this laboration by looking at more editorial structures and how one includes them in a LaTeX document.

[1]This is a footnote.

2.1 Lists

A simple bulleted list:

- First item
- Second item
- Third item

An enumerated list with three sublists:

1. First item
2. The second item contains a sublist
   (a) The first item of the sublist contains in turn yet another sublist
      i. First item of the subsublist
      ii. Second item of the subsublist
   (b) Second item of the sublist
3. The third item consists of a longer text that will not end until later. Namely after the next sentence. This is the last sentence of this item.

It is possible to nest lists in at most four levels.

2.2 Formulae

A mathematical equation within a paragraph must be put between two dollar signs. An interesting equality is $e^{i\pi} + 1 = 0$. Any formula is read from the left to the right. The previous formula reads "e raised to i pi, plus one, equals zero".

A few examples more: $f_n = f_{n-1} + f_{n-2}$ and $a^{(p-1)/2} \equiv 1 \pmod{p}$.

The equality

$$\Lambda(n)\log n + \sum_{d|n} \Lambda(d)\Lambda\left(\frac{n}{d}\right) = \sum_{d|n} \mu(d)\log^2 \frac{n}{d} \tag{1}$$

occurs in analytical number theory and is called Selberg's identity. You can use (1) to prove the prime number theorem.

Equations, that are important or take too much space within a paragraph, should be displayed. Often you wish to refer to a equation later (or earlier) in the text. We can tell LaTeX to number such an equation in the right margin:

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}. \tag{2}$$

The equation (2) is well-known to mathematicians and is applied every now and then. Here comes yet another displayed equation, but this one has no number assigned to it:

$$\frac{\sin mx}{\sin x} = (-4)^{(m-1)/2} \prod_{j=1}^{(m-1)/2} \left(\sin^2 x - \sin^2 \frac{2\pi j}{m} \right).$$

Sometimes one single row is not enough for typesetting displayed equations. The first example below shows how to make each row of the equation centered on its own:

$$\begin{gather} x_{n+1} \equiv x_n \pmod{p^n}, \tag{3} \\ \prod_{p \leq \infty} |x|_p = 1, \tag{4} \end{gather}$$

and the second example shows how to align the rows vertically to a fixed point (right before the equality sign in this case):

$$\begin{aligned} \sum_{k=1}^{n+1} f_k &= \sum_{k=1}^{n} f_k + f_{n+1} = f_{n+2} - 1 + f_{n+1} \\ &= f_{n+2} + f_{n+1} - 2 = f_{n+3} - 1. \end{aligned}$$

Note that if a sentence ends with an equation, a point should be added at the end.

2.3 Tables

The following table was valid as of August 6th, 2010.

		World record	
Name	**Country**	**Event**	**Result**
Therese Alshammar	Sweden	50 m butterfly	25.07
Wilson Kipketer	Denmark	800 m	1:41.11
Jan Železný	Czech Republic	Javelin	98.48
Sergey Bubka	Ukraine	Pole vault	6.14