Appendix D

Mathematica

Mathematica is a computer algebra system, i.e. a computer program that can manipulate analytic expresssions like algebraic or differential equations.

D.1 Getting started

To start the graphical interface to Mathematica, use the command '`mathematica`'. After entering a line like

```
In[1]:=  Sin[x]^2 + Cos[x]^2 // Simplify
```

type Ctl-Ret to get the result.

For the plain-text interface, use the command '`math`'. Due to the stupidity of the Mathematica makers, there is no command-line history editing available in plain-text mode. You can work around this by running Mathematica from within *Emacs*: Start *Emacs*,

```
unix>  emacs
```

then, in the emacs window, type `\key{Esc}-x shell`, followed by Ret. You should get a shell prompt from which you can start '`math`'. Going back/forward in the command history is done by Esc-p and Esc-n (for **p**revious and **n**ext).

Another option is to run Mathematica through *rlwrap*.

D.1.1 Syntax basics

Notation in Mathematica is relative straight-forward. Comments are enclosed by starred brackets `(* like this *)`. Mathematical functions often have their usual name (but not e.g. arsinh, etc), but capitalized (e.g. `Sin`, `Exp`, `Ln`). Round brackets '`()`' are used for grouping, while square brackets '`[]`' indicate function or operator arguments:

```
In[1]:=  Sin[x]^2 + Cos[x]^2 // Simplify
In[2]:=  Expand[(x+y)^3]
In[3]:=  D[x^3*Exp[-x],x]
In[4]:=  Integrate[x*Sin[x], x]      (* indefinite integral *)
```

Grouping of arguments in *lists* is done with curly braces '`{}`':

```
{In[5]:= } Integrate[Sin[x]/x, {x, 0, Infinity}] (* definite integral *)
```

The result of the last operation can be recalled using '`%`', the second last output is '`%%`', etc.

A backslash '`\`' can be used for continuation lines, while a semicolon '`;`' separates multiple statements in own line.

D.2 Mathematical constants

`Pi`: $\pi \approx 3.141592653590$

`E`: $e \approx 2.718281828459$

`I`: $\mathrm{i} = \sqrt{-1}$

`Infinity`: ∞

`EulerGamma`: $\gamma \approx 0.577215664$

`Catalan`: $C \approx 0.9159655942$

D.3 Floating-point approximations

To get the first 40 digits of Catalan's constant, use the `N` (numerical value) function:

```
In[1]:=  N[Pi]        (* default precision [~5 digits] *)
In[1]:=  Pi // N      (* same thing *)
In[1]:=  N[Pi,40]     (* 40 digits *)
```

D.4 Some mathematical functions

`Sqrt[z]`: square root

`Sin[z],Cos[z],Tan[z],Cot[z]`: trigonometric functions

`ArcSin[z],ArcCos[z],ArcTan[z],ArcCot[z]`: cyclometric functions

`Sinh[z],Cosh[z],Tanh[z],Coth[z]`: hyperbolic functions

`ArcSinh[z],ArcCosh[z],ArcTanh[z],ArcCoth[z]`: area (inverse hyperbolic) functions

`Exp[z],Log[z]`: exponential function and natural logarithm

`Log[10,z]`: base-10 logarithm

More esoteric mathematical functions:

`BesselJ[n,z]`: Bessel function of first kind

`HermiteH[n,z]`: Hermite polynomial

`UnitStep[x]`: Heaviside step function

`DiracDelta[x]`: Delta function

`Fibonacci[z]`: Fibonacci number

`Prime[z]`: *N*th prime number

D.5 Algebra and simplification

```
In[1]:=  Nest[Log,x,3]              (* iterated log *)

In[1]:=  Solve[x^2-9==0, x]         (*yields {{x -> -3}, {x -> 3}} *)
In[1]:=  x ./ Solve[x^2-9==0, x]    (*yields { -3, 3}  *)

In[1]:=  FindRoot[Tan[x]==x, {x, 4.4}] (* numerical root-finding *)
In[1]:=  NRoots[x^4 - 3 x^3 + 1 == 0, x] (* numerically find all
                                            roots of polynomial *)

In[1]:=  Factor[x^2 + 4 c x + 4 c^2]
In[2]:=  Factor[x^2 + 9, GaussianIntegers->True]

In[1]:=  Expand[(x + 2 c)^2]

In[1]:=  Together[1/(1+x)+1/(1-x)]   (* common denominator *)
In[2]:=  Apart[1/(1-x^2)]            (* split fraction *)
```

```
In[1]:=  Collect[x y + x^2 - 2 (y+1) x, y]

In[1]:=  Simplify[Sin[x]^2 + Cos[x]^2]
In[1]:=  Sin[x]^2 + Cos[x]^2 // Simplify      (* ditto *)
In[1]:=  FullySimplify[Sin[x]^2 + Cos[x]^2]   (* similar *)

In[1]:=  a + Log[E^y] /. Log[E^z_] -> z       (* manual transform. rule *)
In[1]:=  Log[E^z_] -> z            (* global transformation rule
                                      [acts on all following expressions] *)
In[1]:=  fac[n_] := n fac[n-1]; fac[0] = 1
```

D.6 Calculus and similar

```
In[1]:=  D[Exp(-x)*x^n, x]                 (* derivative *)
In[2]:=  D[(x+y+z)^n, y]                   (* partial derivative *)

In[1]:=  Integrate[(x+y)^3, x]             (* indefinite integral *)
In[1]:=  Integrate[(x+y)^3,{x, 0, 2 y}]    (* definite integral *)

In[1]:=  NIntegrate[Sin[x]/Sqrt[x], {x,0,5}]  (* numerical integration *)
In[1]:=  NIntegrate[Sin[x]/Sqrt[x], {x,0,Infinity}, Method->Oscillatory]

In[1]:=  Sum[n^2, {k, 1, n}]               (* indefinite sum *)
In[2]:=  Sum[1/k^2, {k, 1, Infinity}]      (* definite sum *)

In[1]:=  Limit[Tan[x], x->Pi/2, Direction->1]  (* from left *)

In[1]:=  DSolve[y'[x]==y[x]+x, y[x], x]   (* solve differential equation *)
In[w]:=  DSolve[y''[x]+y[x]==0, y[x], x]  (* solve differential equation *)
```

D.7 Defining constants and functions

```
In[1]:=  c = 17                          (* constant *)
In[1]:=  f[x_] := Sin[x] Exp[x]          (* function *)
```

The ':=' *defers* evaluation to the time when 'f' is called.

D.8 Logical operators

== equality ('a == b')

!= inequality ('a != b')

<, >, <=, >=

D.9 Plotting

```
In[1]:=  Plot[Sin[x]/x, {x,-2,2}]        (* 2-dimensional *)
In[2]:=  Plot[{Sin[x], x, x-x^3/6, x-x^3/6+x^5/120}, {x,-2,2}]
In[3]:=  ParametricPlot[Sin[2 x], Cos[3 x], {x,0,2*Pi}]   (* parametric *)

In[1]:=  f[x_,y_] := Sin[2*x]/Exp[x]*Exp[-y^2]
In[2]:=  Plot3D[f[x,y], {x,-1,1}, {y,-1,1}]   (* surface plot *)
In[3]:=  ContourPlot[f[x,y], {x,-1,1}, {y,-1,1}]
In[4]:=  DensityPlot[f[x,y], {x,-1,1}, {y,-1,1}]
```

D.10 Evaluating expressions

```
In[1]:=  x^2 /. x -> 5          (* evaluate for x=5 without modifying x *)
In[2]:=  Cos[x] /. x -> {0, Pi}

In[1]:=  sol = Solve[x^3 + 3 x^3 -x + 5 ==0, x]
In[2]:=  x^3 + 3 x^3 -x + 5 /. sol   (* backsubstitution *)
```

D.11 List manipulation

```
In[1]:=  {2,3,4} + 7 + {0,-2,2}^2
In[2]:=  Join[{a,b}, {c,b,e}]    (* list concatenation -> {a,b,c,b,e} *)
In[3]:=  Union[{a,b}, {c,b,e}]   (* set union -> {a,b,c,e} *)

In[1]:=  {a,b,c,d,e}[[2]]        (* list element (counts from 1) -> b *)
```

D.12 Linear algebra

```
In[1]:=  {a,b} . {c,d}                    (* dot product *)
In[1]:=  Cross[{x1,x2,x3}, {y1,y2,y3}]    (* cross product *)
In[2]:=  {{a,b},{c,d}} . {e,f}            (* matrix product *)

In[1]:=  mat := {{0,-1},{1,0}}
In[2]:=  MatrixPower[mat,3]
In[3]:=  Table[%] // MatrixForm           (* nicer printing *)
In[4]:=  Table[%] // TraditionalForm

In[1]:=  Array[a, {3,3}]                  (* a[1,1] a[1,2] a[1,3]
                                             a[2,1] a[2,2] a[3,2]
                                             a[3,1] a[3,2] a[3,3] *)

In[1]:=  mat[[All,2]]                  (* second column *)
In[2]:=  mat[[2]]                      (* second row *)
In[3]:=  mat[[2,All]]                  (* second row *)

In[1]:=  Eigenvalues[mat]
In[1]:=  Eigenvectors[mat]
In[1]:=  Eigensystem[mat]
```

D.13 Miscellanea

D.13.1 A few more functions

```
In[1]:=  Clear[a, b, c]      (* clear values + defs for symbols *)
In[1]:=  Remove[a, b, c]     (* remove symbols completely *)
                             (* I have no clue what the difference is.. *)

In[1]:=  Permutations[{a, b, c}]

In[1]:=  Print["Brillig"]
```

D.13.2 How to make animations in Mathematica

Basically all animations in Mathematica consist of a sequence of plots which when rendered in rapid succession, give the appearance of motion. The easiest way to create an animation is to the use the 'Table' function.

Although you can visualize the animation directly in Mathematica, I suggest that you save the necessary files to PNG or (with older versions of Mathematica) GIF files. This way, you can use any software you wish to assemble these files and make the movies/animations.

Generating the graphs with Mathematica

1. Select the number of graphs, say n, you want to use for the animation (number of frames)
2. Select the command that will generate the graphs. Choose a name for each of these graphs, say $fig[k]$, where k is a number between 1 and n.
3. Use the following command to generate each and name each picture:

   ```
   n = (*enter here the number of frames*);
   Evaluate[Map[fig,Range[n]]]=(*enter cmd to generate the graphs*)
   ```

   Make sure that the number of pictures that your command generates is equal to n.
4. Select a path and directory to save all the graphs. Each graph must be saved in GIF format. We can do this at once for all graphs using the following command

   ```
   Display[StringJoin["*enter path and directory here*/fig[",
       ToString[#],"].gif"],fig[#],"GIF"]&/@
   ```

```
Range[n]
```

This command will save all the graphs in the directory you selected under the names $fig[k].gif$, for each k between 1 and n.

Example: The following command in Mathematica will generate red graphs of the function $\sin x$ on the intervals $[-a, a]$ for 10 different values of a, with black background and without the coordinate axes.

Red graphs

```
n = 10;
Evaluate[Map[fig, Range[n]]] =
    Table[Evaluate[Plot[ Sin[x],
                         {x, -4Pi*k/n , 4Pi*k/n},
                         PlotStyle→RGBColor[1,0,0],
                         Axes→False,
                         Background→RGBColor[0,0,0]]],
          {k, 1, n}]
Display[StringJoin[``/home/rouyed/Save_Files/fig[``,
                   ToString[#],
                   "].gif"],
        fig[#], "GIF"]&/@
Range[n]
```

Now it is up to you to assemble the graphs with whatever software you have at hand.

D.13.3 3D Animation

Try this notebook on your machine:

Red graphs

```
<< Graphics`
aa = SurfaceOfRevolution[Sin[x],
                         {x, -Pi, Pi},
                         RevolutionAxis → {1, 0, 0},
                         Boxed → False,
                         Axes → False]
AbsoluteOptions[aa, ViewPoint]
dist  =  Sqrt[1.3^2 + 2.4^2]
Table[Show[aa,
           ViewPoint → {dist*Cos[t], dist*Sin[t], 2},
           PlotRange → {{- Pi, Pi}, {- Pi, Pi}, {-1, 1}},
```

```
        SphericalRegion → True],
    {t, Pi/12, 2*Pi, Pi/12}]
```