Lecture 4: Introduction to Mathematica III

Simplifying and Picking Apart Expression, Calculus, Numerical Evaluation

A great advantage of using a symbolic algebra software package like MATHEMATICA® is that it reduces or even eliminates errors that inevitably creep into pencil and paper calculations. However, this advantage does come with a price: what was once a simple task of arranging an expression into a convenient form is something that has to be negotiated with MATHEMATICA® . In fact, there are cases where you cannot even coerce MATHEMATICA® into representing an expression the way that *you* want it.

A MATHEMATICA® session often results in very cumbersome expressions. You can decide to live with them, or use one of MATHEMATICA® 's many simplification algorithms. Section 1.4.5 (or Help Browser/The Mathematica Book/A Practical Introduction/Algebraic Calculations/Advanced Topic: Putting Expressions into Different Forms) of the MATHEMATICA® book has a good summary of frequently used simplification algorithms. Another method is to identify patterns and replace them with your own definitions.

MATHEMATICA® has its own internal representation for rational functions (i.e., $\frac{\text{numerator expression}}{\text{denominator expression}}$) and has special operations for dealing with these. Generally, advanced simplification methods usually require a working knowledge of of MATHEMATICA® 's internal representations.

Lecture 04 MATHEMATICA® Example 1

Operations on Polynomials

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

There are built-in simplification operations, such as `Simplify`, but they will not always result in a form that is most useful to the user. Crafting an expression into a pleasing form is an artform.

```
1  PaulENomeal = (1 + 2a + 3x + 4z)^4
2  FatPEN = Expand[PaulENomeal]
3  Factor[FatPEN]
4  PaulinX = Collect[FatPEN, x]
5  Coefficient[PaulinX, x, 0]
6  PaulSpiffedUp = Sum[
     Simplify[Coefficient[PaulinX, x, i]] x^i, {i, 0, 20}]
7  Simplify[PaulSpiffedUp]
8  RashENell = (x+y)/(x-y) + (x-y)/(y+x)
9  Apart[RashENell]
10 Together[RashENell]
11 Apart[Together[RashENell]]
12 Numerator[Together[RashENell]]
13 Simplify[RashENell]
14 Factor[RashENell]
15 RootBoy = Sqrt[(x+y)^2]
16 Simplify[RootBoy]
17 Simplify[RootBoy, x ∈ Reals && y ∈ Reals]
18 Simplify[RootBoy, x ≥ 0 && y ≥ 0]
19 RootBoy /. Sqrt[(expr_)^2] → expr
```

2: `Expand` performs all multiplication and leaves the result as a sum.

3: `Factor` has an algorithm to find common terms in a sum and write the result of a factor and a cofactor.

4: `Collect` will turn in an expression into a polynomial in a user-selected variable.

5: `Coefficient` picks out coefficients of user-specified powers of a variable.

6: This is an example of using `Simplify` to operate only upon on a polynomial coefficients

8: Besides polynomial, other frequently encountered forms are *rational forms*.

9: `Apart` will re-express a rational form as a sum.

10: `Together` will collect all terms in a sum into a single rational form.

15: MATHEMATICA® is fastidious about simplifying roots and makes no assumptions—unless they are specified— about whether a variable is real, complex, positive, or negative.

16: Many users become frustrated that `Simplify` doesn't do what the user thinks must be correct...

17: `Simplify` will accept `Assumptions`.

19: This is brute force—and not really a good idea.

Lecture 04 MATHEMATICA® Example 2

A Second Look at Calculus: Limits, Derivatives, Integrals

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

Examples of `Limit` and calculus with built-in assumptions

2: This would be a challenging limit to find for most first-year calculus students.

5: This definite integral results in a fairly complicated symbolic form which *should* be real because it should be the value of `AMessyExpression` at $x = e$ but not obviously so by inspection. Looking at the numerical version gives a hint. (n.b. `Chop` is a useful way to remove small numerical inaccuracies.)

7: It took some effort, but this gives an aesthetic form for the solution.

9: Some indefinite do not have closed-form solutions, even with extra assumptions...

12: But, in some cases, the definite integral will have a closed-form solution.

14: `Series` is one of the most useful and powerful MATHEMATICA® functions; especially to replace a complicated function with a simpler approximation in the neighborhood`Series` of a point.

15: `Normal` converts a `SeriesData` form by chopping off the *trailing order* function `O`.

```
1  AMessyExpression = Log[x Sin[x]] / (1/x)
2  Limit[AMessyExpression, x → 0]
3  DMess = D[AMessyExpression, x]
4  Integrate[DMess, x]
5  Integrate[DMess, {x, 0, e}] // N
6  (AMessyExpression /. x → e) - (AMessyExpression /. x → 0)
7  (AMessyExpression /. x → e) -
     Limit[AMessyExpression, x → 0]
8  e Log[e Sin[e]] // N
9  Integrate[Sin[x] / Sqrt[(x^2 + a^2)], x]
10 Integrate[Sin[x] / Sqrt[(x^2 + a^2)], x, Assumptions → a ≥ 0]
11 Integrate[Sin[x] / Sqrt[a^2 + x^2], x, Assumptions → Re[a^2] > 0]
12 UglyInfiniteIntegral = Integrate[Sin[x] / Sqrt[(x^2 + a^2)],
     {x, 0, ∞}, Assumptions → Re[a^2] > 0]
13 N[UglyInfiniteIntegral /. a → 1]
```

The Taylor expansion capabilities in *Mathematica* are very useful

```
14 Series[AMessyExpression, {x, 0, 4}]
15 FitAtZero = Series[AMessyExpression, {x, 0, 4}] // Normal
16 Plot[{AMessyExpression, FitAtZero}, {x, 0, 3}, PlotStyle →
     {{Thickness[0.02], Hue[1]}, {Thickness[0.01], Hue[0.5]}}]
```

Lecture 04 MATHEMATICA® Example 3

Solving Equations

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

`Solve`, its resulting rules, and how to extract solutions from the rules.

```
1 TheEquation = a x^2 + b x + c
```

Note the use of **Equal** (==) rather than **Set** (=) in the following; using "=" will produce an error message.

```
2 TheZeroes = Solve[TheEquation == 0, x]
```

Note that the roots are given as **Rules**. Now we ask *Mathematica* to verify that the solutions it found are indeed roots to the specified equation. Here is a prototypical example of using **Replace** (/.) to accomplish this:

```
3 TheEquation /. TheZeroes
4 Simplify[TheEquation /. TheZeroes]
```

More examples of using **Solve**:

```
5 a[i_] := i + 1
6 TheQuinticEquation = Sum[a[i] x^i, {i, 0, 5}]
7 Solve[TheQuinticEquation == 0, x]
8 Quad1 = a x^2 + y + 3
  Quad2 = a y^2 + x + 1
9 Solve[{Quad1 == 0, Quad2 == 0}, {x, y}]
```

2: `Solve` takes a *logical equality* (or logical equalities) as a first argument. It returns a list of solutions in the form of rules. Here, the list of ruls is assigned to `TheZeroes`.

3: If the resulting solution rules are applied original equation, the zeroes specified in the logical equality in item 2 should result.

7: The zeroes of a quintic polynomial do not have general closed forms. Here MATHEMATICA® will return a symbolic representation of the solution rules. This representation indicates that the solution doesn't have a closed form, but the form is suitable for subsequent numerical analysis.

9: This is an example of a solution to coupled quadratic equations: there are four solutions.

Sometimes, no closed form solution is possible. MATHEMATICA® will try to give you rules (in perhaps a very strange form) but it really means that you don't have a solution to work with. One usually resorts to a numerical technique when no closed form solution is possible— MATHEMATICA® has a large number of built-in numerical techniques to help out. A numerical solution is an approximation to the actual answer. Good numerical algorithms can anticipate where numerical errors creep in and account for them, but it is always a good idea to check a numerical solution to make sure it approximates the solution the original equation.

Of course, to get a numerical solution, the equation in question must evaluate to a number. This means if you want to know the numerical approximate solutions $x(b)$ that satisfy $x^6 + 3x^2 + bx = 0$, you have to iterate over values of b and "build up" your function $x(b)$ one b at a time.

Sections 1.6.1–1.6.7 (or Help Browser/The Mathematica Book/A Practical Introduction/Algebraic Calculations/Numerical Mathematics) of the MATHEMATICA® book have an overview of frequently used numerical algorithms.

Lecture 04 MATHEMATICA® Example 4

Numerical Algorithms and Solutions

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

Examples of numerical algorithms `NIntegrate` `FindRoot`

3: `NIntegrate` can find solutions in cases where `Integrate` cannot find a closed-form solutions. It is necessary that the integrand should evaluate to a number at all points in the domain of integration (it is possible that the integrand could have singularities at a limited set of isolated points).

4: `NSolve` will find roots to *polynomial forms*, but not for more general expressions.

5: `FindRoot` will operate on general expressions and find solutions, but additional information is required to inform where to search.

```
Numerical Solutions
1 Integrate[Sin[x]/Sqrt[(x^2 + a^2)], x]
2 Integrate[Sin[x]/Sqrt[(x^2 + a^2)], {x, 0, 1}]
3 NIntegrate[(Sin[x]/Sqrt[(x^2 + a^2)]) /. a → 1, {x, 0, 2 Pi}]
4 Plot[
    NIntegrate[Sin[x]/Sqrt[(x^2 + a^2)], {x, 0, 2 Pi}], {a, 0, 10}]
5 Plot[{AMessyExpression, FitAtZero}, {x, 0, 3}, PlotStyle →
    {{Thickness[0.02], Hue[1]}, {Thickness[0.01], Hue[0.5]}}]
6 NSolve[AMessyExpression == 0, x]
7 FindRoot[AMessyExpression == 0, {x, .5, 1.5}]
8 FindRoot[FitAtZero == 0, {x, .5, 1.5}]
9 FindRoot[AMessyExpression == 0, {x, 2.5, 3}]
```

1. You will want to save your work.
2. You will want to modify your old saved work
3. You will want to use your output as input to another program
4. You will want to use the output of another program as input to MATHEMATICA® .

You have probably learned that you can save your MATHEMATICA® notebook with a menu. This is one way to take care of the first two items above. There are more ways to do this and if you want to do something specialized like the last two items, then you will have to make MATHEMATICA® interact with files. Because an operating system has to allow many different kinds of programs interact with its files, the internal operations to do input/output (I/O) seem somewhat more complicated than they should be. MATHEMATICA® has a few simple ways to do I/O—and it has some more complex ways to do it as well.

It is useful to have a few working examples that you can modify for your purposes. The examples will serve you well about 90% of the time. For the rest of the 10%, one has to take up the task of learning the guts of I/O—hopefully, beginners can ignore the gory bits.

Lecture 04 MATHEMATICA® Example 5

Interacting with the Filesystem

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

Reading and writing data directly and through the use of a *file stream*. A user should check (and sometimes) change the *working directory* to interact with files using `Directory` or `SetDirectory`. Otherwise, the full path to a file must be given.

1: Simple redirection of an expression into a file is acheived with `>>` The working directory must be *writable*. Selected symbols can be saved in files all at once using `Save`.

3: A file containing a MATHEMATICA® expression can be read in with `<<` The file must be *readable*.

7: The contents of a file can be displayed using `!!`.

9: This opens a filestream for subsequent use. The use of filestreams is useful for cases where data is written incrementally during a calculation and this method can be generalized to different kinds of devices. Another use of file streams is when the user wants to have the program compute the file name.

11: An example of writing with a file stream.

15: It is good practice to close open file streams when writing is finished.

File Input and Output

```
1  AMessyExpression >> AFile.m
2  Clear[AMessyExpression]
3  << Afile.m
4  AMessyExpression
5  AMessyExpression = << AFile.m
6  AMessyExpression
7  !! Afile.m
8  Close["ANewFileName"]
9  AFileHandle =
     OpenWrite["ANewFileName", FormatType → OutputForm]
10 RandomPairs = Table[{Random[], Random[]}, {i, 20}]
11 Write[AFileHandle, RandomPairs]
12 !! ANewFileName
13 Write[AFileHandle, MatrixForm[RandomPairs]]
14 !! ANewFileName
15 Close[AFileHandle]
```

Lecture 04 MATHEMATICA® Example 6

Using Packages

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

There are a number of packages that come with MATHEMATICA® (and more that can be bought for special purposes). You should look through the various packages in the help browser to get an idea of what is there—it is also a good idea to take a look at the inside of a package by editing a package file with an editor. By doing this, you will see some of internal structure of MATHEMATICA® and good examples of professional programming.

1: A package is read in using the input operator `<<` or with `Needs`. After reading a bit about a package, it is straightforward to construct simple examples such as in this example of the `WorldPlot` subpackage in the `Miscellaneous` package.

```
Fortunately, others have gone to the trouble of writing files full of
useful stuff--and you can load this stuff into Mathematica for your
very own use.  Some people produce useful stuff and you can
buy it, which is nice if you find it valuable--and you can write stuff
and gain value by selling it, which might be even more nice.
Mathematica comes with a group of Standard Packages, that
you can load in to do special tasks.  The Packages are listed
under "Add-ons & Links" in the Help Browser.  For example, take
a look at the specialized package under "Miscellaneous" called
"World Plot"...

1 | << Miscellaneous`WorldPlot`

2 | WorldPlot[{{"USA", "France", "Germany", "Italy", "Belgium",
        "Luxembourg", "Switzerland"}, {RGBColor[1, 0, 0],
      RGBColor[0, 0.5, 0], RGBColor[0.5, 0, 0],
      RGBColor[0, 0, 0], RGBColor[0.4, 0.4, 0.1],
      RGBColor[0, 0, 0.5], RGBColor[0.9, 0.6, 0.6]}}]

3 | WorldPlot[{{"USA", "France", "Germany", "Italy", "Belgium",
        "Luxembourg", "Switzerland"}, {RGBColor[1, 0, 0],
      RGBColor[0, 0.5, 0], RGBColor[0.5, 0, 0],
      RGBColor[0, 0, 0], RGBColor[0.4, 0.4, 0.1],
      RGBColor[0, 0, 0.5], RGBColor[0.9, 0.6, 0.6]}},
    WorldProjection --> Mollweide]

4 | WorldPlot[{{"USA", "France", "Germany", "Italy", "Belgium",
        "Luxembourg", "Switzerland"}, {RGBColor[1, 0, 0],
      RGBColor[0, 0.5, 0], RGBColor[0.5, 0, 0],
      RGBColor[0, 0, 0], RGBColor[0.4, 0.4, 0.1],
      RGBColor[0, 0, 0.5], RGBColor[0.9, 0.6, 0.6]}},
    WorldProjection → LambertAzimuthal]
```