AN INTRODUCTION TO MAPLE

CHRISTOPHER J HILLAR

Introduction

This document introduces Maple, a commercial computer program for mathematical computation. You can learn about Maple from its website `http://www.maplesoft.com/`. The numerical, symbolic, and graphical features of Maple are prodigious; as such, its usefulness to the working mathematician goes without saying. This guide is meant to be a gentle introduction to some of the powerful features that this system provides.

Once installed, you cay run Maple either in shell mode by typing `maple` in a terminal or by typing `xmaple` which runs Maple in a graphical user interface. As with most high-level computer systems, Maple has its own programming language that allows users to make their own functions and libraries, building on Maple's internal procedures.

At the end of this document are several exercises of varying levels of difficulty.

Acknowledgments

This tutorial is modeled after the Singular tutorial by Luis Garcia.

1. First steps

Once Maple is started, it awaits an input after the prompt >. Every statement has to be terminated by ; (or : if you would like to suppress printing the output).

```
5^2+2*3-1;
# 30
```

The symbol # is used by Maple to indicate a comment, and any text following it is disregarded when executing the line.

All objects have a type, however, unlike some other systems (such as Singular), Maple keeps track of types internally (there is an exception when one uses Maple's programming language; we will see this later on). This might create some confusion when different types interact (and it does), but in general, Maple's type conversion decisions are pretty good.

An assignment is done with the symbol :=, while comparison is accomplished by using the symbols = or <> along with the function `evalb` (again, this will change slightly when programming).

```
text := "Hello World!":
print(text);
evalb( text = "Hello World!" );
num1 := 42.000:
num2 := 42:
evalb( num1 <> num2 );
```

```
convert(num1 + num2,string);

# "Hello World!"
# true
# false
# "84.000"
```

The last displayed result is always available with the symbol % (second to last with %% and so on). This is particularly useful if you are doing an long interactive computation and you forgot to store the result in a variable.

```
z3 := 3:
f := (x-z1)*(x-z2)*(x-z3);
eval( f,{x=z1} );
eval( f,{z1=t,z2=2} );
eval( %,{x=4});
g := expand(%%);
sort(collect(g,x),x);

# f := (x - z1) (x - z2) (x - 3)
# 0
# (x - t) (x - 2) (x - 3)
# 8 - 2t
# g := x^3  - 5 x^2  + 6 x - t x^2  + 5 t x - 6 t
# x^3  + (-t - 5) x^2  + (5 t + 6) x - 6 t
```

The previous example shows many Maple features which are worth describing. The first line defines a variable $z3$ to be the integer 3 (and suppresses the output). Next, a polynomial expression in the variables $x, z1, z2$, and $z3$ is defined using the variable f. Notice that Maple understands that $z3$ has already been assigned, and thus it replaces each instance of $z3$ with this value. Unlike Singular, Maple needs no ring declarations; you can bring any variables you choose into the worksheet at any time.

In the next line, the function is `eval` is used to take an expression and *evaluate* it upon substitution of variables with other expressions (possibly involving other variables). In this case, we see that f evaluates to 0 when $x = z1$. It is important to realize that the function `eval` does not change f in any way, which gives us its versatility. The command `expand` takes an expression and returns its most, well, expanded form. Finally, we have used the two functions `collect` and `sort` to visualize the polynomial g as an expression in x with the coefficients (in t) collected.

We should remark that this last line could also have been generated by making a call to the useful function `coeff`:

```
coeff(g,x,3)*x^3 + coeff(g,x,2)*x^2 + coeff(g,x,1)*x + coeff(g,x,0);
# x^3  + (-t - 5) x^2  + (5 t + 6) x - 6 t
```

The best way to learn Maple is to read the online documentation either through Maple's web site or through the many tutorials on the web. Within the program, you can access

help by preceding a key word with the symbol ?. For instance, to learn more about the function `coeff`, we would type:

```
?coeff

# coeff(p, x)
# coeff(p, x, n)
# coeff(p, x^n)
#
# Parameters
# p - polynomial in x
# x - variable (an expression)
# n - (optional) an integer
#
# Description
# -The coeff function extracts the coefficient of x^n in the polynomial p.
.........
```

Maple provides some useful built-in data structures such as arrays and matrices. To work with matrices as algebraic objects, however, one must include the linear algebra library by typing the following command. Its output should contain the list of functions within the library.

```
with(linalg);
```

We begin by defining an array. One way is the following.

```
n := 5:
arr := array(1..n):
arr[3] := "3rd":
print(arr);
# [arr[1], arr[2], "3rd", arr[4], arr[5]]
```

In the case of matrices, we might create a 3×3 matrix in two ways as follows.

```
M := matrix(3,3,[1,2,3,4,5,6,7,8,9]):
N := [[1,2,3],[4,5,6],[7,8,9]]:
evalb( M = N );
M[1,2] := 0:
evalm(M);
trace(M);
det(M);
evalm(det(M)*inverse(M));

# false
#      [ 1 0 3 ]
# M := [ 4 5 6 ]
#      [ 7 8 9 ]
# 15
# -12
```

```
# [ -3 24 -15 ]
# [  6 -12  6 ]
# [ -3 -8   5 ]
```

This last example reveals that one must be careful when dealing with matrices and equality in Maple. In general, you might need to use the function `evalm` to convert a matrix from its data structure representation to its algebraic object (for instance, to evaluate $M + N$ as a matrix, one would type `evalm(M+N)`).

Yet another way to make a matrix (or array) is with a loop.

```
P := matrix(3,3):
for i from 0 to 2 do
  for j from 1 to 3 do
     P[i+1,j] := 3*i+j;
  end;
end;
evalm(P);
# creates the same matrix as M
```

To format the output of your worksheet, the command `print` is useful. Below, the concatenation operator || is used to combine objects into one string.

```
print("The number n has value " || n);
# "The number n has value 5"
```

You can plot curves and surfaces in Maple with the `plot` command. Before getting to more technical stuff, let's create the pictures from the first chapter of the notes at `www.math.tamu.edu/~sottile/conferences/Summer_IMA07/Sottile.pdf`.

```
with(plots):
# plane curves
p := y^2-x^3-x^2:
implicitplot(p,x=-2..2,y=-2..2);
implicitplot(p+1/20,x=-2..2,y=-2..2);
implicitplot(p-1/20,x=-2..2,y=-2..2);

# hyperboloids
hp := x*y+z;
hos := x^2-x+y^2+y*z;
implicitplot3d(hp,x=-2..2,y=-2..2,z=-2..2);
implicitplot3d(hos,x=-2..2,y=-2..2,z=-2..2);

solve({hos,hp},{x,y,z});  # finds intersection parametrically
```

2. Equation Solving in Maple

Maple has many algebraic and numerical tools for finding the solutions to systems of (not necessarily polynomial) equations. Perhaps the simplest black-box solver is provided by the appropriately named function `solve`. As a first example, we will solve the following system of linear equations in x, y, z and u and free parameters a, b, c and d.

$$\begin{aligned} 3x + y + z - u &= a \\ 13x + 8y + 6z - 7u &= b \\ 14x + 10y + 6z - 7u &= c \\ 7x + 4y + 3z - 3u &= d \end{aligned}$$

To input and solve this system in Maple, we execute the following line of code.

```
solve({3*x+y+z-u-a, 13*x+8*y+6*z-7*u-b, 14*x+10*y+6*z-7*u-c,
       7*x+4*y+3*z-3*u-d},{x,y,z,u});
```

In general, the first input parameter to `solve` is a list of expressions presented in the form $\{f_1, \ldots, f_n\}$ followed by a list of variables to "solve" for. In this case (and generally), the output from the calculation is given as (sequences of) lists such as

$$\begin{aligned} \{u &= -(6/5a + 4/5b + 1/5c - 12/5d), \\ z &= -(16/5a - 1/5b + 6/5c - 17/5d), \\ y &= -(3/5a + 2/5b - 2/5c - 1/5d), \\ x &= -(-6/5a + 1/5b - 1/5c + 2/5d)\} \end{aligned}$$

For those interested in a matrix approach to solving these equations, the following code is appropriate.

```
M := [[3,1,1,-1],[13,8,6,-7],[14,10,6,-7],[7,4,3,-3]]:
v := matrix(4,1,[a,b,c,d]):
evalm(inverse(M)&*v);    # &* is used to multiply two matrices
```

It is constructive to play around with the function `solve` to see what it does on various types of inputs. For instance, how does it handle equations with an infinite number of solutions or for which there are no solutions?

```
# degenerate linear equation
solve({t*x + t^2*y - t, x + t*y - 1},{x,y});
# {x = -t y + 1, y = y}

# linear equation with no solution
solve({t*x + t^2*y - 0, x + t*y - 1},{x,y});
```

For the first example, Maple returns a solution indicating that y is a free parameter and that x depends on this parameter in the manner indicated. In the second example, `solve` outputs nothing, indicating that there is no solution to the system of equations. In general, however, one must be skeptical of the output from the function `solve`. It does its best with the tools it has, but there is no guarantee that it is giving you all correct solutions. In the simple example below, Maple (incorrectly) does not exclude the case $(x, y) = (0, 0)$ in its list of solutions.

```
solve({y/x},{x,y});
# {x = x, y = 0}
```

A companion to the function `solve` is a numerical solver `fsolve` which is called in the same way as `solve`. An important difference, however, is that the number of equations must match the number of variables or else an error message will result.

```
fsolve(x^3-3*x+1,x);
# -1.879385242, .3472963553, 1.532088886

evalf(fsolve(x^3-3*x+1,x),20);
# -1.8793852415718167681, 0.34729635533386069770, 1.5320888862379560704
```

Above, we used another function `evalf` to extend the accuracy of our solution to 20 digits. This function is also important when one has a symbolic solution to an equation that one wants to see approximately. For instance, we could have solved the above equation first symbolically and then approximated the formula numerically.

```
solve(x^3-3*x+1,x);
evalf(%);
# 1.532088886 - 0.1*10^(-9) I, -1.879385241 - 0.1732050808*10^(-9) I, ...
```

The careful reader will notice that the output from these two examples is different – there is a very small imaginary part (the complex number $\sqrt{-1}$ is denoted by I in Maple) that is detected by the second command. Moreover, increasing the number of digits of accuracy does not remove the problem. So how do we resolve this issue? Fortunately, there is a built-in function called `realroot` which gives (provably correct) bounding boxes for the real roots of univariate polynomials.

```
e := 1:
realroot(x^3-3*x+1, e);
# [[0, 1], [1, 2], [-2, -1]]
```

The output here indicates that there are at least three real roots to the given equation and that they lie inside each of the indicated intervals. The variable e is used to bound the size of the isolating intervals. Setting $e = 1/100$, for instance, will produce bounding intervals that are of length at most $1/100$:

$$[[\frac{11}{32}, \frac{45}{128}], [\frac{49}{32}, \frac{197}{128}], [-\frac{241}{128}, -\frac{15}{8}]]$$

We close this section with the function `eliminate`, which is very useful for human-guided equation solving.

```
# intersecting hyperboloid and sphere
hyp := x^2+(y-3/2)^2-z^2-1/4;
sph := x^2+y^2+z^2-4;
eliminate( {hyp, sph}, x);
# [{x = RootOf(_Z^2+y^2+z^2-4)}, {-6+3*y+2*z^2}]
```

In this example, the variable x is eliminated from the equations, resulting in the new equation $-6+3y+2z^2 = 0$ listed in the output. Thus, we see that with z a free variable, we have $y = (6-2z^2)/3$ and x is given by the funny expression $x = RootOf(\_Z^2+y^2+z^2-4)$. Although confusing, this is just Maple's way of symbolically representing roots of univariate

polynomials. Translated into mathematics, the expression is simply saying that given y and z, the value for x can be chosen to be a root of the equation $Z^2+y^2+z^2-4=0$. While this might seem cumbersome, it is unavoidable. For instance, how would one represent and manipulate the solution to an unsolvable quintic?

```
soln := solve(x^5-x+1,x);
a := soln[1];
simplify(a^10);
```

There are other solving tools available such as the Gröbner basis libraries included in later versions of Maple. However, we refer to `www.mapleprimes.com/blog/roman-pearce/improvements-to-maple-11-groebner` which has numerous examples displaying the power of Faugere's FGb program implemented in Maple 11.

3. Basic Programming in Maple

We have already encountered some of the building blocks for programming in Maple. For instance, we saw how to use `for` loops to fill the entries of a matrix. We will now explain how to write our own callable functions in Maple. Our first example will be a function to compute the factorial $n!$ of an integer n.

```
fact := proc(n)
 local i,prod;
 if ( n < 0 ) then
    return("n must be positive");
 end;
 prod := 1;
 for i from 1 to n do
   prod := i * prod;
 end;
 return prod;
end;
```

The first line above tells Maple that the name of our function is `fact` and that it takes one input. To call our function once we have entered the text above into Maple, we simply type

```
fact(3);
fact(-1);
# 6
# "n must be positive"
```

The second line in our procedure is there so that Maple knows only to use the indicated variables within a function call to `fact` and nowhere else in your worksheet. For a recursive implementation of $n!$, one has the following code.

```
fact := proc(n)
 if ( n = 1 ) then
    return(1);
 end;
 return n * fact(n-1);
```

```
end;
```

Another type of loop is the `while` loop.

```
collatz := proc(n)
  local m,count;
  count := 0;
  m := n;
  while ( m > 1 ) do
    if ( modp(m,2) = 0 ) then
     m := m/2;
    else
     m := 3*m+1;
    end;
    count := count + 1;
  end;
return count;
end;
```

This function returns the number of iterations required to bring an integer n down to 1 using the Collatz recurrence. It is a deep open problem to determine if the `while` loop above always breaks.

Finally, we mention a useful package `combinat` in Maple for indexing over permutations and combinations.

```
with(combinat);

# indexing over all combinations
C := choose(5,2);
for c in C do
  print(c);
end;

# indexing over all permutations
P := permute(3);
for p in P do
  print(p);
end;

# indexing over all subsets
S := subsets({x,y,z}):
while not S[finished] do
  S[nextvalue]()
end;
```

4. Exercises

(1) Write a function which takes as input a square matrix and its size and returns its trace.

(2) Find all solutions to the system $\{x^2+y^2+z^2=1, x^2+z^2=y, x=z\}$.

(3) How many real roots does the equation $x^{100}-x^{51}+x-2$ have? (Could you have predicted this before-hand?).

(4) Find out how to factor polynomials in Maple and then factor x^4+4. Compare this with the following command: `evala(AFactor(`x^4+4`));`

(5) Suppose that we have Maple variables $x[1], x[2]$, and $x[3]$; we shall consider polynomials f in these three variables. Write a function that takes as input a polynomial f and a permutation σ (expressed in word form as an array such as [1,3,2]) and returns the polynomial produced from f by permuting the variables according to σ. For example, if $f = x[1]^2 + x[2]x[3]$ and $\sigma = [3,1,2]$, then we would like to return

$$\sigma f = x[3]^2 + x[1]x[2].$$

(6) a) Let $f = x^3 - 3x + 1$. Look up the Maple commands to find the Galois group of f. b) Let a be a root of f in the interval $[1,2]$. Given that there are polynomials $g(t)$ and $h(t)$ with rational coefficients such that $f(x) = (x-a)(x-g(a))(x-h(a))$, find the polynomials g and h.

(7) If you are familiar with Gröbner bases, look up how Maple handles these computations. Work out explicitly some Gröbner bases for the polynomials $\{xy+z, x^2-x+y^2+yx\}$ using different term orders. Compare the performance of other examples with that of Singular and Macaulay 2.

Department of Mathematics, Texas A&M University, College Station, TX 77843-3368, USA

E-mail address: chillar@math.tamu.edu

URL: http://www.math.tamu.edu/~chillar