MATH549 Exercise Sheet 4

Deadline for submission: Monday 27th October
Please try to do as much of this sheet as you can before the tutorial

1 Introduction

This first sheet on Maple works in much the same way as the LaTeX sheets – you should work through the steps sequentially, ending up with a file to email to me by the deadline above.

Maple is an extensive program, and it isn't possible to cover all of its features in these sheets. The same is true of LaTeX, but at least with LaTeX there's a "common core" of features which can be taught in an introductory course, and which are enough to typeset most documents. With Maple, the particular features you are likely to need depend very much on your field of study.

One of the most important skills to learn is therefore how to use Maple's help system effectively. The Maple sheets are designed with this in mind. Much of this sheet is taken up with sequences of Maple commands for you to type in. **Please don't just type them in and go on to the next section: look up the commands in the help system, study the additional examples given there, and experiment with other uses of the commands. Only by doing this will you learn to use Maple well.** When it comes to the exercises, some of them just require you to do things which are similar to what you've already seen, but others require you to search the help system to find relevant commands.

There are hints for some of the harder exercises on the module webpage. **Please don't turn to the hints until you've spent some time attempting the exercises without them.** Remember that there is no credit given for your solutions to these problem sheets, and you'll learn Maple better if you do the exercises without help.

2 A Brief Tour of Some Maple Commands

The aim of this section is to give you an idea of some of the things that Maple can do. Please don't just type in exactly the commands which you find on this sheet: experiment with them, looking them up in the online help system if necessary.

2.1 Using Maple as a calculator

Start up the "Maple 11 Classic Worksheet" as you did in Sheet 3. *Please* make sure you're using the Classic Worksheet version (with an orange icon), rather than the version with a red icon. Type the following in the Maple window (don't type the `>` at the beginning of each line – this is the prompt which Maple gives you – but do type the semicolon at the end of each line).

```
>1+1;
>2^5;
>2^1000;
>factorial(500);
>2/3+1/4;
>2.0/3+1/4;
>Pi^2;
>evalf(Pi^2);
>evalf(Pi^2,1000);
```

Note that Maple works with *exact* arithmetic (using integers and fractions) wherever possible. However, if you give it *floating point* numbers as input (for example, `2.0` rather than `2`), then it will produce (approximate) floating point results, to as many decimal digits as you require. The `evalf` command produces a floating point approximation.

```
>?evalf;
```

Browse through the help entry for `evalf`, clicking on some of the links within the help topic. Note in particular the `Examples` section: often looking at such examples is the quickest way to understand how a command works. I encourage you to look at the help pages for each command which you use in this Brief Tour (and afterwards), and to experiment as much as possible with each command. Note that you can also get help on a command by putting your cursor anywhere within the command in your worksheet and hitting `F1`.

Return to your Maple worksheet by selecting `Untitled (1)` from the `Window` menu (or close the help page by typing `CTRL-F4`: this is a good idea as otherwise you'll find you soon have 25 help pages open). Save your worksheet with a filename such as `test`. You should do all your experimentation in this worksheet, which you won't submit to me: you'll do the exercises which come later in a separate worksheet, which is what you'll submit.

```
>sin(Pi/4);
>sin(1);
>evalf(sin(1));
>arcsin(1/2);
>arcsin(2);
>evalf(arcsin(2));
>evalc(arcsin(2));
>5/16<8/27;
>evalb(5/16<8/27);
>ifactor(243290200817664);
>isprime(243290200817664);
>nextprime(243290200817664);
>isprime(243290200817737);
```

2.2 Variables

The following lines show how you can assign values to *variables:*

```
>a:=7; # set the variable a equal to 7
>a; # show the value of variable a
>b:=a*5;
>a:=a^2: # note the colon, which suppresses output.
>a;
>a*b;
>restart;
>a;
```

is the comment symbol in Maple, which works just like % in LaTeX. There's no need for you to type these comments: I put them in when there's something I want you to notice about the examples. The `restart` command clears all variables, as if you'd restarted Maple.

2.3 Working with functions

```
>f:=(x+1)^6 + (x-1)^6;
>g:=expand(f);
>factor(f);
>factor(g);
>eval(f,x=0);

>h:=diff(f,x); # The derivative of f with respect to x.
>k:=subs(x=t,h);
>int(h,x);  # indefinite integral w.r.t. x
>int(h,x=-1..0);  # definite integral

>ComplicatedFunc:=sin(x^2)*exp(-3*cos(x^3)+log(x^2+1))/(1+exp(x));
>complicatedfunc;
>ComplicatedFunc; # Maple is case-sensitive
>simplify(ComplicatedFunc);
>eval(ComplicatedFunc,x=0);
>eval(ComplicatedFunc,x=2);
>yikes:=diff(ComplicatedFunc,x);
>latex(yikes);
>int(ComplicatedFunc,x=0..1); # Even Maple can't do impossible things
>evalf(int(ComplicatedFunc,x=0..1)); # But it can approximate them

>TS:=series(ComplicatedFunc,x,10); # Taylor series about x=0 up to term in x^9
>poly:=convert(TS, polynom); # What does this do?
>coeff(poly,x,7);
```

The next few lines illustrate an alternative way of defining and working with functions. Note carefully the difference between what we're doing here and what we did above.

```
>restart;
>f:=x -> (x+1)^6 + (x-1)^6;
```

```
>f(0);
>f(5);
>f(t);
>g:=D(f);
>g(t);
>int(g(x),x);
>int(g(x),x=-1..0);
```

The second approach is generally better for all but the simplest purposes. When you write `f := x -> x^2` you're really defining a *function*, and Maple will treat it as such; on the other hand, when you write `f := x^2`, Maple will just insert `x^2` in place of every `f` that it sees. The following is a simple example of what can go wrong with this approach if you're not careful:

```
>f:=x^2+3*x-2;
> # you type in lots of stuff, and after a while go...
>x:=3;
>f; # Oops
```

The input to a function doesn't have to be a number. For example, it could be another function: in the following, `F` is a function which returns the coefficient of x^7 in the Taylor series expansion about $x = 0$ of its input (in fact there's a Maple function `coeftayl` which does this).

```
>restart;
>F:=f->coeff(convert(series(f(x),x,8),polynom),x,7);
>F(sin);
>ComplicatedFunc:= x->sin(x^2)*exp(-3*cos(x^3)+log(x^2+1))/(1+exp(x));
>F(ComplicatedFunc);
```

Here are some examples with functions of two or more variables:

```
>restart;
>f:=(x+1)^3+3*x^2*(x+2*y)^3+y^3;
>expand(f);
>collect(f,x);
>collect(f,y);
>eval(f,[x=1,y=-1]);
>diff(f,x,y); # d^2f/dxdy
>diff(f,x,x,y); # d^3f/dxdxdy

>g:=(x,y,z)->x*exp(y)*cos(z)+sin(z)/exp(y);
>g(2,1,Pi);
>diff(g(x,y,z),y,z,z);
>D[2,3,3](g);

>H:=(f,n)->coeff(convert(series(f(x),x,n+1),polynom),x,n);
>H(exp,4);
```

2.4 Features of the Maple worksheet.

a) Try *right*-clicking on some of the blue output lines in your worksheet. Experiment with the options in the pop-up menu, and notice how these options differ for different types of output (e.g. numbers, functions). As you work through the exercise sheet, get into the habit of right-clicking new types of output that you come across (e.g. matrices, lists, sets) to see what options are available.

b) A command in the Maple worksheet is *executed* when you put your cursor on it and press `ENTER`. It is *not* executed simply by virtue of being in your worksheet. As a result, commands in the worksheet are executed in the order in which you've pressed `ENTER` on them, not necessarily the order in which they appear. For example, suppose that you spend several hours working on your favourite polynomial, `f:=x^3-x^2+x-1;`. You save your worksheet and go to bed, starting on it again early next morning. You open the worksheet, go to the bottom, and type `diff(f,x);`. Maple gives the answer `0`: this is because you haven't executed the commands in the worksheet since you opened it. You can either go through pressing `ENTER` on each command, or choose `Execute->Worksheet` from the `Edit` menu.
A useful shortcut in Maple is to type `%` to mean "the result of the last command executed". Thus, for example, having typed `int(1/x^2,x);` you can type `eval(%,x=2);` to evaluate the answer at $x = 2$. Note, though, that it does *not* mean "the result of the command immediately above me in the worksheet". Using `%` carelessly can lead to errors which are quite difficult to track down. It's almost always better to assign the results of your commands to names, which can then be used later.

c) If you want to insert some command in the middle of your worksheet, you can type `CTRL-J` or `CTRL-K`, which put a new `>` prompt just after or just before the line your cursor is on.

d) If you type `CTRL-T`, then Maple switches to *text* mode, in which you can type headings, explanatory comments, and so on. When you're in text mode, a tool bar appears with drop-down boxes to choose fonts etc. Type `CTRL-M` to switch back to *maths* mode. You could experiment with some of the other commands in the `Insert` menu too.

e) The black "square brackets" which appear on the left hand side of your worksheet denote *execution groups*. When you press `ENTER` while your cursor is anywhere in an execution group, the whole group is executed.

To put several commands into one execution group, press `SHIFT-ENTER` rather than just `ENTER` between the commands. As a simple example, try entering the following, with `SHIFT-ENTER` between each pair of lines:

```
>x:=7:
 y:=3:
 x*y;
```

Notice how the "square bracket" encloses all three lines. If you now go back and change the second line to `y:=12:` and press `ENTER`, you can see that the whole

execution group is executed. This can be a very useful technique.

You can break up an execution group by putting the cursor at the beginning of the line where you want to split it and hitting `F3`; and you can join several existing commands into a single execution group by selecting them all and hitting `F4`.

2.5 Solving equations and plotting

```
>restart;
>f:=x^5+3*x^4-10*x^2-16*x-8;
>solve(f=0,x);
>solve(f>0,x);
>plot(f,x=-3..2.5); # explains the output of the previous line

>g:=x^3+4*x^2-2*x-1;
>solve(g=0,x); # Maple can solve, complicated answer
>fsolve(g=0,x);

>h:=x^4-3*x^3+1;
>solve(h=0,x); # Maple can't solve
>fsolve(h=0,x);
>plot(h,x=-infinity..infinity); # main features of graph

>solve(sin(x)=2*cos(2*x),x);
>evalf(%);
>plot([sin(x),2*cos(2*x)],x=-Pi..Pi); # compare with output of previous lines

>solutions:=solve({x^2+y^2=1,x*y=1/2},{x,y});
>allvalues(solutions);
>with(plots): # See section 3
>implicitplot([x^2+y^2=1,x*y=1/2],x=-2..2,y=-2..2,color=[red,blue]);
```

2.6 Exercises

Start a new Maple worksheet and save it as `yourname4`. Go to `text` mode and put your name at the top, followed by `Exercise 2.6a)` on a new line, then revert to `maths` mode. As you work through the exercises in this and later sections, start each exercise with the exercise number in this way, and also use text mode to insert any relevant comments on what you're doing. If you get stuck, remember there are some hints on the module webpage.

a) Find the smallest prime number which is bigger than 10,000,000.

b) Put `Exercise 2.6b)` at the start of a new line. Enter the function

$$f(x) = \frac{e^x \sin(2x)}{x^2 + 1}$$

using the notation `f := exp(x)*...` Find $f(1)$ to 20 decimal places. Find $\frac{df}{dx}$, and get Maple to produce the LaTeX code necessary to insert the result in a

LaTeX document. Plot both $f(x)$ and its Taylor series expansion about $x = 0$ up to the term in x^{10}, using the range $-1 \leq x \leq 1$ (put both graphs on the same axes). Now repeat this exercise using instead the notation `f := x -> exp(x)*...`

c) Look up the help on the `dsolve` command, and use it to solve the differential equation

$$\frac{d^2y}{dx^2} + y = 0 \qquad\qquad y(0) = 1,\ y(\frac{\pi}{2}) = 0.$$

Similarly, solve

$$\frac{dy}{dx} = (x-1)y + 1 \qquad\qquad y(0) = 1,$$

and plot the solution $y(x)$ in the range $-2 \leq x \leq 2$.

d) Type `?index[functions]` to get a list of all Maple functions. Browse through the list and pick out any three that look interesting to you. Look at the help information on them, and write a command using each one with a short explanation of what the command is doing. For example, you could write:

The following sets a to be the greatest integer less than or equal to 15/7.
`a := floor(15/7);`

(Try to make your examples as interesting as you can, and certainly not just copies of examples in the Maple help system.)

3 Packages

Save your file `yourname4`, and go back to the file `test` to work through the following commands.

Additional functions are contained in Maple's *packages*. To get a list of all the available packages, type `?index[packages];`. If you want help on, for example, `LinearAlgebra`, the Linear Algebra package, you can either click on the link to it from this list or type `?LinearAlgebra;`.

You can load up the package, so making its commands available, by typing

```
>with(LinearAlgebra);
```

(end with a colon instead of a semicolon if you don't want to see a list of all the new functions which are available).

3.1 Using the `LinearAlgebra` package

Try out the following lines and spend some time looking at the help for various functions in `LinearAlgebra`.

```
>with(LinearAlgebra):
>M:=<<2,1,0,1>|<1,2,0,0>|<2,1,-3,1>|<1,1,1,0>>;
>N:=Matrix([[2,1,3],[1,3,2],[2,1,1]]); # Alternative method
>Determinant(M);
>Id:=IdentityMatrix(4); # Why not I:=IdentityMatrix(4);?
>M+Id;
>M.Id;
>P:=M-t*Id;
>Determinant(P);
>CharacteristicPolynomial(M,t);
>C:=Column(Id,2);
>M.C;
>SubMatrix(M,[2..3],[2..4]);
```

Incidentally, with long command names like `CharacteristicPolynomial`, Maple's *command completion* is useful. Try typing `Char` followed by `CTRL-SPACE`, then `Pol` followed by `CTRL-SPACE`. A word of explanation is in order concerning the two different notations for multiplication: why do we write `t*Id,` but `M.Id` and `M.C`? The answer is that, in Maple, *commutative* multiplication is denoted `*`, while *non-commutative* multiplication is denoted with a dot. And you know that matrix multiplication is non-commutative, don't you?

3.2 Exercises

It's probably a good idea to type `restart;` in your worksheet `yourname4` before starting each new set of exercises. Remember that you need to load `LinearAlgebra` before starting to work with matrices.

a) Let A be the matrix $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ and B be the matrix $\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$. Use Maple to multiply A and B together, find the eigenvalues and eigenvectors of each (include some text which interprets the answers that Maple gives here), and invert each of them (what answer does Maple give when you try to invert B?). Use Maple to multiply the matrices $\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ and $\begin{pmatrix} j & k & l \\ m & n & p \\ q & r & s \end{pmatrix}$. What is the determinant of the result? Factorize your answer.

b) Load the `numtheory` package. Enter the commands `divisors(28);`, `tau(28);`, and `sigma(28);`, adding a line to explain what each command is doing.

4 Lists, Sets, Loops, and Conditionals

Starting in sheet 5, you'll be learning about *programming* in Maple. The topics treated in this section, while they may not immediately seem as relevant as those treated so far, are going to be very important.

4.1 Lists

A `list` is, well, a list: any list of any valid Maple objects. You enter a list by putting it in square brackets, separating the different elements with commas. Try the following commands, which do things such as finding the length of a list, extracting elements of lists, concatenating lists, and so on. (Remember to return to the worksheet `test` to try out these lines.)

```
>list1:=[2,4,6,8];
>list2:=[-3,-7,15];
>nops(list1);
>op(list1);   # ‘‘removes square brackets’’
>list1[2];
>list1[3];
>list1[-1];
>list1[-2];
>list1[2..4];
>list1[5];
>op(2..4,list1);
>list3:=[0,op(list1),10];
>list4:=[op(list1), op(list2)];
>list5:=[op(1..3,list4),-100,op(4..nops(list4),list4)];
>list6:=[egg,f,list3,x->x^2];
>list6[4];
>list6[4](3);
```

Two useful commands are `seq`, which you can use to generate a list following some regular pattern; and `map`, which you can use to apply a function to every element of a list.

```
>list1:=[seq(i,i=0..20)];  # or you could write list1:=[$0..20];
>list2:=[seq(sin(Pi/i),i=1..6)]; # or list2:=[sin(Pi/i) $ i=1..6];
>list3:=[seq(i^2+1,i=0..10)]; # or list3:=[i^2+1 $ i=0..10];
>map(factorial,list1);
>map(evalf,list2);
>map(x->x^2,list1);
>map(isprime,list3);
```

Following on from the last example, you can use the commands `select` and `remove` to find all elements of a list which do or don't satisfy some predicate (a *predicate* is a function, like `isprime`, which returns one of the *boolean* values `true` or `false`).

```
>list1:=[$0..100]:
>select(isprime,list1);
```

```
>list2:=remove(isprime,list1);
>with(numtheory):
>MyPredicate := n-> tau(n)=8; # returns true if n has exactly 8 divisors
>select(MyPredicate,list1);
```

4.2 Sets

A set is like a list, except that

- It can only contain one copy of each element, and
- Maple keeps its elements in an order of its choosing, rather than the order you enter them.

```
>set1:={2,4,6,5,4,3,2,1};
>set2:={seq(i^2,i=1..3)};
>{op(set1),op(set2)};
>[op(set1),op(set2)];
>set1 intersect set2;
>member(5,set1);
>member(5,set2);
```

4.3 Loops

In the following examples, the code for each example is spread over several lines (to make its structure clearer). Each example begins with the `>` prompt. When entering such examples, press `SHIFT-ENTER` rather than just `ENTER` at the end of each line but the last, to avoid Maple complaining about "`premature end of input`". (The indentation in these examples makes no difference to Maple, but it makes them easier for a human to read.)

```
>for i from 3 to 9 do
    print(i);
  end do;

>for i from 21 to 32 by 2 do
     print (i); print(factorial(i));
  end do;

>for i from 2 to 5 do
    for j from 2 to 5 do
       print(i,j,i*j);
    end do;
 end do;

>for i from 2 to 5 do
    for j from i to 5 do
       print(i,j,i*j);
    end do;
 end do;
```

```
>list1:=[2,3,5,7,11];
>for i in list1 do
    print(2*i);
 end do;

>ProductSoFar:=1:
 SumSoFar:=0:
 for i from 1 to 10 do
    ProductSoFar:=ProductSoFar*i;
    SumSoFar:=SumSoFar+i;
 end do:                    # Try replacing this : with ;
 ProductSoFar;
 SumSoFar;

>i:=1:
 while i<1000 do
    print(i);
    i:=2*i;
 end do:
```

4.4 Conditionals (if ... then ... else)

Try the following. Note that `elif` means "else if", that `<=` means $\leq$, and that `<>` means $\neq$.

```
>for i from 0 to 10 do
    print(i);
    if i>5 then
       print("i is greater than 5");
    end if;
 end do;
>for i from 0 to 10 do
    print(i);
    if i mod 2 = 0 then
       print("i is even");
    else
       print("i is odd");
    end if;
 end do;
```

(Similarly, wrap each of the following examples in `for i from 0 to 10 do print(i); ... end do;`)

```
    if i<>3 and isprime(i) then
       print("i is prime and isn't 3");
    end if;

    if i<4 or not isprime(i) then
       print("i is small or not prime");
    end if;

    if i<4 then
```

```
   print("i is small");
elif i<=7 then
   print("i is medium-sized");
else
   print("i is big");
end if;
```

4.5 Exercises

Some of these exercises are quite hard. Remember the hints on the module webpage.

a) Make two lists, one containing the values $i^2 + 2i + 3$ for $1 \leq i \leq 10$, and the other containing the (floating point) values $\frac{i}{i^2+1}$ for $3 \leq i \leq 8$. Use `op` to obtain a new list by putting the values in the second list at the end of the first, adding the value -13 at the front.

b) Set `list1` be any list of integers that you choose, provided that it contains both positive and negative values. Write a `for` statement which computes the sum of the elements of `list1`. Write another `for` statement (involving `if`) which computes the sum of the *positive* elements of `list1`.

c) Set `list2` to be any list of the same length as `list1`. Write a `for` statement which creates a new list `list3` which interleaves the elements of `list1` and `list2`: that is,
`[list1[1], list2[1], list1[2], list2[2], ...]`

d) Write a line of text in your file explaining what the following does:

```
>old:=2:
 new:=3;
 while new-old<100 do
    old:=new;
    new:=nextprime(new);
 end do:
 old;
 new;
```

e) A *perfect* number is one which is the sum of its proper divisors. For example, 6 is perfect since it has divisors 1, 2, 3 (and 6), and $1 + 2 + 3 = 6$. The next perfect number is $28 = 1 + 2 + 4 + 7 + 14$. Write a predicate `IsPerfect`, and use it to find the two other perfect numbers which are less than 10000.

5 Submission

Send me an email (`t.hall@liv.ac.uk`), attaching your Maple worksheet `yourname4`. Please tidy it up before sending it, and make sure that is only contains solutions to the exercises, not any other test commands you may have typed.

A Lists and Arrays

If you use lists a lot, you'll sooner or later come across the following error message:

```
Error, assigning to a long list, please use arrays.
```

This happens if you have a list with more than 100 elements, and you try to change one of them:

```
>list1:=[seq(i,i=1..101)]:
>list1[1]:=2; # Maple complains about this.
```

The reason for this is that, because of the way Maple stores lists in the computer memory, every time you change an element the whole list needs to be copied within memory. For a long list this can obviously be very time-consuming.

The solution, as the error message suggests, is to use an `Array` instead of a `list`. Changing elements of an `Array` is efficient: in addition, an `Array` can be multi-dimensional, and its indices can cover any range rather than always starting at 1. The disadvantage is that you have to specify the size (or at least the maximum size) of an `Array` when you create it, and afterwards you can't make it grow or shrink.

My advice, at least initially, is to use `list`s when you can, and `Array`s when you have to — probably for one of the following reasons:

- You want to modify elements of a long `list`.
- You need more than one dimension.
- Your program runs too slowly using `list`s.

It's probably best to ignore the rest of this appendix, but to remember that it's here if you start to have problems with `list`s.

To make things even more complicated, there is also something called an `array`. An `array` is the old-fashioned version of an `Array`, which you're advised not to use any more...

Before looking at some examples of how to use `Array`s, note that you can convert between `list`s and `Array`s. Thus if you really want to do what Maple wouldn't let you do above, you can type

```
>list1:=[seq(i,i=1..101)]:
>Array1:=Array(list1); # convert the list to an Array
>Array1[1]:=2; # now we're allowed to modify an element
>list1:=convert(Array1,list); # convert the array back to a list
```

Of course, this is very wicked: you're deliberately doing what Maple advised you not to. Don't expect it to be efficient. But if your program runs as fast as you need it to, then what do you care?

Here are some examples of the use of `Array`s.

```
>A:=Array(1..20);   # A is an array with room for 20 elements
>for i from 1 to 10 do A[i]:=i^2; end do: # We just assign to the first 10
>ArrayElems(A); # Here are the elements of A. "(5)=25" means A[5] is 25
>A[15]:=-11;
>ArrayElems(A); # notice that A[11] to A[14] still don't have any value.
>for i from 11 to 20 do A[i]:=-i; end do: # assign to everything else
```

```
>ArrayElems(A);
>A[21]:=15; # Maple doesn't like this.

>B:=Array([seq(i^2,i=1..10)]); # Makes list and converts to array

>C:=Array(-3..1); # C has room for 5 elements, labelled C[-3] through to C[1]
>for i from -3 to 1 do C[i]:=2^i; end do:
>ArrayElems(C);

>E:=Array(1..3,1..4); # A two-dimensional array.
>for i from 1 to 3 do for j from 1 to 4 do E[i,j]:=x^i*y^j; end do; end do:
>ArrayElems(E);
>print(E);

>F:=Array(-1..1,-1..1,-1..1); # A three-dimensional array.
>for i from -1 to 1 do for j from -1 to 1 do for k from -1 to 1 do
    F[i,j,k] := x^i*y^j*z^k;
 end do; end do; end do:
>ArrayElems(F);
>print(F);  # Try double clicking on the output.
>map(x->x^3,F); # Double click on the output.
```