WEEK 1

Introduction

1. Getting started

These instructions will assume that you are comfortable with using Python in Jupyter notebooks in VS Code, and that you have installed the packages `sympy`, `numpy`, `scipy` and `plotly` using `pip` or `conda`. You will also need to import these packages and set some options. The most convenient way to do this is as follows. Distributed together with this file is a directory called `header` containing a single file `__init__.py`. You should copy this directory to be a subdirectory of the directory where you keep your notebooks for this course, then you should include the line

```
from header import *
```

as the top line of each of your notebooks.

Numbers in square brackets refer to the SymPy syntax primer which you can find on the course home page.

2. Python as a calculator

Exercise 2.1. In a code cell, enter

```
2+2
```

then press SHIFT-ENTER. Assuming that everything is set up correctly, you should see 4.

Next, in a new code cell, try the following calculation[1]:

```
(3 + 7 + 10) * (1000 - 8) / (900 + 90 + 2) - 17
```

Note that we use `*` for multiplication [5.1].

In both of the examples above, we had only a single calculation in each code cell. We can combine two or more calculations in the same code cell, but then only the last one will be printed. If we want to print all of the results, then we need to add explicit print statements, like

```
print(2+2)
print((3 + 7 + 10) * (1000 - 8) / (900 + 90 + 2) - 17)
```

Note that in the second calculation, the answer is reported as 3.0 rather than 3. This means that Python is treating it as a real number whose fractional part happens to be zero, which has a different internal representation from the integer 3. If we write `n // m` instead of `n / m` then we will always get a genuine integer, but this will just discard the remainder, if there is one.

Exercise 2.2. Try the following calculations. In each case the answer will have some kind of pattern to it; see if you can explain it.

(a) $6^{20}\ 15^{20}/9^{20}$ (Enter this as `6**20*15**20//9**20;`)
(b) $(10^{10}-1)/99$
(c) $(10^{10}-10-9^2)/9^2$ (The explanation here is harder, and should be considered a challenge for enthusiasts only.)
(d) $(10^9+1)(10^{10}-10-9^2)/9^2$ (You should be able to explain how this relates to (c), even if you cannot explain (c) itself.)

Again you could write either `n // m` or `n / m` when dividing an integer n by an integer m. You can use the second form to convince yourself that the answer is really an integer so that there is no remainder, then switch to the first form to get rid of the superfluous .0.

Python will usually convert fractions like 1/7 to approximate floating point numbers like 0.142857. If you want to work with exact fractions, you need to enter something like `sp.Rational(1, 7)` instead

[1] From the Beaver's Lesson: `http://tinyurl.com/2dv793`, verses 16–17

of 1/7. If you have an exact fraction x then you can enter `float(x)` to convert it to a floating point number.

EXERCISE 2.3. Calculate the following as exact fractions, then as numerical approximations. What do you see?

$$\text{(a)}\quad 3+\frac{1}{7} \qquad \text{(b)}\quad 3+\cfrac{1}{7+\cfrac{1}{16}} \qquad \text{(c)}\quad 3+\cfrac{1}{7+\cfrac{1}{15+\cfrac{1}{1+\cfrac{1}{293}}}}$$

You need to be very careful with brackets when entering these expressions. For example, the last one should be

```
3 + 1/(7 + 1/(15 + 1/(1 + sp.Rational(1, 293))))
```

Note here that we only need to use `sp.Rational()` for the innermost fraction of 1/139. As soon as we have one exact fraction, Python knows that it should work consistently with exact fractions from there onwards.

The full story about the precision of floating point arithmetic is complicated, but it is roughly true to say that calculations are usually done to 16 significant figures. If we need more accuracy than that, we can use the function `sp.N(x,k)`, which will calculate x to k significant figures.

EXERCISE 2.4. Enter `x = sp.N(sp.exp(1),4997)` to calculate the number $e = \exp(1) \approx 2.718$ to 4997 digits. If we just do `print(x)` then we will see all 4997 digits on the same line, which is inconvenient. It is better to do the following, to print the result on multiple lines with 100 digits per line:

```
s = str(sp.N(exp(1),4997))
m = 100
for i in range(0, len(s), m):
    print(s[i:i+m])
```

You will see that the last five digits are 66666. This is the first place in the decimal expansion of e where a digit is repeated five times.

EXERCISE 2.5. It is a remarkable fact that $e^{\pi\sqrt{163}}$ is extremely close to being an integer (for a reason involving some very advanced mathematics). Let us check this.

(x) Enter `x = sp.N(sp.exp(sp.pi * sp.sqrt(163)),40)` (which sets x equal to $e^{\pi\sqrt{163}}$) and then `print(x)`. You should see that there is a long string of nines after the decimal point, indicating that x is close to being an integer.

(c) Enter `z = sp.ceiling(x)`, to set z equal to the integer closest to x. Then print $z - x$ to see that the difference between x and z has absolute value less than 10^{-12}.

(d) Now x and z have numerical values, which will be inconvenient later if we want to use them as abstract symbols. We therefore enter `del x, z` to remove the values of x and z.

3. Symbolic algebra

EXERCISE 3.1. Enter

```
x, y = sp.symbols('x y')
A = (x**2 - 4*y**2)*(x**3 - x*y**2)
```

This declares x and y to be abstract symbols, and defines $A = (x^2 - 4y^2)(x^3 - xy^2)$. (Note that in Python x^n is expressed as `x ** n`, not as `x^n`, and we also need a single star for multiplication.)

Now try the expressions `A.simplify()`, `A.factor()`, `A.expand()` and `sp.polys.polyfuncs.horner(A)`. This gives four different ways to rewrite the expression A, all of which are potentially useful for different applications. Which of them do you prefer?

There are some things to say about how to display symbolic expressions like these. Generally you should enter `display(A)`, and then you will see $(x^2 - 4y^2)(x^3 - xy^2)$. If you just evaluate A as the last expression in a cell, then it will also have the same effect. However, if you enter `print(A)`, you will instead see `(x**2 - 4*y**2)*(x**3 - x*y**2)`, which is probably not what you want. You might also want to construct a sentence containing some mathematical expressions, which you can do like this:

```
display(Latex(
 "The factored form of $" + sp.latex(A) + "$ is $" +
 sp.latex(sp.factor(A)) + "$"
))
```

EXERCISE 3.2. Now enter the expression

$$A = \frac{2x}{x^2-1} + \frac{1}{x+x^2} + \frac{1}{x-x^2},$$

and then enter `A.simplify()` to simplify it. See if you can get the same answer by hand.

4. Plotting

EXERCISE 4.1. We first want to plot the graph of $y = \sin(x)$ from $x = -4$ to $x = 4$. For this we enter

```
x = np.linspace(-4, 4, 201)
y = np.sin(x)
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=y))
fig.show()
```

The first line sets x to be an array $[-4, -3.96, -3.92, \dots]$ consisting of 201 equally spaced points between -2 and 2 (inclusive). The second line sets y to be the array $[\sin(-4), \sin(-3.96), \sin(-3.92), \dots]$ obtained by taking the sine of each entry in x. The third line creates an object called `fig` representing a picture, to which we can add curves, shapes text and so on. In this case we just add a single curve passing through the points with coordinates (x_i, y_i); this is done by the line `fig.add_trace(go.Scatter(x=x, y=y))`. Finally, the line `fig.show()` to show the picture that we have constructed. This is not always necessary, if the last line in a cell is related to plotting then Python will often decide for itself that the resulting plot should be shown, but it does not hurt to make it explicit.

Now go back and add the lines

```
y1 = x - x**3/6 + x**5/120 - x**7/5040
fig.add_trace(go.Scatter(x=x, y=y1))
```

before `fig.show()`. This will plot the function

$$y = x - x^3/6 + x^5/120 - x^7/5040$$

along with $y = \sin(x)$, with the two curves in different colours. What do you notice?

5. Solving

EXERCISE 5.1. Ask Python to solve the equations

$$x^2 + y^2 = 1$$
$$(x-1)^2 + (y-1)^2 = 1,$$

like this:

```
x, y = sp.symbols('x y')
sp.solve([sp.Eq(x^2+y^2,1),sp.Eq((x-1)^2+(y-1)^2,1)])
```

or

```
x, y = sp.symbols('x y')
sp.solve([x^2+y^2-1,(x-1)^2+(y-1)^2-1])
```

Can you find the solution by hand? Can you draw (by hand) the curves $x^2+y^2 = 1$ and $(x-1)^2+(y-1)^2 = 1$, and thus obtain the solution graphically?

6. Calculus

EXERCISE 6.1. Read [12.1,**??**]. Then ask Python to differentiate the function $\ln(\ln(\ln(x)))$ with respect to x (making sure that you first declare x to be an abstract symbol). Now differentiate the functions $(3x+4)/(2x+3)$ and $(1+x^2+x^4/2)e^{-x^2}$, and simplify your answers.

EXERCISE 6.2. Ask Python to evaluate the integral $\int_0^1 \sqrt{1-x^2}\,dx$, as explained in [13.3]

WEEK 2

Solving equations

Sympy has powerful facilities for solving equations, but they are not perfect. Often one has to rearrange an equation in some way before sympy will succeed in solving it. It is quite common for sympy to give a complicated solution, when a slightly different approach might give a simpler one. Thus, solving equations is more of an art than a science.

Numbers in square brackets refer to the "Python reference" notes, which were distributed in the first lecture.

1. Algebraic equations

Exercise 1.1. Enter

```
f = lambda x: 2*x**4-2222*x**3+224220*x**2-2222000*x+2000000
```

to define the function

$$f(x) = 2x^4 - 2222x^3 + 224220x^2 - 2222000x + 2000000.$$

(See [10.1 – **??**] for more about the syntax used here for defining functions.) Then solve $f(x) = 0$ (as explained in [8.1]) to find the roots. Use the answer to factorize $f(x)$, and then enter `display(sp.factor(f(x)));` to check your answer.

Exercise 1.2. Declare x to be an abstract symbol, then define $y = x^4 - x^3 - x^2 - x/8 + 1/64$ and $z = 16x^3 - 24x^2 - 6x + 2$. (Here we do not use the lambda notation, because we do not have a (x) on the left hand side. We also need to enter $1/64$ as `sp.Rational(1,64)` to ensure that it is treated as an exact fraction.) Ask sympy to find the values of x where $y = 0$; you should get four solutions. It is best to use a command like `sols = sp.solve(y)` so that you can reuse the solutions later; now `sols[0]` will refer to the first solution, `sols[1]` will refer to the second, and so on. While sympy's solutions are correct, they are not given in the simplest possible form. It turns out that the solutions are given by $x = (1 \pm \sqrt{2})(1 \pm \sqrt{3})/4$. (There are four possible ways to choose the two $\pm$ signs, giving four different solutions.) Enter these four numbers and ask sympy to check that they are the same as the solutions that it found earlier. (Remember that the best way to check whether $A = B$ is to simplify $A - B$ and check whether you get zero.)

Next, read [6.1 – 6.2]. Use the `subs()` method to find the value of z at the four points where y is zero. (There are some interesting phenomena going on in this calculation, that most naturally explained using Galois Theory.)

Exercise 1.3. As general background, you should be aware that the last two exercises were chosen carefully to work out nicely. For a typical polynomial there may be no formula for the roots, and even if there is, it may be very ugly. Moreover, even if there is a formula and the roots are all real numbers, the formula may still involve complex numbers as an intermediate stage. For an example of this, ask for the roots of $x^3 - 3x + 1$. Notice that the answer involves i (the square root of minus one) in several places.

In a case like this, it is more useful to find numerical approximations to the roots instead of a massive exact formula. Read [8.7], and ask Python to find an approximate solution to the equation $y = 0$, where $y = x^3 - 3x + 1$. It is also helpful to plot the graph (say for $-2 \leq x \leq 2$) and see where the roots lie: read [11.1] for help with this. From either of these approaches, we see that there is precisely one negative root for this equation. Use [12.1] and [6.1] to find the value of dy/dx at this negative root.

Exercise 1.4. Enter the definition

$$g(x) = \frac{b^2 - c^2 + (1 + c^2)x}{1 - c^2 + c^2 x}.$$

We will study the fixed points of g, or in other words the values where $g(x) = x$.

(a) Ask Python to solve the equation $g(x) = x$ for x. (Remember to use the syntax in [10.1] when entering the definition of g.) You should get two solutions.
(b) For some special value(s) of b and/or c, Python's answer does not make sense. Go back to the definition of g and the equation $g(x) = x$, and work out by hand what happens in those special cases.
(c) For which value(s) of b and/or c are Python's two solutions actually the same?
(d) Give a self-contained summary of your conclusions, that could be read and understood by someone who had not attempted the question.

2. Approximate solutions

EXERCISE 2.1. Define $f(x) = \sin(\pi x + e^{-x})$ (remembering that e^{-x} is `exp(-x)` and π is `Pi`). If you ask Python to solve this, it will give an answer in terms of an obscure function called `LambertW`. [1] We will ignore this for the moment, and look instead for a numerical approximation to the roots, concentrating on the case where $x \geq 0$.

(a) First plot the graph of $f(x)$, say from $x = 0$ to $x = 10$. Roughly where are the roots? Can you explain why they are where they are?
(b) Enter `sc.optimize.root_scalar(f1, x0=0).root`. This finds a root at about $x = -0.55$, ignoring all the roots with $x \geq 0$ that we saw in the graph. To find a root near $x = 2$ instead, enter `sc.optimize.root_scalar(f1, x0=2).root`. To find roots near $x = 1$, $x = 2$, $x = 3$ and so on, up to $x = 10$, enter this:
```
roots = np.array([sc.optimize.root_scalar(f1, x0=n).root for n in range(1,11)])
```
(c) Now define $r(n) = n - e^{-n}/\pi - e^{-2n}/\pi^2$ (using syntax as in [10.2]). In (a) and (b) we saw that for every integer n, there is a root that is close to $x = n$. I claim that this root is even closer to $x = r(n)$. To see this, enter the definition of r and then
```
roots_approx = np.array([r(n) for n in range(1,11)])
```
Compare this with your final answer in (b).

3. Infinite families of solutions

EXERCISE 3.1. Consider the equation $\tan(\pi x)^2 = 3$.

(a) To find the solutions graphically, plot the function $\tan(\pi x)^2 - 3$ for a reasonable range of values of x. (Because $\tan(\pi x)$ blows up to infinity for certain values of x, it is necessary to cut down the vertical range to get a meaningful picture [**??**]. You should see that there are many roots in the picture, repeating in a regular way, so there are actually infinitely many roots if we allow x to run from $-\infty$ to $+\infty$.
(b) By inspecting the graph carefully, you should be able to guess the exact values of the roots.
(c) Use `sp.solve()` to solve the equation $\tan(\pi x)^2 = 3$ for x. Python will report only two solutions, although we have seen that there are really infinitely many.
(d) You should have seen in (b) that the solutions are all of the form $x = n + 1/3$ or $x = m - 1/3$, where n and m are integers. To persuade sympy to find this answer, we need to use `sp.solveset()` instead of `sp.solve()`, and also include the optional argument `domain.sp.S.Reals` to indicate that we are looking for real number solutions. This will give the correct answer, but in a form that is not fully simplified or optimally efficient.

4. Linear equations

EXERCISE 4.1. Consider the equations $x/2 + y/3 + z/4 = 1$, $x/3 + y/4 + z/5 = 2$, $x/4 + y/5 + z/6 = 3$.
To solve these, we enter

```
x, y, z = sp.symbols('x y z')
eqns = [
 sp.Eq(x/2 + y/3 + z/4, 1),
 sp.Eq(x/3 + y/4 + z/5, 2),
 sp.Eq(x/4 + y/5 + z/6, 3)
]
```

[1] If you are curious, you can enter `?LambertW` or visit `http://mathworld.wolfram.com/LambertW-Function.html`.

```
sol = sp.solve(eqns)
```

It would also work to do all this in one step:

```
sp.solve({x/2+y/3+z/4-1,x/3+y/4+z/5-2,x/4+y/5+z/6-3})
```

However, this can make things cramped and hard to organise, especially if our equations are just one piece of a more complex problem. Now find the value of $x^2 + y^2 + z^2$ at the point where the above equations are satisfied.

EXERCISE 4.2. Solve the following systems of equations by hand. You should find that

- one system has no solutions
- one system has a single, fully-determined solution
- one system has a solution in which one of the variables is free to take any value; this means that there are infinitely many different solutions.

(a) $p + q + r = 0 \qquad p + 2q + 3r = 1$

(b) $u + v = 1001, \qquad u + 2v = 1002, \qquad u + 3v = 1006$

(c) $x + y + z = 2, \qquad x + 2y + 3z = 2, \qquad x + 4y + 9z = 2$

When you have found the solutions by hand, find them again using Python [8.1,8.4].

WEEK 3

Plotting 1

Before starting each exercise, you should restart Python, either by entering the `restart;` command, or by clicking the button with the circulating arrow at the right hand end of the toolbar.

Numbers in square brackets refer to the "Python reference" notes, which were distributed in the first lecture.

There are some hints at the end.

EXERCISE 0.1. Plot each of the following functions [11.1] for a suitable range of values of x. Experiment to find a range that displays the interesting features. Write two or three lines (for each function) describing those features.

(a) $2e^{-t}\sin(30t)$
(b) $2\sin(20t) + 3\sin(21t)$
(c) $\sin(x) + \sin(3x)/3 + \sin(5x)/5 + \sin(7x)/7$
(d) $\cos(\pi x^2)$

EXERCISE 0.2. Consider the function

$$f(x) = (x^3 - x)(x^2 - 4/9)(x^2 - 1/9).$$

Enter this definition, using the syntax explained in [10.1]. Plot the graph for various ranges of x, and describe the main features that you see. Compare $f(x)$ with the functions x, x^2, x^3 and so on, by plotting them together [11.5]. Which of these matches $f(x)$ most closely for large x? Can you explain why?

EXERCISE 0.3. Consider the function

$$g(x) = (2x + 3)/(3x - 4).$$

(a) Plot the graph for various ranges of x, and describe the main features that you see. You may find it helps to restrict the vertical range [**??**] as well as the horizontal one.
(b) What is the value of x where $g(x)$ is discontinuous? You can either work this out from the formula, or read this off from the graph, or use the Python command `discont(g(x),x)`. You can also replot the graph, skipping over the discontinuity, as explained in [11.8].
(c) Plot the line $y = 5$ along with the function $g(x)$ [1]. You should see that the line crosses the graph. If you change the 5 to -1 and redo the plot, then again you see that the line $y = -1$ crosses the graph. However, there is one horizontal line (somewhere between $y = -1$ and $y = 5$) that does not cross the graph. Can you find out which line it is? You can either use plotting and trial and error, or analyse the situation algebraically.
(d) Enter `limit(g(x),x=infinity);` and `limit(g(x),x=-infinity);`. How are these related to part (c)?

EXERCISE 0.4. Consider the function

$$f(x) = \frac{1 - e^x + e^{2x} - e^{3x}}{\sin(x)}.$$

(Remember [10.1] when entering this in Python.) What is $f(0)$? The formula above gives $f(0) = (1 - 1 + 1 - 1)/0 = 0/0$, which is meaningless. However, if we plot the graph of $f(x)$ (for x from -1 to 1, say) we see that there is a perfectly definite value for $f(0)$; what is it? (See the hints at the end for a more careful discussion of the logic.)

[1] You should use the syntax in [11.5]. You should also remember that the command to plot $y = 5$ is `plot(5,...)`, not `plot(y=5,...)`.

EXERCISE 0.5. We now use Python to investigate the properties of a new function that you probably have not met before: the Bessel function $J_2(x)$. This is one of a whole family of Bessel functions, which have many applications, for example in studying the vibration of drums or the behaviour of fibre-optic cables. In Python it is called `BesselJ(2,x)`.

(a) Plot $J_2(x)$ for $-50 \leq x \leq 50$. Describe the main features.

(b) Plot $J_2(x)$ and $x^2/5$ in the same picture, from $x = -0.1$ to $x = +0.1$. Now change the 5 to something else, and repeat. Which value makes the two curves match up most closely?

(c) Plot $J_2(x)$ from $x = 10$ to 300. You should see some strange wiggles in the graph, which are in fact not really there; they appear because Python has not calculated enough points to draw an accurate picture. To fix this, we use the `numpoints` option (as in [11.9]):

```
plot(BesselJ(2,x),x=10..300,numpoints=200);
pic := %:
```

(The second line here saves the picture, so we do not have to recalculate it later [11.12].)

You should see an oscillation dying slowly away. This leads us to ask how large the oscillations are, and what is their frequency.

(d) For the size of the oscillations, try plotting $0.15x^{-1}$ alongside $J_2(x)$, as in [**??**]:

```
plots[display](pic,plot(0.15*x^(-1),x=10..300,color=blue));
```

Do the same with $0.3x^{-1/3}$ and $0.5x^{-1/2}$ and some other similar functions. (Do not retype the whole line; just edit the relevant numbers, press ENTER, and Python will redraw the graph.) If you get the numbers right, then the blue curve will just touch the tops of all the waves. (In fact it does not touch exactly, but you need to zoom in very close to see that.)

(e) For the frequency of the oscillations, try plotting $J_2(x)$ alongside $\sin(x - \pi/4)$, for various ranges of x. This should convince you that J_2 oscillates with approximately the same frequency as $\sin(x)$, at least when x is reasonably large.

(f) Can you combine (d) and (e) to find an easier function function $f(x)$ that is very close to $J_2(x)$ for large x? Plot f and J_2 together to check your answer.

EXERCISE 0.6. Put

$$g(a,x) = \frac{(x - 1 - \sin(a/4))(x - 1 - \cos(a/4))(x + 1 - \sin(a/4))(x + 1 - \cos(a/4))}{1 + x^4}.$$

(Remember [10.1,10.4] when entering this in Python.) The object of this exercise is to describe the properties of g.

(a) Plot $g(2,x)$ for x from -500 to 500. Write two or three lines describing the main features of the plot. Then plot $g(3,x)$ and $g(4,x)$; you should see that on this scale, they look almost exactly the same.

(b) Plot $g(2,x)$ and $g(4,x)$ for x from -2 to 2. In both cases, the curve dips below the x axis. Now plot $g(3,x)$ instead. It appears that the curve does not dip below the axis, but just touches it somewhere near $x = -0.3$, and again near $x = 1.7$. To investigate further, we zoom in. We plot only from $x = -0.4$ to $x = -0.2$, to focus on the region of interest. We also restrict the vertical range [**??**] to be from -0.01 to 0, so only the part of the curve below the axis will be shown. Finally, we specify `numpoints=1000` to make sure that Python plots the graph very accurately [11.9].

We see that the graph still dips slightly below the axis. However, there is a certain number a close to 3 for which the curve does not dip below the axis at all — see if you can find it. Plot the graph for various different values of a, zooming in further if necessary, and also thinking about which values of a might make something special happen in the formula for $g(a,x)$.

(c) Plot $1 - 8g(a,0)$ for a reasonable range of values of a. You should recognize the resulting graph, and so should be able to give a simple formula for $1 - 8g(a,0)$. Can you derive this formula algebraically from the definition of $g(a,x)$?

(d) Plot $g(a,0.5)$ for a from -40 to 40. What are the main features? You should see three tall peaks and three lower peaks. Consider the tall peak closest to the x-axis. Click on it and copy down the coordinates that you see in the little box at the top left of the Python window. They should be $(3, 0.9)$ approximately. To get more accurate numbers, recall that the peak occurs where the derivative of the function is zero. Here we are thinking of g as a function of a, so the slope is dg/da, or in Python terms:

```
slope := diff(g(a,0.5),a);
```

We need to solve numerically for the place where the slope is zero, close to $a = 3$:

```
fsolve(slope = 0,a=3);
```

You should recognize the answer. Now find the x-coordinates of the other two tall peaks in the same way. Can you give exact formulae for these as well? (Hint: divide them by the x-coordinate of the first peak.)

(e) Now try doing some 3-dimensional plots, for example

```
plot3d(g(a,x),a=-20..20,x=-15..15,axes=boxed,grid=[100,100]);
```

Hints

Exercise 0.4.

```
f := (x) -> (1 - exp(x) + exp(2*x) - exp(3*x))/sin(x);
plot(f(x),x=-1..1);
```

The graph crosses the y-axis at $y = -2$, suggesting that $f(0) = -2$.

If we want to be perfectly logical, we should say something a little bit different, however. This is our function f, so it is our right and privilege to define it however we want. The original formula does not make sense for $x = 0$, so $f(0)$ is not yet defined. We can, if we choose, declare that $f(0)$ is simply undefined. Alternatively, we can, if we feel perverse, declare that $f(0) = 42$. Alternatively, we can declare that $f(0) = -2$. The graph shows only that the last option is the most reasonable one, not that we are forced to take it.

Exercise 0.5.

(b) `plot({BesselJ(2,x),x^2/5},x=-0.1..0.1);`
The best match occurs with $x^2/8$ instead of $x^2/5$.

(d) The graph of the function $y = 0.8x^{-1/2}$ touches the tops of the waves (not exactly, but to a very good approximation).

(f) The function $J_2(x)$ is very close to $-0.8\sin(x + \pi/4)/\sqrt{x}$ (for large x). Enter the following to plot them together:

```
plot({BesselJ(2,x),-0.8*sin(x+Pi/4)/sqrt(x)},x=1..40);
```

Exercise 0.6. The definition of g is

```
g := (a,x) -> (x-1-sin(a/4))*(x-1-cos(a/4))*
              (x+1-sin(a/4))*(x+1-cos(a/4))/(1+x^4);
```

WEEK 4

Plotting 2

Before starting each exercise, you should restart Python, either by entering the `restart;` command, or by clicking the button with the circulating arrow at the right hand end of the toolbar. After restarting Python, enter `with(plots):` to (re)load the `plots` package. (It is safest to do this for all questions, although it is unnecessary for some. If the `display` command gives a big mess of coordinates instead of a picture, you probably forgot to enter `with(plots):`. Any warning messages produced by this command can safely be ignored.)

1. Parametric plotting

So far we have mostly drawn graphs where y is given as a function of x. We now instead draw some graphs where both x and y are given as functions of another variable, say t.

EXERCISE 1.1. (a) Plot the curve given by $x = \cos(10t)/(1+t^2)$ and $y = \sin(10t)/(1+t^2)$, as in [11.10,**??**].

(b) Plot the curve $(x, y) = (\sin(3t), \sin(2t))$ for t from 0 to 2π (this is called a Lissajous figure). Replace the 2 and 3 by larger numbers, and investigate how the picture changes.

(c) Plot the curve $(x, y) = (t - \sin(t), 1 - \cos(t))$ for t from 0 to 8π. This is called a cycloid; it is the path traced out by a point on the edge of a wheel as the wheel rolls along the ground. It looks wrong because Python draws it using different scales on the x and y axes. To fix this, click on the graph, and then click on the button marked "1:1" on the toolbar. You can click the button repeatedly to switch between the distorted and undistorted pictures. Alternatively, use the option `scaling=constrained` when you first draw the graph [11.4], like this:

```
plot([t-sin(t),1-cos(t),t=0..8*Pi],scaling=constrained);
```

(d) Plot the curve $(x, y) = (2t/(1+t^2), (1-t^2)/(1+t^2))$, again using [11.10]. What do you see? Can you explain it?

EXERCISE 1.2. Consider the curve $(x, y) = (4t^2 - 1, 8t^3 - 8t)$, which is called a nodal cubic. In this exercise it will be convenient to enter these definitions separately from the `plot` command:

```
x := 4*t^2-1;
y := 8*t^3-8*t;
```

(You must remember to remove these definitions by restarting Python [3.8] before you go on to the next exercise.)

(a) Plot the curve from $t = -1.5$ to $t = 1.5$: `plot([x,y,t=-1.5..1.5]);`

(b) Notice that the curve crosses over itself. What are the x and y coordinates of the point where this happens, and what are the corresponding values of t?

(c) The curve crosses the y-axis twice. What are the values of t and y at these two points?

(d) Now plot the function $(X-3)\sqrt{X+1}$ for X from -1 to 8. (Python will not let us use x here because x has been defined in terms of t and so is no longer a free variable [**??**]. We could use almost any other letter, but X seems a natural choice.) To compare this precisely with (a), combine the two pictures like this [**??**,**??**]:

```
display(
  plot((X-3)*sqrt(X+1),X=-1..8,color=blue),
  plot([x,y,t=-2..2],color=red),
  view=[-2..8,-15..15]
);
```

(If this gives a big mess of coordinates instead of a picture, you probably forgot to enter `with(plots):`.) What do you observe? Can you test it algebraically?

EXERCISE 1.3. **Do not enter** the following commands. Instead, read them carefully, study the reference notes, and sketch what Python would plot if you entered the commands.

(a) `plot([sin(x),cos(x),x=0..4*Pi]);`
(b) `plot([sin(x),cos(x)],x=0..4*Pi);`
(c) `plot([x,sin(x),x=0..4*Pi]);`
(d) `plot([y,sin(y),y=0..4*Pi]);`
(e) `plot([sin(y),y,y=0..4*Pi]);`
(e) `plot([sin(y),y],y=0..4*Pi);`

2. Implicit plotting

Often we want to plot a curve given by an equation like $x^2 + y^2 = 100$, where y is not explicitly a function of x. Here we might let x and y run from -11 to 11. Read [11.11] and ask Python to plot the graph.

EXERCISE 2.1. Plot the curve $y^2 = x^3 - x$, for a range of values of x and y that shows the interesting features. The `implicitplot` command will give you a rather jagged picture; you can improve it as in [**??**].

Now plot $y^2 = x^3 - x + a$ for various values of a between 0 and 1. You should see that at a certain value of a, the picture changes from being two separate curves to a single, connected curve. Find the relevant value of a approximately, by trial and error. Can you work out an exact formula?

EXERCISE 2.2. Plot the curve $x^2 + y^2 + a(\sin(2\pi x) + \cos(2\pi y)) = 100$ for various values of a, starting with $a = 1$, $a = 5$ and $a = 20$. You will need lots of points to get a good picture, so use the option `grid=[200,200]`. Describe the main features. (I have not yet worked out how to explain them; you can take that as a challenge!)

3. Plotting lists of points

EXERCISE 3.1. Read [11.13 — **??**]. Ask Python to draw the seven points at $x = 1, \ldots, 7$ with y-values 10, 40, 20, 30, 50, 10 and 20. Do this with a line joining the specified points [11.13] and then with just the points themselves [**??**]. Try changing the list of y-values in various ways and redrawing the graph.

EXERCISE 3.2.
(a) Enter `ithprime(1);`, `ithprime(2);` and so on to list the first few prime numbers.
(b) Enter `seq(ithprime(i),i=1..100);` to list the first hundred primes in one go [14.1].
(c) To plot this list, enter

```
listplot([seq(ithprime(i),i=1..100)],style=POINT);
```

Note that the `seq` command gives a list with no brackets, but the `listplot` command needs a list with square brackets, so we have put square brackets around the `seq` command.

To save this picture for future use, enter

```
pic1 := %:
```

Note that this ends with a colon, not a semicolon. If you use a semicolon, then Python will print out all the coordinates of all the points in the picture, giving several pages of useless output.

(d) It is a very interesting and important fact that the n'th prime is approximately $n(\ln(\ln(n)) + \ln(n) - 1)$. To check this, plot the function $x(\ln(\ln(x)) + \ln(x) - 1)$ for x from 1 to 100, then enter `pic2:=%:` (with a colon, again) to save the picture. Then enter `display(pic1,pic2);` to compare the two plots.
(e) Repeat for the first 1000 primes. This can be done in one step, as follows:

```
display(
 listplot([seq(ithprime(i),i=1..1000)],style=POINT,symbol=POINT),
 plot(x*(ln(ln(x)) + ln(x) - 1),x=1..1000)
);
```

(To lay this out neatly, as above, hold the SHIFT key when you press RETURN at the end of each line. Python will not attempt to carry out the command until you press RETURN without the SHIFT key.)

EXERCISE 3.3. (a) Enter `20!;` to calculate the number $20! = 1 \times 2 \times 3 \times \cdots \times 19 \times 20$. Then enter `evalf(20!);` to give the same answer in scientific notation, making it easier to see the approximate size: about 0.24×10^{19}.

(b) Now calculate $1!, 2!, \ldots, 20!$ using the `seq` command. Then plot these values, as in Exercise 3.2(c). The result is not very informative — why not?

(d) Instead, plot the values of $\ln(n!)$ for n from 1 to 20; this gives a more useful picture. Save the graph by entering `pic1:=%:`.

(e) Define $f(x) = \sqrt{2\pi}x^{x+1/2}e^{-x}$ (using [10.1]). It is an interesting and useful fact that $f(n)$ is a good approximation to $n!$, and so $\ln(f(n))$ is a good approximation to $\ln(n!)$. To check this, plot $\ln(f(x))$ for x from 1 to 20 (using the ordinary `plot()` command rather than `listplot()`). Save the result by entering `pic2:=%:`, then enter `display(pic1,pic2);` to combine the two graphs.

EXERCISE 3.4. Define a sequence of functions $f(n, c)$ by $f(0, c) = 0.5$ and $f(n+1, c) = c\,f(n, c)(1 - f(n, c))$, so for example

$$f(1, c) = c \times 0.5 \times (1 - 0.5) = 0.25c$$
$$f(2, c) = c \times (0.25c) * (1 - 0.25c) = 0.25c^2 - 0.0625c^3.$$

(This comes from a very simple model of population dynamics, where c is a parameter depending on the reproductive behaviour of a certain species, and $f_n(c)$ is the population density of that species in the n'th year.) It turns out that the long-term pattern of this sequence depends in an intricate and interesting way on c; this is the starting point of the theory of chaotic dynamics.

We can enter the definition in Python as follows:

```
f := proc(n,c)
 option remember;
 if n = 0 then
  return 0.5;
 else
  return c * f(n-1,c) * (1 - f(n-1,c));
 end if;
end;
```

Now use the `listplot` command to plot the values $f(i, 2.9)$ for i from 1 to 500. Just plot the points, not the lines joining them [**??**]. You should see that the points bounce around a bit when i is small, but they settle down so that $f(i, 2.9)$ is close to 0.655 when i is large. Now change the parameter c from 2.9 to 3.1. How does the pattern change? Try various values between 2.9 and 3.1 to see in more detail what happens and when. Then look at the range $3.4 \leq c \leq 3.5$, then $3.52 \leq c \leq 3.56$, then $3.62 \leq c \leq 3.63$.

WEEK 5

Differentiation 1

EXERCISE 0.1. On the course home page (`https://neilstrickland.github.io/maths_with_maple/`), click on the link marked `defineq.mw`. This should open up a worksheet containing a single line starting `q:=sscanf("f*6...` Click on this line and press ENTER. This will set up a "mystery function" called $q(x)$, whose properties we will investigate. [1]

(a) Plot $q(x)$. Experiment to find a range of x values that shows all the main features of the graph.
(b) For which values of x do we have $q'(x) < 0$? (You can answer this by just looking at the graph.) How about $q'(x) = 0$ or $q'(x) > 0$?
(c) Enter `Q:=(x,h)->(q(x+h)-q(x))/h;`, so that $q'(x)$ is the limit of $Q(x,h)$ as h tends to zero [10.4]. Put `e:=exp(1);` for convenience [7.3]. Tell Python to do all numerical calculations to 30 digits [**??**].
(d) Enter `Q(e^2,0.1)` to get an approximate value for $q'(e^2)$. Replace 0.1 by a smaller number to get a better approximation. What do you think is the exact value of $q'(e^2)$?
(e) Work out $q'(e^{-2})$ and $q'(e^4)$ in the same way. Guess the formula for $q'(e^t)$.
(f) Suppose that $x > 0$. What is the number t such that $x = e^t$? Deduce a formula for $q'(x)$ (valid for $x > 0$).

EXERCISE 0.2. We will call the following *Rolle's principle*: between any two roots of a function $f(x)$, there is at least one root of $f'(x)$. [2]

(a) How can you find the roots of $f'(x)$ by looking at the graph of $f(x)$?
(b) Define $f(x) = x^3 - x$ using [10.1]. Plot the graph. You should see that Rolle's principle is correct in this case. We will now analyse this more precisely using Python (although it would be easy to do it by hand).
  – Ask Python [8.2] to find the roots of $f(x)$.
  – Ask Python [12.1,**??**] to find $f'(x)$. (Note that `f'(x)` is **not** valid Python syntax.)
  – Ask Python to find the roots of $f'(x)$.
  – Plot $f(x)$ and $f'(x)$ together, and observe how the roots of $f(x)$ alternate with the roots of $f'(x)$.
(c) Repeat part (b) for the function $g(x) = \sin(x) + \sin(3x)/3$. Python will give you six roots for $g(x)$, but four of them are complex numbers, so they can be ignored. It will just give three real roots for $g'(x)$. However, both $g(x)$ and $g'(x)$ are periodic, so they really have infinitely many roots. You should work out from the graph where the rest of the roots lie. (You could ask Python to do this by setting `_EnvAllSolutions:=true;`, but it does not do a good job: the answer comes out in a complicated and confusing form.)
(d) Can you explain roughly why Rolle's principle is true? (Part (a) is relevant.)
(e) Consider $h(x) = \tan(x)$. Is Rolle's principle true in this case? How does this relate to your answer to (d)?

EXERCISE 0.3. Suppose that y is a function of x, and we write $y' = dy/dx$ and so on. The *schwartzian derivative* of y is defined to be

$$S(y) = y'''/y' - \frac{3}{2}(y''/y')^2.$$

(a) Work out $S(x^n)$ by hand. For which n is $S(x^n) = 0$? For which n is $S(x^n)$ undefined?
(b) Enter the definition into Python:

[1] For a non-mathematical challenge, you can enter `?option` and `?interface`, read the resulting help pages carefully, and work out how to print out a more comprehensible definition of $q(x)$. You will then need to read more help pages to understand the definition.

[2] As we will see, there are some exceptions. However, there is a more precisely formulated version, called *Rolle's Theorem* , for which there are no exceptions (that is what "theorem" means).

```
S := (u) -> diff(u,x,x,x)/diff(u,x) - (3/2)*(diff(u,x,x)/diff(u,x))^2;
```

Now use Python to check your answer to (a).

(c) Define $y = (ax + b)/(cx + d)$ (where a, b, c and d are constants), then ask Python to calculate and simplify y', y'', y''' and $S(y)$. You should find that $S(y) = 0$. In fact, functions of this form are the *only* ones for which $S(y) = 0$.

(d) Now define $z = (ap^x + b)/(cp^x + d)$ (where p is yet another constant) and simplify $S(z)$. You should find that $S(z)$ is a constant, not depending on x. However, some creativity is required to persuade Python to do the right simplifications. Here is one approach that works: expand out $1/S(z)$, simplify it, and then take 1 over the result.

(e) Now define

$$T(u) = (u')^{1/2}((u')^{-1/2})''.$$

Enter this definition in Python (with syntax similar to part (b)). Choose some functions w at random (eg $w = \ln(x)$, or $w = \sin(x)^2$) and calculate $S(w)$ and $T(w)$. What do you notice? Can you prove it?

EXERCISE 0.4. Consider the function $y = \cos(-2\ln(x))$ (defined for $x > 0$).

(a) Plot the graph from $x = 0$ to $x = 0.01$, and then from $x = 0$ to $x = 1000$.

(b) One of the roots of y is at $x = a_0 = e^{-\pi/4}$. Let m_0 be the gradient dy/dx at $x = a_0$. Find and simplify m_0.

(c) What is the equation of the line L_0 that passes through $(a_0, 0)$ with gradient m_0? Plot this line together with the graph of y, for x from 0 to $2a_0$.

(c) What are the other roots of y? Repeat part (b) for several other roots. What do you notice about the lines L_i? Can you prove it?

WEEK 6

Differentiation 2

1. Maxima and minima

EXERCISE 1.1. Plot the function $y = x^n e^{-x}$ for various values of n. Solve $dy/dx = 0$ and so find the maximum value of y (assuming $x > 0$).

EXERCISE 1.2. Consider the function

$$p(x) = -10x^6 + 156x^5 - 945x^4 + 2780x^3 - 4080x^2 + 2880x.$$

Plot the graph for a suitable range. Find all the critical points of $p(x)$ (where $p'(x) = 0$). Which of these are inflection points (where $p''(x) = 0$ as well)? What is the maximum value of $p(x)$, and for what value of x does it occur?

2. Implicit derivatives

EXERCISE 2.1. Put $u = x\sin(x^2 + y^2) - y\cos(x^2 + y^2)$. Plot the curve where $u = 0$, using the `implicitplot` command [11.11] . Remember that you need `with(plots):` to make this work [??]. A reasonable range is to let x and y run from -5 to 5, and you should ask Python to plot extra points [??] to get a decent picture. The curve is called a *Fermat spiral*.

Now find the slope of the curve using the `implicitdiff` command [12.3], and call the answer `slope1`. Python will give the answer in a form involving $\sin(x^2 + y^2)$ and $\cos(x^2 + y^2)$. Can you use the relation $u = 0$ to rewrite `slope1` in a form that does not involve sin or cos?

We next claim that the curve can also be described parametrically by $(x, y) = (t\cos(t^2), t\sin(t^2))$. In checking this, we will use the symbol `xt` for "x in terms of t", and thus enter

```
xt := t*cos(t^2);  yt := t*sin(t^2);
```

Use the `subs` command [6.1] to substitute `xt` for `x` and `yt` for `y` in u, and simplify the result. You should get zero, indicating that the point $(t\cos(t^2), t\sin(t^2))$ really does lie on the curve $u = 0$. (It is not very hard to do this by hand as well.) You can also enter `plot([xt,yt,t=-5..5]);` and see that you get the same picture as before.

Now calculate dx/dt and dy/dt, and so get a formula for dy/dx in terms of t. Call this one `slope2`. Use the `subs` command to rewrite `slope1` in terms of t (rather than x and y), simplify the result, and check that it is the same as `slope2`.

EXERCISE 2.2. Consider the curve with equation $u = 0$, where

$$u = (x^2 + y^2)^2 + 85(x^2 + y^2) - 500 + 18x(3y^2 - x^2).$$

It turns out that this can also be given parametrically by

$$(x, y) = (6\cos(t) + 8\cos(t)^2 - 4, 2\sin(t)(3 - 4\cos(t))).$$

Analyse this situation just as in the previous question: plot the graph in two different ways, find the slope in two different ways, and check that they are really the same.

3. Higher derivatives

EXERCISE 3.1. Enter the following definition into Python [10.2,12.2]:

$$r(n) = \frac{d^n}{dx^n}\left(\frac{x^n \ln(x)}{n!}\right).$$

Find $r(n)$ for a reasonable range of numbers n, using the `seq(...)` command [14.1]. Then experiment to find a formula for $r(n) - r(n-1)$. Finally, deduce a formula for $r(n)$, of the form

$$r(n) = \text{something} + \sum_{k=1}^{n} (\text{something}).$$

EXERCISE 3.2. Put $y = t^2 e^t$. Simplify the expression

$$z = y''' + ay'' + by' + cy,$$

where a, b and c are constants. Hence find a, b and c such that y satisfies the differential equation

$$y''' + ay'' + by' + cy = 0$$

for all t.

EXERCISE 3.3. Put $p(n) = e^{x^2} \frac{d^n}{dx^n}(e^{-x^2})$. Enter this definition in Python [10.2,12.2], and then calculate $p(n)$ for n from 1 to 10. (It is best to define $p(n)$ to be `sort(expand(exp(x^2) * ...))`, to get the answer in a convenient form. To print $p(1), \ldots, p(10)$ on separate lines, enter `seq(print(p(n)),n=1..10)` rather than just `seq(p(n),n=1..10)`.)

Write down as many things as you can about the general form of $p(n)$, distinguishing between the case where n is even and the case where n is odd. Using these, predict as much as you can about $p(12)$, and then check your predictions.

In doing this question, you should see the numbers 2, 4, 8, 16, 32, 64 and so on, which you should recognise as powers of 2. You will also see the numbers 2, 12, 120, 1680, 30240, which are much less likely to be familiar. To see what they are, enter the sequence in the search box at `http://www.research.att.com/~njas/seque` (The response is long and complicated, but the formula that you need appears in the first few lines.)

It turns out that $p(n)$ is closely related to the Hermite polynomials, which Python calls `HermiteH(n,x)`. Enter `q:=(n)->simplify(HermiteH(n,x));`, then work out $q(n)$ for various values of n, then write down the precise relationship between $p(n)$ and $q(n)$.

WEEK 7

Integration 1

Setup

Visit `https://neilstrickland.github.io/maths_with_maple/` and click with your **right** mouse button on the link marked `ratint.mw`. Then select `Save Target As..`, save the worksheet somewhere, start up Python separately, and then open the worksheet using the File menu. When the worksheet has started up, click the button on the Python toolbar marked with a triple exclamation mark (!!!). This will execute all the commands in the worksheet, which does two things:

(a) It defines a command called `ratint()`. This works in essentially the same way as the usual integration command `int()`, but it gives the answer in a form which is slightly better for the questions on this sheet.

(b) It defines a number of rational functions $g_1(x), g_2(x), \dots$ (which you should enter as `g[1](x)`, `g[2](x)` and so on). These can be used as examples later (when the question says "for various rational functions ...").

If you use `restart` in doing this problem sheet, you will lose the definition of `ratint()` and the functions $g_i(x)$. It is better to use `unassign` instead. Remember that the syntax is like `unassign('a','b')`, with single quote marks.

Questions

Your task is to answer the following questions, using a mixture of mathematical thinking and experimental calculations with Python. When calculating integrals, use `ratint(...,x);` instead of `int(...,x);`.

Exercise 0.1. When x is large, the graph of $y = \int \left(\frac{x^3+1}{x^2+1}\right)^3 dx$ looks like $y = cx^n$ for some constant c and integer n. Find c and n.

Exercise 0.2. Let a, b, c and d be nonzero constants. The graph of $y = \int \frac{ax^2+b}{cx^2+d}\,dx$ looks like a straight line for large x. What is the slope of the line?

Exercise 0.3. Are the following true or false?

(a) Integrals of rational functions sometimes involve terms like $a\ln(x^2+ux+v)$, where a, u and v are constants.

(b) Integrals of rational functions sometimes involve terms like $x\ln(x+u)$, where u is constant.

(c) Integrals of rational functions sometimes involve terms like $a\ln(x+u)^2$, where a and u are constants.

Exercise 0.4. For various different values of b and c (which may be positive, negative or zero), do the following:

(i) Plot $y = x^2 + bx + c$.

(ii) Calculate $b^2 - 4c$.

(iii) Find $\int \frac{dx}{x^2+bx+c}$, and observe whether it involves the functions arctan or ln.

What is the relationship between (i), (ii) and (iii)?

Exercise 0.5. For various rational functions $g(x)$, do the following:

(i) Find $\int g(x)\,dx$, and look for any terms like $\ln(|x-u|)$ (but ignore terms like $\ln(x^2+px+q)$).

(ii) Plot $g(x)$ together with $\int g(x)\,dx$. (You will generally need to specify the vertical range in order to get a useful picture.)

How are the numbers in (i) related to the pictures in (ii)?

EXERCISE 0.6. For various rational functions $g(x)$, do the following:

(i) Enter `d:=denom(factor(g(x)));` to find the denominator of $g(x)$ (in other words, the term on the bottom when $g(x)$ is written as a single fraction).
(ii) Plot d.
(iii) Find $\int g(x)\,dx$, and look for terms like $a/(x-u)$ or $a/(x-u)^n$.

How are the numbers u and n in (iii) related to the answers to (i) and (ii)?

Now enter `r:=randpoly(x)/randpoly(x);` to generate a random rational function, and then `ratint(r,x);` to integrate it. Now go back in the worksheet to the line `r:=randpoly(x)/randpoly(x);` and press ENTER twice, to generate and integrate a new example. Repeat this many times. You should see that you only get ln() and arctan() terms, together with multiples of x^n for some small values of n. Can you see why terms like $a/(x-u)$ hardly ever occur?

WEEK 8

Integration 2

This sheet deliberately gives very little help with Python syntax, and the same will be true for the remaining sheets. As you will see from the past papers on the course website, you will need a fairly detailed knowledge of the syntax for the exam.

EXERCISE 0.1. Enter the following definitions:

$$p_0 = \sqrt{\tfrac{1}{2}} \qquad p_1 = \sqrt{\tfrac{3}{2}}x \qquad p_2 = \sqrt{\tfrac{5}{8}}(3x^2-1) \qquad p_3 = \sqrt{\tfrac{7}{8}}(5x^3-3x)$$

(Remember that p_2 should be entered as `p[2]`, and so on.) Find $\int_{-1}^{1} p_i p_j \, dx$ for various i and j (including the case $i = j$). What pattern do you observe? There is a function p_4 of the form ax^4+bx^2+c (with $a > 0$) that makes the pattern continue. What are a, b and c? (The first step is to define $p_4 = ax^4 + bx^2 + c$, leaving a, b and c as variables. Then calculate $\int_{-1}^{1} p_2 p_4 \, dx$ and so on, giving answers that depend on a, b and c. This should give you some equations relating a, b and c, which you should solve.

Now enter the definition

$$q(n) = \frac{\sqrt{n+1/2}}{2^n\,(n!)} \frac{d^n}{dx^n}\left((x^2-1)^n\right).$$

Check that $q(n) = p_n$ for $n = 0, \ldots, 4$. (The functions p_n are called *Legendre polynomials*, and the fact that $p_n = q(n)$ is called *Rodrigues's formula*).

EXERCISE 0.2. For various positive integers n and m, calculate and factorise the integral $\int x^n \ln(x)^m \, dx$. What terms appear in the answer? How does this depend on n and m? Write down your conclusions as a self-contained statement that would make sense to someone who had not read the question.

EXERCISE 0.3. You should be aware that there are many integrals for which the answer simply cannot be written in terms of familiar functions. In same cases, Python will write the answer in terms of more obscure functions instead. For example, try the following:

$$\textbf{(a)} \int e^{-x^2} \, dx \qquad \textbf{(b)} \int \frac{dx}{\ln(x)} \qquad \textbf{(c)} \int \frac{dx}{\sqrt{1-x^2}\sqrt{1-2x^2}} \qquad \textbf{(d)} \int (x^8+1)^{-1/2} \, dx$$

In some other cases, Python will just give up. For example, try the following:

$$\textbf{(e)} \int \sin(x)\ln(\ln(x)) \, dx \qquad \textbf{(f)} \int \sin(\sin(\sin(x))) \, dx \qquad \textbf{(g)} \int \frac{dx}{\sqrt{1+x+x^{10}}}$$

What is the simplest function that you can find that makes Python give up? (There is one that you can enter with just three characters.)

EXERCISE 0.4. Consider the function

$$y = \sin(x) + \tfrac{1}{3}\sin(3x) + \tfrac{1}{5}\sin(5x) + \tfrac{1}{7}\sin(7x).$$

Plot y and $\int y \, dx$. Describe and explain the relationship between the shapes of these two graphs.

EXERCISE 0.5. Put $y = x\,e^{-x}\sin(20x)$ and $z = 20\int y \, dx$. Plot y and z for various ranges of x. Describe in detail, and explain, the relationship between the two graphs. Then plot $y^2 + z^2$. This is very close to a function with a much simpler formula; work out what it is.

EXERCISE 0.6. Find a, b and c such that $\int_0^\infty x^k(ax^2+bx+c)e^{-x} \, dx = k$ for $k = 1, 2$ and 3.

Python syntax revision

EXERCISE 0.7. Find an approximate solution to $x(\ln(\ln(x)) + \ln(x) - 1) = 541$ close to $x = 100$.

EXERCISE 0.8. What would you enter to generate the sequence $a + b^2, a + b^3, a + b^4, \ldots, a + b^{10}$?

EXERCISE 0.9. Plot the curve $x^4 + y^4 = 1$ together with the curve given parametrically by $x = \cos(t)/(2 + \cos(4t))$ and $y = \sin(t)/(2 + \cos(4t))$.

EXERCISE 0.10. Find π^{76}/e^{87} to 100 decimal places.

EXERCISE 0.11. Simplify the expression $\sin(2x)\tan(2x)\dfrac{d^2}{dx^2}\log(\tan(x))$.

WEEK 9

Integration 3

Please refer to the "Notes on Python syntax" as you do these questions.

EXERCISE 0.1. Enter the definitions $x_0 = \sin(\theta)\sin(\phi)$, $x_1 = \sin(\theta)\cos(\phi)$, and $x_2 = \cos(\theta)$. Simplify $x_0^2 + x_1^2 + x_2^2$.

EXERCISE 0.2. Enter the definition

$$u = 8 - (x-y)^2(x+y)^2(16 - 4x^2 - y^4).$$

Substitute $x = 2\cos(t)$ and $y = 2\sin(t)$ in u and simplify the result. Try both the `simplify()` command and the `combine()` command.

EXERCISE 0.3. Solve the equations

$$\begin{aligned} ax + by + cz &= 1 \\ ay + bz + cx &= 1 \\ az + bx + cy &= 1 \end{aligned}$$

to find x, y and z in terms of a, b and c.

EXERCISE 0.4. Find an approximate solution of $x^4 + \sin(x) = 10^4$ close to $x = 10$. Arrange your syntax so that Python gives you an equation, not just a number.

EXERCISE 0.5. Enter the definition $f(x) = (e^x + x^7)/(e^x - x^7)$. Find a numerical approximation to $f(5)$. Then find numerical approximations for the whole sequence $f(0), f(1), \ldots, f(50)$.

EXERCISE 0.6. Plot the curve $x^4 + y^4 = 4$, together with the curve given by $x = (2 + \sin(8t))\cos(t)$ and $y = (2 + \sin(8t))\sin(t)$ (for $0 \le t \le 2\pi$).

EXERCISE 0.7. Plot $y = \tan(\pi x)$, $y = \cot(\pi x)$, $y = -1$ and $y = 1$ on the same graph, for x from -4 to 4. Tell Python to restrict the vertical range from -2 to 2, to use the same scale on the two axes, and to skip over discontinuities.

EXERCISE 0.8. Enter the definitions

$$\begin{aligned} f(t) &= \frac{(2+\sqrt{3})t - 1}{2 + \sqrt{3} + t} \\ g(t) &= f(f(f(t))) \\ h(t) &= g(g(t)). \end{aligned}$$

Simplify $g(t)$ and $h(t)$.

EXERCISE 0.9. Find and simplify $\left(\frac{1}{y}\frac{d^8y}{dx^8}\right)^{1/8}$, where $y = \sin(10x)$.

EXERCISE 0.10. Plot the curves $y = n^{3/2}x^n(1-x)^2$ for $n = 1, \ldots, 10$ on the same graph, with x running from 0 to 1.

EXERCISE 0.11. Put $y = \dfrac{x^4(1-x)^4}{1+x^2}$. Find the indefinite integral $\int y\,dx$, the definite integral $\int_0^1 y\,dx$ and the derivative dy/dx. Plot the graph of y from $x = 0$ to $x = 1$; you should see a single hump. Find numerical approximations to the position and height of the hump.

EXERCISE 0.12. Define $a(n)$ to be the approximate numerical value of $(1 + 1/n)^n/e$. Find the sequence $a(1), a(2), \ldots, a(100)$. What is the first n for which $a(n) > 0.99$?

WEEK 10

Taylor series

1. Taylor series

EXERCISE 1.1. We will study the Taylor series at $x = 0$ of the function

$$y = \ln\left(\sqrt{\frac{1+x}{1-x}}\right).$$

This has the form $y = \sum_{k=0}^{\infty} a_k x^k$, where a_k is $1/k!$ times the value of $d^k y/dx^k$ at $x = 0$. You should start by entering the definition of y.

(a) Find [12.2] and simplify d^5y/dx^5. Then put $x = 0$ using the `subs` command [6.1], and divide by 5! to get a_5.
(b) Find a_1, a_2, a_3 and a_4 in the same way.
(c) To do this more efficiently, enter
```
a = lambda n: sp.diff(y,x,n).subs({x : 0}) / sp.factorial(n)
```
then use the `seq` command [14.1] to calculate $a(1), \ldots, a(5)$ in one go. (This syntax does not work properly when $n = 0$, but it is easy to see that $a_0 = 0$ anyway.)
(d) Now enter `add(a(k)*x^k,k=1..12);` to get the 13'th order Taylor series for y at $x = 0$. Then do the same thing more easily using the `series` command [12.5].
(e) Guess the complete Taylor series for y at $x = 0$.

EXERCISE 1.2. (a) Find the 12th order Taylor series for $\sin(x)$ at $x = 0$, using the `series` command [12.5]. Convert the answer to an ordinary polynomial as in [12.6]. Call the result s.
(b) Plot s and $\sin(x)$ together [11.5] for x from -8 to 8, with the vertical range also restricted [**??**] from -10 to 10. You should see that the two graphs are very close together for x between -5 and 5, but that they diverge very rapidly outside that range.
(c) Now define $t(n)$ to be the n'th order Taylor series for the function $\sin(x)$ at $x = 0$. (Use syntax like `t:=(n)->convert(series(...)...)`). Plot $t(n)$ and $\sin(x)$ together (with the same ranges as in (b)) for various n. As you make n bigger, the two graphs will get closer together. How big must n be for the two graphs to look identical? What then happens if you expand the horizontal range to run from -10 to 10?
(d) To plot $\sin(x)$ together with $t(2), t(6), t(10), t(14), \ldots, t(42)$, enter
```
plot([sin(x),seq(t(4*n+2),n=0..10)],x=0..20,-2..10);
```
Now modify this to get a good picture of $t(4), t(8), \ldots, t(44)$. Why do we not bother with $t(n)$ for odd n?
(e) Now define $r(n)$ to be the n'th order Taylor series for $\cos(x)$ at $x = 0$. Expand out $t(10)^2 + r(10)^2$, and call the result q. What should this be equal to, approximately? How could you check this?

EXERCISE 1.3. Find the Taylor series for $y = \frac{x(1+x)}{(1-x)^3}$ at $x = 0$, to some reasonably high order. You should be able to guess from this that

$$y = \sum_{k=1}^{\infty} (\text{something}) x^k.$$

You can ask Python to confirm this by entering
```
sum((something) * x^k,k=1..infinity);
```
You should see that Python's answer is the same as y (rearranged slightly).

EXERCISE 1.4. Enter `series(sin(x),x=Pi/2,12)` to find the 12th order Taylor series of $\sin(x)$ around $x = \pi/2$. How is this related to the 12th order Taylor series of $\cos(x)$ around $x = 0$? Why?

2. Python syntax revision

The questions in this section all cover things you have done before, but this time there are no hints about the syntax. Look at the notes on Python (`http://www.shef.ac.uk/nps/MAS100/notes/primer.pdf`) and/or the earlier lab sheets if there are things that you do not remember.

EXERCISE 2.1. Find $\cos(\ln(\pi + 20))$ to 20 decimal places.

EXERCISE 2.2. Enter the definition $f(x) = x^2 - 29/16$. Ask Python to work out $f(0)$. If it just gives you "$f(0)$" back, then you used the wrong syntax to define f; read Section 10 of the Python notes. What is $f(f(f(-1/4)))$?

EXERCISE 2.3. Using the various commands for manipulating algebraic expressions, find out the relationship between

$$a = (1+x)^5 - 3(1+x)^4 + 5(1+x)^3 - 3(1+x)^2 + 3(1+x) + 3.$$

and

$$b = (7x^2 - 6x - x^8)/(x-1)^2.$$

EXERCISE 2.4. Find the coefficient of x^6 in the expression

$$\left(\frac{(x^{12}-1)(x^2-1)}{(x^6-1)(x^4-1)}\right)^{10}.$$

(You will need to simplify the expression first.)

EXERCISE 2.5. The equations

$$\begin{aligned} x^2 + y^2 + z^2 &= 9 \\ (x-1)^2 + (y-1)^2 + (z-1)^2 &= 2 \\ 4x^2 + yz &= 2x(y+z) \end{aligned}$$

have only one solution in which x, y and z are all integers. What is that solution?

EXERCISE 2.6. Find all the (infinitely many) solutions to the equation $\sin(\theta)^2 = 3\cos(\theta)^2$.

EXERCISE 2.7. Find the solution to the equations $x^2 - y^2 = 2xy = 1$ for which x and y are real numbers and $x > 0$. Your answer should not involve the word `RootOf`.

EXERCISE 2.8. Find an approximate solution to $x = \log(x + 20)$ close to $x = 3$.

WEEK 11

Revision

1. Miscellaneous problems

EXERCISE 1.1. Plot the curve given by $x = \cos(t)/2 - \cos(2t)/4$ and $y = \sin(t)/2 - \sin(2t)/4$, together with the circle of radius $1/4$ centred at $(-1, 0)$, and two circles of radius 0.0945 centred at the points $(-0.1225, \pm 0.7449)$. Display all these in the same picture with the same scale in both directions, and with no axes. (This is a basic outline of the Mandelbrot set. Google will find you much prettier pictures.)

EXERCISE 1.2. Plot all the curves $|x|^n + |y|^n = n^{n/2}$ in the same picture for $n = 1, \ldots, 9$. Use the `seq()` command to avoid excessive typing, make sure that you use a range that includes all nine curves, and add an option to make Python draw a more accurate picture.

EXERCISE 1.3. Use the `listplot` command to plot the values $50^n/n!$ for n from 1 to 100. Mark these values as separate points, not connected by lines.

EXERCISE 1.4. There is a certain function $\zeta(z)$ defined for complex numbers z, called the Riemann zeta function. It features in the most famous open problem in all of pure mathematics, called the Riemann Hypothesis: is it true that whenever s and t are positive real numbers with $\zeta(s + it) = 0$, we have $s = 1/2$? There is a great deal of evidence for this but no proof. If true, it will have important consequences for the distribution of prime numbers (and perhaps therefore for cryptography).

(a) Enter `with(MTM)` to load the definition of $\zeta(z)$.
(b) Enter `f:=(t)->Re(zeta(1/2+I*t))`, to define $f(t)$ to be the real part of $\zeta(1/2 + it)$. Define $g(t)$ to be the imaginary part in the same way.
(c) Plot the graphs of $f(t)$ and $g(t)$ together. Note that $\zeta(1/2 + it)$ is zero when both graphs cross the horizontal axis in the same place. What is the smallest positive value of t where this happens?
(d) Now instead plot the curve given parametrically by $x = f(t)$ and $y = g(t)$ (so (x, y) corresponds to $\zeta(1/2 + it)$ in the Argand diagram). In this picture, zeros of the ζ-function correspond to the places where the curve passes through the origin. You should see that there are many of them (if you use a suitable range for t).
(e) Now change the $1/2$ to some other value and repeat steps (c) and (d). The Riemann hypothesis predicts that you should not find any zeros. Thus, the two graphs in (c) will never crosss the horizontal axis at the same time, and the curve in (d) will never pass through the origin. Check this.

EXERCISE 1.5. Put $y = \exp(-1/x)$, for $x > 0$. Investigate the behaviour of y and the derivatives $d^k y/dx^k$, by calculating formulae and plotting graphs.

EXERCISE 1.6. Use Python to expand out the expression

$$(x + y + z)^3 - 3(x + y + z)(xy + yz + zx) + 3xyz.$$

EXERCISE 1.7. Use Python to expand out the following expressions:

$$(x + 1)^3 - 3(x + 1)^2 + 3(x + 1) - 1$$
$$(x + 1)^4 - 4(x + 1)^3 + 6(x + 1)^2 - 4(x + 1) + 1$$
$$(x + 1)^5 - 5(x + 1)^4 + 10(x + 1)^3 - 10(x + 1)^2 + 5(x + 1) - 1.$$

What is the pattern? Can you see a simple explanation?

EXERCISE 1.8. Four of the following five expressions are the same; which is the odd one out? (You may assume that all variables are positive, and that no divisions by zero occur, so it is valid to use

`simplify(...,symbolic);`)

$$\frac{u^{-1}+v^{-1}}{x^{-1}+y^{-1}} \qquad \frac{uxv^{-1}+x}{uxy^{-1}+u} \qquad \frac{(x-y)(u/v-v/u)}{(u-v)(x/y-y/x)} \qquad \frac{(x^2+y^2)(u/v-v/u)}{(u^2+v^2)(x/y-y/x)} \qquad \frac{u^3xy+u^2vxy}{u^3vx+u^3vy}$$

EXERCISE 1.9. Simplify the expressions U and V, defined below:

$$X = (2+\cos(a))\cos(b) \qquad Y = (2+\cos(a))\sin(b) \qquad Z = \sin(a) \qquad U = (X^2+Y^2+Z^2-5)/4 \qquad V = X/\sqrt{X^2+Y^2}$$

EXERCISE 1.10. Put

$$\begin{aligned} x &= \frac{2u}{u^2+v^2+1} \\ y &= \frac{2v}{u^2+v^2+1} \\ z &= \frac{u^2+v^2-1}{u^2+v^2+1} \\ r &= \sqrt{x^2+y^2+z^2} \\ w &= x/(1-z). \end{aligned}$$

Simplify r and w.

EXERCISE 1.11. Which of the following are true? (You may assume that all variables are positive, and that no divisions by zero occur, so it is valid to use `simplify(...,symbolic);`)

(A) $(x^{ab}x^{bc}x^{ca})^{1/(abc)} = x^{1/a+1/b+1/c}$ (B) $\frac{(x/y)/z}{x/(y/z)} = 1$

(C) $(x+1)^6 - 6x(x+1)^4 + 9x^2(x+1)^2 = x^6+1$ (D) $\frac{2}{x^3} + \frac{2}{x^9-x^3} = \frac{1}{x^3-1} + \frac{1}{x^3+1}$

(E) If $z = \frac{px+qy}{p+q}$, then $\frac{px+qy+rz}{p+q+r} = z$.

2. Solitons

In this exercise we will investigate the behaviour of some interesting functions called solitons, which arise in the theory of water waves in a shallow channel. Put

$$\begin{aligned} q &= \sqrt{2} & p &= \log(3+2q) \\ r &= x-4t & s &= q(x-8t) \\ T &= 32\cosh(2r-p) + 16\cosh(2s-p) + 16 & B &= 4(1+q)\cosh(r)\cosh(s) + (4q-8)\exp(r+s) \\ \phi_0 &= 2\cosh(r)^{-2} & \phi_1 &= 4\cosh(s)^{-2} \\ \phi_2 &= 2\cosh(r-p)^{-2} & \phi_3 &= 4\cosh(s-p)^{-2} \\ \phi_4 &= T/B^2. \end{aligned}$$

Note that the functions ϕ_i depend x (position) and also t (time). We can therefore differentiate them with respect to x or with respect to t. In this situation where there are several variables, it is traditional to use the notation $\partial\phi_i/\partial x$ and $\partial\phi_i/\partial t$ rather than $d\phi_i/dx$ and $d\phi_i/dt$. These can be entered in Python in the usual way, as `diff(phi[i],x)` and `diff(phi[i],t)`.

Check that the functions ϕ_i all satisfy the Korteweg-de Vries equation:

$$\frac{\partial\phi_i}{\partial t} + \frac{\partial^3\phi_i}{\partial x^3} + 6\phi_i\frac{\partial\phi_i}{\partial x} = 0.$$

Python will do this quite happily for $i = 0,\dots,3$. For ϕ_4, however, you need to help by telling Python to convert all hyperbolic functions to explicit exponentials at the beginning. The easiest thing is to enter `phi[4] := convert(phi[4],exp);` before working out the derivatives.

Next, do some plots to investigate the behaviour of these functions. One approach is to use the `subs()` command to put $t = 1$ (say), and then plot the result for $-30 \le x \le 30$. You can then change t and do the same again. It is illuminating to plot several of the functions ϕ_i in the same graph.

You can also make a movie. For example:

```
with(plots):
animate(
 plot,
 [[phi[2],phi[3]+5,phi[4]+10],x=-30..30],
 t=-3..3,
```

```
 frames=100, scaling=constrained, axes=none
);
```

This makes Python work quite hard, so it may take a minute or so. When the plot appears, click on it; then some playback controls will appear near the top of the Python window, which should be fairly self-explanatory. Note that we have plotted ϕ_2, $\phi_3 + 5$ and $\phi_4 + 10$; the $+5$ and $+10$ terms just shift the graphs of ϕ_3 and ϕ_4 upwards, so we can see them more clearly.

Experiment with further movies of this type, including functions like $\phi_0 + \phi_2$ as well as the functions ϕ_i themselves. See if you can write a clear, self-contained summary of how the functions ϕ_i are related to each other.

Next, the *momentum* of ϕ_i is defined to be the integral $M_i = \int_{-\infty}^{\infty} \phi_i \, dx$. As ϕ_i depends on t as well as x, you might think that M_i depends on t. However, it turns out that the dependence on t cancels out, so M_i is a constant. Ask Python to calculate M_i for $i = 0, \ldots, 3$, and you will see that this is true.

Python is not clever enough to calculate M_4 symbolically. However, it can do it numerically if we tell it to fix a particular t (say $t = 0$) and give it a hint about the best numerical method to use:

```
M[4] := int(subs(t=0,phi[4]),x=-infinity..infinity,numeric,method=_Gquad);
```

(If we leave out `method=_Gquad` then Python will use a different method which gives the same answer but takes several minutes, whereas the method above takes only a few seconds.) You should then check that $M_4 = M_0 + M_3$.

We also define the *energy* of ϕ_i to be the integral $E_i = \int_{-\infty}^{\infty} \phi_i^2 \, dx$. Repeat the above steps for the energy, giving exact answers for $E_0, \ldots, E_3$ and an approximate answer for E_4, and check that $E_4 = E_0 + E_3$.