MATH549 Exercise Sheet 6

Deadline for submission: Monday 10th November
Please try to do as much of this sheet as you can before the tutorial
Please don't continue working on this sheet into week 7

Introduction

There are no new topics covered in this sheet, just some exercises which are designed to help you to develop your Maple programming skills.

Exercises 1 and 2 are less difficult than the others: you're given an explicit algorithm to implement as a Maple procedure. Exercises 3 to 5 are more extended investigations of a variety of topics, and are intended to mimic the sort of programming that you might end up doing for your project: if you can do all the steps of any one of these exercises, you're doing quite well.

You should certainly start by trying exercises 1 and 2. **Don't feel you have to attempt all of the other exercises** (though the more you do, the more practice you'll get). It's better if you can do one of these harder exercises completely, rather than doing bits of all three.

You should put the programs that you write in separate files. Thus any programs you write for exercise 1 should be in a file `yourname1.txt`, any for exercise 2 should be in `yourname2.txt`, and so on. In addition you should submit a single Maple worksheet `yourname6` which reads in each of the files in turn, and then has a number of commands testing and experimenting with the programs. It's a good idea to `restart` before each new exercise in this worksheet.

Make sure that your programs are well commented, and include appropriate error handling. Make sure that your Maple worksheet includes adequate commentary on what you're doing (in text mode).

As usual, you can find some hints on the module webpage.

Exercise 1: Russian multiplication

The *Russian Multiplication Algorithm*, described below, is a method for multiplying two positive integers M and N. Write a program which carries it out: thus `RussianMultiplication(M,N)` should return the number $M \times N$. (What it does isn't very exciting, then …) If you can, include a line or two of text in your worksheet explaining why the algorithm works (this is a *mathematical* question).

The algorithm

Suppose we want to multiply two positive integers M and N. We assume that $M \leq N$ (if not, swap them round). We use one other variable R (the "running total").

a) Let

$$R = \begin{cases} 0 & \text{if } M \text{ is even} \\ N & \text{if } M \text{ is odd.} \end{cases}$$

b) As long as $M > 1$, repeat the following steps in order:

   i) Halve M (ignoring any remainder).

   ii) Double N.

   iii) If M is odd then add N to R.

c) Then the answer is R.

For example, if we want to multiply 19 and 13, then the triple (M, N, R) starts as $(13, 19, 19)$ (we have $R = 19$ since M is odd), and then in successive steps becomes: $(6, 38, 19)$; $(3, 76, 95)$; and $(1, 152, 247)$. So $19 \times 13 = 247$.

Exercise 2: Cornacchia's algorithm

Let p be a prime number and d be an integer between 1 and $p - 1$. Cornacchia's algorithm solves the *Diophantine equation* $x^2 + dy^2 = p$: that is, it either gives *integers* x and y which satisfy the equation, or asserts that no solution (in integers) exists.

Write a program which carries out Cornacchia's algorithm: that is, `Cornacchia(p,d)` should return the x and y values of a solution, if one exists, and otherwise should return `FAIL`.

Find all of the solutions given by the algorithm for which p is between 2 and 97, checking that each one really is a solution. (There are 203 such solutions in total.)

The algorithm

a) Find an integer x_0 with $(p-1)/2 < x_0 < p$ such that $x_0^2 + d$ is divisible by p. If there is no such x_0, then the Diophantine equation has no solution.

b) Let $a = p$, $b = x_0$, and $l = \lfloor \sqrt{p} \rfloor$ (the greatest integer less than or equal to $\sqrt{p}$).

c) As long as $b > l$, repeat the following: set $r = a \bmod b$, set $a = b$, and set $b = r$.

d) If $(p - b^2)/d$ is not a perfect square, then the Diophantine equation has no solution. Otherwise, a solution is:

$$x = b, \qquad y = \sqrt{\frac{p - b^2}{d}}.$$

Exercise 3: The Hailstone iteration again

Recall the *Hailstone iteration* from Exercise sheet 5.

a) Write a program `HailstoneSequence(n)` which returns the Hailstone sequence of an integer $n \geq 1$ (as a list). Thus `HailstoneSequence(11)` should return the list `[11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]`.

b) Write another program `HailstoneMax(n)` which returns the maximum entry of the Hailstone sequence of n. (It'll almost certainly be easiest if you use your program `HailstoneSequence` within `HailstoneMax`.) Make a list of the values of `HailstoneMax(n)` as n goes from 1 to 1000. Which is the most common value in the list?

c) Write a program `HailstonePlot(maxn, minshow, maxshow)` which does the following: for each n between 1 and `maxn`, it computes the Hailstone sequence of n. Then for each entry h in the Hailstone sequence which is between `minshow` and `maxshow`, it plots a point with coordinates (n, h). You should use the Maple command `pointplot` (in the `plots` package) to produce the plot.
Figure 1 shows an example of what you should aim for. (Try some other examples too.)
(The point of `minshow` is that low values are very common in Hailstone sequences: if you want to run `HailstonePlot` with larger values of `maxn`, then it will be substantially faster if you set, say, `minshow=50`, thus omitting all these low values from the plot.)

d) Can you explain the "ghostly" diagonal lines in the plot? (This is a *mathematical* question, though if you can deduce the equations of some of these lines you might like to verify your reasoning using Maple.)

Exercise 4: The Burau representation

This exercise is about representations of certain groups, but if you know nothing about group theory or representation theory you can still do it. There's a brief explanation of what it's all about at the end.

Let $n \geq 3$ be an integer. To each *non-zero* integer i with $-(n-1) \leq i \leq (n-1)$, associate an n by n matrix $M_n^i(t)$ as follows.

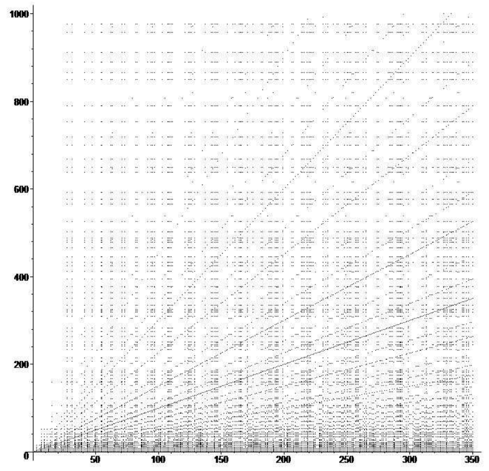

Figure 1: `HailstonePlot(350,1,1000)`

- Suppose $i > 0$. Then make $M_n^i(t)$ by starting with the n by n identity matrix, and changing the following entries: the one in position (i,i) to $1-t$; the one in position $(i+1,i+1)$ to 0; the one in position $(i,i+1)$ to t; and the one in position $(i+1,i)$ to 1. Thus, for example, $M_5^2(t)$ is the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1-t & t & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

  (A good way to look at it is: start with the n by n identity matrix, and then change a 2 by 2 block, with top left corner in position (i,i), to $\begin{matrix} 1-t & t \\ 1 & 0 \end{matrix}$.)

- Suppose $i < 0$. Then similarly start with the n by n identity matrix, and change a 2 by 2 block with top left corner in position $(-i,-i)$ to $\begin{matrix} 0 & 1 \\ \frac{1}{t} & 1-\frac{1}{t} \end{matrix}$. Thus, for example, $M_5^{-4}(t)$ is the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{t} & 1-\frac{1}{t} \end{pmatrix}.$$

a) Write a program `BurauGenerator(n,i)` which returns the matrix $M_n^i(t)$. Check that `BurauGenerator(5,2)` and `BurauGenerator(5,-4)` give the examples shown above.

b) Now write a program `BurauMatrix(n,L)`, where `L` is a list of non-zero integers between $-(n-1)$ and $(n-1)$, which returns the *product* of the matrices $M_n^i(t)$

for each i in the list. For example, `BurauMatrix(3,[1,1,-2,-1])` should return

$$M_3^1(t)\,M_3^1(t),M_3^{-2}(t)\,M_3^{-1}(t)=\begin{pmatrix}0 & 1-t+t^2 & t-t^2\\ 0 & 1-t & t\\ \frac{1}{t^2} & \frac{1}{t}-\frac{1}{t^2} & 1-\frac{1}{t}\end{pmatrix}.$$

As a check, make sure that `BurauMatrix(4,[2,3,2,-3,-2,-3])` returns the 4 by 4 identity matrix.

c) Write a program `BurauMatrixEval(n,L,s)`, where `s` is a complex number, `which returns BurauMatrix(n,L)` evaluated at $t=s$. For example, `BurauMatrixEval(4,[1,2,-3],-1+I)` should give

$$\begin{pmatrix}2-i & -1+3i & 0 & -2i\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -(1+i)/2 & (3+i)/2\end{pmatrix},$$

which is what you get when you put $t=-1+i$ in `BurauMatrix(4,[1,2,-3])`.

d) Write a program `BurauMatrixMaxAbsEigenvalue(n,L,s)` which returns the absolute value of the eigenvalue of `BurauMatrixEval(n,L,s)` which has largest absolute value. (That is: the eigenvalues of this matrix are complex numbers. Take the absolute values of each of those numbers, and return the biggest.) Note that you should calculate the eigenvalues *numerically* (i.e. using `evalf`), since it isn't possible to compute them symbolically when `n` is large. Check that `BurauMatrixMaxAbsEigenvalue(4,[1,2,-3],-1+I)` gives 2.2754, which is the size of the eigenvalue $2.1428+0.7655i$ of the matrix in c).

e) Write a program `BurauPlot(n,L)` which plots a graph of `BurauMatrixMaxAbsEigenvalue(n,L,`$e^{2\pi ix}$`)` as x goes from 0 to 1. For example, `BurauPlot(4,[1,2,-3])` should produce the graph of Figure 2.

f) Write a program `RandomBurau(n, length)` which calls `BurauPlot(n,L)`, where `L` is a random list with `length` elements (each between $-(n-1)$ and $(n-1)$). Try `RandomBurau(10,30);`.

Background

(For anyone interested with some knowledge of group and representation theory.)

For $n\geq 3$, the n-braid group B_n is generated by the $n-1$ elements $\sigma_1,\ldots,\sigma_{n-1}$, with relations $\sigma_i\sigma_{i+1}\sigma_i=\sigma_{i+1}\sigma_i\sigma_{i+1}$ for $1\leq i\leq n-2$, and $\sigma_i\sigma_j=\sigma_j\sigma_i$ whenever i and j are more than 1 apart (its elements can be thought of as *braids* with n strings). The Burau representation R is a representation of B_n in the space of invertible n by n matrices whose entries are polynomials in t and $1/t$. In the above, `BurauMatrix(4,[1,2,-3])` is $R(\sigma_1\sigma_2\sigma_3^{-1})$. This exercise is motivated by an ongoing research project: the maximum values attained by the plots are lower bounds for the *topological entropy* of the braids concerned.

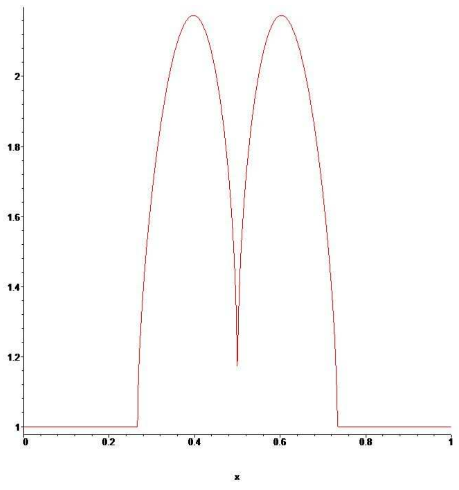

Figure 2: `BurauPlot(4,[1,2,-3])`

Exercise 5: The Logistic map

In this exercise, I've deliberately not given any guidance about the Maple methods to be used, but just set a problem to be answered using Maple. Any approach which gives the right answers is acceptable, but you'll probably find it easier if you write some procedures rather than just typing commands into your worksheet.

For each r between 0 and 4, the *logistic map* $f_r : [0,1] \to [0,1]$ is defined by $f_r(x) = rx(1-x)$. The second iterate f_r^2 is defined by

$$\begin{aligned} f_r^2(x) &= f_r(f_r(x)) = rf_r(x)(1-f_r(x)) = r(rx(1-x))(1-rx(1-x)) \\ &= -r^3x^4 + 2r^3x^3 - (r^3+r^2)x^2 + r^2x. \end{aligned}$$

Similarly $f_r^3(x) = f_r(f_r(f_r(x)))$, $f_r^4(x) = f_r(f_r(f_r(f_r(x))))$, and so on.

a) x^* is a *period 3 point* of f_r if $f_r^3(x^*) = x^*$, but $f_r(x^*) \neq x^*$. Find the period 3 points of $f_{3.5}$ and of $f_{3.9}$ (floating point, not exact).
Given (this isn't obvious) that there is some value r_0 such that f_r has no period 3 points for any $r < r_0$, but has period 3 points for all $r > r_0$, find r_0 to 6 decimal places.

b) A period 3 point x^* of f_r is *stable* if

$$-1 < \frac{df_r^3}{dx}(x^*) < 1.$$

Given (this isn't obvious) that there is a value r_1 such that f_r has stable period 3 points for all $r_0 < r < r_1$, but not for any $r > r_1$, find r_1 to 6 decimal places.