

Mathematical Experiments for Mathematics Majors

Michael Monagan
Department of Mathematics
Simon Fraser University.
`mmonagan@sfu.ca`

Abstract

At most universities in my province, British Columbia, pure mathematics majors now take one or two programming courses but they may spend little time subsequently writing programs to solve mathematical problems. At Simon Fraser University we require all mathematics majors (pure and applied) to take two programming courses and two additional courses, MACM 204 using Maple and MACM 203 using Matlab where the students are writing programs to solve mathematics problems and learning how to do mathematical experiments. This paper gives details of seven mathematics experiments that I ask students to do in Maple. The experiments help students develop and practice their programming skills. We believe the experiments, and the time spent mastering Maple and Matlab, will better prepare mathematics majors for careers in industry, research and teaching.

1 Introduction

At Simon Fraser University I teach the second year course MACM 204 Computing with Calculus. The course is a required course for all our mathematics majors. The prerequisites are an integral calculus course and a first programming course. Students attend one lecture hour and one lab hour per week for 13 weeks. We also require all our mathematics majors to take the second year course MACM 203 Computing with Linear Algebra which uses Matlab. Both MACM 203 and MACM 204 are two credit courses. The final exams are done in Maple and in Matlab.

The mathematics covered in MACM 204 is one variable calculus, a little bit of multivariate calculus (partial derivatives) and some modelling with first order systems of differential equations. One goal of MACM 204 is to review and apply one variable calculus.

A second goal of MACM 204 is to get mathematics majors to use a mathematical software package (we use Maple) to perform a variety of calculations for calculus. Obviously, we want the students to be able to calculate indefinite integrals and definite integrals, solve (systems) of algebraic equations, and solve differential equations. We want them to be able to do these calculations both exactly, and numerically.

A third goal is to teach the students to visualize everything they are doing. Maple, Matlab and Mathematica have a wide range of graphics capabilities. From a simple plot of $f(x)$ to plotting an implicit surface $f(x, y, z) = 0$ to creating field plots for systems of differential equations. This improves students understanding and prepares them to teach the mathematics.

A fourth goal is to teach the students how to do a “mathematical experiment”. This is the focus of this article. The experiment may be to disprove a conjecture, check a formula, find

an optimum solution, or generate an animation of a mathematical object. Doing mathematical experiments usually requires programming, hence the programming prerequisite. Indeed MACM 204 provides students a first opportunity to practice their programming skills on mathematical problems instead of on computing problems.

In this article I share seven mathematical experiments that I've found to be interesting and instructive for students. In brief they are

1. testing products of functions for orthogonality,
2. prime number races (See [1]),
3. comparing two numerical methods for measuring river flow (discharge) (See [4]),
4. generating random walks in the plane,
5. visualizing eigenvalues and eigenvectors (See [3]),
6. simulating mortgage interest charges and payments, and
7. simulating vaccination rates in the Kermack-McKendrick virus model (See [2]).

The first six experiments require programming. The experiments are chosen for their variety and level of difficulty. The first two experiments are just loops; for loops and while loops. The code is short; but even so, students who have taken only a first programming course will find these exercises challenging enough. The remaining exercises require more substantial programs but not more than one page of code. The mathematical experiments are, hopefully, of sufficient mathematical interest that the students will want to spend the time needed to get their code to work. Also, in order that the experiments are not the same every year I have suggested some variations on the experiments that you may use.

The reader may ask, why not instead integrate the use of the software package into the mathematics courses instead of having a separate course? Many universities take that approach for mathematics courses. We also have used Maple for our multivariate calculus course and Matlab for our linear algebra course. There are problems with that approach.

- 1 Students won't yet have a good grasp of the mathematics. They have difficulty understanding how to interpret the results and why they are doing it.
- 2 Adding computing exercises to mathematics courses increases the workload of the students, especially if there are programming exercises. Our students resented this.
- 3 There is no time to learn how to properly use the software package because the focus of the course is on the mathematics, as it should be.
- 4 If the software package is not required on exams many students take shortcuts on the computing exercises. Many will cheat. The final exam questions in MACM 203 and MACM 204 are all done in Matlab and Maple respectively; this discourages cheating.

I end with a discussion about writing efficient code with mathematics majors in mind. Also, I wanted to avoid talking about generative AI and cheating in this article. I will make some comments about what ChatGPT does when asked to write Maple code.

2 Experiment 1: Checking theorems

Consider the following statement.

$$\int_{-\pi/2}^{\pi/2} \sin(mx) \cos(nx) dx = 0 \text{ for all nonnegative integers } m > 0 \text{ and } n > 0.$$

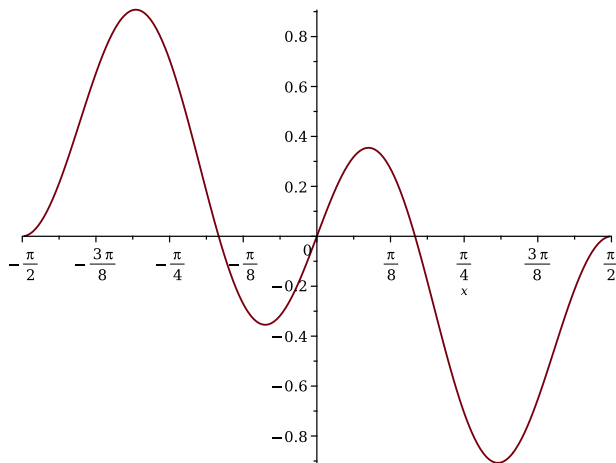
I ask the students to use Maple to check the integral for $1 \leq m \leq 5$ and $1 \leq n \leq 5$. Here is some Maple code to do this. ChatGPT will generate this code.

```
> for m from 1 to 5 do
>   for n from 1 to 5 do
>     print(m,n,int(sin(m*x)*cos(n*x),x=-Pi/2..Pi/2));
>   od;
> od;
```

```
1, 1, 0
1, 2, 0
1, 3, 0
...
5, 5, 0
```

After giving this exercise for several years, upon reflection, I found it unsatisfactory because it provides no insight as to why the integrals are 0. So I now ask the students to also graph some of the functions and, in one or two sentences, explain why the integrals are zero. Here is one plot

```
> plot(sin(3*x)*cos(2*x),x=-Pi/2..Pi/2 );
```



I am expecting an answer like “For each positive (and negative) area on the right of the y axis, there is an equal negative (positive) area on the left which cancel out so that the integral is 0. I was hoping that some students might remember from their Calculus courses since $\sin(3x)$ is an odd function and $\cos(2x)$ is even, their product $\sin(3x) \cos(2x)$ is odd thus the integral taken over $[-\pi/2, \pi/2]$ is 0. I was pleasantly surprised to find this answer.

The integral is zero because $\sin(mx) \cos(nx)$ simplifies to a sum of sine functions, which are odd. The integral of an odd function over a symmetric interval is always zero.¹

What is the sum of the sine functions the student is referring to? Maple's `combine` command combines products of trig functions using trig identities for $\sin A \sin B$, $\sin A \cos B$, and $\cos A \cos B$ to get

$$\frac{1}{2} \sin(x(m+n)) + \frac{1}{2} \sin(x(m-n)).$$

3 Experiment 2: Prime number races

I found this experiment in an article by Andrew Granville and Greg Martin [1].

Let P_i be the set of the first i primes. So $P_4 = \{2, 3, 5, 7\}$. Let s_i be the number of primes in P_i which are congruent to 1 mod 4 and let t_i be the number of primes in P_i that are congruent to 3 mod 4. So $s_4 = 1$ and $t_4 = 2$.

Consider the sequence $(s_1, t_1), (s_2, t_2), (s_3, t_3), (s_4, t_4), \dots$. Observe that $s_2 < t_2$, $s_3 = t_3$ and $s_4 < t_4$. The prime number race is; Are there more primes which are 3 mod 4 than 1 mod 4? That is, is $t_i > s_i$ as i increases most of the time or is $s_i > t_i$?

What I like about this problem is that it invites us to compute. Below is some initial data that I've computed where p_i is the i 'th prime.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	100	1000	2000
p_i	2	3	5	7	11	13	17	19	23	29	31	37	41	547	7919	17389
s_i	0	0	1	1	1	2	3	3	3	4	4	5	6	47	495	986
t_i	0	1	1	2	3	3	3	4	5	5	6	6	6	53	504	1013

From the data we see that after $p_{13} = 41$ where $s_{13} = t_{13} = 6$, $t_i > s_i$ and there is a sizable gap at $p_{2000} = 17,389$; we have the primes congruent to 3 mod 4 are ahead by $1013 - 986 + 1 = 26$. The primes congruent to 3 mod 4 are winning the race. Does $s_i > t_i$ occur? It turns out that it does. Eventually the primes congruent to 1 mod 4 catch up and overtake those congruent to 3 mod 4. The two experiments for the students are:

- Find the first prime for which $s_i > t_i$, and
- Go through the primes less than 10^6 and determine for what %age of the cases is $s_i > t_i$, $s_i = t_i$ and $s_i < t_i$.

Here is some Maple code to answer the first part.

```
> p := 2:
> s,t := 0,0:
> while s <= t do
>   p := nextprime(p);
>   if p mod 4 = 1 then s++; else t++; fi;
> od:
> p,s,t;

26861, 1473, 1472
```

¹I could not generate this answer using ChatGPT but neither can I know if it wasn't generated by ChatGPT.

Thus at $p_{2946} = 26,861$ the primes congruent to 1 overtake those congruent to 4 for the first time. For the second part I get 0.194%, 0.0284%, and 99.7776% respectively which shows that primes congruent to 3 mod 4 are ahead almost all the time.

Another prime number race is to consider the primes mod 10. Which of 1,3,7,9 wins the race? According to Granville and Martin [1] the primes congruent to 3 mod 10 and 7 mod 10 are ahead more often than 1 mod 10 and 9 mod 10.

3.1 Generative AI and ChatGPT

Can generative AI generate Maple code for the prime number race experiment? The answer is yes. I asked ChatGPT this question.

If P_i is the set of the first i primes and S_i is the number of primes in P_i which are congruent to 1 mod 4 and T_i is the number of primes in P_i which are congruent to 3 mod 4, write some Maple code to determine the first i such that $T_i > S_i$.

This is what I got. The Maple code is correct.

```
# Initialize counters
S := 0: # count of primes = 1 mod 4
T := 0: # count of primes = 3 mod 4

i := 1:

# Loop through primes until T > S
while true do
    p := ithprime(i);
    if p mod 4 = 1 then
        S := S + 1;
    elif p mod 4 = 3 then
        T := T + 1;
    end if;

    if S > T then
        printf("The smallest i such that S_i > T_i is %d\n", i);
        break;
    end if;

    i := i + 1;
end do;
```

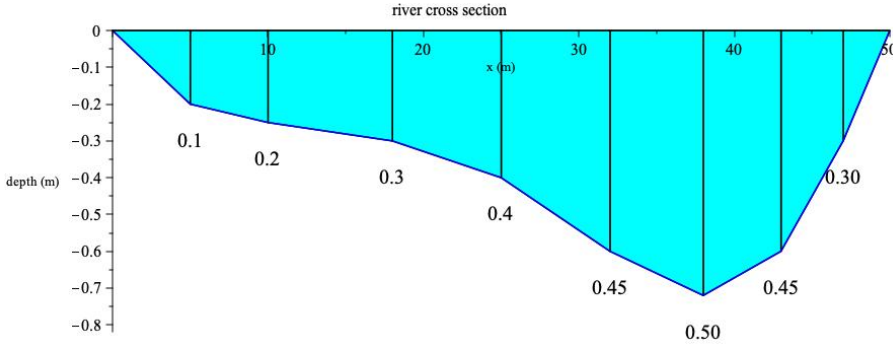
4 Experiment 3: Measuring river flow (discharge)

For my first technical job we had to measure the flow of rivers in Hawkes Bay, a province in New Zealand. We would wade across a river and, at selected positions x , measure the depth d of the river and average velocity v at that position. To measure the average velocity at position x we used a flow meter which is essentially a propeller. For a depth under 1m, we positioned the propeller at a depth 40% above the bottom of the river and used the velocity at that depth as an estimate for the

average velocity at position x . For a depth over 1m, we took the average of two measurements, one at 20% below the surface and the other at 20% above the bottom, as an estimate for the average velocity at position x . For a wide river we would get some data like this.

position x	0m	5m	10m	18m	25m	32m	38m	43m	47m	50m
depth d (m)	0.0	0.2	0.25	0.3	0.4	0.6	0.7	0.6	0.3	0.0
velocity v m/s	0.0	0.1	0.2	0.3	0.4	0.45	0.5	0.45	0.3	0.0

Let $x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n$ be the positions accross the river where the measurements are taken. Let d_i be the depth of the river at position x_i and v_i be the velocity of the river at position x_i . The computational question is how to estimate the total flow F from the data values (x_i, d_i, v_i) for $0 \leq i \leq n$. Here is a cross section of the river for the given data.



The flow F in m^3/s is given by the area of a cross section of the river times the average velocity of the river on that cross section. To calculate the flow from the data we divide the river into sections s_i for $1 \leq i \leq n$ where section s_i spans $x_{i-1} \leq x \leq x_i$. Now the flow on section s_i is the area of the section times the average velocity on that section. The obvious way to approximate this is to approximate the area of the section with a trapezoid and use the average of the two velocities v_i and v_{i-1} to approximate the average velocity on that section. This gives a flow estimate of

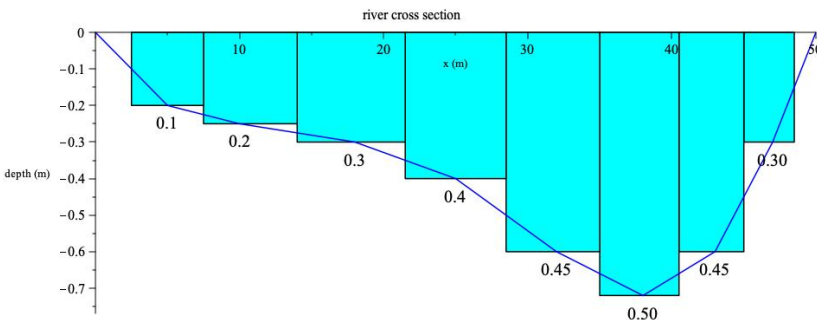
$$\frac{1}{4}(x_i - x_{i-1})(d_i + d_{i-1})(v_i + v_{i-1}) \text{ m}^3/s$$

for section s_i . This is the method that we used in New Zealand to calculate the flow. I call this method the Trapezoidal method. Terzi in [5] calls it the “mean section method”. I was asked to program it on a new HP plotter.

Many years later I learned of a different way of estimating the flow that I call the midpoint method. Terzi [5] calls it the “mid section method”. For each position x_i for $0 < i < n$ consider the section from $x = (x_{i-1} + x_i)/2$ to $x = (x_i + x_{i+1})/2$. Approximate the flow with

$$d_i v_i \left(\frac{x_i + x_{i+1}}{2} - \frac{x_i + x_{i-1}}{2} \right) = \frac{1}{2} d_i v_i (x_{i+1} - x_{i-1}) \text{ m}^3/s.$$

This approximation is illustrated in the following figure.



This is not the midpoint method from Calculus! That midpoint method would use a depth at the midpoint of section s_i which is $x = \frac{x_i + x_{i+1}}{2}$. But we don't have depth at that point! We can't go back to the river to try to get it!

The experiment for the students is to determine which approximation is best. To do this we need to have a river for which we can compute the exact flow F . I teach students that if we are given functions $d(x)$ for the depth at position x and $v(x)$ for the average velocity of the river at position x the flow F on $a \leq x \leq b$ is given by

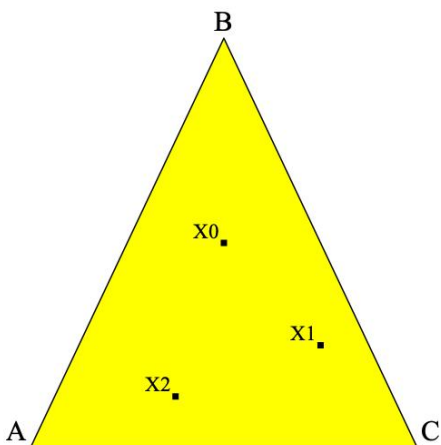
$$F = \int_a^b d(x)v(x)dx \text{ m/s}^3.$$

For the experiment one may use $d(x) = x(10 - x)/25 \text{ m}$ and $v(x) = x(10 - x)/100 \text{ m/s}$ for $0 \leq x \leq 10 \text{ m}$. Here the flow $F = \int_0^{10} d(x)v(x)dx = \frac{4}{3} \text{ m}^3/\text{s}$. I ask the student to program and calculate the flow using the Trapezoidal method and Midpoint method for $n = 7$ measurements taken at equally spaced intervals, so 1.25m apart.

What the student will find is that the Midpoint method is more accurate than the Trapezoidal method. The Trapezoidal method is biased; the Midpoint method is not. Assuming the river bed has a concave up shape with velocity greater where the depth is greater, the Trapezoidal method underestimates the flow.

5 Experiment 4: Random walks in the plane

Let $A = (0,0)$, $B = (1,1)$ and $C = (2,0)$ be vertices of a triangle. Suppose X is a vertex inside the triangle. The random walk is the following. Pick one of A, B, C at random and go half way from X to that vertex. Set X to be the new position. Repeat this many times. The following figure shows what happens if we start with $X_0 = (1, 0.5)$ and pick C then A to get X_1 and X_2 .



I ask the students to simulate the random walk and create a plot of the points and identify the image that they get. To do this I have shown the students how to create a random number generator using Maple's **rand** command in a lab. Here is my Maple code

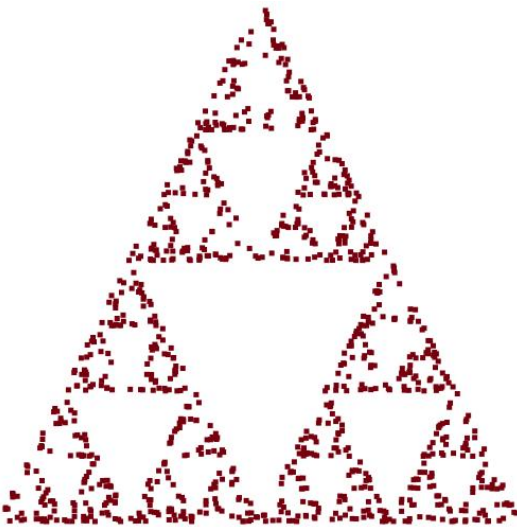
```
> A,B,C := [0,0],[1,1],[2,0];
> n := 1000;
> Walk := Array(0..n): # for storing X0, X1, X2, ...
> X := [0.5,0.5]:
```

```

> Walk[0] := X:
> R := rand(1..3): # R is a random number generator for {1,2,3}
> for i to n do
>   u := R();
>   if u=1 then X := (X+A)/2;
>   elif u=2 then X := (X+B)/2;
>   else X := (X+C)/2;
>   fi;
>   Walk[i] := X;
> od:
> L := convert(Walk,list):
> plot( L, style=point, symbol=solidbox );

```

Many of the students will update X by computing the new x and y co-ordinates separately. Here is the figure (it's Sierpinski's gasket) for $n = 1000$.



A variation on the problem is to start with four vertices $A = (0,0)$, $B = (0,2)$, $C = (2,2)$, $D = (2,0)$ of a square and start with a vertex X inside the square. Pick one of A, B, C, D at random. If you pick A go half way to A . If you pick B go $2/3$ of the way to B . If you pick C go half way to C . If you pick D go $2/3$ of the way to D . This random walk also generates a fractal.

6 Experiment 5: Visualizing eigenvalues and eigenvectors

When we teach eigenvalue and eigenvectors we start with the definition and an example. Then most texts give some properties and show how to calculate the eigenvalues and eigenvectors via the characteristic polynomial. For example, here is a matrix and it's eigenvalues and eigenvectors computed in exactly Maple.

```

> A := Matrix([[1,1],[1,0]]);

```

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

```

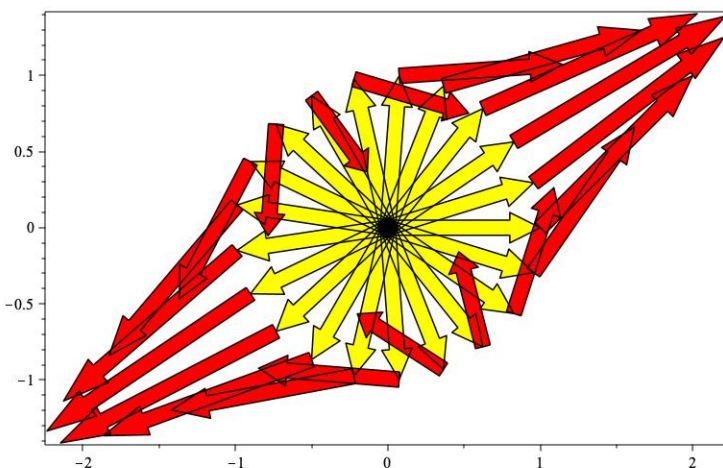
> with(LinearAlgebra):
> E,V := Eigenvectors(A);

```

$$E, V := \begin{bmatrix} \frac{\sqrt{5}}{2} + \frac{1}{2} \\ -\frac{\sqrt{5}}{2} + \frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{\frac{\sqrt{5}}{2} - \frac{1}{2}} & \frac{1}{-\frac{\sqrt{5}}{2} - \frac{1}{2}} \\ 1 & 1 \end{bmatrix}$$

Here E is the vector of eigenvalues of A and the columns of the matrix V are the corresponding eigenvectors of A .

I've always thought we should spend more time on the understanding the definition from a visual perspective. Because, it will help the students remember the definition and some of the properties. So I get them to write a program to create the following figure. In the figure the yellow vectors are unit vectors equally spaced around the unit circle and the red vectors are obtained by multiplying A by the unit vectors. One can see where the eigenvectors are and one can estimate their eigenvalues from the figure. I then ask the students to experiment with different 2 by 2 matrices, including a rotation matrix, whose eigenspaces are qualitatively different.



Here is my solution.

```
> with(plots):
> n := 21;
> A := Matrix([[1,1],[1,0]]);
> U := Array(1..n):
> V := Array(1..n):
> for i to n do
>   theta := 2.0*Pi*i/n;
>   u := <cos(theta),sin(theta)>;
>   v := A.u;
>   U[i] := arrow([0,0],u,width=0.1,color=yellow);
>   V[i] := arrow(u,v,width=0.1,color=red);
> od:
> L := convert(V,list),convert(U,list):
> display(L);
```

To prepare for this, in the lab, I show them the trigonometric parameterization of the circle of radius r which is $x = r \cos(\theta)$, $y = r \sin(\theta)$ and how to plot an arrow in Maple. For the students, knowing the trigonometric parameterization of the circle is perhaps the most useful parameterization to know.

7 Experiment 6: Simulating a mortgage payment

Suppose you take out a 30 year mortgage of \$200,000 at an annual interest rate of $r = 5\%$ per year compounded daily. Let $M(t)$ be the amount you owe on a 30 year mortgage at time t . That bank (should) determine the annual payment P \$/year such that $M(30) = 0$. We can approximate the process with a DE.

$$M'(t) = rM - P \text{ \$/year}$$

First solve the DE with the initial value $M(0) = 200000$ then solve for P such that $M(30) = 0$.

```
> r := 0.05; # annual interest rate
> de := diff(M(t),t) = r*M(t)-P;
> sol := dsolve( {de,M(0)=200000} );
```

$$\text{sol} := M(t) = 20P + (200000 - 20P)e^{t/20}$$

```
> P := fsolve( eval(rhs(sol),t=30), P );
P := 12872.16917
> interest := 30*P-200000;
186165.0751
```

The experiment is

- (a) Starting with mortgage $M(0) = \$200,000$ simulate daily interest charges at a rate of $5\%/365$ per day and monthly mortgage payments at a rate of $\$P/12$ per month for 30 years. Why do you not get to \$0?
- (b) Repeat this with daily interest charges and daily payments of $\$P/365$. How close to $M(30) = 0$ to you get?

Here is my Maple code for part (a).

```
> M := 200000;
> r := 0.05/365;
> days := [31,28,31,30,31,30,31,31,30,31,30,31];
> for y to 30 do
>   for m to 12 do
>     for d from 1 to days[m] do M := M+r/365*M; od;
>     M := M-P/12; # pay on the last day of the month
>   od;
> od;
> M; # should be close to 0
1733.447541
```

8 Experiment 7: The Kermack-McKendrick virus model

At the end of my MACM 204 course I spend two weeks on first order differential equations then two weeks of the course on first order systems of differential equations, ending with a study of the Kermack-McKendrick virus model from [2]. The model is a SIR model. Here S represents

the population that is susceptible to the virus, I those that are infected and R those who have recovered (or died) from the virus. The model is given by

$$\frac{dS}{dt} = -\beta SI, \quad \frac{dI}{dt} = \beta SI - \alpha I, \quad \frac{dR}{dt} = \alpha I$$

where the parameter $\beta > 0$ captures the infection rate and $\alpha > 0$ the recovery rate. The following Maple code generates a plot of $S(t)$, $I(t)$ and $R(t)$ for a hypothetical scenario.

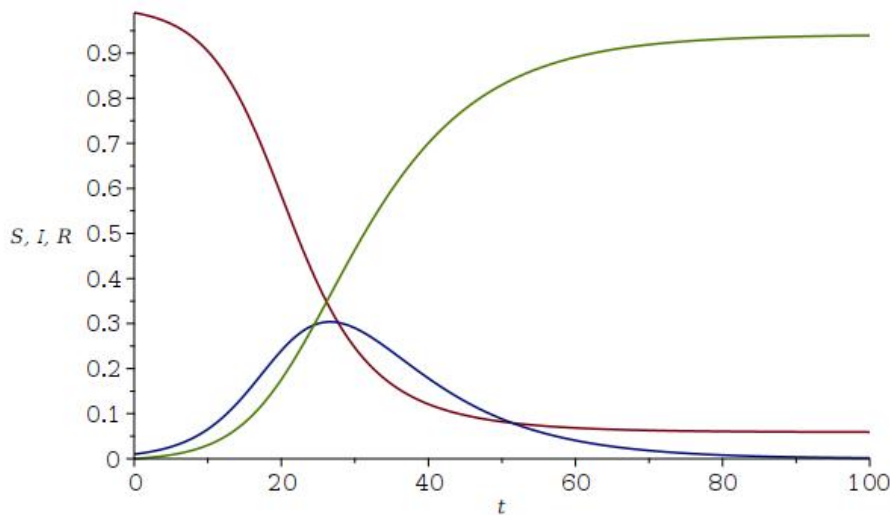
```
> local I; # so that we can use I for I(t)
> de1 := diff(S(t),t) = -beta*S(t)*I(t):
> de2 := diff(I(t),t) = +beta*S(t)*I(t)-alpha*I(t);
```

$$\frac{d}{dt}I(t) = \beta S(t)I(t) - \alpha I(t)$$

```
> de3 := diff(R(t),t) = alpha*I(t):
> alpha := 0.1:
> beta := 0.3:
```

The following Maple code solves the system of ODEs numerically for $S(0) = 0.99$, $I(0) = 0.01$ and $R(0) = 0.0$, and graphs the solutions on $0 \leq t \leq 100$.

```
> F := dsolve( {de1,de2,de3,S(0)=0.99,I(0)=0.01,R(0)=0.0}, numeric ):
> plots[odeplot]( F,[[t,S(t)],[t,I(t)],[t,R(t)]], t=0..100, numpoints=200 );
```



The plot shows that the virus is epidemic, that is, the number of infected individuals is increasing at time $t = 0$. It also shows that after a long time about 5% of the population survives, that is, does not get infected.

Public health officials want to know what proportion of the population needs to be vaccinated to achieve “herd immunity”, which means the infection rate will decrease. The mathematical experiment for the students is to simulate vaccinations of 50% by making $R(0) = 0.50$ say and setting $S(0) = 1 - R(0) - I(0) = 0.49$, solving the system then testing to see if $I'(0)$ is positive (the virus is epidemic), zero, or negative (the virus is endemic).

9 Programming and Quadratic Algorithms

Doing computational mathematical experiments means the students have to program. So we require a first programming course as a prerequisite. Most students, after a first programming course, will be weak at programming.

How much time should we spend teaching them how to program in MACM 204? In MACM 204 I used to spend two weeks (two lectures and two labs) teaching the students how to program in Maple. I spent one week on Maple conditional statements, for loops and while loops, and doing examples using Maple sets and lists. I then spent a second week teaching them how to write Maple procedures with a focus on the difference between local variables and global variables. I now spend a third week on programming with arrays and explaining the difference between a quadratic algorithm and a linear algorithm. I explain why.

I used to pose the prime number race experiment from Section 2 this way.

Let P_i be the **set** of the first i primes. Let S_i be the **subset** of primes in P_i which are congruent to 1 mod 4 and let T_i be the **subset** of primes in T_i which are congruent to 3 mod 4. So $P_4 = \{2, 3, 5, 7\}$, $S_4 = \{5\}$ and $T_4 = \{3, 7\}$. Find the first i such that $|S_i| > |T_i|$ and print out the prime when this happens.

Most students, naturally, constructed the sets S_i and T_i ! They did not realize that they just need counters for the sizes of the sets. So they wrote Maple code like this.

```
> p := 2:
> S := {}:
> T := {}:
> i := 1:
> while nops(S) <= nops(T) do
>   p := nextprime(p);
>   if p mod 4 = 1 then S := S union {p} else T := T union {p} end if;
>   i := i+1;
> end do:
> i,p,nops(S),nops(T);
```

2946, 26861, 1473, 1472

The problem with this code is that it creates all intermediate sets S_i and T_i . The Maple operation `S union {p}` creates a new set of size one larger by copying the elements of S into new storage and appending the new prime p . So the time the operation `S union {p}` takes is proportional to $|S|^2$ the size of the set. If there are n primes in S then the total time to create the sets S_i for $1 \leq i \leq n$ is $O(n^2)$, that is, quadratic in n . Using counters for the sizes of the sets instead like this

```
> p := 2:
> s,t := 0,0:
> i := 1:
> while s <= t do
>   p := nextprime(p);
```

²Sets in Maple are sorted. Inserting one new element into a set using `S union {p}` and sorting the new set is linear in the size of S

```

>   if p mod 4 = 1 then s++; else t++; end if;
>   i++;
> end do:
> i,p,s,t;

```

2946, 26861, 1473, 1472

is much more efficient; the total cost of the algorithm will be $O(n)$, that is, linear in n . This linear loop takes 0.036 seconds on my computer. The quadratic algorithm using sets takes 1.360 seconds, a factor of $1.36/0.036 = 38$ times slower! Here $n = 1473$ is not too big. If n were 10 times bigger the linear loop would take 0.36 seconds and the quadratic loop would take 136 seconds. This quadratic problem arises in other Maple constructions including lists and sums. It is an issue in all Computer Algebra Systems. Let me give another example.

Looking back at the random walk experiment from Section 5, many students will use Maple lists instead of Maple arrays because I showed how to use lists early in the course and the `plot` command accepts a list of points as input. In other words they code

```

> A,B,C := [0,0],[1,1],[2,0];
> n := 1000;
> Walk := []; # empty list
> X := [0.5,0.5]:
> Walk[0] := X:
> R := rand(1..3): # R is a random number generator for {1,2,3}
> for i to n do
>   u := R();
>   if u=1 then X := (X+A)/2;
>   elif u=2 then X := (X+B)/2;
>   else X := (X+C)/2;
>   fi;
>   Walk := [op(Walk),X]; # append X to the list Walk
> od:
> plot( Walk, style=point, symbol=solidbox );

```

Here the operation `[op(Walk),X]` creates a new list by copying the entries of the list `Walk` into new storage and appending `X`. This operation is also linear in the size of the list `Walk`. This code takes 0.032 seconds on my computer. Using an Array instead takes 0.014 seconds. Obviously for $n = 1000$ it does not really matter whether we use a quadratic algorithm or linear algorithm here. But, to create a high quality image of the fractal we might want to use $n = 50,000$ or more points. If we do this the quadratic code above takes 70.34 seconds whereas the linear code takes 0.33 seconds.

So an efficient solution to the random walk problem is to use an array not a list. And an efficient solution to the prime number race problem is to use counters not sets. What if we asked the students to create an array of the primes congruent to 3 mod 4 instead of a set S_i ? The difficulty is that we don't know in advance how many primes are in the set. Maple has a solution for this. The `Append(A,x)` command in the `ArrayTools` package efficiently appends a new element x to an Array A . Here is how we can use it.

```

> with(ArrayTools):
> p := 2:
> s := 0:

```

```

> T := Array(1..0): # start with an empty array
> i := 1:
> while s <= numelems(T) do
>   p := nextprime(p);
>   if p mod 4 = 1 then s++; else Append(T,p); end if;
>   i++;
> end do:
> i,p,s,numelems(T);

```

2946, 26861, 1473, 1472

This code took 0.043 seconds on my computer. For comparison the code with two counters took 0.036 seconds and code with sets took 1.36 seconds. I believe this lesson about the difference between a quadratic algorithm verses a linear algorithm is the most important lesson on writing efficient code that we can teach students. Yet I find that students don't care about this difference unless they experience the pain of waiting for a quadratic algorithm to finish. So I force them to time the execution of the two random walk algorithms for $n = 50,000$, the list version and the array version, so they have to wait for the quadratic algorithm to finish.

How does the `Append(A,x)` command work? If the array A is full, to insert x into A , instead of increasing the size by one, Maple increases the size by 50% of the current size. So if $|A| = 10$ Maple will create a new array of size 15, copy the 10 elements from the old array into the new one, insert x at index 11, so that the new array has space to insert 4 elements without having to increase its size. If we insert n elements using `Append(A,x)` the total time is linear in n not quadratic in n .

10 Conclusion

MACM 204 (and also MACM 203) teach students how to use Maple (Matlab) to do mathematical calculations using built in commands and also how to do mathematical experiments. Instead of requiring mathematics majors to do another "mathematics course" we think it more beneficial if they instead learn to use mathematical software packages and develop their programming skills on problems from mathematics rather than problems from computer science. Our department recently considered adding a third such course, MACM 205 Computing with Discrete Mathematics using Python, but we don't have the teaching resources to do this at this time.

References

- [1] Andrew Granville and Greg Martin. Prime Number Races. *The American Mathematical Monthly* **113**(1):1–33, 2018.
- [2] Kermack, W. O. and McKendrick, A. G. A Contribution to the Mathematical Theory of Epidemics. *Proc. Royal Soc. London A* **115**:700–721, 1927.
- [3] Michael Monagan. 2D and 3D Graphical Routines for Teaching Linear Algebra, *Proceedings of the 2002 Maple Summer Workshop*, Waterloo Maple Inc., 2002.
- [4] Michael Monagan. Measuring River Flow. *The Maple Reporter*, Issue 12, Maplesoft, 2013.
- [5] R.A. Terzi. Hydrometric Field Measurement, 1981.
https://publications.gc.ca/collections/collection_2021/eccc/en37/En37-525-1981-eng.pdf