MATH36032 Coursework One

Student ID: 10098348

March 2021

Introduction

This is my report for MATH36032 project one. The goal of which was to solve three problems in number theory using Matlab. The main body of the report is split into three sections, one for each section. In each section, I provide an explanation of the problem and the Matlab functions used to solve the problem. The full code for each function is presented in an appendix.

1 Question One

The aim of this question is to approximate the Euler-Mascheroni constant, γ, by a rational number, p/q where p and q are natural numbers. The Euler-Mascheroni constant is defined by

$$\gamma = \lim_{n\to\infty}\Big(-ln(n) + \sum_{k=1}^{n}\frac{1}{k}\Big).$$

Up to 15 decimal places, it has approximate value 0.577215664901533. Given an integer N, we want to find natural numbers p and q such that $|p/q - \gamma|$ is minimal amongst all positive integers p and q with $p + q \leq N$.

To do this, we defined a function in Matlab called AppEM, which takes an integer N as input and returns two integers p and q such that p/q is the best approximation of γ satisfying certain conditons. Aside from the already stated conditions (p and q are natural numbers and $p+q \leq N$), we also want that if there is more than one pair which provide the best estimate then the function will return the pair $[p, q]$ with smallest sum.

The function works as follows, first we define emconst as the approximate value of γ to 15 decimal places and we give p and q initial value 1. We also define a variable absdiff which will later be used to store the value of $|emconst - p/q|$ but first we initialise it with a high value that isn't important.

Now we start two for loops with variables $p1$ and $q1$, both of which run from 1 to N, with the $q1$ loop inside the $p1$ loop. We calculate $p1/q1$, an approximation for the E-M constant, and store it in a variable app. Then we calculate the absolute difference between the approximation and emconst and store it as appdiff. If this value is less than absdiff and $p + q \leq N$, then we set the value of absdiff equal to appdiff and p and q equal to $p1$ and $q1$ respectively. If, instead, appdiff = absdiff and $p + q \leq N$ and $p1 + q1 < p + q$, we set $p = p1$ and $q = q1$. If appdiff > absdiff, then all values remain unchanged.

For input $N = 2021$, we get output

```
>> [p,q] = AppEM(2021)

p =

   228


q =

   395
```

2 Question Two

The goal of this question is to find the smallest MyLucky number greater than or equal to a given number. A MyLucky number, n, is defined by two conditions:

- all of n's prime factors are odd and distinct (and therefore n has at least two prime factors)
- if p is a prime factor of n, then $p - 1$ divides $n - 1$.

To solve this problem, we defined two functions. The first, isMyLuckynum, tested if a given number was MyLucky and the second, MyLuckynum, used the first and found the smallest MyLucky number greater than or equal to a given number.

The function isMyLuckynum takes a double as input and outputs a logical - true if the input is MyLucky and false if it is not. First, for input x, use the inbuilt factor function to create an array of factors of x, and store it in f. We then check if the factors are distinct by checking if the length of the array is equal to the length of the vector containing the unique elements of f and

we also check if there is more than one factor. If either of these conditions is not met the function outputs false. If they are, we start a for loop and use it check if a factor is even or if for a factor p, $p-1$ does not divide $x-1$, using Matlab's inbuilt mod function and if they are then we set the output to false and break out of the loop. If all the factors are odd and "factor -1" divides $x-1$, for all factors, then we set the output to true.

The function MyLuckynum takes a double, N, as input and outputs a double, n, where n is the smallest MyLucky number greater than or equal to N. The function makes use of $isMyLuckynum$ and works as follows. We use a while loop so that whilst N is not MyLucky (i.e. isMyLuckynum outputs false) we increase N by 1. If N is MyLucky, or when it eventually equals a MyLucky number - after passing through the while loop - we set $n = N$ and output.

With input $N = 2021$, we get output

```
>> n=MyLuckynum(2021)

n =

        2465
```

Carmichael numbers are composite numbers n such that $a^{(n-1)} \equiv 1 \pmod{n}$ for all a coprime to n. Korselt's Criterion[1] gives an alternate definition for Carmichael numbers - n being odd and squarefree (not divisible by a perfect square) and for all prime factors p, $p-1$ divides $n-1$. This is precisely the same as the conditions for MyLucky numbers. Looking at the OEIS series for Carmichael numbers[2], we see that the smallest Carmichael (equivalently, MyLucky) number greater than or equal to 2021 is 2465, confirming the output of the function for $n = 2021$.

We can compare the output of the function with the series for Carmichael numbers for other outputs too, to show that the two values are in agreement. A few examples are shown in the table below.

Input	Output of function	smallest Carmichael number $\geq$ input in OEIS series
1	561	561
1729	1729	1729
5679	6601	6601
39604	41041	41041
99601	101101	101101
414221	449065	449065

3 Question Three

The problem here, is, given an integer N, to find an integer n that is a beautiful square such that n^2 is closest to N. A beautiful square number is a number whose square consists of all nine non-zero digits and each digit appears only once. To do this we wrote a function Beautisqnum. The function Beautisqnum makes use of a function dig2num which is the same as the code from labdemo5.1[3], just turned into a function. The steps behind the function Beautisqnum are explained below.

First, we create a variable diff which will later be used to store the absolute difference between the input N and the guess at n^2, but here initialise it with a value of 10^{10} or higher. Then we start a for over i loop from $\lceil\sqrt{123456789}\rceil = 11112$ to $\lfloor\sqrt{987654321}\rfloor = 31426$. (We can do this because we want to find n such that $123456789 < n^2 < 987654321$ satisfying certain conditions.)

Inside the loop, calculate i^2 and use the function num2dig to create a vector, dig, of digits of i^2. We use dig to check that i^2 satisfies the conditions for a beautiful square number. We do this by using an if statement with the conditions that the length of dig is equal to the length of the vector containing unique elements of dig and that the minimum and maximum elements of the vector are 1 and 9, respectively. Since i^2 is a nine digit number, by choice of i, these conditions are the same as for i being a beautiful square number.

If i^2 is the square of a beautiful square number then we calculate the absolute difference between N and i^2. If this calculation is less than or equal to the value of diff, then we set diff equal to it and set n equal to i. If the conditions above are not met then we continue and pass control to the next iteration of the loop. This method outputs the beautiful square number, n such that n^2 is closest to N. For example, for input $N = 360322021$, we get output:

```
>> n = Beautisqnum(360322021)

n =

       19023
```

References

1. People.math.gatech.edu. 2011. Korselt's crietrion for Carmichael numbers. [online] Available at: (http://people.math.gatech.edu/ mbaker/pdf/korselt.pdf) [Accessed 9 March 2021].

2. Sloane, N., n.d. A002997 - OEIS. [online] Oeis.org. Available at: (https://oeis.org/A002997) [Accessed 9 March 2021].

3. Gajjar, J., 2021. labdemo5. [online] Available at: (https://online. manchester. ac.uk/bbcswebdav/pid-12215832-dt-content-rid-66441489_1/ courses/I3133-MATH-36032-1201-2SE- 009260/labdemo5.html) [Accessed 10 March 2021].

Appendix

Code for question 1

AppEM.m:

```
function [p, q] = AppEM(N)
%The function AppEM will find the best approximation of the Eurler-
%Mascheroni constant by a rational number p/q, with p,q natuaral numbers
%and p + q <= N such that abs(p/q - emconst) is smallest among all positive
%integers

emconst = 0.577215664901533;

absdiff = 100000000000000; %initialise with high value
p = 1; q = 1; %initialise

for p1=1:N
    for q1=1:N
        app = p1/q1;
        appdiff = abs(emconst - app);
        if appdiff < absdiff & p1 + q1 <= N
            absdiff = appdiff; %if the new approximation is closer to
            p = p1;            %emconst than the current one and satisfies
            q = q1;            % the "p+q<=N condition" then change acc to
                               %appaccp to p1 and q to q1
        elseif appdiff == absdiff & p1 + q1 < p + q & p1 + q1 <= N
                p = p1; %if the new approximation is equal to the current
                q = q1; %one but the new value of "p+q" is less than the
                        %old one then change p to p1 and q to q1
            end
        end
    end
end
```

Code for question 2

isMyLuckynum.m:

```
function y = isMyLuckynum(x)
%isMyLuckynum tests if a number is MyLucky and returns true if it is and
%false if it is not

f = factor(x);
```

```
if length(f) > 1 & length(f) == length(unique(f))
    for i = 1:length(f)
        if mod(f(i),2) ~= 1 | mod(x-1, f(i)-1) ~= 0;;
            y = false; %if a factor is even or (factor-1) does not divide
            break      %(n-1) then set y = false and break out of loop
        else
            y = true; %if the conditions are met, set equal to true
        end
    end
else
    y = false; %if the conditions aren't met set equal to false
end

end
```

MyLuckynum.m:

```
function n = MyLuckynum(N)
%The function MyLuckynum finds the samllest Mylucky number that is greater
%than or equal to N. It uses the isMyLucynum function as an integral
%component.

while ~isMyLuckynum(N)
    N = N+1; %if input is not MyLucky change it to N+1 until it is
end
if isMyLuckynum(N)
    n = N; %if/when N is MyLucky output n = N
end
```

Code for question 3

dig2num.m[3]:

```
function dig = num2dig(x)
%When given a number x num2dig outputs a vector of digits of x
%It is the same code as num2dig from labdemo5.1

%dig = rem(n,10);
%n = floor(n/10);
dig = [];
while (x>0)
    dig = [rem(x,10) dig];
```

```
    x = floor(x/10);
end
end
```

Beautisqnum.m:

```
function n = Beautisqnum(N)
%The function Beautisqnum returns output n, such that n^2 is a perfect
%square and all nine digits of n^2 are distinct, which is closest to the
%input N

diff = 100000000000; %initialise with high value

for i = 11112:31426 % = ceil(sqrt(123456789)):floor(sqrt(987654321))
    i2 = i^2;
    dig = num2dig(i2); %create array of digits of i^2
    if length(dig)==length(unique(dig)) & max(dig)==9 & min(dig)==1
    %conditions to check if digits of i^2 are unique and between 1 and 9
        calc = abs(N - i2); %calculate absolute difference between input
                            %and current i^2
        if calc <= diff;  %if above calculation is less than current
            diff = calc;  %current difference, then replace diff with calc
            n = i;        %and n with i
        end
    else
        continue %if conditions are not met continue with for loop
    end
end
end
```