MATH49111 Coursework 1

Rudi Agnew 1013652

Abstract

The aim of this project was to approximate the exponential function using the recurrence relation between terms in its Maclaurin series. The code for this project was done in C++ whilst the graphing was done in MATLAB.

1 A recurrence relation

The Maclaurin series for the exponential function is as follows

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

Let the nth term of this series be denoted $T_n(x)$ such that $T_0(x) = 1$, $T_1(x) = x$ and so on. Now for n = 2 we have:

$$T_n = T_2 = \frac{x^2}{2 \cdot 1} \quad \text{and} \quad T_{n-1} = T_1 = \frac{x}{1}.$$

Clearly for these two terms to be equal we need to multiply T_{n-1} by $x/2$ or x/n. Looking back at n = 1 this still makes sense as for 1 to equal x we need to multiply the 1 by x which is the same as multiplying by $x/1$. Similarly for n = 3 we have:

$$T_3 = \frac{x^3}{3 \cdot 2 \cdot 1} \quad \text{and} \quad T_2 = \frac{x^2}{2 \cdot 1}.$$

Again for these two to be equal we need to multiply the lower term by x/n, this leads to the recurrence relation

$$T_n = \frac{xT_{n-1}}{n} \qquad \text{for } n > 0. \tag{1}$$

2 Approximating the exponential function

My task was then to write a C++ function that approximated e^x using its Maclaurin series. To avoid having to deal with large factorials, I made use of (1) in my code. To sum N+1 terms I always included $T_0(x)$ in the sum and then used a for loop to sum the remaining N terms.

Listing 1: ExpSeries

```
double ExpSeries(double x, int N)
{
        // approximates exp(x) by Maclaurin series using a recurrence relation
        // recurrence relation : T_n = x/n * T_n-1
        // defining first term
        double a = 1;
        // Sums N+1 terms as first term is always included here
        double sum = a;
        for (int i = 1; i <= N; i++) //sums N terms
        {
                // calculates next term using recurrence relation
                double b = (x / i) * a;
                sum += b;
                a = b;
        }
        return sum;
}
```

To test my code I inputted a few examples, namely x = 1.0, N = 2 which should give 1 + 1 + 1/2 = 2.5 and indeed it gave the correct result. I also tested x = 1, N = 50 to see if my approximation was working well and it was. Just to be sure I changed x to 5 and using N = 2 which should give $1 + 5 + 5^2/2 = 18.5$ and again my code worked.

Listing 2: Calling ExpSeries()

```
int main()
{
        std::cout << ExpSeries(1.0, 2) << std::endl;
        std::cout << ExpSeries(1.0, 50) << std::endl;
        std::cout << ExpSeries(5.0, 2) << std::endl;
        return 0;
}
```

Output:

```
2.5
2.71828
18.5
```

3 Generating data

Here I used my function to generate a data set. The first column being x, second the value of exp(x) from the standard library and then the next columns being the two, three and four term series approximation to exp(x).

Listing 3: Main function to generate data set

```
int main()
{
        // declare object of type std::ofstream
        std::ofstream File;

        // try opening file
        File.open("Data.txt");

        // if file failed to open, exit main returning 1
        if (!File) return 1;

        // writing to the file
        for (double z = -1.0; z < 1.02; z+=0.02)
        {
                File.width(10); File << z;
                File.width(10); File << exp(z);
                File.width(10); File << ExpSeries(z, 1);
                File.width(10); File << ExpSeries(z, 2);
                File.width(10); File << ExpSeries(z, 3) << std::endl;
        }

        // close the file
        File.close();

        return 0;
}
```

This returned 101 rows of data evenly spaced between -1 and 1. Here are the first few rows.

```
        -1  0.367879         0       0.5  0.333333
     -0.98  0.375311      0.02    0.5002  0.343335
     -0.96  0.382893      0.04    0.5008  0.353344
```

4 Graphing the data

I then used MATLAB to graph my data.

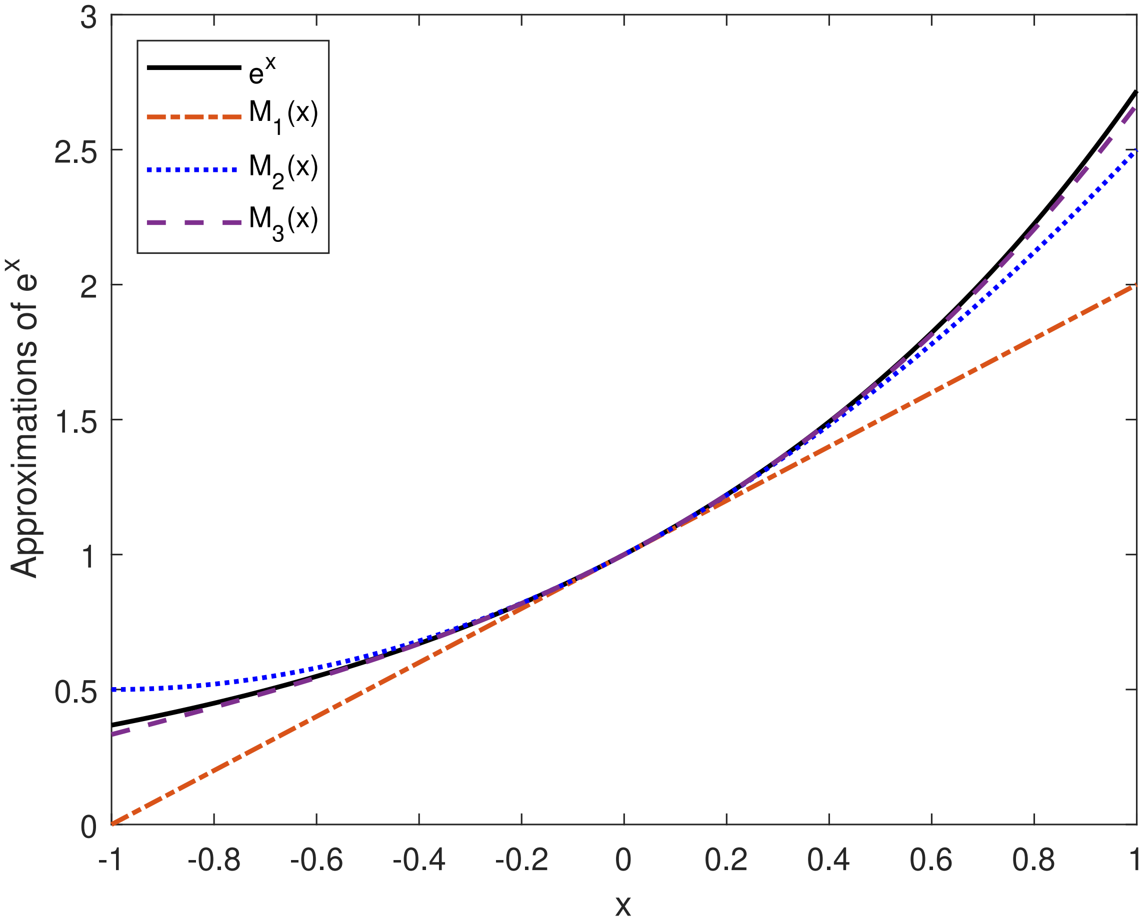

Figure 1: Figure showing different approximations of exp(x).