An approximation to e^x MTH 125

Infinite Series for ex

• The infinite series for the exponential function is:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots + \frac{x^n}{n!} + \dots$$

Compute Using Partial Sums

■ We can approximate e^x using partial sums:

$$\begin{split} E_1 &= 1 \\ E_2 &= 1 + x \\ E_3 &= 1 + x + \frac{x^2}{2} \\ &\cdot \end{split}$$

$$E_n = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots + \frac{x^n}{n!}$$

But how do we know when to stop? Or how many terms to take?

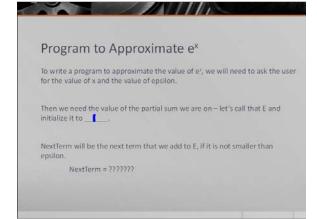
Partial Sums continued

 One way to know how many terms to use is to keep adding the terms of the series until the last term we add is suitably small, say 0.000001.

$$\frac{|x|^n}{n!} \le \epsilon$$

where $\epsilon = 0.000001$.

 Thus n will be the smallest value for which the above inequality holds.



Finding the Next Term				
	to get from	to	multiply by	1
	1	×	×	1
	×	x2/2	*/2	2
	x²/2	2/6	×/3	3
	x3/L	×1/24	×/4	1

```
NextTerm = NextTerm * x/n

initialize to 1
```

```
Checking if NextTerm is too small

Using a while loop we can first check if NextTerm is less than or equal to epsilon before we add it to E.

We must use fabs(NextTerm) though since, depending on the value of x, NextTerm could be negative, and a negative number is ALWAYS less than or equal to a positive epsilon.

If we use fabs, we must include the library header, cmath

#include <cmath>
```

```
Complete Program to Approximate e<sup>x</sup>

#include <iostream>
#include <cmath> // to use fabs
/* The purpose of this program is to approximate e<sup>x</sup> by a partial series */
int main()
{
    // declare and initialize all variables
    double x, epsilon; // will be initialized by user
    double E = 1.;
    double NextTerm = 1.;
    double n = 1.;
```

```
couft << "Please enter values for x and epsilon: ";
cin >> x >> epsilon;
// echo the values that were inputted
cout << "x = " << x << " epsilon = " << epsilon << endl;
// while loop to approximate E
while (fabs(NextTerm) > epsilon)
{
    NextTerm = NextTerm * x / n; // compute the next term
    E = E + NextTerm; // update the approximation
    cout << "E = " << E << "n = " << n << endl; // output the current approx
    n++; // increase n by 1
} // end of while loop
    return 0;
} // end of main
```

```
Test Run

We need to test our program, so why don't we use a value of x = 1, so that we get an approximation for e itself.

Also use epsilon = 0.0000001

We should see E = 2.718281828, which is the value of e
```