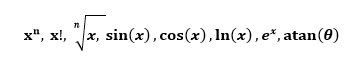
**Taylor Series: Natural Log**

**Introduction:**

Computers, microprocessors, and calculators can add, subtract, multiply, and divide. These operations are done in binary, of course, since numbers are stored in binary.

With this rather limited set of operations, how does your calculator or MATLAB determine values for these functions?



To calculate values for these functions, we need an iterative algorithm or a numerical method that only requires the basic arithmetic operations of addition, subtraction, multiplication, and division. Taylor series is one method for calculating the natural log of a number.

The Taylor Series for the ln(x) for any x in the range 0 < x < 2 is given by:

Obviously, we can’t add an infinite number of terms together, but we will use a finite number of terms to get an estimate for ln(x).

1. In order to see how this algorithm works, fill in Table 1 for x = 1.25.

**Table 1: Iterative Algorithm for ln(x)**

|  |  |
| --- | --- |
| **Initial** | **Estimate = 0** |
| **k = 1** | **Estimate = Estimate + (x – 1)1 / 1 =** |
| **k = 2** | **Estimate = Estimate – (x – 1)2 / 2 =** |
| **k = 3** | **Estimate = Estimate + (x – 1)3 / 3 =** |
| **k = 4** | **Estimate = Estimate (x – 1)4/ 4 =** |

**Note: ln(1.25) = 0.2231. The algorithm provides a pretty accurate estimate after only four iterations for this particular number, x = 1.25.**

1. Write a script file to estimate the natural log of a number which is greater than 0 and does not exceed 2 using a finite number of terms from the Taylor Series.

* Your program should first prompt the user for the number,x, and for the number of desired terms, N.
* Your program should check and see if x is an invalid number; that is, ***x is*** ***less than or equal to 0*** **or greater than 2**. Use a **while** loop for this! If the number is invalid, prompt the user to enter a new valid value for the number. The **while** loop is nice because it will continue to prompt the user until the user finally enters an acceptable value of x.
* Your program should then use a **for** loop (**for** k = 1:N) to calculate the estimate of the natural log using N terms. **Remember: in MATLAB, natural log is log.**

**Hint: Look at Table 1. Each iteration, Estimate = Estimate + New Term. The equation for the New Term changes every iteration. It obviously depends on x. See if you can figure out how to relate the equation to k (the index variable for your loop) also.**

* After the **for** loop, add an fprintf statement to display the estimate of the ln(x) with ***8 places*** behind the decimal point.

1. Test your program to make sure it doesn’t accept invalid values for x. Try negative values, zero, and values above 2.
2. Test your program using the values you computed by hand in Table 1. That is, choose x = 1.25 and try 1, 2, 3, and 4 terms.
3. Now use your program to complete Table 2.

**Table 2: Program Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **Number of Terms** | **Actual Value for ln(x)** | **Estimate for ln(x)** |
| 1.5 | 1 | 0.40546511 |  |
| 1.5 | 3 | 0.40546511 |  |
| 1.5 | 6 | 0.40546511 |  |
| 1.5 | 7 | 0.40546511 |  |
| 1.5 | 10 | 0.40546511 |  |
| 1.5 | 20 | 0.40546511 |  |

1. Now modify your script file to allow for x values larger than 2 by using the relationship:

* You will still prompt the user for a number and for the desired number of iterations
* Modify your check for invalid inputs – now the only invalid inputs are x values that are less than or equal to 0.
* Modify your code to handle x values that are greater than 2 using the relationship described above.
* Modify your fprintf statement to display the original number, x, entered by the user and the estimate of the ln(x) with 8 places behind the decimal point.

1. Now run your script to complete the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **Number of Terms** | **Actual ln(x)** | **Program Output ln(x)** |
| **4** | **4** |  |  |
| **4** | **8** |  |  |
| **4** | **12** |  |  |
| **0.4** | **4** |  |  |
| **0.4** | **8** |  |  |
| **0.4** | **12** |  |  |
| **100** | **100** |  |  |
| **100** | **200** |  |  |
| **100** | **300** |  |  |
| **100** | **400** |  |  |

**Comment: This particular algorithm requires a lot of terms to get accurate estimates for x-values close to zero and larger x-values. There are much better algorithms than Taylor series for the natural log function.**