

## Homework #1: Error Analysis and fun with Exp

1. Find the absolute error, relative error, and decimal precision (number of significant decimal digits) for the following objects  $f$  and their approximations  $\hat{f}$ .

(a)  $f = \pi$ ,  $\hat{f} = 3.14$

(b)  $f = \pi$ ,  $\hat{f} = 22/7$

(c)  $f = \log n!$ ,  $\hat{f} = n \log n - n$  for  $n = 5, 10, 100$  (This is Stirling's approximation)

(d)  $f = e^3$ ,  $\hat{f} = T_n(3)$  where  $T_n$  is the  $n$ th Taylor Polynomial approximation to  $e^x$  expanded around  $x = 0$ . Consider  $n = 1, 2, 3$ . What value of  $n$  is required for this approximation to be good to 6 digits of decimal precision?

2. Given the Taylor polynomial expansions

$$\frac{1}{1-h} = 1 + h + h^2 + h^3 + \mathbf{O}(h^4), \quad \cos(h) = 1 - \frac{h^2}{2!} + \frac{h^4}{4!} + \mathbf{O}(h^6)$$

determine the order of approximation for their sum and product.

3. Consider the function  $f(x) = \sqrt{x^2 + 1} - x$ .

(a) In infinite precision, find all real solutions  $x$  such that  $f(x) = 0$ .

(b) Now assume floating point math using a normalized floating point system with only two decimal digits of precision (i.e.  $x, f = d_1.d_2 \times 10^E$ ) and find the smallest positive value of  $x$  such that  $f(x) = 0$ . (Does your answer depend on whether you chop or round?).

(c) show that an equivalent formulation for  $f(x)$  is

$$f(x) = \frac{1}{\sqrt{x^2 + 1} + x}$$

(d) using the new formula and the  $x$  you found in part 2b, what is  $f(x)$  in two-digit floating point math?

(e) How many of these digits are significant?

4. Matlab Warmups:

(a) Write a short matlab function (errors.m) to the interface

```
function [errAbs, errRel, precision] = errors(f, fHat)
```

to calculate the absolute error, relative error and decimal precision given a true solution  $f$  and an approximate solution  $\hat{f}$ . Use this function to check your answers to problem 1.

(b) Let  $p = [ 1 \ 3 \ -2 \ -2 ]$  be a matlab vector containing the coefficients of the cubic polynomial  $P(x) = x^3 + 3x^2 - 2x - 2$ . Then  $y = \text{polyval}(p, x)$  is the matlab statement that evaluates  $P(x)$  at points contained in the vector  $x$ . (e.g. for  $x = [ 0 \ 1 \ 2 ]$  you should get  $y = [ -2 \ 0 \ 14 ]$ ).

- i. given `p`, write two lines of matlab that calculate the vector `pPrime` such that `dydx=polyval(pPrime,x)` evaluates the derivate  $P'(x)$  at point  $x$ . No loops!
- ii. check your answer at `x=[ 0 1 2]`.

5. The great Exp challenge: Consider the  $n$ th order Taylor polynomial approximation to  $e^x$

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

(a) Show that the upper bound on the *relative error*

$$r_n = \frac{|e^x - T_n(x)|}{|e^x|}$$

is given by

$$r_n \leq \left| \frac{x^{n+1}}{(n+1)!} \right|$$

(b) *Extra credit:* Show that for large  $x$  and  $n$ ,  $r_n \leq \epsilon_{\text{mach}}$  implies that we need at least  $n > ex$  terms in the series (where  $e = \exp(1)$ ) (Hint: use Stirling's approximation  $\log n! \approx n \log n - n$ ). (actually  $\sim 1000$  terms are needed to evaluate  $\exp(700)$ ).

(c) Consider the matlab code

```
n=N:-1:0;
p=1./factorial(n);
tN = polyval(p,x);
```

- i. What is the maximum number of terms  $N$ , that this algorithm can calculate? (Hint: just plug and chug in Matlab, you can't do this analytically). What sets this limit?
  - ii. For this maximum value of  $N$ , what is the relative error of this algorithm at  $x = 30$  and  $x = -30$ . What is the floating point problem for  $x < 0$ ?
- (d) Can you write a nested multiplication form for  $T_n(x)$  similar to Horner's rule, that places no limits on  $N$  in iee double precision? (it will still have problems for negative  $x$ ).
- (e) Write a matlab function, `estimateExp`, that accurately computes  $T_n$ .

Your function file should be named `estimateExp.m`, and the function should be declared:

```
function expEstimate = estimateExp(x)
```

Aron Ahmadia has kindly provided a driver function for you, `testEstimateExp.m`, which when called from the command line with `estimateExp` as a function pointer argument, will automatically compare `estimateExp` against Matlab's built-in `exp` function over a linearly spaced range of values, plot the results for you, and determine the average error of your function in ulps.

The basic call

```
>> testEstimateExp(@estimateExp)
```

will test your function for 20 points in the interval  $x \in [-20, 20]$ . To adjust the interval range and number of points use.

```
>> testEstimateExp(@estimateExp, lowerLimit, upperLimit,  
numPoints)
```

To achieve a perfect score for this question, ensure that your algorithm can achieve an average error of less than 8 ulps in the range  $[-20,20]$ . Your code documentation should reflect any assumptions you have made about the input, and tersely justify your stopping criterion. Keep it simple! This function shouldn't take much more than 15 actual lines of code or total much more than 40 lines with comments and spacing.

**Deliverable:** Turn in a print-out of your best figure from `testEstimateExp` and a copy of your function `estimateExp.m`.

*Note to the Over-Achievers: It is possible to write code that will average less than 1 ulp in the range  $[-20,20]$ , how well can you do? How well does your code perform in the range  $[-700,700]$ ?*