These are the lecture notes for CSC349A Numerical Analysis taught by Rich Little. They roughly correspond to the material covered in each lecture in the classroom but the actual classroom presentation might deviate significantly from them depending on the flow of the course delivery. They are provided as a reference to the instructor as well as supporting material for students who miss the lectures. They are simply notes to support the lecture so the text is not detailed and they are not thoroughly checked. Use at your own risk.

1 Approximations and Measuring Errors

There are several types of different erros that can arise in engineering scientific applications and it is important to understand both their sources and effects when studying numerical methods.

- **Formulation or modeling error** arises because of incomplete mathematical models. For example the simplified parachutist model we described will never exactly predict the motion of a real parachutist.
- **Data uncertainty/inherent erorr** Noisy data due to inexact measurements or observation.
- **Truncation error (Chapter 4)** results from using inexact approximations instead of an exact mathematical procedure such as the errors between the Euler method numerical procedure for predicting velocity versus the analytical solution obtained through differential calculus.
- **Roundoff error (Chapter 3)** is due to the fixed, finitite precision representations used to represent real/complex numbers in computers.

We start by discussing different ways of measuring error. If p denotes the true (exact) value of some quantity, and p^* denotes some approximation to p then the **absolute true error** is defined as:

$$|E_t| = |p - p^*| \tag{1}$$

The absolute error only makes sense if you have a sense of the magnitude of p, the quantity you are approximating. For example consider $p = 1234321$ and $p* = 1234000$, the $|E_t| = 321$ seems large, although $p*$ is quite accurate

approximations	number of correct significant digits	relative error
3.1	2	0.013
3.14	3	0.00051
3.141	4	0.00019
3.1415	5	0.000029

and agrees with p to 4 significant digits. In constrast, if $p = 0.001234$ and $p* = 0.001111$, then $|E_t| = 0.000123$ seems small, although $p*$ is not very accurate and agrees with p to only 1 significant digit.

A better approach is to introduce the concept of relative error, in which the magnitude of the exact value p is used to "normalize" the error. More specifically the **relative error** is defined as:

$$|\varepsilon_t| = \frac{|p - p^*|}{|p|} = |1 - \frac{p^*}{p}| \tag{2}$$

The *significant digits* of a number are those that can be used with confidence. It is a discrete way of measuring error in approximations in contrast to the relative error which is a continuous way of measuring error in approximation. The relative error also indicates the number of correct significant digits in an approximation p^*. For example for $p = \pi = 3.14159265\ldots$:

Frequently it is useful to relate the relative error to the number of correct significant digits. This can be done with the following: If

$$|\varepsilon_t| < 5 \times 10^{-n} \tag{3}$$

then $p*$ approximates p to n **significant digits**.

In order to calculate the relative or absolute error of a particular quantity we need to know the value of the approximation p^* as well as the true value p. In many cases we don't know the true value. In fact numerical methods in engineering are used when we don't know the true value otherwise we could just simply use it instead of computing a numeric approximation. The main reason these error measures are useful is that they can give us good estimates of how well a particular method works for the case where we have exact results increasing our confidence that they will work well for the cases for which there are no exact results.

There are many algorithms in numerical methods that operate in an iterative function in which an estimate of solution is used to compute a better estimate and the process is repeated until some convergence criterion is met.

In this cases it is common to use our best current estimate of the true value (i.e the previous estimate) to compute the error compared to the current estimate.

In this case of iterative algorithms we can use the following approximation to the relative errror:

$$|\varepsilon_a| = \frac{|p_i - p_{i-1}|}{p_i} \tag{4}$$

Similarly to the above relationship we can use the following relation to estimate the number of significant digits for the relative error in iterative algorithms: If

$$|\varepsilon_a| < 0.5 \times 10^{-n} \tag{5}$$

then p_i approximates p to *at least* n significant digits.

2 Example of Error Estimate in Iterative Methods

These concepts will hopefully become clearer through an example (Example 3.2 pages 58-59 of the 6th edition and 61-62 of the 7th edition).

Mathematicians like to represent functions by infinite series. For example the exponential function can be computed using:

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} \tag{6}$$

As more terms are added to the series the approximation becomes better and better. This is called a McLaurin series expansion. Let $p_0 = 1$, $p_1 = 1 + x$, $p_2 = 1 + x + \frac{x^2}{2!}$ and so on.

We can use this series approximation at different levels of precision to compute the value of e^x for $x = 0.5$. We can use the true value $p = e^{0.5} = 1.648721..$ to compute the true relative error $|\varepsilon_t|$.

Note that the value of $|\varepsilon_t|$, which can only be computed if we know the true (exact) answer p, can be used to estimate the number of correct significant digits in each approximation. For example: For p_2 we have $0.0144 < 5 \times 10^{-2}$ so the approximation $p_2 = 1.625$ has at least 2 correct significant digits. For p_4 we have $0.000172 < 5 \times 10^{-4}$ and therefore $p_4 = 1.6484375$ has at least 4 correct significant digits.

i	p_i	$\lvert\varepsilon_t\rvert = \frac{\lvert e^{0.5} - p_i \rvert}{\lvert e^{0.5} \rvert}$	$\lvert\varepsilon_a\rvert = \frac{\lvert p_i - p_{i-1} \rvert}{\lvert p_i \rvert}$
0	1	0.393	
1	1.5	0.0902	0.333
2	1.625	0.0144	0.0769
3	1.645833	0.00175	0.0127
4	1.6484375	0.000172	0.00158
5	1.6486979	0.0000142	0.000158

In practice, if a sequence of approximations to some unknown value is computed using an iterative algorithm, (we will use several such algorithms during this course), then the exact relative error $|\varepsilon_t|$ can not be computed. However, the relative error in each approximation p_i can be approximated by $|\varepsilon_a|$ and can be used to estimate conservatively the number of correct significant digits. For examples for i=5 in the table above we have $|\varepsilon_a| = 0.000158 < 0.5 \times 10^{-3}$ implying that $p_5 = 1.6486979$ has at least 3 correct significant digits (notice that in fact it has 4 which means this is a conservative estimate).

Armed with the syntax and concepts we learned for programming in MATLAB it is straightforward to write a MATLAB function to compute the table above showing how the true and approximate relative error decrease as more terms are added to the series. Here is the corresponding code of the function *mclaurin_exponential*:

```
function [ em ] = mclaurin_exponential( x, n )
% mclaurin_exponential:
%     Prints the true relative error and the approximate
%     relative error when approximating e with a
%     McLaurin series. The code corresponds to example
%     3.2 of the textbook and also is covered in handout 2.

    em = 1;          % the approximation
    e = exp(1);      % the true value
    et = abs(e^(x) - em) / e^(x);   % true rel error
    ea = 0;                              % approx rel error
    prev_em = 0;
    i = 0;
    fprintf('i \t pi \t\t true error \t approximation error\n');
```

```
    fprintf('%d \t %4.6f \t %4.6f \t %4.6f\n', i, em, et, ea);

    for i = 1:n
        prev_em = em;               % store prev approx
        em = em + (x^i / factorial(i));
        et = abs(e^(x) - em) / e^(x);   % true rel error
        ea = abs((em - prev_em) / em);      % approx rel error
        fprintf('%d \t %4.6f \t %4.6f \t %4.6f\n', i, em, et, ea);
    end
end
```

To produce the table we simply provide the value of x and the number of series terms to use for the approximation (or number of iterations) n.

```
>> mclaurin_exponential(0.5, 6)
i   pi    true error   approximation error
0   1.000000   0.393469   0.000000
1   1.500000   0.090204   0.333333
2   1.625000   0.014388   0.076923
3   1.645833   0.001752   0.012658
4   1.648438   0.000172   0.001580
5   1.648698   0.000014   0.000158
6   1.648720   0.000001   0.000013

ans =

   1.648719618055555
```