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 Overview

The material covered in this lecture is mostly based on handouts 3 and 4. This material consists of the floating point number representation, round off errors, and subtractive cancellation. Several examples of using idealized floating point arithmetic with ($b = 10$, $k = 4$, and chopping or rounding) were presented in class. I strongly encourage you to practice several problems using idealized floating point arithmetic to gain experience as it is frequently part of midterm and final questions.

The only additional note I would like to make that is not mentioned in the handouts is that when calculating errors in most cases the true estimate used is simply an estimate with higher precision than the one we are computing rather than the actual true value. For example when approximating π it is impossible to obtain the true value as it has an infinite number of digits. Instead we calculate the error of a a poor approximation (let's say with 4 significant digits) with one that is more accurate like the one provided by a calculator (for example 10 significant digits).

2 Round-off Error

Consider the interval between any 2 consecutive powers of the base b let's say b^{t-1} and b^t. Then the first floating point number in that interval will be $0.100\ldots0 \times b^t = b^{t-1}$. The next higher number in that representation will be $0.100\ldots1 \times b^t$ and so on until the highest number that uses b^{t-1} as the base which will be $0.(b-1)(b-1)\ldots(b-1) \times b^t$. For example in decimal that would be $0.999\ldots9 \times 10^t$.

All floating point numbers with exponent equal to t are in the interval

$[b^{t-1}, b^t)$. In this interval there are exactly $(b-1)b^{k-1}$ distinct floating point numbers and they are equally spaced.

The distance between any 2 consecutive numbers is:

$$\frac{b^t - b^{t-1}}{(b-1)b^{k-1}} = \frac{(b-1)b^{t-1}}{(b-1)b^{k-1}} = b^{t-k} \tag{1}$$

The spacing between numbers gets larger as t gets larger. In this course we will frequently be using $b = 10$ and precision $k = 4$ as it makes calculations by hand easier and more easily understood by humans.

Generally, when we represent a real number by a floating-point number of some precision k, we have to decide what k digits to use. We usually make this decision by either **chopping** all the digits to the right of the k^{th} digit or **rounding** the k^{th} digit up or down based on the value of the $(k+1)^{st}$ digit. Each of these creates a different round-off error. Let's consider chopping first.

For any real number p in interval $[b^{t-1}, b^t)$, the upper bound of the absolute true error of p^*, the floating-point representation of p with chopping, is given by,

$$|p - p^*| < b^{t-k}$$

which is the distance between any two adjacent values in the interval.

To calculate the upper bound on the relative error, we need a bound on $1/|p|$. But, since $p \in [b^{t-1}, b^t)$, we know that $p \geq b^{t-1}$ and thus $\frac{1}{p} \leq \frac{1}{b^{t-1}}$. Combining this with what we know about the absolute error we get,

$$\frac{|p - p^*|}{|p|} < \frac{b^{t-k}}{b^{t-1}} = b^{(t-k)-(t-1)} = b^{1-k}$$

This quantity b^{1-k} is called the **unit round-off** (or the **machine epsilon**). Note that it is independent of t and the magnitude of p. The number $k-1$ indicates approximately the number of significant base b digits in a floating-point approximation to a real number p.

3 Floating-point arithmetic

Floating-point arithmetic is a simulation of real arithmetic. We will use the notation *fl* to denote the floating-point representation of a real number x

as $fl(x)$ as well as the floating-point representation of arithmetic operations such as:

$$fl(a+b), fl(a-b), fl(a \times b), fl(a/b)$$

where a and b are floating-point numbers.

The implementation of these floating-point operations (in either software or hardware) depends on several factors, and includes for examples choices such as whether to use rounding or chopping and the number of significant digits used for floating-point addition and subtraction.

For simplicity, we will consider only **"idealized" floating-point arithmetic** which is defined as follows. Let $\bullet$ denote any of the basic arithmetic operations $+ - \times /$ and let x and y denote floating point numbers. $fl(x \bullet y)$ is obtained by performing **exact arithmetic** on x and y, and then **rounding or chopping** this result to k significant digits.

Note 1: although no actual digital computers or calculators implement floating-point arithmetic that way (it's too expensive as it would require a very long accumulator for doing addition and subtraction), idealized floating-point arithmetic:

- Behaves very much like any actual implementation
- Is very simple to do in hand computations
- Has accuracy almost identical to that of any implementation

Note 2: If fl is applied to an arithmetic expression containing more than one arithmetic operation, then each of the arithmetic operations must be replaced by its corresponding floating-point operation. For example:

$$fl(x+y-z) = fl(fl(x+y)-z) \tag{2}$$
$$fl(xy+z/cos(x)) = fl(fl(x \times y) + fl(z/fl(cos(x)))) \tag{3}$$

Each fl operation is computed according to the rules of idealized floating-point arithmetic, that is, the exact value of the result is rounded or chopped to k-significant digts before proceeding with the rest of the computation. Note that we will compute $fl(cos(x)), fl(\sqrt{(}x)), fl(e^x)$ and so on this way.

Note 3: with idealized floating-point arithmetic, the maximum relative error in $fl(x \bullet y)$ is the same as the maximum relative error in converting a real number z to floating-point form. Thus, for a **single** floating-point

$+ - \times /$, the **relative error is very small**: it is $< b^{1-k}$ with chopping, or $\frac{1}{2}b^{1-k}$ (with rounding). However, the relative erorr in a floating-point computation **might be large** if more than one floating-point operation is performed. For example, compute $fl(x+y+z)$ when

$$x = +0.1234 \times 10^0, \quad y = -0.5508 \times 10^{-4}, \quad z = -0.1232 \times 10^0$$

using base $b = 10$, precision $k = 4$, rounding idealized floating-point arithmetic.

$$fl(x+y) = +0.1233 \times 10^0 \quad \text{since } \ x + y = 0.12334492$$
$$fl(x+y+z) = +0.1000 \times 10^{-3} \quad \text{since } \ .1233 - .1232 = 0.0001$$

Since the exact value of $x + y + z = 0.00014492$, the relative error is:

$$\left| \frac{0.00014492 - 0.0001}{0.00014492} \right| = 0.31 \quad \text{or} \quad 31\%$$

Note, however, that this large relative error can be avoided by changing the order in which these 3 numbers are added together. Consider the evaluation of

$$fl(x+z+y) = fl(fl(x+z)+y)$$

We obtain:

$$fl(x+z) = 0.0002 \quad \text{or} \quad 0.2000 \times 10^{-3}$$

$$fl(fl(x+z)+y) = 0.1449 \times 10^{-3} \quad \text{since} \quad 0.0002 - 0.00005508 = 0.00014492,$$

which has a relative error of only 0.000138 or 0.0138%.

4 Subtractive cancellation

Subtractive cancellation refers to the loss of significant digits during a floating-point computation due to the subtraction of *nearly* equal floating-point numbers.

Note that if $\hat{x}$ is an approximation to $x > 0$ and $\hat{y}$ is an approximation to $y > 0$, and if for example $\hat{x}$ agrees with x to 8 significant digits and $\hat{y}$ agrees with y to 8 significant digits, then

$$\begin{aligned} \hat{x} \times \hat{y} &\approx x \times y \\ \hat{x}/\hat{y} &\approx x/y \\ \hat{x} + \hat{y} &\approx x + y \end{aligned}$$

will also agree to about 8 significant digits. However, this may not be true for subtraction: it is possible that none of the significant digits in $\hat{x} - \hat{y}$ and $x - y$ agree.

The following examples illustrate subtractive cancellation, and show how it can be avoided in each of these cases.

4.1 Examples: see Handout 4

Using $b = 10$, $k = 4$ idealized floating-point aritmetic evaluate each of the following expressions for the given x.

- Example 1: $fl(\sqrt{x^2+1} - x)$ at $x = 65.43$ using rounding
- Example 2: $fl(x - \sin x)$ at $x = 0.01234$ using chopping
- Example 3: $fl(1 - \sin x)$ at $x = 1.56$ radians using rounding
- Example 4: $fl(\frac{1}{2x-1} - \frac{x+2}{x-2})$ vs. $fl(\frac{-2x(x+1)}{(2x-1)(x-2)})$ for values of x near -1 and 2 using rounding