COMPUTER SCIENCE 349A

Handout Number 3

Two methods of representing a real number *p* in floating-point: **rounding** and **chopping**.

Example

Let $b = 10$, $k = 4$ and $p = 2/3$.

	floating-point approximation to p	absolute error	relative error
chopping	$+0.6666 \times 10^0$	$0.0000666\cdots$	0.0001
rounding	$+0.6667 \times 10^0$	$0.0000333\cdots$	0.00005

Note that the above absolute errors are **round-off errors** (that is, they are the difference between a real number *p* and a floating-point approximation to *p*).

Question: What is the maximum possible relative error in the *k*-digit, base *b*, floating-point representation p^* of a real number *p*?

With chopping:

Every real number *p* lies in some interval $[b^{t-1}, b^t)$ for some integer *t*.

The distance between 2 floating-point numbers in this interval is b^{t-k}.

Therefore, the absolute error satisfies

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

Thus, the relative error satisfies

$$\frac{|p - p^*|}{|p|} < \frac{b^{t-k}}{b^{t-1}} \quad \text{since} \quad p \geq b^{t-1}$$

which implies that

$$\frac{|p - p^*|}{|p|} < b^{1-k}.$$

See (3.9) on page 68 of the 6th edition (page 71 of the 7th edition). The 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*.

Example

$b = 2$, $k = 24$ (a 32 bit word)

The unit round-off is $b^{1-k} = 2^{-23} \approx 10^{-7}$, implying that in such a floating-point system, a real number has about 23 correct binary digits or 7 correct decimal digits.

With rounding

Similar to above, except now the absolute error satisfies

$$|p - p^*| \le \frac{1}{2} b^{t-k} .$$

Thus, the relative error satisfies

$$\frac{|p - p^*|}{|p|} \le \frac{0.5 b^{t-k}}{b^{t-1}} = \frac{1}{2} b^{1-k} ,$$

which is the **unit round-off** in this case. See (3.10) on page 68 of the 6^{th} ed. or page 71 of the 7^{th} ed.

FLOATING-POINT ARITHMETIC (pages 70-73 of the 6^{th} ed. or pages 73-76 of the 7^{th} ed.)

-- a simulation of real arithmetic

-- **Notation**: we'll use the symbol $f\ell$. For example, if x denotes a real number, then $f\ell(x)$ denotes its floating-point representation. Similarly, if a and b are floating-point numbers, then

$$f\ell(a+b), \quad f\ell(a-b), \quad f\ell(a \times b), \quad f\ell(a/b)$$

denote the floating-point sum, difference, product and quotient, respectively, of a and b.

The implementation of these floating-point operations (in either software or hardware) depends on several factors, and includes, for example, a choice regarding

-- rounding or chopping

-- the number of significant digits used for floating-point addition and subtraction.

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

<u>Note 1</u>: although no actual digital computers or calculators implement floating-point arithmetic this way (it's too expensive, as it requires 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, and

-- has accuracy almost identical to that of any actual implementation.

Note 2. If $f\ell$ 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,

$$f\ell(x+y-z) \text{ means } f\ell(f\ell(x+y)-z)$$

and

$$f\ell(xy+z/\cos(x)) \text{ means } f\ell(f\ell(x\times y)+f\ell(z/f\ell(\cos(x)))\,.$$

Each $f\ell$ 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 digits before proceeding with the rest of the computation. Note that we'll compute $f\ell(\cos x)$, $f\ell(\sqrt{x})$, $f\ell(e^x)$ and so on this way.

Note 3: with idealized floating-point arithmetic, the maximum relative error in $f\ell(x \bullet y)$ 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 error in a floating-point computation might be large if more than one floating-point operation is performed. For example, compute $f\ell(x+y+z)$ 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.

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

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

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

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

$$f\ell(x+z+y) = f\ell(f\ell(x+z)+y)\,.$$

We obtain

$$f\ell(x+z) = 0.0002 \text{ or } 0.2000\times 10^{-3}$$
$$f\ell(f\ell(x+z)+y) = 0.1449\times 10^{-3} \text{ since } 0.0002-0.00005508 = 0.00014492\,,$$

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