Lectures on Scientific Computing

Lise-Marie Imbert-Gerard

8/28/18

Prof. Lise-Marie Imbert-Gerard
LMIG@math.umd.edu OH. T 11-12 Th 8:30-9:30
Office: MATH 4408/CISC 4119
Grading: 40% HW 60% Final

Course Website will be updated shortly
Note: lookup research topics (numerical solutions for wave equations)
References for the course will come from Bindel & Goodman

1 Sources of Error

Chapter 2 B&G
We want to understand that there are various sources of error. When you figure out what the biggest source of error is you can focus on it to increase the accuracy of your algorithm's output. We need to understand the **likely** relative size of errors

We can't tell exactly what the error is, but we want to find out how big the error is in relation to other errors.

Overall goal is to reduce the largest source of error.

1.1 Relative and Absolute Error Cancellation

When we approximate A by $\hat{A}$ (A is actual number and $\hat{A}$ is the approximation) the absolute error is $e = \hat{A} - A \leftrightarrow \hat{A} = A + e$. The relative error is $\epsilon = \frac{e}{A} \leftrightarrow \hat{A} = A(1 + \epsilon)$.

Note: if you have a problem where the solution is zero figure that out before you run your equations to avoid dividing by 0.

example:

A	$\hat{A}$	e	ϵ
$\sqrt{175}$	13	≈ 0.23	2%
$\sqrt{5}$	2	≈ 0.24	11%

Warning: Relative errors can grow through cancellation. If you do, for example, $1.23456 - 1.2345 = 1x10^{-5}$ but you only know the accuracy up to the second digit you have no idea of the accuracy of the final digits in the solution

Formally, *Catastrophic Cancellation* refers to the loss off many digits in a substraction. Cancellation may also accumulate over many steps.

1.2 Computer Arithmetic

1. Bits
2. Floating Point Basics

Definition: In Scientific Notation a real number has three parts: a sign $\pm$, a mantissa $\epsilon(1, 10)$, and an exponent $\pm m \times 10^e$.

A normal binary floating point has three parts: a sign $(0, 1)$, a mantissa $(1, 2)$, and an exponent $(-1)^s m \times 2^e$

For example, $-2.7520 \times 10^3 = (-1)^1(1 + 2^{-2} + (2)^{-4} + 2^{-5}) \times 2^{(2^3+2^1+2^0)}$

You need to make sure that you specify the number of bits necessary to store the mantissa and the exponent. A floating point word is 1 bit for the sign and p-1 bits for the mantissa. **The machine precision is** $\epsilon_m \approx 2^{-p}$. p simply represents the total number of bits used to specify the word. Therefore a typical 32-bit binary floating point number will have $p = 24$, from 1 bit for the sign, 23 bits for the mantissa, and 8 bits for the exponent. For a 64-bit binary floating point number you have $p = 53$, from 1 bit for the sign, 52 bits for the mantissa, and 11 bits for the exponent.

The number of bits reserved for the exponent is set by the IEEE floating point standard.

The IEEE floating point standard:

1. Single: 32 bits
2. Double: 64 bits

Type	bits	p	e_m (smallest exponent)	e_M (largest exponent)	ϵ_{Mach} (machine epsilon)
Single:	32	24	-126	127	6×10^{-8}
Double:	64	53	-1022	1023	10^{-16}

The *machine epsilon* is an upper bound on the relative error due to rounding in floating point arithmetic. It can be calculated using the definition $\epsilon_{Mach} = 2^{-p}$, which is half the distance between 1 and the next floating point number.

When you have a value such as $\frac{1}{3}$ it cannot be stored exactly in a floating point system so it is rounded. Round-off error is one of the most important sources of error.

Definition: If x is a real number in a given format, we write $\hat{x} = \text{round}(x)$ which describes the floating point number closest to x in that system. If x is halfway between two closest floating point numbers the one that is picked to represent x is the one whose last bit is 0.

Finding $\hat{x}$ is called rounding and the rounding error is the error that is produced when you approximate $x \leftrightarrow \hat{x}$. There for the rounding error is defined as $\hat{x} - x$ and the relative error satisfies $|\frac{\hat{x}-x}{x}| \leq \epsilon_M$. i.e. the relative error can never be larger than the machine precision(ϵ_M).

The IEEE standard for floating point operations is to return the exact answer correctly rounded. As a consequence, multiplication and addition are individually commutative ($a + b = b + a$, $ab = ba$). Because we are doing floating point arithmetic, division by 2 is always performed exactly. This is because in binary numbers division by 2 only changes a single bit.

One important thing to keep in mind is that just because something is exact in real arithmetic doesn't mean it is exact in floating point arithmetic.

1.3 Modeling Floating Point Error

We are going to look at **Generation** and **Propagation** of rounding errors. i.e. How is it first created and how does it propagate through the calculation?

example:

$fl(x + y) = (x + y)(1 + \epsilon_1)$ with $|\epsilon_1| \leq \epsilon_M$
$fl(x + y + z) = fl(fl(x + y) + z) = ((x + y)((1 + \epsilon_2) + z)(1 + \epsilon_3)$ with $|\epsilon_2||\epsilon_3| \leq \epsilon_M$
$= (x + y + z) + (x + y)\epsilon_2 + (x + y + z)\epsilon_3 + (x + y)\epsilon_2\epsilon_3$

REMEMBER: Every time you calculate $fl(x + y)$ you get a new ϵ value.

We always have:
$\epsilon_i + \epsilon_j\epsilon_k \approx \epsilon_i$
$(1 + \epsilon_i)(1 + \epsilon_j) \approx 1 + \epsilon_i + \epsilon_j$ if $|\epsilon_i|, |\epsilon_j|, |\epsilon_k| \leq \epsilon_M$

$\sqrt{1+\epsilon_i} \approx 1 + \frac{\epsilon_i}{2}$

example:

$x = 1 - \sqrt{1-\delta}$ root $x^2 - 2x + \delta = 0$ where δ is small ($|\delta| << 1$).

We have a catastrophic cancellation case here, because $sqrt 1 - \delta$ is close to 1. First see we can approximate $x \approx \frac{\delta}{2}$.