FLOATING POINT ARITHMETHIC - ERROR ANALYSIS

- **Brief review of floating point arithmetic**
- **Model of floating point arithmetic**
- **Notation, backward and forward errors**

Roundoff errors and floating-point arithmetic

➤ The basic problem: The set A of all possible representable numbers on a given machine is finite - but we would like to use this set to perform standard arithmetic operations (+,*,-,/) on an infinite set. The usual algebra rules are no longer satisfied since results of operations are rounded.

➤ Basic algebra breaks down in floating point arithmetic.

Example: In floating point arithmetic.

$$a + (b + c) \; != \; (a + b) + c$$

✍ Matlab experiment: For 10,000 random numbers find number of instances when the above is true. Same thing for the multiplication..

Floating point representation:

Real numbers are represented in two parts: A mantissa (significand) and an exponent. If the representation is in the base β then:

$$x = \pm(.d_1 d_2 \cdots d_t)\beta^e$$

- $.d_1 d_2 \cdots d_t$ is a fraction in the base-β representation (Generally the form is normalized in that $d_1 \neq 0$), and e is an integer
- Often, more convenient to rewrite the above as:

$$x = \pm(m/\beta^t) \times \beta^e \equiv \pm m \times \beta^{e-t}$$

- Mantissa m is an integer with $0 \leq m \leq \beta^t - 1$.

Machine precision - machine epsilon

➤ Notation : $fl(x)$ = closest floating point representation of real number x ('rounding')

➤ When a number x is very small, there is a point when $1+x == 1$ in a machine sense. The computer no longer makes a difference between 1 and $1 + x$.

Machine epsilon: The smallest number ϵ such that $1 + \epsilon$ is a float that is different from one, is called machine epsilon. Denoted by $\mathbf{macheps}$ or $\mathbf{eps}$, it represents the distance from 1 to the next larger floating point number.

➤ With previous representation, $\mathbf{eps}$ is equal to $\beta^{-(t-1)}$.

$\boxed{\textit{Example:}}$ In IEEE standard double precision, $\beta = 2$, and $t = 53$ (includes 'hidden bit'). Therefore $\mathbf{eps} = 2^{-52}$.

Unit Round-off A real number x can be approximated by a floating number $fl(x)$ with relative error no larger than $\underline{\mathbf{u}} = \frac{1}{2}\beta^{-(t-1)}$.

➤ $\underline{\mathbf{u}}$ is called Unit Round-off.

➤ In fact can easily show:

$$fl(x) = x(1+\delta) \text{ with } |\delta| < \underline{\mathbf{u}}$$

✍ Matlab experiment: find the machine epsilon on your computer.

➤ Many discussions on what conditions/ rules should be satisfied by floating point arithmetic. The IEEE standard is a set of standards adopted by many CPU manufacturers.

Rule 1.

$$fl(x) = x(1+\epsilon), \quad \text{where} \quad |\epsilon| \le \underline{\mathbf{u}}$$

Rule 2. For all operations $\odot$ (one of $+, -, *, /$)

$$fl(x \odot y) = (x \odot y)(1+\epsilon_{\odot}), \quad \text{where} \quad |\epsilon_{\odot}| \le \underline{\mathbf{u}}$$

Rule 3. For $+, *$ operations

$$fl(a \odot b) = fl(b \odot a)$$

✍ Matlab experiment: Verify experimentally Rule 3 with 10,000 randomly generated numbers a_i, b_i.

Example: Consider the sum of 3 numbers: $y = a + b + c$.

- Done as $fl(fl(a + b) + c)$

$$\begin{aligned}
\eta &= fl(a + b) = (a + b)(1 + \epsilon_1) \\
y_1 &= fl(\eta + c) = (\eta + c)(1 + \epsilon_2) \\
&= \left[(a + b)(1 + \epsilon_1) + c\right](1 + \epsilon_2) \\
&= \left[(a + b + c) + (a + b)\epsilon_1)\right](1 + \epsilon_2) \\
&= (a + b + c)\left[1 + \frac{a + b}{a + b + c}\epsilon_1(1 + \epsilon_2) + \epsilon_2\right]
\end{aligned}$$

So disregarding the high order term $\epsilon_1\epsilon_2$

$$\begin{aligned}
fl(fl(a + b) + c) &= (a + b + c)(1 + \epsilon_3) \\
\epsilon_3 &\approx \frac{a + b}{a + b + c}\epsilon_1 + \epsilon_2
\end{aligned}$$

➤ If we redid the computation as $y_2 = fl(a + fl(b + c))$ we would find

$$fl(a + fl(b + c)) = (a + b + c)(1 + \epsilon_4)$$
$$\epsilon_4 \approx \frac{b + c}{a + b + c}\epsilon_1 + \epsilon_2$$

➤ The error is amplified by the factor $(a + b)/y$ in the first case and $(b + c)/y$ in the second case.

➤ In order to sum n numbers accurately, it is better to start with small numbers first. [However, sorting before adding is not worth it.]

➤ But watch out if the numbers have mixed signs!

The absolute value notation

➤ For a given vector x, $|x|$ is the vector with components $|x_i|$, i.e., $|x|$ is the component-wise absolute value of x.

➤ Similarly for matrices:

$$|A| = \{|a_{ij}|\}_{i=1,...,m;\ j=1,...,n}$$

➤ An obvious result: The basic inequality

$$|fl(a_{ij}) - a_{ij}| \leq \underline{\mathbf{u}}\ |a_{ij}|$$

translates into

$$fl(A) = A + E \qquad \text{with} \qquad |E| \leq \underline{\mathbf{u}}\ |A|$$

➤ $A \leq B$ means $a_{ij} \leq b_{ij}$ for all $1 \leq i \leq m;\ 1 \leq j \leq n$

Error Analysis: Inner product

➤ Inner products are in the innermost parts of many calculations. Their analysis is important.

Lemma: If $|\delta_i| \le \underline{\mathbf{u}}$ and $n\underline{\mathbf{u}} < 1$ then

$$\Pi_{i=1}^{n}(1+\delta_i) = 1+\theta_n \quad \text{where} \quad |\theta_n| \le \frac{n\underline{\mathbf{u}}}{1-n\underline{\mathbf{u}}}$$

➤ Common notation $\gamma_n \equiv \frac{n\underline{\mathbf{u}}}{1-n\underline{\mathbf{u}}}$

✍ Prove the lemma [Hint: use induction]

➤ Can use the following simpler result:

Lemma: If $|\delta_i| \le \underline{\mathbf{u}}$ and $n\underline{\mathbf{u}} < .01$ then

$$\Pi_{i=1}^{n}(1+\delta_i) = 1 + \theta_n \quad \text{where} \quad |\theta_n| \le 1.01 n \underline{\mathbf{u}}$$

Example: Previous sum of numbers can be written

$$\begin{aligned} fl(a+b+c) &= a(1+\epsilon_1)(1+\epsilon_2) \\ &\quad + b(1+\epsilon_1)(1+\epsilon_2) \;+\; c(1+\epsilon_2) \\ &= a(1+\theta_1) + b(1+\theta_2) + c(1+\theta_3) \\ &= \text{exact sum of slightly perturbed inputs,} \end{aligned}$$

where all θ_i's satisfy $|\theta_i| \le 1.01 n \underline{\mathbf{u}}$ (here $n = 2$).

➤ Alternatively, can write 'forward' bound:

$$|fl(a+b+c) - (a+b+c)| \le |a\theta_1| + |b\theta_2| + |c\theta_3|.$$

Backward and forward errors

- Assume the approximation $\hat{y}$ to $y = \text{alg}(x)$ is computed by some algorithm with arithmetic precision ϵ. Possible analysis: find an upper bound for the Forward error

$$|\Delta y| = |y - \hat{y}|$$

- This is not always easy.

Alternative question: find equivalent perturbation on initial data (x) that produces the result $\hat{y}$. In other words, find Δx so that:

$$\text{alg}(x + \Delta x) = \hat{y}$$

- The value of $|\Delta x|$ is called the backward error. An analysis to find an upper bound for $|\Delta x|$ is called Backward error analysis.

TB: 13-15; GvL 2.7; Ort 9.2; AB: 1.4.1–.2 – Float

Example:

$$A = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \quad B = \begin{pmatrix} d & e \\ 0 & f \end{pmatrix}$$

Consider the product: $fl(A.B) =$

$$\left[\begin{array}{c|c} (ad)(1+\epsilon_1) & [ae(1+\epsilon_2) + bf(1+\epsilon_3)]\,(1+\epsilon_4) \\ \hline 0 & cf(1+\epsilon_5) \end{array}\right]$$

with $\epsilon_i \le \underline{\mathbf{u}}$, for $i = 1, ..., 5$. Result can be written as:

$$\left[\begin{array}{c|c} a & b(1+\epsilon_3)(1+\epsilon_4) \\ \hline 0 & c(1+\epsilon_5) \end{array}\right] \left[\begin{array}{c|c} d(1+\epsilon_1) & e(1+\epsilon_2)(1+\epsilon_4) \\ \hline 0 & f \end{array}\right]$$

- So $fl(A.B) = (A + E_A)(B + E_B)$.
- Backward errors E_A, E_B satisfy:

$$|E_A| \le 2\underline{\mathbf{u}}\,|A| + O(\underline{\mathbf{u}}^{\,2}) \ ; \qquad |E_B| \le 2\underline{\mathbf{u}}\,|B| + O(\underline{\mathbf{u}}^{\,2})$$

➤ When solving $Ax = b$ by Gaussian Elimination, we will see that a bound on $\|e_x\|$ such that this holds exactly:

$$A(x_{\text{computed}} + e_x) = b$$

is much harder to find than bounds on $\|E_A\|$, $\|e_b\|$ such that this holds exactly:

$$(A + E_A)x_{\text{computed}} = (b + e_b).$$

Note: In many instances backward errors are more meaningful than forward errors: if initial data is accurate only to 4 digits say, then my algorithm for computing x need not guarantee a backward error of less then 10^{-10} for example. A backward error of order 10^{-4} is acceptable.

Main result on inner products:

- Backward error expression:

$$fl(x^Ty) = [x \mathbin{.*} (1+d_x)]^T [y \mathbin{.*} (1+d_y)]$$

where $\|d_\square\|_\infty \le 1.01 n \underline{\mathbf{u}}$, $\square = x, y$.

- Can show equality valid even if one of the d_x, d_y absent.

- Forward error expression: $|fl(x^Ty) - x^Ty| \le \gamma_n \; |x|^T \; |y|$

with $0 \le \gamma_n \le 1.01 n \underline{\mathbf{u}}$.

- Elementwise absolute value $|x|$ and multiply $.*$ notation.

- Above assumes $n\underline{\mathbf{u}} \le .01$.
  For $\underline{\mathbf{u}} = 2.0 \times 10^{-16}$, this holds for $n \le 4.5 \times 10^{13}$.

- Consequence of lemma:

$$|fl(A * B) - A * B| \leq \gamma_n \ |A| * |B|$$

- Another way to write the result (less precise) is

$$|fl(x^T y) - x^T y| \leq \ n \ \underline{\mathbf{u}} \ \ |x|^T \ |y| + O(\underline{\mathbf{u}}^{\,2})$$

✍ Assume you use single precision for which you have $\underline{\mathbf{u}} = 2. \times 10^{-6}$. What is the largest n for which $n\underline{\mathbf{u}} \leq 0.01$ holds? Any conclusions for the use of single precision arithmetic?

✍ What does the main result on inner products imply for the case when $y = x$? [Contrast the relative accuracy you get in this case vs. the general case when $y \neq x$]

✍ Show for any x, y, there exist $\Delta x, \Delta y$ such that

$$fl(x^T y) = (x + \Delta x)^T y, \quad \text{with} \quad |\Delta x| \le \gamma_n |x|$$
$$fl(x^T y) = x^T (y + \Delta y), \quad \text{with} \quad |\Delta y| \le \gamma_n |y|$$

✍ (Continuation) Let A an $m \times n$ matrix, x an n-vector, and $y = Ax$. Show that there exist a matrix ΔA such

$$fl(y) = (A + \Delta A)x, \quad \text{with} \quad |\Delta A| \le \gamma_n |A|$$

✍ (Continuation) From the above derive a result about a column of the product of two matrices A and B. Does a similar result hold for the product AB as a whole?

TB: 13-15; GvL 2.7; Ort 9.2; AB: 1.4.1–.2 – Float

Supplemental notes: Floating Point Arithmetic

In most computing systems, real numbers are represented in two parts: A mantissa and an exponent. If the representation is in the base β then:

$$x = \pm(.d_1 d_2 \cdots d_m)_\beta \beta^e$$

- $.d_1 d_2 \cdots d_m$ is a fraction in the base-β representation
- e is an integer - can be negative, positive or zero.
- Generally the form is normalized in that $d_1 \neq 0$.

Example: In base 10 (for illustration)

1. 1000.12345 can be written as

$$\mathbf{0.100012345_{10} \times 10^4}$$

2. 0.000812345 can be written as

$$\mathbf{0.812345_{10} \times 10^{-3}}$$

➤ Problem with floating point arithmetic: we have to live with limited precision.

Example: Assume that we have only 5 digits of accuray in the mantissa and 2 digits for the exponent (excluding sign).

$.d_1$	d_2	d_3	d_4	d_5	e_1	e_2

Try to add $1000.2 = .10002e+03$ and $1.07 = .10700e+01$:

$$\mathbf{1000.2} = \boxed{.1\,|\,0\,|\,0\,|\,0\,|\,2\,\|\,0\,|\,4} \; ; \qquad \mathbf{1.07} = \boxed{.1\,|\,0\,|\,7\,|\,0\,|\,0\,\|\,0\,|\,1}$$

First task: align decimal points. The one with smallest exponent will be (internally) rewritten so its exponent matches the largest one:

$$\mathbf{1.07 = 0.000107\ \times 10^4}$$

Second task: add mantissas:

$$\begin{array}{rl} & 0.\ 1\ 0\ 0\ 0\ 2 \\ + & 0.\ 0\ 0\ 0\ 1\ 0\ 7 \\ \hline = & 0.\ 1\ 0\ 0\ 1\ 2\ 7 \end{array}$$

TB: 13; GvL 2.7; Ort 9.2; AB: 1.4.1–. – FloatSuppl

Third task:

round result. Result has 6 digits - can use only 5 so we can

- Chop result: | .1 | 0 | 0 | 1 | 2 | ;
- Round result: | .1 | 0 | 0 | 1 | 3 | ;

Fourth task:

Normalize result if needed (not needed here)

result with rounding: | .1 | 0 | 0 | 1 | 3 || 0 | 4 | ;

✍ Redo the same thing with 7000.2 + 4000.3 or 6999.2 + 4000.3.

Some More Examples

➤ Each operation $fl(x \odot y)$ proceeds in 4 steps:

1. Line up exponents (for addition & subtraction).
2. Compute temporary exact answer.
3. Normalize temporary result.
4. Round to nearest representable number (round-to-even in case of a tie).

	.40015 e+02	.40010 e+02	.41015 e-98
+	.60010 e+02	.50001 e-04	-.41010 e-98
temporary	1.00025 e+02	.400105001e+02	.00005 e-98
normalize	.100025e+03	.400105$\oplus$ e+02	.00050 e-99
round	.10002 e+03	.40011 e+02	.00050 e-99
note:	round to even	round to nearest $\oplus$=not all 0's	too small: unnormalized
	exactly halfway between values	closer to upper value	exponent is at minimum

TB: 13; GvL 2.7; Ort 9.2; AB: 1.4.1–. – FloatSuppl

The IEEE standard

$\boxed{\textit{32 bit}}$ (Single precision) :

$\pm$	8 bits	$\leftarrow$ 23 bits $\rightarrow$
sign	exponent	mantissa

➤ In binary: The leading one in mantissa does not need to be represented. One bit gained. ➤ Hidden bit.

➤ Largest exponent: $\mathbf{2^7 - 1 = 127}$; Smallest: $= \mathbf{-126}$. ['bias' of 127]

64 bit (Double precision) :

$\pm$	11 bits	$\leftarrow$ 52 bits $\rightarrow$
sign	exponent	mantissa

- Bias of 1023 so if c is the contents of exponent field actual exponent value is 2^{c-1023}

- $e + bias = 2047$ (all ones) $=$ special use

- Largest exponent: $\mathbf{1023}$; Smallest $=$ -1022.

- Including the hidden bit, mantissa has total of 53 bits (52 bits represented, one hidden).

Take the number 1.0 and see what will happen if you add $\mathbf{1/2, 1/4, \ldots, 2^{-i}}$. Do not forget the hidden bit!

Hidden bit (Not represented)

Expon. $\downarrow$ $\leftarrow$ 52 bits $\rightarrow$

e	1	1	0	0	0	0	0	0	0	0	0	0
e	1	0	1	0	0	0	0	0	0	0	0	0
e	1	0	0	1	0	0	0	0	0	0	0	0

....

e	1	0	0	0	0	0	0	0	0	0	0	1
e	1	0	0	0	0	0	0	0	0	0	0	0

(Note: The 'e' part has 12 bits and includes the sign)

- Conclusion

$$\boldsymbol{fl(1+2^{-52}) \neq 1} \text{ but: } \boldsymbol{fl(1+2^{-53}) == 1} \text{ !!}$$

Special Values

- Exponent field = 00000000000 (smallest possible value)
  No hidden bit. All bits == 0 means exactly zero.
- Allow for unnormalized numbers,
  leading to gradual underflow.
- Exponent field = 11111111111 (largest possible value)
  Number represented is "Inf" "-Inf" or "NaN".

Appendix to set 3: Analysis of inner products

Consider $$s_n = fl(x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n)$$

- In what follows η_i's comme from $*$, ϵ_i's comme from $+$
- They satisfy: $|\eta_i| \le \underline{\mathbf{u}}$ and $|\epsilon_i| \le \underline{\mathbf{u}}$.
- The inner product s_n is computed as:

1. $s_1 = fl(x_1y_1) = (x_1y_1)(1+\eta_1)$
2. $$\begin{aligned} s_2 &= fl(s_1 + fl(x_2y_2)) = fl(s_1 + x_2y_2(1+\eta_2)) \\ &= (x_1y_1(1+\eta_1) + x_2y_2(1+\eta_2))\,(1+\epsilon_2) \\ &= x_1y_1(1+\eta_1)(1+\epsilon_2) + x_2y_2(1+\eta_2)(1+\epsilon_2) \end{aligned}$$
3. $$\begin{aligned} s_3 &= fl(s_2 + fl(x_3y_3)) = fl(s_2 + x_3y_3(1+\eta_3)) \\ &= (s_2 + x_3y_3(1+\eta_3))(1+\epsilon_3) \end{aligned}$$

Expand:
$$\begin{aligned} s_3 = \; & x_1 y_1 (1+\eta_1)(1+\epsilon_2)(1+\epsilon_3) \\ & + x_2 y_2 (1+\eta_2)(1+\epsilon_2)(1+\epsilon_3) \\ & + x_3 y_3 (1+\eta_3)(1+\epsilon_3) \end{aligned}$$

➤ Induction would show that [with convention that $\epsilon_1 \equiv 0$]

$$s_n = \sum_{i=1}^{n} x_i y_i (1+\eta_i) \prod_{j=i}^{n} (1+\epsilon_j)$$

Q: How many terms in the coefficient of $x_i y_i$ do we have?

A:
- When $i > 1$: $1 + (n - i + 1) = n - i + 2$
- When $i = 1$: n (since $\epsilon_1 = 0$ does not count)

➤ Bottom line: always $\leq n$.

- For each of these products

$$(1+\eta_i)\ \prod_{j=i}^{n}(1+\epsilon_j) = 1+\theta_i, \quad \text{with} \quad |\theta_i| \le \gamma_n \mathbf{u} \quad \text{so:}$$

$$s_n = \sum_{i=1}^{n} x_i y_i (1+\theta_i) \quad \text{with} \quad |\theta_i| \le \gamma_n \quad \text{or:}$$

$$\boxed{fl\left(\sum_{i=1}^{n} x_i y_i\right) = \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i y_i \theta_i \quad \text{with} \quad |\theta_i| \le \gamma_n}$$

- This leads to the final result (forward form)

$$\left| fl\left(\sum_{i=1}^{n} x_i y_i\right) - \sum_{i=1}^{n} x_i y_i \right| \le \gamma_n \sum_{i=1}^{n} |x_i||y_i|$$

- or (backward form)

$$fl\left(\sum_{i=1}^{n} x_i y_i\right) = \sum_{i=1}^{n} x_i y_i (1+\theta_i) \quad \text{with} \quad |\theta_i| \le \gamma_n$$