Floating point arithmetic - 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, \( a + (b + c) \neq (a + b) + c \)

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

Machine precision - machine epsilon

- Notation: \( \text{fl}(x) = \) 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 \( \epsilon \) such that \( 1 + \epsilon \) is a float that is different from one, is called machine epsilon. Denoted by \text{macheps} or \( \epsilon \), it represents the distance from 1 to the next larger floating point number.

- With previous representation, \( \epsilon \) is equal to \( \beta^{-(t-1)} \).
In IEEE standard double precision, \( \beta = 2 \), and \( t = 53 \) (includes ‘hidden bit’). Therefore \( \varepsilon_p = 2^{-52} \).

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

- \( u \) is called Unit Round-off.
- In fact, can easily show:
  \[
  \text{fl}(x) = x(1 + \delta) \text{ with } |\delta| < u
  \]

**Example:** Consider the sum of 3 numbers: \( y = a + b + c \).

- Done as \( \text{fl}(\text{fl}(a + b) + c) \)
  \[
  \begin{align*}
  \eta &= \text{fl}(a + b) = (a + b)(1 + \varepsilon_1) \\
  y_1 &= \text{fl}(\eta + c) = (\eta + c)(1 + \varepsilon_2) \\
  &= [(a + b)(1 + \varepsilon_1) + c](1 + \varepsilon_2) \\
  &= [(a + b + c) + (a + b)\varepsilon_1](1 + \varepsilon_2) \\
  &= (a + b + c)
  \left[ 1 + \frac{a + b}{a + b + c}\varepsilon_1 + \varepsilon_2 \right]
  \end{align*}
  \]

So disregarding the high order term \( \varepsilon_1\varepsilon_2 \)

\[
\text{fl}(\text{fl}(a + b) + c) = (a + b + c)(1 + \varepsilon_3) \quad \varepsilon_3 \approx \frac{a + b}{a + b + c}\varepsilon_1 + \varepsilon_2
\]

**Rule 1.**

\[
\text{fl}(x) = x(1 + \varepsilon), \quad \text{where } |\varepsilon| \leq u
\]

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

\[
\text{fl}(x \odot y) = (x \odot y)(1 + \varepsilon_\odot), \quad \text{where } |\varepsilon_\odot| \leq u
\]

**Rule 3.** For \(+, \ast\) operations

\[
\text{fl}(a \odot b) = \text{fl}(b \odot a)
\]

**Matlab experiment:** Verify experimentally Rule 3 with 10,000 randomly generated numbers \( a_i, b_i \).

- If we redid the computation as \( y_2 = \text{fl}(a + \text{fl}(b + c)) \) we would find

\[
\text{fl}(a + \text{fl}(b + c)) = (a + b + c)(1 + \varepsilon_4)
\]

\[
\varepsilon_4 \approx \frac{b + c}{a + b + c}\varepsilon_1 + \varepsilon_2
\]

- 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
  \[
  |fl(a_{ij}) - a_{ij}| \leq u |a_{ij}|
  \]
  translates into
  \[
  fl(A) = A + E \quad \text{with} \quad |E| \leq u |A|
  \]

- \( A \leq B \) means \( a_{ij} \leq b_{ij} \) for all \( 1 \leq i \leq m; \ 1 \leq j \leq n \)

Backward and forward errors

- Assume the approximation \( \hat{y} \) to \( y = \text{alg}(x) \) is computed by some algorithm with arithmetic precision \( \varepsilon \). 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 \( \Delta x \) 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.

Example:

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

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

\[
\begin{bmatrix}
ad(1 + \epsilon_1) & [ae(1 + \epsilon_2) + bf(1 + \epsilon_3)](1 + \epsilon_4) \\
0 & cf(1 + \epsilon_5)
\end{bmatrix}
\]

with \( \epsilon_i \leq u \), for \( i = 1, ..., 5 \). Result can be written as:

\[
\begin{bmatrix}
a & b(1 + \epsilon_3)(1 + \epsilon_4) \\
0 & c(1 + \epsilon_5)
\end{bmatrix} \begin{bmatrix}
d(1 + \epsilon_1) & e(1 + \epsilon_2)(1 + \epsilon_4) \\
0 & f
\end{bmatrix}
\]

- So \( fl(A.B) = (A + E_A)(B + E_B) \).
- Backward errors \( E_A, E_B \) satisfy:
  \[
  |E_A| \leq 2u |A| + O(u^2) ; \quad |E_B| \leq 2u |B| + O(u^2)
  \]

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.
**Error Analysis: Inner product**

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

**Lemma:** If $|\delta_i| \leq u$ and $nu < 1$ then

$$\Pi_{i=1}^n (1 + \delta_i) = 1 + \theta_n \quad \text{where} \quad |\theta_n| \leq \frac{nu}{1 - nu}$$

- Common notation $\gamma_n \equiv \frac{nu}{1 - nu}$

Prove the lemma [Hint: use induction]

---

**Analysis of inner products (cont.)**

Consider

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

- In what follows $\eta_i$'s come from $\ast$, $\epsilon_i$'s come from $+$
- They satisfy: $|\eta_i| \leq u$ and $|\epsilon_i| \leq u$.
- The inner product $s_n$ is computed as:

1. $s_1 = fl(x_1 y_1) = (x_1 y_1)(1 + \eta_1)$
2. $s_2 = fl(s_1 + fl(x_2 y_2)) = fl(s_1 + x_2 y_2(1 + \eta_2))$
   $$= (x_1 y_1(1 + \eta_1) + x_2 y_2(1 + \eta_2))(1 + \epsilon_2)$$
   $$= x_1 y_1(1 + \eta_1)(1 + \epsilon_2) + x_2 y_2(1 + \eta_2)(1 + \epsilon_2)$$
3. $s_3 = fl(s_2 + fl(x_3 y_3)) = fl(s_2 + x_3 y_3(1 + \eta_3))$
   $$= (s_2 + x_3 y_3(1 + \eta_3))(1 + \epsilon_3)$$

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 $\leq n$. 

Can use the following simpler result:

**Lemma:** If $|\delta_i| \leq u$ and $nu < 0.01$ then

$$\Pi_{i=1}^n (1 + \delta_i) = 1 + \theta_n \quad \text{where} \quad |\theta_n| \leq 1.01nu$$

**Example:** Previous sum of numbers can be written

$$fl(a + b + c) = a(1 + \epsilon_1)(1 + \epsilon_2) + b(1 + \epsilon_1)(1 + \epsilon_2) + c(1 + \epsilon_2)$$

$$= a(1 + \theta_1) + b(1 + \theta_2) + c(1 + \theta_3)$$

exact sum of slightly perturbed inputs,

where all $\theta_i$'s satisfy $|\theta_i| \leq 1.01nu$ (here $n = 2$).

Alternatively, can write 'forward' bound:

$$|fl(a + b + c) - (a + b + c)| \leq |a\theta_1| + |b\theta_2| + |c\theta_3|.$$
For each of these products
\[(1 + \eta_i) \prod_{j=i}^{n}(1 + \epsilon_j) = 1 + \theta_i, \text{ with } |\theta_i| \leq \gamma_n u \]
so:
\[s_n = \sum_{i=1}^{n} x_i y_i (1 + \theta_i) \text{ with } |\theta_i| \leq \gamma_n \]
or:
\[\sum_{i=1}^{n} x_i y_i (1 + \theta_i) \text{ with } |\theta_i| \leq \gamma_n \]

This leads to the final result (forward form)
\[fl \left( \sum_{i=1}^{n} x_i y_i \right) - \sum_{i=1}^{n} x_i y_i \leq \gamma_n \sum_{i=1}^{n} |x_i| |y_i| \]

or (backward form)
\[fl \left( \sum_{i=1}^{n} x_i y_i \right) = \sum_{i=1}^{n} x_i y_i (1 + \theta_i) \text{ with } |\theta_i| \leq \gamma_n \]

Consequence of lemma:
\[|fl(A \ast B) - A \ast B| \leq \gamma_n |A| \ast |B| \]

Another way to write the result (less precise) is
\[|fl(x^T y) - x^T y| \leq n u |x|^T |y| + O(u^2) \]

Main result on inner products:

Backward error expression:
\[fl(x^T y) = [x \ast (1 + d_x)]^T[y \ast (1 + d_y)] \]

where \( \|d\|_\infty \leq 1.01 n u \), \( \Box = x, y \).

Can show equality valid even if one of the \( d_x, d_y \) absent.

Forward error expression:
\[|fl(x^T y) - x^T y| \leq \gamma_n |x|^T |y| \]

with \( 0 \leq \gamma_n \leq 1.01 n u \).

Elementwise absolute value \(|x|\) and multiply \( \ast \) notation.

Above assumes \( n u \leq .01 \).
For \( u = 2.0 \times 10^{-16} \), this holds for \( n \leq 4.5 \times 10^{13} \).

Assume you use single precision for which you have \( u = 2. \times 10^{-6} \). What is the largest \( n \) for which \( n u \leq .01 \) 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
\[
fl(x^T y) = (x + \Delta x)^T y, \quad \text{with} \quad |\Delta x| \leq \gamma_n |x|
\]
\[
fl(x^T y) = x^T (y + \Delta y), \quad \text{with} \quad |\Delta y| \leq \gamma_n |y|
\]

(Continuation) Let \( A \) an \( m \times n \) matrix, \( x \) an \( n \)-vector, and \( y = Ax \). Show that there exist a matrix \( \Delta A \) such
\[
fl(y) = (A + \Delta A)x, \quad \text{with} \quad |\Delta A| \leq \gamma_n |A|
\]

(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?

The computed solution \( \hat{x} \) of the triangular system \( Ux = b \) computed by the back-substitution algorithm satisfies:
\[
(U + E)\hat{x} = b
\]
with
\[
|E| \leq n \ u \ |U| + O(u^2)
\]

- Backward error analysis. Computed \( x \) solves a slightly perturbed system.
- Backward error not large in general. It is said that triangular solve is “backward stable”.

For no zero pivots are encountered during Gaussian elimination (no pivoting) then the computed factors \( \hat{L} \) and \( \hat{U} \) satisfy
\[
\hat{L}\hat{U} = A + H
\]
with
\[
|H| \leq 3(n - 1) \times u \left( |A| + |\hat{L}| |\hat{U}| \right) + O(u^2)
\]
Solution \( \hat{x} \) computed via \( \hat{L}\hat{y} = b \) and \( \hat{U}\hat{x} = \hat{y} \) is s. t.
\[
(A + E)\hat{x} = b \quad \text{with}
\]
\[
|E| \leq n u \left( 3|A| + 5 |\hat{L}| |\hat{U}| \right) + O(u^2)
\]
“Backward” error estimate.

| \( |\hat{L}| \) and |\( \hat{U} | \) are not known in advance – they can be large.

What if partial pivoting is used?

Permutations introduce no errors. Equivalent to standard LU factorization on matrix \( PA \).

| \( |\hat{L}| \) is small since \( l_{ij} \leq 1 \). Therefore, only \( U \) is “uncertain”

In practice partial pivoting is “stable” – i.e., it is highly unlikely to have a very large \( U \).