LECTURE 4

Floating Point Error Analysis

We now state two theorems regarding the propagation of round off errors for sums and products.

0.1. Round-off errors induced by machine addition.

THEOREM 4.1. *Let $x_0, x_1, \ldots, x_n$ be positive machine numbers in a computer whose unit roundoff error is ε. Then the relative roundoff error of the sum*

$$\sum_{k=0}^{n} x_k$$

is at most $(1+\varepsilon)^n - 1 \approx n\varepsilon$.

Proof. Suppose we try to compute a sum $x_0 + x_1 + \cdots + x_n$ of machine numbers. This would be carried out iteratively by first computing $x_0 + x_1$, rounding off to $fl\,(x_0 + x_1)$ and then computing $fl\,(x_0 + x_1) + x_2$ and rounding off to $fl\,(fl\,(x_0 + x_1) + x_2)$, etc. To see how the round off errors propagate let

$$S_k = x_0 + x_1 + \cdots + x_k$$

be the exact k^{th} partial sum and let

$$S_k^* = fl\,(S_k^* + x_k)$$

be the machine-computed k^{th} partial sum. Define

$$\begin{aligned} \rho_k &= \frac{S_k^* - S_k}{S_k} \\ \delta_k &= \frac{S_{k+1}^* - (S_k^* + x_{k+1})}{S_k^* + x_{k+1}} \end{aligned}$$

ρ_k is just the relative error at the k^{th} stage of the calculation, while δ_k is the round-off error incurred at the next step. Our hypothesis is that $|\delta_k|$ is always $\leq \varepsilon$. Using

$$\begin{aligned} S_{k+1}^* &= (S_k^* + x_{k+1})\,\delta_k + S_k^* + x_{k+1} = (S_k^* + x_{k+1})\,(1 + \delta_k) \\ S_k^* &= S_k \rho_k + S_k \\ S_{k+1} &= S_k + x_{k+1} \end{aligned}$$

we have

$$\begin{aligned}
\rho_{k+1} &= \frac{S_{k+1}^* - S_{k+1}}{S_{k+1}} \\
&= \frac{(S_k^* + x_{k+1})(1+\delta_k) - (S_k + x_{k+1})}{S_{k+1}} \\
&= \frac{(S_k\rho_k + S_k + x_{k+1})(1+\delta_k) - (S_k + x_{k+1})}{S_{k+1}} \\
&= \frac{(S_k(1+\rho_k) + x_{k+1})(1+\delta_k) - (S_k + x_{k+1})}{S_{k+1}} \\
&= \frac{S_k + S_k\rho_k + x_{k+1} + (S_k + S_k\rho_k + x_{k+1})\delta_k - S_k - x_{k+1}}{S_{k+1}} \\
&= \frac{S_k\rho_k + (S_k + S_k\rho_k + x_{k+1})\delta_k}{S_{k+1}} \\
&= \frac{S_k\rho_k + (S_{k+1} + S_k\rho_k)\delta_k}{S_{k+1}} \\
&= \delta_k + \frac{S_k}{S_{k+1}}(1+\delta_k)\rho_k
\end{aligned}$$

Since $|\delta_k| \le \varepsilon$ and $S_k/S_{k+1} \le 1$ we can conclude

$$|\rho_{k+1}| \le \varepsilon + (1+\varepsilon)\rho_k$$

Setting $\theta = 1+\varepsilon$, we write this as

$$|\rho_{k+1}| \le \varepsilon + \theta\rho_k$$

We now iterate this formula starting with $\rho_0 = 0$:

$$\begin{aligned}
|\rho_0| &= 0 \\
|\rho_1| &= \varepsilon + \theta\cdot 0 = \varepsilon \\
|\rho_2| &= \varepsilon + \theta(\varepsilon) = \varepsilon + \theta\varepsilon \\
|\rho_3| &= \varepsilon + \theta(\varepsilon + \theta\varepsilon) = \varepsilon + \varepsilon\theta + \varepsilon\theta^2 \\
&\vdots \\
|\rho_n| &= \varepsilon + \varepsilon\theta + \cdots + \varepsilon\theta^{n-1} \\
&= \varepsilon\left(1 + \theta + \cdots + \theta^{n-1}\right) \\
&= \varepsilon\left(\frac{\theta^n - 1}{\theta - 1}\right) \\
&= \varepsilon\frac{(1+\varepsilon)^n - 1}{\varepsilon} \\
&= (1+\varepsilon)^n - 1
\end{aligned}$$

□

0.2. Subtraction of Nearly Equal Quanities. Another, but often avoidable, means of introducing large relative errors is by computing the difference between two nearly equal floating point numbers.

For example, let

$$\begin{aligned}
x &= 0.3721478693 \\
y &= 0.3720230572 \\
x - y &= 0.0001248121
\end{aligned}$$

and suppose that the difference $x - y$ is computed on a decimal computer allowing a 5-digit mantissa (i.e., 5 significant figures)

$$\begin{aligned} fl(x) &= 0.37215 \\ fl(y) &= 0.37202 \end{aligned}$$

two numbers, each with 5 significant digits. When the machine computes the difference between $fl(x)$ and $fl(y)$ it obtains

$$fl(x) - fl(y) = 0.00013 = 1.3 \times 10^4$$

a number with only two significant digits. The relative error is thus fairly large

$$\left| \frac{x - y - [fl(x) - fl(y)]}{x = y} \right| = \left| \frac{0.000124812 - 0.00013}{0.000124812} \right| \approx 4\%$$

The following theorem gives bounds on the relative error that can be introduced by such subtraction errors.

THEOREM 4.2. ***(Loss of Precision Theorem.)*** *If x and y are positive normalized binary machine numbers such that $x > y$, and*

$$2^{-q} \leq 1 - \frac{y}{x} \leq 2^{-p}$$

then at most q and at least p significant binary digits will be lost in the subtraction $x - y$.

Proof. Let

$$\begin{aligned} x &= r \times 2^n \quad , \quad 1 \leq r < 2 \\ y &= s \times 2^m \quad , \quad 1 \leq s < 2 \end{aligned}$$

be the normalized binary floating point forms for x and y. Since x is larger than y, $m \leq n$. In order to carry out the subtraction, a computer will first have to shift the floating point of y so that the machine representations of x and y have the same number of decimal places; effectively writing y as

$$y = \left(s \times 2^{m-n}\right) \times 2^n$$

We then have

$$\times - y = \left(r - s \times 2^{m-n}\right) \times 2^n$$

The mantissa of this expression satisfies

$$r - s \times 2^{m-n} = r \left(1 - \frac{s \times 2^m}{r \times 2^n}\right) = r \left(1 - \frac{y}{x}\right) < 2^{-p}$$

To normalize this expression, expression then a shift of at least p digits is required. This means that at least p bits of precision have been lost. On the other hand, since the new mantissa also satisfies

$$r - s \times 2^{m-n} = r \left(1 - \frac{y}{x}\right) > 2^{-q}$$

then a shift of no more than q digits will be necessary to put the mantissa in standard form. Thus, at most q bits of precision have been lost.