

Notes on floating point number, numerical computations and pitfalls*

November 16, 2010

1 Floating point numbers

An n -digit floating point number in base β has the form

$$x = \pm(0.d_1d_2\cdots d_n)_\beta \times \beta^e$$

where $0.d_1d_2\cdots d_n$ is a β -fraction called the *mantissa* and e is an integer called the *exponent*. Such a floating point number is called *normalized* if $d_1 \neq 0$, or else, $d_1 = d_2 = \cdots = d_n = 0$. The exponent e is limited to a range

$$m < e < M$$

Usually, $m = -M$.

1.1 Rounding, chopping, overflow and underflow

There are two common ways of translating a given real number x in to an n β -digit floating point number $fl(x)$ - *rounding* and *chopping*.

For example, if two decimal digit floating point numbers are used

$$fl(2/3) = \begin{cases} (0.67) \times 10^0 & \text{rounded} \\ (0.66) \times 10^0 & \text{chopped} \end{cases}$$

and

$$fl(-838) = \begin{cases} -(0.84) \times 10^3 & \text{rounded} \\ -(0.83) \times 10^3 & \text{chopped} \end{cases}$$

The conversion is undefined if $|x| \geq \beta^M$ (*overflow*) or $0 < |x| \geq \beta^{m-n}$ (*underflow*), where m and M are the bounds on the exponent.

*Reference: Elementary numerical analysis by Samuel D. Conte and Carl de Boor

The difference between x and $fl(x)$ is called the *round-off error*. If we write

$$fl(x) = x(1 + \delta)$$

then it is possible to bound δ independently of x . It is not difficult to see that

$$\begin{aligned} |\delta| &< \frac{1}{2}\beta^{1-n} && \text{in rounding} \\ -\beta^{1-n} &< \delta \leq 0 && \text{in chopping} \end{aligned}$$

The maximum possible value of $|\delta|$ is often called the *unit roundoff* or *machine epsilon* and is denoted by \mathbf{u} . It is the smallest machine number such that

$$fl(1 + \mathbf{u}) > 1$$

1.2 Floating point operations

If \odot denotes one of the arithmetic operations (addition, subtraction, multiplication or division) and \odot^* represents the floating point version of the same operation, then, usually

$$x \odot y \neq x \odot^* y$$

However, it is reasonable to assume (actually the computer arithmetic is so constructed) that

$$x \odot^* y = fl(x \odot y) = (x \odot y)(1 + \delta)$$

for some δ , with $|\delta| < \mathbf{u}$.

1.3 Error analysis

Consider the computation of $f(x) = x^{2^n}$ at a point x_0 by n squarings:

$$x_1 = x_0^2, x_2 = x_1^2, \dots, x_n = x_{n-1}^2$$

with $fl(x_0) = x_0$.

In floating point arithmetic, we compute:

$$\hat{x}_1 = x_0^2(1 + \delta_1), \hat{x}_2 = \hat{x}_1^2(1 + \delta_2), \dots, \hat{x}_n = \hat{x}_{n-1}^2(1 + \delta_n)$$

with $|\delta_i| \leq \mathbf{u}, \forall i$. The computed answer is, therefore,

$$\hat{x}_n = x_0^{2^n} (1 + \delta_1)^{2^{n-1}} \dots (1 + \delta_{n-1}^2)(1 + \delta_n)$$

Now, if $|\delta_1|, |\delta_2|, \dots, |\delta_r| \leq \mathbf{u}$, then there exist δ and η with $|\delta|, |\eta| \leq \mathbf{u}$, such that

$$(1 + \delta_1)(1 + \delta_2) \dots (1 + \delta_r) = (1 + \delta)^r = (1 + \eta)^{r+1}$$

Thus,

$$\hat{x}_n = x_0^{2^n} (1 + \delta)^{2^n} = f(x_0(1 + \delta))$$

for some δ with $\delta \leq \mathbf{u}$. In other words, the computed value \hat{x}_n is the exact answer for a perturbed input $x = x_0(1 + \delta)$.

The above is an example of *backward error analysis*.

1.4 Loss of significance

If x^* is an approximation to x , then the error in x^* is $x - x^*$. The *relative error* in x^* , as an approximation to x is the number

$$(x - x^*)/x$$

Every floating point operation in a computational process may give an error which, once generated, may then be amplified or reduced in subsequent computations.

One of the most common (and often avoidable) ways of increasing the importance of an error is commonly called **loss of significant digits**. x^* approximates x to r significant β -digits provided the absolute error $|x - x^*|$ is at most $\frac{1}{2}$ in the r^{th} significant β -digit of x , i.e.,

$$|x - x^*| \leq \frac{1}{2} \beta^{s-r+1}$$

with s the largest integer such that $\beta^s \leq |x|$. For example, $x^* = 3$ agrees with $x = \pi$ to one significant decimal digit, where as $x^* = \frac{22}{7} = 3.1428\dots$ is correct to three significant digits.

Suppose we are to calculate $z = x - y$ and we have approximations x^* and y^* for x and y , respectively, each of which is good to r digits. Then, $z^* = x^* - y^*$ may not be good to r digits. For example, if

$$x^* = (0.76545421) \times 10^1 \quad y^* = (0.76544200) \times 10^1$$

are each correct to seven decimal digits, then their exact difference

$$z^* = x^* - y^* = (0.12210000) \times 10^{-3}$$

is good only to three digits as an approximation to x , since the fourth digit of z^* is derived from the eighth digits of x^* and y^* , both possibly in error.

Hence, the *relative error* in z^* is possibly 10,000 times the relative error in x^* or y^* .

Such errors can be avoided by anticipating their occurrence. For example the calculation of

$$f(x) = 1 - \cos x$$

is prone to loss of significant digits for x near 0, however, the alternate representation

$$f(x) = \frac{\sin x^2}{1 + \cos x}$$

can be evaluated quite accurately for small values of x and is, on the other hand, problematic near $x = \pi$.

1.5 Condition and instability

Some numerical problems are inherently unstable. Consider for example the problem of finding the intersection of two straight lines which are nearly parallel. A slight perturbation of the input (slope) causes a large perturbation of the computed result. No algorithm can make such problems stable and the problem has to be addressed at the modeling stage. The word *condition* is used to describe the sensitivity of a function $f(x)$ to changes in the argument x . *Condition* is usually measured by the maximum relative change in the function value $f(x)$ caused by a unit relative change in the argument. The condition of $f(x)$ at x is sometimes measured as:

$$\max \left\{ \left| \frac{f(x) - f(x^*)}{f(x)} \right| / \left| \frac{x - x^*}{x} \right| : |x - x^*| \text{ is "small"} \right\}$$

The above simplifies (as an approximation) to

$$\left| \frac{f'(x)x}{f(x)} \right|$$

The larger the condition, the more ill-conditioned is the problem.

For example, if $f(x) = \sqrt{x}$, then $f'(x) = \frac{1}{2}\sqrt{x}$, hence the condition of $f(x)$ is, approximately,

$$\left| \frac{f'(x)x}{f(x)} \right| = \frac{1}{2}$$

This shows that computation of square roots is a well-conditioned process and it actually reduces the error.

On the other hand, if

$$f(x) = \frac{10}{1 - x^2}$$

then the condition is

$$\frac{2x^2}{|1-x^2|}$$

and this number can be quite large for $|x|$ near 1. It doesn't make sense to write a program to compute this function near $|x| = 1$ and re-formulation of the problem is the only solution.

The related notion of *instability* describes the sensitivity of a numerical (computational) process to the inevitable rounding errors committed during its execution using finite precision arithmetic. For example, consider the function

$$f(x) = \sqrt{x+1} - \sqrt{x}$$

for “large” x , say for $x \approx 10^4$. The condition of $f(x)$ there is $\frac{1}{2}$, which is quite good. But if we calculate $f(12345)$ in six-decimal arithmetic, we find

$$f(12345) = \sqrt{12346} - \sqrt{12345} = 111.113 - 111.108 = 0.005$$

while, actually, $f(12345) = 0.0045000326262\dots$. While the problem was well-conditioned, one of the steps in the computational process (subtraction) was ill-conditioned resulting in poor overall result. An alternative is to re-formulate the problem as

$$f(x) = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

Then, it is easy to show that all constituent steps in the computation are well-conditioned.

2 Fixed point iteration and Newton's method for root finding

2.1 Fixed point iteration

Consider the problem of obtaining the root of the equation

$$f(x) = 0$$

A popular method is to derive an equation from the above of the form

$$x = g(x)$$

so that any solution of the second equation (a *fixed point* for g) is a solution to $f(x) = 0$. Given an initial point x_0 one successively computes

$$x_{n+1} = g(x_n), \quad \text{for } n = 0, 1, 2, \dots$$

For the algorithm to be useful, we must show

1. for the given starting point x_0 , we can successively compute x_1, x_2, \dots
2. the sequence x_1, x_2, \dots converges to some point ξ
3. the limit ξ is a fixed point of $g(x)$, i.e., $\xi = g(\xi)$

To this end, let us make the following assumptions:

A1 There is an interval $I = [a, b]$ such that, for all $x \in I$, $g(x)$ is defined and $g(x) \in I$, that is, the function $g(x)$ maps I into itself.

A2 The iteration function is continuous on $I = [a, b]$.

The first assumption guarantees, by induction on n , that if $x_0 \in I$, then for all n , $x_n \in I$; hence $x_{n+1} = g(x_n)$ is defined and is in I .

Also, assumptions **A1** and **A2** together imply that $g(x)$ has a fixed point in $I = [a, b]$. For, if $g(a) = a$ or $g(b) = b$ it is obviously so. Otherwise, $g(a) > a$ and $g(b) < b$. Define $h(x) = g(x) - x$. Clearly, $h(a) > 0$ and $h(b) < 0$, and by the intermediate value theorem for continuous functions there must exist a ξ such that $h(\xi) = 0$ or $g(\xi) = \xi$.

In order to analyse the convergence let us define

$$e_n = \xi - x_n, \quad \text{for } n = 0, 1, 2, \dots$$

Since $\xi = g(\xi)$ and $x_n = g(x_{n-1})$, we have

$$e_n = \xi - x_n = g(\xi) - g(x_{n-1}) = g'(\eta_n)e_{n-1}$$

for some η_n between ξ and x_{n-1} by the mean value theorem of derivatives. Clearly, the sequence will converge to ξ if $g'(\eta_n) < 1$. Hence, the following assumption, in addition to **A1** and **A2**, guarantees the convergence of *fixed point iteration*.

A3 The iteration function is differentiable on $I = [a, b]$. Further there exists a non-negative constant $K < 1$ such that

$$\forall x \in I, |g'(x)| \leq K$$

2.2 Newton's method for root finding

Newton's method is a special case of *fixed point iteration*.

The Taylor's series for $f(x)$ at a point $x = x_n + h$ is given as

$$f(x_n + h) = f(x_n) + f'(x_n)h + \frac{1}{2}f''(x_n)h^2 + \dots$$

Retaining only up to the first order term we obtain

$$f(x_n + h) = f(x_n) + f'(x_n)h$$

This equation is the equation of the tangent to the curve at $(x_n, f(x_n))$. We can find the point where this tangent meets the x -axis by estimating the update h such that $f(x_n + h) = 0$. This is given by

$$h = -\frac{f(x_n)}{f'(x_n)}$$

This update would take us to the root of $f(x)$ if it is indeed a linear function passing through the origin. In the general case, we obtain the iteration

$$x_{n+1} = x_n + h = x_n - \frac{f(x_n)}{f'(x_n)}$$

This iteration is clearly a special case of *fixed point iteration* with

$$g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

Now, we had derived that for fixed point iteration

$$e_{n+1} \approx g'(\xi)e_n$$

Clearly, the smaller the value of $|g'(\xi)|$ the more rapidly $e_n \rightarrow 0$ as $n \rightarrow \infty$. The convergence of fixed point iteration should be the fastest when $|g'(\xi)| = 0$.

If $g(x)$ is twice differentiable then we get from Taylor's series:

$$e_{n+1} = \xi - x_{n+1} = g(\xi) - g(x_n) = -g'(\xi)(x_n - \xi) - \frac{1}{2}g''(\psi_n)(x_n - \xi)^2$$

for some ψ_n between ξ and x_n , that is

$$e_{n+1} = g'(\xi)e_n - \frac{1}{2}g''(\psi_n)e_n^2$$

Hence if $g'(\xi) = 0$ and $g''(x)$ is continuous at ξ , then

$$e_{n+1} \approx -\frac{1}{2}g''(\xi)e_n^2$$

and we say that the iteration converges quadratically (e_{n+1} is a quadratic function of e_n).

It is left to the reader to verify that the Newton's iteration

$$g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

is of this kind.

3 Some problems

3.1 Discretization

1. Derive the trapezoidal rule for numerical integration which goes somewhat like:

$$\int_a^b f(x)dx \simeq \sum_{k=0}^{n-1} \frac{1}{2}h[f(x_k) + f(x_{k+1})]$$

for discrete values of x_k in interval $[a, b]$. Compute $\int_0^\pi \sin(x)dx$ using trapezoidal rule (use $h = 0.1$ and $h = 0.01$) and compare with the exact result.

2. Consider the differential equation

$$\begin{aligned} y'(x) &= 2xy(x) - 2x^2 + 1, \quad 0 \leq x \leq 1 \\ y(0) &= 1 \end{aligned}$$

- (a) Show/verify that the exact solution is the function $y(x) = e^{x^2} + x$.
- (b) If we approximate the derivative operation with a divided difference

$$y'(x_k) = (y_{k+1} - y_k)/(x_{k+1} - x_k)$$

then show that solution can be approximated by the iteration

$$\begin{aligned} y_{k+1} &= y_k + h(2x_k y_k - 2x_k^2 + 1), \quad k = 0, 1, \dots, n \\ y_0 &= 1 \end{aligned}$$

where $x_k = kh$, $k = 0, 1, \dots, n$ and $h = 1/n$.

- (c) Use $h = 0.1$ to solve the differential equation numerically and compare (plot) your answers with the exact solution.
3. (a) Derive the Newton's iteration for computing $\sqrt{2}$ given by

$$\begin{aligned} x_{k+1} &= \frac{1}{2}[x_k + (2/x_k)], \quad k = 0, 1, \dots \\ x_0 &= 1 \end{aligned}$$

- (b) Show the Newton's iteration takes $O(\log n)$ steps to obtain n decimal digits of accuracy.
- (c) Numerically compute $\sqrt{2}$ using Newton's iteration and verify the rate of convergence.
4. Which of the following are rounding errors and which are truncation errors?
- (a) Replace $\sin(x)$ by $x - (x^3/3!) + (x^5)/5! \dots$
- (b) Use 3.1415926536 for π .
- (c) Use the value x_{10} for $\sqrt{2}$, where x_k is given by Newton's iteration above.
- (d) Divide 1.0 by 3.0 and call the result 0.3333.

3.2 Unstable and Ill-conditioned problems

1. Consider the differential equation

$$\begin{aligned} y'(x) &= (2/\pi)xy(y - \pi), \quad 0 \leq x \leq 10 \\ y(0) &= y_0 \end{aligned}$$

- (a) Show/verify that the exact solution to this equation is

$$y(x) = \pi y_0 / [y_0 + (\pi - y_0)e^{x^2}]$$

- (b) Taking $y_0 = \pi$ compute the solution for
- i. an 8 digit rounded approximation for π
 - ii. a 9 digit rounded approximation for π

What can you say about the results?

2. Solve the system

$$\begin{aligned} 2x &- 4y &= 1 \\ -2.998x &+ 6.001y &= 2 \end{aligned}$$

using any method you know. Compare the solution with the solution to the system obtained by changing the last equation to $-2.998x + 6y = 2$. Is this problem stable?

3. Examine the stability of the equation

$$x^3 - 102x^2 + 201x - 100 = 0$$

which has a solution $x^* = 1$. Change one of the coefficients (say 201 to 200) and show that $x^* = 1$ is no longer even close to a solution.

3.3 Unstable methods

1. Consider the quadratic

$$ax^2 + bx + c = 0, \quad a \neq 0$$

Consider $a = 1$, $b = 1000.01$, $c = -2.5245315$. Suppose that $\sqrt{b^2 - 4ac}$ is computed correctly to 8 digits, what is the number of digits of accuracy in x ? What is the source of the error?

2. Show that the solution to the quadratic can be re-written as

$$x = -2c / (b^2 + \sqrt{b^2 - 4ac})$$

Write a program to evaluate x for several values of a , b and c (with b large and positive and a , c of moderate size). Compare the results obtained with the usual formula and the formula above.

3. Consider the problem of determining the value of the integral

$$\int_0^1 x^{20} e^{x-1} dx$$

If we let

$$I_k = \int_0^1 x^k e^{x-1} dx$$

Then, integration by parts gives us (please verify)

$$\begin{aligned} I_k &= 1 - kI_{k-1} \\ I_0 &= \int_0^1 e^{x-1} dx = 1 - (1/e) \end{aligned}$$

Thus we can compute I_{20} by successively computing I_1, I_2, \dots . Compute the (k, I_k) table with a program, plot, and see if it makes sense. What are the errors due to?

Compare the results with that obtained using the following recursion (which you can *easily!* derive by integrating by parts twice).

$$I_k = (1/\pi) - [k(k-1)/\pi^2]I_{k-2}, \quad k = 2, 4, 6, \dots$$

4. The standard deviation of a set of numbers x_1, x_2, \dots, x_n is defined as

$$s = (1/n) \sum_{i=1}^n (x_i - \bar{x})^2$$

where \bar{x} is the average. An alternative formula that is often used is

$$s = (1/n) \sum_{i=1}^n x_i^2 - \bar{x}^2$$

- (a) Discuss the instability of the second formula for the case where the x_i are all very close to each other.
- (b) Observe that s should always be positive. Write a small program to evaluate the two formulas and find values of x_1, \dots, x_{10} for which the second one gives negative results.