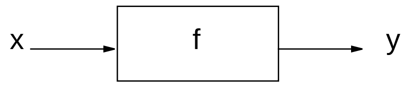
Machine Arithmetic and Related Ugliness

Why should I bother?



In the example, $y = f(x)$.

Consider a small perturbation in x (say x').

$y + y' = f(x + x')$.

We should start to get worried if a small perturbation in the input leads to a large perturbation in the output. This is called an ill-conditioned system.

Ill Conditioning:

This can be a result of the inherent nature of the function f (i.e. the problem itself is ill conditioned) or a function of how f is computed (i.e. the algorithm is unstable).

Example: $y = f(x) = (x / 10^{100}) * 10^{100}$

It is easy to see that y = x
And depending on how this computation is performed:
$y = (x / 10^{100}) * 10^{100}$
or $y = x * (10^{100} / 10^{100})$
we may get the right answer (or not!).

Example: Legendre Polynomials using two-step recurrance relations.

Example:

```
main() {
    double f;
     f = 1.8;
    if (f * 1.1 == 1.98)
    printf("A-HA!");
}
```


How are numbers stored in a computer?

$x : (+/-)\ b_n b_{n-1} b_{n-2} ... b_0 . b_{-1} b_{-2} ...$
where x is simply the sum of $b_i 2^i$.

Notice that the representation is not unique:

$$0.01\bar{1} = 0.1$$

We can force uniqueness by assuming finite precision.

$$x = f\ 2^e$$

Here, f is the mantissa and e is the exponent.

$f: (+/-)\ .b_{-1} b_{-2} ... b_{-t}$
$e: (+/-)\ c_{s-1} c_{s-2}\ ...\ c_0$

$\max |x| = (1 - 2^{-t}) 2^{2^\wedge s - 1}$
$\min |x| = 2^{-2^\wedge s}$

Overflow and underflow on exceeding these.

Storage Formats:

Floating point:

The point floats to keep b_{-1} equal to one all the time.

Fixed point:

I know what I am doing so dont bother floating the point around.

Others:

Complex numbers, rational number representations.

Rounding:

The bed of Procrustes: One size fits all.

$$x = \pm\left(\sum_{k=1}^{\infty}\left(b_{-k}2^{-k}\right)\right)2^{e}$$

$$\tilde{x} = \pm\left(\sum_{k=1}^{t}(\tilde{b}_{-k}2^{-k})\right)2^{\tilde{e}}$$

i) Chopping: Retain the first t bits of the mantissa.

ii) Symmetric rounding: If the first discarded bit is 1, round up, if it is 0, round down.

$$\tilde{x} = chop\left(x + \frac{1}{2}2^{-t}2^{e}\right)$$

We can show that (x - chop(x))/x <= 2.2^{-t} and (x - round(x))/x <= 2^{-t}

Machine Arithmetic:

Multiplication:

$$x(1+\varepsilon_x) \times y(1+\varepsilon_y) \approx x \cdot y(1+\varepsilon_x+\varepsilon_y)$$

$$\varepsilon_{xy} \approx (\varepsilon_x + \varepsilon_y)$$

Division:

$$\frac{x(1+\varepsilon_x)}{y(1+\varepsilon_y)} \approx \frac{x}{y}(1+\varepsilon_x-\varepsilon_y)$$

$$\varepsilon_{\frac{x}{y}} \approx (\varepsilon_x - \varepsilon_y)$$

Addition/subtraction:

$$x(1+\varepsilon_x) + y(1+\varepsilon_y) \approx (x+y)\left(1 + \frac{x\varepsilon_x + y\varepsilon_y}{x+y}\right)$$

$$\varepsilon_{x+y} \approx \frac{x\varepsilon_x + y\varepsilon_y}{x+y}$$

Examples:

i) $(a - b)^2 = a^2 + b^2 - 2ab$.

Try this for a = 1.8 and b = 1.7 with a 2 digit accuracy.

You will see that the result is -0.10 !

ii) sqrt (x + d) - sqrt(x) causes problems for x > 0 and |d| being small.
(recast the expression in terms of division and addition since these are more benign operations.

iii) y = f(x + d) - f(x) for d tending to 0 can be expanded using taylor series and dropping higher order terms (in d).

Conditioning of a Problem:

Given a function f of the form y = f(x), the conditioning of the function reflects on the sensitivity of the function output to perturbations in the input.

$$y + \delta y = f(x + \delta x)$$

Using a Taylor expansion and neglecting higher order terms, we get:

$$\delta y \approx f'(x)\delta x$$

$$\frac{\delta y}{y} \approx \frac{x(f'(x))}{f(x)} \cdot \frac{\delta x}{x}$$

The condition of f at x is defined as:

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

Examples: Compute the condition of functions x, x^2, x^n, e^x.

Conditioning of a Problem:

Consider the general case when

$$x = [x_1, x_2, \ldots, x_m]^T$$

and

$$y = [y_1, y_2, \ldots, y_n]^T$$

Here, output y_i can be expressed as:

$$y_i = f_i(x_1, x_2, \ldots, x_m)$$

We assume that each function f_i has a derivative with respect to each of the m components at the point *m*.

In this case, we can perturb each of the m components of x and see its impact on y_i.

$$\Upsilon_{ij}(x) = (cond_{ij} f)(x) = \left|\frac{x_j(f_i'(x))}{f_i(x)}\right|$$

This gives a matrix of condition numbers and any convenient norm of the matrix yields a single condition number.

Example:

$$y_1 = f(x) = x_1 + x_2$$

Here, $y = [y_1]$ and $x = [x_1, x_2]^T$.

$$\Upsilon_{11}(x) = (cond_{11} f)(x) = \left|\frac{x_1(f_1'(x))}{f_1(x)}\right|$$

or

$$\Upsilon_{11}(x) = (cond_{11} f)(x) = \left|\frac{x_1}{x_1 + x_2}\right|$$

and

$$\Upsilon_{12}(x) = (cond_{12} f)(x) = \left|\frac{x_2}{x_1 + x_2}\right|$$

This indicates poor conditioning if x_1 and x_2 have opposite signs and approximately identical magnitudes that are not close to zero.

(Notice the relation to error propagation due to rounding during addition/subtraction.)

Example:

$$I_n = \int_0^1 \frac{t^n}{t+5} dt = y$$

This is solved by the recurrence relation:

$$y_k = -5y_k + \frac{1}{k}$$

with $y = y_n$ and y_0 has some representational error e.
We can show that:

$$y = (-5)^n y_0 + p_n$$

From this, it is easy to show that the conditioning of this function varies with n as 5^n.

You can fix this problem by using the recursive formula:

$$y_{k-1} = \frac{1}{5}\left(\frac{1}{k} - y_k\right)$$

See text for complete proof.