SINGLE precision macheps $= 2^{-23} \approx 1.2 \times 10^{-7}$

DOUBLE ———— $= 2^{-52} \approx 2.2 \times 10^{-16}$

The gap between a number and the next larger floating point number is:

$$\underset{\substack{\uparrow \\ \text{unit in last position}}}{ulp}(x) = (0.\underbrace{00\cdots0}_{p-2}1)_2 \times 2^E = 2^{-(p-1)} \times 2^E = \varepsilon 2^E$$

(we assume $x>0$, if $x<0$ then this is gap between x and next smaller floating point number)

<u>Bottom line</u>: There is an error committed by just representing the floating point number in a computer!

One can show that:

$$\boxed{fl(x) = x(1+\delta)} \quad \text{where} \quad |\delta| \leq \varepsilon = \text{machine epsilon}$$

(floating point rep of a number x)

(in the case of round to nearest $|\delta| < \frac{\varepsilon}{2}$)

This means the <u>relative</u> error of rounding is:

$$\left| \frac{x - fl(x)}{x} \right| = |\delta| \leq \varepsilon$$

A key feature of the IEEE standard is that it guarantees correctly rounded operations:

$$\boxed{fl(x \odot y) = (x \odot y)(1+\delta)}$$

where $|\delta| \leq \varepsilon$ = machine epsilon

where $\odot = +, -, \times, \div$ operations, x, y are machine numbers

$\Rightarrow$ any simple operation is guaranteed to give a result w/rel-error $\leq$ macheps.

This is achieved in hardware by working with higher precision intermediate results and then rounding to desired precision.. (sometimes called "guard bits")

However the result of a sequence of two or more arithmetic operations is not guaranteed to be correctly rounded.

Take for example:

$x = 1$ and $(x+y)-z = 2^{-25}$

$y = 2^{-25}$

$z = 1$

(all representable with no error in say IEEE single precision).

$x+y = 1+2^{-25} = 1.\underbrace{0 \ldots 0}_{24 \text{ zeros}}1$ (in base 2)

$fl(x+y) = 1$ (because $\varepsilon = 2^{-23}$ in single precision)

thus:

$fl(fl(x+y)-z) = 0$ whereas the exact result 2^{-25} can be represented with no error!

(note: • result would be correct if IEEE double precision is used
• we used round to nearest number. Other rounding method may give different results.)

Other aspects of IEEE standard you may encounter are:

signed 0: $+0 = -0$ (sign bit does mean something in this case)

infinity: $\frac{1}{+0} = +\infty$, $\frac{1}{-0} = -\infty$

NaN = result of an invalid operation (i.e. bug)
= Not A Number

when do you encounter NaNs?

it makes sense to define:

$a \times 0 = 0$ if a finite

$\frac{a}{0} = \infty$ if $a > 0$

$a \times \infty = \infty$ if $a > 0$

etc...

However the operations:

$$\left.\begin{matrix} 0 \times \infty \\ \frac{0}{0} \\ \infty - \infty \end{matrix}\right\} \text{give NaN.}$$

<u>Question</u>: what about

$\frac{\infty}{0} = \infty$

$\frac{0}{\infty} = 0$

$\frac{\infty}{\infty} = \text{NaN}$ (I could have $x_k \to \infty$, $y_k \to \infty$ but $\frac{x_k}{y_k}$ undefined limit. if I don't know more)

+ parallel resistor example

If x approximates a number x_*:

$x - x_* =$ <u>error</u>

$|x - x_*| =$ <u>absolute error</u>

$\frac{|x - x_*|}{|x|} =$ <u>relative error</u> (if x and x^* are close it doesn't affect so that much to change denominator by $|x^*|$)

Chapter 2 Nonlinear equations

We will study algorithms to find the roots or zeros of some function $f: \mathbb{R} \to \mathbb{R}$, i.e. the points x for which $f(x) = 0$. Numerical algorithms are essential as many times it is not possible to find an explicit formula for the root of a function. (without going too far, it is known that for polynomials of degree ≥ 5 there is no all purpose formula for finding the roots, as we have for e.g. a quadratic).

Other examples of non-linear equations:

$$f(x) = x - \tan x = 0$$

$$f(\lambda) = \alpha e^{\lambda} + \beta \frac{(e^{\lambda t} - 1)}{\lambda} - \gamma = 0 \qquad (t \text{ is given})$$

(population model, see book's intro for chap §2)

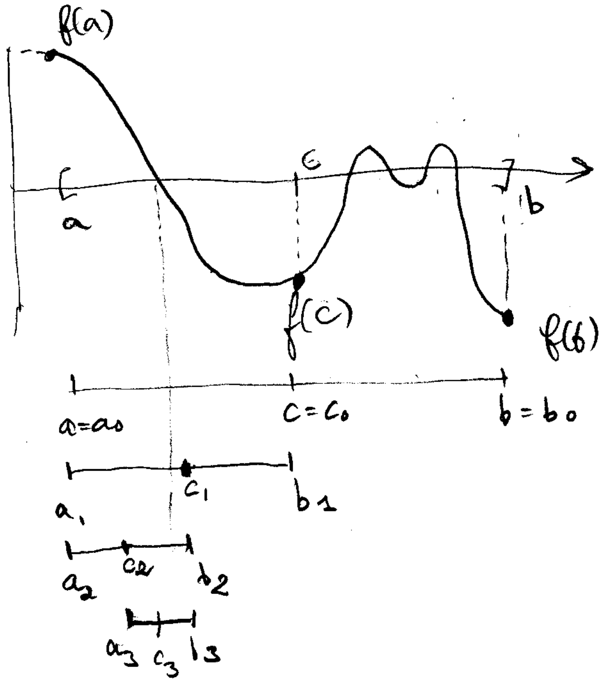
§2.1 The bisection or interval halving method

Key idea: By the intermediate value theorem, any continuous function on $[a,b]$ such that

$$f(a) f(b) < 0 \quad (\text{i.e. function has opposite signs at endpts})$$

must have at least one zero in $[a,b]$.

This principle can be used to progressively narrow down an interval containing a root of $f(x)$.



Here $f(a)f(b) < 0$ and $f(a)f(c) < 0$ means there is a root in $[a, c]$ interval. Then reiterate.

Note: it is clear that if there are several roots in interval, bisection will converge to one of them. If we need a particular root then initial interval must contain only this root...

Here is how the algorithm looks like in pseudocode:

```
inputs: a, b, stopping criterion parameters, Maxit.
outputs: a, b
u = f(a); v = f(b); e = b - a;
if sign(u) = sign(v) then stop   (* bisection cannot be applied,
                                    user needs to narrow down
                                    location of root *)

for k = 1 : Maxit.
    e = e/2
    c = a + e          (* midpoint *)
    w = f(c)
    if |e| < δ or |w| < ε then stop   (* stopping criteria ... can be
                                         something else *)
    if sign(w) sign(u) < 0
        b = c; v = w       (* root is in left interval *)
    else
        a = c; u = w       (* root is in right interval *)
```

Stopping criterion:

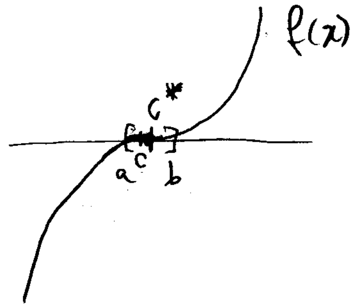
Here we have 3 conditions that terminate iterations:

- # of iterations > Max iterations
  (good to have this safeguard)
- $|b-a| < \delta$ error small enough i.e. we know our root within $|b-a|$
- $|f(c)| < \varepsilon$ c is sufficiently good a loop.

⚠ Never test for a number == 0 in algorithms this is very unlikely! and you will get very easily breakable code
(unfortunately this is the first stopping criterion suggestion in algorithm 2.1 in book... please avoid doing it.

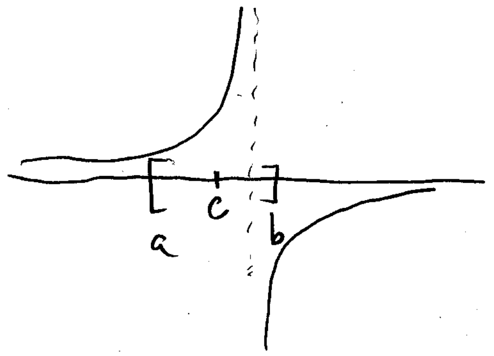
Why do we need all these criteria?

that we should limit # of iter is clear
also we can have the following pathological cases:



c^* = multiple root so f is very flat in a neighborhood of c^*
$\rightsquigarrow$ hard to give root location with a lot of precision

$|f(c)| < \varepsilon$ but $|b-a| > \delta$



Nobody stops user to supply a discontinuous function in which case it is better to end iterations gracefully.

$|b-a| < \delta$ but $|f(c)| > \varepsilon$

Another stopping criterion that is suggested by your textbook is:

$$\frac{|c_n - c_{n-1}|}{|c_n|} < \varepsilon \, , \quad c_n \neq 0$$

which means we stop when the relative difference in two consecutive approximations of the root become small.

The use of signs instead of values of $f(a), f(b), f(c)$ is to avoid potential overflows: when multiplying two floating point numbers, the result could be larger than largest representable number.

Error analysis: denote by $[a_0, b_0]$, $[a_1, b_1]$, ... the intervals that bracket the root, as generated by algorithm.
We of course have:

$$a_0 \leq a_1 \leq a_2 \leq \cdots \leq b_0$$
$$a_0 \leq \cdots \leq b_2 \leq b_1 \leq b_0$$
$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n) \quad \text{for } n \geq 0.$$

Since $\{a_n\}$ is an increasing sequence bdd above it converges
$\{b_n\}$ ——— decreasing ——— below ———

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n) = \frac{1}{2^2}(b_{n-1} - a_{n-1})$$
$$\cdots$$
$$= \frac{1}{2^{n+1}}(b_0 - a_0)$$

$$\Rightarrow \lim_{n\to\infty} b_n - a_n = \lim \frac{1}{2^n}(b_0 - a_0) = 0$$

$\Rightarrow$ the limit of $\{a_n\}$ and $\{b_n\}$ is the same, call it r.

If we take the limit:

$$0 \geq f(a_n) f(b_n) \longrightarrow \quad 0 \geq (f(r))^2$$

(and we can since f is continuous) $\quad \Rightarrow f(r) = 0$
r is root.

If we are at step n in algorithm which point in $[a_n, b_n]$ Should we take as our approx to the root? The best point is (if we do not know anything else about root) is the midpoint.

$$c_n = \frac{a_n + b_n}{2}$$

$$\underset{\text{error}}{\underset{\parallel}{|r - c_n|}} \leq \frac{1}{2}(b_n - a_n) = \frac{1}{2}\,\frac{1}{2^n}(b_0 - a_0) = 2^{-(n+1)}(b_0 - a_0)$$

Here is a theorem summarizing our findings.

<u>Theorem (Bisection method)</u>

If $[a_0, b_0], [a_1, b_1], [a_2, b_2]$ are the intervals generated by the bisection method then the limits $\lim_{n\to\infty} a_n$, $\lim_{n\to\infty} b_n$ exist and are equal, the limit being a root of f.

If $r = \lim_{n\to\infty} c_n = \frac{a_n + b_n}{2}$ Then:

$$|r - c_n| \leq \frac{1}{2^{n+1}}(b_0 - a_0)$$

With this explicit error bound it is easy to estimate how many iterates are needed to achieve a desired precision.

§ 2.3 Newton's method :

extremely important algorithm that can be generalized to systems of non-linear equations and can be used for example in optimization.

key idea: Let x^* be a root of $f(x)$ and assume $f \in C^2$. Then Taylor's theorem says:

$$0 = f(x^*) = f(x+h) = f(x) + h f'(x) + O(h^2)$$

where $h = x^* - x$.

If we solve for h: $h = -\dfrac{f(x)}{f'(x)}$.

Thus if x is close to x^* we can expect $x + h = x - \dfrac{f(x)}{f'(x)}$

to be even closer (we will prove this rigorously).

Putting everything together we get the iteration:

$$x_0 = \text{given}$$
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (\text{breaks down if } f'(x^*) = 0)$$

Newton's method

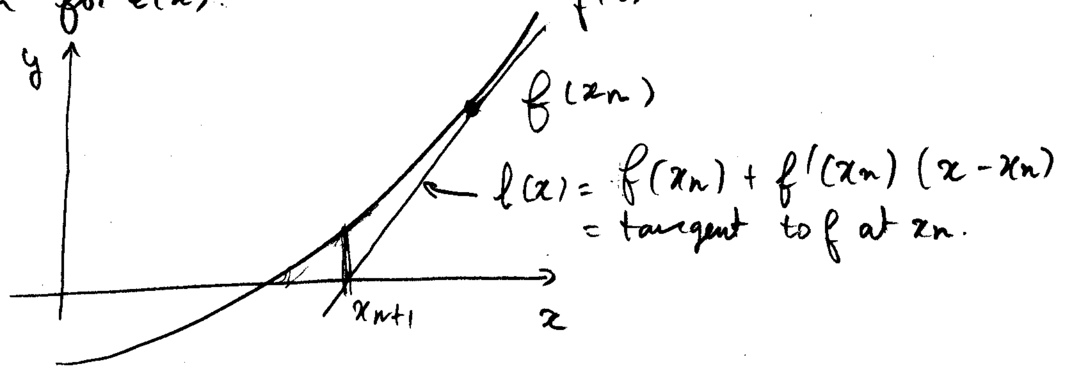
```
$v = f(x_0)$
if $|v| < \varepsilon$ then stop      (* $\varepsilon$ = tolerance *)
for $k = 1$ : Maxit
    $x_{new} = x - v/f'(x)$
      $v = f(x_{new})$
    if $|x_{new} - x| < \delta$ or $|v| < \varepsilon$ stop
    $x = x_{new}$
```

Graphical interpretation: What we are doing is approximating the problem of finding a root of a non-linear function by a sequence of linear root-finding problems. Indeed if we replace $f(x)$ by its first two terms in Taylor expansion around a point c:

$$f(x) = f(c) + f'(c)(x-c) + \frac{f''(c)}{2}(x-c)^2 + \cdots$$

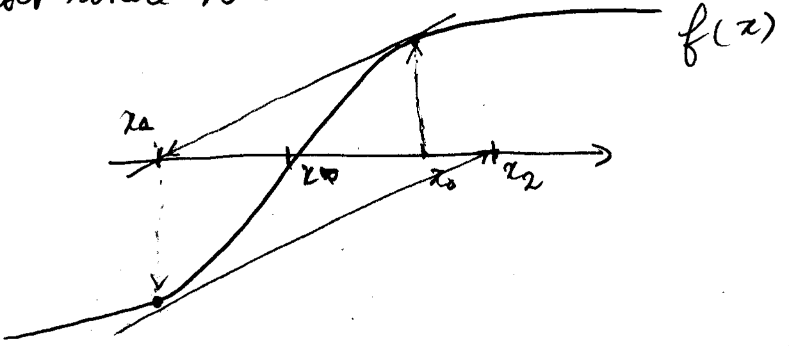
$$l(x) = f(c) + f'(c)(x-c) \quad = \quad \text{tangent of } f(x) \text{ at } c.$$

What Newton's method does is to solve the (trivial) root finding problem for $l(x)$.



$f(x)$
y
$f(x_n)$
$l(x) = f(x_n) + f'(x_n)(x - x_n)$
$=$ tangent to f at x_n.
x_{n+1}
x

Just by looking at this graphical construction one can come up with cases where Newton's method does not converge!



So any convergence result for Newton's method works only if initial guess is sufficiently close to x_*.

We shall prove in 2 ways the convergence of N. M.
The first way is straightforward and can be generalized easily to more dimensions.
For the second way we have to wait until we see the powerful contracting mapping theorem.
Let us first introduce the following notion of continuity.

<u>Def (Lipschitz continuity)</u> A function f is said to be Lipschitz continuous with Lipschitz constant $L > 0$ if:

$$|f(x) - f(y)| \leq L|x-y|$$

Note that Lipschitz continuity $\Longrightarrow$ continuity:
for some $\varepsilon > 0$ $\exists\, \delta = \frac{\varepsilon}{L}$ s.t. $|x-y| < \delta = \frac{\varepsilon}{L}$

$$\Rightarrow \quad |f(x) - f(y)| \leq L\frac{\varepsilon}{L} = \varepsilon.$$

Also if $f \in C^1[a,b]$ ($[a,b]$ bdd interval) then f is Lipschitz cont.
Using Taylor's theorem:

$$f(x) = f(y) + f'(\xi)(x-y) \quad \text{for some } \xi \in [a,b]$$

$$\Rightarrow \quad |f(x) - f(y)| \leq |f'(\xi)|\,|x-y| \leq L\,|x-y|$$

where $L = \max\limits_{x \in [a,b]} |f'(x)|$, which exists and is attained because $f'(x) =$ cont. on closed & bdd = compact set.

We are now ready to show the following theorem. (slightly more powerful than books' theorem)

Theorem (convergence of Newton's method)

Let x_* be a simple zero of $f(x)$ i.e. s.t. $f'(x_*) \neq 0$, and assume $f'(x)$ is Lipschitz continuous on a nbd of x_* with Lipschitz constant L. Then:

$\exists\, \varepsilon > 0,\ C > 0$ s.t.

$$|x_0 - x_*| < \varepsilon \Rightarrow |x_{n+1} - x_*| \leq C\,|x_n - x_*|^2$$

translated in english: Newton's method converges quadratically provided initial guess / iterate x_0 is close enough to x_*.

Proof: We work by induction

$$x_{k+1} - x_* = x_k - \frac{f(x_k)}{f'(x_k)} - x_*$$

$$= \frac{1}{f'(x_k)}\Big(\underbrace{f(x_*)}_{=0} - f(x_k) + f'(x_k)(x_k - x_*)\Big)$$

Using fundamental theorem of calculus:

$$f(x_k) - f(x_*) = \int_{x_*}^{x_k} dx\, f'(x) \quad \text{thus:}$$

$$A = f(x_*) - f(x_k) + f'(x_k)(x_k - x_*) = \int_{x_*}^{x_k} (f'(x_k) - f'(x))\,dx$$

$$\overset{\text{c.ov.}}{=} \int_{1}^{0} \big(f'(x_k) - f'(x_k + t(x_* - x_k))\big)(x_* - x_k)\,dt$$

$x = x_k + t(x_* - x_k)$

$$|A| \leq \int_0^1 |f'(x_k) - f'(x_k + t(x_* - x_k))|\,|x_* - x_k|\,dt$$

$$\leq \int_0^1 L t\,|x_* - x_k|^2\,dt \quad \text{by Lipschitz cont. of } f'$$

$$= \frac{L}{2}|x_k - x_*|^2$$

Therefore:

$$|x_{k+1} - x_*| \leq \frac{1}{|f'(x_k)|} \frac{L}{2} |x_k - x_*|^2$$

which screams for quadratic convergence if we can show $|f'(x_k)|^{-1}$ is bounded in some neighborhood of x_*.

Incidentally this justifies us dividing by $f'(x_k)$.

We will show that for ε sufficiently small:

$$(*) \quad 2|f'(x_k)| \geq |f'(x_*)| \underset{\uparrow \text{since } f'(x_*) \neq 0}{>} 0$$

From $(*)$ we get:

$$|x_{k+1} - x_*| \leq |f'(x_*)|^{-1} L |x_k - x_*|^2$$

So in addition if $f'(x_*)$ is small convergence is not as good as if $f'(x_*)$ was large (makes sense from graphical perspective).

<u>Proof of $(*)$:</u> Since $f'(x)$ is continuous there is ε_1 for which:

$$|x - x_*| < \varepsilon_1 \Rightarrow \operatorname{sgn} f'(x) = \operatorname{sgn} f'(x_*)$$

if $f'(x_*) > 0$:

$$f'(x) = f'(x_*) + f'(x) - f'(x_*)$$
$$\geq f'(x_*) - | \qquad |$$
$$\geq f'(x_*) - L|x - x_*|$$

Choosing $\varepsilon = \min\left(\varepsilon_1, \frac{|f'(x_*)|}{2L}\right)$:

$$f'(x) \geq f'(x_*) - \frac{1}{2}|f'(x_*)|$$
$$= \frac{f'(x_*)}{2} \quad \text{since } f'(x_*) > 0$$

$$\Rightarrow |f'(x)| \geq \frac{1}{2}|f'(x_*)|$$

if $f'(x_*) < 0$

$$f'(x) = f'(x_*) + f'(x) - f'(x_*)$$
$$\leq f'(x_*) + |f'(x) - f'(x_*)|$$
$$\leq f'(x_*) + L|x - x_*|$$

Choosing $\varepsilon = \min\left(\varepsilon_1, \frac{|f'(x_*)|}{2}\right)$

$$\underbrace{f'(x)}_{-|f'(x)|} \leq \underbrace{f'(x_*)}_{-|f'(x_*)|} + \frac{|f'(x_*)|}{2}$$

$$-|f'(x)| \leq -\frac{|f'(x_*)|}{2}$$

$$|f'(x)| \geq \frac{1}{2}|f'(x_*)|$$

Local convergence of Newton's method become global in some particular cases. For example:

<u>Theorem (Newton's method for convex functions)</u>

Let f be $C^2(\mathbb{R})$, convex and increasing. If f has a zero it is unique and Newton's method converges regardless of initial iterate x_0.

<u>Secant method</u>:

<u>key idea</u>: for Newton's method we need to know derivative of f, and this may not be available or be very costly. (this is especially true for high dimensions)

$\rightsquigarrow$ replace derivative by secant:

$$f'(x_k) = \lim_{x \to x_k} \frac{f(x) - f(x_k)}{x - x_k}$$

$$\approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

Therefore the secant update is:

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

and algorithm is similar to Newton's method, with the only difference being that we need to keep two iterates at any time.

The convergence for the secant method is <u>local</u> like for Newton's method.

The convergence rate is of $\frac{1+\sqrt{5}}{2} \approx 1.6$

(do you recognize this rate from computer lab #2?)