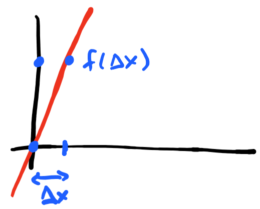
# Polynomial conditioning and Horner's rule

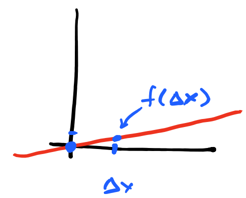
Thursday, August 27, 2020 9:22 PM

## Conditioning of the polynomial root-finding problem

- linear example: $f(x) = 100 \cdot x$ vs. $f(x) = .001 \cdot x$




True root is at $x=0$




Here, if we have a small change $\Delta x$, it looks almost correct, since $f(\Delta x)$ is almost $0$.

- Quadratic case, analysis:

$p(x) = a \cdot x^2 + bx + c$, quadratic polynomial

What is the conditioning of the root finding problem with respect to the input $a$? i.e., $a$ is known up to precision $\varepsilon$

let $f(a)$ represent the problem of returning a root ← which one? for now leave it vague

( consider $b, c$ to be in $\infty$ precision )

So if $r = f(a)$ ← algorithm, not polynomial is a root, ($p(r)=0$) then $a \cdot r^2 + b \cdot r + c = 0$

want $K(a) = \left| \frac{a}{f(a)} \cdot \frac{dr}{da} \right|$ ← how to find? → Do implicit differentiation!

$$\frac{d}{da}\left( a \cdot r^2 + b \cdot r + c \right) = \underbrace{\frac{d}{da}(0)}_{=0}$$

$$\Big(\underbrace{\tfrac{d}{da} a}_{=1}\Big) r^2 + a \cdot \frac{d}{da} r^2 + b \cdot \frac{d}{da} r + 0$$

$$r^2 + 2ar \cdot \frac{dr}{da} + b \cdot \frac{dr}{da} = 0$$

i.e. $\frac{dr}{da} = \frac{-r^2}{2a \cdot r + b} = \frac{-r^2}{\pm\sqrt{b^2 - 4ac}}$ (used quadratic formula)

hence $K(a) = \left| \frac{a}{f(a)} \cdot \frac{dr}{da} \right| = \left| \frac{a}{r} \cdot \frac{-r^2}{\sqrt{b^2-4ac}} \right| = \left| \frac{r}{t_1 - t_2} \right|$

where $t_1, t_2$ are the roots

$t_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

$t_2 = \frac{-b - \sqrt{\cdots}}{2a}$

so $t_1 - t_2 = \frac{1}{2a}\left(2 \cdot \sqrt{b^2 - 4ac}\right)$

**Interpretation:**

$r$ is a root ($t_1$ or $t_2$),

so if $\left| \frac{t_1}{t_1 - t_2} \right|$ or $\left| \frac{t_2}{t_1 - t_2} \right|$ is large, it's an ill-conditioned problem

Ex: $t_1 = 10$, $t_2 = 10.1$ ( i.e., $p(x) = (x-10)(x-10.1)$ )

is just a little ill-conditioned ( $K \approx \frac{10}{0.1} = 100$ )

i.e., expect to have an answer with 14 correct digits (in double precision, since start w/ about 16 digits)

**Ex:** $t_1 = 10{,}000$, $t_2 = 10{,}000.1$
is even worse conditioned ( $\kappa \approx \frac{10^5}{.1} = 10^6$ ), lose 6 digits

## Algorithms

Is the quadratic formula stable? (see companion Demo)

$$r_{\pm} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

Problem: if $b^2 \gg 4ac$, then $\sqrt{b^2-4ac} \approx |b|$

so $-b + |b|$ (if $b>0$)

or $-b - |b|$ (if $b<0$) has a lot of

Subtractive cancellation.

Ex of subtractive cancellation (w/ 6 digits of precision)

$$
\begin{array}{rl}
x = & 3.14159\overbrace{00000}^{\text{garbage digits (not to be trusted)}} \\
y = & 3.14165\,00000 \\
\hline
y - x = & 0.00016\,00000
\end{array}
$$

stored as $\underbrace{1.6\,0000}_{\text{Since we're in 6 digit precision, we think we can trust these}}0 \times 10^{-4}$

but these are not to be trusted!

Quadratic formula isn't stable...
but easy to fix (see demo)

## Evaluating polynomials

(seems easier than root-finding..., right?)
see "Ch1_Stability_simple.ipynb" Demo

$$p(x) = \underbrace{(x-10)^2}_{\text{Algo 1}} = \underbrace{x^2 - 20x + 100}_{\text{Algo 2}}$$

which algorithm is better?

1) Stability. We did this in the stability lecture, and concluded

Algo 1 is better than Algo 2

2) Speed. It's not that obvious, so what about

$$p(x) = (x-10)^4 = x^4 - 40x^3 + 600x^2 - 4000x + 10000$$

Fast!
1. $\tilde{x} = x - 10$
2. $\tilde{\tilde{x}} = \tilde{x}\cdot\tilde{x}$
3. $p(x) = \tilde{\tilde{x}}\cdot\tilde{\tilde{x}}$

... so 3 "flops"
Floating Point Operation*

1. $\tilde{x} = x^2$
2. $\tilde{\tilde{x}} = \tilde{x}\cdot x \quad (=x^3)$
3. $\tilde{\tilde{\tilde{x}}} = \tilde{\tilde{x}}\cdot x \quad (=x^4)$

$$\tilde{\tilde{\tilde{x}}} \underset{7.}{-} \underbrace{40\cdot\tilde{\tilde{x}}}_{4.} \underset{8}{+} \underbrace{600\cdot\tilde{x}}_{5.} \underset{9}{-} \underbrace{4000\cdot x}_{6.} \underset{10}{+} 1000$$

... so 10 "flops"

* we usually ignore constants, since actual cost depends on hardware, e.g., some

CPUs do fused add & multiply ... so Algo 1 is faster 2

i.e., if $n$ coefficients,

cost is $O(n)$

Unfortunately, if we're given $p(x) = x^4 - 40x^3 + 600x^2 - 4000x + 10000$

it's not obvious (w/o solving a root finding problem) that

this is equivalent to $p(x) = (x-10)^4$

but there are better methods than the naive algorithm

One of the oldest / best-known is Horner's Rule (1819)

very simple:

if $p(x) = x^4 - 40x^3 + 6000x^2 - 4000x + 10000$

$$= \Big(\big((\underset{1}{x - 40})\underset{2}{x} \underset{3}{+} 6000\big)\underset{4}{x} \underset{5}{-} 4000\Big)\underset{6}{x} \underset{7}{+} 10000$$ 7 flops

(usually 8 flops if leading coefficient isn't 1)

Speed: for a general $n^{th}$ degree polynomial, Horner's rule takes $n$ multiplies and $n$ additions, and there's no faster algorithm

Stability: good backward stability (see N. Higham '02)

## Software

| | Evaluating $p(x)$ | Solving $p(x)=0$ |
|---|---|---|
| MATLAB: | polyval | roots |
| Python: | numpy.polyval | numpy.roots |