In general

approx of $f(x)$

$$T_n(x) = \sum_{k=0}^{n} \frac{1}{k!}(x-x_0)^k f^{(k)}(x_0)$$

error

$$R_n(x) = \int_{x_0}^{x} \frac{1}{n!}(x-t)^n f^{(n+1)}(t)\,dt$$

Taylor's theorem: If f has $k+1$ continuous derivatives then $f(x) = T_n(x) + R_n(x)$ where T_n and R_n are as above.

Use the theorem to estimate errors in the approximations

$$f(x) \approx T_n(x)$$

How? like this.

$$|R_n(x)| \leq \max_{t \in I} |f^{(n+1)}(t)| \cdot \frac{1}{(n+1)!} \cdot |(x-x_0)^{n+1}|$$

Therefore,

① might be small ← ② →

$$|f(x) - T_n(x)| \leq \frac{1}{(n+1)!} \max_{t \in I} |f^{(n+1)}(t)| \; |x-x_0|^{n+1}$$

Ideally this error is small. What makes it small

(of degree n)

Note if $f(t)$ were a polynomial to begin with then $f^{(n+1)}(t) = 0$ so $\max_{t \in I} |f^{(n+1)}(t)| = 0$.

Thus $f(x) = T_n(x)$
↑ same polynomial as we started with...

Note the idea of approximating a polynomial by a polynomial of lower degree makes sense and is call "economization." It can be used to construct efficient subroutines for computing functions such as e^x, $\sin x$ and $\cos x$...

This is a different than just neglecting the higher order terms... we don't discuss it further...

focus on terms ① and ②

① might be small

$$|f(x) - T_n(x)| \leq \underbrace{\frac{1}{(n+1)!}}_{①} \max_{t \in I} |f^{(n+1)}(t)| \underbrace{|x - x_0|^{n+1}}_{②}$$

① small when n is large

② small when x is close to x_0.

If the range of x's is specified how big does n have to be to meet an error tolerance?

If the degree n of the Taylor polynomial is specified how close does x have to be to meet an error tolerance?

Taylor theorem

Idea: Approximate a function f by a polynomial and then compute with the polynomial...

Sythetic division...

Example $T(x) = x^3 - 2x^2 + 2x + 3$

How to compute $T(2)$ efficiently?

Plug in and Count the operations

$$T(2) = 2^3 - 2\cdot 2^2 + 2\cdot 2 + 3$$

2 mult, 1 sub, 1 mult, 1 mult, 1 add, 1 add

Multiplications: 5 ← most time spent in multiplies

Add/Subtract: 3

Factor first $1\cdot x^3 - 2x^2 + 2x + 3$

$$(1x^2 - 2x + 2)x + 3$$

nested factors

$$((1x - 2)x + 2)x + 3$$

$$T(2) \approx ((1.2 - 2)2 + 2)2 + 3$$

mult mult

Multiplications: 2 ← actually 3 mult counting the green 1.

Add/Subtract: 3

Note: The optimizer in a compiler will not move the parethesis and refactor an algebraic expression involving floating point because the differences in rounding errors would change the answer (by a little bit? or lots?).

Example of algorithms that try to cancel round error is Kahan Summation...

Kahan summation algorithm

7 languages

Article Talk Tools

From Wikipedia, the free encyclopedia

In numerical analysis, the **Kahan summation algorithm**, also known as **compensated summation**,[1] significantly reduces the numerical error in the total obtained by adding a sequence of finite-precision floating-point numbers, compared to the obvious approach. This is done by keeping a separate *running compensation* (a variable to accumulate small errors), in effect extending the precision of the sum by the precision of the compensation variable.

Another idea to reduce rounding errors it to add the smaller numbers up first... less chances for loss of precision if the numbers added together have similar magnitudes...

Evaluate a polynomial using division...

$$x-2 \;\overline{)\; x^3 - 2x^2 + 2x + 3}$$

$$x^2 + 0x + 2 + \frac{7}{x-2}$$

$$x^3 - 2x^2$$

$$0x^2 + 2x + 3$$

$$0 + 0$$

$$2x + 3$$

$$2x - 4$$

$$7$$

Thus

$$x^3 - 2x^2 + 2x + 3 = (x^2 + 0x + 2)(x-2) + 7$$

$$T(x) = (x^2 + 0x + 2)(x-2) + 7$$

$$T(2) = (2^2 + 0\cdot 2 + 2)(2-2) + 7 = 7$$

don't need to compute this part

always 0 since I used $x-2$ as the divisor exactly because I wanted $T(2)$.

One you've done the division the answer $T(2)$ was just the remainder 7.

How quickly can you perform the division?

Synthetic division algorithm.

$$\frac{x^3 - 12x^2 - 42}{x - 3}$$

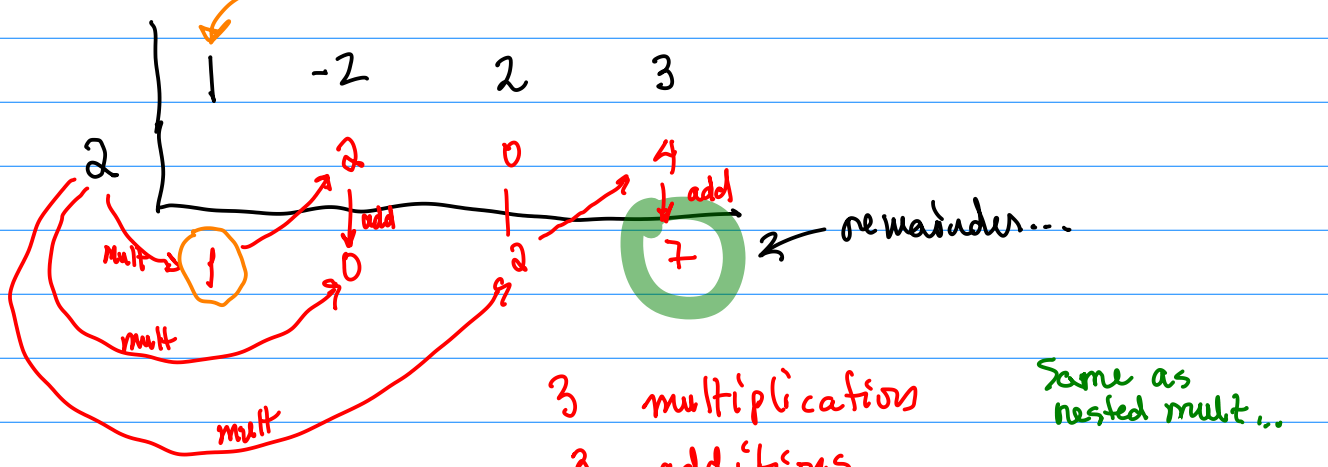
The numerator can be written as $p(x) = x^3 - 12x^2 + 0x - 42$.

The zero of the denominator $g(x)$ is 3.

The coefficients of $p(x)$ are arranged as follows, with the zero c

$$\begin{array}{r|llll} 3 & 1 & -12 & 0 & -42 \\ \hline & & & & \end{array}$$

Try this with our polynomial: $x^3 - 2x^2 + 2x + 3$



3 multiplications

3 additions

Same as nested mult...

For example, the values of the polynomial

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \qquad 2.$$

and its derivative

$$P'(x) = a_1 + 2a_2 x + \cdots + n a_n x^{n-1} \qquad 3.$$

for $x = \bar{x}$ may be generated recursively under the scheme:

$$p_k = p_{k-1}\bar{x} + a_{n-k}, \quad q_k = q_{k-1}\bar{x} + p_{k-1}; \quad k = 1, 2, \ldots, n \qquad 4.$$

with $p_0 = a_n$ and $q_0 = 0$.

Synthetic division or nested multiplication

something extra to think about.