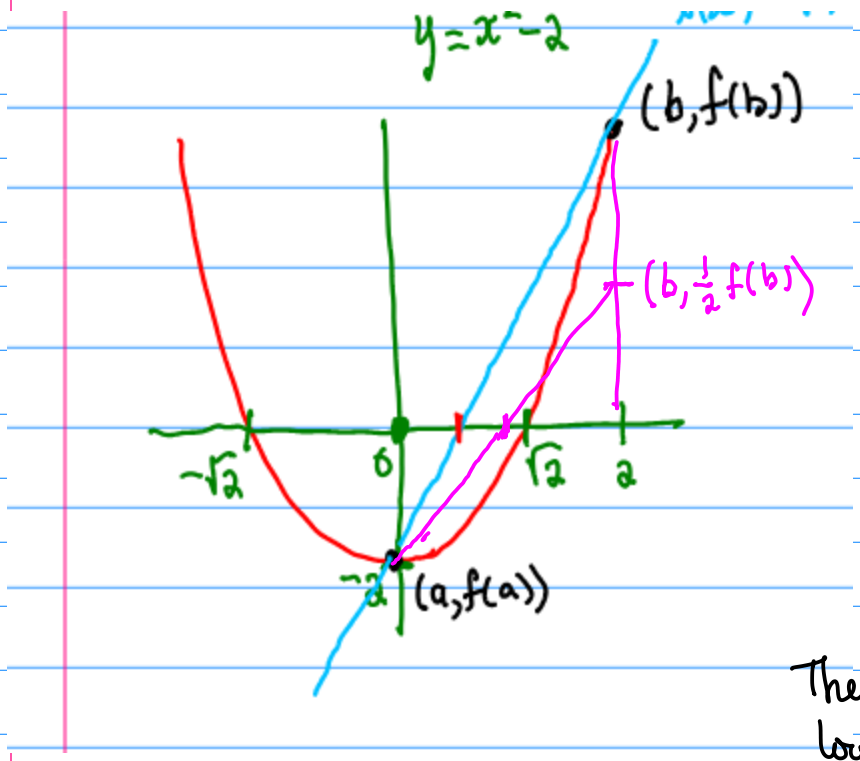


- Bisection
- False position
- Illinois algorithm

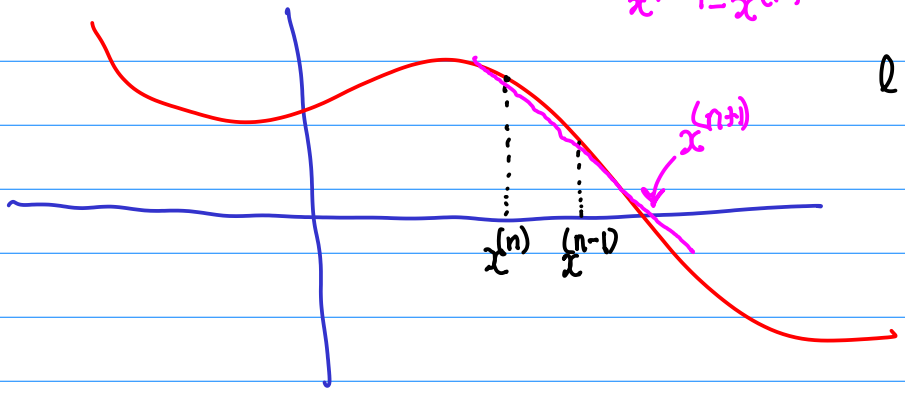
They all provide upper and lower bounds on the location of the root and are guaranteed to converge...

Can simplify things if we focus only on the approximating sequence for the root and forget about the intervals that contain the root.

Instead of a bracket containing the root, just use the last two approximations for the interpolation or extrapolation.

$$l(t) = \frac{(t - x^{(n)})}{x^{(n-1)} - x^{(n)}} f(x^{(n-1)}) + \frac{(t - x^{(n-1)})}{x^{(n)} - x^{(n-1)}} f(x^{(n)})$$

$l(t) = 0$ and solve



$$\frac{(t-x^{(n)})}{x^{(n-1)}-x^{(n)}} f(x^{(n-1)}) + \frac{(t-x^{(n-1)})}{x^{(n)}-x^{(n-1)}} f(x^{(n)}) = 0$$

$$\frac{t}{x^{(n-1)}-x^{(n)}} \left(f(x^{(n-1)}) - f(x^{(n)})\right) = \frac{1}{x^{(n-1)}-x^{(n)}} \left(x^{(n)} f(x^{(n-1)}) - x^{(n-1)} f(x^{(n)})\right)$$

$$t = \frac{x^{(n)} f(x^{(n-1)}) - x^{(n-1)} f(x^{(n)})}{f(x^{(n-1)}) - f(x^{(n)})}$$

since extrapolations can go wrong this has no guaranteed convergence to the root, but it's simpler ...

Therefore

factor out ... to avoid loss of precision

$$x^{(n+1)} = \frac{x^{(n)} f(x^{(n-1)}) - x^{(n-1)} f(x^{(n)})}{f(x^{(n-1)}) - f(x^{(n)})} = x^{(n)} + \text{something small ...}$$

$$\frac{x^{(n)} f(x^{(n-1)}) - x^{(n-1)} f(x^{(n)})}{f(x^{(n-1)}) - f(x^{(n)})} - x^{(n)} \frac{f(x^{(n-1)}) - f(x^{(n)})}{f(x^{(n-1)}) - f(x^{(n)})}$$

$$= \frac{x^{(n)} f(x^{(n-1)}) - x^{(n-1)} f(x^{(n)}) - x^{(n)} f(x^{(n-1)}) + x^{(n)} f(x^{(n)})}{f(x^{(n-1)}) - f(x^{(n)})}$$

$$= \frac{-x^{(n-1)} + x^{(n)}}{f(x^{(n-1)}) - f(x^{(n)})} f(x^{(n)}) = -\frac{x^{(n-1)} - x^{(n)}}{f(x^{(n-1)}) - f(x^{(n)})} f(x^{(n)})$$

still loss of precision ... but only in the small correction to the previous guess ...

$$x^{(n+1)} = x^{(n)} - \frac{x^{(n-1)} - x^{(n)}}{f(x^{(n-1)}) - f(x^{(n)})} f(x^{(n)})$$

keeping all precision in previous guess

This is called the Secant method.

How fast does it converge? How to measure it?

- How complicated is each iteration?
- How many iterations are needed?
- What is the total time using computer X writes programming language Y.

Order of convergence: Let $x^{(n)}$ be an approximating sequence such that $x^{(n)} \to x^*$ where x^* is the exact answer.

Let $e_n = x^{(n)} - x^*$. Suppose there is a constant K. large enough and $p \geq 1$ such that

$$|e_{n+1}| \leq K |e_n|^p \quad \text{for all } n.$$

Then the order is at least p.

If there is k and p such that

$$|e_{n+1}| \geq k |e_n|^p \quad \text{for all } n.$$

The the order is at most p.

Remark:

If $\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = C$ for some C the order is exactly p.

Questions: What is p for the secant method? $p = \frac{1+\sqrt{5}}{2} \approx 1.618$

p for the Illinois algorithm? $p = \sqrt[3]{3} \approx 1.442$

p for the bisection method? $p = 1 \approx 1$

$p = \frac{1+\sqrt{5}}{2} \approx 1.618$

Means 61.8% more significant digits per iteration. If 100 bits are correct then 61 bits more the next iteration.

$p = \sqrt[3]{3} \approx 1.442$

44.2% more significant digits per iteration.

$p = 1 \quad \approx 1$

Only one bit more per iteration. Even if 100 bits are correct, only one more is obtained on the next iteration.