MA3227 Numerical Analysis II

Lecture 12: Nonlinear Equations

Simon Etter

2019/2020

Nonlinear Equations

Introduction

Over the past four weeks, we discussed how to solve linear equations:

$$\text{Given } A \in \mathbb{R}^{N \times N} \text{ and } b \in \mathbb{R}^N,$$
$$\text{find } x \in \mathbb{R}^N \text{ such that } Ax = b \iff Ax - b = 0.$$

In this week, we will discuss how to solve nonlinear equations:

$$\text{Given a continuous function } f : \mathbb{R}^N \to \mathbb{R}^N,$$
$$\text{find } x \in \mathbb{R}^N \text{ such that } f(x) = 0.$$

As indicated, the right-hand side for nonlinear equations is usually assumed to be zero. This eliminates one parameter from the problem formulation and does not reduce generality since we can always rewrite

$$f(x) = b \iff f(x) - b = 0.$$

Terminology

- A point $x^\star$ such that $f(x^\star) = 0$ is called a *root* of f.
- Solving nonlinear equations is also called *root-finding*.

Nonlinear Equations

Introduction (continued)

Nonlinear equations differ from linear systems in a number of ways.

One-dimensional case

- A linear system with $N = 1$ reduces to the trivial equation $ax = b$.
- For nonlinear equations, already the case $N = 1$ can be nontrivial.

Existence and uniqueness of solutions

- For linear systems, we have existence and uniqueness if A is invertible.
- For nonlinear systems and $N = 1$, we can guarantee that a solution exists if we have $a, b \in \mathbb{R}$ such that $\operatorname{sign}\big(f(a)\big) = -\operatorname{sign}\big(f(b)\big)$. However, the solution may not be unique.
- For nonlinear systems and $N > 1$, we generally cannot guarantee that a solution exists.

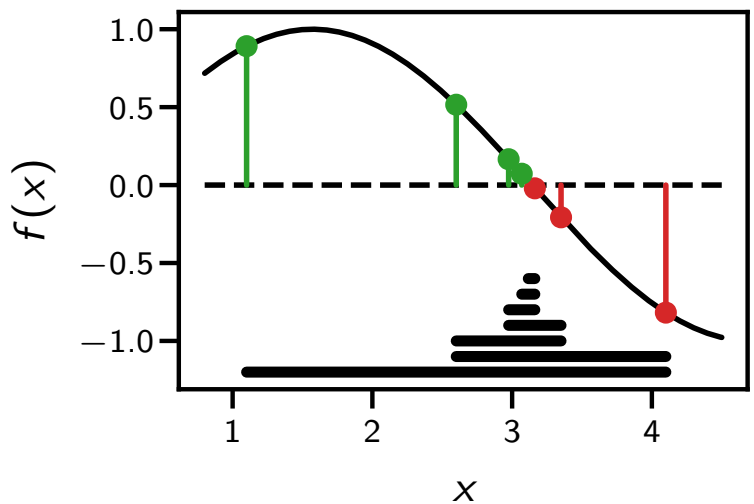
Nonlinear Equations

Bisection method

Let us begin with the simplest nonlinear case:
assume $f : \mathbb{R} \to \mathbb{R}$ is continuous and we have $a < b$ such that

$$\operatorname{sign}\big(f(a)\big) = -\operatorname{sign}\big(f(b)\big).$$

A simple algorithm for determining the root $x \in [a, b]$ is to recursively cut the search-interval $[a, b]$ in half. This is known as the bisection method.



Nonlinear Equations

Algorithm 1 Bisection method

1: **for** $k = 0, 2, 3, \ldots$ **do**
2: $\quad m_k = (b_k + a_k)/2$
3: $\quad$ **if** $\operatorname{sign}\big(f(a_k)\big) = \operatorname{sign}\big(f(m_k)\big)$ **then**
4: $\quad\quad a_{k+1} = m_k \quad$ and $\quad b_{k+1} = b_k$
5: $\quad$ **else**
6: $\quad\quad a_{k+1} = a_k \quad$ and $\quad b_{k+1} = m_k$
7: $\quad$ **end if**
8: **end for**

Strictly speaking, bisection does not determine a root $x^\star$ but a sequence of intervals $[a_k, b_k]$ bracketing a root of $f(x)$. If we take the midpoint m_k as the approximation to $x^\star$, then we have the simple error bound

$$|m_k - x^\star| \leq (b_0 - a_0)\, 2^{-k-1}.$$

This result is very powerful when combined with the fact that there are only a finite number of machine-representable numbers. See next slide.

Nonlinear Equations

Bisection method in finite arithmetic

Virtually all modern computers store real numbers using 64 bits.
We therefore conclude:

- There are only 2^{64} machine-representable numbers.
  In particular, the initial search interval $[a_0, b_0]$ passed to the bisection algorithm contains at most $n_0 \leq 2^{64}$ machine numbers.
- A good implementation of the bisection step can ensure that the number n_k of machine numbers in the interval $[a_k, b_k]$ is cut in half in every step, i.e. we have $n_k \leq \max\{n_0\, 2^{-k}, 2\}$.
- After at most $k = 63$ bisection steps, we therefore obtain a search interval $[a_k, b_k]$ which contains at most $n_k = 2$ machine numbers.
- $n_k = 2$ implies that $[a_k, b_k]$ is the tightest possible bracketing interval subject to the given set of machine numbers!

  We generally cannot hope for $n_k < 2$ because this would imply

$$\begin{aligned} a_k = b_k \quad &\Longleftrightarrow \quad \operatorname{sign}\big(f(a_k)\big) = -\operatorname{sign}\big(f(b_k)\big) \\ &\Longleftrightarrow \quad a_k = b_k = x^\star \text{ with } f(x^\star) = 0 \end{aligned}$$

  This can only happen if the root $x^\star$ is machine-representable.

Nonlinear Equations

Bisection method in finite arithmetic (continued)

We have shown on the previous slide:

> Bisection finds the tightest possible interval $[a_k, b_k]$ such that $\text{sign}\big(f(a)\big) = -\text{sign}\big(f(b)\big)$ using at most $\mathcal{O}(64)$ function evaluations!

$\mathcal{O}(64)$ means 64 $\pm$ something negligible.

I use this notation to avoid getting distracted by the details.

Thm: Optimality of bisection method

No root-finding algorithm can find $[a_k, b_k]$ as above using fewer than $\mathcal{O}(64)$ function evaluations for all $f(x)$.

Proof. See next slide.

Nonlinear Equations

Proof (continued).

A sample $f(x)$ tells us nothing about the value of $f(x')$ for any $x' \neq x$ unless we know that f is Lipschitz-continuous ($|f(x) - f(y)| \leq L\,|x - y|$) with some fixed Lipschitz constant L, which we generally do not. This in particular implies that the only information to be gained from a sample $f(x)$ is $\operatorname{sign}\big(f(x)\big)$.

Any root-finding algorithm can therefore be visualised as a binary tree where in each node we evaluate $f(x)$ at some point and then descend into one of the two branches depending on $\operatorname{sign}\big(f(x)\big)$.

For the algorithm to be correct, it must be able to output all $\mathcal{O}(2^{64})$ possible intervals $[a_k, b_k]$ with a_k, b_k being neighbouring machine numbers. This implies that the above tree must have at least $\mathcal{O}(2^{64})$ leaves.

Any binary tree with $\mathcal{O}(2^{64})$ leaves must have at least $\mathcal{O}(64)$ levels, which means that any algorithm must perform at least $\mathcal{O}(2^{64})$ function evaluations.

Remark: Bisection method = binary search over the real numbers.

Nonlinear Equations

Discussion

We have just seen that bisection is in some sense the possible best root-finding algorithm. Nevertheless, we will discuss other root-finding algorithms in the following. Reasons for doing so include:

- Other algorithms are faster than bisection for some $f(x)$.
  However, no algorithm can be more than $\mathcal{O}(64)$ times faster, and if it is faster for some f then it must perform worse on other f.
- Bisection does not generalise to $N > 1$ equations.

Runtime of root-finding algorithms

In most applications, the most time-consuming part in any root-finding algorithm is to evaluate the function $f(x)$ at the trial points.

For this reason, the computational cost of root-finding algorithms is usually measured in terms of number of function evaluations.

Nonlinear Equations

Root-finding by linear approximation

We will next discuss three root-finding algorithms based on linear approximation. The common idea is that if $f(x) \approx mx + b$, then $x = -\frac{b}{m}$ should be a good approximation to the exact root $x^\star$.

These three methods are obtained by varying

- how the linear approximation $f(x) = mx + b$ is determined, and
- how we proceed with the approximate root $x = -\frac{m}{b}$.

There are many more root-finding algorithms which I will not discuss. However, I will briefly comment on how the presented methods can be improved once we have finished discussing them.

Nonlinear Equations

False position method (sometimes also regula falsi)

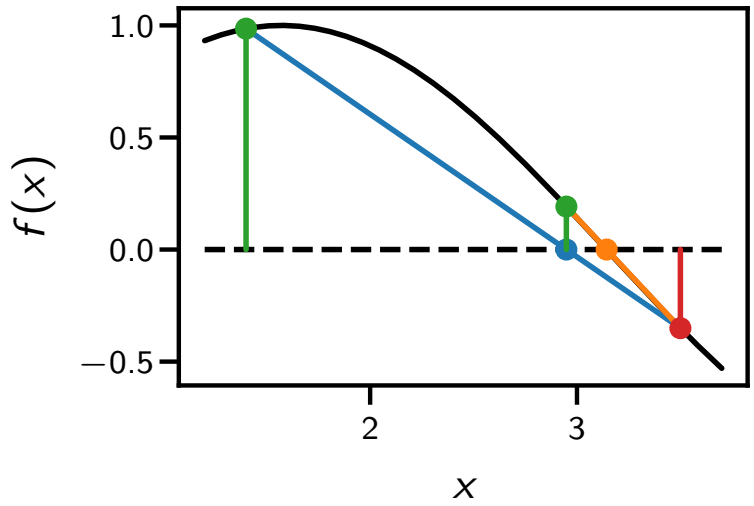
False position is a minor variation on the bisection method.

Instead of choosing the bisection point m_k as the midpoint in $[a_k, b_k]$, false position chooses it as the root of the linear interpolant $g(x)$ through $(a_k, f(x_k))$ and $(b_k, f(b_k))$. This interpolant is given by

$$g(x) = f(a_k)\,\tfrac{b_k - x}{b_k - a_k} + f(b_k)\,\tfrac{x - a_k}{b_k - a_k}$$

and thus

$$m_k = \frac{a_k\, f(b_k) - b_k\, f(a_k)}{f(b_k) - f(a_k)}.$$



Nonlinear Equations

Terminology

Bisection and false position are called "bracketing methods".

Implementation of bracketing methods

Bisection and false position may seem straightforward to implement, but unless you are careful you risk not getting the best possible answer or getting stuck in an infinite loop.

See `bracket()` and `test()`.

Nonlinear Equations

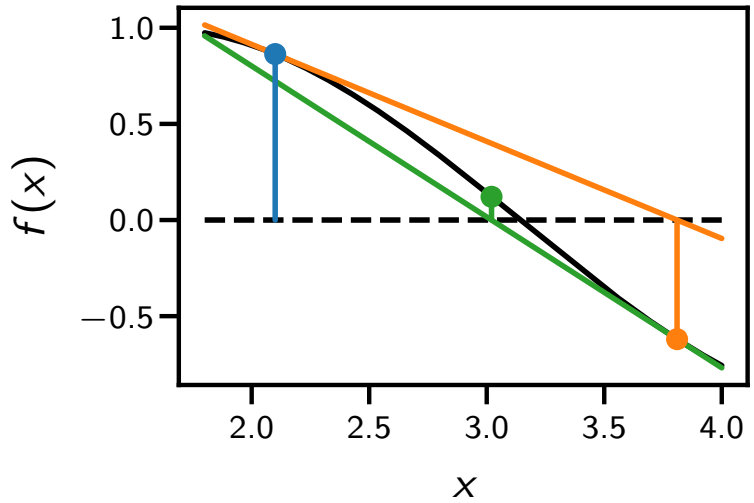
Newton's method (sometimes also Newton-Raphson method)

Newton's method generates a sequence of points x_k where the next iterate x_{k+1} is chosen as the root of the tangent $g(x)$ to $f(x)$ at the current point $x = x_k$. This tangent is given by

$$g(x) = f(x_k) + f'(x_k)\,(x - x_k);$$

hence x_{k+1} is given by

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



Nonlinear Equations

Secant method

A drawback of Newton's method is that it requires $f'(x)$ which may be difficult to compute.

The secant method avoids this issue by taking the linear interpolant $g(x)$ through $(x_k, f(x_k))$ and $(x_{k-1}, f(x_{k-1}))$ as an approximation to the tangent. This interpolant is given by

$$g(x) = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} (x - x_k)$$

and thus x_{k+1} is given by

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

The second formula is analogous to that for m_k in the false position method. False position and secant thus agree regarding the linear approximation, but they handle the resulting roots differently.

Nonlinear Equations

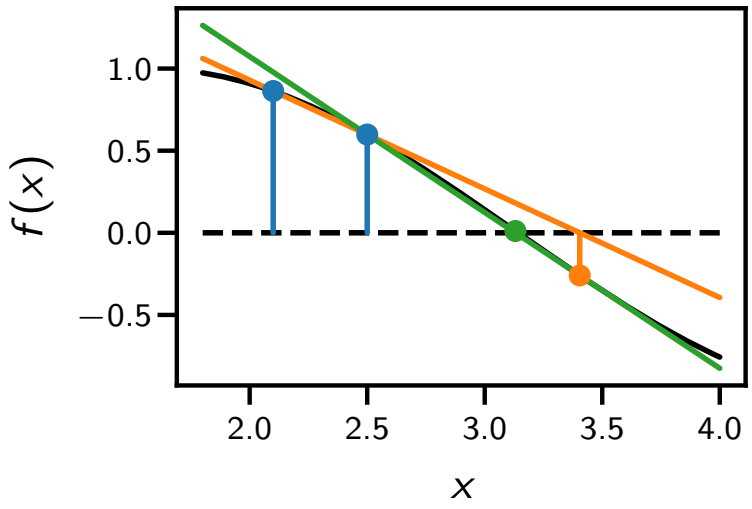
Secant method (continued)

Copied from previous slide:

$$g(x) = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} (x - x_k)$$

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

Illustration:



Nonlinear Equations

Discussion

We have seen:

- Bisection always terminates after a finite number of steps.
- Bisection always returns an answer which is as accurate as possible.
- No algorithm can have better worst-case performance than bisection.

We have also seen:

- False position returns an answer which is as accurate as possible.
- False position in principle terminates after a finite number of steps. However, this "finite" number may be too large to be practical.

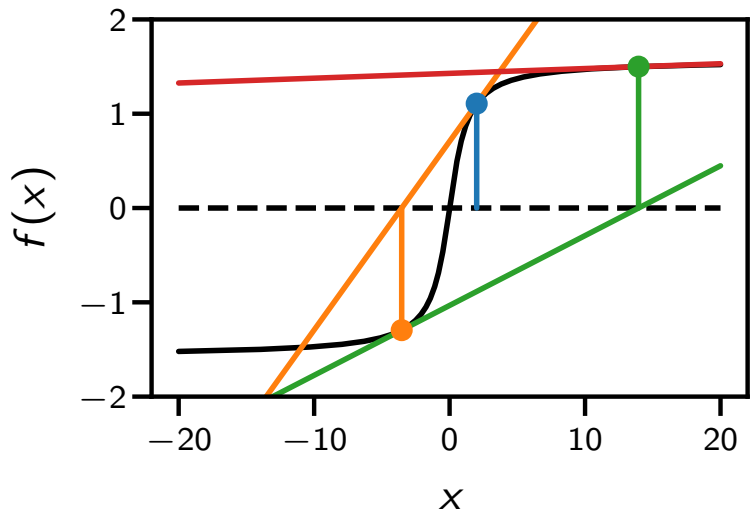
Furthermore, we can verify numerically that both Newton and secant diverge for $f(x) = \tan^{-1}(x)$ and large enough starting values.

See `divergence()` for a numerical demonstration, and see the next slide for an explanation.

Nonlinear Equations

Discussion (continued)

The following plot illustrates the first two Newton steps for $f(x) = \tan^{-1}(x)$ and an initial value $x_0 = 2$.



Nonlinear Equations

Convergence criteria for root-finders

Bisection has the nice property that it always terminates after a reasonable number of steps.

The above shows that we are now longer this fortunate for general root-finders. Instead, we need to provide a set of conditions which tell the computer that if any of these conditions are satisfied, then there is no point in continuing.

For example, the root-finders in the `Roots.jl` package accept the following arguments.

```
xatol - absolute tolerance for x values. [...]
xrtol - relative tolerance for x values. [...]
atol - absolute tolerance for f(x) values.
rtol - relative tolerance for f(x) values.
maxevals - limit on maximum number of iterations
maxfnevals - limit on maximum number of function evaluations
```

Nonlinear Equations

Introduction to error analysis

We have seen that linear approximation methods lose some of the reliability of the bisection method. The hope is that in return we get faster convergence for most functions $f(x)$ of practical interest.

We now study the asymptotic behaviour of $x_k - x$ as a function of k to see whether this hope is justified.

Our analysis will assume that $f(x)$ has as many continuous derivatives as required. Furthermore, we will make use of the Taylor expansions

$$f(x_k) = \underbrace{f(x^\star)}_{=0} + f'(x^\star)\,(x_k - x^\star) + \tfrac{1}{2}\,f''(x^\star)\,(x_k - x^\star)^2 + \mathcal{O}\big((\cdot)^3\big)$$

$$f'(x_k) = f'(x^\star) + f''(x^\star)\,(x_k - x^\star) + \mathcal{O}\big((\cdot)^2\big)$$

of $f(x)$ and $f'(x)$ around the root $x^\star$ approached by the root-finder.

I use $\mathcal{O}\big((\cdot)^\alpha\big)$ as a short-hand for $\mathcal{O}\big((x_k - x)^\alpha\big)$ to make sure the formulae fit on a line.

Consequently, our results will hold only locally, i.e. only for values x_k which are already close to the limiting root $x^\star$.

Nonlinear Equations

Convergence of Newton's method

Subtracting the exact solution $x^\star$ from both sides of the Newton formula and inserting the above Taylor-expansions for $f(x_k)$ and $f'(x_k)$ yields

$$\begin{aligned} x_{k+1} - x^\star &= x_k - x^\star - \frac{f(x_k)}{f'(x_k)} \\ &= x_k - x^\star - \frac{f'(x^\star)\,(x_k - x^\star) + \frac{1}{2}\,f''(x^\star)\,(x_k - x^\star)^2 + \mathcal{O}\big((\cdot)^3\big)}{f'(x^\star) + f''(x^\star)\,(x_k - x^\star) + \mathcal{O}\big((\cdot)^2\big)} \\ &= \frac{1}{2}\frac{f''(x^\star)}{f'(x^\star)}\,(x_k - x^\star)^2 + \mathcal{O}\big((\cdot)^3\big). \end{aligned}$$

Conclusion: $x_{k+1} - x^\star = \mathcal{O}\big((x_k - x)^2\big)$.

This is often called *quadratic convergence*.

Nonlinear Equations

Convergence of Newton's method (continued)

One way to interpret quadratic convergence is as follows:
If x_k has n correct digits, then x_{k+1} will have roughly $2n$ correct digits.

This is extremely rapid convergence:

- Newton requires $\mathcal{O}\big(\log_2(16)\big) = \mathcal{O}(4)$ steps to reduce a 10^{-1} initial error to 10^{-16}.
- Bisection requires $\mathcal{O}\big(\log_2(10^{15})\big) = \mathcal{O}(50)$ steps to achieve the same.

See `newton_convergence()` for further illustration.

Nonlinear Equations

Convergence of the secant method

Subtracting the exact solution $x^\star$ from both sides of the secant formula yields

$$x_{k+1} - x = x_k - x - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\, f(x_k).$$

Let us introduce $e_k = x_k - x^\star$.

We obtain using the Taylor series for $f(x)$ that

$$\begin{aligned} e_{k+1} &= e_k - \frac{(e_k - e_{k-1})\left(f'(x^\star)\, e_k + \frac{1}{2}\, f''(x^\star)\, e_k^2 + \mathcal{O}\big((\cdot)^3\big)\right)}{f'(x^\star)\,(e_k - e_{k-1}) + \frac{1}{2}\, f''(x^\star)\,(e_k^2 - e_{k-1}^2) + \mathcal{O}\big((\cdot)^3\big)} \\ &= e_k - \frac{f'(x^\star)\, e_k + \frac{1}{2}\, f''(x^\star)\, e_k^2 + \mathcal{O}\big((\cdot)^3\big)}{f'(x^\star) + \frac{1}{2}\, f''(x^\star)\,(e_k + e_{k-1}) + \mathcal{O}\big((\cdot)^2\big)} \\ &= \frac{1}{2}\, \frac{f''(x^\star)}{f'(x^\star)}\, e_k\, e_{k-1} + \mathcal{O}\big((\cdot)^3\big) \end{aligned}$$

Nonlinear Equations

Convergence of the secant method (continued)

Given our experience with Newton's method, we suspect that $e_{k+1} = \mathcal{O}(e_k^\alpha)$ for some $\alpha > 1$. Inserting this ansatz into the above formula yields

$$\mathcal{O}\big(e_k^\alpha\big) = \mathcal{O}\big(e_k^1\, e_k^{1/\alpha}\big).$$

Comparing powers, we obtain

$$\alpha = 1 + \tfrac{1}{\alpha} \qquad \Longleftrightarrow \qquad \alpha = \frac{1 \pm \sqrt{5}}{2}.$$

Thus, the error recursion for the secant method is

$$e_{k+1} = \mathcal{O}\big(e_k^\alpha\big) \qquad \text{where} \qquad \alpha = \frac{1+\sqrt{5}}{2} = 1.618\ldots.$$

Conclusion: secant converges more slowly than Newton's method, but the convergence is still super-linear and hence very fast.

See `secant_convergence()`.

Nonlinear Equations

Convergence of the secant method (continued)

Despite the above result, one may argue that the secant method is in fact faster than Newton's method:

Each iteration of Newton requires evaluating $f(x_k)$ and $f'(x_k)$, while secant requires only $f(x_k)$.

For every two function evaluations, Newton thus achieves $e_{k+1} = \mathcal{O}(e_k^2)$ while secant achieves $e_{k+2} = \mathcal{O}(e_k^{2\times 1.618\ldots}) = \mathcal{O}(e_k^{3.236\ldots})$.

Nonlinear Equations

Convergence of the false position method

We have seen that false position uses the same formula as secant to determine the next evaluation point, but then it handles the result differently.

In the worst case, false position only ever updates one of its endpoints which we may assume to be a_k without loss of generality.

Building on the error recursion formula for the secant method, we then obtain

$$a_k - x^\star = \frac{1}{2}\,\frac{f''(x^\star)}{f'(x^\star)}\,(b_k - x^\star)\,(a_k - x^\star) + \mathcal{O}\big((\cdot)^3\big).$$

This implies that $a_k \to x^\star$ converges exponentially with rate

$$\rho = \frac{1}{2}\,\frac{f''(x^\star)}{f'(x^\star)}\,(b_k - x^\star).$$

See `false_position_convergence()`.

Nonlinear Equations

Remark: nonzero derivatives

Throughout the above analysis, we have assumed that $f'(x^\star)$ and $f''(x^\star)$ are nonzero. As a rule of thumb, the convergence is better if $f'(x^\star) \neq 0$ but $f''(x^\star) = 0$, and the convergence is worse if $f'(x^\star) = 0$.

For example, for Newton's method one can easily show:

- $f'(x^\star) = 0$ but $f''(x^\star) \neq 0 \implies x_{k+1} - x^\star = \frac{x_k - x^\star}{2} + \mathcal{O}\big((\cdot)^2\big)$.
  See `newton_convergence_slow()`.
- $f'(x^\star) \neq 0$ but $f''(x^\star) = 0 \implies x_{k+1} - x^\star = \mathcal{O}\big((x_k - x^\star)^3\big)$.

Nonlinear Equations

Other rootfinding algorithms

There are many more root-finding algorithms than what I presented in this lecture. These methods typically pursue either of the following goals.

- Converge even faster than Newton.
- Combine different algorithms to obtain the speed of an advanced method with the reliability of bisection.

However:

- There is no point in trying to outperform Newton if Newton converges in four or five iterations, which is often the case.
- We have seen that no algorithm can be both as reliable as bisection and perform faster.

Nonlinear Equations

Summary

- Bisection is a very reliable and reasonably fast method for solving a single nonlinear equation. Its error satisfies $e_{k+1} \leq e_k/2$.
- Newton's method is given by $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$. Its error satisfies

$$e_{k+1} = \frac{1}{2}\frac{f''(x^\star)}{f'(x^\star)}\, e_k^2 + \mathcal{O}\big(e_k^3\big).$$

- Secant method is Newton's method with a finite-difference approximation for $f'(x)$. Its error satisfies

$$e_{k+1} = \frac{1}{2}\frac{f''(x^\star)}{f'(x^\star)}\, e_k^{1.618\ldots} + \mathcal{O}\big(e_k^{2\times 1.618\ldots}\big).$$

- False position is a mix between the bisection and secant methods. It in principle guarantees finite termination and can be faster than bisection, but it can also be much slower.