CLASS NOTES

Models, Algorithms and Data: Introduction to computing

2019

Petros Koumoutsakos, Jens Honore Walther, Julija Zavadlav

(Last update: March 8, 2019)

IMPORTANT DISCLAIMERS

Much of the material (ideas, definitions, concepts, examples, etc) in these notes is taken for teaching purposes, from several references:

- Numerical Analysis by R. L. Burden and J. D. Faires
- The Nature of Mathematical Modeling by N. Gershenfeld
- A First Course in Numerical methods by U. M. Ascher and C. Greif

These notes are only informally distributed and intended ONLY as study aid for the final exam of ETHZ students that were registered for the course Models, Algorithms and Data (MAD): Introduction to computing 2019. The notes have been checked, however they may still contain errors so use with care.

LECTURE 2

Nonlinear systems I: Bisection, Newton, Secant methods

2.1 Introduction

So far we have looked at equations and systems of equations where the unknown coefficients were entering linearly into the equations. Now, we consider general nonlinear equations and how to numerically solve such equations.

Example 2.1: Population growth

Let $N(t)$ be the number of people at time t. Let λ be the birth rate. Then the equation describing the population growth has the form:

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = \lambda N(t).$$

Assume that we also have an immigration at a constant rate u. Then the population growth equation will include an additional term:

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = \lambda N(t) + u.$$

The solution of the Eq. (2.1) is given by the following formula:

$$N(t) = N_0 e^{\lambda t} + \frac{u}{\lambda}(e^{\lambda t} - 1),$$

where N_0 is a constant indicating the initial population size.

Numerical example. Suppose we initially have 1,000,000 people. Assume that 15,000 people immigrate every year and that 1,080,000 people are present at the end of the year. What is the birth rate of this population? To find the birth rate we need to solve the following equation for λ (remember that $t = 1$ [year]):

$$1,080,000 = 1,000,000e^{\lambda} + \frac{15,000}{\lambda}(e^{\lambda} - 1).$$

We can reformulate this nonlinear equation into

$$1,000,000e^{\lambda} + \frac{15,000}{\lambda}(e^{\lambda} - 1) - 1,080,000 = 0.$$

A non-linear equation $g(x) = h(x)$ can always be written as $g(x) - h(x) = f(x) = 0$. Like this we transform the problem to finding a root (zero) or the function $f(x)$, i.e.

$$\boxed{f(x^*) = 0.} \tag{2.1.1}$$

This is therefore a **root-finding problem** (see Figure 2.1).

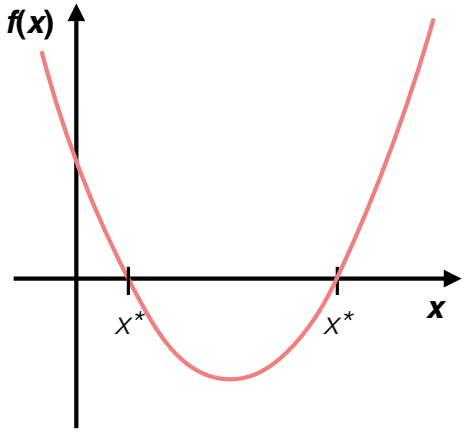

Figure 2.1: Function $f(x)$ is equal to zero at x^*.

In general, there is no formula to find a root of a non-linear function. There is even a proof that for polynomial functions of degree 5 or higher there is no algebraic expression for the roots (Abel–Ruffini theorem). For this reason, the non-linear equations are solved with **iterative schemes** that given initial guess $x^{(0)}$ generate a sequence $x^{(0)}, x^{(1)}, \ldots, x^{(k)}, \ldots$, which is usually written as $\{x^{(k)}\}_{k=0}^{\infty}$.

In practice, we only do finite amount of steps and terminate the algorithm, when some stopping criteria is satisfied. For example, when $|x^{(k)} - x^{(k-1)}| < \text{tol}$, where tol is a user defined tolerance. Therefore, we do not expect to obtain the solution x^* exactly.

2.2 Preliminaries

Before we discuss different algorithms for non-linear equations, we need to consider the following:

- When does a solution to our problem exist theoretically?
- Is there a ways to find this solution numerically?
- How fast will we reach the solution within some tolerance?

Provided there is a root, we want to find a root-finding algorithm to be: efficient (fast converging) and robust (rarely fails to find a solution).

Existence of a root

Recall that for linear problems finding the solution of $A\vec{x} = \vec{b}$ required that A is non-singular. To determine this criterion for a non-linear equation, we look at the Intermediate Value Theorem.

Intermediate Value Theorem:
If a function f is continuous on interval $[a,b]$, i.e. $f \in C([a,b])$, then for all K between $f(a)$ and $f(b)$, i.e. $K \in [f(a), f(b)]$, there exists a number $x^* \in (a,b)$ for which $f(x^*) = K$.
Consequence:
If a function $f(x)$ is continuous on interval $[a,b]$ with $f(a)$ and $f(b)$ of opposite sign, i.e., $f(a)f(b) < 0$ then according to the Intermediate Value Theorem there is a $x^* \in (a,b)$ with $f(x^*) = 0$.

Note: There is no simple analog of this theorem for the multi-dimensional case.

Sensitivity and Conditioning

Consider an approximate solution $\tilde{x}$ to $f(x) = 0$, where x^* is the true solution.

$$\left|f(\tilde{x})\right| \approx 0 \quad \overset{?}{\rightarrow} \quad \left|\tilde{x} - x^*\right| \approx 0, \tag{2.2.1}$$

As we have already seen in the linear setting a small residual $\left|f(\tilde{x})\right|$ implies an accurate solution (i.e., $\left|\tilde{x} - x^*\right| \approx 0$) only if the problem is well-conditioned. Note that this is very important, as we generally do not know the true solution x^* apriori and are only able to evaluate $\left|f(\tilde{x})\right|$.

well-conditioned A small change in the "input" causes a small change in the "output".

ill-conditioned A small change in the "input" causes a large change in the "output".

We can measure the behaviour of the function by the condition number:

$$\kappa = \frac{|\delta y|}{|\delta x|} = \frac{|f(x+\delta x) - f(x)|}{|\delta x|}, \tag{2.2.2}$$

where δx is the change in the input and $\delta y = f(x+\delta x) - f(x)$ is the change in the output for $y = f(x)$. Assuming that δx is small we can expand $f(x+\delta x)$ using a Taylor series $f(x+\delta x) \approx f(x) + f'(x)\delta x$, where we have neglected higher order terms. Therefore we can conclude that $\delta y \approx f'(x)\delta x$ and $\kappa = |f'(x)|$. Carefull, this is a condition number for the forward evaluation of the function, i.e. when given x we evaluate $y = f(x)$.

The root-finding problem is a reverse problem to function evaluation. For a root x^* of a function $f(x)$, we need to evaluate $x^* = f^{-1}(0)$, where f^{-1} is an inverse function. Using the previous result and the inverse function theorem we find the condition number for our root finding problem as

$$\kappa = \left|\frac{\partial}{\partial y}\left[f^{-1}(y)\right]\right| = \frac{1}{|f'(x^*)|}. \tag{2.2.3}$$

We see that for small $|f'(x^*)|$ the root-finding problem is ill-conditioned. Small $|f'(x^*)|$ means that the tangent line is nearly horizontal, see Figure 2.2.

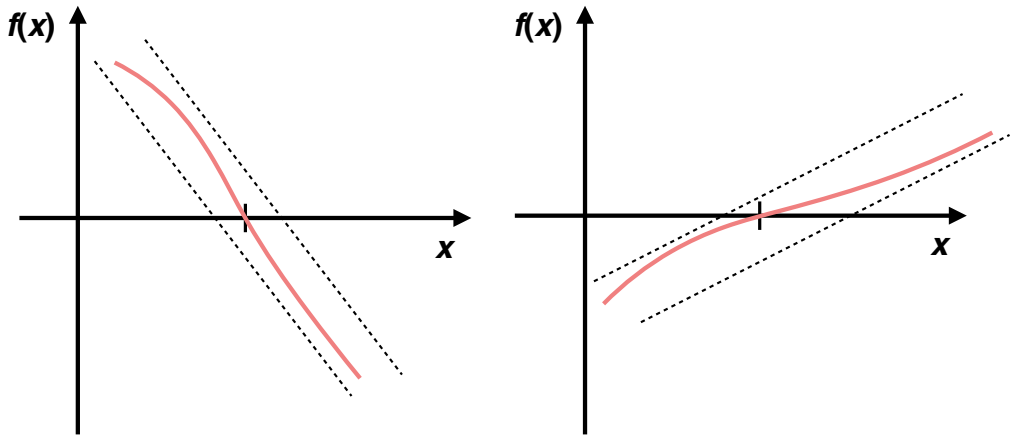

Figure 2.2: The well (left) and ill-conditioned (right) roots.

Note: If $f'(x^*) = 0$, the problem is ill-conditioned. This is the case for roots with multiplicity $m > 1$. Thus, roots with multiplicity $m > 1$ are ill-conditioned[1] (see Figure 2.3 for examples).

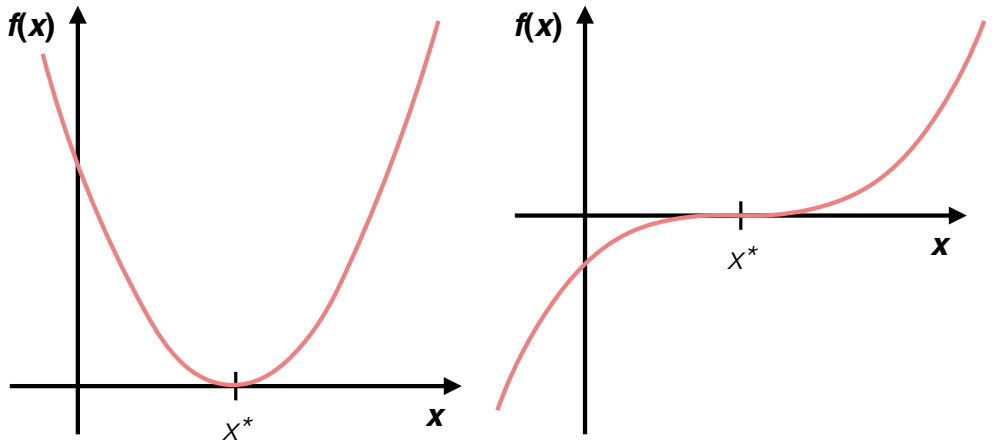

Figure 2.3: Graph of the function $f(x) = x^2 - 2x + 1 = (x-1)^2$ (left), which has a root $x^* = 1$ with $m = 2$ and $f(x) = x^3 - 3x^2 + 3x - 1 = (x-1)^3$ (right), which has a root $x^* = 1$ with $m = 3$.

[1] Recall that for a root of order k we have $f(x) = f'(x) = \cdots = f^{(k-1)}(x) = 0$ and $f^{(k)}(x) \neq 0$

Convergence Rate

We would like that the sequence $\{x^{(k)}\}_{k=0}^{\infty}$ produced by an algorithm converges to a true solution x^* as fast as possible. In order to quantify the rate of convergence we regard the error for $x^{(k)}$, given by

$$E^{(k)} = x^{(k)} - x^* \tag{2.2.4}$$

If the sequence of $x^{(k)} \overset{k\to\infty}{\longrightarrow} x^*$ we know that there is some r, such that the following exists

$$\lim_{k\to\infty} \frac{|E^{(k+1)}|}{|E^{(k)}|^r} = C \tag{2.2.5}$$

The value $r \in \mathbb{R}_{\geq 1}$ is the **order of convergence** and the constant $C \in [0,\infty)$ is the **convergence/asymptotic error constant**. Common cases according to the value of r:

- $r = 1$: linear convergence for $(0 < C < 1)$; if $C = 0$ superlinear; if $C = 1$ sublinear
- $r = 2$: quadratic

Example 2.2: Convergence Rates

1. $x^{(k)} = \frac{1}{2^k}$, $\lim_{k\to\infty} x^{(k)} = x^* = 0$

$$\lim_{k\to\infty} \frac{|x^{(k+1)}-x^*|}{|x^{(k)}-x^*|^r} = \lim_{k\to\infty} \frac{(2^k)^r}{2^{k+1}} = \begin{cases} \frac{1}{2}, & r = 1 \\ \infty, & r > 1 \end{cases} \longrightarrow \text{Linear}$$

2. $x^{(k)} = \frac{1}{k!}$, $\lim_{k\to\infty} x^{(k)} = x^* = 0$

$$\lim_{k\to\infty} \frac{|x^{(k+1)}-x^*|}{|x^{(k)}-x^*|^r} = \lim_{k\to\infty} \frac{(k!)^r}{(k+1)!} = \begin{cases} 0, & r = 1 \\ \infty, & r > 1 \end{cases} \longrightarrow \text{Superlinear}$$

3. $x^{(k)} = \frac{1}{k^2}$, $\lim_{k\to\infty} x^{(k)} = x^* = 0$

$$\lim_{k\to\infty} \frac{|x^{(k+1)}-x^*|}{|x^{(k)}-x^*|^r} = \lim_{k\to\infty} \frac{(k^2)^r}{(k+1)^2} = \begin{cases} 1, & r = 1 \\ \infty, & r > 1 \end{cases} \longrightarrow \text{Sublinear}$$

4. $x^{(k)} = \frac{1}{2^{2^k}}$, $\lim_{k\to\infty} x^{(k)} = x^* = 0$

$$\lim_{k\to\infty} \frac{|x^{(k+1)}-x^*|}{|x^{(k)}-x^*|^r} = \lim_{k\to\infty} \frac{(2^{2^k})^r}{2^{2^{k+1}}} = \lim_{k\to\infty} \frac{(2^{2^k})^r}{(2^{2^k})^2} = \begin{cases} 0, & 1 \leq r < 2 \\ 1, & r = 2 \\ \infty, & r > 2 \end{cases} \longrightarrow \text{Quadratic}$$

2.3 Bisection Method

The bisection method is based on the intermediate value theorem. The method works by beginning with an initial interval and then halving the length until a solution has been isolated as accurately as desired. Thereby the new interval is chosen such that the two points always have opposite sign (see Figure 2.4 and Algorithm 1).

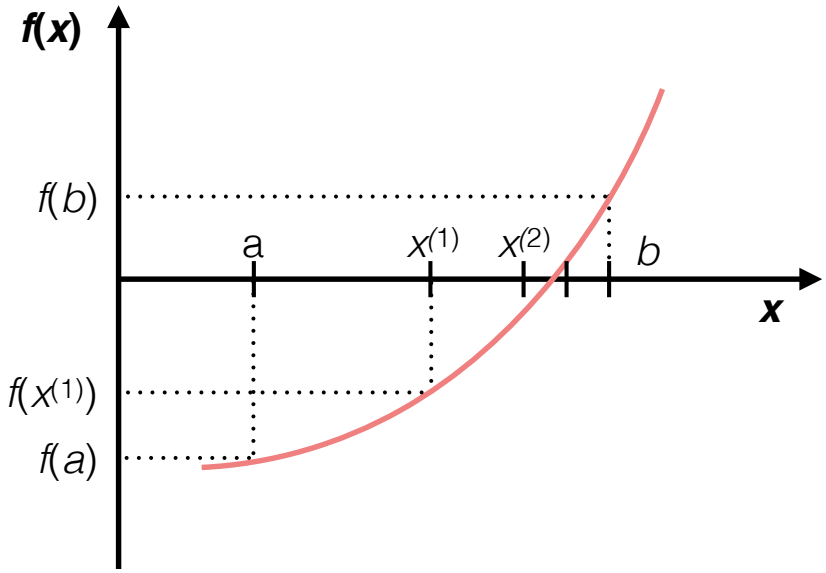

Figure 2.4: Iterative approach used in the bisection method.

Convergence rate

At each iteration the worst case for the error, namely the length of the interval is reduced by 1/2, i.e. $|E^{(k)}| = |x^{(k)} - x^*| \leq (b-a)/2^k$, where $[a,b]$ is the starting interval. Taking this worst case we find

$$\lim_{k\to\infty} \frac{|E^{(k+1)}|}{|E^{(k)}|^r} = \lim_{k\to\infty} \frac{2^{rk}}{2^{k+1}} = \begin{cases} \frac{1}{2}, & r = 1 \\ \infty, & \text{else} \end{cases} \tag{2.3.1}$$

Comparing Eq. (2.3.1) with Eq. (2.2.5) we observe that $r = 1$ and $C = 0.5$, i.e., the bisection method converges linearly. In terms of binary numbers, one bit of accuracy is gained in the approximate solution for each iteration of the bisection method.

Given the starting interval $[a,b]$ and an error tolerance tol, we can compute the required number of iterations k to achieve the required tolerance (regardless of f)

$$|E^{(k)}| = \text{tol} \quad \rightarrow \quad \frac{b-a}{2^k} = \text{tol} \quad \rightarrow \quad k = \log_2\left(\frac{b-a}{\text{tol}}\right). \tag{2.3.2}$$

Algorithm 1 Bisection Method

Input:

a, b, {initial interval}
tol, {tolerance}
$k_{\max}$, {maximal number of iterations}

Output:

$x^{(k)}$, {approximate solution after k iterations}

Steps:

$k \leftarrow 1$
while $\left(b-a\right) > tol$ and $k < k_{\max}$ **do**
  $x^{(k)} \leftarrow (a+b)/2$
  if $\text{sign}(f(a)) = \text{sign}(f(x^{(k)}))$ **then**
    $a \leftarrow x^{(k)}$
  else
    $b \leftarrow x^{(k)}$
  end if
  $k \leftarrow k+1$
end while

Pros

- The bisection method is certain to converge.
- The bisection method makes no use of actual function values, only of their signs.
- The function doesn't need to be differentiable, the only assumption on f is that it is continuous

Cons

- The convergence is slow.
- The initial interval needs to be known beforehand.
- Can not be easily generalized to higher dimensions and many unknowns

2.4 Newton's Method

Assume a function f is differentiable and has a zero at x^*. Furthermore, let $x^{(k)}$ be an approximation to x^* such that $f'(x^{(k)}) \neq 0$ and $|x^* - x^{(k)}|$ is small. We can then expand the function f about x^* using

Taylor series

$$0 = f(x^*) \approx f(x^{(k)}) + f'(x^{(k)})(x^* - x^{(k)}). \tag{2.4.1}$$

Solving the Eq. (2.4.1) for x^* only gives the true solution for linear functions, since we did only keep the terms up to this order

$$x^* \approx x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}. \tag{2.4.2}$$

Nonetheless we will use this results in order to update $x^{(k)}$. Given the initial guess $x^{(0)}$ Newton's method generates a sequence $\{x^{(k)}\}_{k=0}^{\infty}$ where

$$\boxed{x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}}. \tag{2.4.3}$$

Graphically the procedure is visualized in Figure 2.5. Newton's method approximates the non-linear function f by the tangent line to f at $x^{(k)}$, which can be easily seen when realizing that $f'(x^{(0)}) = \tan(\theta) = \frac{f(x^{(0)})}{x^{(0)} - x^{(1)}}$.

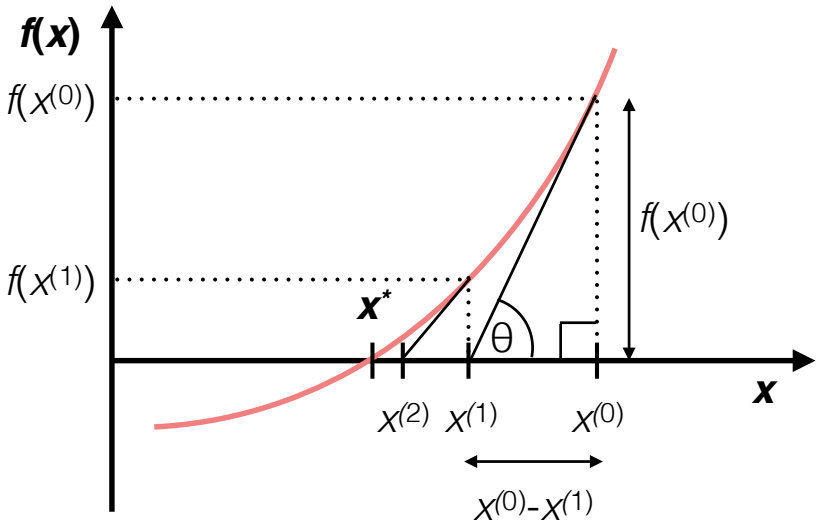

Figure 2.5: Graphical example of Newton iterations as we follow the tangents of $f(x)$.

Convergence rate

In order to compute the error let us consider the case of a differentiable function f on some interval $[a,b]$ that has a simple root $x^* \in [a,b]$ ($f(x^*) = 0$ and $f'(x^*) \neq 0$). Expanding this function in Taylor

series around the root gives

$$\begin{aligned}
\overbrace{f(x^*)}^{0} &\approx f(x^{(k)}) + f'(x^{(k)})(x^* - x^{(k)}) + \frac{1}{2}f''(x^{(k)})(x^* - x^{(k)})^2 \\
0 &\approx \underbrace{\frac{f(x^{(k)})}{f'(x^{(k)})} - x^{(k)}}_{-x^{(k+1)}} + x^* + \frac{f''(x^{(k)})}{2f'(x^{(k)})}(x^* - x^{(k)})^2 \\
0 &\approx \underbrace{x^* - x^{(k+1)}}_{-E^{(k+1)}} + \frac{f''(\xi)}{2f'(x^{(k)})}\underbrace{(x^* - x^{(k)})^2}_{[E^{(k)}]^2} \\
E^{(k+1)} &\approx \frac{f''(\xi)}{2f'(x^{(k)})}[E^{(k)}]^2
\end{aligned} \tag{2.4.4}$$

In the limit $k \to \infty$ this allows us to compute the convergence rate as

$$\lim_{k\to\infty} \frac{|E^{(k+1)}|}{|E^{(k)}|^2} = \lim_{k\to\infty} \frac{|f''(x^{(k)})|}{2|f'(x^{(k)})|} = \frac{|f''(x^*)|}{2|f'(x^*)|} = C < \infty \quad \to \quad r = 2 \tag{2.4.5}$$

Thus in this case Newton's method will converge quadratically. It turns out that for roots of multiplicity $m > 1$, the convergence (if the method is converging) is linear unless we modify the iteration (Eq. (2.4.3)) to $x^{(k+1)} = x^{(k)} - mf(x^{(k)})/f'(x^{(k)})$.

Pros

- Quadratic convergence

Cons

- The method is not guaranteed to converge. It is sensitive to initial conditions.
- If for some k the $f'(x^{(k)}) = 0$ we cannot proceed.
- Newton's method requires the evaluation of both function and derivative at each iteration.

2.5 Secant Method

Evaluating the derivative may be expensive or inconvenient. The key idea of the Secant method is to replace the derivative in Newton's method with a numerical approximation. Let

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}} \tag{2.5.1}$$

then the method reads as

$$\boxed{x^{(k+1)} = x^{(k)} - f(x^{(k)})\frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}} \tag{2.5.2}$$

The secant method approximates the non-linear function f by a secant line through previous two iterations (see Figure 2.6).

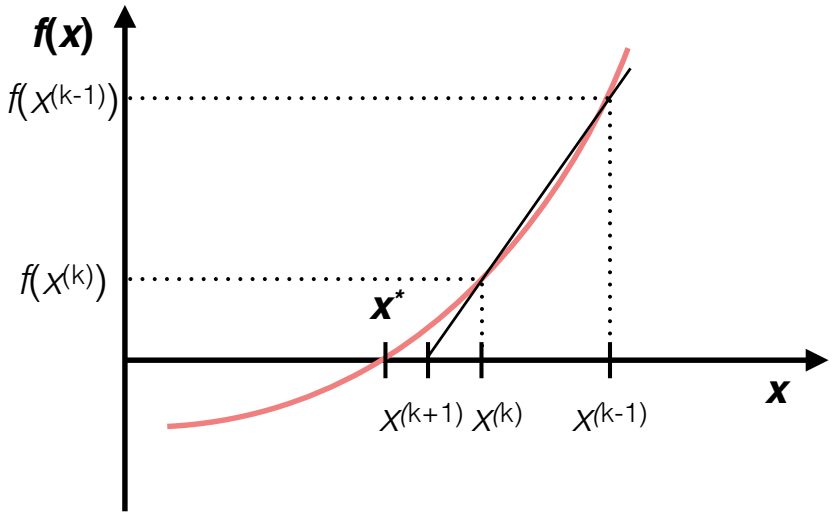

Figure 2.6: Iterative approach used in the secant method.

Convergence rate

It can be shown that the convergence rate of the Secant method for simple roots is $r \approx 1.618$, i.e., between linear and quadratic[2]. As we saw above the secant method uses a linear approximation of the function to construct its derivative. This idea can be extended so that higher order interpolations. Quadratic interpolation (Müller's method) has convergence $r \approx 1.839$.

Pros

- We avoid the evaluation of the derivative.
- At each step, we need to perform 1 function evaluation.

Cons

- The convergence rate is not quadratic.
- We need two initial approximations.

[2]For the interested readers more details about the convergence of the secant method can be found in P. Díez, A note on the convergence of the secant method for simple and multiple roots, Applied Mathematics Letters, Volume 16, Issue 8, 2003, Pages 1211-1215, https://doi.org/10.1016/S0893-9659(03)90119-4

Example 2.3: Comparing Bisection, Newton and Secant methods

Consider the function $f(x)$ given by

$$f(x) = 2\cosh(x/4) - x$$

The Bisection (starting with [0,10]), Newton's (with $x^{(0)} = 8$) and Secant methods (with $x^{(0)} = 10$ and $x^{(1)} = 8$) will give us the following sequence

k	0	1	2	3	4	5	6
Bisection: $f(x^{(k)})$	-1.22	0.96e-1	0.85	0.35	0.12	0.95e-2	-0.43e-1
Newton: $f(x^{(k)})$	-4.76e-1	8.43e-2	1.56e-3	5.65e-7	7.28e-14	1.78e-15	
Secant: $f(x^{(k)})$	2.26	-4.76e-1	-1.64e-1	2.45e-2	-9.93e-4	-5.62e-6	1.30e-9

We see that for the Bisection the error is approximately half of the error in the previous step. For Newton, the accurate digits essentially double at each iteration, while for the Secant method the accurate digits increase more than linearly but less than quadratically.

Exam checklist

- What are the key algorithms for solving non-linear equations?
- How do we define a convergence rate of an algorithm?
- When is a root-finding problem ill-conditioned?
- Which statements are true for investigated algorithms. (i) The algorithm is efficient. (ii) The algorithm always finds a solution. (iii) The algorithm requires a continuous function. (iv) The algorithm requires an evaluation of the derivative of the function.
- Suppose Newton's method does not converge. Does this mean that there is no root in the vicinity of the starting point?
- When is Newton's method converging linearly?
- When to use derivatives or their approximation in Newton's methods?
- What is a graphical interpretation of Newton's and Secant methods?
- How fast will Newton's/Secant method converge for a linear function? Will it converge for any initial value?
- Can you suggest a method that will in part overcome the disadvantage of Bisection in terms of slow convergence and at the same time overcome the disadvantage of Newton's method of local convergence?

Exercises

Use Newton's method to find the roots of the following nonlinear functions.

1. $f(x) = x^3 - 2x^2 - 11x + 12$ using different initial values $x^{(0)} = 2.352875270, 2.352836327$, and 2.352836323.
   Solution:
   Using different initial conditions, the method will converge to different roots: $4, -3$ and 1. This example, demonstrates Newton's method sensitivity to initial conditions.

2. $f(x) = x^3 - 2x^2 + 2$ using initial value $x^{(0)} = 0$.
   Solution:
   We obtain a sequence: $0, 1, 0, 1, \ldots$ Newton's method is stuck in a cycle.

3. $f(x) = x^2$ using initial value $x^{(0)} = 1$.
   Solution:
   The first derivative is $f'(x) = 2x$, which is 0 at x^*. The second derivative is $f''(x) = 2$. The iteration update is in this case:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} = x^{(k)} - \frac{[x^{(k)}]^2}{2x^{(k)}} = \frac{x^{(k)}}{2}$$

   From Eq. (2.4.5) we see that

$$\lim_{x \to 0} \frac{|f''(x)|}{2|f'(x)|} = \lim_{x \to 0} \frac{2}{2|2x|} = \infty$$

   This example, demonstrates the linear convergence for Newton's method in the case of a multiple root ($f'(x^*) = 0$).