Solution of nonlinear equations

Goal: find the roots (or zeroes) of a nonlinear function:

$$\text{given } f : [a, b] \to \mathbb{R}, \text{ find } \alpha \in \mathbb{R} \text{ such that } f(\alpha) = 0.$$

Various applications, e.g. optimization: finding stationary points of a function leads to compute the roots of f'.

Solution of nonlinear equations

Goal: find the roots (or zeroes) of a nonlinear function:

$$\text{given } f : [a, b] \to \mathbb{R}, \text{ find } \alpha \in \mathbb{R} \text{ such that } f(\alpha) = 0.$$

Various applications, e.g. optimization: finding stationary points of a function leads to compute the roots of f'.

When f is linear (and its graphic is a straight line) the problem is very easy. But when the analytic expression of f is more complicated, even though we have an idea of the location of its roots (with the help of graphics), we are unable to compute them exactly. Even finding the roots of polynomials of higher degree is difficult.

Solution of nonlinear equations

Goal: find the roots (or zeroes) of a nonlinear function:

$$\text{given } f : [a, b] \to \mathbb{R}, \text{ find } \alpha \in \mathbb{R} \text{ such that } f(\alpha) = 0.$$

Various applications, e.g. optimization: finding stationary points of a function leads to compute the roots of f'.

When f is linear (and its graphic is a straight line) the problem is very easy. But when the analytic expression of f is more complicated, even though we have an idea of the location of its roots (with the help of graphics), we are unable to compute them exactly. Even finding the roots of polynomials of higher degree is difficult.

All the methods available are iterative: starting from an initial guess $x^{(0)}$, we construct a sequence of approximate solutions $x^{(k)}$ such that

$$\lim_{k \to \infty} x^{(k)} = \alpha.$$

Questions/comments regarding iterative methods:

Questions/comments regarding iterative methods:

- Does the sequence converge?

Questions/comments regarding iterative methods:

- Does the sequence converge?
- Does convergence depend on the initial guess $x^{(0)}$? (In general, yes)

Questions/comments regarding iterative methods:

- Does the sequence converge?
- Does convergence depend on the initial guess $x^{(0)}$? (In general, yes)
- How fast is the convergence? (or: what is the *order of convergence*?)

Questions/comments regarding iterative methods:

- Does the sequence converge?
- Does convergence depend on the initial guess $x^{(0)}$? (In general, yes)
- How fast is the convergence? (or: what is the *order of convergence*?)
- When to stop the procedure? (how many iterations should we do? need for reliable stopping criteria)

Questions/comments regarding iterative methods:

- Does the sequence converge?
- Does convergence depend on the initial guess $x^{(0)}$? (In general, yes)
- How fast is the convergence? (or: what is the *order of convergence*?)
- When to stop the procedure? (how many iterations should we do? need for reliable stopping criteria)

Bisection method

Simplest and robust method, based on the intermediate value theorem:

Questions/comments regarding iterative methods:

- Does the sequence converge?
- Does convergence depend on the initial guess $x^{(0)}$? (In general, yes)
- How fast is the convergence? (or: what is the *order of convergence*?)
- When to stop the procedure? (how many iterations should we do? need for reliable stopping criteria)

Bisection method

Simplest and robust method, based on the intermediate value theorem:

Theorem

(Bolzano) *Let* $f : [a, b] \to \mathbb{R}$ *be a continuous function that has opposite signs in* $[a, b]$ *(meaning, to be precise, that* $f(a)f(b) < 0$*). Then there exists* $\alpha \in]a, b[$ *such that* $f(\alpha) = 0$*.*

Questions/comments regarding iterative methods:

- Does the sequence converge?
- Does convergence depend on the initial guess $x^{(0)}$? (In general, yes)
- How fast is the convergence? (or: what is the *order of convergence*?)
- When to stop the procedure? (how many iterations should we do? need for reliable stopping criteria)

Bisection method

Simplest and robust method, based on the intermediate value theorem:

Theorem

(Bolzano) *Let $f : [a, b] \to \mathbb{R}$ be a continuous function that has opposite signs in $[a, b]$ (meaning, to be precise, that $f(a)f(b) < 0$). Then there exists $\alpha \in]a, b[$ such that $f(\alpha) = 0$.*

Note that the root α does not need to be unique (take $f(x) = cos(x)$ on $[0, 3\pi]$). Hence, under the hypotheses of Bolzano's theorem, we will look for *a root of the equation* essentially without choosing which one.

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

- divide the interval in two by computing the midpoint c of the interval: $c = (a + b)/2$;

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

- divide the interval in two by computing the midpoint c of the interval: $c = (a + b)/2$;
- compute the value $f(c)$;

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

- divide the interval in two by computing the midpoint c of the interval: $c = (a + b)/2$;
- compute the value $f(c)$;
- if $f(c) = 0$ we found the root (very unlikely!),

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

- divide the interval in two by computing the midpoint c of the interval: $c = (a + b)/2$;
- compute the value $f(c)$;
- if $f(c) = 0$ we found the root (very unlikely!),
- if not, that is, if $f(c) \neq 0$, there are two possibilities:

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

- divide the interval in two by computing the midpoint c of the interval: $c = (a+b)/2$;
- compute the value $f(c)$;
- if $f(c) = 0$ we found the root (very unlikely!),
- if not, that is, if $f(c) \neq 0$, there are two possibilities:
    - $f(a)f(c) < 0$ (and then f has opposite signs on $[a, c]$),

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

- divide the interval in two by computing the midpoint c of the interval: $c = (a+b)/2$;
- compute the value $f(c)$;
- if $f(c) = 0$ we found the root (very unlikely!),
- if not, that is, if $f(c) \neq 0$, there are two possibilities:
  - $f(a)f(c) < 0$ (and then f has opposite signs on $[a, c]$),
  - or $f(b)f(c) < 0$ (and then f has opposite signs on $]c, b[$).

Bisection method

Idea: to construct a sequence by repeatedly bisecting the interval and selecting to proceed the sub-interval where the function has opposite signs. In the hypotheses of the intermediate value Theorem, we will

- divide the interval in two by computing the midpoint c of the interval: $c = (a + b)/2$;
- compute the value $f(c)$;
- if $f(c) = 0$ we found the root (very unlikely!),
- if not, that is, if $f(c) \neq 0$, there are two possibilities:
  - $f(a)f(c) < 0$ (and then f has opposite signs on $[a, c]$),
  - or $f(b)f(c) < 0$ (and then f has opposite signs on $]c, b[$).

The method selects the subinterval where f has opposite signs as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step. The process is continued until the interval is sufficiently small.

Bisection method: Algorithm

INPUT: Function f, endpoints a, b, tolerance TOL, max # iterations NMAX

Bisection method: Algorithm

INPUT: Function f, endpoints a, b, tolerance TOL, max # iterations NMAX
CONDITIONS: $a < b$, either $f(a) < 0$ and $f(b) > 0$ or $f(a) > 0$ and $f(b) < 0$ (or simply check that $f(a)f(b) < 0$)

Bisection method: Algorithm

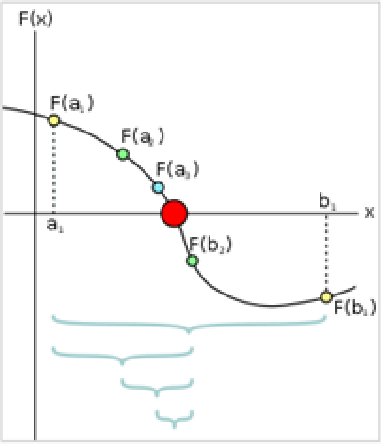
INPUT: Function f, endpoints a, b, tolerance TOL, max # iterations NMAX
CONDITIONS: $a < b$, either $f(a) < 0$ and $f(b) > 0$ or $f(a) > 0$ and $f(b) < 0$ (or simply check that $f(a)f(b) < 0$)
OUTPUT: value which differs from a root of f(x)=0 by less than TOL

Bisection method: Algorithm

INPUT: Function f, endpoints a, b, tolerance TOL, max # iterations NMAX
CONDITIONS: $a < b$, either $f(a) < 0$ and $f(b) > 0$ or $f(a) > 0$ and $f(b) < 0$ (or simply check that $f(a)f(b) < 0$)
OUTPUT: value which differs from a root of f(x)=0 by less than TOL

N =1
While N $\leq$NMAX (limit iterations to prevent infinite loop)
$c = (a + b)/2$ (new midpoint)
  If $f(c) = 0$ or $(b - a)/2 < TOL$ then (solution found)
  Output (c)
  Stop
  End
N = N + 1 (increment step counter)
If $sign(f(c)) = sign(f(a))$ then $a = c$ else $b = c$ (new interval)
End
Output("Method failed.") max number of steps exceeded

Bisection method: Example



A few steps of the bisection method applied over the starting range $[a_1;b_1]$. The bigger red dot is the root of the function.

Bisection method: Analysis

In the hypotheses of Bolzano's theorem (f continuous with opposite signs at the endpoints of its interval of definition) the bisection method **converges always** to a root of f, but it is slow: the absolute value of the error is halved at each step, that is, the method converges linearly.

Bisection method: Analysis

In the hypotheses of Bolzano's theorem (f continuous with opposite signs at the endpoints of its interval of definition) the bisection method **converges always** to a root of f, but it is slow: the absolute value of the error is halved at each step, that is, the method converges linearly.

If c_1 is the midpoint of [a,b], and c_k is the midpoint of the interval at the k^{th} step, the error is bounded by

$$|c_k - \alpha| \leq \frac{b-a}{2^k}$$

Bisection method: Analysis

In the hypotheses of Bolzano's theorem (f continuous with opposite signs at the endpoints of its interval of definition) the bisection method **converges always** to a root of f, but it is slow: the absolute value of the error is halved at each step, that is, the method converges linearly.

If c_1 is the midpoint of [a,b], and c_k is the midpoint of the interval at the k^{th} step, the error is bounded by

$$|c_k - \alpha| \leq \frac{b-a}{2^k}$$

This relation can be used to determine in advance the number of iterations needed to converge to a root within a given tolerance:

$$\frac{b-a}{2^k} \leq TOL \implies k \geq \log_2(b-a) - \log_2 TOL$$

Bisection method: Analysis

In the hypotheses of Bolzano's theorem (f continuous with opposite signs at the endpoints of its interval of definition) the bisection method **converges always** to a root of f, but it is slow: the absolute value of the error is halved at each step, that is, the method converges linearly.

If c_1 is the midpoint of [a,b], and c_k is the midpoint of the interval at the k^{th} step, the error is bounded by

$$|c_k - \alpha| \leq \frac{b-a}{2^k}$$

This relation can be used to determine in advance the number of iterations needed to converge to a root within a given tolerance:

$$\frac{b-a}{2^k} \leq TOL \implies k \geq \log_2(b-a) - \log_2 TOL$$

Ex: $b - a = 1$, TOL$= 10^{-3}$ gives $k \geq 3\log_2 10$, TOL$= 10^{-4}$ gives $k \geq 4\log_2 10$ and so on. Since $\log_2 10 \simeq 3.32$, to gain one order of accuracy we need a little more than 3 iterations.

Newton's method

For each iterate x_k, the function f is approximated by its tangent in x_k:

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

Then we impose that the right-hand side is 0 for $x = x_{k+1}$. Thus,

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

Newton's method

For each iterate x_k, the function f is approximated by its tangent in x_k:

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

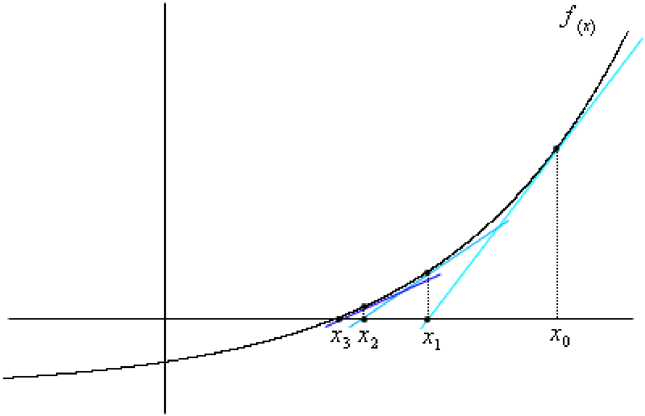
Then we impose that the right-hand side is 0 for $x = x_{k+1}$. Thus,

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

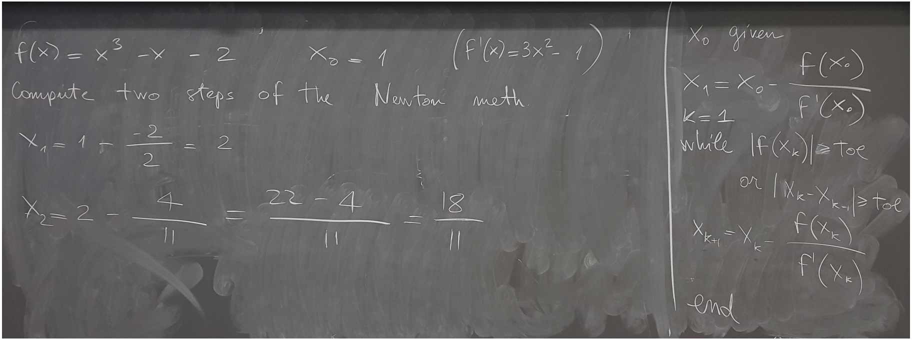
More assumptions needed on f:

- f must be differentiable, and f' must not vanish.
- the initial guess x_0 must be chosen well, otherwise the method might fail
- suitable stopping criteria have to be introduced to decide when to stop the procedure (no intervals here.......).

Example



Exercise


$f(x) = x^3 - x - 2 \qquad x_0 = 1 \qquad (f'(x) = 3x^2 - 1)$
Compute two steps of the Newton meth.
$x_1 = 1 - \frac{-2}{2} = 2$
$x_2 = 2 - \frac{4}{11} = \frac{22-4}{11} = \frac{18}{11}$
x_0 given
$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$
$k = 1$
while $|f(x_k)| \geq tol$
or $|x_k - x_{k-1}| \geq tol$
$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$
end

Newton's method: Convergence theorem

Theorem

Let $f \in C^2([a,b])$ such that:

1. $f(a)f(b) < 0 \qquad (*)$
2. $f'(x) \neq 0 \quad \forall x \in [a,b] \qquad (**)$
3. $f''(x) \neq 0 \quad \forall x \in [a,b] \qquad (***)$

Let the initial guess x_0 be a Fourier point (i.e., a point where f and f'' have the same sign). Then Newton sequence

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad k = 0, 1, 2, \cdots \tag{1}$$

converges to the **unique** *α such that $f(\alpha) = 0$. Moreover, the order of convergence is 2, that is:*

$$\exists C > 0 : \quad |x_{k+1} - \alpha| \leq C|x_k - \alpha|^2. \tag{2}$$

Newton's method: Proof of the Theorem

Proof.

Since f is continuous and has opposite signs at the endpoints then the equation $f(x) = 0$ has at least one solution, say α. Moreover condition (**) implies that α is unique (f is monotone).

Newton's method: Proof of the Theorem

Proof.

Since f is continuous and has opposite signs at the endpoints then the equation $f(x) = 0$ has at least one solution, say α. Moreover condition (**) implies that α is unique (f is monotone).

To prove convergence, let us assume for instance that f is as follows: $f(a) < 0$, $f(b) > 0$, $f' > 0$, $f'' > 0$, so that the initial guess x_0 is any point where $f(x_0) > 0$. We shall prove that Newton's sequence $\{x_n\}$ is a monotonic decreasing sequence bounded by below.

Newton's method: Proof of the Theorem

Proof.

Since f is continuous and has opposite signs at the endpoints then the equation $f(x) = 0$ has at least one solution, say α. Moreover condition (**) implies that α is unique (f is monotone).

To prove convergence, let us assume for instance that f is as follows: $f(a) < 0$, $f(b) > 0$, $f' > 0$, $f'' > 0$, so that the initial guess x_0 is any point where $f(x_0) > 0$. We shall prove that Newton's sequence $\{x_n\}$ is a monotonic decreasing sequence bounded by below.

Since $f(x_0) > 0$ and $f' > 0$ we have

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} < x_0$$

Newton's method: Proof of the Theorem

Proof.

Since f is continuous and has opposite signs at the endpoints then the equation $f(x) = 0$ has at least one solution, say α. Moreover condition (**) implies that α is unique (f is monotone).

To prove convergence, let us assume for instance that f is as follows: $f(a) < 0$, $f(b) > 0$, $f' > 0$, $f'' > 0$, so that the initial guess x_0 is any point where $f(x_0) > 0$. We shall prove that Newton's sequence $\{x_n\}$ is a monotonic decreasing sequence bounded by below.

Since $f(x_0) > 0$ and $f' > 0$ we have

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} < x_0$$

Since $f'' > 0$, the tangent to f in $(x_0, f(x_0))$ crosses the x-axis before α. Hence,

$$\alpha < x_1 < x_0$$

We had $\alpha < x_1 < x_0$, implying that $f(x_1) > 0$ so that x_1 is itself a Fourier point. Then we restart with x_1 as initial point, and repeating the same argument as before we would get

$$\alpha < x_2 < x_1$$

with $f(x_2) > 0$.

We had $\alpha < x_1 < x_0$, implying that $f(x_1) > 0$ so that x_1 is itself a Fourier point. Then we restart with x_1 as initial point, and repeating the same argument as before we would get

$$\alpha < x_2 < x_1$$

with $f(x_2) > 0$.
Proceeding in this way we have

$$\alpha < x_k < x_{k-1}$$

for all positive integer k.

We had $\alpha < x_1 < x_0$, implying that $f(x_1) > 0$ so that x_1 is itself a Fourier point. Then we restart with x_1 as initial point, and repeating the same argument as before we would get

$$\alpha < x_2 < x_1$$

with $f(x_2) > 0$.
Proceeding in this way we have

$$\alpha < x_k < x_{k-1}$$

for all positive integer k.
Hence, $\{x_n\}$ being a monotonic decreasing sequence bounded by below, it has a limit, that is,

$$\exists\, \eta \quad \text{such that} \quad \lim_{k\to\infty} x_k = \eta.$$

Taking the limit in (1) for $k \to \infty$ (and remembering that both f and f' are continuous, and f' is always $\neq 0$), we have

$$\lim_{k\to\infty}(x_{k+1}) = \lim_{k\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) \Longrightarrow \eta = \eta - \frac{f(\eta)}{f'(\eta)} \Longrightarrow f(\eta) = 0$$

[1] see https://en.wikipedia.org/wiki/Taylor%27s_theorem#Explicit_formulas_for_the_remainder

Taking the limit in (1) for $k \to \infty$ (and remembering that both f and f' are continuous, and f' is always $\neq 0$), we have

$$\lim_{k\to\infty}(x_{k+1}) = \lim_{k\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) \Longrightarrow \eta = \eta - \frac{f(\eta)}{f'(\eta)} \Longrightarrow f(\eta) = 0$$

[1] see
https://en.wikipedia.org/wiki/Taylor%27s_theorem#Explicit_formulas_for_the_remainder

Taking the limit in (1) for $k \to \infty$ (and remembering that both f and f' are continuous, and f' is always $\neq 0$), we have

$$\lim_{k\to\infty}(x_{k+1}) = \lim_{k\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) \Longrightarrow \eta = \eta - \frac{f(\eta)}{f'(\eta)} \implies f(\eta) = 0$$

Then, η is a root of $f(x) = 0$, and since the root is unique, $\eta \equiv \alpha$.

[1] see
https://en.wikipedia.org/wiki/Taylor%27s_theorem#Explicit_formulas_for_the_remainder

Taking the limit in (1) for $k \to \infty$ (and remembering that both f and f' are continuous, and f' is always $\neq 0$), we have

$$\lim_{k\to\infty}(x_{k+1}) = \lim_{k\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) \Longrightarrow \eta = \eta - \frac{f(\eta)}{f'(\eta)} \Longrightarrow f(\eta) = 0$$

Then, η is a root of $f(x) = 0$, and since the root is unique, $\eta \equiv \alpha$.
It remains to prove (2). For this, use Taylor expansion centered in x_k, with Lagrange remainder[1]

[1] see
`https://en.wikipedia.org/wiki/Taylor%27s_theorem#Explicit_formulas_for_the_remainder`

Taking the limit in (1) for $k \to \infty$ (and remembering that both f and f' are continuous, and f' is always $\neq 0$), we have

$$\lim_{k\to\infty}(x_{k+1}) = \lim_{k\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) \Longrightarrow \eta = \eta - \frac{f(\eta)}{f'(\eta)} \implies f(\eta) = 0$$

Then, η is a root of $f(x) = 0$, and since the root is unique, $\eta \equiv \alpha$.
It remains to prove (2). For this, use Taylor expansion centered in x_k, with Lagrange remainder[1]

$$f(\alpha) = f(x_k) + (\alpha - x_k)f'(x_k) + \frac{(\alpha - x_k)^2}{2}f''(z) \quad z \text{ between } \alpha \text{ and } x_k.$$

[1] see https://en.wikipedia.org/wiki/Taylor%27s_theorem#Explicit_formulas_for_the_remainder

Taking the limit in (1) for $k \to \infty$ (and remembering that both f and f' are continuous, and f' is always $\neq 0$), we have

$$\lim_{k\to\infty}(x_{k+1}) = \lim_{k\to\infty}\left(x_k - \frac{f(x_k)}{f'(x_k)}\right) \Longrightarrow \eta = \eta - \frac{f(\eta)}{f'(\eta)} \Longrightarrow f(\eta) = 0$$

Then, η is a root of $f(x) = 0$, and since the root is unique, $\eta \equiv \alpha$.
It remains to prove (2). For this, use Taylor expansion centered in x_k, with Lagrange remainder[1]

$$f(\alpha) = f(x_k) + (\alpha - x_k)f'(x_k) + \frac{(\alpha - x_k)^2}{2}f''(z) \quad z \text{ between } \alpha \text{ and } x_k.$$

Now: $f(\alpha) = 0$, $f'(x)$ is always $\neq 0$ so we can divide by $f'(x_k)$ and get

$$0 = \underbrace{\frac{f(x_k)}{f'(x_k)} - x_k}_{-x_{k+1}} + \alpha + \frac{(\alpha - x_k)^2}{2f'(x_k)}f''(z)$$

[1] see https://en.wikipedia.org/wiki/Taylor%27s_theorem#Explicit_formulas_for_the_remainder

We found

$$0 = \underbrace{\frac{f(x_k)}{f'(x_k)} - x_k}_{-x_{k+1}} + \alpha + \frac{(\alpha - x_k)^2}{2f'(x_k)} f''(z)$$

We found

$$0 = \underbrace{\frac{f(x_k)}{f'(x_k)} - x_k}_{-x_{k+1}} + \alpha + \frac{(\alpha - x_k)^2}{2f'(x_k)} f''(z)$$

that we re-write as

$$x_{k+1} - \alpha = \frac{(\alpha - x_k)^2}{2f'(x_k)} f''(z).$$

We found

$$0 = \underbrace{\frac{f(x_k)}{f'(x_k)} - x_k}_{-x_{k+1}} + \alpha + \frac{(\alpha - x_k)^2}{2f'(x_k)} f''(z)$$

that we re-write as

$$x_{k+1} - \alpha = \frac{(\alpha - x_k)^2}{2f'(x_k)} f''(z).$$

Thus,

$$|x_{k+1} - \alpha| = \frac{(\alpha - x_k)^2}{2} \frac{|f''(z)|}{|f'(x_k)|} \leq \frac{(\alpha - x_k)^2}{2} \frac{\max |f''(x)|}{\min |f'(x)|}$$

We found

$$0 = \underbrace{\frac{f(x_k)}{f'(x_k)} - x_k}_{-x_{k+1}} + \alpha + \frac{(\alpha - x_k)^2}{2f'(x_k)} f''(z)$$

that we re-write as

$$x_{k+1} - \alpha = \frac{(\alpha - x_k)^2}{2f'(x_k)} f''(z).$$

Thus,

$$|x_{k+1} - \alpha| = \frac{(\alpha - x_k)^2}{2} \frac{|f''(z)|}{|f'(x_k)|} \leq \frac{(\alpha - x_k)^2}{2} \frac{\max |f''(x)|}{\min |f'(x)|}$$

Therefore (2) holds with

$$C = \frac{\max |f''(x)|}{\min |f'(x)|}$$

which exists since both $|f'(x)|$ and $|f''(x)|$ are continuous on the closed interval, and $f'(x)$ is always different from zero.

Newton's method: Practical use of the theorem

The practical use of the above Convergence theorem is not easy.
• Often difficult, if not impossible, to check that all the assumptions are verified.
In practice, we interpret the Theorem as: *if x_0 is "close enough" to the (unknown) root, the method converges, and converges fast*.
• Suggestions: the graphics of the function (if available), and a few bisection steps help in locating the root with a rough approximation. Then choose x_0 in order to start Newton's method and obtain a much more accurate evaluation of the root.

If α is a multiple root ($f'(\alpha) = 0$) the method is in troubles.

Newton's method: Stopping criteria 1

Unlike with bisection method, here there are no intervals that become smaller and smaller, but just the sequence of iterates.

Newton's method: Stopping criteria 1

Unlike with bisection method, here there are no intervals that become smaller and smaller, but just the sequence of iterates.
A reasonable criterion could be

- **test on the iterates**: stop at the first iteration n such that

$$|x_n - x_{n-1}| \leq Tol,$$

  and take x_n as "root".

Newton's method: Stopping criteria 1

Unlike with bisection method, here there are no intervals that become smaller and smaller, but just the sequence of iterates.
A reasonable criterion could be

- **test on the iterates**: stop at the first iteration n such that

$$|x_n - x_{n-1}| \leq Tol,$$

  and take x_n as "root".

This would work, unless the function is very **steep** in the vicinity of the root (that is, if $|f'(\alpha)| >> 1$): the tangents being almost vertical, two iterates might be very close to each other but not close enough to the root to make $f(x_n)$ also small, and the risk is to stop when $f(x_n)$ is still big.

Newton's method: Stopping criteria 2

In this situation it would be better to use the

- **test on the residual**: stop at the first iteration n such that

$$|f(x_n)| \leq Tol,$$

and take x_n as "root".

Newton's method: Stopping criteria 2

In this situation it would be better to use the

- **test on the residual**: stop at the first iteration n such that

$$|f(x_n)| \leq Tol,$$

  and take x_n as "root".

In contrast to the previous criterion, this one would fail if the function is very **flat** in the vicinity of the root (that is, if $|f'(\alpha)| << 1$). In this case $|f(x_n)|$ could be small, but x_n could still be far from the root.

Newton's method: Stopping criteria 2

In this situation it would be better to use the

- **test on the residual**: stop at the first iteration n such that

$$|f(x_n)| \leq Tol,$$

  and take x_n as "root".

In contrast to the previous criterion, this one would fail if the function is very **flat** in the vicinity of the root (that is, if $|f'(\alpha)| << 1$). In this case $|f(x_n)|$ could be small, but x_n could still be far from the root.

What to do then??

Newton's method: Stopping criteria 2

In this situation it would be better to use the

- **test on the residual**: stop at the first iteration n such that

$$|f(x_n)| \leq Tol,$$

and take x_n as "root".

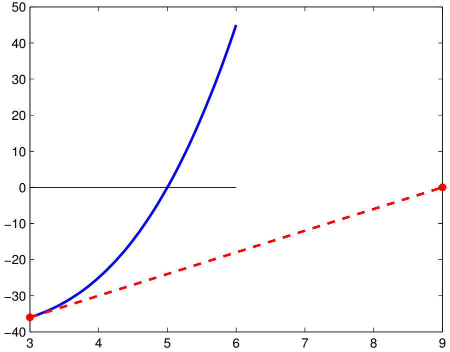
In contrast to the previous criterion, this one would fail if the function is very **flat** in the vicinity of the root (that is, if $|f'(\alpha)| << 1$). In this case $|f(x_n)|$ could be small, but x_n could still be far from the root.

What to do then??

Safer to use both criteria, and stop when both of them are verified.

Newton's method: Examples of choices of x_0

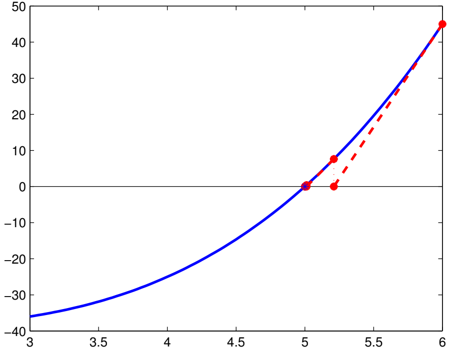
$$f(x) = x^3 - 5x^2 + 9x - 45 \quad \text{in } [3,6] \quad \alpha = 5$$



Bad x_0: $x_0 = 3 \Rightarrow x_1 = 9$ outside $[3,6]$

Newton's method: Examples of choices of x_0

$$f(x) = x^3 - 5x^2 + 9x - 45 \quad \text{in } [3,6] \quad \alpha = 5$$



Good x_0: 3 iterations with $Tol = 1.e-3$

Newton's method: Solution of nonlinear systems

We have to solve a system of N nonlinear equations:

$$\begin{cases} f_1(x_1, x_2, \cdots, x_N) = 0 \\ f_2(x_1, x_2, \cdots, x_N) = 0 \\ \qquad\vdots \\ f_N(x_1, x_2, \cdots, x_N) = 0 \end{cases}$$

Newton's method: Solution of nonlinear systems

We have to solve a system of N nonlinear equations:

$$\begin{cases} f_1(x_1, x_2, \cdots, x_N) = 0 \\ f_2(x_1, x_2, \cdots, x_N) = 0 \\ \qquad \vdots \\ f_N(x_1, x_2, \cdots, x_N) = 0 \end{cases}$$

or, in compact form,

$$\underline{F}(\underline{x}) = \underline{0},$$

having set

$$\underline{x} = (x_1, x_2, \cdots, x_N), \quad \underline{F} = (f_1, f_2, \cdots, f_N)$$

Newton method

We mimic what done for a single equation $f(x) = 0$: starting from an initial guess x_0 we constructed a sequence by linearizing f at each point and replacing it by its tangent, i.e., its Taylor polynomial of degree 1.

Newton method

We mimic what done for a single equation $f(x) = 0$: starting from an initial guess x_0 we constructed a sequence by linearizing f at each point and replacing it by its tangent, i.e., its Taylor polynomial of degree 1.

For systems we do the same:
starting from a point $\underline{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \cdots, x_N^{(0)})$ we construct a sequence $\{\underline{x}^{(k)}\}$ by

Newton method

We mimic what done for a single equation $f(x) = 0$: starting from an initial guess x_0 we constructed a sequence by linearizing f at each point and replacing it by its tangent, i.e., its Taylor polynomial of degree 1.

For systems we do the same:
starting from a point $\underline{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \cdots, x_N^{(0)})$ we construct a sequence $\{\underline{x}^{(k)}\}$ by

- linearising $\underline{F}$ at each point through its Taylor expansion of degree 1:

$$\underline{F}(\underline{x}) \simeq \underline{F}(\underline{x}^{(k)}) + J_F(\underline{x}^{(k)})(\underline{x} - \underline{x}^{(k)})$$

Newton method

We mimic what done for a single equation $f(x) = 0$: starting from an initial guess x_0 we constructed a sequence by linearizing f at each point and replacing it by its tangent, i.e., its Taylor polynomial of degree 1.

For systems we do the same:
starting from a point $\underline{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \cdots, x_N^{(0)})$ we construct a sequence $\{\underline{x}^{(k)}\}$ by

- linearising $\underline{F}$ at each point through its Taylor expansion of degree 1:

$$\underline{F}(\underline{x}) \simeq \underline{F}(\underline{x}^{(k)}) + J_F(\underline{x}^{(k)})(\underline{x} - \underline{x}^{(k)})$$

- and then defining $\underline{x}^{(k+1)}$ as the solution of

$$\underline{F}(\underline{x}^{(k)}) + J_F(\underline{x}^{(k)})(\underline{x}^{(k+1)} - \underline{x}^{(k)}) = \underline{0}.$$

$J_F(\underline{x}^{(k)})$ is the **jacobian matrix** of $\underline{F}$ evaluated at the point $\underline{x}^{(k)}$:

$$J_F(\underline{x}) = \begin{bmatrix} \dfrac{\partial f_1(\underline{x})}{\partial x_1} & \dfrac{\partial f_1(\underline{x})}{\partial x_2} & \cdots\cdots & \dfrac{\partial f_1(\underline{x})}{\partial x_N} \\ \dfrac{\partial f_2(\underline{x})}{\partial x_1} & \dfrac{\partial f_2(\underline{x})}{\partial x_2} & \cdots\cdots & \dfrac{\partial f_2(\underline{x})}{\partial x_N} \\ & \vdots & & \\ & \vdots & & \\ \dfrac{\partial f_N(\underline{x})}{\partial x_1} & \dfrac{\partial f_N(\underline{x})}{\partial x_2} & \cdots\cdots & \dfrac{\partial f_N(\underline{x})}{\partial x_N} \end{bmatrix},$$

$J_F(\underline{x}^{(k)})$ is the **jacobian matrix** of $\underline{F}$ evaluated at the point $\underline{x}^{(k)}$:

$$J_F(\underline{x}) = \begin{bmatrix} \frac{\partial f_1(\underline{x})}{\partial x_1} & \frac{\partial f_1(\underline{x})}{\partial x_2} & \cdots\cdots & \frac{\partial f_1(\underline{x})}{\partial x_N} \\ \frac{\partial f_2(\underline{x})}{\partial x_1} & \frac{\partial f_2(\underline{x})}{\partial x_2} & \cdots\cdots & \frac{\partial f_2(\underline{x})}{\partial x_N} \\ & \vdots & & \\ & \vdots & & \\ \frac{\partial f_N(\underline{x})}{\partial x_1} & \frac{\partial f_N(\underline{x})}{\partial x_2} & \cdots\cdots & \frac{\partial f_N(\underline{x})}{\partial x_N} \end{bmatrix},$$

System $\underline{F}(\underline{x}^{(k)}) + J_F(\underline{x}^{(k)})(\underline{x}^{(k+1)} - \underline{x}^{(k)}) = \underline{0}$ can obviously be written as: $\underline{x}^{k+1} = \underline{x}^{(k)} - (J_F(\underline{x}^{(k)}))^{-1}\underline{F}(\underline{x}^{(k)})$.

$J_F(\underline{x}^{(k)})$ is the **jacobian matrix** of $\underline{F}$ evaluated at the point $\underline{x}^{(k)}$:

$$J_F(\underline{x}) = \begin{bmatrix} \dfrac{\partial f_1(\underline{x})}{\partial x_1} & \dfrac{\partial f_1(\underline{x})}{\partial x_2} & \cdots\cdots & \dfrac{\partial f_1(\underline{x})}{\partial x_N} \\ \dfrac{\partial f_2(\underline{x})}{\partial x_1} & \dfrac{\partial f_2(\underline{x})}{\partial x_2} & \cdots\cdots & \dfrac{\partial f_2(\underline{x})}{\partial x_N} \\ & \vdots & & \\ & \vdots & & \\ \dfrac{\partial f_N(\underline{x})}{\partial x_1} & \dfrac{\partial f_N(\underline{x})}{\partial x_2} & \cdots\cdots & \dfrac{\partial f_N(\underline{x})}{\partial x_N} \end{bmatrix},$$

System $\underline{F}(\underline{x}^{(k)}) + J_F(\underline{x}^{(k)})(\underline{x}^{(k+1)} - \underline{x}^{(k)}) = \underline{0}$ can obviously be written as: $\underline{x}^{k+1} = \underline{x}^{(k)} - (J_F(\underline{x}^{(k)}))^{-1}\underline{F}(\underline{x}^{(k)})$.
In the actual computation of $\underline{x}^{k+1}$ we **do not** compute the inverse matrix $(J_F(\underline{x}^{(k)}))^{-1}$, but we solve the system

$$J_F(\underline{x}^{(k)})\underline{x}^{k+1} = J_F(\underline{x}^{(k)})\underline{x}^{(k)} - \underline{F}(\underline{x}^{(k)}).$$

Newton's method: Algorithm

Given $\underline{x}^{(0)} \in \mathbb{R}^N$, for $k = 0, 1, \cdots$

solve $J_F(\underline{x}^{(k)})\underline{x}^{k+1} = J_F(\underline{x}^{(k)})\underline{x}^{(k)} - \underline{F}(\underline{x}^{(k)})$ by the following steps

Newton's method: Algorithm

Given $\underline{x}^{(0)} \in \mathbb{R}^N$, for $k = 0, 1, \cdots$

solve $J_F(\underline{x}^{(k)})\underline{x}^{k+1} = J_F(\underline{x}^{(k)})\underline{x}^{(k)} - \underline{F}(\underline{x}^{(k)})$ by the following steps

- solve $J_F(\underline{x}^{(k)})\underline{\delta}^{(k)} = -\underline{F}(\underline{x}^{(k)})$

Newton's method: Algorithm

Given $\underline{x}^{(0)} \in \mathbb{R}^N$, for $k = 0, 1, \cdots$

solve $J_F(\underline{x}^{(k)})\underline{x}^{k+1} = J_F(\underline{x}^{(k)})\underline{x}^{(k)} - \underline{F}(\underline{x}^{(k)})$ by the following steps

- solve $J_F(\underline{x}^{(k)})\underline{\delta}^{(k)} = -\underline{F}(\underline{x}^{(k)})$
- set $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \underline{\delta}^{(k)}$

Newton's method: Algorithm

Given $\underline{x}^{(0)} \in \mathbb{R}^N$, for $k = 0, 1, \cdots$

solve $J_F(\underline{x}^{(k)})\underline{x}^{k+1} = J_F(\underline{x}^{(k)})\underline{x}^{(k)} - \underline{F}(\underline{x}^{(k)})$ by the following steps

- solve $J_F(\underline{x}^{(k)})\underline{\delta}^{(k)} = -\underline{F}(\underline{x}^{(k)})$
- set $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \underline{\delta}^{(k)}$

At each iteration k we have to solve a linear system with matrix $J_F(\underline{x}^{(k)})$ (that is the most expensive part of the algorithm).

Newton's method: Algorithm

Given $\underline{x}^{(0)} \in \mathbb{R}^N$, for $k = 0, 1, \cdots$

solve $J_F(\underline{x}^{(k)})\underline{x}^{k+1} = J_F(\underline{x}^{(k)})\underline{x}^{(k)} - \underline{F}(\underline{x}^{(k)})$ by the following steps

- solve $J_F(\underline{x}^{(k)})\underline{\delta}^{(k)} = -\underline{F}(\underline{x}^{(k)})$
- set $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \underline{\delta}^{(k)}$

At each iteration k we have to solve a linear system with matrix $J_F(\underline{x}^{(k)})$ (that is the most expensive part of the algorithm).

Note that by introducing the unknown $\underline{\delta}^{(k)}$ we pay an extra sum ($\underline{x}^{(k+1)} = \underline{x}^{(k)} + \underline{\delta}^{(k)}$) but we save the (much more expensive) matrix-vector multiplication $J_F(\underline{x}^{(k)})\underline{x}^{(k)}$.

Newton's method: Stopping criteria

They are the same two criteria that we saw for scalar equations:

Newton's method: Stopping criteria

They are the same two criteria that we saw for scalar equations:

- **test on the iterates**: stop at iteration k such that

$$\|\underline{x}^{(k)} - \underline{x}^{(k-1)}\| \leq Tol$$

for some vector norm, and take $\underline{x}^{(k)}$ as "root".

Newton's method: Stopping criteria

They are the same two criteria that we saw for scalar equations:

- **test on the iterates**: stop at iteration k such that

$$\|\underline{x}^{(k)} - \underline{x}^{(k-1)}\| \leq Tol$$

for some vector norm, and take $\underline{x}^{(k)}$ as "root".

- **test on the residual**: stop at iteration k such that

$$\|\underline{F}(\underline{x}^{(k)})\| \leq Tol,$$

and take $\underline{x}^{(k)}$ as "root".

Newton's method: Stopping criteria

They are the same two criteria that we saw for scalar equations:

- **test on the iterates**: stop at iteration k such that

$$\|\underline{x}^{(k)} - \underline{x}^{(k-1)}\| \leq Tol$$

for some vector norm, and take $\underline{x}^{(k)}$ as "root".

- **test on the residual**: stop at iteration k such that

$$\|\underline{F}(\underline{x}^{(k)})\| \leq Tol,$$

and take $\underline{x}^{(k)}$ as "root".

Here too, it would be wise in practice to use **both** criteria, and stop when both of them are satisfied.