

# Nonlinear Equations

## 1 Introduction

In applications we usually need to find several unknown values  $x_1, \dots, x_n$ . We have  $n$  equations for  $x_1, \dots, x_n$

$$\begin{aligned} f_1(x_1, \dots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \dots, x_n) &= 0 \end{aligned}$$

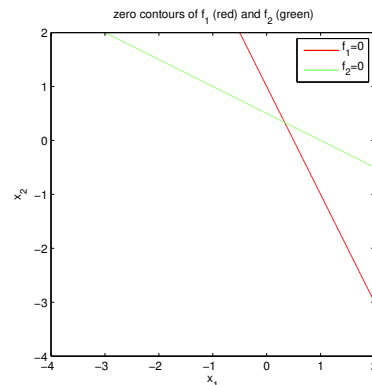
and we want to find the solutions.

In many cases the problem can be (approximatively) described by **linear equations**. In this case we have  $n$  linear equations for  $n$  unknowns. We will get a unique solution if the matrix is nonsingular.

**Example** with  $n = 2$ : Find  $x_1, x_2$  such that

$$2x_1 + x_2 - 1 = 0$$

$$x_1 + 2x_2 - 1 = 0$$



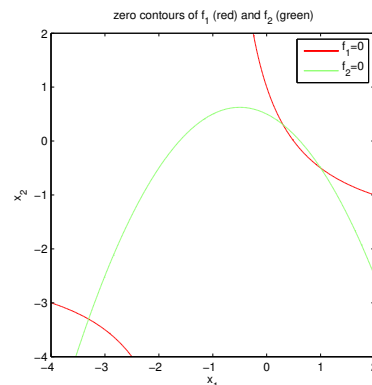
Here we have one solution  $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$  which is the intersection of the red and the green line.

In other cases the problem is nonlinear, and we obtain  $n$  **nonlinear equations**.

**Example** with  $n = 2$ : Find  $x_1, x_2$  such that

$$2x_1 + x_2 + x_1x_2 - 1 = 0$$

$$x_1 + 2x_2 + x_1^2 - 1 = 0$$



Here we have three solutions  $\begin{bmatrix} .3028 \\ .3028 \end{bmatrix}$ ,  $\begin{bmatrix} 1 \\ -5 \end{bmatrix}$ ,  $\begin{bmatrix} -3.3028 \\ -3.3028 \end{bmatrix}$ . Here is how I found the first solution in **Matlab**:

```
f = @(x) [ 2*x(1)+x(2)+x(1)*x(2)-1 ; x(1)+2*x(2)+x(1)^2-1 ] % Define function f
xs = fsolve(f,[0;0]) % Find solution near [0;0]
```

## 2 One nonlinear equation

### 2.1 Bisection Method

Assume that the function  $f$  is continuous. If we have two function values  $f(a), f(b)$  with opposite signs then the intermediate value theorem guarantees that there must be a point  $x_* \in (a, b)$  with  $f(x_*) = 0$ . This motivates the bisection method:

**Bisection method** Find  $x_* \in [a_0, b_0]$  such that  $f(x_*) = 0$ .

Given:

- subroutine to evaluate  $f(x)$
- initial guesses  $a_0, b_0$  where  $f(a_0)$  and  $f(b_0)$  have different signs

this algorithm computes a sequence of intervals  $[a_k, b_k]$  such that  $x_* \in [a_k, b_k]$

**Algorithm:** For  $k = 0, 1, 2, \dots$ :

$c_k := (a_k + b_k)/2$

evaluate  $f(c_k)$

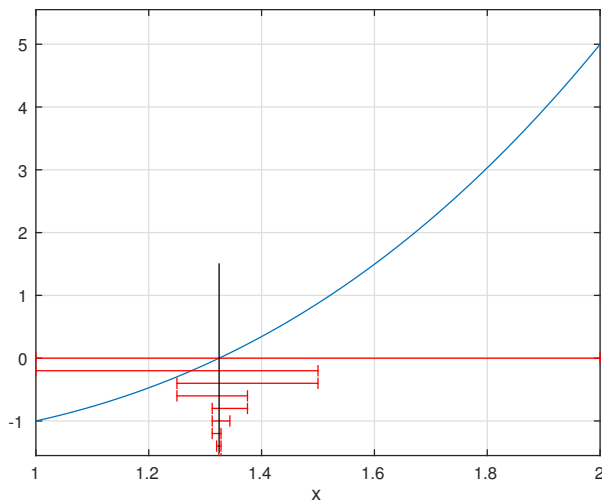
If  $f(c_k) = 0$ : stop

If  $f(c_k), f(a_k)$  have different sign:  $[a_{k+1}, b_{k+1}] := [a_k, c_k]$

If  $f(c_k), f(a_k)$  have same sign:  $[a_{k+1}, b_{k+1}] := [c_k, b_k]$

**Example:** For  $f(x) = x^3 - x - 1$  and  $a_0 = 1, b_0 = 2$  we obtain the following intervals:  $f(1) = -1 < 0, f(2) = 5 > 0$

| $k$      | $[a_k, b_k]$                             | $b_k - a_k$          |                             |
|----------|------------------------------------------|----------------------|-----------------------------|
| 0        | [1.0, 2.0]                               | 1                    | $f(1.5) = .875 > 0$         |
| 1        | [1.0, 1.5]                               | 0.5                  | $f(1.25) = -.29688 < 0$     |
| 2        | [1.25, 1.5]                              | 0.25                 | $f(1.375) = .22461 > 0$     |
| 3        | [1.25, 1.375]                            | 0.125                | $f(1.3125) = -.051514 < 0$  |
| 4        | [1.3125, 1.375]                          | $6.3 \cdot 10^{-2}$  | $f(1.34375) = .082611 > 0$  |
| 5        | [1.3125, 1.34375]                        | $3.1 \cdot 10^{-2}$  | $f(1.328125) = .014576 > 0$ |
| 6        | [1.3125, 1.328125]                       | $1.6 \cdot 10^{-2}$  |                             |
| $\vdots$ | $\vdots$                                 | $\vdots$             |                             |
| 50       | [1.3247179572447454, 1.3247179572447463] | $8.9 \cdot 10^{-16}$ |                             |
| 51       | [1.3247179572447458, 1.3247179572447463] | $4.4 \cdot 10^{-16}$ |                             |
| 52       | [1.3247179572447458, 1.3247179572447461] | $2.2 \cdot 10^{-16}$ |                             |



In practice we assume that the function is given as a subroutine which we can call (in Matlab we can specify a function in an m-file). Then the only substantial cost for each step is this call to this external function:

### The cost of one bisection step is one function evaluation.

In theory we can let this continue forever, unless we hit for some  $k$  the solution  $x_*$  exactly. Then we get an infinite sequence  $a_k$  with  $a_{k+1} \geq a_k$  and an infinite sequence  $b_k$  with  $b_{k+1} \leq b_k$ .

**Theorem 2.1.** Assume that the function  $f$  is continuous on  $[a_0, b_0]$ . If  $f(a_0)$  and  $f(b_0)$  have different sign, then the bisection method converges:

$$\lim_{k \rightarrow \infty} a_k = \lim_{k \rightarrow \infty} b_k = x_* \quad \text{with } f(x_*) = 0.$$

*Proof.* As  $b_k - a_k = 2^{-k}(b_0 - a_0)$  goes to zero as  $k \rightarrow \infty$  we must have that both  $a_k$  and  $b_k$  converge to the same point  $x_*$  as  $k \rightarrow \infty$ . Since  $f(a_k) \leq 0$  and  $f(b_k) \geq 0$  we must have  $f(x_*) = 0$  by the continuity of  $f$ .  $\square$

Note that the midpoint  $c_k$  satisfies  $|c_k - x_*| \leq (b_k - a_k)/2$ , therefore we have **decreasing error bounds  $E_k$  satisfying**

$$|c_k - x_*| \leq E_k, \quad E_{k+1} = \frac{1}{2}E_k$$

For each step we have  $E_{k+1} = \frac{1}{2}E_k$ , so the error is reduced by at least a factor  $\frac{1}{2}$ .

Essentially, we obtain one new binary digit of the result with each bisection step, or about 3 new decimal digits with 10 bisection steps.

If for some method we get approximations  $x_k$  satisfying  $|x_k - x_*| \leq E_k$  where the error bounds satisfy

$$E_k \leq CE_k$$

with a fixed factor  $C < 1$  we say that the method is **convergent of order 1** (since we have  $E_{k+1} \leq C \cdot E_k^1$  with exponent 1).

#### Bisection method:

- assumption:  $f(x)$  is continuous, we have  $a_0, b_0$  where  $f(a_0), f(b_0)$  have different sign
- cost per step: 1 function evaluation
- improvement of error bound for each step:  $E_{k+1} = \frac{1}{2}E_k$
- “bracketing”: we can guarantee  $a_k \leq x_* \leq b_k$  for each step

## 2.2 Secant Method

Assume that we have two function values  $f(a)$  and  $f(b)$ . Based on this information we want to find a good guess  $c$  for the solution  $x_*$ : We can approximate  $f(x)$  by the linear interpolation

$$p(x) = f(b) + f[a, b](x - b)$$

where  $f[a, b] = \frac{f(b) - f(a)}{b - a}$ . Then we find  $c$  such that  $p(c) = 0$ : Solving  $f(b) + f[a, b](c - b) = 0$  for  $c$  gives

$$c = b - f(b)/f[a, b].$$

If we have two initial guesses  $x_0, x_1$  we can use this to find an improved guess  $x_2$ . Using  $x_1, x_2$  we find  $x_3$ , etc.

**Algorithm: Secant Method** Find  $x_* \in [a_0, b_0]$  such that  $f(x_*) = 0$ .

Given:

- subroutine to evaluate  $f(x)$
- two initial guesses  $x_0, x_1$

Then this algorithm computes a sequence  $x_k$  which may or may not converge to a root  $x_*$ .

**Algorithm:** For  $k = 1, 2, 3, \dots$ :

$$x_{k+1} := x_k - f(x_k)/f[x_{k-1}, x_k]$$

**Example:** Consider the function  $f(x) = x^3 - x - 1$  and the initial guesses  $x_0 = 1, x_1 = 2$ . We can measure the “number of correct digits” as  $N_k := -\log_{10} |x_k - x_*|$ . We want to see how  $N_k$  grows with each step.

| $k$ | $x_k$                                                                    | $N_k$ | $N_k/N_{k-1}$ |
|-----|--------------------------------------------------------------------------|-------|---------------|
| 2   | <u>1.166666666666666666666666666666666666666666666666666666666666667</u> | 0.8   |               |
| 3   | <u>1.25311203319502074688796680497925311203319502074688796680</u>        | 1.1   | 1.4292        |
| 4   | <u>1.33720644584165640039570185142407424920450200352602540210</u>        | 1.9   | 1.6624        |
| 5   | <u>1.32385009638764090385381405944349497906016891171355395298</u>        | 3.1   | 1.6084        |
| 6   | <u>1.32470793653208797064174666448878197322217786129602539372</u>        | 5.0   | 1.6329        |
| 7   | <u>1.32471796535381767575939715803702106931272485403036767120</u>        | 8.1   | 1.6185        |
| 8   | <u>1.32471795724467030158200181494839983320960900093813494824</u>        | 13.1  | 1.6216        |
| 9   | <u>1.32471795724474602596033662546920425813781535049399611421</u>        | 21.2  | 1.6190        |
| 10  | <u>1.32471795724474602596090885447809738111461705368557107541</u>        | 34.4  | 1.6191        |
| 11  | <u>1.32471795724474602596090885447809734073440405690173336451</u>        | 55.7  | 1.6185        |

## Errors of the secant method

During the algorithm we have  $a = x_{k-1}$  and  $b = x_k$ . We then compute  $c = x_{k+1}$  using the secant. We want to show that the new error  $|c - x_*|$  is small:

From the interpolation error we know that

$$f(x_*) - p(x_*) = R(x_*), \quad R(x_*) = \frac{1}{\gamma} f''(t) \cdot (x_* - a)(x_* - b)$$

(where  $t$  is somewhere between  $a, b, x_*$ ). If  $|f''(t)| \leq C_2$  we have  $|R(x_*)| \leq \frac{C_2}{2} |x_* - a| \cdot |x_* - b|$ .

Note that  $f(x_*) = 0 = p(c)$ . Hence

$$\underbrace{p(c) - p(x_*)}_{f[a,b] \cdot (c - x_*)} = R(x_*)$$

since  $p(x)$  is a linear function with slope  $f[a, b]$ . Therefore

$$c - x_* = \frac{R(x_*)}{f[a, b]}$$

We have  $f[a, b] = f'(s)$  with  $s \in [a, b]$ . If  $|f'(s)| \geq C_1 > 0$  we therefore have with  $D := \frac{C_2}{2C_1}$

$$\boxed{|c - x_*| \leq D|a - x_*| \cdot |b - x_*|} \quad (1)$$

Since  $a = x_{k-1}$ ,  $b = x_k$ ,  $c = x_{k+1}$  we obtain

$$|x_{k+1} - x_*| \leq D |x_{k-1} - x_*| \cdot |x_k - x_*|$$

Let  $e_k := D|x_k - x_*|$ . Multiplying by  $D$  gives

$$e_{k+1} \leq e_{k-1}e_k.$$

Now assume that

$$e_0 \leq q, \quad e_1 \leq q \quad \text{with } q < 1$$

Then we obtain

$$e_0 \leq q^1, \quad e_1 \leq q^2, \quad e_2 \leq q^3, \quad e_3 \leq q^5, \quad \dots \quad e_k \leq q^{F_k}$$

with the **Fibonacci number**  $F_k$  (defined by  $F_0 = 1, F_1 = 1, F_{k+1} = F_k + F_{k-1}$ ). Since  $q < 1$  and  $F_k \rightarrow \infty$  for  $k \rightarrow \infty$  we obtain convergence  $e_k = D \cdot |x_k - x_*| \rightarrow 0$  if our assumptions

$$|f''(t)| \leq C_2, \quad |f'(t)| \geq C_1 > 0 \quad (2)$$

are satisfied. The order of convergence corresponds to the ratio  $F_k/F_{k-1}$  which converges to the golden ratio  $\frac{\sqrt{5}+1}{2}$ .

**Theorem 2.2.** Assume that  $f(x_*) = 0$  and

- $f'(x)$  and  $f''(x)$  exist and are continuous near  $x_*$
- $f'(x_*) \neq 0$ .

Then there exists  $\delta > 0$ ,  $C > 0$  such that for  $|x_0 - x_*| \leq \delta$ ,  $|x_1 - x_*| \leq \delta$  we have

- $\lim_{k \rightarrow \infty} x_k = x_*$  (convergence)
- $|x_k - x_*| \leq E_k$  and  $E_{k+1} \leq CE_k^\alpha$  with  $\alpha = \frac{\sqrt{5}+1}{2}$  (convergence with order  $\alpha > 1$ )

*Proof.* Pick  $\varepsilon > 0$  such that on the interval  $B_\varepsilon = [x_* - \varepsilon, x_* + \varepsilon]$  we have that  $|f'(x)| > 0$  and  $f''$  is continuous:

$$\text{For } x \in B_\varepsilon: \quad |f'(x)| \geq C_1 > 0, \quad |f''(x)| \leq C_2 \quad (3)$$

with some constants  $C_1, C_2$ . Let  $D = \frac{C_2}{2C_1}$ . Pick  $q < 1$  such that  $\delta := q/D \leq \varepsilon$ .

Now assume  $|x_{k-1} - x_*| \leq \delta$ ,  $|x_k - x_*| \leq \delta$ . Since  $\delta \leq \varepsilon$  we have  $x_{k-1}, x_k, x_* \in B_\varepsilon$ . We now have

$$|x_{k+1} - x_*| = \frac{|f''(t)|}{2|f'(s)|} |x_k - x_*| \cdot |x_{k-1} - x_*|$$

where the intermediate points  $s, t$  are located between  $x_0, x_1, x_*$ . Hence we have  $s, t \in B_\varepsilon$  and (3) gives

$$|x_{k+1} - x_*| \leq D |x_k - x_*| \cdot |x_{k-1} - x_*| \leq \underbrace{D\delta}_{q < 1} \cdot \delta < \delta$$

so that we also have  $|x_{k+1} - x_*| \leq \delta$ .

Therefore we obtain by induction that  $|x_k - x_*| \leq \delta$  for  $k = 0, 1, 2, \dots$ , and that

$$|x_{k+1} - x_*| \leq D |x_k - x_*| \cdot |x_{k-1} - x_*|$$

As we saw above, this implies that  $e_k := D|x_k - x_*|$  satisfies  $e_k \leq q^{F_k}$  where  $F_k$  are the Fibonacci numbers. Since  $q < 1$  and  $F_k \rightarrow \infty$  we obtain convergence  $\lim_{k \rightarrow \infty} x_k = x_*$ .

It remains to prove convergence of order  $\alpha = \frac{\sqrt{5}+1}{2}$ : We have shown  $e_k \leq \tilde{E}_k := q^{F_k}$ . Since the Fibonacci numbers satisfy  $F_{k+1} - \alpha F_k = (1 - \alpha)^{k+1} \geq -1$  we have

$$\begin{aligned} F_{k+1} &\geq \alpha F_k - 1 \\ \Rightarrow q^{F_{k+1}} &\leq q^{\alpha F_k} \cdot q^{-1} \\ \Rightarrow \tilde{E}_{k+1} &\leq \tilde{E}_k^\alpha \cdot q^{-1} \end{aligned}$$

□

## 2.3 Newton Method

For the secant method we used the interpolating polynomial with the nodes  $a, b$ . Now assume that  $a = b$ , and that we know  $f(a)$  and  $f'(a)$ . We can approximate  $f(x)$  by the linear interpolation

$$p(x) = f(a) + f'[a, a](x - a)$$

where  $f[a, a] = f'(a)$ . Then we find  $c$  such that  $p(c) = 0$ : Solving  $f(a) + f'[a, a](c - a) = 0$  for  $c$  gives

$$c = b - f(a)/f'[a, a].$$

If we have an initial guesses  $x_0$  we can use this to find an improved guess  $x_1$ , etc.:

**Algorithm: Newton Method**

- Initial guess  $x_0$
- For  $k = 1, 2, 3, \dots$ :  

$$x_{k+1} := x_k - f(x_k)/f'(x_k)$$

**Example:** Consider the function  $f(x) = x^3 - x - 1$  and the initial guess  $x_0 = 1$ . We can measure the “number of correct digits” as  $N_k := -\log_{10} |x_k - x_*|$ :

[illegible]

## Errors of the Newton method

For the errors we obtain from (1) with  $a = b = x_k$ ,  $c = x_{k+1}$  that

$$|x_{k+1} - x_*| \leq D|x_k - x_*|^2$$

if the assumptions (2) hold. Multiplying this by  $D$  gives with  $e_k := D|x_k - x_*|$  that

$$e_{k+1} \leq e_k^2$$

If  $e_0 \leq q < 1$  we therefore obtain  $e_1 \leq q^2, e_2 \leq q^4, e_3 \leq q^8, \dots$

$$e_k < q^{(2^k)}$$

This means that the error converges to zero as  $k \rightarrow \infty$ , and we obtain the following theorem:

**Theorem 2.3.** Assume that  $f(x_*) = 0$  and

- $f'(x)$  and  $f''(x)$  exist and are continuous near  $x_*$
- $f'(x_*) \neq 0$ .

Then there exists  $\delta > 0$ ,  $C > 0$  such that for  $|x_0 - x_*| \leq \delta$  we have

- $\lim_{k \rightarrow \infty} x_k = x_*$  (convergence)
- $|x_{k+1} - x_*| \leq C|x_k - x_*|^2$  (convergence of order 2)

*Proof.* Exactly like the proof of Theorem 2.2.

## 2.4 Local Convergence

Note that for both the secant and the Newton method we can only guarantee convergence if the initial guesses are sufficiently close to the solution  $x_*$ . This type of result is called “local convergence”: If the initial guess is too far away we may not have convergence at all. If however the initial guess satisfies  $|x_0 - x_*| \leq \delta$  then the accuracy increases very quickly (doubling the number of correct digits with each step for the Newton method).

This is in contrast to the bisection method which is guaranteed to converge, no matter how large the initial interval  $[a_0, b_0]$  is (but we need  $f(a_0)f(b_0) < 0$ ). This type of result is called “global convergence”.

**Example 1:** Consider the Newton method for the function  $f(x) = \tan^{-1}(x)$  which has the zero  $x_* = 0$ .

For  $|x_0| < 1.39$  the Newton method converges to 0.

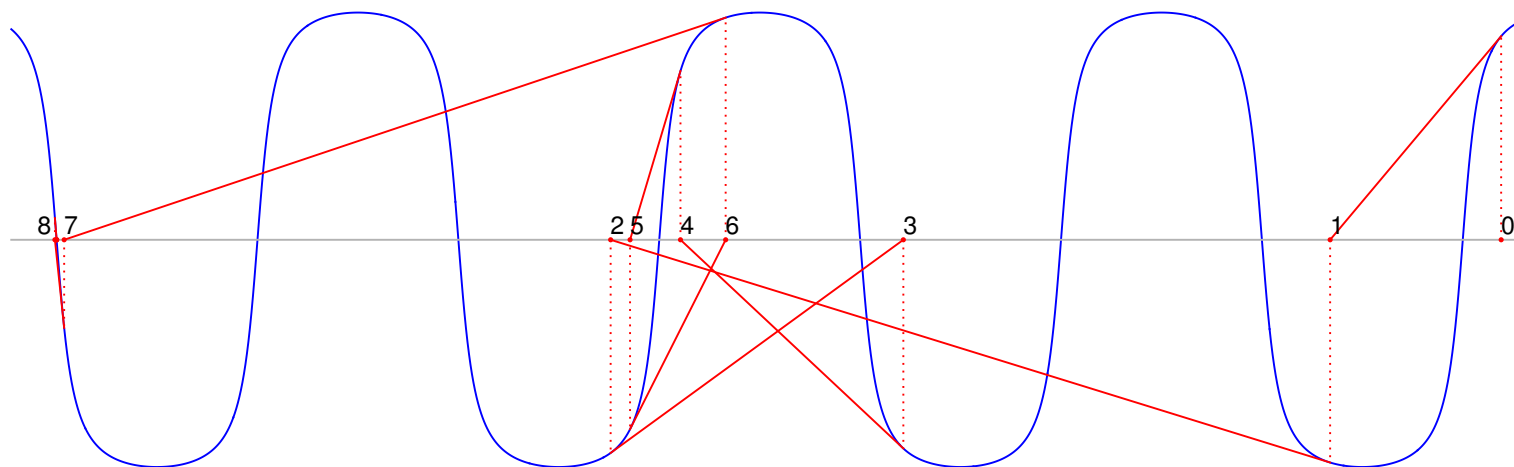
For  $|x_0| > 1.4$  the Newton method diverges, with  $|x_k| \rightarrow \infty$  as  $k \rightarrow \infty$ .

For  $x_0 = 1.39174520027073 \dots$  the Newton method gives  $x_1 = -x_0, x_2 = x_0, x_3 = -x_0, \dots$

**Example 2:** Consider the Newton method for  $f(x) = \tan^{-1}(5 \sin x)$  which has zeros  $k\pi$ ,  $k \in \mathbb{Z}$ .

For  $x_0 = 0.6$  we obtain

| $k$ | $x_k$                                    |
|-----|------------------------------------------|
| 1   | -2.07458463997807                        |
| 2   | -13.3266896191295                        |
| 3   | -8.74903429240199                        |
| 4   | -12.2345381742364                        |
| 5   | -13.0229622940453                        |
| 6   | -11.5280491387632                        |
| 7   | - <u>21.8729923205714</u>                |
| 8   | - <u>22.0175237627052</u>                |
| 9   | - <u>21.9908377144459</u>                |
| 10  | - <u>21.9911485756392</u>                |
| 11  | - <u>21.9911485751285526692385036807</u> |



This illustrates a typical behavior for the Newton and secant methods: First the values jump around wildly. Then we hit “by chance” one of the  $\delta$ -intervals around a zero. Now the superlinear convergence sets in, and we get full machine accuracy in about 4 iterations.

## 2.5 Hybrid methods and the Matlab command `fzero`

The bisection method has several advantages:

- it is guaranteed to converge, even if the function is not differentiable (only continuity is needed)
- it gives at each step an interval containing the zero  $x_*$  (“bracketing”)

However, the bisection method converges very slowly (one new binary digit of accuracy with each step).

The Newton and secant method converge much faster (“superlinear convergence”: the number of correct digits gets multiplied by  $\alpha > 1$  with each step), however this only works if

- $f'(x)$  and  $f''(x)$  exist and are continuous
- $f'(x_*) \neq 0$
- we are already close to the zero ( $\delta$  in Theorem 2.2, Theorem 2.3).