NUMERICKÉ METODY

VÍTĚZSLAV VESELÝ

PODPORA VÝUKY NA POČÍTAČOVÉ SÍTI

`http://www.math.muni.cz/~vesely`

a) **MATLAB:**

Algoritmy:

Příkazy pro připojení a výpis obsahu knihoven:

`d = jmathews('chap_x')`

chap_x ... adresář korespondující s číslem kapitoly z učebnice
J. H. Mathews: NUMERICAL METHODS

`d = vesely('nummet')` ... výpis podknihoven autora

`d = vesely('nummet/podknihovna')` ... výpis algoritmů v podknihovně

`d` ... výstup=úplná cesta do adresáře knihovny (podknihovny)

Příklady:

`zktest([],'?')` ... výpis témat

`zktest([],'téma')` ... zadání příkladů zvoleného tématu.

Pomocí příkazu `zktest` lze studovat i vzorová řešení, která jsou k některým příkladům připojena, provádět aktivní přezkoušení aj.
(podrobnosti jsou v `help zktest`).

b) **Jazyky C, PASCAL, FORTRAN:**

Algoritmy:

V knihovně `d = vesely('nummet')` lze nalézt podknihovny

`mathews.c, mathews.pas a mathews.for`,

které obsahují analogické algoritmy jako knihovna MATLABu `jmathews` zpracované po řadě v jazycích C, PASCAL a FORTRAN.

Date: 3. března 2000.

TYPICKÉ SCHÉMA NUMERICKÉ ANALÝZY ŘEŠENÉHO PROBLÉMU

Obecné schéma	**Příklad**
1. REÁLNÝ (FYZIKÁLNÍ) PROBLÉM	Koule vyrobená z materiálu o hustotě $\rho = 0,638$ g/cm^3 a poloměru r [cm] je ponořena do vody. Zjistěte závislost hloubky ponoru d [cm] v závislosti na r pro $8 \leqslant r \leqslant 12$ [cm].
↓	↓
Reálný problém idealizujeme pomocí vhodného fyzikálně-matematického modelu a dopouštíme se tak **chyby formulace problému**.	Předpokládáme • ideálně homogenní materiál • ideálně čistou vodu o hustotě 1 g/cm^3 • zanedbáváme vliv povrchového napětí vody
↓	↓
2. MATEMATICKÝ PROBLÉM Problém formulujeme jako explicitní nebo implicitní funkční vztah: $y = F(x)$ nebo $f(x, y) = 0$, kde x reprezentuje vstupní data z vhodného definičního oboru a y jsou příslušná výstupní data.	Spočtěme objem vody vytlačené ponořenou částí koule: $M_V(d) = \int_0^d \pi r^2(t)\,dt = \int_0^d \pi(r^2 - (r-t)^2)\,dt = \int_0^d \pi(2rt - t^2)\,dt = \underline{\pi(3rd^2 - d^3)/3}.$ Podle Archimedova zákona hmotnost koule se musí rovnat hmotnosti vytlačené vody: $M_k := \rho 4\pi r^3/3 = 1 \cdot M_V(d)$. Obdrželi jsme tak matematický problém **nalézt d splňující (implicitní) kubickou rovnici**: $d^3 - 3rd^2 + 4\rho r^3 = 0$ pro $x = [r, \rho] \in [8, 12] \times \{\rho\},\ y = d.$
↓	↓
K nalezení numerického řešení je zpravidla nutné provést **numerickou aproximaci** (**diskretizaci**) matematického problému. Vzniklá chyba se nazývá **chybou numerické aproximace** nebo také **diskretizační chybou**.	$x \rightsquigarrow \boldsymbol{x} = [\boldsymbol{r}, \rho]$ a $y \rightsquigarrow \boldsymbol{y} = \boldsymbol{d}$, kde $\boldsymbol{r} := [r_0, r_1, \ldots, r_N]$, $\boldsymbol{d} := [d_0, d_1, \ldots, d_N]$, $r_j = 8 + j(12 - 8)/N$. Nekonečný definiční obor $[8, 12] \times \{\rho\}$ původní rovnice tedy nahradíme konečným diskrétním oborem pro $j = 0, 1, \ldots, N$.
↓	↓
3. NUMERICKÝ PROBLÉM Po diskretizaci obdržíme: $\boldsymbol{y} = F(\boldsymbol{x})$ nebo $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}$, kde $\boldsymbol{x}$ je vektor vstupních dat $\boldsymbol{y}$ je příslušný vektor výstupních dat.	Pro každé $j = 0, 1, \ldots, N$ řešíme vzhledem k d_j rovnici: $d_j^3 - 3r_j d_j^2 + 4\rho r_j^3 = 0.$
↓	↓
Zvolíme vhodnou **numerickou metodu** (postup) řešení numerického problému a provedeme její algoritmizaci. Ve složitějších případech je snaha problém rozložit na jednodušší problémy se známým řešením (výstup jednoho je vstupem jiného). Zvolená metoda rovněž vnáší do řešení tzv. **chybu metody.** Nevhodně zvolená metoda může vést k znehodnocení výsledků v důsledku kumulace například zaokrouhlovacích chyb.	Můžeme si vybrat například z těchto dvou numerických postupů: a) spočteme analytické řešení pomocí Cartanových vzorců. b) užijeme některou iterační metodu na řešení nelineárních rovnic, kdy počáteční odhad řešení se postupně zpřesňuje až do dosažení požadované přesnosti.
↓	

4. ALGORITMUS ŘEŠENÍ Počítáme $\boldsymbol{y} = F(\boldsymbol{x})$, resp. řešíme $f(\boldsymbol{x}, \boldsymbol{y})$ v konečném počtu kroků pro zadaný vstupní vektor $\boldsymbol{x}$. Některé z nich mohou samy představovat dílčí algoritmy řešící některý již známý numerický problém.	a) postupně dosazujeme do Cartanových vzorců r_j pro $j = 0, 1, \ldots, N$ a vypočítáváme příslušná reálná řešení d_j. b) zkonstruujeme konvergentní iterační schéma $\boldsymbol{d}^{(n)} = \varphi(\boldsymbol{d}^{(n-1)})$, $n = 1, 2, \ldots$ pro vhodný počáteční odhad $\boldsymbol{d}^{(0)}$ takové, že $\boldsymbol{d}^{(n)} \to \boldsymbol{d}$ pro $n \to \infty$. Za řešení považujeme $\boldsymbol{d}^{(M)} \approx \boldsymbol{d}$ takové, že $\|\boldsymbol{d} - \boldsymbol{d}^{(M)}\| < \varepsilon$ ve vhodné normě $\|\cdot\|$.
Algoritmus **kódujeme** ve vhodném programovacím jazyku, což po jeho vykonání na počítači vede k výsledkům zatíženým navíc **zaokrouhlovacími chybami** v důsledku konečné aritmetiky počítače a **chybami (nedokonale změřených) vstupních dat**.	Hustota ρ je zjištěna fyzikálním měřením s omezenou přesností.
5. IMPLEMENTACE + VÝPOČET Programová realizace kroků algoritmu v konkrétním programovacím jazyku (C,Pascal, Fortran, MATLAB) na konkrétním typu počítače.	Spočtení přibližných hodnot d_j, $j = 0, 1, \ldots, N$ zatížených vlivem všech výše popsaných chyb.

OBECNÉ DOPORUČENÍ PRO VOLBU NUMERICKÉ METODY

Při transformaci matematického problému na numerický se vždy snažíme maximálně využít všech informací o hledaném řešení. Takto získané algoritmy jsou sice speciálnější, ale na rozdíl od univerzálně použitelných jsou obvykle efektivnější (rychlejší, přesnější).

a) užití Cartanových vzorců představuje speciální metodu pro řešení kubické rovnice (dosazení do vzorce je jednoduché, rychlé a přesné).

b) užití iterační metody představuje univerzální postup použitelný pro širokou třídu nelineárních rovnic (pomalá: počet iterací závisí na požadované přesnosti).

NUMERICKÁ STABILITA

V důsledku nevhodné kumulace chyb může algoritmus, resp. numerická metoda dávat zcela znehodnocené výsledky, kdy malá chyba na vstupu má za následek velkou chybu na výstupu.

Definice. Algoritmus (resp. numerická metoda) se nazývá **numericky stabilní**, jestliže
$\forall \varepsilon > 0 \, \exists \delta > 0 : \ \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\| < \delta \ \Rightarrow \ \|\boldsymbol{y} - \widetilde{\boldsymbol{y}}\| < \varepsilon$
při provádění výpočtu se zaokrouhlovacími chybami (resp. pouze s chybou numerické metody bez zaokrouhlovacích chyb).
V opačném případě se algoritmus (resp. numerická metoda) nazývá **numericky nestabilní** a je třeba zvolit jiný algoritmus (resp. jinou numerickou metodu).

1. Úvod

1.1. **Pomocná tvrzení.**

Označení .
$\mathbb{N}$... množina všech přirozených čísel
$\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ ... množina všech nezáporných celých čísel
$\mathbb{Z}$... množina všech celých čísel
$\mathbb{R}$... množina všech reálných čísel
$\mathbb{C}$... množina všech komplexních čísel
$\mathrm{int}(x)$ ... celá část čísla x
(a, b) ... otevřený interval
$[a, b]$ ... uzavřený interval
$f(x), x \in S$ ... reálná funkce f definovaná na množině $S \subset \mathbb{R}$
$\mathcal{I}(x_0, x_1, \ldots, x_n) = \{x \mid \min(x_0, x_1, \ldots, x_n) < x < \max(x_0, x_1, \ldots, x_n)\}$
$\mathcal{I}[x_0, x_1, \ldots, x_n] = \{x \mid \min(x_0, x_1, \ldots, x_n) \le x \le \max(x_0, x_1, \ldots, x_n)\}$
$s := v$, resp. $v =: s \ldots$ označení výrazu v symbolem s.

Definice (Limita funkce)**.**
L je **limitou** funkce $f(x), x \in S$ v bodě $x_0 \Leftrightarrow \forall \varepsilon > 0\ \exists \delta > 0 :\ |x - x_0| < \delta \Rightarrow |f(x) - L| < \varepsilon$.
Píšeme

$$\begin{aligned} \lim_{x \to x_0} f(x) &= L \\ \lim_{h \to 0} f(x_0 + h) &= L \end{aligned}$$

Definice (Spojitost funkce)**.**
Řekneme, že $f(x)$ je **spojitá** v bodě x_0, jestliže

$$\begin{aligned} \lim_{x \to x_0} f(x) &= f(x_0) \\ \lim_{h \to 0} f(x_0 + h) &= f(x_0) \end{aligned}$$

Značíme

$C(S), C[a, b]$ ... množina všech funkcí spojitých v každém bodě $x \in S$, resp. $x \in [a, b]$.

Definice (Limita posloupnosti a součet nekonečné řady)**.**
L je **limitou** posloupnosti $\{x_n\}_{n=1}^{\infty} \Leftrightarrow \forall \varepsilon > 0\ \exists N = N(\varepsilon) \in \mathbb{N} : |x_n - L| < \varepsilon\ \forall n \ge N$.
Píšeme

$$\begin{aligned} \lim_{n \to \infty} x_n &= L \\ \lim_{n \to \infty} (L - x_n) &= 0 \\ x_n \to L \quad &\text{pro} \quad n \to \infty. \end{aligned}$$

$\{\varepsilon_n\}_{n=1}^{\infty},\ \varepsilon_n = L - x_n$ ... posloupnost chyb.

$s_n = \sum_{k=1}^{n} x_k$ ... částečný součet nekonečné řady
$\lim_{n \to \infty} s_n = s := \sum_{k=1}^{\infty} x_k$ ... součet nekonečné řady.

Věta 1.1.
$f(x),\ x \in S$ *je spojitá v* $x_0 \Leftrightarrow \forall \{x_n\}_{n=1}^{\infty},\ x_n \in S,\ \lim_{n \to \infty} x_n = x_0$ *platí* $\lim_{n \to \infty} f(x_n) = f(x_0)$.

Věta 1.2 (Bolzanova věta)**.**
$f \in C[a, b],\ L \in \mathcal{I}[f(a), f(b)] \Rightarrow \exists c \in [a, b] :\ f(c) = L$.

Věta 1.3 (Weierstrasseova věta)**.**
$f \in C[a, b] \Rightarrow \exists\ M_1 = \min_{x \in [a,b]} f(x),\ M_2 = \max_{x \in [a,b]} f(x)$ *a* $x_1, x_2 \in [a, b] :\ M_1 = f(x_1),\ M_2 = f(x_2)$.

Důsledek 1.4. $f \in C[a, b]$ *nabývá na* $[a, b]$ *všech hodnot mezi svou nejmenší a největší hodnotou (plyne z 1.2 a 1.3).*

Definice (Derivace funkce)**.**
Nechť $f(x)$ je definována v nějakém okolí bodu x_0: $x \in (x_0 - \delta, x_0 + \delta),\ \delta > 0$.

Pak limitu (pokud existuje)

$$\lim_{x\to x_0}\frac{f(x)-f(x_0)}{x-x_0}=:f'(x_0)$$
$$\lim_{h\to 0}\frac{f(x_0+h)-f(x_0)}{h}=:f'(x_0)$$

nazýváme **derivací** funkce $f(x)$ v bodě x_0.

Značíme

$C^n(S), C^n[a,b],\ n\in\mathbb{N},\ n\geq 0\ldots$ množina všech funkcí majících v každém bodě $x\in S$, resp. $x\in[a,b]$ spojité všechny derivace až do řádu n ($C^0:=C$).

Věta 1.5. *$f'(x_0)$ existuje $\Rightarrow$ $f(x)$ je spojitá v bodě x_0.*

Věta 1.6 (Rolleova věta)**.**
$f\in C[a,b],\ f'(x)$ existuje na $(a,b),\ f(a)=f(b)=0\ \Rightarrow\ \exists\, c\in(a,b):\ f'(c)=0$.

Důsledek 1.7 (Zobecněná Rolleova věta)**.**
Nechť $f\in C[a,b],\ f'(x),\ldots,f^{(n)}(x)$ existují na (a,b) a $f(x_0)=\cdots=f(x_n)=0$ pro $x_0,\ldots,x_n\in[a,b]$. Pak $\exists\, c\in(a,b):\ f^{(n)}(c)=0$.

Důkaz. Indukcí vzhledem k n (bez újmy na obecnosti lze předpokládat $x_0<x_1<\cdots<x_n$):

1) Pro $n=1$ plyne tvrzení z 1.6 pro $a=x_0,\ b=x_1$.
2) Nechť $n>1$ a tvrzení platí pro $n-1$. Pak $\exists\, c_i\in(x_{i-1},x_i),\ i=1,\ldots,n:\ f'(c_i)=0$ dle 1.6. Podle indukčního předpokladu $\exists\, c\in(a,b):\ f'^{(n-1)}(c)=f^{(n)}(c)=0$. □

Věta 1.8 (Věta o extrémech diferencovatelné funkce)**.**
$f\in C[a,b],\ f'(x)$ existuje na $(a,b),\ f(x_1)=\min_{x\in[a,b]}f(x),\ f(x_2)=\max_{x\in[a,b]}f(x)\ \Rightarrow\ x_i=a$ nebo $x_i=b$ nebo $f'(x_i)=0$ pro $i=1,2$.

Věta 1.9 (Lagrangeova věta o střední hodnotě pro derivace)**.**
$f\in C[a,b]$ a existuje vlastní nebo nevlastní $f'(x)$ na $(a,b)\ \Rightarrow\ \exists\, c\in(a,b)$:

$$f'(c)=\frac{f(b)-f(a)}{b-a}.$$

Věta 1.10 (Věta o primitivní funkci)**.**
$f\in C[a,b]\ \Rightarrow\ \exists$ funkce $F(x)\in C^1[a,b]$ nazývaná **primitivní funkce** *nebo také* **antiderivace** *funkce $f(x)$ taková, že $F'(x)=f(x)$, přičemž platí $\int_a^b f(x)\,dx=F(b)-F(a)$.*
Zejména tedy můžeme psát

$$\frac{d}{dx}\int_a^x f(t)\,dt=f(x).$$

Věta 1.11 (1. Věta o střední hodnotě pro integrály)**.**
$f\in C[a,b]\ \Rightarrow\ \exists\, c\in(a,b)$:

$$\int_a^b f(x)\,dx=f(c)(b-a).$$

Věta 1.12 (2. Věta o střední hodnotě pro integrály)**.**
$f\in C[a,b],\ g(x)$ integrovatelná a neměnící znaménko na $[a,b]\ \Rightarrow\ \exists\, c\in(a,b)$:

$$\int_a^b f(x)g(x)\,dx=f(c)\int_a^b g(x)\,dx.$$

Poznámka (Riemannova suma).
Nechť $f\in C[a,b],\ a=x_0<x_1<\cdots<x_n=b,\ t_k\in(x_{k-1},x_k),\ \Delta x_k=x_k-x_{k-1}$.

Pak

$$\sum_{k=1}^n f(t_k)\Delta x_k\xrightarrow{n\to\infty}\int_a^b f(x)\,dx.$$

$\sum_{k=1}^n f(t_k)\Delta x_k$ je tzv. **Riemannova suma**, pomocí níž lze $\int_a^b f(x)\,dx$ aproximovat s libovolnou přesností, jestliže zvolíme n dostatečně velké.

Věta 1.13 (Taylorova věta)**.**
$f \in C^{n+1}[a,b],\ x_0 \in [a,b] \ \Rightarrow\ pro\ \forall\, x \in [a,b]\ \exists\, \xi = \xi(x) \in \mathcal{I}(x_0, x)$:

$$\begin{aligned} f(x) &= P_n(x) + R_n(x) \\ P_n(x) &= f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n \\ R_n(x) &= \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}. \end{aligned}$$

Důsledek 1.14.
Polynom $P(x)$ stupně n je jednoznačně určen svými derivacemi $P(x_0),\ P'(x_0), \ldots, P^{(n)}(x_0)$ v libovolném bodě x_0.

Důkaz. Položíme-li $f = P$, pak $f^{n+1}(x) \equiv 0 \ \Rightarrow\ R_n(x) \equiv 0$ pro každé x_0. □

Věta 1.15 (Taylorova věta pro funkce více proměnných)**.**
Nechť funkce $f(\boldsymbol{x}) := f(x_1, x_2, \ldots, x_m)$ má v bodě $\boldsymbol{x}_0 = [x_1^0, x_2^0, \ldots, x_m^0]$ a jeho nějakém okolí spojité parciální derivace až do řádu $n+1$ včetně. Pak pro libovolný bod $\boldsymbol{x}$ tohoto okolí existuje $\boldsymbol{\xi} = [x_1^0 + \xi_1(x_1 - x_1^0), x_2^0 + \xi_2(x_2 - x_2^0), \ldots, x_m^0 + \xi_m(x_m - x_m^0)],\ 0 < \xi_i < 1$ tak, že

$$\begin{aligned} f(\boldsymbol{x}) &= P_n(\boldsymbol{x}) + R_n(\boldsymbol{x}) \\ P_n(\boldsymbol{x}) &= f(\boldsymbol{x}_0) + \frac{\mathrm{d}f(\boldsymbol{x}_0)}{1!} + \frac{\mathrm{d}^2 f(\boldsymbol{x}_0)}{2!} + \cdots + \frac{\mathrm{d}^n f(\boldsymbol{x}_0)}{n!} \\ R_n(\boldsymbol{x}) &= \frac{\mathrm{d}^{n+1} f(\boldsymbol{\xi})}{(n+1)!}, \end{aligned}$$

kde

$$\mathrm{d}^k f(\boldsymbol{x}_0) = \left(\frac{\partial}{\partial x_1}(x_1 - x_1^0) + \frac{\partial}{\partial x_2}(x_2 - x_2^0) + \cdots + \frac{\partial}{\partial x_n}(x_m - x_m^0) \right)^k f(\boldsymbol{x_0})$$

pro $k = 1, 2, \ldots$.

Důsledek 1.16. *Speciálně pro funkci f o m proměnných, která má spojité všechny parciální derivace až do řádu 2 v nějakém okolí bodu $\boldsymbol{x}_0 = [x_1^0, x_2^0, \ldots, x_m^0]$ platí pro každé $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]$ z tohoto okolí:*

$$f(x_1, x_2, \ldots, x_m) \approx f(\boldsymbol{x}_0) + \frac{\partial f(\boldsymbol{x}_0)}{\partial x_1}(x_1 - x_1^0) + \frac{\partial f(\boldsymbol{x}_0)}{\partial x_2}(x_2 - x_2^0) + \cdots + \frac{\partial f(\boldsymbol{x}_0)}{\partial x_n}(x_m - x_m^0).$$

1.2. **Vyhodnocení polynomu — Hornerovo schéma.**
(Příklady v MATLABu: `zktest([],'horner')`)

$$\boxed{P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0, \quad P(z) = ?}$$

Přímý výpočet:

$$\begin{array}{llllll} z^2 = zz & z^3 = z^2 z & \ldots & z^n = z^{n-1}z & \Rightarrow & n-1 \text{ násobení} \\ a_1 z & a_2 z^2 & \ldots & a_n z^n & \Rightarrow & n \text{ násobení} \\ a_0 + a_1 z & + \ldots + & a_n z^n & & \Rightarrow & n \text{ sečítání} \end{array}$$

$$\Downarrow$$

$$\boxed{\text{Celkem } 2n-1 \text{ násobení a } n \text{ sečítání}}$$

Princip Hornerova schématu:

$$\underbrace{(\ldots((}_{n-1} a_n z + a_{n-1})z + a_{n-2})z + \cdots + a_1)z + a_0$$

$$\Downarrow$$

$$\boxed{\text{Celkem } n \text{ násobení a } n \text{ sečítání} = \textbf{teoretické minimum}}$$

Věta 1.17 (Hornerovo schéma)**.**
$P(x) = P_1(x)(x-z) + P_0$, $P_1(x) = b_n x^{n-1} + b_{n-1}x^{n-2} + \cdots + b_1$, $b_0 := P_0 = P(z)$, kde

$$\left.\begin{array}{rcl} b_n & = & a_n \\ b_k & = & a_k + z b_{k+1}, \ k = n-1, n-2, \ldots, 1, 0 \end{array}\right\} \Rightarrow n \text{ násobení a } n \text{ sečítání.}$$

Důkaz.

$$\begin{array}{lllllllllllll} P_1(x)x + P_0 & = & b_n x^n & + & b_{n-1}x^{n-1} & + & \ldots & + & b_2 x^2 & + & b_1 x & + & b_0 \\ -zP_1(x) & = & & - & z b_n x^{n-1} & - & \ldots & - & b_3 z x^2 & - & b_2 z x & - & b_1 z \\ \hline P(x) & = & a_n x^n & + & a_{n-1}x^{n-1} & + & \ldots & + & a_2 x_2 & + & a_1 x & + & a_0 \end{array}$$

Odtud porovnáním koeficientů obdržíme požadované vztahy pro b_k. □

Algoritmus.

$$\begin{array}{llllllllll} P: & a_n & & a_{n-1} & & a_{n-2} & \ldots & a_1 & & a_0 \\ & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & \ldots & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = \\ & b_n & & b_{n-1} & & b_{n-2} & \ldots & b_1 & & \boxed{b_0 = P(z)} \end{array}$$

Důsledek 1.18 (Rozšířené Hornerovo schéma)**.**
$P_0(x) := P(x) = P_n(z)(x-z)^n + \cdots + P_2(z)(x-z)^2 + P_1(z)(x-z) + P_0(z)$, *kde*
$P_k(x) = P_{k+1}(x)(x-z) + P_k(z)$, $k = 0, 1, \ldots, n-1$.

Důkaz. Rozklad z věty 1.17 aplikujeme n-krát postupně na $P_0(x), P_1(x), \ldots, P_{n-1}(x)$:
$P_0(x) =$
$(\ldots(\underbrace{\underbrace{(P_n(z)(x-z) + P_{n-1}(z))}_{P_{n-1}(x)}(x-z) + P_{n-2}(z)}_{P_{n-2}(x)})(x-z) + \cdots + P_1(z))(x-z) + P_0(z).$ □

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{P_1(x)}$$

Důsledek 1.19 (Výpočet k-té derivace pomocí rozšířeného Hornerova schématu)**.**
$P_0^{(k)}(z) = k! P_k(z)$ *pro* $k = 0, 1, \ldots, n$.

Důkaz.

$$\left[P_j(z)(x-z)^j\right]^{(k)}_{x=z} = \begin{cases} 0 & \text{pro} \quad j \neq k \\ k! P_k(z) & \text{pro} \quad j = k \end{cases}.$$

□

Algoritmus.

$$
\begin{array}{lccccccccccccc}
P_0: & a_n & & a_{n-1} & & a_{n-2} & \dots & a_2 & & a_1 & & a_0 & & \\
 & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & \dots & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & & \\
P_1: & b_n^{(1)} & & b_{n-1}^{(1)} & & b_{n-2}^{(1)} & \dots & b_2^{(1)} & & b_1^{(1)} & & b_0^{(1)} & = & \boxed{P_0(z)} \\
 & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & \dots & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & & & & \\
P_2: & b_n^{(2)} & & b_{n-1}^{(2)} & & b_{n-2}^{(2)} & \dots & b_2^{(2)} & & b_1^{(2)} & = & \boxed{P_1(z)} & & \\
 & & & \vdots & & & \dots & & & \vdots & & & & \\
 & \downarrow = & \overset{\times z+}{\nearrow} & \downarrow = & & & & & & & & & & \\
P_n: & b_n^{(n)} & & b_{n-1}^{(n)} & = & \boxed{P_{n-1}(z)} & & & & & & & & \\
 & \downarrow = & & & & & & & & & & & & \\
 & \boxed{P_n(z)} & & & & & & & & & & & &
\end{array}
$$

1.3. **Analýza chyb.**
(Příklady v MATLABu: `zktest([],'numchyby')`)

Definice 1.20. Nechť $\widetilde{p}$ je aproximace čísla p ($\widetilde{p} \approx p$). Pak $E_p = p - \widetilde{p}$ nazýváme **absolutní chybou** a $R_p = \frac{p-\widetilde{p}}{p}$, $p \neq 0$ nazýváme **relativní chybou** této aproximace. Píšeme též $p = \widetilde{p} \pm \varepsilon_p$, pokud $|E_p| \leq \varepsilon_p$ (tj. $p \in [\widetilde{p} - \varepsilon_p, \widetilde{p} + \varepsilon_p]$).

Příklad 1.21. R_p je objektivnější indikátor, neboť chybu vztahuje relativně k velikosti čísla:

(1) $x = 3,141592,\ \widetilde{x} = 3,14 \Rightarrow E_x = 0,001592,\ R_x = 0,000507$
(2) $y = \underbrace{1\,000\,000}_{\text{velké}},\ \widetilde{y} = 999\,996 \Rightarrow \underbrace{E_y = 4}_{\text{velká}},\ \underbrace{R_y = 0,000004}_{\text{malá}}$
(3) $z = \underbrace{0,000012}_{\text{malé}},\ \widetilde{z} = 0,000009 \Rightarrow \underbrace{E_z = 0,000003}_{\text{malá}},\ \underbrace{R_z = 0,25}_{\text{velká}}$

Definice 1.22. Říkáme, že $\widetilde{p}$ **aproximuje** p **na** d **významných cifer**, jestliže d je největší nezáporné celé číslo takové, že

$$\boxed{|R_p| = \frac{|p-\widetilde{p}|}{|p|} < \frac{10^{-d}}{2} = 5 \times 10^{-(d+1)}}$$

Příklad 1.23. Určíme počet významných cifer pro aproximace z příkladu 1.21:

(1) $R_x = 0,000507 \approx \frac{10^{-3}}{2} \Rightarrow \boxed{d \approx 3}$ ($\widetilde{x} = 3,14\ldots3$ platné cifry dle očekávání)
(2) $0,0000005 < R_y = 0,000004 < 0,000005 \Rightarrow \boxed{d = 5}$
(3) $0,05 < R_z = 0,25 < 0,5 \Rightarrow \boxed{d = 0}$

DRUHY CHYB

A) **Chyba formulace problému** (reálný problém $\rightsquigarrow$ matematický problém)
Tuto chybu nelze zpravidla kvantifikovat, pouze můžeme prověřit, zda spočtené výsledky jsou v souladu s realitou.

B) **Chyba numerické aproximace**
(matematický problém $\rightsquigarrow$ numerický problém $\rightsquigarrow$ algoritmus)
Tato chyba se podle okolností nazývá
- *Diskretizační chyba*: např. při výpočtu $\int_a^b f(x)\,dx$ aproximujeme $f(x)$ po částech konstantní funkcí (viz *Riemannovu sumu* v odst. 1.1).
- *Chyba useknutím*: vzniká v situacích, kdy nějaký nekonečný limitní proces ukončíme po konečném počtu kroků, např. nekonečnou řadu $\sum_{n=1}^{\infty} x_n$ usekneme na její částečný součet $\sum_{n=1}^{N} x_n$.

Obecně lze říci, že chyby tohoto druhu vznikají při nahrazení komplikovaného matematického výrazu jednodušší formulí. Snažíme se přitom nalézt horní odhad absolutní nebo relativní chyby, které jsme se dopustili.

Za typický příklad může posloužit přibližný výpočet funkční hodnoty $f(x) \approx P_n(x)$ pomocí *Taylorova rozvoje* 1.13 v okolí nějakého vhodného blízkého bodu x_0. Potom

$$E_{f(x)} = f(x) - P_n(x) = R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1},\ \xi \in \mathcal{I}(x, x_0).$$

Pokud známe horní odhad $|f^{(n+1)}(t)| \leq L$ pro $t \in \mathcal{I}(x, x_0)$, spočteme snadno horní odhad pro $E_{f(x)}$: $|E_{f(x)}| \leq \frac{L}{(n+1)!}|x - x_0|^{n+1} =: \varepsilon_{f(x)}$.

Příklad 1.24. $e^u = 1 + u + \frac{u^2}{2!} + \frac{u^3}{3!} + \ldots$ je Taylorův rozvoj v bodě $u = 0$.
Pro $u = x^2,\ x \in \left[0, \frac{1}{2}\right],\ n = 4$ dostáváme $L = \max_{u \in \left[0, \frac{1}{4}\right]} |e^u| = e^{\frac{1}{4}}$ (totiž $\frac{d^5}{du^5} e^u = e^u$) a odtud

$$e^{x^2} \approx 1 + x^2 + \frac{x^4}{2!} + \frac{x^6}{3!} + \frac{x^8}{4!},\ |E_{e^{x^2}}| \leq \frac{\sqrt[4]{e}|x^{10}|}{5!} \leq \frac{\sqrt[4]{e}}{5!2^{10}} = 1,045 \times 10^{-5} \text{ pro } x \in \left[0, \frac{1}{2}\right].$$

Pomocí této aproximace pak přibližně spočteme integrál:

$$\begin{aligned} p &= 0,544987104184 = \int_0^{1/2} e^{x^2}\,dx \approx \\ &\approx \left[x + \frac{x^3}{3} + \frac{x^5}{2!5} + \frac{x^7}{3!7} + \frac{x^9}{4!9}\right]_0^{\frac{1}{2}} = 0,544986720817 = \widetilde{p} \\ E_p &= p - \widetilde{p} = 0,000000383367,\ |E_p| < \frac{1}{2}\frac{\sqrt[4]{e}}{5!2^{10}} = \frac{\sqrt[4]{e}}{5!2^{11}} \doteq 5,225 \times 10^{-6} =: \varepsilon_p. \end{aligned}$$

$$\frac{10^{-6}}{2} < R_p \doteq 7,03442 \times 10^{-7} < \frac{10^{-5}}{2} \quad \Rightarrow \quad \text{aproximace na } \mathbf{5 \text{ významných cifer}}$$

$$R_p \approx \widetilde{R}_p = \frac{E_p}{\widetilde{p}} \le \frac{\varepsilon_p}{\widetilde{p}} \doteq 9,587 \times 10^{-6} \quad \Rightarrow \quad \text{aproximace na } \mathbf{4 \text{ významné cifry}}$$
(pesimističtější než skutečnost)

POZOR! Aproximace $R_p \approx \widetilde{R}_p$ může být nebezpečně zkreslující pro p (a tedy i pro $\widetilde{p}$) blízké k nule.

Podobně jako v předchozím příkladě je často funkce $f(h)$ nahrazena aproximací $\widetilde{f}(h)$, přičemž je znám horní odhad chyby ve tvaru $M|h^n|,\ M > 0$. Toto vede k následující definici.

Definice 1.25.

(1) Nechť $f(h) \approx \widetilde{f}(h)$ a $\exists M > 0$ a $n \in \mathbb{N}$ tak, že $|f(h) - \widetilde{f}(h)| \le M|h^n|$ pro dostatečně malá h.
Pak říkáme, že $\widetilde{f}(h)$ **aproximuje** $f(h)$ **s řádem aproximace** $O(h^n)$ a píšeme

$$\boxed{f(h) = \widetilde{f}(h) + O(h^n) \quad \text{pro} \quad h \to 0}\,.$$

(2) Jestliže $\lim_{n\to\infty} a_n = a,\ \lim_{n\to\infty} r_n = 0$ a $\exists K > 0$ tak, že $|a_n - a| < K|r_n|$ pro dostatečně velké n, pak říkáme, že a_n **konverguje k** a **s řádem konvergence** $O(r_n)$ a píšeme

$$\boxed{a_n = a + O(r_n) \quad \text{pro} \quad n \to \infty}\,.$$

Poznámka 1.26. Předchozí definici lze ještě dále zobecnit v následujícím smyslu:
Nechť a je konečné nebo $a = \pm\infty$, pak píšeme

(1) $f(x) = O(g(x))$ pro $x \to a \ \Leftrightarrow\ \exists M > 0$ a $\varepsilon > 0$:
$|f(x)| \le M|g(x)|$ pro $\forall x,\ |x - a| < \varepsilon$
Interpretace: v okolí bodu a se $f(x)$ neliší významně od $g(x)$.

(2) $f(x) = o(g(x))$ pro $x \to a \ \Leftrightarrow\ \lim_{x\to a} \frac{f(x)}{g(x)} = 0$
Interpretace: v okolí bodu a konverguje $f(x)$ k nule rychleji, než $g(x)$.

Věta 1.27. *Nechť $f(h) = \widetilde{f}(h) + O(h^n)$ pro $h \to 0$ a $g(h) = \widetilde{g}(h) + O(h^m)$ pro $h \to 0$ a $r = \min(n, m)$. Pak*

(1) $f(h) \pm g(h) = \widetilde{f}(h) \pm \widetilde{g}(h) + O(h^r)$.
(2) $f(h)\,g(h) = \widetilde{f}(h)\,\widetilde{g}(h) + O(h^r)$, *pokud f a g jsou v nějakém okolí nuly ohraničené.*
(3) $\frac{1}{g(h)} = \frac{1}{\widetilde{g}(h)} + O(h^m)$, *pokud v nějakém okolí nuly je $|g(h)| \ge \sigma > 0$ pro vhodné σ.*
(4) $\frac{f(h)}{g(h)} = \frac{\widetilde{f}(h)}{\widetilde{g}(h)} + O(h^r)$, *pokud pokud f a $\frac{1}{g}$ jsou v nějakém okolí nuly ohraničené.*

Důkaz. Nechť pro dosti malá h je $|f(h) - \widetilde{f}(h)| \le M_1|h|^n$ a $|g(h) - \widetilde{g}(h)| \le M_2|h|^m$. Potom

(1) $|f(h) \pm g(h) - (\widetilde{f}(h) \pm \widetilde{g}(h))| \le |f(h) - \widetilde{f}(h)| + |g(h) - \widetilde{g}(h)| \le$
$M_1|h|^n + M_2|h|^m = \underbrace{(M_1|h|^{n-r} + M_2|h|^{m-r})}_{\le M}|h|^r$.

(2) $|fg - \widetilde{f}\widetilde{g}| = |fg - \widetilde{f}g + \widetilde{f}g - \widetilde{f}\widetilde{g}| \le |g||f - \widetilde{f}| + |\widetilde{f}||g| - \widetilde{g}| \le$
$\underbrace{|g|}_{\le L_1} M_1|h|^n + \underbrace{|\widetilde{f}|}_{\le L_2} M_2|h|^m \le \underbrace{(L_1M_1|h|^{n-r} + L_2M_2|h|^{m-r})}_{\le M}|h|^r$,

kde skutečně $\widetilde{f}(h)$ je ohraničená pro dosti malá h ($|\widetilde{f}(h)| \le L_2$), neboť
$|f(h) - \widetilde{f}(h)| \le M_2|h|^n \Rightarrow |f(h) - \widetilde{f}(h)| \to 0$ pro $h \to 0 \Rightarrow$
$f(h) - \varepsilon < \widetilde{f}(h) < f(h) + \varepsilon$ pro dosti malá h, přičemž $f(h)$ je ohraničená dle předpokladu.

(3) Platí $1/|g\widetilde{g}| \le 2/\sigma^2$, neboť
$\sigma \le |g| = |g| - |\widetilde{g}| + |\widetilde{g}| \le ||g| - |\widetilde{g}|| + |\widetilde{g}| \le |g - \widetilde{g}| + |\widetilde{g}| \le \sigma/2 + |\widetilde{g}|$,
kde skutečně pro dosti malá h je $|g - \widetilde{g}| \le \sigma/2$ vzhledem k
$|g(h) - \widetilde{g}(h)| \to 0$ pro $h \to 0$. Pak
$|1/g - 1/\widetilde{g}| = \underbrace{(1/|g\widetilde{g}|)}_{\le 2/\sigma^2} |g - \widetilde{g}| \le \underbrace{(2M_2/\sigma^2)}_{M} |h|^m$.

(4) je přímým důsledkem (2) a (3).

□

Věta 1.28 (Řád aproximace Taylorovým polynomem)**.**
Nechť $f \in C^{n+1}[a,b]$; $x_0, x_0 + h \in [a,b]$, potom

$$\boxed{f(x_0 + h) = \textstyle\sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} h^k + O(h^{n+1})}\,.$$

Důkaz. Vyjádření plyne z věty 1.13, jestliže položíme $h := x - x_0$. Pak
$|R_n(x_0 + h)| = |f(x_0 + h) - P_n(x_0 + h)| = \left|\frac{f^{n+1}(\xi(h))}{(n+1)!}\right| |h|^{n+1}$, kde $\xi(h) \in \mathfrak{I}[x_0, x_0 + h]$ a $f^{(n+1)}(x)$ je na tomto intervalu spojitá a tedy i ohraničená podle 1.3. □

Příklad 1.29 (O-aritmetika)**.**
Pro $e^h = 1 + h + \frac{h^2}{2!} + \frac{h^3}{3!} + O(h^4)$ a $\cos(h) = 1 - \frac{h^2}{2!} + \frac{h^4}{4!} + O(h^6)$
spočteme formálně řád aproximace pro jejich součet a součin.
a) součet:

$$\begin{aligned} e^h + \cos(h) &= 1 + h + \frac{h^2}{2!} + \frac{h^3}{3!} + O(h^4) + 1 - \frac{h^2}{2!} + \frac{h^4}{4!} + O(h^6) = \\ &= 2 + h + \frac{h^3}{3!} + \underbrace{\underbrace{\frac{h^4}{4!}}_{O(h^4)} + \underbrace{O(h^4) + O(h^6)}_{O(h^4)}}_{O(h^4)}. \end{aligned}$$

b) součin:

$$\begin{aligned} e^h \cos(h) &= \underbrace{\left(1 + h + \frac{h^2}{2!} + \frac{h^3}{3!}\right)\left(1 - \frac{h^2}{2!} + \frac{h^4}{4!}\right)}_{1 + h + \frac{h^3}{3!} - \frac{h^3}{2!} + O(h^4)} + \underbrace{\left(1 + h + \frac{h^2}{2!} + \frac{h^3}{3!}\right) O(h^6)}_{O(h^6)} + \\ &+ \underbrace{\underbrace{\left(1 - \frac{h^2}{2!} + \frac{h^4}{4!}\right) O(h^4)}_{O(h^4)} + \underbrace{O(h^4) O(h^6)}_{O(h^{10})}}_{O(h^4)} = \\ &= 1 + h - \frac{h^3}{3} + \underbrace{O(h^4) + O(h^6) + O(h^4)}_{O(h^4)}. \end{aligned}$$

Vidíme, že tytéž výsledky bylo možno obdržet přímou aplikací obecné věty 1.27.

C) **Chyby vstupních dat** (vstup $\rightsquigarrow$ výpočet programem)
Zdrojem těchto chyb jsou nepřesnosti vzniklé při pořizování vstupních dat. V některých případech je znám apriori jejich horní odhad (například chyba kalibrace měřícího přístroje), jindy nikoliv (např. chyby lidského činitele).

D) **Strojová chyba** (algoritmus $\rightsquigarrow$ výpočet programem)
Zdrojem chyb během výpočtu je nepřesné zobrazování čísel v paměti počítače jako důsledek její konečné velikosti.

(1) Strojová reprezentace celých čísel na n bitů (počítání modulo 2^n)

(i) *bez znaménka (unsigned integer)*
$a \ge 0 \Rightarrow$ rozsah: $0 \le a \le 2^n - 1 = (\underbrace{11\ldots1}_{n})_2$

(*ii*) *se znaménkem (signed integer)*
a libovolné $\Rightarrow$ rozsah: $(\underbrace{10\ldots0}_{n})_2 = -2^{n-1} \le a \le 2^{n-1}-1 = (\underbrace{01\ldots1}_{n})_2$.
Současně vidíme, že prvý bit určuje znaménko: 1=minus, 0=plus.
Je zřejmé, že **v rámci uvedených rozsahů jsou celočíselné výpočty absolutně přesné**, zatímco **mimo ně naopak zcela chybné**.

Příklad 1.30 ($n = 3$)**.**

(*i*) $0 \le a \le 7$: $\overset{\text{mod } 8}{\boxed{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7}}\ 0\ 1\ 2\ 3\ldots$

(*ii*) $-4 \le a \le 3$: $0\ \text{-}7\ \text{-}6\ \text{-}5\ \overset{\text{mod } 8}{\boxed{\text{-}4\ \text{-}3\ \text{-}2\ \text{-}1\ 0\ 1\ 2\ 3}}\ldots$

K číslu a najdeme v modulární aritmetice číslo opačné $-a$ snadno z rovnice: $\underbrace{a+\bar{a}}_{(11\ldots1)_2} +1 =$ $2^n \equiv 0 \mod 2^n$. Odtud dostáváme $-a = \bar{a}+1$, kde $\bar{a}$ vznikne z a negací po bitech.
Například

$$\begin{array}{rcl} a=2 & = & (010)_2 \\ \bar{2} & = & (101)_2 \\ \hline 2+\bar{2} & = & (111)_2 \\ & \Downarrow & \\ -2=\bar{2}+1 & = & (110)_2 \\ & = & 6 \mod 8. \end{array}$$

V počítačích se celá čísla zobrazují zpravidla v těchto přesnostech:

$n=8$	(1 bajt)	...	(un)signed char (1 znak)
$n=16$	(2 bajty)	...	(un)signed short integer (poloviční přesnost)
$n=32$	(4 bajty)	...	(un)signed integer (jednoduchá přesnost)
$n=64$	(8 bajtů)	...	(un)signed long integer (dvojnásobná přesnost)

(2) Strojová reprezentace reálných čísel na n bitů
Nechť $q \ge 2$ značí základ číselné soustavy. V počítačích pracujeme s čísly nejčastěji v soustavách $q = 2, 8, 16$. Přesná reálná čísla se reprezentují v tzv. **semilogaritmickém tvaru pohyblivé řádové čárky** (normalizovaná mantisa + exponent):

$$\boxed{p = \overbrace{\pm d_1, d_2 d_3 \ldots d_k d_{k+1} \ldots}^{\text{mantisa}} \times q^e},$$

kde $e \in \mathbb{Z}$ je exponent a $1 \le d_1 \le q-1,\ 0 \le d_j \le q-1$ (pro $j > 1$) jsou cifry mantisy. Zejména v případě $q = 2$ je $d_1 = 1$, takže tento bit je možno využít pro zobrazení znaménka mantisy (1=minus, 0=plus).

Strojová reálná čísla se ukládají pouze s konečným počtem k cifer mantisy. Obdržíme tak přibližnou reprezentaci, která vznikne buď pouhým **odseknutím (chopping)** přebývajících cifer nebo se navíc poslední k-tá cifra d_k **zaokrouhlí (rounding)**. Současně se také vhodně **omezí rozsah exponentu**: $-e_{\min} \le e \le e_{\max}$. Dostáváme tedy tyto aproximace:

a) $\widetilde{p} = \mathrm{fl}_{\mathrm{chop}}(p) = \pm d_1, d_2 d_3 \ldots d_k \times q^e$,
s absolutní chybou aproximace $0 \le |p - \widetilde{p}| < q^{e-(k-1)}$,

b) $\widetilde{p} = \mathrm{fl}_{\mathrm{round}}(p) = \pm d_1, d_2 d_3 \ldots d_{k-1} \widetilde{d_k} \times q^e,\ \ \widetilde{d_k} = \mathrm{round}(d_k, d_{k+1} \ldots)$
s absolutní chybou aproximace $0 \le |p - \widetilde{p}| \le \frac{1}{2} q^{e-(k-1)}$.

Příklad 1.31.

(a) $q = 10,\ k = 6$
$p = \frac{22}{7} = 3,142857\overline{142857} \Rightarrow \left\{ \begin{array}{rcl} \mathrm{fl}_{\mathrm{chop}}(p) & = & 3,14285 \times 10^0 \\ \mathrm{fl}_{\mathrm{round}}(p) & = & 3,14286 \times 10^0 \end{array} \right.$

(b) $q = 2,\ k = 5$
$p = 0{,}1 = (1,1001\overline{1001})_2 \times 2^{-4} \Rightarrow \left\{ \begin{array}{rcl} \mathrm{fl}_{\mathrm{chop}}(p) & = & (1,1001)_2 \times 2^{-4} \\ \mathrm{fl}_{\mathrm{round}}(p) & = & (1,1010)_2 \times 2^{-4} \end{array} \right.$
Tento příklad je současně ilustrací čísla, které má **konečný počet cifer v dekadické soustavě, ale nekonečný počet cifer v binární soustavě** a **není** tedy v paměti počítače zobrazeno přesně.

V počítačích se reálná čísla zobrazují zpravidla v těchto přesnostech:

a) jednoduchá přesnost (4 bajty):
$n = 32 = 24$ bitů mantisy $+ 8$ bitů exponentu
Rozsah exponentu: $\underbrace{-2^7}_{-128} \le e \le \underbrace{2^7 - 1}_{127}$
Dekadický rozsah:
$2,938736 \times 10^{-39}$ až $1,701412 \times 10^{38}$, kde
$2,938736 \times 10^{-39} \doteq 1 \times 2^{-128}$ a
$1,701412 \times 10^{38} \doteq \underbrace{(1,11\ldots1)_2}_{\approx 2} \times 2^{127} \doteq 2 \times 2^{127} = 2^{128}$
Dekadická přesnost mantisy:
$2^{-23} \doteq 1,2 \times 10^{-7} \Rightarrow$ **cca 7 dekadických cifer přesnosti**, což však vzhledem k příkladu 1.31(b) neznamená, že každé číslo s nejvýše 7 dekadickými ciframi musí být zobrazeno přesně.

b) dvojnásobná přesnost (8 bajtů):
$n = 64 = 53$ bitů mantisy $+ 11$ bitů exponentu
Rozsah exponentu: $\underbrace{-2^{10}}_{-1024} \le e \le \underbrace{2^{10} - 1}_{1023}$.
Dekadický rozsah:
$5,562684646268003 \times 10^{-309}$ až $8,988465674311580 \times 10^{307}$, kde
$5,562684646268003 \times 10^{-309} \doteq 1 \times 2^{-1024}$ a
$8,988465674311580 \times 10^{307} \doteq \underbrace{(1,11\ldots1)_2}_{\approx 2} \times 2^{1023} \doteq 2 \times 2^{1023} = 2^{1024}$.
Dekadická přesnost mantisy:
$2^{-52} \doteq 2,2 \times 10^{-16} \Rightarrow$ **cca 16 dekadických cifer přesnosti**.

Běžně používaná binární reprezentace dle normy IEEE (např. počítače třídy PC) je v poněkud modifikovaném tvaru:

$$\boxed{\widetilde{p} = \mathrm{fl}_{\mathrm{IEEE}}(p) = s\ \bar{e}_{10}e_9e_8\ldots e_0 d_2, d_3\ldots d_{53}}\,,$$

kde

- $s 1 d_2, d_3 \ldots d_{53}$ … mantisa se znaménkovým bitem s (1=minus, 0=plus) a dvěma binárními místy před řádovou čárkou ($d_1 = 1$ a d_2),
- $e = (e_{10}e_9e_8\ldots e_0)_2$ … exponent v 11-bitové binární reprezentaci se znaménkem podle (1)(*ii*) v symetrickém rozsahu $-(2^{10} - 1) \le e \le 2^{10} - 1$. Tedy $\bar{e}_{10} = 1$ odpovídá nezáporným hodnotám exponentu a $\bar{e}_{10} = 0$ záporným hodnotám, přičemž případ $e = -2^{10}$ byl vyloučen, neboť pro něj jsou všechny bity exponentu $\bar{e}_{10}e_9e_8\ldots e_0$ nulové a vznikla by kolize s vyjádřením čísla 0, která je dána nulovými hodnotami všech bitů IEEE reprezentace (jinak totiž tyto nulové bity by určovaly kladné číslo $(10,0\ldots0)_2 \times 2^{-2^{10}}$).

ŠÍŘENÍ CHYB PŘI VÝPOČTU A ZTRÁTA PŘESNOSTI

Zdroje chyb:

- chyby vstupních dat (nepřesná měření),
- strojové chyby čísel v pohyblivé řádové čárce.

Budeme vyšetřovat šíření chyb při provádění aritmetických operací.

Příklad 1.32. Nechť $p = \widetilde{p} + E_p$ a $q = \widetilde{q} + E_q$, pak dostáváme pro

a) součet a rozdíl

$$\boxed{E_{p\pm q} = E_p \pm E_q} \quad \text{a} \quad \boxed{R_{p\pm q} = \tfrac{E_p \pm E_q}{p \pm q}}\,.$$

Důkaz. $E_{p\pm q} = (p \pm q) - (\widetilde{p} \pm \widetilde{q}) = p - \widetilde{p} \pm (q - \widetilde{q}) = E_p \pm E_q$. □

b) násobení

$$\boxed{E_{pq} = \widetilde{p}E_q + \widetilde{q}E_p + E_pE_q \approx \widetilde{p}E_q + \widetilde{q}E_p} \quad \text{a} \quad \boxed{R_{pq} = \tfrac{E_{pq}}{pq} \approx R_p + R_q}\,.$$

Důkaz. $E_{pq} = pq - \widetilde{p}\widetilde{q} = (\widetilde{p}+E_p)(\widetilde{q}+E_q) - \widetilde{p}\widetilde{q} = \widetilde{p}E_q + \widetilde{q}E_p + E_pE_q$, kde $E_pE_q \approx 0$. Pak $R_{pq} \approx \frac{\widetilde{p}E_q+\widetilde{q}E_p}{pq} = \frac{\widetilde{p}}{p}\frac{E_q}{q} + \frac{\widetilde{q}}{q}\frac{E_p}{p} \approx R_p + R_q$, jestliže položíme $\frac{\widetilde{p}}{p} \approx 1$ a $\frac{\widetilde{q}}{q} \approx 1$. □

Důsledek 1.33. $|E_p| \le \varepsilon_p,\ |E_q| \le \varepsilon_q \Rightarrow$

$$\begin{array}{lrcl} a) & |E_{p\pm q}| & \le & |E_p| + |E_q| \le \varepsilon_p + \varepsilon_q. \\ b) & |E_{pq}| & \approx & |\widetilde{p}E_q + \widetilde{q}E_p| \le |\widetilde{p}|\varepsilon_q + |\widetilde{q}|\varepsilon_p. \end{array}$$

Pro odhad absolutní a relativní chyby obecné operace dané funkčním předpisem lze užít následující obecnou větu.

Věta 1.34. *Nechť $q = f(p_1, p_2, \dots, p_n) =: f(\boldsymbol{p})$ a $p_i = \widetilde{p}_i + E_i$ pro $i = 1, 2, \dots, n$. Položme $\widetilde{\boldsymbol{p}} = [\widetilde{p}_1, \widetilde{p}_2, \dots, \widetilde{p}_n]$ a $\boldsymbol{p} = [p_1, p_2, \dots, p_n]$. Má-li f spojité všechny parciální derivace až do řádu 2 v nějakém okolí $O(\widetilde{\boldsymbol{p}})$, $\boldsymbol{p} \in O(\widetilde{\boldsymbol{p}})$, potom*

$$\boxed{E_q = E_{f(\boldsymbol{p})} \approx \sum_{i=1}^n \frac{\partial f(\widetilde{\boldsymbol{p}})}{\partial p_i} E_i} \quad a \quad \boxed{R_q = R_{f(\boldsymbol{p})} \approx \sum_{i=1}^n \frac{\partial \ln|f(\widetilde{\boldsymbol{p}})|}{\partial p_i} E_i},$$

pokud $|f| > 0$ v $O(\widetilde{\boldsymbol{p}})$.

Důkaz. Užijeme Taylorovu aproximaci z 1.16 v okolí $\widetilde{p}$:

$$E_{f(\widetilde{\boldsymbol{p}})} = f(\boldsymbol{p}) - f(\widetilde{\boldsymbol{p}}) \approx \sum_{i=1}^n \frac{\partial f(\widetilde{\boldsymbol{p}})}{\partial p_i}(p_i - \widetilde{p}_i) = \sum_{i=1}^n \frac{\partial f(\widetilde{\boldsymbol{p}})}{\partial p_i} E_i.$$

Přitom zanedbání vyšších členů Taylorova rozvoje je oprávněné, neboť v nich vystupují součiny typu $E_iE_j \dots$.

Použijeme-li místo f funkce $g(p_1, p_2, \dots, p_n) := \ln|f(p_1, p_2, \dots, p_n)|$, pak

$$\frac{\partial g(\widetilde{\boldsymbol{p}})}{\partial p_i} = \frac{\frac{\partial f(\widetilde{\boldsymbol{p}})}{\partial p_i}}{f(\widetilde{\boldsymbol{p}})} \Rightarrow g(\boldsymbol{p}) - g(\widetilde{\boldsymbol{p}}) \approx \sum_{i=1}^n \frac{\partial f(\widetilde{\boldsymbol{p}})}{\partial p_i}\frac{E_i}{f(\widetilde{\boldsymbol{p}})} \approx \frac{E_{f(\widetilde{\boldsymbol{p}})}}{f(\widetilde{\boldsymbol{p}})} = \widetilde{R}_{f(\boldsymbol{p})} \approx R_{f(\boldsymbol{p})}.$$

□

Důsledek 1.35.
$|E_i| \le \varepsilon_i$ *pro* $i = 1, 2, \dots, n \ \Rightarrow \boxed{E_q = E_{f(\boldsymbol{p})} \lessapprox \sum_{i=1}^n \frac{\partial f(\widetilde{\boldsymbol{p}})}{\partial p_i}\varepsilon_i}$.

Poznámka 1.36. Vztahy z příkladu 1.32 dostaneme ihned z věty 1.34, jestliže položíme $f(p,q) = p+q$, resp. $f(p,q) = pq$. Podobně odvodíme ještě vztahy pro podíl, jestliže $f(p,q) = \frac{p}{q}$:

$$\begin{aligned} E_{\frac{p}{q}} &\approx \frac{1}{\widetilde{q}}E_p - \frac{\widetilde{p}}{\widetilde{q}^2}E_q = \frac{\widetilde{q}E_p - \widetilde{p}E_q}{\widetilde{q}^2} \Rightarrow |E_{\frac{p}{q}}| \lessapprox \frac{|\widetilde{p}|\varepsilon_q + |\widetilde{q}|\varepsilon_p}{\widetilde{q}^2} = \frac{\varepsilon_{pq}}{\widetilde{q}^2}\ , \\ R_{\frac{p}{q}} &\approx \widetilde{R}_{\frac{p}{q}} \approx \frac{\widetilde{q}E_p - \widetilde{p}E_q}{\widetilde{q}^2}\,\frac{\widetilde{q}}{\widetilde{p}} = \frac{E_p}{\widetilde{p}} - \frac{E_q}{\widetilde{q}} = \widetilde{R}_p - \widetilde{R}_q \approx R_p - R_q\ . \end{aligned}$$

Poznámka 1.37 (Ztráta přesnosti při odečítání blízkých čísel).
Podle 1.32a) je $R_{p-q} \le \frac{|E_p|+|E_q|}{|p-q|}$, kde $p - q \to 0 \Rightarrow |R_{p-q}| \to \infty$.

Situaci demonstruje následující příklad:
$p = 3,1415926536,\ q = 3,1415957341$ mají 11 významných cifer, tj. podle 1.22 platí $|R_p|, |R_q| < 5 \times 10^{-12}$. Avšak $p - q = -0,0000030805$ má jen 4 významné cifry, tj. $|R_{p-q}| < 5 \times 10^{-5}$.
Vskutku:
$|E_p| = p|R_p|$ a $|E_q| = q|R_q| \Rightarrow |E_p + E_q| \le |E_p| + |E_q| < \max(p,q) \times 2 \times 5 \times 10^{-12} = q \times 10^{-11} \Rightarrow |R_{p-q}| = \frac{|E_p+E_q|}{|p-q|} < \frac{q\times 10^{-11}}{3,0805\times 10^{-6}} < \frac{3,15\times 10^{-11}}{3\times 10^{-6}} = 1,05 \times 10^{-5} < 5 \times 10^{-5}$.

Příklad 1.38.
Tento příklad ilustruje jednostrannou kumulaci chyb při opakovaném sečítání na počítači pracujícím s přesností na 9 dekadických cifer (absolutní chyby se sečítají podle 1.32a)):

$$\sum_{k=1}^{100000} 0{,}1 = 9999{,}9947.$$

Důvodem je skutečnost, že číslo 0,1 má podle 1.31(b) nekonečně mnoho binárních cifer a jeho strojová reprezentace je tedy pouze přibližná.

2. Řešení nelineárních rovnic

(Algoritmy v MATLABu: `vesely('nummet/iterace')`, `jmathews('chap_2')`)

Formulace problému:
Je dána rovnice

$$f(x) = 0, \text{ resp. } g(x) = x, \ x \in [a, b]. \tag{2.1}$$

Nalezněte (alespoň jedno) řešení, tj. $p \in [a, b]$ takové, že $f(p) = 0$, resp. $g(p) = p$. Řešení p rovnice (2.1) nazýváme **kořenem** funkce f, resp. **pevným bodem** funkce g.
Poznámka. $f(x) = 0 \ \Leftrightarrow \ g(x) = x$, kde $g(x) = x + f(x)$ nebo $g(x) = x - f(x)$.

Postup řešení:

a) Separace kořenů: Nalezneme dělení $a = a_0 < a_1 < \cdots < a_K = b$ tak, že v každém subintervalu $(a_k, a_{k+1}),\ k = 0, 1, \ldots, K-1$ leží právě jedno řešení rovnice (2.1).
Postup: zvolíme dostatečně hustou síť $x_n = a + n\Delta x,\ n = 0, 1, \ldots, N,\ \Delta x = \frac{b-a}{N}$ a vykreslíme graf funkce $f(x)$, resp. $g(x) - x$ na intervalu $[a, b]$. Například v MATLABu stačí provést:
`dx=(b-a)/N; x=a:dx:b; plot(x,f(x));`
Z grafu odhadneme přibližně pozice dělících bodů a_k, které separují průsečíky grafu s osou x. Musíme ovšem dávat pozor, aby Δx bylo dostatečně malé. Jinak by se mohlo stát, že mineme kořeny ležící ve vzdálenosti menší než Δx.
Místo hrubé vizuální lokalizace kořenů lze postupovat i exaktněji. Například knihovna `jmathews` nabízí funkci `approot.m` (+ demo `a2_4.m`), která vrací přibližné pozice kořenů $r_1, \ldots, r_M$. Je zde použit následující jednoduchý algoritmus pro nalezení r_j:
$r_j = \frac{1}{2}(x_{n+1} + x_n)$ pro některé $n \geq 1$, je-li splněna některá z těchto podmínek ($y_n := f(x_n)$):
(i) $y_{n-1}y_n < 0$ (funkce mění znaménko) nebo
(ii) $|y_n| < \varepsilon$ a $(y_n - y_{n-1})(y_{n+1} - y_n) < 0$ (funkční hodnota v x_n je malá a tečna mění znaménko — snaha zachytit dvojnásobný kořen, který je bodem dotyku grafu s osou x).
Po provedené separaci pak v každém subintervalu $[a_k, a_{k+1}]$ zpřesňujeme pozici (jediného) kořene vhodnou iterační metodou (viz b)).
Nadále tedy můžeme předpokládat, že $p \in [a, b]$ je jediné řešení rovnice (2.1).

b) Nalezení řešení $p \in [a, b]$**:** Zkonstruujeme vhodnou konvergentní posloupnost $\{p_n\}_{n=0}^{\infty}$, $p_n \to p$, kde p_0 je počáteční odhad ad a). Pro dostatečně velké N pak můžeme p_N považovat za dostatečně přesnou aproximaci hledaného kořene.

Definice 2.1. Řekneme, že kořen p rovnice $f(x) = 0$ má **násobnost** M ($M \in \mathbb{N}$), jestliže existuje funkce $h(x)$ taková, že $f(x) = (x - p)^M h(x)$ a $0 \neq |h(p)| < +\infty$.

Lemma 2.2. *$f(x) \in C^M[a, b],\ f(p) = f'(p) = \cdots = f^{(M-1)}(p) = 0,\ f^{(M)}(p) \neq 0 \ \Rightarrow$ p je kořenem $f(x)$ o násobnosti M.*

Kritéria pro ukončení iteračního procesu

Nechť $|E_n| \leq \varepsilon_n$ a $|R_n| \leq \rho_n$ jsou horní odhady po řadě absolutní a relativní chyby.

(KAE) Test, zda absolutní chyba je menší než zadaná přesnost $\delta > 0$
(a) $\varepsilon_n < \delta$, pokud známe ε_n,
jinak
(b) $|E_n| \approx |p_n - p_{n-1}| < \delta$
(nemusí být spolehlivé, pokud $f(p_{n-1})f(p_n) > 0$, kdy kořen neleží v $\mathcal{I}[p_{n-1}, p_n]$)

(KRE) Test, zda relativní chyba je menší než zadaná přesnost $\delta > 0$
(a) $\rho_n < \delta$, pokud známe ρ_n,
jinak
(b) $|R_n| \approx \frac{2|p_n - p_{n-1}|}{|p_n| + |p_{n-1}|} < \delta$, kde $\frac{|p_n| + |p_{n-1}|}{2} \approx |p|$
(nemusí být spolehlivé, pokud $f(p_{n-1})f(p_n) > 0$, kdy kořen neleží v $\mathcal{I}[p_{n-1}, p_n]$ nebo $p_n \approx 0$)

(KY) Test, zda funkční hodnota je menší než zadaná tolerance $\varepsilon > 0$: $|f(p_n)| < \varepsilon$

(KD) Detekce dvojnásobného kořene
$|f(p_{n-1})|, |f(p_n)|, |f(p_{n+1})| < \varepsilon$ (malé funkční hodnoty v okolí p_n)
a současně
$(f(p_n) - f(p_{n-1}))(f(p_{n+1}) - f(p_n)) < 0$ (tečna mění znaménko).
Lokalizace kořenů s vyšší násobností než 2 je problematická.

Výše uvedená kritéria lze vhodně kombinovat užitím logických spojek disjunkce |, případně konjunkce &, například (KAE) | (KY), případně (KAE) & (KY).

2.1. Řád konvergence. Aitkenova metoda zrychlení konvergence.

Definice 2.3 (Řád konvergence)**.** Nechť $p_n \to p$ pro $n = 0, 1, 2, \ldots$, $e_n := p - p_n$. Jestliže existují $A > 0, R > 0$ tak, že

$$\lim_{n\to\infty} \frac{|p - p_{n+1}|}{|p - p_n|^R} = \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^R} = A \quad \text{(tzv. asymptotická chybová konstanta)}, \tag{2.2}$$

pak říkáme, že $\{p_n\}_{n=0}^{\infty}$ **konverguje k** p **s řádem konvergence** R.
Speciální případy: **lineární konvergence** $(R = 1)$, **kvadratická konvergence** $(R = 2)$.

Poznámka. Pro velké n (2.2) $\Rightarrow |e_{n+1}| \approx A|e_n|^R$, tj. se zvětšujícím se R roste rychlost konvergence $e_n \to 0$. Například pro $R = 2$ a $|e_n| \approx 10^{-2}$ můžeme očekávat $|e_{n+1}| \approx 10^{-4}$. Pro některé posloupnosti přitom R nemusí být celé číslo.

Definice 2.4. Pro $\{p_n\}_{n=0}^{\infty}$ nazýváme $\{\Delta^k p_n\}_{n=0}^{\infty}$ posloupností jejích k**-tých diferencí** $(k \in \mathbb{N})$, jestliže

$$\Delta p_n := p_{n+1} - p_n,$$
$$\Delta^2 p_n := \Delta(\Delta p_n) = \underbrace{p_{n+2} - p_{n+1}}_{\Delta p_{n+1}} - \underbrace{(p_{n+1} - p_n)}_{\Delta p_n} = p_{n+2} - 2p_{n+1} + p_n,$$
$$\vdots$$
$$\Delta^k p_n := \Delta(\Delta^{k-1} p_n) \quad .$$

Věta 2.5 (Zrychlení konvergence Aitkenovou extrapolací)**.**
Nechť $\{p_n\}_{n=0}^{\infty}$, $p_n \to p$. *Jestliže platí*

(1) $p - p_n \neq 0$ *pro dosti velká* n,
(2) $\exists\ A,\ |A| < 1$ *tak, že* $\lim_{n\to\infty} \frac{p-p_{n+1}}{p-p_n} = A$ *(alespoň lineární konvergence, neboť může být* $A = 0$*),*

potom posloupnost $\{q_n\}_{n=0}^{\infty}$, *kde*

$$q_n := p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n} = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n} = \frac{p_n p_{n+2} - p_{n+1}^2}{p_{n+2} - 2p_{n+1} + p_n} \tag{2.3a}$$

konverguje k p *rychleji než* $\{p_n\}_{n=0}^{\infty}$ *v tom smyslu, že* $\lim_{n\to\infty} |\frac{p-q_n}{p-p_n}| = 0$, *přičemž* $\Delta^2 p_n \neq 0$ *pro dosti velká* n.

Poznámka 2.6 (Přibližné odvození vztahu (2.3a)).
Dle (2) platí pro velká n přibližně:

$$\frac{p - p_{n+1}}{p - p_n} \approx A \approx \frac{p - p_{n+2}}{p - p_{n+1}} \Rightarrow (p - p_{n+1})^2 \approx (p - p_n)(p - p_{n+2}) \Rightarrow$$
$$p^2 - 2pp_{n+1} + p_{n+1}^2 \approx p^2 - pp_{n+2} - pp_n + p_n p_{n+2}.$$

Odtud po úpravě obdržíme

$$p \approx \frac{p_n p_{n+2} - p_{n+1}^2}{p_{n+2} - 2p_{n+1} + p_n},$$

což odpovídá (2.3a).
Aitkenova extrapolace se někdy nazývá **Aitkenův** Δ^2**-proces**.
Většinou alternativně používáme zrychlenou posloupnost $\{p'_n\}_{n=0}^{\infty}$ $(n \geq 2)$:

$$p'_n := q_{n-2} = \frac{p_{n-2}p_n - p_{n-1}^2}{p_n - 2p_{n-1} + p_{n-2}} = p_n - \frac{(p_n - p_{n-1})^2}{p_n - 2p_{n-1} + p_{n-2}} = p_n - \frac{(\Delta p_{n-1})^2}{\Delta^2 p_{n-2}}. \tag{2.3b}$$

Její formální výhodou je, že pro výpočet p'_n používá pouze hodnoty p_j pro $j \leq n$ a nikoliv budoucí hodnoty jako v (2.3a).

Příklady (MATLAB)**.**
`zktest([],'itaitken')`

Algoritmus (MATLAB)**.**
`vesely('nummet/iterace'): aitken.m`

2.2. **Metody dělení intervalu.**
Předpoklady: $f(x) \in C[a,b],\ f(a)f(b) < 0,\ f(p) = 0.$

Definice 2.7.
Nechť $\varphi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ je libovolná funkce dvou proměnných taková, že $x_1 < \varphi(x_1, x_2) < x_2$ platí pro každé $x_1, x_2 \in \mathbb{R}$, $x_1 < x_2$. Konstruujeme konečnou nebo nekonečnou posloupnost intervalů $\{[a_n, b_n]\}_n$ tak, že $[a_0, b_0] \supset [a_1, b_1] \supset \cdots \supset [a_n, b_n] \supset \ldots$, $p \in (a_n, b_n)$ pro $n = 0, 1, \ldots$, takto:

(1) $a_0 := a,\ b_0 := b$
(2) Jestliže $[a_n, b_n]$ je již zkonstruováno, položíme $p_n := \varphi(a_n, b_n)$. Mohou nastat tři případy:
   (α) je-li $f(p_n) = 0$, pak $p = p_n$ je kořen a konstrukce končí;
   jinak položíme:
   (β) $a_{n+1} := a_n,\ b_{n+1} := p_n$, pokud $f(a_n)f(p_n) < 0$ ($\Leftrightarrow\ f(p_n)f(b_n) > 0$)
   (γ) $a_{n+1} := p_n,\ b_{n+1} := b_n$, pokud $f(a_n)f(p_n) > 0$ ($\Leftrightarrow\ f(p_n)f(b_n) < 0$)

Výše popsaná konstrukce posloupnosti $\{p\}_n$ se nazývá **metodou dělení intervalu s dělící funkcí** φ.

Věta 2.8. *Jestliže $b_n - a_n \to 0$, pak $|p_n - p| \le b_{n+1} - a_{n+1}$ a tedy $p_n \to p$.*

Speciální případy

(1) Metoda půlení intervalu (bisekce)

$\varphi(x_1, x_2) := \frac{x_1+x_2}{2}$, tj. $p_n = \frac{a_n+b_n}{2}$.

Příklady (MATLAB)**.**
`zktest([],'itbisek')`

Algoritmus (MATLAB)**.**
`vesely('nummet/iterace'): itbisek.m`
`jmathews('chap_2'): bisect.m` *(+ demo* `a2_2.m`*)*

(2) Metoda regula-falsi

p_n určíme jako průsečík spojnice bodů $[a_n, f(a_n)]$ a $[b_n, f(b_n)]$ s osou x:

$\frac{f(b_n)-f(a_n)}{b_n-a_n} = \frac{f(b_n)-0}{b_n-p_n} \Rightarrow p_n = b_n - \frac{f(b_n)(b_n-a_n)}{f(b_n)-f(a_n)}$.

Tedy dělící funkce je v tomto případě definována předpisem:
$\varphi(x_1, x_2) := x_2 - \frac{f(x_2)(x_2-x_1)}{f(x_2)-f(x_1)}$.

Příklady (MATLAB)**.**
`zktest([],'itregfal')`

Algoritmus (MATLAB)**.**
`vesely('nummet/iterace'): regfalsi.m, itregfal.m`
`jmathews('chap_2'): regula.m` *(+ demo* `a2_3.m`*)*

(3) Metoda zlatého řezu

$y := \varphi(x_1, x_2)$ je určeno poměrem $\frac{y-x_1}{x_2-y} = \frac{x_2-x_1}{y-x_1}$, což je ekvivalentní s řešením kvadratické rovnice $(y - x_1)^2 - (x_2 - y)(x_2 - x_1) = 0$ vzhledem k y.

(4) Metoda Monte Carlo (náhodné střílení)

$\varphi(x_1, x_2)$ je náhodné číslo mezi x_1 a x_2.

Algoritmus (MATLAB)**.**
`x1 + (x2-x1)*rand(1)`

Věta 2.9 (Konvergence metody půlení intervalu)**.** *Nechť $f \in C[a,b],\ f(a)f(b) < 0$ a $p \in (a,b)$ je jediný kořen $f(p) = 0$. Je-li $\{p_n\}_n$ posloupnost metody půlení intervalu, pak $|p - p_n| \le \frac{b-a}{2^{n+1}}$ pro $n = 0, 1, \ldots$ a tedy $\lim_{n\to\infty} p_n = p$.*

Důsledek 2.10. *$N = \mathrm{int}(\frac{\ln(b-a)-\ln\delta}{\ln 2})$ je počet iterací metody půlení intervalu, po jehož dosažení je chyba aproximace kořene p menší než předepsaná tolerance $\delta > 0$, tj. $|p - p_N| < \delta$.*

Důsledek 2.11. $\lim_{n\to\infty} |\frac{p-p_{n+1}}{p-p_n}| \approx \lim_{n\to\infty} \frac{b-a}{2^{n+2}} \frac{2^{n+1}}{b-a} = 0,5$ *(lineární konvergence).*

Poznámka. Metoda půlení intervalu konverguje pomalu — podle 2.11 se po každé iteraci chyba redukuje přibližně na jednu polovinu, neboli přesnost se zvýší o jedno binární místo. Protože $10^{-1} \approx (\frac{1}{2})^{3,3}$ je ke zvýšení přesnosti o jedno desetinné místo potřeba nejméně tří iterací.

Věta 2.12 (Konvergence metody regula falsi)**.** *Nechť $f \in C[a,b]$, $f(a)f(b) < 0$ a $p \in (a,b)$ je jediný kořen $f(p) = 0$. Pak posloupnost iterací $\{p_n\}_n$ metody regula falsi konverguje k p lineárně.*

Věta 2.13. *Je-li f konvexní (resp. konkávní) na intervalu $[a_n, b_n]$ a $x_n \in \{a_n, b_n\}$ je ten z krajních bodů, pro nějž $f(x_n) > 0$ (resp. $f(x_n) < 0$), pak x_n zůstává v dalších iteracích metody regula falsi pevný, tj. $x_n = a_m$ nebo $x_n = b_m$ pro každé $m \geq n$.*

Poznámka 2.14. Dá se ukázat, že v konkávním, resp. konvexním případě je $|p - p_{n+1}| = A_n|p - p_n|$, kde $A_n < \frac{1}{2}$ pro velká n. Konvergence je lineární jako u metody půlení intervalu, avšak s asymptotickou chybovou konstantou $\lim_{n\to\infty} A_n = A < \frac{1}{2}$. Lze tedy očekávat, že metoda regula falsi bude mírně rychlejší než metoda půlení intervalu. Je rovněž vidět, že na rozdíl od metody půlení intervalu $b_m - a_m \nrightarrow 0$, neboť $b_m - a_m \geq p - a_n$ nebo $b_m - a_m \geq b_n - p$ pro dostatečně velká $m \geq n$.

2.3. **Metoda pevného bodu.**

Rovnice: $x = g(x), x \in [a,b]$.

Definice 2.15. Bod $p \in [a,b]$, pro nějž $p = g(p)$ nazýváme **pevným bodem funkce** $g(x)$.

Definice 2.16 (Metoda pevného bodu)**.**
Nechť $p_0 \in \mathbb{R}$ libovolné a $p_{n+1} = g(p_n)$ pro $n = 0, 1, \ldots$. Pak posloupnost $\{p_n\}_{n=0}^{\infty}$ nazýváme **posloupností iterací metody pevného bodu (PB-iterací) funkce** g. Bod p_0 nazýváme **startovacím bodem této posloupnosti**.

Věta 2.17. *Nechť $g(x)$ je spojitá na $\mathbb{R}$ a $\{p_n\}_{n=0}^{\infty}$ je posloupností jejích PB-iterací, která je konvergentní, $p = \lim_{n\to\infty} p_n$. Pak p je pevným bodem funkce g.*

Věta 2.18. *Nechť $g \in C[a,b]$. Potom*

(1) *$g([a,b]) \subseteq [a,b] \Rightarrow g$ má pevný bod v $[a,b]$.*
(2) *Jestliže navíc g splňuje Lipschitzovu podmínku s konstantou $0 \leq K < 1$, tj. $|g(x) - g(x')| \leq K|x - x'|\ \forall\ x, x' \in [a,b]$ (nebo existuje g' na (a,b), $|g'(x)| \leq K < 1\ \forall\ x \in (a,b)$), pak g má jediný pevný bod v $[a,b]$.*

Věta 2.19 (Věta o pevném bodu)**.** *Nechť p je pevný bod funkce $g(x)$ a $\mathcal{I} = [p - \delta, p + \delta] \subseteq [a,b]$, jeho δ-okolí ($\delta > 0$). Nechť $g(x)$ je na $\mathcal{I}$ definována (resp. zde navíc má derivaci $g'(x)$) a $\{p_n\}_{n=0}^{\infty}$ je její posloupnost iterací se startovacím bodem $p_0 \in \mathcal{I}$. Pak*

(1) *$|g(x) - g(p)| \leq K|x - p|$, $0 \leq K < 1$ pro $x \in \mathcal{I}$ (resp. $|g'(x)| \leq K < 1$, pro $x \in \mathcal{I}$) $\Rightarrow p_n \to p$ a $p_n \in \mathcal{I}$ pro každé $n = 0, 1, \ldots$.*
Takový pevný bod p nazýváme **přitahujícím pevným bodem**.
(2) *$|g(x) - g(p)| \geq |x - p|$ pro $x \in \mathcal{I}$ (resp. $|g'(x)| \geq 1$ pro $x \in \mathcal{I}$) $\Rightarrow p_n \nrightarrow p$, pokud $p_0 \neq p$ a současně ($g(x) = p \Rightarrow x = p$ na $[a,b]$).*
Jestliže $|g(x) - g(p)| \geq K|x - p|$, $K > 1$, pro $x \in \mathcal{I}$ (resp. $|g'(x)| \geq K > 1$ pro $x \in \mathcal{I}$), pak navíc $p_n \in \mathcal{I} \Rightarrow \exists\ m > n : p_m \notin \mathcal{I}$.
Takový pevný bod p nazýváme **odpuzujícím pevným bodem**.

Důsledek 2.20. *Za předpokladů věty 2.19(1) platí následující odhady chyby aproximace:*

(1) *$|p - p_n| \leq K^n|p - p_0|$ pro $n \geq 1$.*
(2) *$|p - p_n| \leq \frac{K^n|p_1 - p_0|}{1-K}$ pro $n \geq 1$.*
(3) *$|p - p_n| \leq \frac{K}{1-K}|p_n - p_{n-1}|$ pro $n \geq 1$.*
Jestliže $|g'(x)| \leq K < 1$ na $\mathcal{I}$, pak
$|p - p_n| \leq \frac{|g'(\xi_{n-1})|}{1-|g'(\xi_{n-1})|}|p_n - p_{n-1}| \leq \frac{K}{1-K}|p_n - p_{n-1}|$ pro vhodné $\xi_{n-1} \in \mathcal{I}[p_{n-1}, p]$.
(4) *$K \leq \frac{1}{2} \Rightarrow |p - p_n| \leq |p_n - p_{n-1}|$.*

Poznámka 2.21. Pokud $g'(x) \in C(\mathcal{I})$, pak tvrzení (1) a (2) věty 2.19 lze vzhledem ke spojitosti $g'(x)$ nahradit po řadě jednoduššími kritérii:

(1') $|g'(p)| < 1$,
(2') $|g'(p)| > 1$.

Totiž z nich pak plynou původní (1) a (2) takto:
Pro vhodné $\varepsilon > 0$ lze kolem p nalézt dostatečně malé okolí $\mathcal{O}(p) \subseteq \mathcal{I}$, v němž platí
$|g'(x)| \leq |g'(p)| + \varepsilon < 1$ v případě (1), resp. $|g'(x)| \geq |g'(p)| - \varepsilon > 1$ v případě (2).

Příklad 2.22.

(1) Řešte $x^2 = c$, $c > 0$
(i) $x^2 = c \Leftrightarrow x = \frac{c}{x} \Rightarrow g(x) = \frac{c}{x}$, $p_1 = \frac{c}{p_0}$, $p_2 = \frac{c}{c/p_0} = p_0, \ldots$ a pro $p_0 \neq \sqrt{c}$ dostáváme alternující **divergentní** posloupnost. Totiž $|g'(x)| = |-\frac{c}{x^2}| \Rightarrow |g'(\sqrt{c})| = 1$, $|g'(x)| < 1$ pro $x > \sqrt{c}$ a $|g'(x)| > 1$ pro $x < \sqrt{c}$.

(ii) $x^2 = c \Leftrightarrow 2x^2 = x^2+c \Leftrightarrow x = \frac{1}{2}\big(x+\frac{c}{x}\big) \Rightarrow g(x) = \frac{1}{2}\big(x+\frac{c}{x}\big)$, $g'(x) = \frac{1}{2}\big(1-\frac{c}{x^2}\big)$, $g'(\sqrt{c}) = 0$. Podle 2.19(1) $p_n \to \sqrt{c}$ **konverguje velmi rychle** s uvážením 2.20(1), neboť konstanta $K = 0$.

Vidíme, že volbou $g(x)$ můžeme výrazně ovlivnit konvergenční vlastnosti příslušné posloupnosti PB-iterací.

(2) Rovnice $x = 2\sqrt{x-1}$ má jediný pevný bod $p = 2$ pro $x \geq 1$, přičemž $g'(x) = 1/\sqrt{x-1}$ a tedy $g'(2) = 1, g'(x) > 1$ pro $1 \leq x < 2$ a $g'(x) < 1$ pro $x > 2$. Není tedy splněna žádná z podmínek (1') a (2') z poznámky 2.21, abychom mohli jednoznačně rozhodnout o konvergenci či divergenci. Numerický experiment (viz `zktest(-1,'itpevbod')`) ukazuje divergenci zvolíme-li počáteční aproximaci $p_0 = 1,5 < 2$ v levém okolí a naopak velmi pomalou konvergenci pro $p_0 = 2.5 > 2$ z pravého okolí.

Z (1)(i) a z (2) můžeme tedy usuzovat, že v případě $g'(p) = 1$ posloupnost může i nemusí konvergovat.

Algoritmus (MATLAB)**.**
`vesely('nummet/iterace'): fixpoint.m`
`jmathews('chap_2'): fixpt.m` *(+ demo* `a2_1.m`*)*

2.4. **Newton-Raphsonova metoda.**

Předpoklady: $f(x) \in C^2[a,b]$, $f(p) = 0$, $f'(p) \neq 0$, $p \in [a,b]$.

Konstrukce posloupnosti iterací:

- p_0 ... počáteční aproximace
- Je-li p_n již zkonstruováno, pak p_{n+1} nalezneme jako průsečík tečny v bodě $(p_n, f(p_n))$ s osou x:

$$f'(p_n) = \frac{0 - f(p_n)}{p_{n+1} - p_n}.$$

Odtud dostáváme iterační schéma Newton-Raphsonovy metody:

$$p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)} \text{ pro } n = 0, 1, 2, \ldots \tag{2.4}$$

Poznámka 2.23. Iterační schéma (2.4) lze považovat za iterace pevného bodu funkce $g(x) = x - \frac{f(x)}{f'(x)}$, neboť pro $x \in [p-\delta, p+\delta]$ při dostatečně malém $\delta > 0$ je $f'(x) \neq 0$ a tedy
$x = x - \frac{f(x)}{f'(x)} \Leftrightarrow 0 = -\frac{f(x)}{f'(x)} \Leftrightarrow f(x) = 0.$

Spočtěme $g'(x)$:

$$g'(x) = \left(x - \frac{f(x)}{f'(x)}\right)' = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2} = \frac{f(x)}{f'(x)}\frac{f''(x)}{f'(x)}. \tag{2.5}$$

Protože $f(p) = 0$ a $f'(p) \neq 0$, je $g'(p) = 0$ a posloupnost (2.4) Newton-Raphsonových iterací tedy splňuje předpoklad 2.21(1'). Dokázali jsme tak následující tvrzení.

Věta 2.24 (Newton-Raphsonova věta)**.**
Nechť $f \in C^2[a,b]$ má kořen $p \in [a,b]$, tj. $f(p) = 0$. Jestliže $f'(p) \neq 0$, pak existuje $\delta > 0$ tak, že posloupnost (2.4) *Newton-Raphsonových iterací $\{p_n\}_{n=0}^{\infty}$ konverguje k p pro libovolné $p_0 \in [p-\delta, p+\delta]$, přičemž $p_n \in [p-\delta, p+\delta]$ pro každé $n = 0, 1, 2, \ldots$.*

Důsledek 2.25 (Newton-Raphsonovy iterace pro hledání druhé odmocniny)**.**
Nechť $c \in \mathbb{R}$, $c > 0$ a $p_0 > 0$ je libovolný počáteční odhad $\sqrt{c}$. Definujme posloupnost $\{p_n\}_{n=0}^{\infty}$ rekurzivně takto:

$$p_{n+1} = \frac{1}{2}\left(p_n + \frac{c}{p_n}\right) \text{ pro } n = 0, 1, 2, \ldots,$$

pak $p_n \to \sqrt{c}$ při $n \to \infty$ (viz též příklad 2.22(1)(ii)).

Věta 2.26 (Řád konvergence Newton-Raphsonovy metody)**.**
Nechť posloupnost Newton-Raphsonových iterací (2.4) *konverguje ke kořenu p funkce $f(x)$. Označíme-li $e_n = p - p_n$ pro $n = 0, 1, \ldots$, pak za předpokladů věty 2.24 tato posloupnost konverguje kvadraticky s asymptotickou chybovou konstantou*

$$A := \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^2} = \frac{|f''(p)|}{2|f'(p)|}.$$

Důkaz. Vezmeme prvé dva členy Taylorova rozvoje funkce $f(x)$ v bodě p_n pro $x = p$ (viz 1.13):

$$0 = f(p) = f(p_n) + f'(p_n)(p - p_n) + \frac{f''(\xi_n)}{2!}(p - p_n)^2,\ \xi_n \in \mathcal{I}(p, p_n). \tag{2.6}$$

Po vydělení rovnice (2.6) hodnotou $f'(p_n)$ a úpravě dostaneme po limitním přechodu $n \to +\infty$ požadovaný vztah. □

Věta 2.27 (Zrychlená konvergence Newton-Raphsonovy metody)**.**
Nechť posloupnost Newton-Raphsonových iterací (2.4) *konverguje lineárně ke kořenu p násobnosti $M > 1$ funkce $f(x)$. Potom modifikovaná posloupnost*

$$p_{n+1} = p_n - M\frac{f(p_n)}{f'(p_n)},\ n = 0, 1, \dots \tag{2.7}$$

konverguje kvadraticky k p.

Poznámka 2.28. Z poznámky 2.23 vidíme, že Newton-Raphsonova metoda (2.4) představuje optimální postup jak řešení rovnice $f(x) = 0$ převést na řešení rovnice $x = g(x)$ metodou pevného bodu, neboť $g'(p) = 0$ vede k rychlé konvergenci s nejmenší možnou konstantou $K = 0$ (viz 2.20(1)).

Dobře je toto demonstrováno na případu rovnice $x^2 = c$ z příkladu 2.22(1) , kdy nevhodný postup (i) nevede ke konvergentní posloupnosti. Naopak postup (ii) zdůvodněný ve 2.25 dává rychle konvergentní posloupnost. Například pro $c = 5$ (hledáme $\sqrt{5}$) a $p_0 = 2$ dostáváme:

$$p_1 = \frac{2 + 5/2}{2} = 2,25,\ p_2 = \frac{2,25 + 5/2,25}{2} = 2,23611111,$$
$$p_3 = \frac{2,23611111 + 5/2,23611111}{2} = 2,236067978, \dots, p_k = 2,236067978 \text{ pro } k \geq 4.$$

Tedy již po čtyřech iteracích dostáváme $\sqrt{5}$ s přesností na 9 desetinných míst.

Věta 2.29 (Odhad chyby Newton-Raphsonových iterací)**.**
Za předpokladů věty 2.24 nechť $|\frac{f''(x)}{f'(x)}| \leq L$ pro $x \in \mathcal{I}[p_{n-1}, p]$. Pak platí

(1) $L|p_n - p_{n-1}| \leq K\ < 1 \Rightarrow |p - p_n| \leq \frac{K}{1-K}|p_n - p_{n-1}|$,
(2) $L|p_n - p_{n-1}| \leq \frac{1}{2} \Rightarrow |p - p_n| \leq |p_n - p_{n-1}|$.

Uvedeme ještě dvě tvrzení, pomocí nichž lze snadno ověřit, zda počáteční aproximace p_0 byla zvolena dostatečně blízko p tak, aby byla zaručena konvergence podle 2.24.

Věta 2.30. *Nechť $\{p_n\}_{n=0}^{\infty}$ je posloupnost Newton-Raphsonových iterací* (2.4)*. Označme $I_0 := \mathcal{I}[p_0, p_0 + 2(p_1 - p_0)]$. Jestliže $2|p_1 - p_0|M \leq |f'(p_0)|$, kde $M = \max_{x \in I_0}|f''(x)|$, potom $p_n \in I_0$ pro $n = 0, 1, \dots$ a $\lim_{n\to\infty} p_n = p$, kde p je jediný kořen $f(x)$ na intervalu I_0.*

Věta 2.31. *Nechť $f'(x) \neq 0$, $f''(x)$ nemění znaménko na intervalu $[a, b]$ (tj. $f(x)$ je konvexní nebo konkávní na $[a, b]$) a $f(a)f(b) < 0$. Jestliže $|\frac{f(a)}{f'(a)}| < b - a$ a současně $|\frac{f(b)}{f'(b)}| < b - a$ (tj. iterace odstartovaná v krajním bodě intervalu $[a, b]$ nepadne mimo tento interval), potom posloupnost Newton-Raphsonových iterací* (2.4) *konverguje pro libovolnou počáteční aproximaci $p_0 \in [a, b]$ k některému kořenu $p \in [a, b]$ funkce $f(x)$.*

Příklady (MATLAB)**.**
`zktest([],'itnewton')`

Algoritmus (MATLAB)**.**
`vesely('nummet/iterace'): newton.m`
`jmathews('chap_2'): newton.m` *(+ demo* `a2_5.m`*)*

2.32 (Úskalí Newton-Raphsonovy metody)**.**
Věta 2.24 nedává konstruktivní návod k nalezení δ-okolí kořene p pro volbu počáteční aproximace p_0. Z jejího důkazu vyplývá pouze možnost odhadnout toto okolí z grafu funkce $g'(x) = \frac{f(x)f''(x)}{f'(x)^2}$ tak, aby $|g'(x)| \leq K\ < 1$. Jestliže neprověříme dostatečně předpoklady této věty, může se stát, že zkonstruovaná posloupnost $\{p_n\}_{n=0}^{\infty}$ bude vykazovat neočekávané chování. Typické jsou následující případy:

(1) $f(x) > 0$ nebo $f(x) < 0$ pro každé $x \in \mathbb{R}$.
Tedy $f(x)$ nemá reálný kořen a $\{p_n\}_{n=0}^{\infty}$ nekonverguje, například
$f(x) = x^2 - 4x + 5 = (x - 2)^2 + 1 > 0$.
(2) p_0 je příliš daleko od kořene p.

a) $\{p_n\}_{n=0}^{\infty}$ může konvergovat k jinému kořenu (mimo uvažovaný interval).
Například u funkce $f(x) = \cos(x)$ hledáme $p = \frac{\pi}{2}$ a zvolíme $p_0 = 3$. Potom $p_1 \approx -4,015,\ p_2 \approx -4,853,\ldots,p_n \to -\frac{3\pi}{2} = -4,71238898$
(viz též `zktest(-12,'itnewton')`).

b) $\{p_n\}_{n=0}^{\infty}$ je cyklicky divergentní.
V posloupnosti se objevují tzv. **cykly**, kdy po konečném počtu kroků se členy posloupnosti opakují buď přesně nebo přibližně. Například $f(x) = x^3 - x - 3$ při $p_0 = 0$ dává $p_1 = -3,\ p_2 = -1,961538,\ p_3 = -1,147176,\ p_4 = -0,006579 \approx 0$, takže uvízneme v přibližném cyklu řádu 4, tj. $p_{n+4} \approx p_n$ (viz též `zktest(10,'itnewton')`).

(3) Osa x je asymptotou funkce $f(x)$ pro $x \to +\infty$ nebo pro $x \to -\infty$.
V tomto případě může $p_n \to +\infty$ nebo $p_n \to -\infty$ při $f(p_n) \to 0$ a algoritmus se může zastavit s ohlášením falešného kořene. Jako příklad uvažme funkci $f(x) = xe^{-x}$ a $p_0 = 2$, kdy $f(x) \to 0$ rychle pro $x \to +\infty$, například $f(p_{15}) = 5,36 \times 10^{-8}$ (viz též `zktest(13,'itnewton')`). Používáme-li pro zastavení odhad relativní chyby KRE(b): $2|p_n - p_{n-1}|/(|p_n| + |p_{n-1}|)$, může tento vykazovat falešně malou hodnotu. Z těchto důvodů se někdy toto kritérium modifikuje tak, že v něm hodnotu $|p_{n-1}|$ nahradíme dostatečně malým číslem, například 10^{-6} a tím odhad relativní chyby u falešného kořene poněkud zvětšíme.

(4) $|g'(x)| \not< 1$ na intervalu obsahujícím kořen.
V tomto případě můžeme dostat tzv. **oscilatoricky divergentní** posloupnost $\{p_n\}_{n=0}^{\infty}$, kdy hodnoty p_n jsou střídavě vlevo a vpravo od kořene p a přitom se od něj vzdalují. Například v případě funkce $f(x) = \arctan(x)$ s kořenem $p = 0$ je $g(x) = x - (1 + x^2)\arctan(x)$ a $g'(x) = -2x\arctan(x)$. Pro $p_0 = 1,45$ je $|g'(p_0)| = 2.8044 > 1$ a dostáváme postupně $p_1 \approx -1,55,\ p_2 \approx 1,85,\ p_3 \approx -2,89$, atd.
(viz též `zktest(14,'itnewton')`).

2.5. **Steffensenova metoda.**

Konstrukce posloupnosti Steffensenových iterací $\{s_n\}_{n=0}^{\infty}$:
Posloupnost $\{s_n\}_{n=0}^{\infty}$ vznikne z posloupnosti iterací pevného bodu $\{p_n\}_{n=0}^{\infty}$ (definice 2.16), resp. Newton-Raphsonových iterací (2.4) jakožto speciálního případu (viz poznámku 2.23), jejím zrychlením Aitkenovou extrapolací (věta 2.5 a poznámka 2.6). Zpravidla se používá následující schéma využívající (2.3b):

$$\begin{array}{lll} & & s_0 := p'_0 := p_0 \\ p_1 = g(p'_0), & p_2 = g(p_1), & s_1 := p'_2 \\ p_3 = g(p'_2), & p_4 = g(p_3), & s_2 := p'_4 \\ p_5 = g(p'_4), & p_6 = g(p_5), & s_3 := p'_6 \\ & \vdots & \end{array} \tag{2.8}$$

$$p_{2n-1} = g(p'_{2n-2}),\ p_{2n} = g(p_{2n-1}),\ s_n := p'_{2n} \overset{(2.3b)}{=} p_{2n} - \frac{(p_{2n} - p_{2n-1})^2}{p_{2n} - 2p_{2n-1} + p'_{2n-2}},\ n = 1, 2, \ldots$$

Věta 2.33. *Nechť $\{p_n\}_{n=0}^{\infty}$ je posloupnost PB-iterací funkce $g(x)$ a $\{s_n\}_{n=0}^{\infty}$ příslušná posloupnost Steffensenových iterací* (2.8)*. Potom $\{s_n\}_{n=0}^{\infty}$ je posloupností PB-iterací funkce*

$$G(x) = x - \frac{(g(x) - x)^2}{g(g(x)) - 2g(x) + x} = \frac{x\,g(g(x)) - g^2(x)}{g(g(x)) - 2g(x) + x}. \tag{2.9}$$

Důsledek 2.34 (Steffensenovy iterace pro rovnici $f(x) = 0$)**.**
Buď dána rovnice $f(x) = 0$. Nechť $\{s_n^+\}_{n=0}^{\infty}$ a $\{s_n^-\}_{n=0}^{\infty}$ jsou posloupnosti Steffensenových iterací příslušné PB-iteracím po řadě funkcí $g^+(x) := x + f(x)$ a $g^-(x) := x - f(x)$. Pak pro $n = 0, 1, \ldots$ platí

$$s_{n+1}^+ = s_n^+ - \frac{f^2(s_n^+)}{f(s_n^+ + f(s_n^+)) - f(s_n^+)} \tag{2.10a}$$

$$s_{n+1}^- = s_n^- - \frac{f^2(s_n^-)}{f(s_n^-) - f(s_n^- - f(s_n^-))} \tag{2.10b}$$

Věta 2.35 (Řád konvergence Steffensenovy metody)**.**
Pro posloupnost Steffensenových iterací $\{p_n\}_{n=0}^{\infty}$, kde $p_n := s_n^+$ nebo $p_n := s_n^-$, platí tvrzení věty 2.24

za týchž předpokladů o funkci $f(x)$, přičemž konvergence je opět kvadratická s asymptotickou chybovou konstantou A poněkud větší než ve větě 2.26:

$$A := \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^2} = \frac{|f''(p)|}{2|f'(p)|}|1 + f'(p)|, \text{ kde } e_n := p - p_n.$$

Příklady (MATLAB)**.**
`zktest([],'itsteff')`
Algoritmus (MATLAB)**.**
`vesely('nummet/iterace'): steffplu.m, steffmin.m`
`jmathews('chap_2'): steff.m` *(+ demo* `a2_7.m`*)*

2.6. **Kvazinewtonovy metody.**

Obecný princip konstrukce posloupnosti iterací:
Nechť $f(x)$ je opět definována na intervalu $[a, b]$. V Newtonově metodě $p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)}$ aproximujeme $f'(p_n)$ (směrnice tečny v bodě $(p_n, f(p_n))$) směrnicí přímky procházející body $(p_n, f(p_n))$ a $(\widetilde{p}_n, f(\widetilde{p}_n))$, kde $\widetilde{p}_n$ zvolíme vhodně v okolí p_n, tj. $f'(p_n) \approx \frac{f(p_n)-f(\widetilde{p}_n)}{p_n-\widetilde{p}_n}$.

Iterace kvazinewtonových metod mají tedy obecně tvar:

$$p_{n+1} = p_n - f(p_n)\frac{p_n - \widetilde{p}_n}{f(p_n) - f(\widetilde{p}_n)} \text{ pro } n = 0, 1, \dots \ . \tag{2.11}$$

2.36 (Speciální případy)**.**

(A) Metoda regula-falsi (viz odst. 2.2)

$p_0 := a,\ p_1 := b$ (počáteční iterace)

$\widetilde{p}_n := p_s,$ pro $n = 1, 2, \dots,$ kde $s,\ 0 \le s < n$, je největší index takový, že $f(p_n)f(p_s) < 0$.

(B) Steffensenova metoda (viz tvrzení 2.34)

Iterace (2.10a) a (2.10b) lze přepsat na tvar:

$$s^+_{n+1} = s^+_n - f(s^+_n)\frac{-f(s^+_n)}{f(s^+_n) - f(s^+_n + f(s^+_n))} = s^+_n - f(s^+_n)\frac{s^+_n - (s^+_n + f(s^+_n))}{f(s^+_n) - f(s^+_n + f(s^+_n))},$$

což odpovídá při $p_n := s^+_n$ kvazinewtonově metodě s volbou $\widetilde{p}_n := p_n + \underbrace{f(p_n)}_{\approx 0}$.

Podobně:

$$s^-_{n+1} = s^-_n - f(s^-_n)\frac{f(s^-_n)}{f(s^-_n) - f(s^-_n - f(s^-_n))} = s^-_n - f(s^-_n)\frac{s^-_n - (s^-_n - f(s^-_n))}{f(s^-_n) - f(s^-_n - f(s^-_n))},$$

což odpovídá při $p_n := s^-_n$ kvazinewtonově metodě s volbou $\widetilde{p}_n := p_n - \underbrace{f(p_n)}_{\approx 0}$.

(C) Metoda sečen

$p_0 := a,\ p_1 := b$ (počáteční iterace)

$\widetilde{p}_n := p_{n-1},$ pro $n = 1, 2, \dots,$

tj. obdržíme $p_{n+1} = p_n - f(p_n)\frac{p_n - p_{n-1}}{f(p_n) - f(p_{n-1})}$ pro $n = 1, 2, \dots$.

Příklady (MATLAB)**.**
`zktest([],'itsecant')`
Algoritmus (MATLAB)**.**
`vesely('nummet/iterace'): secant.m`
`jmathews('chap_2'): secant.m` *(+ demo* `a2_6.m`*)*

Věta 2.37 (Konvergence metody sečen)**.**
Nechť $f \in C^2[a, b]$ má kořen $p \in [a, b]$, tj. $f(p) = 0$. Jestliže $f'(p) \neq 0$, pak existuje $\delta > 0$ tak, že posloupnost iterací metody sečen $\{p_n\}_{n=0}^{\infty}$ konverguje k p při libovolné počáteční aproximaci $p_0, p_1 \in [p - \delta, p + \delta]$ s řádem konvergence $R = \frac{1}{2}(1 + \sqrt{5}) \approx 1,618$ a asymptotickou chybovou konstantou:

$$A := \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^R} = \left(\frac{1}{2}\left|\frac{f''(p)}{f'(p)}\right|\right)^{\frac{1}{R}}, \text{ kde } e_n := p - p_n.$$

Poznámka 2.38 (Přehled konvergenčních vlastností (kvazi)newtonových metod).

Metoda	Výpočetní náročnost jedné iterace	Řád konvergence
Newton-Raphsonova	$f(p_n),\ f'(p_n)$	2
Steffensenova	$f(p_n),\ f(p_n \pm f(p_n))$	2
sečen	$f(p_n)$	1,618
regula-falsi	$f(p_n)$	1

Dá se ukázat, že **neexistuje kvadraticky konvergentní iterační metoda vyžadující pouze jedno funkční vyhodnocení $f(p_n)$ v každé iteraci**.

2.7. **Systémy nelineárních rovnic.**

$$\begin{array}{rcl} f_1(x_1,x_2,\dots,x_m) &=& 0 \\ f_2(x_1,x_2,\dots,x_m) &=& 0 \\ \vdots & & \\ f_m(x_1,x_2,\dots,x_m) &=& 0 \end{array} \qquad \text{nebo} \qquad \begin{array}{rcl} x_1 &=& g_1(x_1,x_2,\dots,x_m) \\ x_2 &=& g_2(x_1,x_2,\dots,x_m) \\ \vdots & & \\ x_m &=& g_m(x_1,x_2,\dots,x_m) \end{array}$$

$$\Downarrow \qquad \text{vektorový zápis} \qquad \Downarrow$$

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \qquad\qquad \boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x})$$

Příklady (MATLAB)**.**
`zktest([],'itnelrov')`

2.7.1. *Metoda pevného bodu.*

Hledáme $\boldsymbol{p} := [p_1, p_2, \dots, p_m] : \ \boldsymbol{p} = \boldsymbol{g}(\boldsymbol{p})$.

$m = 1$ (viz poznámku 2.21(1'))	$m > 1$
a) g' spojitá v okolí p	a) Spojité parciální derivace $\frac{\partial g_i}{\partial x_j}$ v okolí $\boldsymbol{p}$ pro $i, j = 1, 2, \dots, m$
b) $\lvert g'(p)\rvert < 1$	b) $\left\lvert \frac{\partial g_i}{\partial x_1}(\boldsymbol{p})\right\rvert + \dots + \left\lvert \frac{\partial g_i}{\partial x_m}(\boldsymbol{p})\right\rvert < 1$ pro $i = 1, 2, \dots, m$
$\Downarrow$	
Pro p_0 dostatečně blízko k p platí:	Pro $\boldsymbol{p}^{(0)} = [p_1^{(0)}, p_2^{(0)}, \dots, p_m^{(0)}]$ dostatečně blízko k $\boldsymbol{p}$ platí:
$p_n \to p$ pro $n \to \infty$, kde	$\boldsymbol{p}^{(n)} \to \boldsymbol{p}$ pro $n \to \infty$, kde
$\boxed{p_{n+1} = g(p_n),\ n = 0, 1, 2, \dots}$	$\boxed{\boldsymbol{p}^{(n+1)} = \boldsymbol{g}(\boldsymbol{p}^{(n)}),\ n = 0, 1, 2, \dots}$
	$\Updownarrow$
	$\boxed{\begin{array}{rcl} p_1^{(n+1)} &=& g_1(p_1^{(n)},\ p_2^{(n)}, \dots, p_m^{(n)}) \\ p_2^{(n+1)} &=& g_2(p_1^{(n)},\ p_2^{(n)}, \dots, p_m^{(n)}) \\ \vdots & & \vdots \\ p_m^{(n+1)} &=& g_m(p_1^{(n)}, p_2^{(n)}, \dots, p_m^{(n)}) \end{array}}$

2.7.2. *Seidelova metoda (vylepšená metoda pevného bodu).*

$$\boxed{\begin{array}{rcl} p_1^{(n+1)} &=& g_1(p_1^{(n)},\ p_2^{(n)},\ \dots, p_m^{(n)}) \\ p_2^{(n+1)} &=& g_2(p_1^{(n+1)},\ p_2^{(n)},\ \dots, p_m^{(n)}) \\ \vdots & & \vdots \\ p_m^{(n+1)} &=& g_m(p_1^{(n+1)}, p_2^{(n+1)}, \dots, p_m^{(n)}) \end{array}} \quad \Leftrightarrow \quad \boxed{\begin{array}{l} p_i^{(n+1)} = g_1(p_1^{(n+1)}, \dots, p_{i-1}^{(n+1)}, p_i^{(n)}, \dots, p_m^{(n)}) \\ \text{pro } i = 1, 2, \dots, m. \end{array}}$$

Algoritmus (MATLAB)**.**
`jmathews('chap_2')`: `fix2dim.m` *(+ demo* `a2_9.m`*)*

2.7.3. *Newton-Raphsonova metoda.*

Hledáme $\boldsymbol{p} := [p_1, p_2, \ldots, p_m] : \ \boldsymbol{f}(\boldsymbol{p}) = \boldsymbol{0}$.

$m = 1$ (viz větu 2.24)	$m > 1$
a) f'' spojitá v okolí p	a) Spojité druhé parciální derivace $\frac{\partial^2 g_i}{\partial x_j \partial x_k}$ v okolí $\boldsymbol{p}$ pro $i, j, k = 1, 2, \ldots, m$
b) $f'(p) \neq 0$	b) Jakobián (matice $m \times m$) $J(\boldsymbol{p}) := \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{p}) & \ldots & \frac{\partial f_1}{\partial x_m}(\boldsymbol{p}) \\ \vdots & \ldots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\boldsymbol{p}) & \ldots & \frac{\partial f_m}{\partial x_m}(\boldsymbol{p}) \end{bmatrix}$ je regulární matice.
$\Downarrow$	
Pro p_0 dostatečně blízko k p platí:	Pro $\boldsymbol{p}^{(0)} = [p_1^{(0)}, p_2^{(0)}, \ldots, p_m^{(0)}]$ dostatečně blízko k $\boldsymbol{p}$ platí:
$p_n \to p$ pro $n \to \infty$, kde	$\boldsymbol{p}^{(n)} \to \boldsymbol{p}$ pro $n \to \infty$, kde
$\boxed{p_{n+1} = p_n - \underbrace{\frac{f(p_n)}{f'(p_n)}}_{f'(p_n)^{-1} f(p_n)}, \ n = 0, 1, 2, \ldots}$	$\boxed{\boldsymbol{p}^{(n+1)} = \boldsymbol{p}^{(n)} - J(\boldsymbol{p}^{(n)})^{-1} \boldsymbol{f}(\boldsymbol{p}^{(n)}), \ n = 0, 1, 2, \ldots}$
	$\Updownarrow$
	$\boxed{\begin{array}{rcl} p_1^{(n+1)} & = & p_1^{(n)} - \sum_{j=1}^m a_{1,j} f_j(p_1^{(n)}, \ldots, p_m^{(n)}) \\ p_2^{(n+1)} & = & p_2^{(n)} - \sum_{j=1}^m a_{2,j} f_j(p_1^{(n)}, \ldots, p_m^{(n)}) \\ \vdots & & \vdots \\ p_m^{(n+1)} & = & p_m^{(n)} - \sum_{j=1}^m a_{m,j} f_j(p_1^{(n)}, \ldots, p_m^{(n)}) \end{array}}$,
	kde $J(\boldsymbol{p}^{(n)})^{-1} =: [a_{i,j}]$.

Algoritmus (MATLAB).
`jmathews('chap_2'): new2dim.m` *(+ demo* `a2_10.m`*)*

Příklad 2.39 ($m = 2, \ x := x_1, y := x_2$).

$x^2 - 2x - y + 0,5 = 0$ ... parabola
$x^2 + 4y^2 - 4 = 0$ ... elipsa

Průsečíky obou křivek určují dvě řešení:
Řešení (1): $\boldsymbol{p} = [-0,2; 1,0]$
Řešení (2): $\boldsymbol{p} = [1,9; 0,3]$.

Demonstrace nalezení numerického řešení Seidelovou a Newtonovou metodou v MATLABu:
viz `jmathews('chap_2'): a2_9.m, a2_10.m`.

Řešení (1) (Seidelovou) metodou pevného bodu pro $\boldsymbol{p}^{(0)} = [0; 1]$

Od 2. rovnice odečteme $8y$, abychom zajistili splnění podmínky 2.7.1 b) pro $i = 2$. Obdržíme ekvivalentní systém pro iterace pevného bodu ve tvaru:

$$x = \frac{x^2 - y + 0,5}{2}$$
$$y = \frac{-x^2 - 4y^2 + 8y + 4}{8}.$$

Řešení (2) (Seidelovou) metodou pevného bodu pro $\boldsymbol{p}^{(0)} = [2; 0]$

Od 1. rovnice odečteme $2x$ **a od 2. rovnice odečteme** $11y$, abychom zajistili splnění podmínky 2.7.1 b) pro $i = 1, 2$ — předchozí volba totiž v tomto kořenu tuto podmínku nesplňuje. Obdržíme

ekvivalentní systém pro iterace pevného bodu ve tvaru:

$$x = \frac{-x^2 + 4x + y - 0,5}{2}$$
$$y = \frac{-x^2 - 4y^2 + 11y + 4}{11}.$$

Řešení (2) Newtonovou metodou, např. pro $[x_0, y_0] := \boldsymbol{p}^{(0)} = [2; 0,25]$

$$J(x,y) = \begin{bmatrix} 2x-2 & -1 \\ 2x & 8y \end{bmatrix},$$
$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - J(x_n, y_n)^{-1} \begin{bmatrix} x_n^2 - 2x_n - y_n + 0,5 \\ x_n^2 + 4y_n^2 - 4 \end{bmatrix}.$$

3. Řešení systému lineárních rovnic

(Algoritmy v MATLABu: `vesely('nummet/linrov')`, `jmathews('chap_3')`)

Formulace problému:
Je dán systém m lineárních rovnic (SLR) o n neznámých $x_1, x_2, \dots, x_n$ ve tvaru:

$$\begin{array}{ccccccccccc} a_{1,1}x_1 & + & a_{1,2}x_2 & + & \dots & + & a_{1,n}x_n & = & b_1 & \dots & \text{1. rovnice} \\ a_{2,1}x_1 & + & a_{2,2}x_2 & + & \dots & + & a_{2,n}x_n & = & b_2 & \dots & \text{2. rovnice} \\ \vdots & & \vdots & & \dots & & \vdots & & \vdots & \dots & \vdots \\ a_{m,1}x_1 & + & a_{m,2}x_2 & + & \dots & + & a_{m,n}x_n & = & b_m & \dots & m\text{-tá rovnice} \end{array} . \tag{3.1a}$$

Ekvivalentní maticový zápis systému (3.1a):

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{3.1b}$$

kde je

$\boldsymbol{A} := [a_{i,j}]_{\substack{1\le i\le m\\ 1\le j\le n}}$ ... matice rozměru $m \times n$ nazývaná **maticí SLR** (3.1a), $a_{i,j} \in \mathbb{R}$

$\boldsymbol{b} := \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$ ... **pravá strana** SLR (3.1a), $b_i \in \mathbb{R}$

$\boldsymbol{x} := \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ ... **vektor neznámých** SLR (3.1a), $x_j \in \mathbb{R}$.

Každý vektor $\boldsymbol{x} \in \mathbb{R}$ vyhovující všem rovnicím SLR (3.1a) nazýváme jeho **řešením**.

Poznámka 3.1.

(1) Je-li $\boldsymbol{A}$ matice rozměru $m \times n$, pak značíme:
$\boldsymbol{a}_{:,j}$ ... j-tý sloupec matice $\boldsymbol{A}$,
$\boldsymbol{a}_{i,:}$ ... i-tý řádek matice $\boldsymbol{A}$ a píšeme

$$\boldsymbol{A} = [\boldsymbol{a}_{:,1}, \boldsymbol{a}_{:,2}, \dots, \boldsymbol{a}_{:,n}] = [\boldsymbol{a}_{1,:}; \boldsymbol{a}_{2,:}; \dots; \boldsymbol{a}_{m,:}] = \begin{bmatrix} \boldsymbol{a}_{1,:} \\ \boldsymbol{a}_{2,:} \\ \vdots \\ \boldsymbol{a}_{m,:} \end{bmatrix}.$$

Je $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{a}_{:,1}x_1 + \boldsymbol{a}_{:,2}x_2 + \dots + \boldsymbol{a}_{:,n}x_n \in \mathcal{R}(\boldsymbol{A})$, kde $\mathcal{R}(\boldsymbol{A})$ značí vektorový podprostor v $\mathbb{R}^m$ **generovaný** všemi sloupci matice $\boldsymbol{A}$ (jeho prvky jsou všechny možné lineární kombinace těchto sloupců).
Je-li $m = n$, pak matice $\boldsymbol{A}$ se nazývá **čtvercovou maticí řádu** n.
Je-li $a_{i,j} = 0$ pro $i < j$ (resp. $i > j$), pak matice $\boldsymbol{A}$ se nazývá **dolní (resp. horní) trojúhelníkovou maticí**.

(2) $r(\boldsymbol{A})$ značí hodnost matice $\boldsymbol{A}$.

(3) $\boldsymbol{I}_m := [\delta_{i,j}]$ ... jednotková matice $m \times m$.

(4) $\boldsymbol{p} = [p(1), p(2), \dots, p(m)]^T$ určuje permutaci p na vzestupně uspořádané množině $\{1, 2, \dots, m\}$, tj. i udává pozici, na kterou je permutací p přemístěn prvek $j := p(i),\ i = 1, 2, \dots, m$.
Permutaci lze ekvivalentně popsat **permutační maticí** $\boldsymbol{P} := [p_{i,j}]$, kde
$p_{i,j} := \delta_{p(i),j}, 1 \le i, j \le m$. Zřejmě platí $\boldsymbol{A}[1, 2, \dots, m]^T = \boldsymbol{p}$, takže pro libovolnou matici $\boldsymbol{A}$ o m řádcích a matici $\boldsymbol{B}$ o m sloupcích dostáváme:

$\boldsymbol{P}\boldsymbol{A} = [\boldsymbol{a}_{p(1),:}; \boldsymbol{a}_{p(2),:}; \dots; \boldsymbol{a}_{p(m),:}]$ ... $p(i)$-tý řádek $\boldsymbol{A}$ je přemístěn na i-tou pozici,
$\boldsymbol{B}\boldsymbol{P}^T = [\boldsymbol{b}_{:,p(1)}, \boldsymbol{b}_{:,p(2)}, \dots, \boldsymbol{b}_{:,p(m)}]$ ... $p(j)$-tý sloupec $\boldsymbol{B}$ je přemístěn na j-tou pozici, neboť $\boldsymbol{B}\boldsymbol{P}^T = (\boldsymbol{P}\boldsymbol{B}^{\boldsymbol{T}})^T$.

Odtud zejména dostáváme $\boldsymbol{P}\boldsymbol{I}_m = \boldsymbol{P}$, takže permutační matice $\boldsymbol{P}$ se dostane permutací řádků jednotkové matice.
Je-li $\boldsymbol{D} := \mathrm{diag}(d_1, d_2, \dots, d_m)$ diagonální matice, pak snadno nahlédneme, že
$\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^T = \mathrm{diag}(d_{p(1)}, d_{p(2)}, \dots, d_{p(m)})$ je rovněž diagonální maticí, jejíž prvky na diagonále jsou permutací p diagonálních prvků původní matice. Jsou-li všechny diagonální prvky v $\boldsymbol{D}$

stejné, pak zřejmě $\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^T = \boldsymbol{D}$. Speciálně tedy $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{P}\boldsymbol{I}_m\boldsymbol{P}^T = \boldsymbol{I}_m$, takže $\boldsymbol{P}$ je ortogonální matice a tedy $\boldsymbol{P}^{-1} = \boldsymbol{P}^T$ je permutační matice příslušná k inverzní permutaci p^{-1}.

Definice 3.2. Nechť $\boldsymbol{A}$ je matice SLR (3.1b), pak matici $\widetilde{\boldsymbol{A}} := [\boldsymbol{A}, \boldsymbol{b}]$ rozměru $m \times (n+1)$ nazýváme **rozšířenou maticí** tohoto systému.

Poznámka 3.3. Každý SLR je jednoznačně popsán svou rozšířenou maticí soustavy $\widetilde{\boldsymbol{A}}$. Uvážíme-li poznámku 3.1(1), můžeme vyslovit následující tvrzení o existenci řešení SLR:
SLR daný maticí $\widetilde{\boldsymbol{A}}$ má řešení $\Leftrightarrow$ $\boldsymbol{b} \in \mathcal{R}(\boldsymbol{A})$ $\Leftrightarrow$ $\boldsymbol{b}$ je lineární kombinací sloupců matice $\boldsymbol{A}$ $\Leftrightarrow$ matice $\boldsymbol{A}$ a $\widetilde{\boldsymbol{A}}$ mají stejnou hodnost.
SLR má přitom nejvýše jedno řešení $\Leftrightarrow$ sloupce matice $\boldsymbol{A}$ jsou lineárně nezávislé (tvoří bázi).
Obdrželi jsme tak jednoduchou úvahou známé tvrzení z lineární algebry.

Definice 3.4. Řekneme, že dva SLR $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ a $\boldsymbol{A}'\boldsymbol{x} = \boldsymbol{b}'$ jsou **ekvivalentní**, jestliže oba systémy mají tytéž množiny řešení. V takovém případě píšeme $\widetilde{\boldsymbol{A}} \sim \widetilde{\boldsymbol{A}}'$, kde $\widetilde{\boldsymbol{A}} := [\boldsymbol{A}, \boldsymbol{b}]$ a $\widetilde{\boldsymbol{A}}' := [\boldsymbol{A}', \boldsymbol{b}]$ značí příslušné rozšířené matice těchto systémů (s týmž počtem sloupců).

Lemma 3.5. *Nechť $\widetilde{\boldsymbol{A}}'$ vznikne z $\widetilde{\boldsymbol{A}}$ provedením konečného počtu elementárních operací jednoho z následujících tří typů:*

- *záměna dvou řádků,*
- *vynásobení řádku nenulovou konstantou,*
- *přičtením lineární kombinace některých řádků k jinému řádku.*

Potom $\widetilde{\boldsymbol{A}} \sim \widetilde{\boldsymbol{A}}'$.

3.1. Přímé metody pro řešení systému lineárních rovnic.

Příklady (MATLAB)**.**
`zktest([],'linrov')`

Věta 3.6 (Algoritmus dopředného a zpětného dosazování)**.**
Nechť $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ je SLR, kde $\boldsymbol{A}$ je čtvercová dolní nebo horní trojúhelníková matice řádu n s nenulovými prvky na diagonále ($a_{k,k} \neq 0$ pro $k = 1, 2, \ldots, n$). Pak existuje právě jedno řešení $\boldsymbol{x}$ tohoto systému, které nalezneme pomocí **algoritmu dopředného nebo zpětného dosazování**.

Algoritmus dopředného dosazování pro dolní trojúhelníkovou matici $\boldsymbol{A}$:

$$\boxed{\begin{array}{lcl} x_1 & = & b_1/a_{1,1} \\ x_k & = & \big(b_k - \sum_{j=1}^{k-1} a_{k,j}x_j\big)/a_{k,k} \text{ pro } k = 2, 3, \ldots, n. \end{array}} \tag{3.2a}$$

Algoritmus zpětného dosazování pro horní trojúhelníkovou matici $\boldsymbol{A}$:

$$\boxed{\begin{array}{lcl} x_n & = & b_n/a_{n,n} \\ x_k & = & \big(b_k - \sum_{j=k+1}^{n} a_{k,j}x_j\big)/a_{k,k} \text{ pro } k = n-1, n-2, \ldots, 1. \end{array}} \tag{3.2b}$$

3.7. Výpočetní složitost algoritmů dopředného a zpětného dosazování.

Označíme-li $\mathcal{O}(\pm)$ počet operací sečítání a odčítání (aditivní složitost) a $\mathcal{O}(*, /)$ počet operací násobení a dělení (multiplikativní složitost), pak užitím známého vztahu pro součet členů aritmetické posloupnosti dostáváme pro algoritmy dopředného a zpětného dosazování kvadratickou aditivní i multiplikativní složitost:

Aditivní složitost:

$$\mathcal{O}(\pm) = 0 + 1 + \ldots n - 1 = \frac{n(n-1)}{2} = \boxed{\frac{n^2 - n}{2}}.$$

Multiplikativní složitost:

$$\mathcal{O}(*, /) = 1 + 2 + \ldots n = \frac{n(n+1)}{2} = \boxed{\frac{n^2 + n}{2}}.$$

Pokud matice $\boldsymbol{A}$ má jednotkovou diagonálu ($a_{k,k} = 1$ pro $k = 1, 2, \ldots, n$), pak v každém kroku ušetříme jedno dělení a multiplikativní i aditivní složitost bude stejná:

$$\mathcal{O}(*, /) = 0 + 1 + \ldots n - 1 = \mathcal{O}(\pm) = \boxed{\frac{n^2 - n}{2}}.$$

Algoritmus (MATLAB)**.**
`vesely('nummet/linrov'): zpetdos.m`
`jmathews('chap_3'): backsub.m` *(+ demo* `a3_1.m`*)*

Věta 3.8 (Gaussova eliminace s výběrem pivotního prvku)**.**
Nechť $\boldsymbol{Ax} = \boldsymbol{b}$ *je SLR se čtvercovou regulární maticí soustavy* $\boldsymbol{A}$ *řádu* n*. Pak existuje ekvivalentní SLR* $\boldsymbol{Ux} = \boldsymbol{y}$*,* $[\boldsymbol{U}, \boldsymbol{y}] \sim [\boldsymbol{A}, \boldsymbol{b}]$ *se čtvercovou horní trojúhelníkovou maticí soustavy* $\boldsymbol{U}$ *řádu* n *a nenulovou diagonálou* ($u_{k,k} \neq 0$ *pro* $k = 1, 2, \ldots, n$)*.*

Důkaz: **Algoritmus Gaussovy eliminace s výběrem pivotního prvku**.
Postupně pro $k = 1, 2, \ldots, n$ konstruujeme posloupnost ekvivalentních systémů:
$[\boldsymbol{A}, \boldsymbol{b}] = \boldsymbol{U}_1 \sim \cdots \sim \boldsymbol{U}_n = [\boldsymbol{U}, \boldsymbol{y}]$, kde

$$\boldsymbol{U}_1 := [\boldsymbol{A}, \boldsymbol{b}] =: [u_{j,i}^{(1)}]$$

$$\boldsymbol{U}_k := \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \ldots & a_{1,k-1}^{(1)} & a_{1,k}^{(1)} & \ldots & a_{1,n}^{(1)} & a_{1,n+1}^{(1)} \\ 0 & a_{2,2}^{(2)} & \ldots & a_{2,k-1}^{(2)} & a_{2,k}^{(2)} & \ldots & a_{2,n}^{(2)} & a_{2,n+1}^{(2)} \\ 0 & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & u_{k,k}^{(k)} & \ldots & u_{k,n}^{(k)} & u_{k,n+1}^{(k)} \\ 0 & 0 & \ldots & 0 & u_{k+1,k}^{(k)} & \ldots & u_{k+1,n}^{(k)} & u_{k+1,n+1}^{(k)} \\ \vdots & \vdots & \ldots & \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & u_{n,k}^{(k)} & \ldots & u_{n,n}^{(k)} & u_{n,n+1}^{(k)} \end{bmatrix} =: [\boldsymbol{A}_k, \boldsymbol{b}_k] \sim [\boldsymbol{A}, \boldsymbol{b}],\ k = 2, \ldots, n$$

$a_{n,n}^{(n)} := u_{n,n}^{(n)},\ a_{n,n+1}^{(n)} := u_{n,n+1}^{(n)}$

$\boldsymbol{U} := \boldsymbol{A}_n = [a_{j,i}^{(j)}],\ \boldsymbol{y} := \boldsymbol{b}_n = [a_{j,n+1}^{(j)}]$.

k-tý krok algoritmu Gaussovy eliminace, $k = 1, 2, \ldots, n-1$*:*
Konstrukce $\boldsymbol{U}_{k+1} \sim \boldsymbol{U}_k$, je-li $\boldsymbol{U}_k$ již zkonstruováno. Provedeme vhodné elementární transformace dle lemmatu 3.5:

Výběr pivotního prvku: $\exists j \geq k : \ u_{j,k}^{(k)} \neq 0$, neboť v opačném případě by determinant $\det(\boldsymbol{A}_k) = a_{1,1}^{(1)} a_{2,2}^{(2)} \ldots a_{k-1,k-1}^{(k-1)} \det([u_{j,i}^{(k)}]) = 0$, což je v rozporu s $[\boldsymbol{A}_k, \boldsymbol{b}_k] \sim [\boldsymbol{A}, \boldsymbol{b}]$. Nechť $p \geq k$ je nejmenší řádkový index v $\boldsymbol{U}_k$, kde $|u_{p,k}^{(k)}|$ je maximální, tj. $|u_{p,k}^{(k)}| = \max(|u_{k,k}^{(k)}|, |u_{k+1,k}^{(k)}|, \ldots |u_{n,k}^{(k)}|)$. Prvek $u_{p,k}^{(k)}$ je tedy nenulový a nazývá se **pivotní prvek**.

Záměna p**-tého a** k**-tého řádku:** Nechť $\boldsymbol{P}_k$ značí permutaci, která zamění p-tý a k-tý řádek v $\boldsymbol{U}_k$. Pivotní prvek se tak stane diagonálním, přičemž pro $j \geq k$ nechť $\boldsymbol{a}_{j,:}^{(k)}$ značí j-tý řádek v takto permutované matici $\boldsymbol{P}_k \boldsymbol{U}_k$.

Eliminace prvků v k**-tém sloupci pod pivotním prvkem:** Pro $j = k+1, \ldots, n$ přičteme k j-tému řádku vhodný násobek k-ho řádku tak, aby prvek na (j,k)-té pozici se vynuloval:

$$\boldsymbol{u}_{j,:}^{(k+1)} := \boldsymbol{a}_{j,:}^{(k)} - m_{j,k} \boldsymbol{a}_{k,:}^{(k)}, \text{ kde } m_{j,k} := \frac{\boldsymbol{a}_{j,k}^{(k)}}{\boldsymbol{a}_{k,k}^{(k)}}. \tag{3.3}$$

Skutečně dostáváme $u_{j,k}^{(k+1)} = 0$, přičemž pro $i < k$ zůstává $u_{j,i}^{(k+1)} = 0$, neboť $a_{j,i}^{(k)} = 0$ i $a_{k,i}^{(k)} = 0$.
Eliminaci (3.3) můžeme užitím syntaxe MATLABu algoritmizovat takto:

```
for  j = k+1 : n
     m_{j,k} = a_{j,k}^{(k)} / a_{k,k}^{(k)};
     a_{j,k}^{(k+1)} = 0;
     for  i = k+1 : n+1                                   (3.4)
          u_{j,i}^{(k+1)} = a_{j,i}^{(k)} - m_{j,k} * a_{k,i}^{(k)};
     end
end
```

Skutečnost, že při výpočtu $m_{j,k}$ se dělí právě pivotním prvkem $a_{k,k}^{(k)}$, který byl zvolen v absolutní hodnotě největší možný, příznivě ovlivňuje numerickou stabilitu (snižuje se riziko dělení čísly blízkými k nule, což by mohlo vést k hodnotám $m_{j,k}$ blízkým k $\pm\infty$). □

Důsledek 3.9 (LU-rozklad)**.**
$\boldsymbol{PA} = \boldsymbol{LU}$*, kde matice* $\boldsymbol{P}$*,* $\boldsymbol{L}$ *a* $\boldsymbol{U}$ *jsou určeny algoritmem Gaussovy eliminace:*

$\boldsymbol{P} = \boldsymbol{P}_{n-1} \dots \boldsymbol{P}_2 \boldsymbol{P}_1$ *je permutační matice popisující výslednou permutaci řádků po skončení Gaussovy eliminace,*
$\boldsymbol{L} =: [l_{j,k}]$ *je dolní trojúhelníková matice s jednotkovou diagonálou* $l_{j,j} = 1$ *a* $l_{j,k} = m_{j,k}$ *pro* $j > k$,
$\boldsymbol{U} =: [u_{j,k}]$ *je horní trojúhelníková matice po Gaussově eliminaci.*

Příklad 3.10 (UKÁZKA KROKŮ GAUSSOVY ELIMINACE A LU-ROZKLADU). Řešený systém lineárních rovnic:

$$\begin{bmatrix} 1 & 2 & 1 & 4 \\ 2 & 0 & 4 & 3 \\ 4 & 2 & 2 & 1 \\ -3 & 1 & 3 & 2 \end{bmatrix} * \boldsymbol{x} = \begin{bmatrix} 13 \\ 28 \\ 20 \\ 6 \end{bmatrix}$$

Gaussova eliminace s pivotováním ve sloupcích pod diagonálou:

krok 1:

$$\boldsymbol{U}_1 = \begin{bmatrix} 1 & 2 & 1 & 4 & 13 \\ 2 & 0 & 4 & 3 & 28 \\ \boxed{4} & 2 & 2 & 1 & 20 \\ -3 & 1 & 3 & 2 & 6 \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$\boldsymbol{P}_1\boldsymbol{U}_1 = \begin{bmatrix} \boxed{4} & 2 & 2 & 1 & 20 \\ 2 & 0 & 4 & 3 & 28 \\ 1 & 2 & 1 & 4 & 13 \\ -3 & 1 & 3 & 2 & 6 \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 4 \end{bmatrix}, \ \boldsymbol{m} = \begin{bmatrix} \\ 1/2 \\ 1/4 \\ -3/4 \end{bmatrix}$$

krok 2:

$$\boldsymbol{U}_2 = \begin{bmatrix} 4 & 2 & 2 & 1 & 20 \\ 0 & -1 & 3 & 5/2 & 18 \\ 0 & 3/2 & 1/2 & 15/4 & 8 \\ 0 & \boxed{5/2} & 9/2 & 11/4 & 21 \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 4 \end{bmatrix}$$

$$\boldsymbol{P}_2\boldsymbol{U}_2 = \begin{bmatrix} 4 & 2 & 2 & 1 & 20 \\ 0 & \boxed{5/2} & 9/2 & 11/4 & 21 \\ 0 & 3/2 & 1/2 & 15/4 & 8 \\ 0 & -1 & 3 & 5/2 & 18 \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} 3 \\ 4 \\ 1 \\ 2 \end{bmatrix}, \ \boldsymbol{m} = \begin{bmatrix} \\ 3/5 \\ -2/5 \end{bmatrix}$$

krok 3:

$$\boldsymbol{U}_3 = \begin{bmatrix} 4 & 2 & 2 & 1 & 20 \\ 0 & 5/2 & 9/2 & 11/4 & 21 \\ 0 & 0 & -11/5 & 21/10 & -23/5 \\ 0 & 0 & \boxed{24/5} & 18/5 & 132/5 \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} 3 \\ 4 \\ 1 \\ 2 \end{bmatrix}$$

$$\boldsymbol{P}_3\boldsymbol{U}_3 = \begin{bmatrix} 4 & 2 & 2 & 1 & 20 \\ 0 & 5/2 & 9/2 & 11/4 & 21 \\ 0 & 0 & \boxed{24/5} & 18/5 & 132/5 \\ 0 & 0 & -11/5 & 21/10 & -23/5 \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} 3 \\ 4 \\ 2 \\ 1 \end{bmatrix}, \ \boldsymbol{m} = \begin{bmatrix} \\ \\ -11/24 \end{bmatrix}$$

krok 4:

$$\boldsymbol{U}_4 = \begin{bmatrix} 4 & 2 & 2 & 1 & 20 \\ 0 & 5/2 & 9/2 & 11/4 & 21 \\ 0 & 0 & 24/5 & 18/5 & 132/5 \\ 0 & 0 & 0 & \boxed{15/4} & 15/2 \end{bmatrix}, \qquad \boldsymbol{p} = \begin{bmatrix} 3 \\ 4 \\ 2 \\ 1 \end{bmatrix}$$

Výsledek:

$$\boldsymbol{x} = \begin{bmatrix} 3 \\ -1 \\ 4 \\ 2 \end{bmatrix}, \qquad \begin{array}{llllll} m(2,1) = & 1/2 & \text{patří k 2. řádku v } A & \longrightarrow & \text{3. řádek v} & P*A \\ m(3,1) = & 1/4 & \text{patří k 1. řádku v } A & \longrightarrow & \text{4. řádek v} & P*A \\ m(4,1) = & -3/4 & \text{patří k 4. řádku v } A & \longrightarrow & \text{2. řádek v} & P*A \\ m(3,2) = & 3/5 & \text{patří k 1. řádku v } A & \longrightarrow & \text{4. řádek v} & P*A \\ m(4,2) = & -2/5 & \text{patří k 2. řádku v } A & \longrightarrow & \text{3. řádek v} & P*A \\ m(4,3) = & -11/24 & \text{patří k 1. řádku v } A & \longrightarrow & \text{4. řádek v} & P*A \end{array}$$

Získaný LU-rozklad:

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3/4 & 1 & 0 & 0 \\ 1/2 & -2/5 & 1 & 0 \\ 1/4 & 3/5 & -11/24 & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} 4 & 2 & 2 & 1 \\ 0 & 5/2 & 9/2 & 11/4 \\ 0 & 0 & 24/5 & 18/5 \\ 0 & 0 & 0 & 15/4 \end{bmatrix}$$

3.11. Konstrukce matic $\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{P}$ pomocí modifikované Gaussovy eliminace
Algoritmus Gaussovy eliminace z důkazu věty 3.8 modifikujeme takto:

- $\boldsymbol{b} := [1, 2, \ldots, n]^T$,
- (3.3) provádíme jen s prvými n sloupci matice $\boldsymbol{U}_k$, přičemž místo nul klademe $u_{j,k}^{(k+1)} = m_{j,k}$ pro $j = k+1, \ldots, n$; neboli na místo eliminovaných prvků ukládáme multiplikativní faktory $m_{j,k}$.
- Po poslednim kroku obdržíme matici $\boldsymbol{U} =: [a_{j,i}^{(j)}]$, kde
  $a_{j,i}^{(j)} = l_{j,i}$ pro $j > i$,
  $a_{j,i}^{(j)} = u_{j,i}$ pro $j \leq i \leq n$ a
  $a_{j,n+1}^{(j)} = p_j$ pro $j = 1, 2, \ldots, n$.

Dostáváme dva výsledné postupy pro řešení SLR na bázi Gaussovy eliminace:

3.12. Přímý algoritmus
Sestává ze dvou kroků:

(1) Pomocí algoritmu Gaussovy eliminace popsaném v důkazu věty 3.8 převedeme $\boldsymbol{Ax} = \boldsymbol{b}$ na ekvivalentní systém $\boldsymbol{Ux} = \boldsymbol{y}$ s horní trojúhelníkovou maticí soustavy $\boldsymbol{U}$.
(2) Soustavu $\boldsymbol{Ux} = \boldsymbol{y}$ řešíme algoritmem zpětného dosazování (3.2b).

Algoritmus (MATLAB)**.**
`vesely('nummet/linrov'): gelim.m, jelim.m` (`jelim.m` *je implementace tzv. Jordanovy metody, která převádí SLR na systém s diagonální maticí soustavy, kdy eliminace probíhá i nad diagonálou)*
`jmathews('chap_3'): uptrbk.m` *(+ demo* `a3_2.m`*)*

3.13. Algoritmus nepřímé faktorizace
Sestává ze čtyř kroků:

(1) Konstrukce matic $\boldsymbol{L}, \boldsymbol{U}$ a $\boldsymbol{P}$ podle 3.11.
(2) Provedení permutace pravé strany $\boldsymbol{Pb}$.
(3) Řešíme soustavu $\boldsymbol{Ly} = \boldsymbol{Pb}$ algoritmem dopředného dosazování (3.2a).
(4) Řešíme soustavu $\boldsymbol{Ux} = \boldsymbol{y}$ algoritmem zpětného dosazování (3.2b).

Kroky (3) a (4) jsme obdrželi takto:
$\boldsymbol{Ax} = \boldsymbol{b} \quad \Leftrightarrow \quad \boldsymbol{PAx} = \boldsymbol{Pb} \quad \Leftrightarrow \quad \boldsymbol{LUx} = \boldsymbol{Pb}$, kde položíme $\boldsymbol{Ux} =: \boldsymbol{y}$.

Algoritmus (MATLAB)**.**
`vesely('nummet/linrov'): lufakt.m, lureseni.m`
`jmathews('chap_3'): lufact.m, lusolve.m` *(+ demo* `a3_3.m`*)*

3.14. Výpočetní složitost algoritmů 3.12 a 3.13.
Oba algoritmy mají stejnou kubickou aditivní i multiplikativní složitost:

Aditivní složitost:

$$\mathcal{O}(\pm) = \boxed{\frac{n^3-n}{3} + \frac{n^2-n}{2}}$$

Multiplikativní složitost:

$$\mathcal{O}(*,/) = \boxed{\frac{n^3-n}{3} + n^2}$$

Z hlediska výpočtové složitosti jsou oba algoritmy ekvivalentní. Vzhledem k tomu, že v obou případech má nejvyšší výpočetní nároky triangularizace (převod na horní trojúhelníkový tvar) dle věty 3.8 (faktor $\frac{n^3-n}{3}$), bude algoritmus 3.13 výhodnější v případě, že řešíme systém opakovaně s různými pravými stranami $\boldsymbol{b}$ a stejnou maticí soustavy $\boldsymbol{A}$. V takovém případě totiž stačí provést triangularizaci $\boldsymbol{PA} = \boldsymbol{LU}$ jen jednou, neboť nezávisí na pravé straně, a pak opakovaně provádět kroky (2) až (4) pro jednotlivé pravé strany.

Tohoto postupu lze například využít při výpočtu inverze regulární čtvercové matice $\boldsymbol{A}$ řádu n, kdy $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}_n$ je ekvivalentní s řešením n systémů lineárních rovnic

$$\boldsymbol{A}\boldsymbol{x}_1 = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}, \boldsymbol{A}\boldsymbol{x}_2 = \begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix}, \ldots, \boldsymbol{A}\boldsymbol{x}_n = \begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix},$$

kde $\boldsymbol{x}_i$ reprezentuje i-tý sloupec hledané inverzní matice $\boldsymbol{A}^{-1}$.

3.15. Problémy s numerickou nestabilitou při řešení SLR.
Matice $\boldsymbol{A}$ se nazývá **špatně podmíněná**, jestliže malé změny pravé strany $\boldsymbol{b}$ mají za následek velké změny řešení $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. Tato situace nastává v případech, kdy $\boldsymbol{A}$ je "skoro singulární", tj. některý sloupcový (řádkový) vektor v $\boldsymbol{A}$ svírá malý úhel s nadrovinou generovanou ostatními sloupci (řádky).

Jako příklad uvažme následující systém dvou lineárních rovnic:

$$x + 2y = 2$$
$$2x + 3y = 3,4$$

Přesné řešení $x = 0,8$, $y = 0,6$ aproximujme dosti nepřesně hodnotami $\widetilde{x} = 1$, $\widetilde{y} = 0,48$. Po jejich dosazení do obou rovnic dostáváme po řadě $1,96$ místo 2 a $3,44$ místo $3,4$, což jsou malé odchylky řádově v setinách ve srovnání s chybou řešení, která je řádově v desetinách.

Příčina tkví v tom, že hledáme průsečík dvou přímek, které svírají velmi malý úhel o směrnicích $-\frac{1}{2}$ a $-\frac{2}{3}$. Řádky $[1,2]$ a $[2,3]$ matice soustavy v tomto případě totiž reprezentují souřadnice kolmic k oběma přímkám a svírají tedy stejný úhel jako přímky samotné. Se zmenšujícím se úhlem se blížíme ke stavu lineární závislosti obou vektorů, tj. ke stavu, kdy matice soustavy je singulární.

Podobně lze zkonstruovat situace, kdy při libovolně velké odchylce řešení lze dosáhnout libovolně malé odchylky od pravé strany.

ZÁVĚR: *Nelze tedy hodnotit přesnost nalezeného řešení podle odchylek od pravé strany po jeho dosazení do rovnic!*

Lze pouze hodnotit podmíněnost matice. Ta se měří číslem podmíněnosti matice, které se zpravidla počítá jako poměr největšího ku nejmenšímu singulárnímu číslu matice (singulární čísla matice $\boldsymbol{A}$ jsou druhé odmocniny vlastních čísel symetrické matice $\boldsymbol{A}^T\boldsymbol{A}$ — tedy v případě symetrické matice splývají s jejími vlastními čísly). Toto číslo podmíněnosti v systému MATLAB počítá funkce `cond`. Čím větší číslo podmíněnosti má daná matice, tím je hůře podmíněná a více se blíží k singulární matici. Jinak počítané reciproké číslo podmíněnosti v intervalu $[0,1]$ vrací funkce `rcond`. V tomto případě špatnou podmíněnost indikují hodnoty blízké k nule.

Poznámka 3.16. Gaussovu eliminaci z věty 3.8 lze modifikovat pro případ jakékoliv (i nečtvercové) matice soustavy $\boldsymbol{A}$. Výsledkem je úprava na **schodovitý horní trojúhelníkový tvar**. Zatímco v případě, kdy sloupce $\boldsymbol{A}$ jsou lineárně nezávislé, dostáváme "šířku" schodů rovnou jedné jako v případě věty 3.8, obecně se může stát, že některé "schody" budou širší (při eliminaci nelze vybrat nenulový pivotní prvek, neboť příslušná poddiagonálová část sloupce je nulová — v takovém případě jej ponecháme a přejdeme na další sloupec, atd.).

Dostaneme-li po skončení schodovitou matici $\boldsymbol{U}_n = [\boldsymbol{A}_n, \boldsymbol{y}]$, kde je "schod" mezi $\boldsymbol{A}_n$ a $\boldsymbol{y}$, pak systém nemá řešení, neboť poslední nenulový prvek vektoru $\boldsymbol{y}$ nemůžeme dostat jako lineární kombinaci nul z posledního řádku matice $\boldsymbol{A}_n$. To znamená, že hodnost $r(\boldsymbol{U}_n) = r(\boldsymbol{A}_n) + 1$. Pokud tato situace nenastane, pak x_i korespondující s nadbytečnými sloupci v širších "schodech" odpovídají volným proměnným. Lineární kombinace těchto sloupců s těmito volnými proměnnými převedeme na pravou stranu. Tím se $\boldsymbol{U}$ stane opět standardní horní trojúhelníkovou maticí a systém dořešíme zpětným dosazováním jako v regulárním případě.

Věta 3.17. *Nechť $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, kde $\boldsymbol{A}$ je regulární pásová tridiagonální matice ($a_{i,j} = 0$ pro $|i-j| > 1$) rozměru $n \times n$:*

$$\boldsymbol{A} = \begin{bmatrix} a_1 & c_1 & 0 & \dots & 0 & 0 & 0 \\ b_2 & a_2 & c_2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & b_{n-1} & a_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \dots & 0 & b_n & a_n \end{bmatrix}.$$

Potom její LU-rozklad $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ je tvaru

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ \beta_2 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & \beta_n & 1 \end{bmatrix}, \qquad \boldsymbol{U} = \begin{bmatrix} \alpha_1 & c_1 & 0 & \dots & 0 & 0 \\ 0 & \alpha_2 & c_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \alpha_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \dots & 0 & \alpha_n \end{bmatrix}, \quad \textit{kde}$$

$$\begin{aligned} &\alpha_1 = a_1 \\ &\beta_i = \frac{b_i}{\alpha_{i-1}} \\ &\alpha_i = a_i - \beta_i c_{i-1} \quad \textit{pro } i = 2, 3, \dots, n. \end{aligned} \tag{3.5a}$$

Poznámka 3.18.

(1) Jinou variantu algoritmu (3.5a) lze odvodit pro

$$\boldsymbol{L} = \begin{bmatrix} \alpha_1 & 0 & \dots & 0 & 0 \\ b_2 & \alpha_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \alpha_{n-1} & 0 \\ 0 & 0 & \dots & b_n & \alpha_n \end{bmatrix}, \qquad \boldsymbol{U} = \begin{bmatrix} 1 & \gamma_1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \gamma_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & \gamma_{n-1} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}, \quad \text{kde}$$

$$\begin{aligned} &\alpha_1 = a_1, \ \gamma_1 = \frac{c_1}{\alpha_1} \\ &\alpha_i = a_i - b_i \gamma_{i-1} \\ &\gamma_i = \frac{c_i}{\alpha_i} \qquad \text{pro } i = 2, 3, \dots, n. \end{aligned} \tag{3.5b}$$

(2) Analogické algoritmy obdržíme pro blokově tridiagonální matice, kde $a_i, b_i, c_i, \alpha_i, \beta_i$ (resp. γ_i) jsou submatice kompatibilních rozměrů. Pak místo $\frac{1}{\alpha_{i-1}}$ násobíme inverzní maticí α_{i-1}^{-1} zprava v (3.5a) (resp. místo $\frac{1}{\alpha_i}$ maticí α_i^{-1} zleva v (3.5b)).

(3) V případě obecné regulární čtvercové matice $\boldsymbol{A}$ lze výše popsaný postup zobecnit pro přímý výpočet prvků matic $\boldsymbol{L}$ a $\boldsymbol{U}$ tak, že postupně sestavujeme rovnice pro prvky 1. řádku $\boldsymbol{U}$, 1. sloupce $\boldsymbol{L}$, 2. řádku $\boldsymbol{U}$, 2. sloupce $\boldsymbol{L}$, atd.

3.2. **Iterační metody pro řešení systému lineárních rovnic.**

Příklady (MATLAB)**.**
`zktest([],'itlinrov')`

Princip:
Rovnici $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, kde $\boldsymbol{A}$ je čtvercová regulární matice řádu n, převedeme na rovnici tvaru $\boldsymbol{x} = \boldsymbol{\Gamma}\boldsymbol{x} + \boldsymbol{\gamma}$ a provádíme iterace metodou pevného bodu 2.7.1 nebo Seidelovou metodou 2.7.2 pro vektorovou lineární funkci $\boldsymbol{g}(\boldsymbol{x}) := \boldsymbol{\Gamma}\boldsymbol{x} + \boldsymbol{\gamma}$ o n proměnných.

Obecný postup hledání $\boldsymbol{\Gamma}$ a $\boldsymbol{\gamma}$:
Matici $\boldsymbol{A}$ vhodně rozložíme do tvaru $\boldsymbol{A} = \boldsymbol{N} - \boldsymbol{M}$, kde $\boldsymbol{N}$ je regulární matice. Pak platí

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \;\Leftrightarrow\; (\boldsymbol{N} - \boldsymbol{M})\boldsymbol{x} = \boldsymbol{b} \;\Leftrightarrow\; \boldsymbol{N}\boldsymbol{x} = \boldsymbol{M}\boldsymbol{x} + \boldsymbol{b} \;\Leftrightarrow\; \boldsymbol{x} = \underbrace{\boldsymbol{N}^{-1}\boldsymbol{M}}_{\boldsymbol{\Gamma}}\boldsymbol{x} + \underbrace{\boldsymbol{N}^{-1}\boldsymbol{b}}_{\boldsymbol{\gamma}}.$$

Iterační proces:

$$\boxed{\begin{array}{ll} \boldsymbol{x}^{(0)} & \dots \text{ počáteční iterace} \\ \boldsymbol{x}^{(k+1)} & = \boldsymbol{\Gamma}\boldsymbol{x}^{(k)} + \boldsymbol{\gamma} \quad \text{pro } k = 0, 1, \dots . \end{array}} \tag{3.6}$$

Definice 3.19 (Jacobiho prosté iterace)**.**
Položíme

$$\boldsymbol{N} := \operatorname{diag}(\boldsymbol{A}) = \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{n,n} \end{bmatrix}, \qquad \boldsymbol{M} := \boldsymbol{N} - \boldsymbol{A} = \begin{bmatrix} 0 & -a_{1,2} & \dots & -a_{1,n} \\ -a_{2,1} & 0 & \dots & -a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n,1} & -a_{n,2} & \dots & 0 \end{bmatrix},$$

pak

$$\boldsymbol{\Gamma} = \begin{bmatrix} 0 & -\frac{a_{1,2}}{a_{1,1}} & \dots & -\frac{a_{1,n}}{a_{1,1}} \\ -\frac{a_{2,1}}{a_{2,2}} & 0 & \dots & -\frac{a_{2,n}}{a_{2,2}} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n,1}}{a_{n,n}} & -\frac{a_{n,2}}{a_{n,n}} & \dots & 0 \end{bmatrix}, \qquad \boldsymbol{\gamma} = \begin{bmatrix} \frac{b_1}{a_{1,1}} \\ \frac{b_2}{a_{2,2}} \\ \vdots \\ \frac{b_n}{a_{n,n}} \end{bmatrix}.$$

Rozepsáním (3.6) do jednotlivých rovnic dostáváme

Jacobiho iterační proces:

$$\boxed{\begin{array}{ll} x_1^{(0)}, \dots, x_n^{(0)} & \dots \text{ počáteční iterace} \\ x_j^{(k+1)} & = \frac{1}{a_{j,j}}\big(b_j - \sum_{\substack{i=1 \\ i \neq j}}^{n} a_{j,i}\, x_i^{(k)}\big), \; j = 1, \dots, n; \; k = 0, 1, \dots . \end{array}} \tag{3.7}$$

Vidíme, že vztah pro $x_j^{(k+1)}$ odpovídá formálnímu vyřešení j-té rovnice vzhledem k proměnné x_j.

Algoritmus (MATLAB)**.**
`jmathews('chap_3'): jacobi.m` *(+ demo* `a3_4.m`*)*

Jestliže nyní v (3.7) použijeme pro $i < j$ již spočtenou iteraci $x_i^{(k+1)}$ místo $x_i^{(k)}$, obdržíme následující zřejmě poněkud rychleji konvergující iterační proces, který je obdobou Seidelovy metody 2.7.2.

Definice 3.20 (Gauss-Seidelovy iterace)**.**
Položíme

$$\boldsymbol{N} := \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ a_{2,1} & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix}, \qquad \boldsymbol{M} := \boldsymbol{N} - \boldsymbol{A} = \begin{bmatrix} 0 & -a_{1,2} & \dots & -a_{1,n} \\ 0 & 0 & \dots & -a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix},$$

pak z $\boldsymbol{N}\boldsymbol{x}^{(k+1)} = \boldsymbol{M}\boldsymbol{x}^{(k)} + \boldsymbol{b}$ dostáváme ihned rovnice pro

Gauss-Seidelův iterační proces:

$$\boxed{\begin{array}{l} x_1^{(0)}, \dots, x_n^{(0)} \ \dots \text{ počáteční iterace} \\ \quad x_j^{(k+1)} \ = \ \frac{1}{a_{j,j}} \big(b_j - \sum_{i=1}^{j-1} a_{j,i}\, x_i^{(k+1)} - \sum_{i=j+1}^{n} a_{j,i}\, x_i^{(k)} \big), \ j = 1, \dots, n; \ k = 0, 1, \dots . \end{array}} \tag{3.8}$$

Algoritmus (MATLAB)**.**
`jmathews('chap_3'): gseid.m` *(+ demo* `a3_5.m`*)*

Definice 3.21. Čtvercová matice $\boldsymbol{A}$ řádu n se nazývá **striktně diagonálně dominantní**, jestliže $|a_{j,j}| > \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{j,i}|$ pro $j = 1, 2, \dots, n$.

Věta 3.22 (Konvergence Jacobiho a Gauss-Seidelovy metody)**.**
Nechť $\boldsymbol{A}$ je striktně diagonálně dominantní matice. Potom SLR $\boldsymbol{Ax} = \boldsymbol{b}$ má jediné řešení a Jacobiho i Gauss-Seidelův iterační proces konverguje k tomuto řešení při libovolně zvolené počáteční aproximaci $\boldsymbol{x}^{(0)}$.

Poznámka 3.23. Obecně z konvergence jedné z obou metod neplyne konvergence druhé. Pokud konvergují obě, pak Gauss-Seidelova metoda konverguje rychleji.

Příklad 3.24. Jak ukazuje následující příklad, může pouhou změnou pořadí rovnic dojít ke ztrátě konvergence. Uvažme SLR

$$\begin{array}{rcrcrcr} 4x & - & y & + & z & = & 7 \\ 4x & - & 8y & + & z & = & -21 \\ -2x & + & y & + & 5z & = & 15 \end{array}$$

Jacobiho iterace tohoto SLR odstartované v počáteční aproximaci $\boldsymbol{x}^{(0)} = [1, 2, 2]$ konvergují k přesnému řešení $\boldsymbol{x} = [2, 4, 3]$, neboť matice soustavy je striktně diagonálně dominantní.

Jestliže zaměníme první a poslední rovnici, tak se tato vlastnost poruší a SLR

$$\begin{array}{rcrcrcr} -2x & + & y & + & 5z & = & 15 \\ 4x & - & 8y & + & z & = & -21 \\ 4x & - & y & + & z & = & 7 \end{array}$$

při téže počáteční iteraci diverguje od přesného řešení.

Poznámka 3.25. Iterační metody jsou vhodné zejména pro rozsáhlé systémy lineárních rovnic s řídkou maticí soustavy (malým počtem nenulových prvků), kde u Gaussovy eliminace bývají problémy s numerickou stabilitou a větší výpočetní nároky vzhledem k nemožnosti účelně využít případné řídkosti matice soustavy.

Snažíme se vhodnou záměnou pořadí rovnic (řádkovou permutací rozšířené matice soustavy) a změnou pořadí nezávisle proměnných (sloupcová permutace matice soustavy) soustředit největší nenulové prvky kolem hlavní diagonály s cílem dosáhnout (pokud možno) striktně diagonální dominantnosti. To je zpravidla snazší u řídké matice, která se stane pásovou (nenulové prvky soustředěny kolem hlavní diagonály).

4. Interpolace a aproximace funkcí pomocí polynomů

(Algoritmy v MATLABu: `vesely('nummet/interp')`, `jmathews('chap_4')`)

Formulace aproximační úlohy:

Snaha nahradit danou funkci $f(x)$ vhodnou funkcí $\varphi \in \Phi$ z předem dané třídy funkcí Φ, které mají jisté příznivější výpočetní vlastnosti (např. jednodušší výpočet derivace, integrálu apod.). Funkci $\varphi(x)$ volíme tak, aby se v nějakém smyslu co nejméně lišila od dané funkce $f(x)$. Velikost odchylky $f(x)-\varphi(x)$ měříme zpravidla vhodnou normou $\|\cdot\|$ na nějakém intervalu $[a,b]$. Nejčastěji se používá tzv. p-norma:

$$\|f(x)-\varphi(x)\|_p := \begin{cases} \left(\int_a^b |f(x)-\varphi(x)|^p\,dx\right)^{1/p} \text{ pro } 1 \le p < \infty \\ \max_{x\in[a,b]} |f(x)-\varphi(x)| \text{ pro } p=\infty \end{cases}.$$

Je-li funkce $f(x)$ zadána pouze tabulkou svých hodnot $f(x_0), f(x_1), \dots, f(x_n)$ na konečné diskrétní **síti uzlových bodů (uzlů)** $x_0, x_1, \dots, x_n$, $n \in \mathbb{N}_0$, $x_i \neq x_j$ pro $i \neq j$, pak místo integrální normy používáme vhodnou diskrétní (vektorovou) normu. Diskrétní analogie integrální p-normy měří velikost odchylky mezi vektory hodnot dané a aproximované funkce na síti uzlů $\boldsymbol{x} := [x_0, x_1, \dots, x_n]^T$:

$$\|f(\boldsymbol{x})-\varphi(\boldsymbol{x})\|_p := \begin{cases} \left(\sum_{i=0}^n |f(x_i)-\varphi(x_i)|^p\right)^{1/p} \text{ pro } 1 \le p < \infty \\ \max_{i=0,1,\dots,n} |f(x_i)-\varphi(x_i)| \text{ pro } p=\infty \end{cases}, \text{ kde}$$

$f(\boldsymbol{x}) := [f(x_0), f(x_1), \dots, f(x_n)]^T$ a $\varphi(\boldsymbol{x}) := [\varphi(x_0), \varphi(x_1), \dots, \varphi(x_n)]^T$.

Formulace interpolační úlohy:

Interpolační úloha je speciálním případem diskrétní aproximační úlohy, kde $\|f(\boldsymbol{x})-\varphi(\boldsymbol{x})\| = 0$, tj. aproximační funkce $\varphi(x)$ nabývá ve všech uzlech x_i stejných hodnot jako funkce $f(x)$: $\varphi(x_i) = f(x_i)$ pro $i = 0, 1, \dots, n$. Pak říkáme, že funkce $\varphi(x)$ **interpoluje funkci** $f(x)$ **na síti uzlů** $\boldsymbol{x}$ **v třídě funkcí** Φ.

- Je-li $x \in \mathcal{I}[x_0, x_1, \dots, x_n]$, pak $\varphi(x)$ nazýváme **interpolovanou hodnotou funkce** f **v bodě** x.
- Je-li $x \notin \mathcal{I}[x_0, x_1, \dots, x_n]$, pak $\varphi(x)$ nazýváme **extrapolovanou hodnotou funkce** f **v bodě** x.

Nadále se budeme zabývat pouze **polynomiální interpolací na síti uzlů** $x_0, x_1, \dots, x_n$ v třídě všech polynomů (s reálnými koeficienty) $\Phi := \mathcal{P}_n := \{P_n(x) \,|\, P_n(x)$ polynom stupně nejvýše $n\}$. Polynom $Q_n(x) \in \mathcal{P}_n$, $Q(x_0) = f(x_0), Q(x_1) = f(x_1), \dots, Q(x_n) = f(x_n)$ nazveme **interpolačním polynomem funkce** $f(x)$ **na síti uzlových bodů** $x_0, x_1, \dots, x_n$.

Při řešení praktických aproximačních, resp. interpolačních úloh se často pracuje s bohatší třídou funkcí $\Phi := \mathcal{S}_n$ tvořenou po částech polynomickými funkcemi, tzv. **splajny**, stupně nejvýše n (viz *Spline Toolbox* v MATLABu).

Báze v $\mathcal{P}_n$:

Každý polynom $P_n(x) = p_0 + p_1 x + \cdots + p_n x^n \in \mathcal{P}_n$, $p_i \in \mathbb{R}$, lze z hlediska sečítání a násobení skalárem ztotožnit s jeho vektorem koeficientů $\boldsymbol{p} := [p_0, p_1, \dots, p_n]$, takže $\mathcal{P}_n$ tvoří vektorový prostor dimenze $n+1$ nad tělesem reálných čísel. Zvolíme-li v $\mathcal{P}_n$ vhodný systém bázových polynomů $\mathbb{B}_n := \{B_{n,0}, B_{n,1}, \dots, B_{n,n}\}$, pak každý polynom $P_n \in \mathcal{P}_n$ lze jediným způsobem vyjádřit jako lineární kombinaci bázových polynomů $B_{n,i}$. Zejména pro interpolační polynom $Q_n(x) =: a_0 + a_1 x + \cdots + a_n x^n$ existují jednoznačně určené souřadnice q_i tohoto polynomu v bázi $\mathbb{B}_n$ tak, že $Q_n(x) = \sum_{i=0}^n q_i B_{n,i}(x)$.

Dále ukážeme, že vždy **existuje jediný** interpolační polynom a budeme se zabývat jeho vyjádřením v různých bázích.

4.1. Základní interpolační úloha v přirozené bázi $\mathbb{B}_n = \{x^0, x^1, \dots, x^n\}$.

Polynomy x^i skutečně tvoří bázi, neboť korespondují s $n+1$ lineárně nezávislými vektory $[0, \dots, 0, \underset{\substack{\uparrow \\ i+1}}{1}, 0, \dots, 0]^T$.

Zřejmě $a_i = q_i$ a tedy koeficienty interpolačního polynomu jsou přímo jeho souřadnicemi v této bázi.

Úloha nalezení $Q_n(x)$ je pak ekvivalentní s řešením SLR:

$$\begin{array}{ccccccccc} a_0 & + & a_1 x_0 & + & \dots & + & a_n x_0^n & = & f(x_0) \\ a_0 & + & a_1 x_1 & + & \dots & + & a_n x_1^n & = & f(x_1) \\ \vdots & & \vdots & & \cdots & & \vdots & & \vdots \\ a_0 & + & a_1 x_n & + & \dots & + & a_n x_n^n & = & f(x_n) \end{array} \quad \Leftrightarrow \quad \boxed{\boldsymbol{V}_{n+1}\boldsymbol{a} = f(\boldsymbol{x})},$$

kde

$$\boldsymbol{V}_{n+1} = \begin{bmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_n & \dots & x_n^n \end{bmatrix} = [\boldsymbol{x}^0, \boldsymbol{x}^1, \dots, \boldsymbol{x}^n]$$

je tzv. **Vandermondova matice řádu** $n+1$.

Pro determinant této matice platí známý vztah

$$|\boldsymbol{V}_{n+1}| = \prod_{i>k} (x_i - x_k) \neq 0,$$

neboť $x_i \neq x_k$ pro $i \neq k$. Matice $\boldsymbol{V}_{n+1}$ **je tedy regulární a SLR má právě jedno řešení**.

Dokázali jsme tak následující tvrzení:

Věta 4.1. *Existuje právě jeden interpolační polynom $Q_n(x)$ stupně nejvýše n procházející body $(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n))$.*

Věta 4.2 (Chyba interpolace)**.**
Nechť $f \in C^{n+1}[a,b]$ a $Q_n(x)$ je její interpolační polynom v $n+1$ uzlových bodech $x_i \in [a,b]$, $i = 0, 1, \dots, n$. Pak pro každé $x \in [a,b]$ existuje $\xi := \xi(x) \in \mathcal{I}[x_0, x_1, \dots, x_n, x]$ tak, že

$$E_n(x) := f(x) - Q_n(x) = \omega_{n+1}(x) \frac{f^{(n+1)}(\xi)}{(n+1)!}, \tag{4.1a}$$

$$|E_n(x)| \leq \big(\max_{x \in [a,b]} |\omega_{n+1}(x)| \big) \frac{M_{n+1}}{(n+1)!}, \ x \in [a,b], \tag{4.1b}$$

kde $\omega_{n+1}(x) := (x - x_0)(x - x_1) \dots (x - x_n)$ a $|f^{(n+1)}(x)| \leq M_{n+1}$ na $[a,b]$ ($f^{(n+1)} \in C$, takže dle věty 1.3 je $f^{(n+1)}(x)$ ohraničená na $[a,b]$).

Důsledek 4.3.
Nechť za předpokladů věty 4.2 je $a = x_0, b = x_n$ a $x_i = x_0 + ih$, $i = 0, 1, \dots, n$, ekvidistantní síť s krokem $h > 0$. Pak pro každé $x \in [x_0, x_n]$ existuje $\xi := \xi(x) \in [x_0, x_n]$ tak, že

$$E_n(x) = E_n(x_0 + th) = \Pi_{n+1}(t) \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1}, \tag{4.2a}$$

$$|E_n(x)| \leq \big(\max_{t \in [0,n]} |\Pi_{n+1}(t)| \big) \frac{M_{n+1}}{(n+1)!} h^{n+1}, \ \text{tj. } f(x) = Q_n(x) + O(h^{n+1}) \text{ pro } x \in [x_0, x_n], \tag{4.2b}$$

kde $\Pi_{n+1}(t) := t(t-1) \dots (t-n)$, $t \in [0, n]$.
Speciálně pro $n = 1, 2, 3$ dostáváme odhady

$$\begin{aligned} |E_1(x)| &\leq \tfrac{M_2}{8} h^2 \quad \text{pro} \quad x \in [x_0, x_1], \\ |E_2(x)| &\leq \tfrac{M_3}{9\sqrt{3}} h^3 \quad \text{pro} \quad x \in [x_0, x_2], \\ |E_3(x)| &\leq \tfrac{M_4}{24} h^4 \quad \text{pro} \quad x \in [x_0, x_3]. \end{aligned} \tag{4.2c}$$

Poznámka 4.4 (Rungeho jev).

Otázka: Platí $\max_{x \in [a,b]} |E_n(x)| \to 0$ pro $n \to \infty$?
Odpověď:

a) **ano pro funkci f, jejíž všechny derivace jsou ohraničeny společnou konstantou M ($M_n \leq M$ pro $n = 1, 2, \dots$)**. Pak totiž $\max_{x \in [a,b]} |\omega_{n+1}(x)|$ roste pomaleji, než $(n+1)!$ pro $n \to \infty$ a tudíž celý výraz v (4.1b) konverguje k nule. Tato podmínka je splněna například pro funkce $\sin(x)$ a e^x.

b) **obecně nikoliv:** například v případě funkce $f(x) = \frac{1}{1+12x^2}$ je $\max_{x \in [-1,1]} |E_n(x)|$ rostoucí nade všechny meze (tzv. **Rungeho jev**), neboť $M_n \to \infty$ zesiluje maxima oscilující funkce $|\omega_{n+1}(x)|$ tak, že se zvětšujícím se počtem uzlů roste amplituda oscilací rychleji, než $(n+1)!$. Rungeho efekt je pro tuto funkci výrazný již na 11-ti bodové ekvidistantní síti ($n = 10$). V MATLABu koeficienty $\boldsymbol{a}$ interpolačního polynomu pro $\boldsymbol{y} = f(\boldsymbol{x})$ snadno spočteme pomocí funkce `a=polyfit(x,y,n)`, která je vrací v obráceném pořadí. Nakonec zvolíme jemnou síť `X=linspace(-1,1).'` pro kreslení grafů. Rungeho efekt pak znázorníme vykreslením průběhů $\boldsymbol{Y} = f(\boldsymbol{X})$ a $\boldsymbol{P} = Q_n(\boldsymbol{X})$ ($\boldsymbol{P}$ vypočteme pomocí `P=polyval(a,X)`) do jednoho grafu příkazem `plot(X,[Y,P])`.

Pro dané n následující věta udává optimální (neekvidistantní) volbu uzlů, pro niž maxima oscilací $|\omega_{n+1}(x)|$ jsou minimalizována a je tak odstraněn Rungeho efekt.

Věta 4.5 (Čebyševovy uzly)**.**
Položme $\widehat{x}_k = t_k\frac{b-a}{2} + \frac{a+b}{2}$, *kde* $t_k = \cos\big((2n+1-2k)\pi/(2n+2)\big)$ *pro* $k = 0, 1, \ldots, n$. *Potom pro* $\widehat{\omega}_{n+1}(x) = (x-\widehat{x}_0)(x-\widehat{x}_1)\ldots(x-\widehat{x}_n)$ *platí* $\max_{x\in[a,b]}|\widehat{\omega}_{n+1}(x)| \le 2\big(\frac{b-a}{4}\big)^{n+1}$, *přičemž pro každou jinou síť* $n+1$ *uzlů* $x_0, x_1, \ldots, x_n$ *je* $\max_{x\in[a,b]}|\omega_{n+1}(x)| \ge \max_{x\in[a,b]}|\widehat{\omega}_{n+1}(x)|$. *Jestliže* $f \in C^{n+1}[a,b]$, *pak interpolační polynom v uzlech* $\widehat{x}_k$ *(tzv.* **Čebyševovy uzly***) dává nejmenší možný odhad chyby*

$$|E_n(x)| \le 2\Big(\frac{b-a}{4}\Big)^{n+1}\frac{\max_{x\in[a,b]}|f^{(n+1)}(x)|}{(n+1)!},\ x\in[a,b]. \tag{4.2d}$$

Poznámka 4.6.
Použijeme-li pro funkci $f(x) = \frac{1}{12+x^2}$ z poznámky 4.4 Čebyševovy uzly, pak $\max_{x\in[a,b]}|E_n(x)| \to 0$ pro $n \to \infty$ a Rungeho efekt je tak odstraněn. Toto platí obecně pro každou funkci i při snížených požadavcích na její hladkost. Dá se totiž ukázat, že pro každou funkci $f \in C^1[a,b]$ její interpolační polynom $Q_n(x)$ v Čebyševových uzlech **vždy konverguje, a to dokonce stejnoměrně,** k funkci f na intervalu $[a,b]$ a je tedy odstraněn Rungeho efekt.

4.2. Lagrangeův interpolační polynom ($\mathbb{B}_n = \{L_{n,0}(x), L_{n,1}(x), \ldots, L_{n,n}(x)\}$).

$$L_{n,i}(x) := \frac{(x-x_0)\ldots(x-x_{i-1})(x-x_{i+1})(x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})(x_i-x_n)} = \frac{\prod_{\substack{j=0\\ j\ne i}}^{n}(x-x_j)}{\prod_{\substack{j=0\\ j\ne i}}^{n}(x_i-x_j)} = \tag{4.3a}$$
$$= \frac{\omega_{n+1}(x)}{(x-x_i)\omega'_{n+1}(x_i)} \quad \ldots \quad \text{tzv. } \textbf{Lagrangeův interpolant}.$$

Zřejmě $L_{n,i}(x_j) = \delta_{i,j}$ (Kroneckerův symbol), takže ihned snadno ověříme lineární nezávislost polynomů $L_{n,i}$:

$$a(x) := \sum_{j=0}^{n} a_j L_{n,j}(x) \equiv 0 \ \Rightarrow\ 0 = a(x_i) = a_i \text{ pro } i = 0, 1, \ldots, n.$$

Tedy $L_{n,0}, L_{n,1}, \ldots, L_{n,n}$ skutečně tvoří tzv. **Lagrangeovu bázi** v $\mathcal{P}_n$.

V této bázi dostáváme následující vyjádření interpolačního polynomu:

$$Q_n(x) = \sum_{i=0}^{n} f(x_i)L_{n,i}(x) = \sum_{i=0}^{n} f(x_i)\frac{\omega_{n+1}(x)}{(x-x_i)\omega'_{n+1}(x_i)} \tag{4.3b}$$

Tedy $q_i = f(x_i)$ a funkční hodnoty $f(x_i)$ tak jsou přímo souřadnicemi interpolačního polynomu v Lagrangeově bázi.

Poznamenejme ještě, že v případě $n = 0$ klademe $\prod_{\substack{j=0\\ j\ne i}}^{n}(x-x_j) = 1$ ve (4.3a), a tedy $L_{0,0} \equiv 1$.

Obecně $L_{n,i}(x_i) = 1$ není maximální hodnotou polynomu $L_{n,i}(x)$ na intervalu $\mathfrak{I}[x_0, x_1, \ldots, x_n]$.

Věta 4.7 (Lagrangeův interpolační polynom pro ekvidistantní uzly)**.**
Je-li $x_i = x_0 + ih$, $h > 0$, $i = 0, 1, \ldots, n$, *síť ekvidistantních uzlů, pak* (4.3a) *a* (4.3b) *nabudou po řadě tvaru:*

$$L_{n,i}(x_0+th) = \frac{(-1)^{n-i}}{i!\,(n-i)!}\prod_{\substack{j=0\\ j\ne i}}^{n}(t-j) = \frac{(-1)^{n-i}}{n!}\binom{n}{i}\frac{\Pi_{n+1}(t)}{t-i} \tag{4.4a}$$

a

$$Q_n(x_0+th) = \sum_{i=0}^{n} f(x_i)L_{n,i}(x_0+th) = \frac{\Pi_{n+1}(t)}{n!}\sum_{i=0}^{n}(-1)^{n-i}\binom{n}{i}\frac{f(x_i)}{t-i}. \tag{4.4b}$$

Algoritmus (MATLAB)**.**
`vesely('nummet/interp'): lagr.m`
`jmathews('chap_4'): lagran.m` *(+ demo* `a4_3.m`*)*

4.3. **Newtonův interpolační polynom ($\mathbb{B}_n = \{\omega_0(x), \omega_1(x), \dots, \omega_n(x)\}$).**

Nevýhodou vyjádření interpolačního polynomu v přirozené či Lagrangeově bázi je nutnost sestrojit interpolační polynom zcela znovu, kdykoliv potřebujeme ke stávající síti přidat další uzel.

Tento problém odstraňuje použití **Newtonovy báze** $\mathbb{B}_n = \{\omega_0(x), \omega_1(x), \dots, \omega_n(x)\}$, kde $\omega_0(x) \equiv 1$ a pro $i > 0$ je $\omega_i(x) = (x - x_0)(x - x_1)\dots(x - x_{i-1})$ polynom stupně i jako ve větě 4.2. Snadno nahlédneme, že tento systém je lineárně nezávislý. Totiž matice, jejímiž sloupci jsou po řadě vektory koeficientů $\omega_i(x)$ pro $i = 0, 1, \dots, n$, tvoří čtvercovou horní trojúhelníkovou matici řádu $n + 1$ s jednotkovou diagonálou ($\omega_i(x)$ má jednotkový koeficient u nejvyšší mocniny x^i). Tato matice je regulární s determinantem rovným jedné, takže její sloupce musí být lineárně nezávislé.

Věta 4.8.
Nechť $Q_k(x) = q_0 + q_1\omega_1(x) + \dots + q_k\omega_k(x)$ je vyjádření interpolačního polynomu funkce $f(x)$ v uzlových bodech $x_0, x_1, \dots, x_k$ $(k = 0, 1, \dots, n)$. Pro tento tzv. **Newtonův tvar interpolačního polynomu** *platí následující rekurentní vztah pro výpočet souřadnic q_i:*

$$\begin{aligned} Q_0(x) &= q_0 = f(x_0) \\ Q_k(x) &= Q_{k-1}(x) + q_k\omega_k(x) \text{ pro } k = 1, \dots, n, \text{ kde} \\ q_k &= \frac{f(x_k) - Q_{k-1}(x_k)}{\omega_k(x_k)} = \frac{f(x_k) - Q_{k-1}(x_k)}{\prod_{i=0}^{k-1}(x_k - x_i)}. \end{aligned} \tag{4.5}$$

Poznamenejme, že pro vyhodnocení Newtonova interpolačního polynomu $Q_n(x)$ v proměnné $x = z$ lze použít zobecnění Hornerova schématu z odstavce 1.2:

$$\underbrace{(\dots((}_{n-1} q_n(z - x_{n-1}) + q_{n-1})(z - x_{n-2}) + q_{n-2})(z - x_{n-3}) + \dots + q_1)(z - x_0) + q_0.$$

Definice 4.9. Koeficient q_k v (4.5) se nazývá **diferenční kvocient k-ho řádu** a značí se $q_k =: f[x_0, x_1, \dots, x_k]$ (alternativně též $q_k =: f_{0,1,\dots,k}$ nebo $q_k =: [x_0, x_1, \dots, x_k]_f$). Newtonův interpolační polynom pak můžeme psát ve tvaru:

$$Q_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \dots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1}), \tag{4.6a}$$

kde $f[x_0] = f(x_0)$.

Věta 4.10. *Diferenční kvocienty lze vyjádřit v explicitním tvaru*

$$q_k = f[x_0, x_1, \dots, x_k] = \sum_{i=0}^{k} \frac{f(x_i)}{\prod_{\substack{j=0 \\ j\neq i}}^{k}(x_i - x_j)} = \sum_{i=0}^{k} \frac{f(x_i)}{\omega'_{k+1}(x_i)}. \tag{4.6b}$$

Důsledek 4.11. *V případě ekvidistantní sítě $x_i = x_0 + ih$, $h > 0$, $i = 0, 1, \dots, k$, nabude* (4.6b) *tvaru*

$$q_k = f[x_0, x_1, \dots, x_k] = \frac{1}{k!\,h^k} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} f(x_i). \tag{4.6c}$$

Důkaz. V důkazu věty 4.7 jsme ukázali

$$\prod_{\substack{j=0 \\ j\neq i}}^{k} (x_i - x_j) = \frac{h^k i!\,(k-i)!}{(-1)^{k-i}} = \frac{h^k k!}{(-1)^{k-i}\binom{k}{i}}.$$

Po dosazení do (4.6b) dostaneme ihned (4.6c). □

Důsledek 4.12.
Je-li p libovolná permutace indexů $\{0, 1, \dots, k\}$, pak $q_k = f[x_0, x_1, \dots, x_k] = f[x_{p(0)}, x_{p(1)}, \dots, x_{p(k)}]$. Hodnota diferenčního kvocientu $f[x_0, x_1, \dots, x_k]$ tedy nezávisí na pořadí uzlových bodů $x_0, x_1, \dots, x_k$.

Důkaz. Tvrzení je bezprostředním důsledkem (4.6b), neboť sumace ani součin zde nezávisí na pořadí. □

Věta 4.13 (Rekurentní vztah pro diferenční kvocienty)**.**

$$f[x_0, x_1, \dots, x_k] = \frac{f[x_1, x_2, \dots, x_k] - f[x_0, x_1, \dots, x_{k-1}]}{x_k - x_0} \tag{4.7a}$$

Důsledek 4.14. *V případě ekvidistantní sítě $x_i = x_0 + ih$, $h > 0$, $i = 0, 1, \dots, k$, nabude* (4.7a) *tvaru*

$$f[x_0, x_1, \dots, x_k] = \frac{\Delta^k f(x_0)}{k!\, h^k}, \tag{4.7b}$$

kde pro k-tou diferenci $\Delta^k f(x_0)$ (viz definici 2.4) platí vztah

$$\Delta^k f(x_0) = \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} f(x_i). \tag{4.7c}$$

Důsledek 4.15 (1. Newtonova interpolační formule pro ekvidistantní uzly)**.**
Je-li $x_i = x_0 + ih$, $h > 0$, $i = 0, 1, \dots, n$, ekvidistantní síť uzlů a položíme-li $x = x_0 + th$, $t \in [0, n]$, pak (4.6a) *lze psát ve tvaru*

$$Q_n(x_0 + th) = f(x_0) + \frac{t}{1!}\Delta f(x_0) + \frac{t(t-1)}{2!}\Delta^2 f(x_0) + \cdots + \frac{t(t-1)\dots(t-(n-1))}{n!}\Delta^n f(x_0) = \tag{4.8a}$$

$$= \sum_{k=0}^{n} \frac{\Pi_k(t)}{k!}\Delta^k f(x_0).$$

Důsledek 4.16 (2. Newtonova interpolační formule pro ekvidistantní uzly)**.**
Je-li $x_i = x_0 + ih$, $h > 0$, $i = 0, 1, \dots, n$, ekvidistantní síť uzlů a položíme-li $x = x_n + th$, $t \in [-n, 0]$, pak (4.6a) *lze psát ve tvaru*

$$Q_n(x_n + th) = f(x_n) + \frac{t}{1!}\Delta f(x_{n-1}) + \frac{t(t+1)}{2!}\Delta^2 f(x_{n-2}) + \cdots + \frac{t(t+1)\dots(t+n-1)}{n!}\Delta^n f(x_0). \tag{4.8b}$$

Poznámka 4.17. Pokud $x_0 < x_1 < \cdots < x_n$, pak vzorec (4.8a) je z hlediska numerické stability vhodný pro interpolaci na začátku intervalu, kde $t = \frac{x - x_0}{h}$ nabývá malých hodnot, zatímco (4.8b) se naopak hodí pro interpolaci na konci intervalu.

Algoritmus 4.18 (Diferenční schéma)**.**

x_i	$f[\cdot]$	$f[\cdot\cdot]$	$f[\cdot\cdot\cdot]$	$f[\cdot\cdot\cdot\cdot]$	$\dots$
x_0	$\boxed{f_0}$				
x_1	f_1	$\frac{f_1 - f_0}{x_1 - x_0} = \boxed{f_{0,1}}$			
x_2	f_2	$\frac{f_2 - f_1}{x_2 - x_1} = f_{1,2}$	$\frac{f_{1,2} - f_{0,1}}{x_2 - x_0} = \boxed{f_{0,1,2}}$		
x_3	f_3	$\frac{f_3 - f_2}{x_3 - x_2} = f_{2,3}$	$\frac{f_{2,3} - f_{1,2}}{x_3 - x_1} = f_{1,2,3}$	$\frac{f_{1,2,3} - f_{0,1,2}}{x_3 - x_0} = \boxed{f_{0,1,2,3}}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

V ekvidistantním případě stačí dle (4.7b) *počítat místo diferenčních kvocientů pouze diference:*

x_i	f_i	Δf_i	$\Delta^2 f_i$	$\Delta^3 f_i$	$\dots$
x_0	$\boxed{f_0}$				
x_1	f_1	$f_1 - f_0 = \boxed{\Delta f_0}$			
x_2	f_2	$f_2 - f_1 = \Delta f_1$	$\Delta f_1 - \Delta f_0 = \boxed{\Delta^2 f_0}$		
x_3	f_3	$f_3 - f_2 = \Delta f_2$	$\Delta f_2 - \Delta f_1 = \Delta^2 f_1$	$\Delta^2 f_1 - \Delta^2 f_0 = \boxed{\Delta^3 f_0}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Algoritmus (MATLAB)**.**
`jmathews('chap_4'): newpoly.m` *(+ demo* `a4_5.m`*)*

Věta 4.19 (Jiné vyjádření chyby interpolace)**.**
Nechť $Q_n(x)$ je interpolační polynom funkce $f(x)$ v uzlových bodech $x_0, x_1, \dots, x_n$, pak pro libovolné $x \notin \{x_0, x_1, \dots, x_n\}$ lze chybu interpolace vyjádřit ve tvaru

$$E_n(x) = f(x) - Q_n(x) = \omega_{n+1}(x) f[x_0, x_1, \dots, x_n, x]. \tag{4.9}$$

Poznámka 4.20. Porovnejme za dodatečného předpokladu $f \in C^{n+1}[a, b]$ oba vztahy pro chybu (4.1a) a (4.9). Vidíme, že musí existovat $\xi = \xi(x) \in \mathcal{I}[x_0, x_1, \dots, x_n, x]$ takové, že

$$f[x_0, x_1, \dots, x_n, x] = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}.$$

Dá se dokonce ukázat, že pro $f \in C^k[a,b]$, lze $f[x_0, x_1, \ldots, x_k]$ spojitě prodloužit na funkci $k+1$ proměnných $x_0, x_1, \ldots, x_k \in [a,b]$ (viz následující dvě tvrzení), kdy již nemusíme předpokládat, že x_i jsou navzájem různé.

Věta 4.21.
Nechť $f \in C^k[a,b]$, $m \le f^{(k)}(x) \le M$ pro každé $x \in \mathcal{I}[x_0, x_1, \ldots, x_k]$, kde $x_0, x_1, \ldots, x_k \in [a,b]$ jsou libovolné. Pak platí

$$f[x_0, x_1, \ldots, x_k] \text{ je spojitá funkce } k+1 \text{ proměnných na } [a,b]^{k+1}, \text{ jejíž hodnota nezávisí na pořadí } x_0, x_1, \ldots, x_k \text{ ve smyslu důsledku 4.12} \tag{4.10}$$

$$\frac{m}{k!} \le f[x_0, x_1, \ldots, x_k] \le \frac{M}{k!} \tag{4.11}$$

$$f[x_0, x_1, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!} \text{ pro vhodné } \xi \in \mathcal{I}[x_0, x_1, \ldots, x_k] \tag{4.12}$$

$$f[\underbrace{x, x, \ldots, x}_{(k+1)\times}] = \frac{f^{(k)}(x)}{k!} \tag{4.13}$$

Důsledek 4.22. *Vztah* (4.9) *platí pro každé $x \in [a,b]$.*

Poznámka 4.23. Jestliže totožné uzly uspořádáme vedle sebe, lze pro výpočet $f[x_0, x_1, \ldots, x_k]$ modifikovat (4.7a) a výsledné diferenční schéma 4.18 tak, že za výrazy typu $f[\underbrace{x_i, x_i, \ldots, x_i}_{(j+1)\times}]$ dosazujeme $\frac{f^{(j)}(x_i)}{j!}$ podle (4.13).

Tento algoritmus je základem tzv. **Hermitovy interpolace**, kde v $(j+1)$-násobném uzlu x_i předepisujeme kromě funkční hodnoty $f(x_i)$ ještě dalších j derivací $f'(x_i), f''(x_i), \ldots, f^{(j)}(x_i)$.

Poznámka 4.24. Pro funkci $f \in C^{n+1}[a,b]$ lze podle důsledku 4.22 a s uvážením tvrzení (4.10) a (4.12) vybrat $\xi(x) \in \mathcal{I}[x_0, x_1, \ldots, x_n, x]$ tak, že $(n+1)!\, f[x_0, x_1, \ldots, x_n, x] = f^{(n+1)}(\xi(x))$ je **spojitá funkce v proměnné** x **na** $[a,b]$ při pevně zvolených uzlových bodech $x_0, x_1, \ldots, x_n$.

Následující věta říká, že zvýšením hladkosti funkce $f(x)$ o r řádů se obdobně zvýší hladkost funkce $f^{(n+1)}(\xi(x))$ až do řádu r.

Věta 4.25.
Nechť $f \in C^{n+r+1}[a,b]$ ($r \in \mathbb{N}_0$) je dána v uzlech $x_0, x_1, \ldots, x_n \in [a,b]$ a $\xi(x) \in \mathcal{I}[x_0, x_1, \ldots, x_n, x]$ je prvek z chyby interpolace (4.1a) *vybraný podle poznámky 4.24 stejně jako v* (4.12).
Potom $f^{(n+1)}(\xi(x)) \in C^r[a,b]$ a platí

$$\frac{d^r}{dx^r} \frac{f^{(n+1)}(\xi(x))}{(n+1)!} = r!\, \frac{f^{(n+r+1)}(\xi_r(x))}{(n+r+1)!}, \tag{4.14}$$

kde $\xi_r := \xi_r(x) \in \mathcal{I}[x_0, x_1, \ldots, x_n, x]$, $\xi_0(x) := \xi(x)$.

5. Numerické derivování

(Algoritmy v MATLABu: `jmathews('chap_6')`)

Princip numerického derivování:

Numerické derivování používáme pro přibližný výpočet prvé nebo vyšší derivace funkce $f(x)$ zadané tabulkou hodnot v uzlových bodech $x_0, x_1, \dots, x_n$. Této metody můžeme s výhodou použít i v případě, kdy výpočet derivací $f(x)$ je obtížný pro složité analytické vyjádření.

Nejčastěji klademe $f'(x) \approx \varphi'(x)$, kde $\varphi(x)$ je vhodná funkce aproximující dobře $f(x)$ na zvoleném intervalu $[a, b]$. Nejjednodušší formule pro numerické derivování funkce $f(x)$ lze získat derivováním jejího interpolačního polynomu: $f'(x) \approx Q_n'(x)$. V případě derivací vyšších řádů postupujeme obdobně.

V každém případě je nutno počítat s tím, že **numerické derivování je méně přesné** než pouhá interpolace, přičemž chyby vzrůstají progresivně s řádem derivace. Je to dáno především tím, že z blízkosti funkčních hodnot v sousedních uzlových bodech nemusí obecně plynout blízkost derivací, pokud funkce není dostatečně hladká. Vysoká hladkost funkce bude tedy nutným požadavkem pro získání hodnot derivací s vyhovující přesností.

Nechť $Q_n(x)$ interpoluje funkci $f(x)$v uzlových bodech $x_0, x_1, \dots, x_n$. Pak z vyjádření $f(x) = Q_n(x) + E_n(x)$ dostáváme

$$\begin{aligned} f'(x) &= Q_n'(x) + E_n'(x) \\ f^{(r)}(x) &= Q_n^{(r)}(x) + E_n^{(r)}(x),\ r \in \mathbb{N}. \end{aligned} \tag{5.1}$$

Zatímco vyjádření derivace $Q_n^{(r)} = \sum_{i=0}^{n} f(x_i) L_{n,i}^{(r)}(x)$ dostáváme přímo z Lagrangeova tvaru (4.3b), formální vyjádření chyby derivace jako r-té derivace chybového členu $E_n(x)$ z (4.1a) je podstatně pracnější a vyžaduje užití věty 4.25.

Omezme se nejprve na případ 1. derivace.

Věta 5.1 (Chyba 1. derivace)**.**
Nechť $f \in C^{n+2}[a,b]^{*)}$ a $Q_n(x)$ je její interpolační polynom v $n+1$ uzlových bodech $x_0, x_1, \dots, x_n \in [a,b]$. Pak pro každé $x \in [a,b]$ platí

$$E_n'(x) = E_{(0)}(x) + E_{(1)}(x),\ \textit{kde} \tag{5.2a}$$

$$\begin{aligned} E_{(0)}(x) &= \omega_{n+1}'(x) \frac{f^{(n+1)}(\xi_0(x))}{(n+1)!}, \\ E_{(1)}(x) &= \omega_{n+1}(x) \frac{f^{(n+2)}(\xi_1(x))}{(n+2)!},\ \textit{kde } \xi_0, \xi_1 \in \mathcal{I}[x_0, x_1, \dots, x_n, x]\ a \\ E_{(1)}(x) &= 0 \textit{ pro } x \in \{x_0, x_1, \dots, x_n\},\ \textit{kdy stačí předpokládat pouze } f \in C^{n+1}[a,b]^{*)}. \end{aligned} \tag{5.2b}$$

Podobně pro r-tou derivaci dostaneme po úpravě.

Věta 5.2 (Chyba r-té derivace)**.**
Nechť $f \in C^{n+r+1}[a,b]^{*)}$, $r \in \mathbb{N}$, a $Q_n(x)$ je její interpolační polynom v $n+1$ uzlových bodech $x_0, x_1, \dots, x_n \in [a,b]$. Pak pro každé $x \in [a,b]$ platí

$$E_n^{(r)}(x) = E_{(0)}(x) + E_{(1)}(x) + \cdots + E_{(r)}(x),\ \textit{kde} \tag{5.3a}$$

$$\begin{aligned} E_{(j)}(x) &= \frac{r!}{(r-j)!} \omega_{n+1}^{(r-j)}(x) \frac{f^{(n+j+1)}(\xi_j(x))}{(n+j+1)!},\ \xi_j \in \mathcal{I}[x_0, x_1, \dots, x_n, x],\ j = 0, 1, \dots, r,\quad a \\ E_{(r)}(x) &= 0 \textit{ pro } x \in \{x_0, x_1, \dots, x_n\},\ \textit{kdy stačí předpokládat pouze } f \in C^{n+r}[a,b]^{*)}. \end{aligned} \tag{5.3b}$$

V případě ekvidistantní sítě $x_i = x_0 + ih$, $h > 0$, $t = \frac{x - x_0}{h}$, je $E_n^{(r)}(x) = O(h^{n+1-r})$ a vidíme jak přesnost klesá s řádem derivace. Vztah (5.3b) *lze v tomto případě psát ve tvaru*

$$\begin{aligned} E_{(j)}(t) &= r! h^{n+1-r} \frac{\Pi_{n+1}^{(r-j)}(t)}{(r-j)!} \frac{f^{(n+j+1)}(\xi_j(x))}{(n+j+1)!},\ \xi_j \in \mathcal{I}[x_0, x_1, \dots, x_n, x],\ j = 0, 1, \dots, r,\quad a \\ E_{(r)}(t) &= 0 \textit{ pro } t \in \{0, 1, \dots, n\},\ \textit{kdy stačí předpokládat pouze } f \in C^{n+r}[a,b]^{*)}. \end{aligned} \tag{5.3c}$$

Užijeme-li pro $Q_n(x)$ Lagrangeova tvaru interpolačního polynomu pro obecné uzly (viz (4.3a) a (4.3b)), resp. pro ekvidistantní uzly (viz (4.4a) a (4.4b)), obdržíme následující výsledné tvary formulí pro výpočet r-té derivace ($r \geq 1$) funkce f.

Věta 5.3. *Nechť $x_0, x_1, \dots, x_n \in [a,b]$, $n \geq 0$, jsou uzlové body funkce f a $x \in [a,b]$ je libovolné. Jestliže označíme* $r' = \begin{cases} r-1 & pro \quad x \in \{x_0, x_1, \dots, x_n\} \\ r & pro \quad x \notin \{x_0, x_1, \dots, x_n\} \end{cases}$, *kde $r \geq 1$ udává řád derivace a $f \in C^{n+r'+1}[a,b]$, pak platí*

a) pro obecné uzly x_i:

$$f^{(r)}(x) = \sum_{i=0}^{n} f(x_i) L_{n,i}^{(r)}(x) + r! \sum_{j=0}^{r'} \frac{\omega_{n+1}^{(r-j)}(x)}{(r-j)!} \frac{f^{(n+j+1)}(\xi_j)}{(n+j+1)!}, \tag{5.4a}$$
$$\xi_j \in \mathcal{I}[x_0, x_1, \dots, x_n, x].$$

b) pro ekvidistantní uzly $x_i = x_0 + ih,\ h > 0$:

$$f^{(r)}(x) = \frac{1}{h^r} \sum_{i=0}^{n} f(x_i) D_i^{(r)}(t) + r!\, h^{n+1-r} \sum_{j=0}^{r'} \frac{\Pi_{n+1}^{(r-j)}(t)}{(r-j)!} \frac{f^{(n+j+1)}(\xi_j)}{(n+j+1)!}, \tag{5.4b}$$
$$\xi_j \in \mathcal{I}[x_0, x_1, \dots, x_n, x],$$

kde $D_i(t) = L_{n,i}(x_0 + th)$ a tedy

$$D_i^{(r)}(t) = \frac{(-1)^{n-i}}{i!\,(n-i)!} \left(\frac{d^r}{dt^r} \prod_{\substack{j=0 \\ j \neq i}}^{n} (t-j) \right),\ t = \frac{x - x_0}{h},\ \textit{tj. zejména } x = x_k \ \Rightarrow\ t = k. \tag{5.4c}$$

Důsledek 5.4. *Speciálně pro $r = 1$ a $f \in \begin{cases} C^{n+1}[a,b] & pro\ x \in \{x_0, x_1, \dots, x_n\} \\ C^{n+2}[a,b] & pro\ x \notin \{x_0, x_1, \dots, x_n\} \end{cases}$ dostáváme*

a) pro obecné uzly x_i:

$$f'(x) = \sum_{i=0}^{n} f(x_i) L'_{n,i}(x) + \omega'_{n+1}(x) \frac{f^{(n+1)}(\xi_0)}{(n+1)!} + \underbrace{\omega_{n+1}(x) \frac{f^{(n+2)}(\xi_1)}{(n+2)!}}_{E_{(1)}(x)}, \tag{5.5a}$$
$$\xi_0, \xi_1 \in \mathcal{I}[x_0, x_1, \dots, x_n, x] \text{ a } E_{(1)}(x) = 0 \text{ pro } x \in \{x_0, \dots, x_n\}.$$

b) pro ekvidistantní uzly $x_i = x_0 + ih,\ h > 0$:

$$f'(x) = \frac{1}{h} \sum_{i=0}^{n} f(x_i) D'_i(t) + h^n \left(\Pi'_{n+1}(t) \frac{f^{(n+1)}(\xi_0)}{(n+1)!} + \underbrace{\Pi_{n+1}(t) \frac{f^{(n+2)}(\xi_1)}{(n+2)!}}_{E_{(1)}(t)} \right), \tag{5.5b}$$
$$\xi_0, \xi_1 \in \mathcal{I}[x_0, x_1, \dots, x_n, x] \text{ a } E_{(1)}(t) = 0 \text{ pro } t \in \{0, \dots, n\},$$

kde

$$D'_i(t) = \frac{(-1)^{n-i}}{i!\,(n-i)!} \left(\prod_{\substack{j=0 \\ j \neq i}}^{n} (t-j) \right)',\ t = \frac{x - x_0}{h},\ \textit{tj. zejména } x = x_k \ \Rightarrow\ t = k. \tag{5.5c}$$

<u>*Poznámka*</u> 5.5 (Speciální případy).
V praxi se často setkáváme s ekvidistantní sítí uzlů a úlohou počítat derivace v těchto uzlech. Z hlediska přesnosti je ovšem výhodnější volba Čebyševových uzlů (Věta 4.5), neboť výskyt Rungeho jevu může zcela znehodnotit odhady derivací v důsledku oscilačního průběhu $Q_n(x)$.

Pro ekvidistantní uzly můžeme ze vztahů (5.5b) a (5.5c) odvodit následující nejčastěji používané formule pro 1. derivaci (značíme $f_i = f(x_i),\ f'_i = f'(x_i)$):

(1) <u>2-bodová formule ($n = 1,\ f \in C^2[a,b],\ E_1 = O(h)$)</u>

$$D'_0(t) = \frac{(-1)^1}{0!\,1!}(t-1)' = -1$$
$$D'_1(t) = \frac{(-1)^0}{1!\,0!}(t-0)' = 1$$
$$\Pi'_2(t) = \big[t(t-1)\big]' = 2t - 1 \ \Rightarrow\ \Pi'_2(0) = -1,\ \Pi'_2(1) = 1.$$

Odtud

$$\begin{aligned} f_0' =& \frac{1}{h}\,(f_1 - f_0) - \frac{h}{2} f''(\xi_0), \\ f_1' =& \frac{1}{h}\,(f_1 - f_0) + \frac{h}{2} f''(\xi_1), \end{aligned} \qquad \xi_0, \xi_1 \in \mathcal{I}[x_0, x_1]. \tag{5.6}$$

(2) 3-bodová formule ($n = 2,\ f \in C^3[a,b],\ E_2 = O(h^2)$)

$$\begin{aligned} D_0'(t) =& \frac{(-1)^2}{0!\,2!}\big[(t-1)(t-2)\big]' = \frac{1}{2}[t^2 - 3t + 2]' = \frac{1}{2}(2t-3). \\ D_1'(t) =& \frac{(-1)^1}{1!\,1!}\big[t(t-2)\big]' = -2(t-1). \\ D_2'(t) =& \frac{(-1)^0}{2!\,0!} = \big[t(t-1)\big]' = \frac{1}{2}(2t-1). \\ \Pi_3'(t) =& \big[t(t-1)(t-2)\big]' = \big[t(t^2-3t+2)\big]' = 3t^2 - 6t + 2 \\ & \Downarrow \\ & \Pi_3'(0) = 2,\ \Pi_3'(1) = -1,\ \Pi_3'(2) = 2. \end{aligned}$$

Odtud

$$\begin{aligned} f_0' =& \frac{1}{h}\left(-\frac{3}{2} f_0 + 2f_1 - \frac{1}{2} f_2\right) + \frac{h^2}{3} f'''(\xi_0), \\ f_1' =& \frac{1}{h}\left(-\frac{1}{2} f_0 + \frac{1}{2} f_2\right) - \frac{h^2}{6} f'''(\xi_1), \\ f_2' =& \frac{1}{h}\left(\frac{1}{2} f_0 - 2f_1 + \frac{3}{2} f_2\right) + \frac{h^2}{3} f'''(\xi_2), \end{aligned} \qquad \xi_0, \xi_1, \xi_2 \in \mathcal{I}[x_0, x_1, x_2]. \tag{5.7}$$

Podobným způsobem lze konstruovat $(n+1)$-bodové formule pro libovolné $n > 2$. Jsou-li dány hodnoty (x_i, f_i) pro $i = 0, 1, \ldots, n$, kde n je velké, není nutné (a ani vhodné) používat $(n+1)$-bodovou formuli pro výpočet $f_0', f_1', \ldots, f_n'$. Stačí totiž provádět postupně pouze **lokální interpolaci** dat v okolí x_i pro $i = 0, 1, \ldots, n$ polynomem Q_m nižšího stupně $m \le n$, takže například pro $m = 1$ můžeme (5.6) přepsat do přeindexovaného tvaru:

$$f_i' = \frac{1}{h}\,(f_{i+1} - f_i) - \frac{h}{2} f''(\xi_i), \quad \xi_i \in [x_i, x_{i+1}]. \tag{5.6.1}$$

$$f_i' = \frac{1}{h}\,(f_i - f_{i-1}) + \frac{h}{2} f''(\xi_i), \quad \xi_i \in [x_{i-1}, x_i]. \tag{5.6.2}$$

Podobně pro $m = 2$ přepíšeme (5.7) do tvaru:

$$f_i' = \frac{1}{2h}\,(-3f_i + 4f_{i+1} - f_{i+2}) + \frac{h^2}{3} f'''(\xi_i), \quad \xi_i \in [x_i, x_{i+2}] \tag{5.7.1}$$

$$f_i' = \frac{1}{2h}\,(f_{i+1} - f_{i-1}) - \frac{h^2}{6} f'''(\xi_i), \qquad \xi_i \in [x_{i-1}, x_{i+1}] \tag{5.7.2}$$

$$f_i' = \frac{1}{2h}\,(f_{i-2} - 4f_{i-1} + 3f_i) + \frac{h^2}{3} f'''(\xi_i), \qquad \xi_i \in [x_{i-2}, x_i] \tag{5.7.3}$$

Způsob použití formulí:

(5.7.1) pro $i = 0$ ... tj. na začátku intervalu (tzv. levostranná formule)
(5.7.2) pro $0 < i < n$ ... tj. uvnitř intervalu (tzv. centrální formule)
(5.7.3) pro $i = n$ ... tj. na konci intervalu (tzv. pravostranná formule)

Analogicky postupujeme v případě formulí s lichým počtem bodů (tj. $m = 2q > 2$ sudé), kdy dostaneme jednu centrální formuli, q levostranných a q pravostranných formulí.

Všimněte si, že **velikost chyby centrální formule** (5.7.2) **je poloviční** ve srovnání s (5.7.1) a (5.7.3). Je logické, že centrální formule dávají nejmenší chybu, neboť využívají stejný počet hodnot z obou stran k x_i, které jsou nejblíž k f_i a tudíž nejpřesněji popisují chování $f(x)$ v okolí x_i.

Všimněte si dále rovněž **symetričnosti formulí** (5.7.1) **a** (5.7.3). Formule (5.7.3) se dostane z (5.7.1) inverzním uspořádáním uzlů: $f(x_{i-j}) = f(x_i + j(-h))$ a záměnou h za $-h$, která odpovídá obrácení znaménka derivace při záměně nezávisle proměnné t za $-t$.

Podobné vlastnosti vykazují i formule s vyšším lichým počtem bodů (5-bodové, 7-bodové atd.), které se nejčastěji používají vzhledem k menší chybě centrální formule. Necentrální formule se aplikují pouze na okrajích intervalu.

Poznámka 5.6 (Problémy s přesností u formulí pro numerické derivování).
Jestliže jsou hodnoty f_i dány s chybou $f_i = \tilde{f}_i + E_i$, $|E_i| \le \varepsilon_i$, může tato okolnost podstatně ovlivnit kvalitu výsledného odhadu f_i'. Ukážeme to na příkladě formule (5.7.2).

Jelikož chyba ε_i a chyba formule se sčítají (viz 1.33), lze celkovou chybu T shora odhadnout takto:

$$\begin{aligned}|T| &= |f_i' - \frac{1}{2h}(\tilde{f}_{i+1} - \tilde{f}_{i-1})| = |f_i' - \frac{1}{2h}(f_{i+1} - f_{i-1}) + \frac{1}{2h}(E_{i+1} - E_{i-1})| \\ &\le \frac{h^2}{6}|f'''(\xi_i)| + \frac{1}{2h}(\varepsilon_{i+1} + \varepsilon_{i-1}).\end{aligned}$$

Položíme-li $\varepsilon = \max(\varepsilon_{i-1}, \varepsilon_{i+1})$ a $M_3 = \max_{x\in[x_{i-1},x_{i+1}]}|f'''(x)|$, pak

$$\boxed{|T| \le \frac{\varepsilon}{h} + \frac{h^2}{6}M_3}$$

První člen chyby $\frac{\varepsilon}{h}$ (např. ε může být **chyba způsobená zaokrouhlováním, měřením apod.**) závisí **nepřímo úměrně na** h, zatímco druhý člen (**chyba metody**) závisí **přímo úměrně na** h. Vzniká otázka, jak optimálně volit h, aby celková chyba byla minimální.

1. **Strategie:** hledáme minimum funkce $g(h) = \frac{\varepsilon}{h} + \frac{h^2}{6}M_3$. Ze vztahu $g'(h) = -\frac{\varepsilon}{h^2} + \frac{h}{3}M_3 = 0$ dostáváme $\boxed{h_{\text{opt}} = \sqrt[3]{\frac{3\varepsilon}{M_3}}}$.
2. **Strategie:** hledáme h tak, aby příspěvek obou členů chyby byl stejný. Ze vztahu $\frac{\varepsilon}{h} = \frac{h^2}{6}M_3$ dostáváme $\boxed{h_{\text{opt}} = \sqrt[3]{\frac{6\varepsilon}{M_3}}}$.

V případě, že hodnoty f_i jsou dány s malou přesností (např. byly získány empiricky), není vhodné použít formule pro numerické derivování přímo, neboť hrozí riziko výrazného zkreslení výsledků. V takových případech je lépe nejdříve naměřené hodnoty "vyhladit" například metodou nejmenších čtverců a pak teprve použít formule pro numerické derivování.

Algoritmus (MATLAB)**.**
`jmathews('chap_6'): difflim.m` *(+ demo* `a6_1.m`*)*
V tomto algoritmu je snaha nalézt "nejlepší" aproximaci $f'(x)$ zvolením optimálního kroku h pro formuli (5.7.2) *jeho postupným zjemňováním metodou půlení:*

$$f'(x) \approx D_k(x) := \frac{f\left(x + \frac{h}{2^k}\right) - f\left(x - \frac{h}{2^k}\right)}{h/2^{k-1}} \quad \text{pro } k = 0, 1, \dots$$

až do dosažení ukončující podmínky:

$$|D_{k+1} - D_k| \ge |D_k - D_{k-1}| \quad \dots \text{odhad chyby začíná vzrůstat,}$$

nebo

$$|D_k - D_{k-1}| < \text{Tolerance} \qquad \dots \text{odhad chyby dosáhl požadované přesnosti.}$$

6. Numerické integrování

(Algoritmy v MATLABu: `jmathews('chap_7')`)

Definice 6.1. Nechť $-\infty \le a \le x_0 < x_1 < \cdots < x_n \le b \le +\infty$ a $W(x)f(x)$ je funkce integrovatelná na intervalu $[a,b]$, kde $W(x)$ je předem daná, tzv. **váhová funkce**, a $f(x)$ je funkce zadaná v uzlových bodech $x_0, x_1, \ldots, x_n$ tabulkou funkčních hodnot (případně i derivacemi některých řádů). **Kvadraturní formulí** nazýváme formuli pro přibližný výpočet integrálu $\int_a^b W(x)f(x)\,dx$, závisející pouze na $W(x)$, uzlech $x_0, x_1, \ldots, x_n$ a zadaných tabulkových hodnotách.

Obecný tvar kvadraturní formule závisející lineárně na tabulkových hodnotách $f(x_i),\ i = 0, 1, \ldots, n$:

$$\boxed{\int_a^b W(x)f(x)\,dx = \sum_{i=0}^{n} A_i f(x_i) + E_f}, \tag{6.1}$$

kde je
E_f ... **chyba kvadraturní formule**,
a pro $i = 0, 1, \ldots, n$ jsou
x_i ... **kvadraturní uzly**
A_i ... **váhové koeficienty (váhy)** závislé pouze na $W(x)$ a x_i } **parametry formule.**

Definice 6.2. Kvadraturní formule (6.1) se nazývá

- **uzavřená**, jestliže $a = x_0$ a $x_n = b$,
- **otevřená**, jestliže $a < x_0$ a $x_n < b$, tj. formule nezávisí na hodnotách funkce v krajních bodech intervalu.
- **polouzavřená**, resp. **polootevřená**, jestliže
  (1) $a = x_0$ a $x_n < b$, tj. je uzavřená zleva a otevřená zprava, nebo
  (2) $a < x_0$ a $x_n = b$, tj. je otevřená zleva a uzavřená zprava.

Definice 6.3.
Celé číslo $\nu \ge 0$ nazveme **stupněm přesnosti** kvadraturní formule (6.1), jestliže $E_P = 0$ pro všechny polynomy $P(x)$ stupně nejvýše ν (tj. $P \in \mathcal{P}_\nu$) a $E_P \neq 0$ pro některý polynom $P(x)$ stupně $\nu + 1$.

Poznámka 6.4.
Zřejmě pro libovolná $\alpha, \beta \in \mathbb{R}$ a funkce $f(x), g(x)$ splňující předpoklady 6.1 je $E_{\alpha f+\beta g} = \alpha E_f + \beta E_g$, neboť dle (6.1) je $E_f = \int_a^b W(x)f(x)\,dx - \sum_{i=0}^n A_i f(x_i)$, kde výraz vpravo závisí lineárně na f.

Odtud ihned vidíme, že v případě formule (6.1) se stupněm přesnosti ν je $E_p \neq 0$ pro **každý** polynom $P(x) =: a_0 + a_1 x + \ldots a_\nu x^\nu + a_{\nu+1}x^{\nu+1}$ stupně $\nu + 1$ ($a_{\nu+1} \neq 0$), neboť

$$E_P = a_0 \underbrace{E_1}_{=0} + a_1 \underbrace{E_x}_{=0} + \cdots + a_\nu \underbrace{E_{x^\nu}}_{=0} + \underbrace{a_{\nu+1}}_{\neq 0} E_{x^{\nu+1}} \neq 0 \quad \Leftrightarrow \quad E_{x^{\nu+1}} \neq 0.$$

Protože $P^{(\nu+1)}(\xi) \equiv (\nu+1)!\,a_{\nu+1}$, dostáváme odtud ihned vyjádření pro chybu ve tvaru:

$$E_P = \underbrace{\frac{E_{x^{\nu+1}}}{(\nu+1)!}}_{=:e_\nu} P^{(\nu+1)}(\xi) =: e_\nu P^{(\nu+1)}(\xi),\ \xi \in \mathbb{R}.$$

Dá se tedy očekávat, že E_f bude mít výše uvedený tvar pro každou funkci $f \in C^{\nu+1}[a,b]$ a vhodné $\xi \in [a,b]$ (viz dále).

V dalším se omezíme na ohraničený interval $[a,b]$, tj. $-\infty < a < b < +\infty$.

Věta 6.5. *Je-li $W(x)$ integrovatelná na $[a,b]$ a $f \in C^{n+1}[a,b]$, potom pro formuli* (6.1) *se stupněm přesnosti $\nu \ge n$ platí*

$$A_i = \int_a^b W(x)L_{n,i}(x)\,dx \text{ pro } i = 0, 1, \ldots, n, \tag{6.1a}$$

$$E_f = \frac{1}{(n+1)!}\int_a^b W(x)\omega_{n+1}(x)f^{(n+1)}(\xi(x))\,dx,\ \xi(x) \in [a,b]. \tag{6.1b}$$

Poznámka 6.6.

(1) Pro $\nu \geq n$ je tedy (6.1) jednoznačně určena vztahy (6.1a) a (6.1b) a dostane se formálním zintegrováním Lagrangeova interpolačního vztahu (4.3b)

$$f(x) = \sum_{i=0}^{n} L_{n,i}(x) f(x_i) + \underbrace{\omega_{n+1}(x) \frac{f^{(n+1)}(\xi(x))}{(n+1)!}}_{E_n(x)},$$

s využitím vyjádření (4.1a) pro chybu $E_n(x)$.

(2) **Metoda neurčitých koeficientů pro určení A_i při daných uzlech x_i, $i = 0, 1, \ldots, n$:** Nechť $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$ je vhodná polynomická báze v $\mathcal{P}_n$ (například $\varphi_j(x) = x^j$). Z podmínky, aby (6.1) byla přesná pro všechny polynomy $\varphi_j(x)$, $0 \leq j \leq n$, dostáváme systém $n+1$ lineárních rovnic pro $n+1$ neznámých A_i:

$$\sum_{i=0}^{n} A_i \varphi_j(x_i) = \int_a^b W(x) \varphi_j(x)\, dx \text{ pro } j = 0, 1, \ldots, n, \tag{6.1c}$$

Protože $\varphi_j(x)$ jsou bázové polynomy stupně nejvýše n bude vzhledem k linearitě chyby 6.4 formule přesná nejen pro ně, ale pro všechny polynomy stupně nejvýše n a bude tedy formulí dosahující přesnosti alespoň n. V případě ekvidistantní sítě obdržíme tzv. **Newton-Cotesovy** formule, které budou dále podrobně popsány v odstavci 6.1.

(3) Ve (2) byly uzly x_i předepsány a volnými parametry byly pouze koeficienty A_i, $i = 0, 1, \ldots, n$. Všech parametrů je tedy celkem $2n+2$. Zvolíme-li obecně k volných parametrů ($1 \leq k \leq 2n+2$), je snaha sestavit k rovnic typu (6.1c). Pokud tento systém je řešitelný, obdržíme analogicky formuli přesnosti alespoň $k-1$. Poznamenejme ovšem, že tento systém je lineární pouze v koeficientech A_i, nikoliv ale v uzlech x_i, což komplikuje nalezení řešení.

Ponecháme-li všechny parametry volné ($k = 2n+2$), pak lze pro několik typů váhových funkcí $W(x)$ tímto postupem zkonstruovat tzv. **Gaussovy kvadraturní formule**, které dosahují **maximálního stupně přesnosti** $\nu = 2n+1$.

(4) **Obecný postup konstrukce kvadraturních formulí** lze vzhledem ke (3) formulovat takto:
- Zvolíme vhodnou váhovou funkci $W(x)$ a zadáme r ($0 \leq r < 2n+2$) vhodných omezujících podmínek pro parametry (například ve (2) bylo předepsáno $r = n+1$ uzlů).
- Sestrojíme $2n+2$ rovnic tvořených r předepsanými podmínkami doplněnými o $k = 2n+2-r$ podmínek typu (6.1c).
- Vyřešením tohoto systému nalezneme všech $2n+2$ parametrů (váhy + uzly) a obdržíme formuli se stupněm přesnosti alespoň $k-1$ ($\nu \geq 2n+1-r$).

6.1. Newton-Cotesovy kvadraturní formule.

Definice 6.7.

(1) Řekneme, že uzly $x_0, x_1, \ldots, x_n$ jsou **symetrické na intervalu** $[a, b]$, jestliže jsou rozmístěny symetricky vzhledem ke středu $s = \frac{a+b}{2}$ intervalu, tj. $s - x_i = x_{n-i} - s =: h_i$ pro $i = 0, 1, \ldots, n$.

(2) Je-li n celé číslo, pak $\mathrm{par}(n) := \begin{cases} 1 & \text{pro } n \text{ sudé} \\ -1 & \text{pro } n \text{ liché} \end{cases}$ udává jeho **paritu**.

(3) Je-li funkce $g(x)$ definována na $[a, b]$, pak řekneme, že je **sudá**, resp. **lichá** na $[a, b]$, jestliže $g(s+t)$, $t \in [-(b-a)/2, (b-a)/2]$, je sudá, resp. lichá; neboli platí $g(s-t) = g(s+t)$, resp. $g(s-t) = -g(s+t)$.
V takovém případě píšeme $\mathrm{par}_{a,b}(g) = 1$, resp. $= -1$. Pokud g není ani sudá, ani lichá, klademe $\mathrm{par}_{a,b}(g) = 0$.

Lemma 6.8.
Nechť $x_0, x_1, \ldots, x_n$ jsou symetrické uzly na $[a, b]$. Potom $\mathrm{par}_{a,b}(\omega_{n+1}) = \mathrm{par}(n+1) = -\mathrm{par}(n)$, neboli $\omega_{n+1}(x)$ je sudá, resp. lichá funkce na $[a, b]$ v závislosti na tom, zda počet uzlů je sudý, resp. lichý.

Věta 6.9. *Nechť $x_0, x_1, \ldots, x_n$ jsou symetrické uzly na $[a, b]$ a $W(x)$ je integrovatelná sudá nebo lichá funkce na $[a, b]$, potom pro koeficienty A_i určené vztahem* (6.1a) *platí symetrie*

$$A_{n-i} = \mathrm{par}_{a,b}(W)\, A_i \text{ pro } i = 0, 1, \ldots, n. \tag{6.2}$$

Věta 6.10.
Nechť $x_0, x_1, \dots, x_n$ jsou symetrické uzly na $[a,b]$, $W(x)$ spojitá na $[a,b]$, $\mathrm{par}_{a,b}(W) = -\mathrm{par}(n+1)$ a $f \in C^{n+2}[a,b]$. Potom chybu formule určenou vztahem (6.1b) *lze vyjádřit ve tvaru*

$$E_f = -\int_a^b q(x)\left(\frac{f^{(n+1)}(\xi(x))}{(n+1)!}\right)' dx = -\int_a^b q(x)\frac{f^{(n+2)}(\eta(x))}{(n+2)!}\,dx, \tag{6.3}$$

kde $q(x) := \int_a^x W(u)\omega_{n+1}(u)\,du$ a $\xi(x), \eta(x) \in [a,b]$, přičemž $f^{(n+2)}(\eta(x)) \in C[a,b]$.

V dalším se již pro jednoduchost omezíme pouze na případ jednotkové váhové funkce $W(x) \equiv 1$.

Lemma 6.11. *Nechť $a = x_{-1} < x_0 < \cdots < x_{n+1} = b$ a $x_i = x_0 + ih$, $h > 0$ $(i = 0, 1, \dots, n)$ jsou ekvidistantní uzly na intervalu $[a,b]$. Označíme-li $I_j := \int_{x_j}^{x_{j+1}} \omega_{n+1}(x)\,dx$ pro $j = -1, 0, 1, \dots, n$, pak platí*

1. $I_j = \mathrm{par}(n+1) I_{n-1-j}$ *pro* $j = -1, 0, 1, \dots, n$.
2. $I_{j-1} = \frac{(\xi_j - x_{n+1})}{(\xi_j - x_0)} I_j$ *pro vhodné* ξ_j, $x_j < \xi_j < x_{j+1}$, $0 \le j < \frac{n}{2}$.
3. $|I_{j-1}| > |I_j|$ *pro* $0 \le j < \frac{n}{2}$.
4. $q_j(x) := \int_{x_j}^x \omega_{n+1}(u)\,du$ *nemění znaménko na* $[x_j, x_{n-j}]$, *pro n sudé,* $-1 \le j < \frac{n}{2}$. *Je-li n liché,* $-1 \le j < \frac{n-1}{2}$, *pak $q_j(x)$ nemění znaménko na* $[x_j, x_{n-1-j}]$.

Důkaz. Viz A. Ralston: Základy numerické matematiky, Academia Praha, 1973, str. 178, cv. 49. □

Věta 6.12 (NEWTON-COTESOVY KVADRATURNÍ FORMULE)**.**
Nechť $x_0, x_1, \dots, x_n$ jsou ekvidistantní uzly, $x_i = x_0 + ih$, $h > 0$ a $a = x_0 - \alpha h$, $b = x_n + \alpha h$, kde $\alpha = 0$ (v případě $(n+1)$-bodové **uzavřené formule***) nebo $\alpha = 1$ (v případě $(n+1)$-bodové* **otevřené formule***). Potom existuje $\xi \in [a,b]$ takové, že platí*

$$\int_a^b f(x)\,dx = \sum_{i=0}^n A_i f(x_i) + \underbrace{h^{n+3}\frac{f^{(n+2)}(\xi)}{(n+2)!}\int_{-\alpha}^{n+\alpha} t\,\Pi_{n+1}(t)\,dt}_{E_f} \tag{6.4a}$$

pro n sudé a $f \in C^{n+2}[a,b]$

a

$$\int_a^b f(x)\,dx = \sum_{i=0}^n A_i f(x_i) + \underbrace{h^{n+2}\frac{f^{(n+1)}(\xi)}{(n+1)!}\int_{-\alpha}^{n+\alpha} \Pi_{n+1}(t)\,dt}_{E_f} \tag{6.4b}$$

pro n liché a $f \in C^{n+1}[a,b]$,

kde

$$A_i = h\frac{(-1)^{n-i}}{i!(n-i)!}\int_{-\alpha}^{n+\alpha}\frac{\Pi_{n+1}(t)}{t-i}\,dt \tag{6.4c}$$

jsou tzv. **Cotesova čísla**, *pro něž platí $A_{n-i} = A_i$, $i = 0, 1, \dots, n$.*

<u>*Poznámka*</u> 6.13.

- n sudé $\overset{(6.4a)}{\Longrightarrow}$
$$\begin{cases} \nu &= n+1 \dots \text{ stupeň přesnosti,} \\ E_f &= h^{\nu+2} e_n f^{(\nu+1)}(\xi) = O(h^{\nu+2}) = O(h^{n+3}) \text{ pro } h \to 0, \\ e_n &= \frac{1}{(n+2)!}\int_{-\alpha}^{n+\alpha} t^2(t-1)\dots(t-n)\,dt. \end{cases}$$
- n liché $\overset{(6.4b)}{\Longrightarrow}$
$$\begin{cases} \nu &= n \dots \text{ stupeň přesnosti,} \\ E_f &= h^{\nu+2} e_n f^{(\nu+1)}(\xi) = O(h^{\nu+2}) = O(h^{n+2}) \text{ pro } h \to 0, \\ e_n &= \frac{1}{(n+1)!}\int_{-\alpha}^{n+\alpha} t(t-1)\dots(t-n)\,dt. \end{cases}$$

Vidíme, že pro lichý počet uzlů (n sudé) je řád konvergence i stupeň přesnosti Newton-Cotesových formulí o 1 vyšší, než pro sudý počet uzlů (n liché).

<u>*Poznámka*</u> 6.14 (Speciální případy Newton-Cotesových formulí pro $W(x) \equiv 1$).

Označíme-li $A_i = hAa_i$, kde $a_i = a_{n-i}$, dostaneme následující tabulky popisující

a) Uzavřené ($\alpha = 0$) Newton-Cotesovy formule pro $1 \le n \le 8$

Název formule	n	A	a_0	a_1	a_2	a_3	a_4	ν	e_n
Lichoběžníkové pravidlo	1	1/2	1	1				1	−1/12
Simpsonovo pravidlo	2	1/3	1	4	1			3	−1/90
Simpsonovo 3/8 pravidlo	3	3/8	1	3	3	1		3	−3/80
Booleovo pravidlo	4	2/45	7	32	12	32	7	5	−8/945
	5	5/288	19	75	50	50	75	5	−275/12096
	6	1/140	41	216	27	272	27	7	−9/1400
	7	7/17280	751	3577	1323	2989	2989	7	−8183/518400
	8	4/14175	989	5888	−928	10946	−4540	9	−2368/467775

Pomocí obecných vzorců z věty 6.12, resp. z poznámky 6.13, odvodíme jako ukázku koeficienty pro prvé dvě uzavřené kvadraturní formule z předchozí tabulky.

(1) Lichoběžníkové pravidlo (2-bodová uzavřená formule): $\alpha = 0,\ n = 1,\ f \in C^2[a,b]$

Podle 6.13 je $\nu = n = 1$ a tedy $E_f = O(h^3)$. Odtud a z (6.4c) dostáváme:

$$A_0 = h\frac{(-1)^{1-0}}{0!\,(1-0)!}\int_0^1 (t-1)\,dt = -h\Big[\frac{(t-1)^2}{2}\Big]_0^1 = -h\Big(0-\frac{1}{2}\Big) = h\frac{1}{2}.$$
$$A_1 = A_{1-1} = A_0 = h\frac{1}{2} \ \Rightarrow\ A = \frac{1}{2},\ a_0 = a_1 = 1.$$
$$e_1 = \frac{1}{(1+1)!}\int_0^1 t(t-1)\,dt = \frac{1}{2}\Big[\frac{t^3}{3}-\frac{t^2}{2}\Big]_0^1 = \frac{1}{2}\Big(\frac{1}{3}-\frac{1}{2}\Big) = -\frac{1}{12}.$$

(2) Simpsonovo pravidlo (3-bodová uzavřená formule): $\alpha = 0,\ n = 2,\ f \in C^4[a,b]$

Podle 6.13 je $\nu = n+1 = 3$ a tedy $E_f = O(h^5)$. Odtud a z (6.4c) dostáváme:

$$A_0 = h\frac{(-1)^{2-0}}{0!\,(2-0)!}\int_0^2 \underbrace{(t-1)(t-2)}_{t^3-3t+2}\,dt = h\frac{1}{2}\Big[\frac{t^3}{3}-\frac{3t^2}{2}+2t\Big]_0^2 = h\frac{1}{2}\Big(\frac{8}{3}-\frac{12}{2}+4\Big) = h\frac{1}{3}.$$
$$A_1 = h\frac{(-1)^{2-1}}{1!\,(2-1)!}\int_0^2 t(t-2)\,dt = -h\Big[\frac{t^3}{3}-t^2\Big]_0^2 = -h\Big(\frac{8}{3}-4\Big) = h\frac{4}{3}.$$
$$A_2 = A_0 = h\frac{1}{3},\ A_1 = h\frac{4}{3} \ \Rightarrow\ A = \frac{1}{3},\ a_0 = a_2 = 1, a_1 = 4.$$
$$e_2 = \frac{1}{(2+2)!}\int_0^2 t^2(t-1)(t-2)\,dt = \frac{1}{24}\Big[\frac{t^5}{5}-\frac{3t^4}{4}+\frac{2t^3}{3}\Big]_0^2 = \frac{1}{24}\Big(\frac{32}{5}-3\frac{16}{4}+2\frac{8}{3}\Big) =$$
$$= \Big(\frac{4}{15}-\frac{1}{2}+\frac{2}{9}\Big) = \frac{24-45+20}{90} = -\frac{1}{90}.$$

Analogicky bychom postupovali pro $n > 2$. Zejména tedy dostáváme následující tvary nejčastěji používaných uzavřených Newton-Cotesových kvadraturních formulí.

Lichoběžníkové pravidlo ($n = 1$):

$$\int_a^b f(x)\,dx = \frac{h}{2}(f_0+f_1) - \frac{h^3}{12}f''(\xi), \tag{6.5a}$$

Simpsonovo pravidlo ($n = 2$):

$$\int_a^b f(x)\,dx = \frac{h}{3}(f_0+4f_1+f_2) - \frac{h^5}{90}f^{(4)}(\xi), \tag{6.5b}$$

Simpsonovo 3/8 pravidlo ($n = 3$):

$$\int_a^b f(x)\,dx = \frac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3) - \frac{3h^5}{80}f^{(4)}(\xi), \tag{6.5c}$$

Booleovo pravidlo ($n = 4$):

$$\int_a^b f(x)\,dx = \frac{2h}{45}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4) - \frac{8h^7}{945}f^{(6)}(\xi) \tag{6.5d}$$

pro vhodné $\xi \in [a, b]$.

b) Otevřené ($\alpha = 1$) Newton-Cotesovy formule pro $0 \le n \le 4$

Název formule	n	A	a_0	a_1	a_2	ν	e_n
Obdélníkové pravidlo	0	2	1			1	1/3
	1	3/2	1	1		1	3/4
	2	4/3	2	−1	2	3	14/45
	3	5/24	11	1	1	3	95/144
	4	3/10	11	−14	26	5	41/140

Pomocí obecných vzorců z věty 6.12, resp. z poznámky 6.13, opět odvodíme jako ukázku koeficienty pro prvé dvě otevřené kvadraturní formule z předchozí tabulky.

(1) Obdélníkové pravidlo (1-bodová otevřená formule): $\alpha = 1,\ n = 0,\ f \in C^2[a, b]$

Podle 6.13 je $\nu = n + 1 = 1$ a tedy $E_f = O(h^3)$. Odtud a z (6.4c) dostáváme:

$$A_0 = h\frac{(-1)^{0-0}}{0!\,(0-0)!}\int_{-1}^{1} 1\,dt = h(1-(-1)) = h2 \ \Rightarrow\ A = 2,\ a_0 = 1.$$
$$e_0 = \frac{1}{(0+2)!}\int_{-1}^{1} t^2\,dt = \frac{1}{2}\Big[\frac{t^3}{3}\Big]_{-1}^{1} = \frac{1}{2}\Big(\frac{1}{3} + \frac{1}{3}\Big) = \frac{1}{3}.$$

(2) 2-bodová otevřená formule: $\alpha = 1,\ n = 1,\ f \in C^2[a, b]$

Podle 6.13 je $\nu = n = 1$ a tedy $E_f = O(h^3)$. Odtud a z (6.4c) dostáváme:

$$A_0 = h\frac{(-1)^{1-0}}{0!\,(1-0)!}\int_{-1}^{2}(t-1)\,dt = -h\Big[\frac{(t-1)^2}{2}\Big]_{-1}^{2} = -h\Big(\frac{1}{2} - \frac{4}{2}\Big) = h\frac{3}{2}.$$
$$A_1 = A_{1-1} = A_0 = h\frac{3}{2} \ \Rightarrow\ A = \frac{3}{2},\ a_0 = a_1 = 1.$$
$$e_1 = \frac{1}{(1+1)!}\int_{-1}^{2} t(t-1)\,dt = \frac{1}{2}\Big[\frac{t^3}{3} - \frac{t^2}{2}\Big]_{-1}^{2} = \frac{1}{2}\Big(\frac{8}{3} - \frac{4}{2} + \frac{1}{3} + \frac{1}{2}\Big)\Big) = \frac{1}{2}\Big(3 - \frac{3}{2}\Big) = \frac{3}{4}.$$

Analogicky bychom postupovali pro $n > 1$. Zejména tedy dostáváme následující tvary nejčastěji používaných otevřených Newton-Cotesových kvadraturních formulí.

Obdélníkové pravidlo ($n = 0$):

$$\int_a^b f(x)\,dx = 2hf_0 + \frac{h^3}{3}f''(\xi), \tag{6.6a}$$

$n = 1$**:**

$$\int_a^b f(x)\,dx = \frac{3h}{2}(f_0 + f_1) + \frac{3h^3}{4}f''(\xi), \tag{6.6b}$$

$n = 2$**:**

$$\int_a^b f(x)\,dx = \frac{4h}{3}(2f_0 - f_1 + 2f_2) + \frac{14h^5}{45}f^{(4)}(\xi), \tag{6.6c}$$

$n = 3$**:**

$$\int_a^b f(x)\,dx = \frac{5h}{24}(11f_0 + f_1 + f_2 + 11f_3) + \frac{95h^5}{144}f^{(5)}(\xi) \tag{6.6d}$$

pro vhodné $\xi \in [a, b]$.

Věta 6.15 (SLOŽENÉ NEWTON-COTESOVY KVADRATURNÍ FORMULE)**.** *Nechť* $x_0, x_1, \ldots, x_N$, $N = (n+2\alpha)M$ *(*$M \geq 1$, $n \geq 0$ *celá) jsou ekvidistantní uzly,* $x_j = x_0 + jh$, $h > 0$ *a* $a = x_0 - \alpha h$, $b = x_n + \alpha h$, *kde* $\alpha = 0$ *(v případě* **uzavřené formule***) nebo* $\alpha = 1$ *(v případě* **otevřené formule***). Potom existuje* $\xi \in [a, b]$ *takové, že platí*

$$\int_a^b f(x)\,dx = \underbrace{\sum_{j=0}^{N} A'_j f(x_j)}_{NC(f,h)} + \underbrace{h^{n+2}\frac{b-a}{n+2\alpha}e_n f^{(n+2)}(\xi)}_{E_{NC}(f,h)=O(h^{n+2})} \tag{6.7a}$$

pro n *sudé,* $f \in C^{n+2}[a, b]$ *a* e_n *z poznámky* 6.13

a

$$\int_a^b f(x)\,dx = \underbrace{\sum_{j=0}^{N} A'_j f(x_j)}_{NC(f,h)} + \underbrace{h^{n+1}\frac{b-a}{n+2\alpha}e_n f^{(n+1)}(\xi)}_{E_{NC}(f,h)=O(h^{n+1})} \tag{6.7b}$$

pro n *liché,* $f \in C^{n+1}[a, b]$ *a* e_n *z poznámky* 6.13,

kde

$$A'_j = \begin{cases} 2(1-\alpha)A_0 & \text{pro } j = m(n+2\alpha),\ 0 < m < M \text{ celé} \\ A_{(j \bmod (n+2\alpha))-\alpha} & \text{pro ostatní } j,\ 0 \leq j \leq N, \end{cases} \tag{6.7c}$$

přičemž $A_{-1} = 0$ *a* A_i *jsou určeny vztahem* (6.4c) *pro* $i = 0, 1, \ldots, n$.

Poznámka 6.16 (Speciální případy složených Newton-Cotesových formulí).

- **Uzavřené složené formule (**$\alpha = 0$**):**

  (1) Lichoběžníkové složené pravidlo (NC= T ... Trapezoidal, $n = 1$)

$$\left.\begin{array}{ccccccccc} \frac{1}{2} & \frac{1}{2} & & & & & & & \\ & \frac{1}{2} & \frac{1}{2} & \ldots & \frac{1}{2} & & & & \\ & & & \vdots & & & & & \\ & & & & \frac{1}{2} & \frac{1}{2} & & & \\ \hline A'_j: & \frac{1}{2} & 1 & 1 & \ldots & 1 & \frac{1}{2} & \times & h \end{array}\right\} \text{viz (6.5a) a (6.7c)}$$

$$\int_a^b f(x)\,dx = T(f, h) + E_T(f, h),$$

kde užitím (6.5a) a (6.7b)

$$T(f,h) = \frac{h}{2}(f_0 + 2f_1 + \cdots + 2f_{N-1} + f_N) = h\left[\frac{f(a)+f(b)}{2} + \sum_{j=1}^{N-1} f(x_j)\right],\ N = M, \tag{6.8a}$$

$$E_T(f,h) = -h^2\frac{b-a}{12}f''(\xi) = O(h^2),\ \xi \in [a, b].$$

Algoritmus (MATLAB)**.**
`jmathews('chap_7'): traprl.m` *(+ demo* `a7_1.m`*)*

(2) Simpsonovo složené pravidlo (NC= S, $n = 2$)

$$\begin{array}{ccccccccccc}
\frac{1}{3} & \frac{4}{3} & \frac{1}{3} & & & & & & & & \\
 & & \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & \dots & \frac{1}{3} & & & & \\
 & & & & & \vdots & & & & & \\
 & & & & & & \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & & \\
\hline
A'_j: & \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \dots & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & \times\ h
\end{array}$$ viz (6.5b) a (6.7c)

$$\int_a^b f(x)\,dx = S(f,h) + E_S(f,h),$$

kde užitím (6.5b) a (6.7a)

$$\begin{aligned}
S(f,h) &= \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{N-2} + 4f_{N-1} + f_N) = \\
&= \frac{h}{3}\left[f(a) + f(b) + 2\sum_{m=1}^{M-1} f(x_{2m}) + 4\sum_{m=1}^{M} f(x_{2m-1})\right],\ N = 2M, \\
E_S(f,h) &= h^4 \frac{b-a}{2} \underbrace{\left(-\frac{1}{90}\right)}_{e_2} f^{(4)}(\xi) = -h^4 \frac{b-a}{180} f^{(4)}(\xi) = O(h^4),\ \xi \in [a,b].
\end{aligned} \tag{6.8b}$$

Algoritmus (MATLAB)**.**
`jmathews('chap_7'): simprl.m` *(+ demo* `a7_2.m`*)*

- **Otevřené složené formule ($\alpha = 1$):**

Obdélníkové složené pravidlo (NC= R . . . Rectangular, $n = 0$)

$$\begin{array}{ccccccccccc}
0 & 2 & 0 & & & & & & & & \\
 & & 0 & 2 & 0 & \dots & 0 & & & & \\
 & & & & & \vdots & & & & & \\
 & & & & & & 0 & 2 & 0 & & \\
\hline
A'_j: & 0 & 2 & 0 & 2 & 0 & \dots & 0 & 2 & 0 & \times\ h
\end{array}$$ viz (6.6a) a (6.7c)

$$\int_a^b f(x)\,dx = R(f,h) + E_R(f,h),$$

kde užitím (6.6a) a (6.7a)

$$\begin{aligned}
R(f,h) &= 2h(f_1 + f_3 + \cdots + f_{N-1}) = 2h\sum_{m=1}^{M} f(x_{2m-1}),\ N = 2M, \\
E_R(f,h) &= h^{0+2}\frac{b-a}{0+2\cdot 1}\underbrace{\frac{1}{3}}_{e_0} f''(\xi) = h^2\frac{b-a}{6} f''(\xi) = O(h^2),\ \xi \in [a,b].
\end{aligned} \tag{6.9a}$$

Katedra aplikované matematiky, Masarykova univerzita v Brně, Janáčkovo nám. 2a, 662 95 Brno
E-mail address: vesely@@math.muni.cz