OBSAH

doc. Ing. Josef Daněk, Ph.D.

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Plzeň 2024

NELINEÁRNÍ ROVNICE

Formulace:

Je dána funkce $f : \mathbb{R} \to \mathbb{R}$ definovaná na intervalu $\langle a, b\rangle$. Hledáme číslo x z intervalu $\langle a, b\rangle$ tak, aby platila rovnost $f(x) = 0$. Číslo x nazveme **řešení** nebo **kořen rovnice**).

Poznámka:

Najít přesné řešení analyticky je možné jen ve velmi jednoduchých případech, např. při řešení lineární rovnice $12x - 3 = 0$, při řešení kvadratické rovnice $4x^2 - 5x + 8 = 0$ nebo např. při řešení rovnice $\sin 5x = \pi$. Proto je nutné pro nalezení kořenů použít nějakou numerickou metodu.

Numerické metody, kterými se budeme zabývat jsou založeny na **iteračních principech**. Pro každou iterační metodu nás budou zajímat odpovědi na dvě otázky:

- Konverguje posloupnost iterací ke hledanému kořenu?
- Jestliže ano, jak rychle?

Jestliže nemáme předběžné informace opoloze kořene, víme pouze, že leží v určitém intervalu $\langle a, b\rangle$, užijeme k výpočtu takové iterační metody, jejíž konvergence nezávisí na volbě počáteční aproximace. Tyto tzv. *vždy konvergentní metody* mají většinou tu nevýhodu, že konvergují pomalu, a hodí se tedy především k určení takové aproximace kořene, která může být použita jako počáteční aproximace pro nějakou rychleji konvergující metodu. Konvergence takové „lepší“ metody už může silně záviset na tom, jak dobrá je tato počáteční aproximace, a na vlastnostech funkce f v okolí kořene. Je tedy rozumné rozdělit metody řešení nelineárních rovnic na:

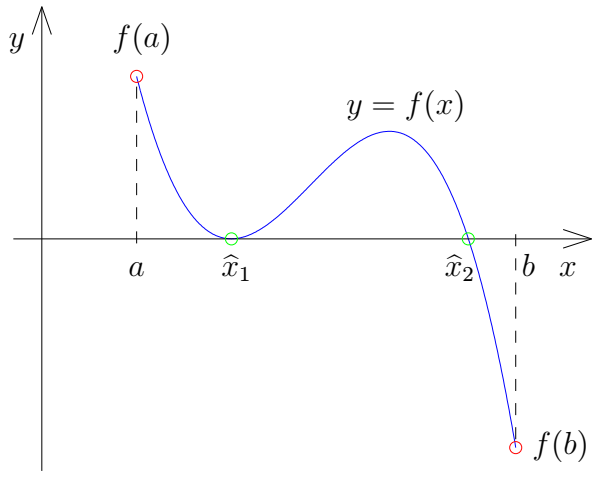
- startovací metody (vždy konvergentní metody)
- zpřesňující metody
- speciální metody (např. pro polymomy)

Tímto rozdělením ovšem nechceme zdůraznit, že startovací metoda konverguje pomalu vždy a naopak, že zpřesňující metoda konverguje vždy rychle.

Poznamenejme, že vždy budeme předpokládat, že daná funkce f je v uvažovaném intervalu $\langle a, b\rangle$ **spojitá** neboť tento předpoklad je důležitý v následující větě.

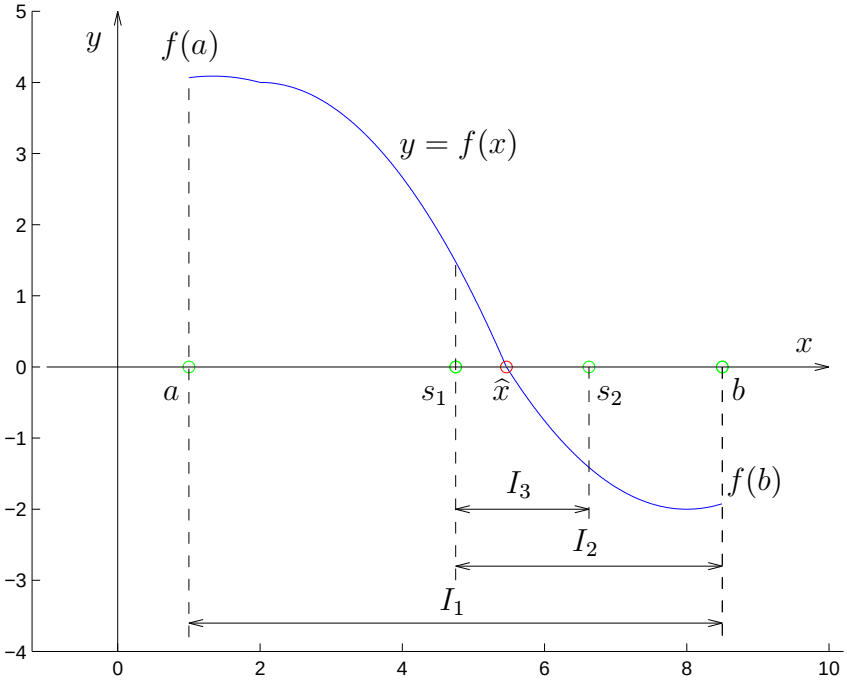
Věta: Předpokládejme, že
(i) reálná funkce f je spojitá pro $x \in \langle a, b\rangle$,
(ii) $f(a).f(b) < 0$.
Potom existuje aspoň jedno řešení x rovnice $f(x) = 0$ na $\langle a, b\rangle$.

Větu ilustruje následující obrázek.



Startovací metody

Nejjednodušší startovací metodou je tzv. **metoda půlení intervalu** nebo-li **bisekce**. Předpokládáme, že jsou splněny předpoklady předchozí věty. Princip bisekce je (jak název říká) založen na půlení zadaného intervalu. Po rozpůlení se testuje funkční hodnota uprostřed intervalu, má-li stejné znaménko jako funkční hodnota $f(a)$, pak se do tohoto středu přesune bod a, v opačném případě se do něj přesune bod b. Celý postup opakujeme znovu. Pro zastavení tohoto iteračního postupu nám poslouží podmínka na velikost vzniklého intervalu. Algoritmus metody je znázorněn na obrázku a jednoduše zapsán dále.



Algoritnus metody bisekce:

1) Zadáme $\varepsilon > 0$, a, b
2) $s = (a + b)/2$
3) Je-li $f(s) = 0$, pak $x = s$, KONEC
4) Je-li $f(s) \neq 0$, pak
   je-li $f(a).f(s) < 0$, pak $b = s$
   jinak $a = s$
5) Je-li $b - a \geq \varepsilon$, pak jdi na 2)
   jinak $x = (a + b)/2$, KONEC

Příklad: Pomocí metody půlení intervalu najděte na intervalu $\langle 1, 4\rangle$ řešení rovnice (proveďte 4 iterace)

$$x^2 + \ln x - \frac{10}{x} = 0.$$

Řešení: Výsledky lze přehledně zapsat do tabulky:

iterace	a	b	$f(a)$	$f(b)$	$s = \dfrac{a+b}{2}$	$f(s)$
0	1	4	-9	$14,8863$	$2,5$	$3,1663$
1	1	$2,5$	-9	$3,1663$	$1,75$	$-2,0922$
2	$1,75$	$2,5$	$-2,0922$	$3,1663$	$2,125$	$0,5635$
3	$1,75$	$2,125$	$-2,0922$	$0,5635$	$1,9375$	$-0,7460$
4	$1,9375$	$2,125$	$-0,7460$	$0,5635$	$\mathbf{2,0312}$	$-0,0884$

Z tabulky vidíme, že funkční hodnota ve středu posledního intervalu, tj. v bodě $2,0312$, je $-0,0884$. Střed intervalu z poslední iterace prohlásíme za přibližné řešení dané rovnice. Píšeme tedy, že $\widehat{x} \approx 2,0312$. Pro úplnost dodejme, že přesné řešení zapsané na 4 desetinná místa je $2,0439$. □

Poznámka: Z uvedeného příkladu je zřejmé, že se metoda bisekce řadí mezi startovací metody (vždy konverguje, ale velmi pomalu). Výhodou je kromě její jednoduchosti i fakt, že se dá předem určit počet kroků, potřebných k dosažení požadované přesnosti (lze ukázat, že pro zpřesnění o jedno desetinné místo je třeba provést přibližně 3,3 iterace). Nevýhodou kromě pomalé konvergence je fakt, že se tato metoda nedá použít pro určení komplexního kořene (např. u polynomů).

Dále se budeme věnovat velmi důležité **metodě prosté iterace**. Všechny dále uvažované metody budou ve své podstatě speciálním případem této metody.

Princip metody prosté iterace je založen na tom, že původní rovnici $f(x) = 0$ přepíšeme na tvar

$$x = \varphi(x).$$

Zde je třeba si uvědomit, že existuje celá řada možností, jak to udělat. Samozřejmě na konkrétní volbě funkce φ bude záviset nejen samotná konvergence metody, ale i její rychlost.

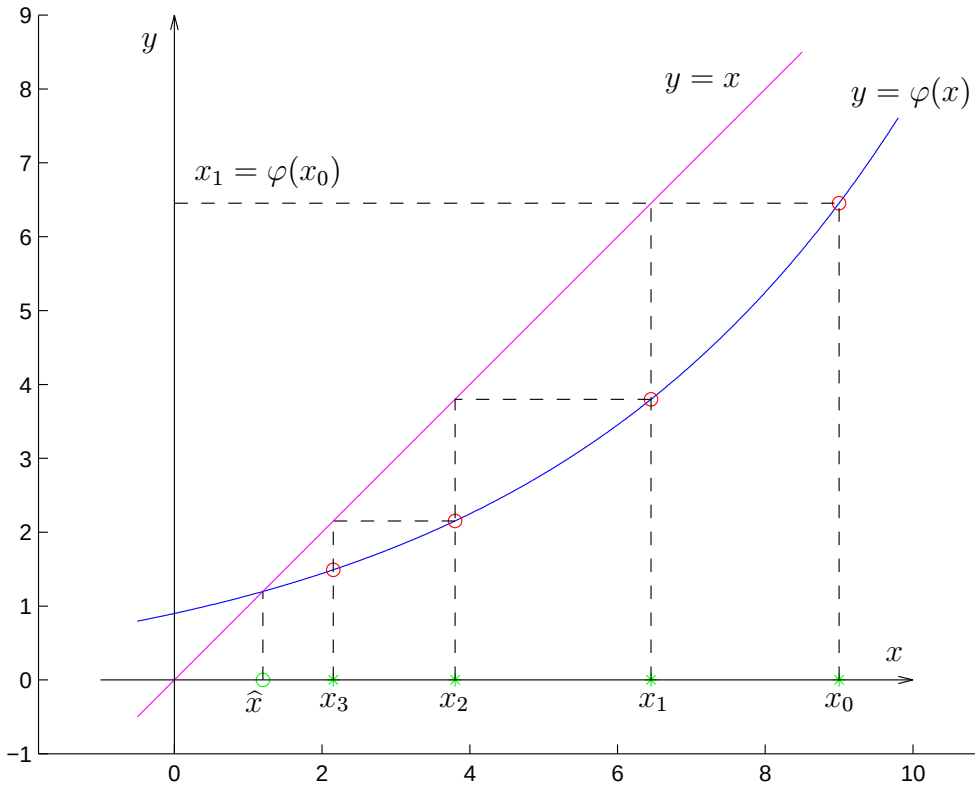
Princip metody prosté iterace je zřejmý z následujícího algoritmu a obrázku. Nakreslíme-li si grafy funkcí $y = x$ a $y = \varphi(x)$, je jasné, že řešení bude v bodě, kde se tyto grafy protnou. Nejprve si zvolíme v intervalu $\langle a, b\rangle$ počáteční nultou iteraci řešení x_0 a potom konstruujeme posloupnost iterací x_k pomocí předpisu

$$x_{k+1} = \varphi(x_k).$$

Pro zastavení iteračního postupu použijeme podmínku pro rozdíl dvou po sobě jdoucích iterací.

Algoritnus metody prosté iterace:

1) Zadáme $x_0 \in \langle a, b\rangle$, $\varepsilon > 0$
2) $x_{k+1} = \varphi(x_k)$
3) Je-li $|x_{k+1} - x_k| < \varepsilon$, pak $x = x_{k+1}$, KONEC
   jinak jdi na 2)



Příklad: Pomocí metody prosté iterace najděte na intervalu $\langle 1, 4\rangle$ řešení rovnice

$$x^2 + \ln x - \frac{10}{x} = 0.$$

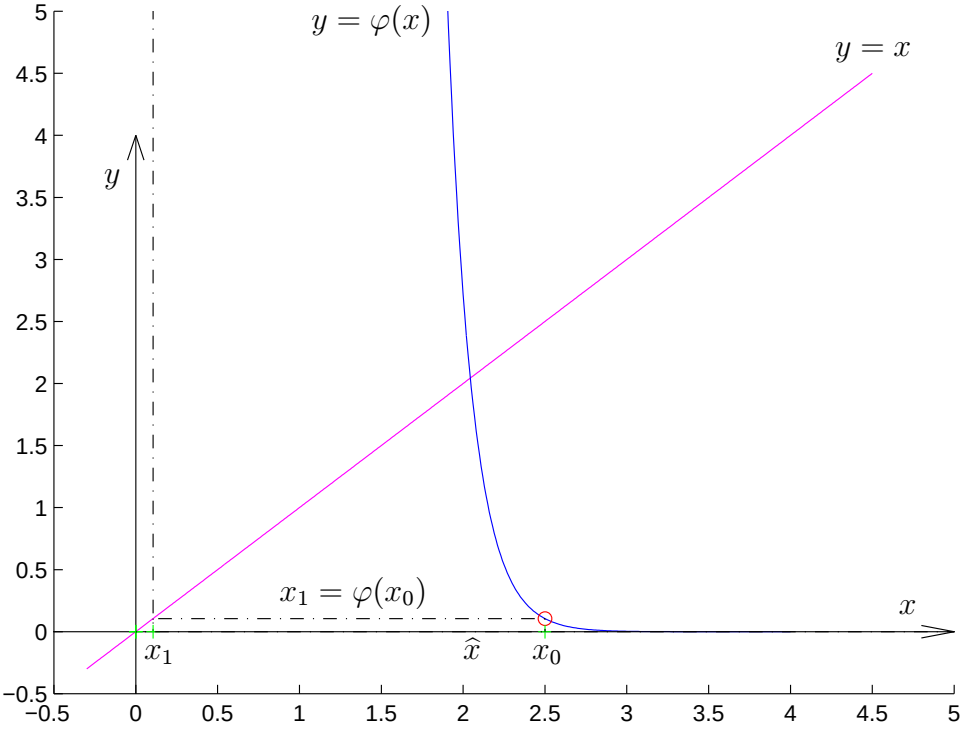
Řešení: Jak již bylo řečeno, způsobů jak přepsat rovnici $f(x) = 0$ na tvar $x = \varphi(x)$ je vždy celá řada. Ukažme si proto jak se bude chovat metoda prosté iterace v několika případech. Pokaždé volíme za počáteční iteraci střed daného intervalu, tj. $x_0 = 2,5$. Výsledky jsou zaznamenány v tabulkách.

1. způsob:

$$\ln x = \frac{10}{x} - x^2 \quad \Rightarrow \quad \boxed{x = e^{(\frac{10}{x} - x^2)}} \quad \text{tj.} \quad \varphi(x) = e^{(\frac{10}{x} - x^2)}$$

k	x_k
0	$2,5$
1	0.1054
2	$1.5845 \cdot 10^{41}$

již první iterace x_1 je mimo zadaný interval, navíc druhá iterace x_2 je velmi velké číslo a proto metoda prosté iterace nekonverguje.

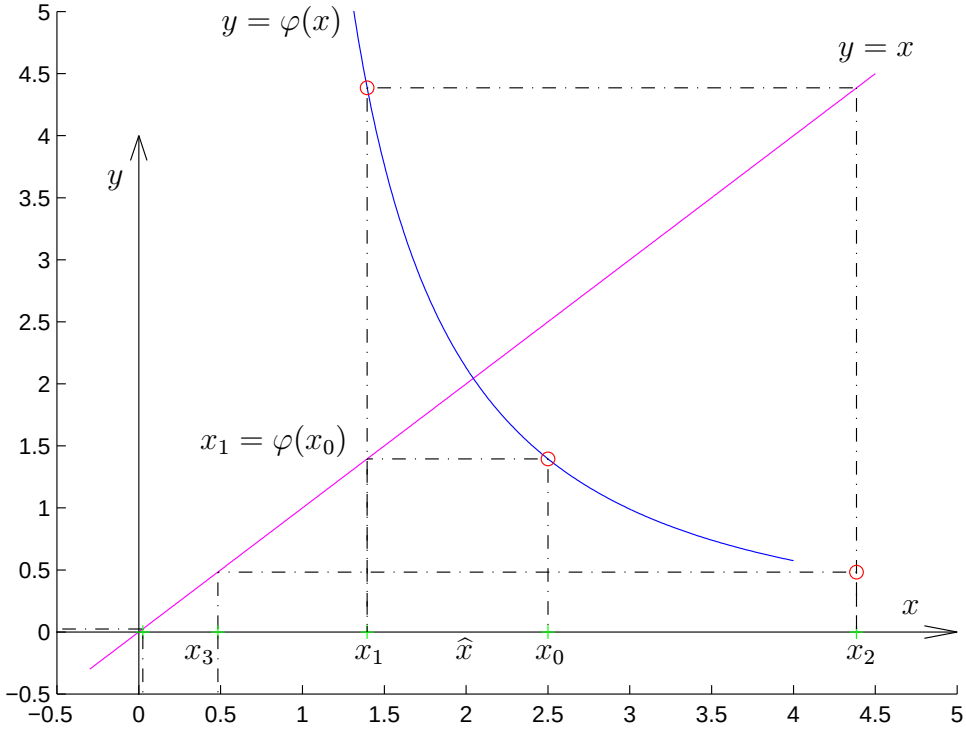


2. způsob:

$$x^2 + \ln x = \frac{10}{x} \quad \Rightarrow \quad \boxed{x = \frac{10}{x^2 + \ln x}} \quad \text{tj.} \quad \varphi(x) = \frac{10}{x^2 + \ln x}$$

k	x_k
0	$2,5$
1	1.3954
2	4.3852
3	0.4829
4	-20.2122

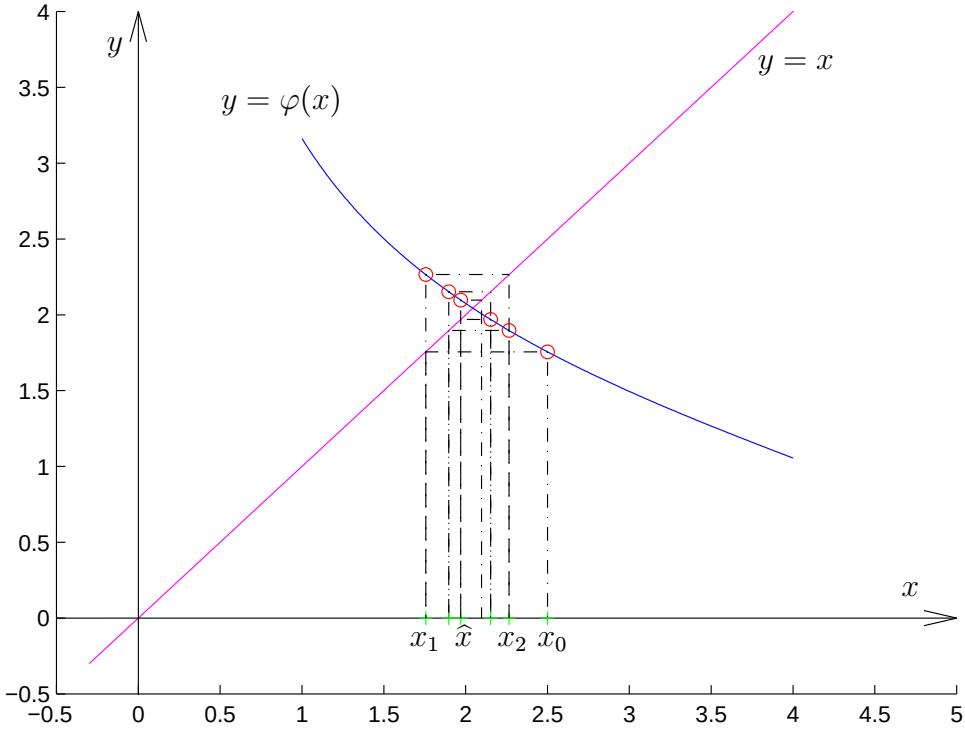
podobně jako v předchozím případě, zde je 2. iterace x_2 mimo zadaný interval a metoda prosté iterace opět nekonverguje.



3. způsob:

$$x^2 = \frac{10}{x} - \ln x \quad \Rightarrow \quad \boxed{x = \sqrt{\frac{10}{x} - \ln x}} \quad \text{tj.} \quad \varphi(x) = \sqrt{\frac{10}{x} - \ln x}$$

k	x_k
0	2,5
1	1.7560
2	2.2653
3	1.8965
4	2.1524
5	1.9696
6	2.0974
7	2.0067
8	2.0704
9	2.0254
10	2.0571
11	2.0347
12	2.0505
13	2.0393
14	2.0472
15	2.0416
16	2.0455
17	2.0428
18	2.0447
19	2.0434

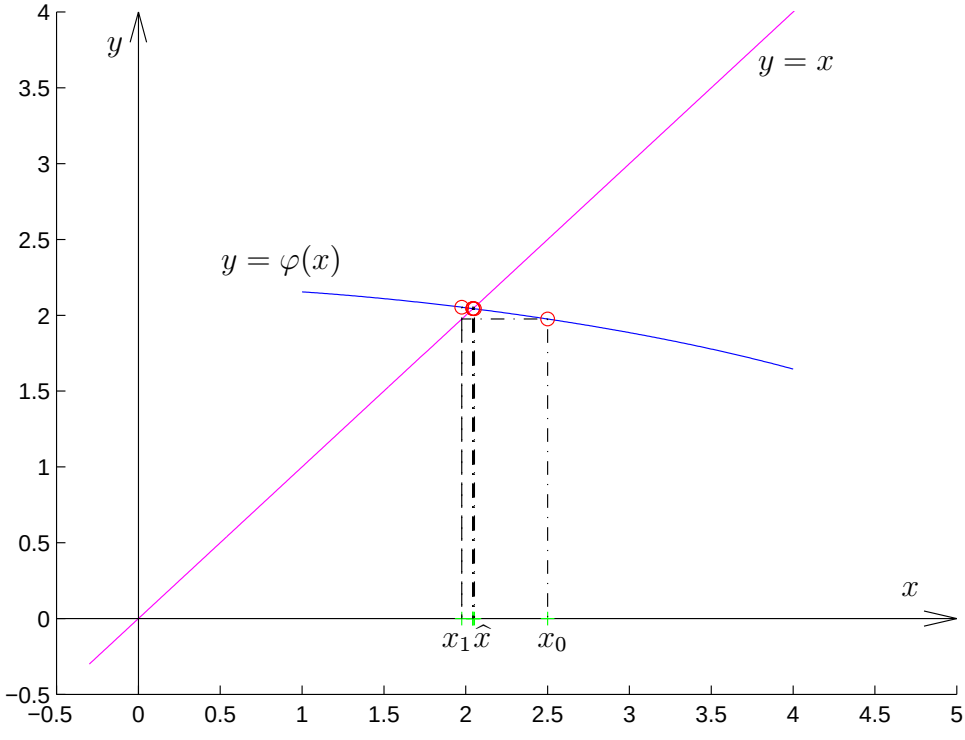


V tomto případě metoda prosté iterace konvergovala k výsledku $\widehat{x}$. Rychlost konvergence ovšem nebyla velká. Pro zastavení výpočtu jsme použili podmínku, aby absolutní hodnota rozdílu dvou po sobě jdoucích iterací byla menší než 0.001. Z tabulky je vidět, že rozdíl mezi 18-tou a 19-tou iterací je přibližně 0.0007. Je třeba poznamenat, že hodnota rozdílu po sobě jdoucích iterací nic neříká o přesnosti vypočteného řešení (viz. Poznámka za příkladem).

4. způsob:

$$x^3 + x\ln x - 10 = 0 \quad \Rightarrow \quad \boxed{x = \sqrt[3]{10 - x\ln x}} \quad \text{tj.} \quad \varphi(x) = \sqrt[3]{10 - x\ln x}$$

k	x_k
0	2,5
1	1.9755
2	2.0532
3	2.0427
4	2.0441
5	2.0439



V tomto posledním případě metoda prosté iterace konvergovala k výsledku $\widehat{x}$ velmi rychle. To dokazuje ten fakt, že kdybychom použili pro zastavení podmínku, aby absolutní hodnota rozdílu dvou po sobě jdoucích iterací byla menší než 10^{-10} potřebovali bychom k tomu pouze 10 iterací.

□

Poznámka: Největší problém metody prosté iterace je nalezení předpisu pro funkci $\varphi = \varphi(x)$ tak, aby metoda konvergovala.

V následující větě, která uvádí postačující podmínky konvergence metody prosté iterace, jsou uvedeny požadavky na vlastnosti funkce φ.

Věta: (Postačující podmínky konvergence metody prosté iterace)
Předpokládejme, že je funkce φ na intervalu $I = \langle a, b\rangle$ spojitá a platí:

(a) $\forall x \in I : \varphi(x) \in I$ (**funkce φ zobrazuje I do sebe**),

(b) $\exists q \in \langle 0, 1) : \ |\varphi(x) - \varphi(y)| \leq q|x - y| \quad \forall x, y \in I$ (**funkce φ je kontrakce**).

Potom

1) v intervalu I existuje právě jeden kořen x rovnice $x = \varphi(x)$,
2) posloupnost $\{x_k\}_{k=1}^{\infty}$ určená formulí $x_k = \varphi(x_{k-1})$ konverguje pro každé $x_0 \in I$ a $\lim\limits_{k\to\infty} x_k = x$.

Poznámka: Pro diferencovatelnou funkci φ lze podmínku (b) nahradit podmínkou

(b‘) $\exists q \in \langle 0, 1) : \ |\varphi'(x)| \leq q \quad \forall x \in I$

Podívejme se na souvislost předpokladů předchozí věty a volby funkce φ v jednotlivých případech předchozího příkladu. Ve všech případech byla funkce φ na zadaném intervalu $\langle 1, 4\rangle$ spojitá a dokonce diferencovatelná. Proto se podívejme na splnění podmínek (a) a (b‘).

Důvodem, proč metoda prosté iterace selhala v 1. a 2. případě je fakt, že funkce φ nesplňovala ani jednu z podmínek (a) a (b‘). Nesplnění podmínky (a) jsme pocítili již v první, resp. druhé iteraci, tím, že jsme se dostali mimo zadaný interval $\langle 1, 4\rangle$. Samozřejmě to souvisí i s faktem, že nebyla splněna podmínka ani (b‘), tj. absolutní hodnoty derivace funkce φ byly velmi velké.

Ve 3. a 4. případě metoda dokonvergovala k řešení. Snadno se z obrázků přesvědčíme, že v těchto případech byly splněny předpoklady (a) i (b‘). Všimněme si ještě jedné skutečnosti, v posledním případě konvergovala metoda prosté iterace viditelně rychleji než ve třetím případě. Bylo to způsobeno vlastností funkce φ. Z geometrické představy je zřejmé, že metoda bude konvergovat rychleji, jestliže bude graf funkce φ „co nejvíce vodorovný“, tzn. $\varphi' \approx 0$. Skutečně rychlost konvergence metody prosté iterace charakterizuje $|\varphi'(x_k)|$, protože můžeme psát

$$\varphi'(x_k) \approx \frac{\varphi(x_{k+1}) - \varphi(x_k)}{x_{k+1} - x_k} = \frac{x_{k+2} - x_{k+1}}{x_{k+1} - x_k}.$$

Poznámka: Řešíme-li metodou prosté iterace algebraickou rovnici

$$a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = 0,$$

a předpokládáme-li, že řešení je v absolutní hodnotě větší než 1, je většinou vhodné vyjádřit x z nejvyšší mocniny, tj.

$$x = \underbrace{\sqrt[n]{-\frac{a_{n-1}x^{n-1} + \ldots + a_1 x + a_0}{a_n}}}_{\varphi(x)}$$

Jak bylo řečeno hodnota rozdílu dvou po sobě jdoucích iterací neodpovídá obecně chybě přibližného řešení. Geometricky si to lze představit takto:

Odhad chyby metody prosté iterace

- Mějme konvergentní iterační proces: $x_k = \varphi(x_{k-1}), \quad k = 1, 2, \ldots$
  řešení α potom splňuje: $\alpha = \varphi(\alpha)$ a $\lim\limits_{k\to\infty} x_k = \alpha$

- Odečtením rovností dostaneme:

$$x_k - \alpha = \varphi(x_{k-1}) - \varphi(\alpha) \tag{A}$$

- Dále platí:

$$|x_{k-1} - \alpha| = |x_{k-1} - x_k + x_k - \alpha| \leq |x_{k-1} - x_k| + |x_k - \alpha| \tag{B}$$

- Potom:

$$|x_k - \alpha| \overset{(A)}{=} |\varphi(x_{k-1}) - \varphi(\alpha)| \overset{(b)}{\leq} q|x_{k-1} - \alpha| \overset{(B)}{\leq} q|x_{k-1} - x_k| + q|x_k - \alpha|$$

$$\underbrace{(1-q)}_{>0}|x_k - \alpha| \leq q|x_{k-1} - x_k|$$

$$|x_k - \alpha| \leq \frac{q}{1-q}|x_{k-1} - x_k|$$

- Použijeme-li zastavovací podmínku $|x_{k-1} - x_k| < \varepsilon$, potom platí odhad chyby:

$$|x_k - \alpha| \leq \frac{q}{1-q}\,\varepsilon$$

Příklad: Pomocí metody prosté iterace řešte na intervalu $\langle 0, 4\rangle$ rovnici

$$x - \sqrt{x+4} = 0.$$

---

přesné řešení:

$$x = \sqrt{x+4} \quad /^2$$
$$x^2 = x + 4$$
$$x^2 - x - 4 = 0$$
$$x_{1,2} = \frac{1 \pm \sqrt{17}}{2} \quad \Rightarrow \quad \underline{x_1 \approx 2,5615}, \quad x_2 \approx -1,5615 \notin \langle 0, 4\rangle$$

- Rovnici přepíšeme na tvar: $x = \underbrace{\sqrt{x+4}}_{\varphi(x)}$
- Ověříme splnění předpokladů věty o postačujících podmínkách konvergence metody prosté iterace:

  (a) $\forall x \in \langle 0, 4\rangle$:
  $$0 \le \sqrt{x+4} \le 4$$
  $$0 \le x + 4 \le 16$$
  $$-4 \le x \le 12$$

  (b′) $\forall x \in \langle 0, 4\rangle$:
  $$|\varphi'(x)| < 1$$
  $$\left|\frac{1}{2\sqrt{x+4}}\right| < 1$$
  $$\frac{1}{2\sqrt{x+4}} < 1$$
  $$1 < 2\sqrt{x+4}$$
  $$1 < 4x + 16$$
  $$-15 < 4x$$

- Vlastní výpočet: volíme $x_0 = 2$ a pro zastavovací podmínku $\varepsilon = 0.001$.

k	x_k
0	2
1	2.4494
2	2.5395
3	2.5572
4	2.5607
5	2.5613

$\Rightarrow \qquad \tilde{x} = x_5 = 2.5613.$

□

Poznámka: Pokusme se odhadnout velikost chyby přibližného řešení u předchozího příkladu. Platí:

$$\varphi' = \frac{1}{2\sqrt{x+4}} \quad \dots \quad \text{kladná klesající funkce} \quad (\varphi'' < 0)$$

$$\Downarrow$$

$$\max_{0\le x\le 4} |\varphi'(x)| = |\varphi'(0)| = \frac{1}{4} = q \quad \dots \quad \text{podmínka (b')}$$

Zvolili jsme $\varepsilon = 0,001$ a proto platí odhad chyby:

$$|x_5 - \alpha| \le \frac{\frac{1}{4}}{1-\frac{1}{4}} \,.\, 0,001 = 0,000333$$

Rychlost konvergence

Definice: Říkáme, že posloupnost x_k konverguje k číslu α rychlostí r, jestliže pro $k \to \infty$

$$|x_{k+1} - \alpha| = c|x_k - \alpha|^r + O(|x_k - \alpha|^{r+1}).$$

Mluvíme o asymptotické rychlosti konvergence ($k \to \infty$).

Poznámka: $f(x) = O(g(x))$ pro $x \to a \quad \Leftrightarrow \quad \left|\frac{f(x)}{g(x)}\right|$ je omezená pro $x \to a$.

Rychlost konvergence metody prosté iterace

Je-li funkce φ dostatečně hladká, můžeme napsat její Taylorův rozvoj v bodě α a potom pro $x = x_{k-1}$ platí:

$$\varphi(x_{k-1}) = \varphi(\alpha) + \varphi'(\alpha)(x_{k-1} - \alpha) + \frac{\varphi''(\alpha)}{2}(x_{k-1} - \alpha)^2 + \frac{\varphi'''(\xi)}{6}(x_{k-1} - \alpha)^3$$

$$x_k - \alpha = \varphi'(\alpha)(x_{k-1} - \alpha) + \frac{\varphi''(\alpha)}{2}(x_{k-1} - \alpha)^2 + \frac{\varphi'''(\xi)}{6}(x_{k-1} - \alpha)^3$$

- je-li $\varphi'(\alpha) \neq 0$, potom

$$x_k - \alpha = \varphi'(\alpha)(x_{k-1} - \alpha)^1 + O((x_{k-1} - \alpha)^2)$$

  $\Rightarrow$ rychlost konvergence je řádu 1

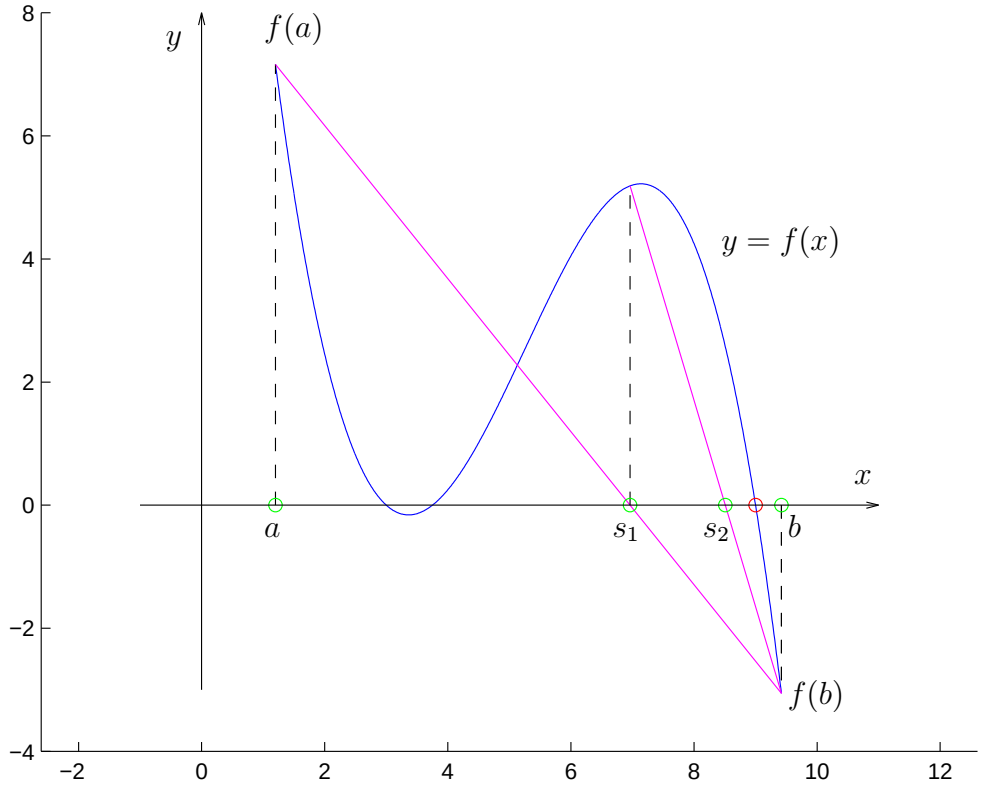
- je-li $\varphi'(\alpha) = 0$ a $\varphi''(\alpha) \neq 0$, potom

$$x_k - \alpha = \frac{\varphi''(\alpha)}{2}(x_{k-1} - \alpha)^2 + O((x_{k-1} - \alpha)^3)$$

  $\Rightarrow$ rychlost konvergence je řádu 2

Nyní si uveďme ještě jednu startovací metodu - metodu **Regula falsi**.

Geometrický význam:
Křivku $y = f(x)$ nahradíme sečnou, která prochází body $[a, f(a)]$ a $[b, f(b)]$. Průsečík s sečny s osou x přiřadíme buď a nebo b (podle znaménka funkční hodnoty).



Algoritnus:

1) Zadáme $\delta > 0$, a, b

2) $s = a - \dfrac{f(a)}{f(b) - f(a)}(b - a)$

3) Je-li $|f(s)| < \delta$, pak $x = s$, KONEC
4) Je-li $|f(s)| \geq \delta$, pak
   je-li $f(a).f(s) < 0$, pak $b = s$, pak jdi na 2)
   jinak $a = s$, pak jdi na 2)

Poznámka: Číslo δ nepředstavuje přesnost! Slouží pouze pro zastavovací podmínku.

Zpřesňující metody

Jako zástupce zpřesňujících metod si uvedeme **Newtonovu metodu**.

Nechť v intervalu $I = \langle a, b \rangle$ leží jediný jednoduchý kořen $\widehat{x}$ rovnice $f(x) = 0$. Jelikož mluvíme o zpřesňující metodě, předpokládáme, že máme zadánu nultou iteraci $x_0 \in I$, která je relativně blízko hledanému řešení. Vyjádříme Taylorův rozvoj funkce f v bodě x_0. Přitom předpokládáme, že existuje derivace funkce f.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(\xi)(x - x_0)^2$$

Rovnici $f(x) = 0$ nahradíme lineární rovnicí

$$f(x_0) + f'(x_0)(x - x_0) = 0$$

Ta má kořen

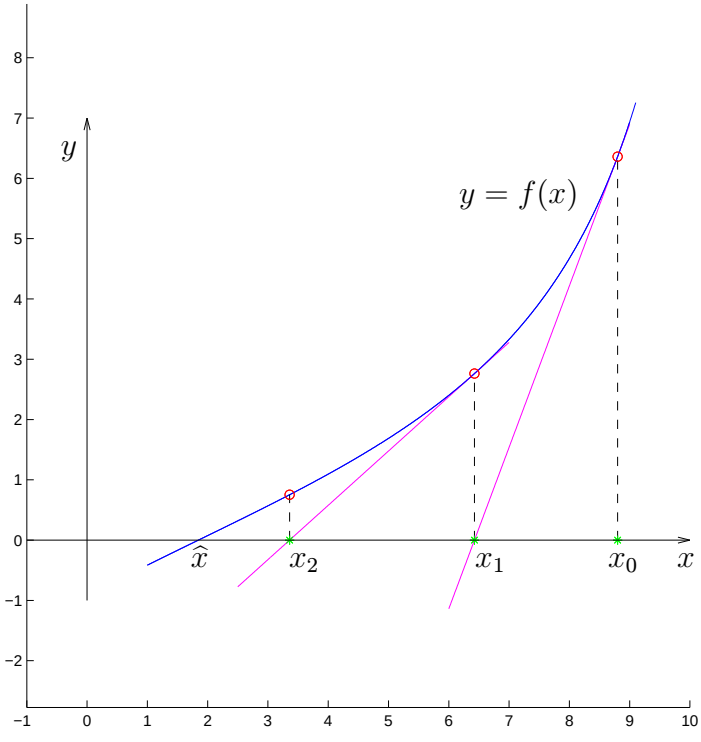
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Celý postup opakujeme a dostáváme iterační formuli

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

Geometrický význam Newtonovy metody:

Křivku $y = f(x)$ nahradíme tečnou ke grafu v bodě x_k a hodnotu x_{k+1} získáme jako průsečík tečny s osou x. Proto se také Newtonova metoda nazývá **metoda tečen** nebo **metoda linearizace**.



Poznámka: Jako zastavovací podmínku lze volit $|x_{k+1} - x_k| < \varepsilon$ nebo $|f(x_k)| < \delta$.

Poznámka: Algoritmus Newtonovy metody je speciálním případem metody prosté iterace. Za funkci φ jsme volili funkci

$$\varphi(x) = x - \frac{f(x)}{f'(x)}$$

Rychlost konvergence Newtonovy metody

- závisí na $\varphi'(\alpha)$ $(\varphi''(\alpha))$ ... viz strana 12.

Platí:

$$\varphi' = 1 - \frac{f'.f' - f.f''}{(f')^2} = 1 - 1 + \frac{f.f''}{(f')^2} = \frac{f.f''}{(f')^2}$$

$$\varphi'(\alpha) = 0, \quad \text{protože } f(\alpha) = 0$$

Platí:

$$\varphi'' = \frac{(f'.f'' + f.f'''). (f')^2 - f.f''.2.f'.f''}{(f')^4}$$

$$\varphi''(\alpha) = \frac{(f')^3.f''}{(f')^4} = \frac{f''(\alpha)}{f'(\alpha)}, \quad \text{protože } f(\alpha) = 0$$

Platí tedy $\varphi'(\alpha) = 0$ a obecně $\varphi''(\alpha) \neq 0 \quad \Rightarrow \quad$ rychlost konvergence je řádu 2.

Příklad: Pomocí Newtonovy metody najděte řešení rovnice

$$f(x) \equiv x^2 + \ln x - \frac{10}{x} = 0.$$

Počáteční aproximaci volte $x_0 = 2,5$ a pro zastavení použijte podmínku $|x_k - x_{k-1}| < 10^{-6}$.

Řešení: V iteračním předpisu se vyskytuje derivace funkce f, proto ji vyjádříme.

$$f'(x) = 2x + \frac{1}{x} + \frac{10}{x^2}.$$

Výsledky přehledně zapíšeme do tabulky.

k	x_k	$f(x_k)$	$f'(x_k)$	$\frac{f(x_k)}{f'(x_k)}$
0	$2,5$	3.1662907	7.0000000	0.4523272
1	2.0476727	0.0260747	6.9686526	0.0037417
2	2.0439310	-0.0000040	6.9708032	0.0000005
3	**2.0439305**			

Vidíme, že stačilo provést 3 iterace. □

Z předcházejícího příkladu se zdá, že Newtonova metoda bude vždy velmi rychle konvergovat. Ukažme si ještě jeden příklad.

Příklad: Pomocí Newtonovy metody najděte řešení rovnice

$$f(x) \equiv x^2 - 4x + 4 = 0.$$

Počáteční aproximaci volte $x_0 = 0,1$ a pro zastavení použijte podmínku $|x_k - x_{k-1}| < 10^{-3}$.

Řešení: Opět je třeba vypočítat derivaci

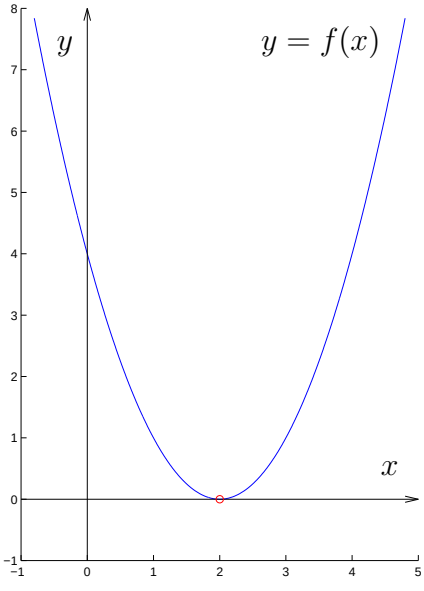
$$f'(x) = 2x - 4.$$

Iterační formule bude mít tvar

$$x_{k+1} = x_k - \frac{x_k^2 - 4x_k + 4}{2x_k - 4} = x_k - \frac{(x_k - 2)^2}{2(x_k - 2)} = x_k - \frac{1}{2}(x_k - 2) = \frac{1}{2}x_k + 1.$$

Výsledky opět zapíšeme do tabulky.

k	x_k
0	0,1
1	1,05
2	1,525
3	1,7625
4	1,8813
5	1,9406
6	1,9703
7	1,9852
8	1,9926



□

Tento příklad ukázal, že v některých situacích je Newtonova metoda mnohem pomalejší. Z obrázku se dá usoudit, kdy tyto situace nastanou. Hodnota derivace funkce f v bodě $\widehat{x} = 2$, který je řešením rovnice $f(x) = 0$, je rovna nule. Graf funkce f se osy x pouze dotkl. To odpovídá faktu, že kořen $\widehat{x} = 2$ je dvojnásobným kořenem dané rovnice. Je-li hodnota derivace f' na okolí kořene rovna 0, případně velmi blízká 0, potom se v iteračním formuli dělí nulou, případně velmi malým číslem. Tento fakt má za následek zpomalení algoritmu.

Podívejme se zpět na předpoklady, za nichž jsme odvodili Newtonovu metodu. Předpokládali jsme, že hledaný kořen je jediný (jeho násobnost je 1). Za tohoto předpokladu konverguje Newtonova metoda rychle. Při zadání úkolu, ale nemusíme být schopní poznat, že násobnost hledaného kořene je větší než 1. Problém s hledáním násobného kořene můžeme elegantně vyřešit tímto způsobem:

Je-li $\widehat{x}$ n-násobným kořenem rovnice $f(x) = 0$, potom je $\widehat{x}$ také $(n-1)$-násobným kořenem rovnice $f'(x) = 0$ a tedy jednoduchým kořenem rovnice

$$\frac{f(x)}{f'(x)} = 0.$$

Poznámka: Dosud jsme řešili nelineární rovnici pouze v $\mathbb{R}$. Algoritmus Newtonovy metody můžeme použít i pro řešení dané rovnice v oboru komplexních čísel.

Příklad: Newtonovou metodou řešte v komplexním oboru rovnici

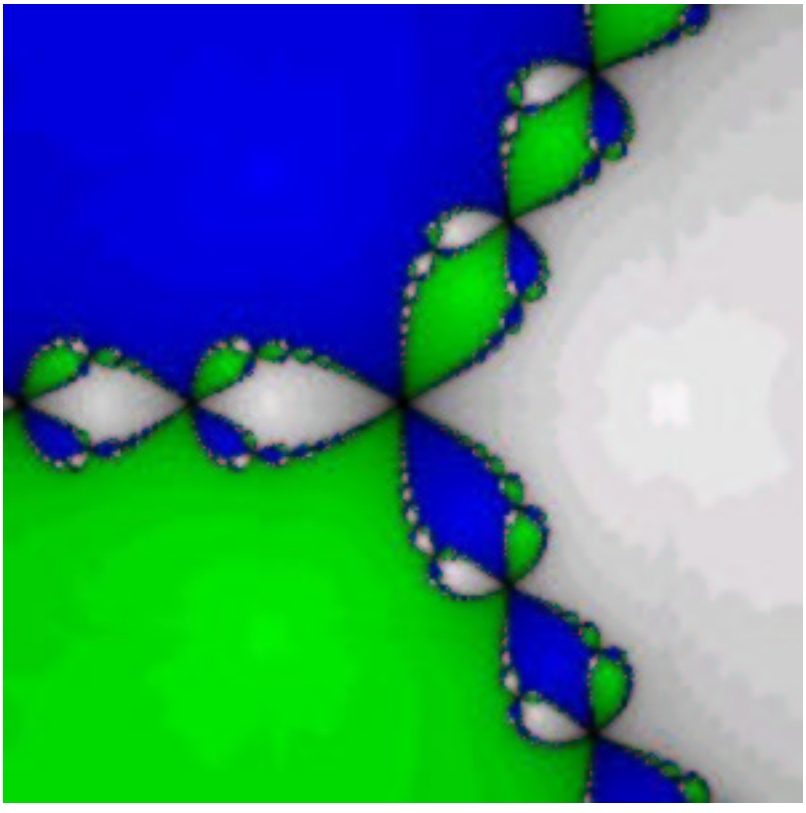
$$z^3 - 1 = 0, \qquad z = x + iy, \quad x, y \in \mathbb{R}$$

Iterační formule bude mít tvar

$$z_{k+1} = z_k - \frac{z_k^3 - 1}{2z_k^2}$$

Je zřejmé, že daná rovnice bude mít 3 řešení: 1, $-\frac{1}{2} + \frac{\sqrt{3}}{2}i$ a $-\frac{1}{2} - \frac{\sqrt{3}}{2}i$.

Řešíme-li danou rovnici Newtonovou metodou pro konkrétní počáteční aproximaci ze čtverce $\langle -1.5; 1.5\rangle \times \langle -1.5; 1.5\rangle$, dostaneme jedno ze tří uvedených řešení. Obarvíme-li bod představující počáteční aproximaci různou barvou, podle toho k jakému řešení dospějeme, získáme fraktálovou strukturu.



□

Poznámka: Na závěr si uvědomme nevýhody Newtonovy metody

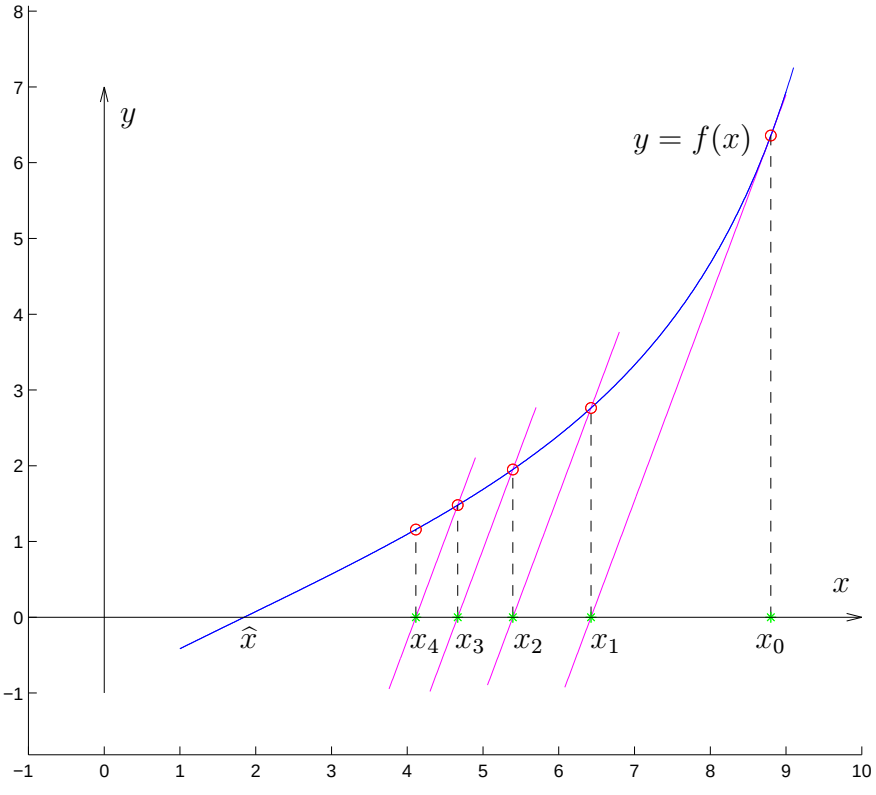
- zadaná funkce f musí být diferencovatelná
- derivace se přímo vyskytuje v iterační formuli
- v každé iteraci musíme kromě funkční hodnoty počítat také hodnotu derivace

Pro odbourání poslední vlastnosti můžeme za předpokladu, že se derivace f' na okolí kořene příliž nemění, Newtonovu metodu modifikovat tak, že hodnotu derivace vypočteme pouze jednou v bodě x_0 a položíme $f'(x_k) \approx f(x_0)$. Dostaneme iterační formuli

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}$$

Geometrický význam modifikované Newtonovy metody

Tečny ke grafu v bodech $[x_k, f(x_k)]$ nahrazujeme přímkami rovnoběžnými s tečnou ke grafu funkce $y = f(x)$ v bodě $[x_0, f(x_0)]$.

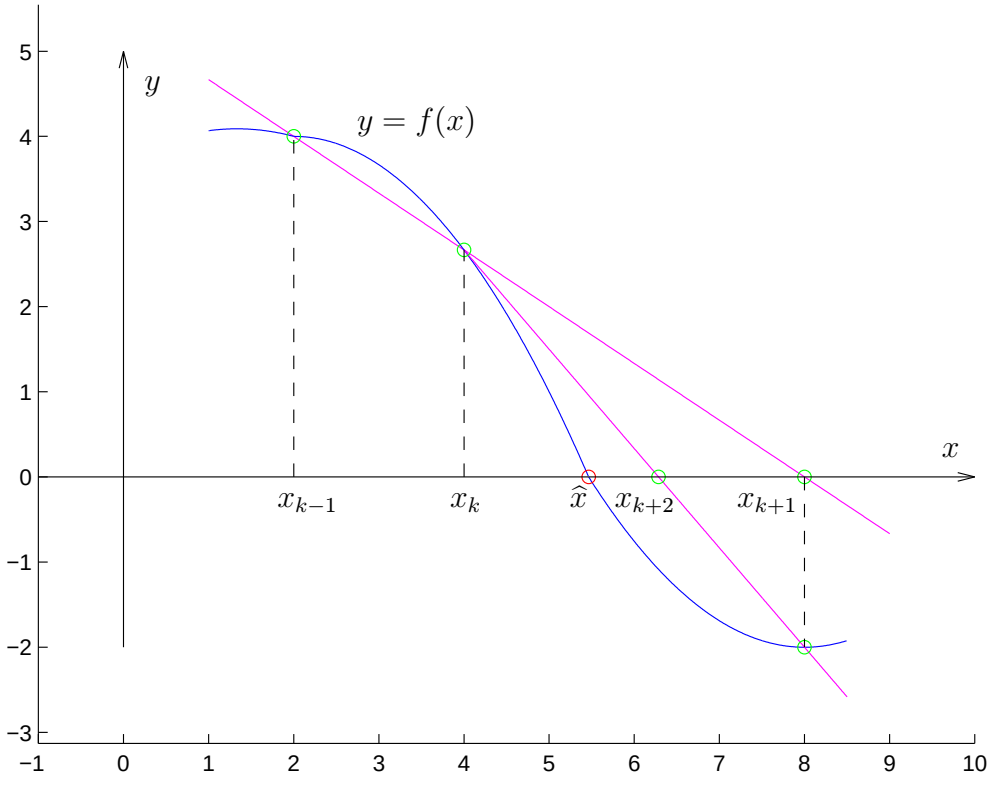


Poznámka: V této modifikaci počítáme pouze jednu hodnotu derivace $f'(x_0)$, a proto je tento postup vhodný je-li derivace $f'(x)$ složitá. Nemění-li $f'(x)$ a $f''(x)$ znaménko, je možné dokázat konvergenci této metody.

Chceme-li modifikovat Newtonovu metodu pro funkce, které nejsou diferencovatelné, nahradíme v iterační formuli derivaci $f'(x_k)$ diferenčním podílem $\dfrac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$. Získáme tzv. **metodu sečen**.

Geometrický význam metody sečen

Mějme dvě dobré aproximace x_{k-1} a x_k kořene x rovnice $f(x) = 0$. Křivku $y = f(x)$ nahradíme přímkou (sečnou), která prochází body $[x_{k-1}, f(x_{k-1})]$ a $[x_k, f(x_k)]$. Další iteraci x_{k+1} získáme jako průsečík sečny s osou x.



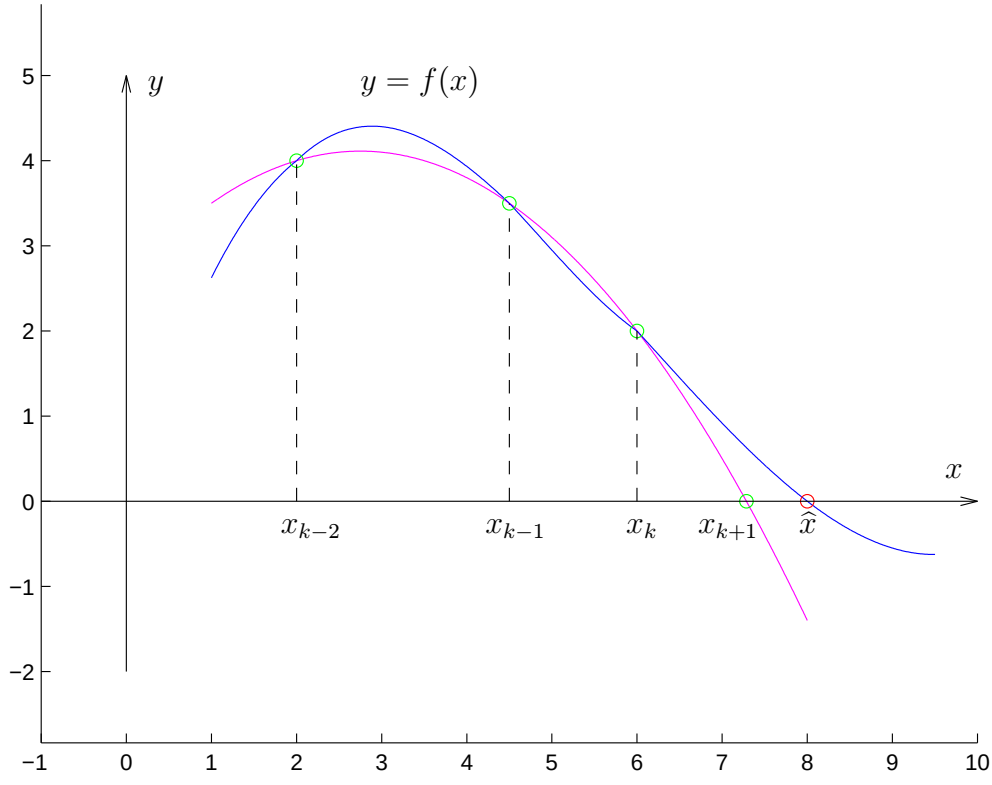
Poznámka: Pro zahájení výpočtu potřebujeme znát dvě počáteční aproximace, ale na rozdíl od Newtonovy metody počítáme v každém kroku pouze jednu novou funkční hodnotu, což je úspora času.

Poznámka: Metoda sečen má obdobný algoritmus jako metoda regula falsi, ovšem nepožadujeme splnění podmínky $f(x_{k-1}).f(x_k) < 0$.

Poznámka: Metoda sečen je tzv. dvoukroková interpolační metoda, analogicky lze odvodit tříkrokovou interpolační metodu, kterou nazýváme **Mullerova metoda**.

Geometrický význam Mullerovy metody

Mějme tři dobré aproximace x_{k-2}, x_{k-1} a x_k kořene x rovnice $f(x) = 0$.
Křivku $y = f(x)$ nahradíme parabolou (kvadratickou funkcí), která prochází body $[x_{k-2}, f(x_{k-2})]$ $[x_{k-1}, f(x_{k-1})]$ a $[x_k, f(x_k)]$.
Další iteraci x_{k+1} získáme jako průsečík paraboly s osou x.
(Ten, který je blíže k x_k.)



Prostředky matlabu pro řešení nelineárních rovnic (viz help v matlabu)

`fzero` pro obecnou nelineární rovnici
`roots` pro kořeny polynomu

ŘEŠENÍ SOUSTAV NELINEÁRNÍCH ROVNIC

Formulace

Jsou dány funkce $F_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, n$ definované na $\langle a_i, b_i \rangle$.

Označme $I = \langle a_1, b_1 \rangle \times \langle a_2, b_2 \rangle \times \ldots \times \langle a_n, b_n \rangle$.

Hledáme $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T \in I$ tak, aby

$$\begin{aligned} F_1(x_1, x_2, \ldots, x_n) &= 0 \\ F_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ F_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

Vektorově:

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

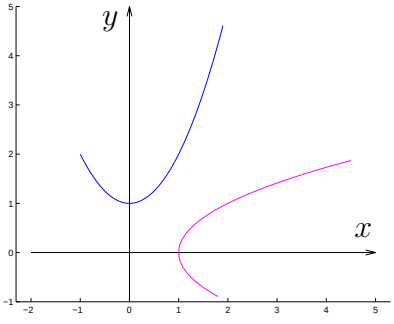
kde $\boldsymbol{F} = (F_1, F_2, \ldots, F_n)^T$.

Příklad: Řešte soustavu dvou rovnic pro dvě neznámé

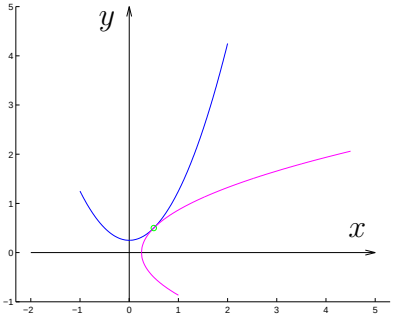
$$x^2 - y + a = 0$$
$$-x + y^2 + a = 0$$

---

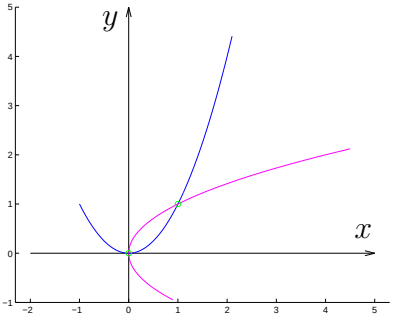
1) $a = 1$



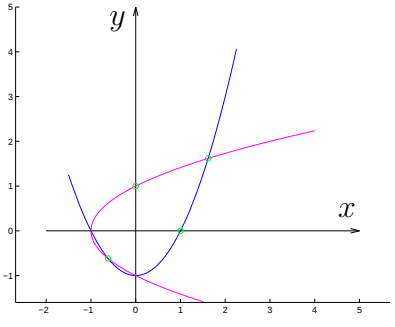
2) $a = \frac{1}{4}$



3) $a = 0$



4) $a = -1$



Metoda prosté iterace

Soustavu rovnic $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$ nahradíme soustavou rovnic $\boldsymbol{x} = \boldsymbol{\Phi}(\boldsymbol{x})$.
(Více možností)

Algoritnus:

1) Zadáme $\boldsymbol{x}^0 \in I$, $\varepsilon > 0$
2) $\boldsymbol{x}^{k+1} = \boldsymbol{\Phi}(\boldsymbol{x}^k)$
3) Je-li $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\| < \varepsilon$, pak $\boldsymbol{x} = \boldsymbol{x}^{k+1}$, KONEC
   jinak jdi na 2)

Věta: (Postačující podmínky konvergence)
Předpokládejme, že je funkce $\boldsymbol{\Phi}$ na I spojitá a platí:
(a) $\forall \boldsymbol{x} \in I : \boldsymbol{\Phi}(\boldsymbol{x}) \in I$ (funkce $\boldsymbol{\Phi}$ zobrazuje I do sebe),
(b) $\exists q \in \langle 0, 1) : \ \|\boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{\Phi}(\boldsymbol{y})\| \leq q\|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y} \in I$
(funkce $\boldsymbol{\Phi}$ je kontrakce).
Potom 1. v množině I existuje právě jedno řešení $\boldsymbol{x}$ soustavy rovnic $\boldsymbol{x} = \boldsymbol{\Phi}(\boldsymbol{x})$,
2. posloupnost $\{\boldsymbol{x}^k\}_{k=1}^{\infty}$ určená formulí $\boldsymbol{x}^k = \boldsymbol{\Phi}(\boldsymbol{x}^{k-1})$ konverguje pro každé $\boldsymbol{x}^0 \in I$ a $\lim\limits_{k\to\infty} \boldsymbol{x}^k = \boldsymbol{x}$.

Poznámka: Pro diferencovatelnou funkci $\boldsymbol{\Phi}$ lze podmínku (b) nahradit
(b‘) $\exists q \in \langle 0, 1) : \ \|\boldsymbol{\Phi}'(\boldsymbol{x})\| \leq q \quad \forall \boldsymbol{x} \in I$

kde $\boldsymbol{\Phi}'(\boldsymbol{x}) = \left[\dfrac{\partial \Phi_i(x_1, x_2, \ldots, x_n)}{\partial x_j}\right]$ je Jacobiho matice.

Výhoda: Výpočet podle rekurentní formule je velmi jednoduchý.

Nevýhoda: Obecně je obtížné najít funkci $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{x})$ tak, aby byla zaručena konvergence.

Příklad:

Řešte metodou prosté iterace soustavu dvou rovnic pro dvě neznámé

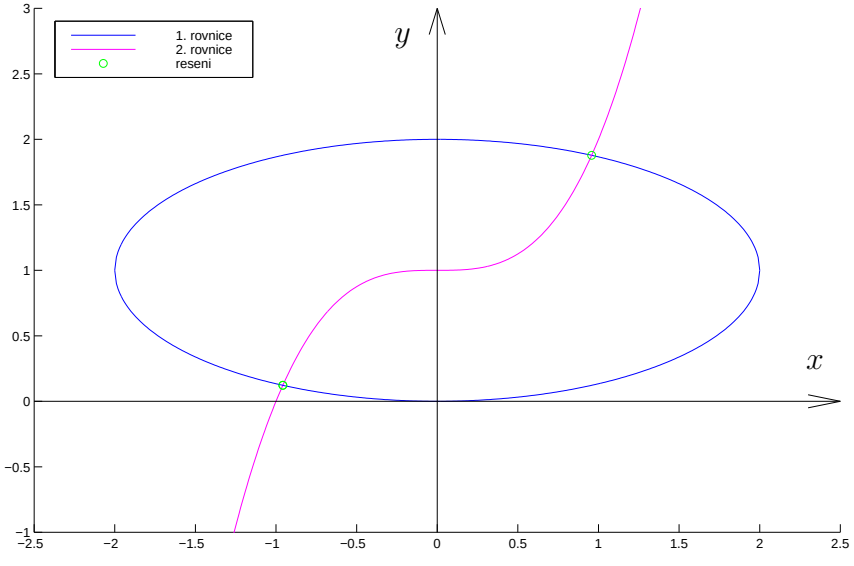
$$x^2 + 4y^2 - 8y = 0$$
$$x^3 - y + 1 = 0$$

---

1. rovnice je rovnicí eplipsy $x^2 + 4(y-1)^2 = 4$

$$\left(\frac{x}{2}\right)^2 + (y-1)^2 = 1$$

2. rovnici upravíme na tvar $y = x^3 + 1$



Z 2.rovnice vyjádříme x:

$$x^3 = y - 1$$

$$x = \sqrt[3]{y - 1}$$

Z 1.rovnice vyjádříme y:

$$4y^2 = 8y - x^2$$

$$y = \frac{1}{2}\sqrt{8y - x^2}$$

Rekurentní formule:

$$x_{k+1} = \sqrt[3]{y_k - 1} \qquad y_{k+1} = \frac{1}{2}\sqrt{8y_k - x_{k(+1)}^2}$$

Řešení: pro $x_0 = 2$, $y_0 = 2$, $\varepsilon = 0,01$

k	x_k	y_k	$\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$
0	2.0000	2.0000	
1	1.0000	1.7321	0.1159
2	0.9013	1.7928	0.0523
3	0.9255	1.8392	0.0283
4	0.9432	1.8612	0.0126
5	0.9514	1.8708	0.0054

Newtonova metoda

Odvození je opět analogické případu funkce jedné reálné proměnné.
Vyjádříme si Taylorův rozvoj funkce $\boldsymbol{F}$ v bodě $\boldsymbol{x}^k$.
(Předpokládáme, že existují derivace !)

$$\boldsymbol{F}(x) = \boldsymbol{F}(x^k) + \boldsymbol{F}'(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) + \frac{1}{2}\boldsymbol{F}''(\boldsymbol{\xi})(\boldsymbol{x} - \boldsymbol{x}_k)^2$$

Soustavu rovnic $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$ nahradíme soustavou lineárních rovnic

$$\boldsymbol{F}(\boldsymbol{x}^k) + \boldsymbol{F}'(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) = \boldsymbol{0}$$

Její řešení označíme $\boldsymbol{x}^{k+1}$, tj.

$$\boldsymbol{F}(\boldsymbol{x}^k) + \boldsymbol{F}'(\boldsymbol{x}^k)\underbrace{(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k)}_{\boldsymbol{h}^k} = \boldsymbol{0}$$

Dostáváme soustavu

$$\boldsymbol{F}'(\boldsymbol{x}^k)\boldsymbol{h}^k = -\boldsymbol{F}(\boldsymbol{x}^k),$$

která má řešení

$$\boldsymbol{h}^k = -[\boldsymbol{F}'(\boldsymbol{x}^k)]^{-1}\boldsymbol{F}(\boldsymbol{x}^k)$$

Novou iteraci $\boldsymbol{x}^{k+1}$ získáme ze vztahu

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{h}^k$$

Poznámka:

$\boldsymbol{F}'(\boldsymbol{x}^k)$ je Jacobiho matice funkce $\boldsymbol{F}(\boldsymbol{x})$ v bodě $\boldsymbol{x}^k$.
Je zřejmé, že musí být regulární (musí existovat matice k ní inverzní).

Příklad:

Řešte Newtonovou metodou soustavu dvou rovnic pro dvě neznámé

$$x^2 + 4y^2 - 8y = 0$$
$$x^3 - y + 1 = 0$$

$$\boldsymbol{F}(x,y) = \begin{bmatrix} x^2 + 4y^2 - 8y \\ x^3 - y + 1 \end{bmatrix}$$

$$\frac{\partial F_1(x,y)}{\partial x} = 2x \qquad \frac{\partial F_1(x,y)}{\partial y} = 8y - 8$$
$$\frac{\partial F_2(x,y)}{\partial x} = 3x^2 \qquad \frac{\partial F_2(x,y)}{\partial y} = -1$$

$$\boldsymbol{F}'(x,y) = \begin{bmatrix} 2x & 8y - 8 \\ 3x^2 & -1 \end{bmatrix}$$

1. iterace

$$\begin{bmatrix} x^1 \\ y^1 \end{bmatrix} = \begin{bmatrix} x^0 \\ y^0 \end{bmatrix} + \begin{bmatrix} h_1^0 \\ h_2^0 \end{bmatrix}$$

Platí

$$\boldsymbol{h}^0 = -[\boldsymbol{F}'(x^0, y^0)]^{-1} \boldsymbol{F}(x^0, y^0)$$

Abychom nemuseli počítat inverzní matici, vypočteme $\boldsymbol{h}^0$ jako řešení soustavy

$$\boldsymbol{F}'(x^0, y^0)\boldsymbol{h}^0 = -\boldsymbol{F}(x^0, y^0)$$

$$\boldsymbol{F}(x^0, y^0) = \begin{bmatrix} 4 \\ 7 \end{bmatrix} \qquad \boldsymbol{F}'(x^0, y^0) = \begin{bmatrix} 4 & 8 \\ 12 & -1 \end{bmatrix}$$

Dostaneme

$$\boldsymbol{h}^0 = \begin{bmatrix} -\frac{3}{5} \\ -\frac{1}{5} \end{bmatrix} \Rightarrow \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} = \begin{bmatrix} x^0 \\ y^0 \end{bmatrix} + \begin{bmatrix} h_1^0 \\ h_2^0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} -\frac{3}{5} \\ -\frac{1}{5} \end{bmatrix} = \begin{bmatrix} 1,4 \\ 1,8 \end{bmatrix}$$

2. iterace

$$\begin{bmatrix} x^2 \\ y^2 \end{bmatrix} = \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} + \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix}$$

Platí

$$\boldsymbol{h}^1 = -[\boldsymbol{F}'(x^1, y^1)]^{-1} \boldsymbol{F}(x^1, y^1)$$

Abychom nemuseli počítat inverzní matici, vypočteme $\boldsymbol{h}^1$ jako řešení soustavy

$$\boldsymbol{F}'(x^1, y^1)\boldsymbol{h}^1 = -\boldsymbol{F}(x^1, y^1)$$

$$\boldsymbol{F}(x^1, y^1) = \begin{bmatrix} 0,52 \\ 1,944 \end{bmatrix} \qquad \boldsymbol{F}'(x^1, y^1) = \begin{bmatrix} 2,8 & 6,4 \\ 5,88 & -1 \end{bmatrix}$$

Dostaneme

$$\boldsymbol{h}^1 = \begin{bmatrix} -0,3206 \\ 0,0590 \end{bmatrix}$$

$$\begin{bmatrix} x^2 \\ y^2 \end{bmatrix} = \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} + \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} 1,4 \\ 1,8 \end{bmatrix} + \begin{bmatrix} -0,3206 \\ 0,0590 \end{bmatrix} = \begin{bmatrix} 1,0794 \\ 1,8590 \end{bmatrix}$$

3. iterace

$$\begin{bmatrix} x^3 \\ y^3 \end{bmatrix} = \begin{bmatrix} x^2 \\ y^2 \end{bmatrix} + \begin{bmatrix} h_1^2 \\ h_2^2 \end{bmatrix}$$

Platí

$$\boldsymbol{h}^2 = -[\boldsymbol{F}'(x^2, y^2)]^{-1} \boldsymbol{F}(x^2, y^2)$$

Abychom nemuseli počítat inverzní matici, vypočteme $\boldsymbol{h}^2$ jako řešení soustavy

$$\boldsymbol{F}'(x^2, y^2)\boldsymbol{h}^2 = -\boldsymbol{F}(x^2, y^2)$$

$$\boldsymbol{F}(x^2, y^2) = \begin{bmatrix} 0,52 \\ 1,944 \end{bmatrix} \qquad \boldsymbol{F}'(x^1, y^1) = \begin{bmatrix} 2,8 & 6,4 \\ 5,88 & -1 \end{bmatrix}$$

Dostaneme

$$\boldsymbol{h}^1 = \begin{bmatrix} -0,3206 \\ 0,0590 \end{bmatrix}$$

$$\begin{bmatrix} x^2 \\ y^2 \end{bmatrix} = \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} + \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} 1,4 \\ 1,8 \end{bmatrix} + \begin{bmatrix} -0,3206 \\ 0,0590 \end{bmatrix} = \begin{bmatrix} 1,0794 \\ 1,8590 \end{bmatrix}$$

k	x_k	y_k	$\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$
0	2.0000	2.0000	
1	1.4000	1.8000	0.6325
2	1.0794	1.8590	0.3260
3	0.9703	1.8763	0.1105
4	0.9577	1.8779	0.0127

SOUSTAVY LINEÁRNÍCH ALGEBRAICKÝCH ROVNIC

Pojmy:

- *Algebraická rovnice* ... rovnice obsahující pouze celé nezáporné mocniny neznámé x, tj. $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0 = 0$, kde n je přirozené číslo.
- *Lineární algebraická rovnice* ... rovnice obsahující pouze první mocninu neznámé, resp. neznámých, tj. $ax + b = 0$, resp. např. $a_1x_1 + a_2x_2 + a_3x_3 = b$.
- *Soustava lineárních algebraických rovnic*

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

Příklad: Řešte soustavu 3 rovnic pro 3 neznámé x, y a z:

$$\begin{array}{rcrcrcr} 3x & + & 2y & + & z & = & 10 \\ 2x & + & 4y & + & 5z & = & 25 \\ 3x & + & 4y & + & 8z & = & 35 \\ \hline \end{array}$$

Řešení: Snažíme se postupně eliminovat jednotlivé neznámé. Docílíme to tím způsobem, že budeme sčítat vhodně přenásobené rovnice.

1. krok Opíšeme 1. rovnici

$$3x + 2y + z = 10$$

K 2. rovnici přičteme určitý násobek 1. rovnice tak, aby se eliminovala neznámá x (vhodný koeficient je $-\frac{2}{3}$).

$$\begin{array}{rcrcrcr} 2x & + & 4y & + & 5z & = & 25 \\ -\frac{2}{3}\cdot(3x & + & 2y & + & z) & = & -\frac{2}{3}\cdot 10 \\ \hline & & 4y - \frac{4}{3}y & + & 5z - \frac{2}{3}z & = & 25 - \frac{20}{3} \\ & & \frac{12-4}{3}y & + & \frac{15-2}{3}z & = & \frac{75-20}{3} \\ & & \frac{8}{3}y & + & \frac{13}{3}z & = & \frac{55}{3} \\ \hline \end{array}$$

Podobně eliminujeme neznámou x z 3. rovnice, tj. k 3. rovnici přičteme -1 násobek první rovnice.

$$\begin{array}{rcrcrcr} 3x & + & 4y & + & 8z & = & 35 \\ -1\cdot(3x & + & 2y & + & z) & = & -1\cdot 10 \\ \hline & & 4y-2y & + & 8z-z & = & 35-10 \\ & & & & 2y & + & 7z & = & 25 \end{array}$$

Obdrželi jsme soustavu

$$\begin{array}{rcrcrcr} 3x & + & 2y & + & z & = & 10 \\ & & \frac{8}{3}y & + & \frac{13}{3}z & = & \frac{55}{3} \\ & & 2y & + & 7z & = & 25 \end{array}$$

2. krok Nyní opíšeme první dvě rovnice a ze třetí budeme eliminovat neznámou y, tj. ke 3. rovnici přičteme $-\frac{2}{\frac{8}{3}}$ násobek 2. rovnice.

$$\begin{array}{rcrcr} 2y & + & 7z & = & 25 \\ -\frac{2}{\frac{8}{3}}\cdot(\frac{8}{3}y & + & \frac{13}{3}z) & = & -\frac{2}{\frac{8}{3}}\cdot\frac{55}{3} \\ \hline 7z-\frac{2}{\frac{8}{3}}\cdot\frac{13}{3}z & = & 25-\frac{2\cdot 55}{8} \\ 7z-\frac{13}{4}z & = & 25-\frac{55}{4} \\ \frac{28-13}{4}z & = & \frac{100-55}{4} \\ \frac{15}{4}z & = & \frac{45}{4} \end{array}$$

Obdrželi jsme soustavu (tzv. **trojúhelníkový tvar**)

$$\begin{array}{rcrcrcr} 3x & + & 2y & + & z & = & 10 \\ & & \frac{8}{3}y & + & \frac{13}{3}z & = & \frac{55}{3} \\ & & & & \frac{15}{4}z & = & \frac{45}{4} \end{array}$$

3. krok z 3. rovnice vypočteme $z = 3$

4. krok Dosadíme z do 2. rovnice a vypočteme y

$$\begin{array}{rcl} \frac{8}{3}y + \frac{13}{3}\cdot 3 &=& \frac{55}{3} \\ \frac{8}{3}y &=& \frac{55-39}{3} \\ y &=& 2 \end{array}$$

5. krok Dosadíme y, z do 1. rovnice a vypočteme x

$$\begin{array}{rcl} 3x + 2\cdot 2 + 3 &=& 10 \\ 3x &=& 10-7 \\ x &=& 1 \end{array}$$

□

Uvedený postup lze samozřejmě aplikovat na obecnou soustavu n rovnic pro n neznámých a nazývá se **Gaussova eliminační metoda (GEM)** ... soustavu nejprve převedeme na trojúhelníkový tvar a potom zpětným chodem vyjadřujeme řešení.

Pro větší přehlednost můžeme použít maticový zápis:

$$\underbrace{\begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 5 \\ 3 & 4 & 8 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 10 \\ 25 \\ 35 \end{bmatrix}}_{\mathbf{b}}, \qquad \text{tj.} \qquad \boxed{\mathbf{Ax = b}}$$

A ... matice koeficientů u neznámých v jednotlivých rovnicích
x ... vektor neznámých x, y a z
b ... vektor pravé strany

Poznámka: (**Násobení matice · vektor**) Výsledek násobení matice typu $m|n$ (m řádek a n sloupců) a vektoru typu $n|1$ (sloupcový vektor o n prvcích) je vektor typu $m|1$ (sloupcový vektor o m prvcích), kde i-tá složka výsledného vektoru je skalárním součinem i-tého řádku matice a daného vektoru.

Mluvíme o numerické matematice, tzn. řešíme dané problémy numericky (nikoli analyticky). Pro řešení problémů odvozujeme postupy (algoritmy), které budou naprogramovány do počítače. Naší snahou je, aby odvozené metody byly nejen rychlé, potřebovaly co nejméně paměti, ale aby byly i použitelné pro ty úlohy, které jsou řešitelné.

Příklad: Řešte soustavu rovnic

$$\begin{array}{rcl} x + 2y + 3z &=& 14 \\ 2x + 4y + 5z &=& 25 \\ 7x + 8y + 9z &=& 50 \end{array}, \qquad \text{tj.} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 14 \\ 25 \\ 50 \end{bmatrix}$$

Řešení: Pro zápis budeme z důvodu přehlednosti používat tvar *matice rozšířené*:

$$\left(\begin{array}{ccc|c} \textcircled{1} & 2 & 3 & 14 \\ 2 & 4 & 5 & 25 \\ 7 & 8 & 9 & 50 \end{array}\right) \quad /\cdot(-\tfrac{2}{1}) \curvearrowright + \quad /\cdot(-\tfrac{7}{1}) \curvearrowright +$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 14 \\ 0 & \textcircled{0} & -1 & -3 \\ 0 & -6 & -12 & -48 \end{array}\right) \quad /\cdot(-\tfrac{6}{0}) \curvearrowright + \quad !!! \text{ dělíme 0}$$

$\Longrightarrow$ Algoritmus **Gaussovy eliminační metody** není pro tento příklad realizovatelný.

Snadno se přesvědčíme, že má daná soustava řešení $x = 1$, $y = 2$ a $z = 3$, ale Gaussova eliminační metoda selhala. □

Otázky:

1. Pro jaké matice $\mathbf{A}$ má soustava $\mathbf{Ax} = \mathbf{b}$ právě jedno řešení?

   → Matice $\mathbf{A}$ musí být **regulární**, tj. všechna vlastní čísla musí být různá od nuly.

   Poznámka: Vlastní číslo matice $\mathbf{A}$ je číslo λ splňující rovnici $\mathbf{Av} = \lambda\mathbf{v}$, kde $\mathbf{v}$ je vlastní vektor odpovídající vlastnímu číslu λ. Číslo λ tedy určitým způsobem charakterizuje matici $\mathbf{A}$.

2. Pro jaké matice $\mathbf{A}$ je algoritmus Gaussovy eliminační metody realizovatelný?

   → *Věta:* Je-li matice **A ostře diagonálně dominantní**, pak je algoritmus Gaussovy eliminační metody realizovatelný.

   Poznámka: Matice $\mathbf{A} = [a_{ij}]_{i,j=1,\ldots,n}$ je ostře diagonálně dominantní, platí-li $|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|$, tj. absolutní hodnota diagonálního prvku je větší než součet absolutních hodnot ostatních prvků v řádku.

   → *Věta:* Je-li matice **A symetrická** a **pozitivně definitní**, pak je algoritmus Gaussovy eliminační metody realizovatelný.

   Poznámka: Matice $\mathbf{A}$ je symetrická, platí-li pro její prvky $a_{ij} = a_{ji}\ \forall i, j = 1, 2, \ldots, n$.

   Poznámka: Matice $\mathbf{A}$ je pozitivně definitní, má-li všechna vlastní čísla kladná.

Poznámka: Pro soustavu s maticí, která splňuje předpoklady některé z uvedených vět, je možné dopředu říci, že půjde řešit pomocí Gaussovy eliminační metody. Obráceně to ovšem neplatí, tj. není-li např. matice soustavy ostře diagonálně dominantní, ještě to obecně neznamená, že nepůjde pomocí Gaussovy eliminační metody řešit.

Např. je-li $\mathbf{A} = \begin{bmatrix} \textcircled{7} & 1 & 2 \\ 1 & \textcircled{4} & 1 \\ 2 & 4 & \textcircled{9} \end{bmatrix}$, tj. diagonální prvky jsou vždy větší než součet zbylých prvků v řádku, pak půjde soustava s touto maticí řešit pomocí Gaussovy eliminační metody.

Abychom zaručili, že soustava půjde vyřešit pro libovolnou regulární matici, musíme algoritmus Gaussovy eliminační metody upravit. Zavedeme tzv. **výběr hlavního prvku (pivotaci)**.

Poznámka: pivot (hlavní prvek) ... první nenulový prvek v daném řádku matice.

Příklad: Pomocí **GEM se sloupcovou pivotací** vyřešte soustavu rovnic z předcházejícího příkladu, kde selhala klasická GEM, tj. řešíme soustavu $\mathbf{Ax} = \mathbf{b}$, kde

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} \quad \text{a} \quad \mathbf{b} = \begin{bmatrix} 14 \\ 25 \\ 50 \end{bmatrix}.$$

Řešení:

1. sloupec ↓

$$\text{vyměň} \left[\begin{array}{c} \rightarrow \\ \\ \rightarrow \end{array}\right. \left(\begin{array}{ccc|c} 1 & 2 & 3 & 14 \\ 2 & 4 & 5 & 25 \\ \textcircled{7} & 8 & 9 & 50 \end{array}\right) \quad \rightsquigarrow \quad \left(\begin{array}{ccc|c} 7 & 8 & 9 & 50 \\ 2 & 4 & 5 & 25 \\ 1 & 2 & 3 & 14 \end{array}\right) \quad \begin{array}{l} / \cdot (-\frac{2}{7}) \\ / \cdot (-\frac{1}{7}) \end{array} \left.\begin{array}{l} \\ \\ \end{array}\right) +$$

2. sloupec ↓

$$\begin{array}{c} \text{není} \\ \text{třeba} \\ \text{měnit} \end{array} \rightarrow \left(\begin{array}{ccc|c} 7 & 8 & 9 & 50 \\ 0 & \frac{12}{7} & \frac{17}{7} & \frac{75}{7} \\ 0 & \frac{6}{7} & \frac{12}{7} & \frac{48}{7} \end{array}\right) \quad / \cdot (-\frac{\frac{6}{7}}{\frac{12}{7}}) = -\frac{1}{2} \left.\begin{array}{l} \\ \\ \end{array}\right) +$$

$$\left(\begin{array}{ccc|c} 7 & 8 & 9 & 50 \\ 0 & \frac{12}{7} & \frac{17}{7} & \frac{75}{7} \\ 0 & 0 & \frac{1}{2} & \frac{3}{2} \end{array}\right) \quad \Longrightarrow \quad \mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

□

Poznámky:

- Při sloupcové pivotaci jsme postupně v každém sloupci (resp. jeho části pod diagonálou včetně) vybírali číslo, které bylo maximální v absolutní hodnotě a v případě, že toto číslo neleželo na diagonále, vyměnili jsme příslušné 2 rovnice. Dále jsme pokračovali jako v GEM bez pivotace, tj. nulovali jsme koeficienty pod diagonálou.
- Sloupcová pivotace není jediná možnost. Podobně můžeme vybírat i maximální prvek v absolutní hodnotě z příslušného řádku (resp. jeho části) a poté vyměnit příslušné sloupce. Pozor! Je ovšem třeba zaměnit i příslušné složky řešení $\mathbf{x}$. V tomto případě hovoříme o **řádkové pivotaci**.
- Další možností je vybírat maximální prvek v absolutní hodnotě z celé matice $\mathbf{A}$ (resp. příslušné podmatice). V tomto případě hovoříme o **úplné pivotaci**. Opět je třeba mít na paměti, že je třeba zaměnit složky ve vektoru řešení. Nevýhodou úplné pivotace je pomalejší výpočet neboť hlavní prvek vyhledáváme z celé dosud neupravené části.

Gaussova eliminační metoda není jedinou metodou pro řešení soustav lineárních algebraických rovnic. Další metodou je např. **metoda LU-rozkladu**.

Opět uvažujeme regulární matici $\mathbf{A}$ řádu n. Matici $\mathbf{A}$ lze rozložit na součin $\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$, kde $\mathbf{L}$ je dolní trojúhelníková matice řádu n a $\mathbf{U}$ je horní trojúhelníková matice řádu n.

Poznámka: Aby byl rozklad jednoznačný, předpokládáme na diagonále matice $\mathbf{L}$ jedničky.

Příklad LU-rozkladu matice $\mathbf{A}$

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 11 \\ 3 & 22 & 35 \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix}}_{\mathbf{L}} \cdot \underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}}_{\mathbf{U}}$$

□

Řešení soustavy $\mathbf{Ax} = \mathbf{b}$ metodou LU-rozkladu:

1) Realizace LU-rozkladu $\quad \mathbf{A} = \mathbf{L} \cdot \mathbf{U}$
2) Řešení soustavy (zpětný chod) $\quad \mathbf{Ly} = \mathbf{b}$
3) Řešení soustavy (zpětný chod) $\quad \mathbf{Ux} = \mathbf{y}$

$$\Bigg\} \qquad \mathbf{Ax} = \mathbf{L}\underbrace{\mathbf{Ux}}_{\mathbf{y}} = \mathbf{b}$$

Příklad: Řešte soustavu $\mathbf{Ax} = \mathbf{b}$ metodou LU-rozkladu, kde

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 11 \\ 3 & 22 & 35 \end{bmatrix} \quad \text{a} \quad \mathbf{b} = \begin{bmatrix} 13 \\ 45 \\ 133 \end{bmatrix}.$$

Řešení:

1. Z předchozího textu známe LU-rozklad matice $\mathbf{A}$.

2. Řešíme soustavu $\mathbf{Ly} = \mathbf{b}$

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & 13 \\ 2 & 1 & 0 & 45 \\ 3 & 4 & 1 & 133 \end{array}\right) \quad \Rightarrow \quad \mathbf{y} = \begin{bmatrix} 13 \\ 19 \\ 18 \end{bmatrix}.$$

3. Řešíme soustavu $\mathbf{Ux} = \mathbf{y}$

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 13 \\ 0 & 4 & 5 & 19 \\ 0 & 0 & 6 & 18 \end{array}\right) \quad \Rightarrow \quad \mathbf{x} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}.$$

□

Poznámka: Metoda LU-rozkladu je vhodná v případech, kdy řešíme více soustav se stejnou maticí $\mathbf{A}$, ale různými pravými stranami $\mathbf{b}$. (Důvod je ten, že jsme investovali práci do samotného LU-rozkladu. Vyřešení dvou soustav s trojúhelníkovou maticí je pouze věc dosazování.)

Poznámka: I v algoritmu LU-rozkladu lze využít pivotaci podobně jako v GEM.

Dosud jsme hovořili o tzv. **přímých metodách** (GEM, metoda LU-rozkladu), tzn. o metodách, které naleznou přesné řešení po konečném počtu kroků. Další skupinou metod pro řešení soustav rovnic jsou tzv. **iterační metody**, které *teoreticky* najdou přesné řešení až po nekonečně mnoha krocích.

Pamatujme si, že v numerické praxi používáme pro řešení soustav s plnou maticí *přímé metody*, zatímco pro speciální (řídké) matice používáme *iterační metody*.

Toto rozdělení je dáno **výpočetní složitostí** těchto metod, tj. počtem matematických operací sčítání, odčítání, násobení a dělení nutných k získání výsledku.

Poznámka: V případě plné matice je výpočetní cena v každé iteraci řádu n^2, srovnáme-li toto s celkovou výpočetní cenou přímých metod, tj. řádově $\frac{2}{3}n^3$, vidíme, že má-li být výpočetní složitost iterační metody stejná jako u přímé metody, musela by iterační metoda najít řešení (s předem zadanou přesností) řádově po n iteracích. Na druhou stranu v případě speciální (řídké) matice je výhodné použít iterační metodu.

V následujícím příkladu si ukážeme princip iteračních metod pro řešení soustav lineárních algebraických rovnic.

Příklad: Pomocí iterační metody řešte soustavu

$$\begin{array}{rcrcrcr} 11x & + & 2y & + & z & = & 15 \\ x & + & 10y & + & 2z & = & 16 \\ 2x & + & 3y & - & 8z & = & 1 \end{array}$$

Princip iteračních metod je založen na iterační formuli, tj. předpisu, kde se neznámé (resp. jedna z nich) vyskytují na obou stranách. Nejjednoduším způsobem jak získat iterační formuli je z i-té rovnice vyjádřit i-tou složku řešení, tzn. v našem případě z první rovnice vyjádříme x, z druhé rovnice vyjádříme y a ze třetí rovnice vyjádříme z. Dostaneme

$$\begin{aligned} x &= \frac{1}{11} \cdot (15 - 2y - z) \\ y &= \frac{1}{10} \cdot (16 - x - 2z) \\ z &= -\frac{1}{8} \cdot (1 - 2x - 3y) \end{aligned}$$

Tyto iterační formule můžeme zapsat jednodušším způsobem pomocí matice a vektorů

$$\underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & -\frac{2}{11} & -\frac{1}{11} \\ -\frac{1}{10} & 0 & -\frac{1}{5} \\ \frac{1}{4} & \frac{3}{8} & 0 \end{bmatrix}}_{\mathbf{H}} \cdot \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} \frac{15}{11} \\ \frac{8}{5} \\ -\frac{1}{8} \end{bmatrix}}_{\mathbf{g}} \qquad \text{tj.} \qquad \mathbf{x} = \mathbf{H} \cdot \mathbf{x} + \mathbf{g}$$

Abychom mohli podle této formule počítat, je třeba zvolit počáteční vektor $\mathbf{x^{(0)}}$, který dosadíme na pravou stranu formule. Získáme novou iteraci $\mathbf{x^{(1)}}$, tu opět dosadíme na pravou stranu formule a získáme $\mathbf{x^{(2)}}$, atd. Tento postup je pro zadané $\mathbf{x^{(0)}}$ popsán rekurentní formulí

$$\mathbf{x^{(k+1)}} = \mathbf{H} \cdot \mathbf{x^{(k)}} + \mathbf{g}.$$

Za vektor $\mathbf{x^{(0)}}$ můžeme volit libovolný vektor, nejjednodušší možností je tedy volit nulový vektor. Konkrétně v našem případě získáme následující iterace (pro jednoduchost zapisujeme na 4 desetinná místa)

$$\begin{aligned} \mathbf{x^{(0)}} &= [0.0000, 0.0000, 0.0000]^T \\ \mathbf{x^{(1)}} &= [1.3636, 1.6000, -0.1250]^T \\ \mathbf{x^{(2)}} &= [1.0841, 1.4886, 0.8159]^T \\ \mathbf{x^{(3)}} &= [1.0188, 1.3284, 0.7043]^T \\ \mathbf{x^{(4)}} &= [1.0581, 1.3573, 0.6279]^T \\ \mathbf{x^{(5)}} &= [1.0598, 1.3686, 0.6485]^T \end{aligned}$$

$$\mathbf{x^{(6)}} = [1.0558, 1.3643, 0.6532]^T$$
$$\mathbf{x^{(7)}} = [1.0562, 1.3638, 0.6506]^T$$
$$\mathbf{x^{(8)}} = [1.0565, 1.3643, 0.6505]^T$$
$$\mathbf{x^{(9)}} = [1.0565, 1.3643, 0.6507]^T$$
$$\mathbf{x^{(10)}} = [1.0564, 1.3642, 0.6507]^T$$
$$\underline{\mathbf{x^{(11)}} = [1.0564, 1.3642, 0.6507]^T}$$

Máme-li iterační předpis a první iteraci, můžeme začít počítat. Samozřejmě je třeba se zamyslet nad podmínkou pro zastavení výpočtu. Pro ukončení výpočtu budeme používat podmínku pro rozdíl dvou po sobě jdoucích iterací, kde tento rozdíl (resp. jeho norma) musí být menší než předem zadaná tolerance. V našem příkladu je z posloupnosti jednotlivých iterací zřejmé, že se 10-tá a 11-tá iterace shodují na 4 desetinná místa ve všech složkách, proto jsme výpočet ukončili a řekneme, že řešení soustavy $\mathbf{x} \approx \mathbf{x^{(11)}}$.

□

Metoda, kterou jsme odvodili v předchozím příkladu, se nazývá **Jacobiova metoda**.

Poznámka:
Uvědomme si souvislost s řešením obecné soustavy nelineárních rovnic $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ pomocí metody prosté iterace. Rovnici jsme si vektorově přepsali na tvar $\mathbf{x} = \mathbf{\Phi}(\mathbf{x})$. Uvažujeme-li soustavu lineárních algebraických rovnic, tj. funkce $\mathbf{F}$ je lineární

$$\underbrace{\mathbf{Ax} - \mathbf{b}}_{\mathbf{F(x)}} = \mathbf{0}$$

můžeme potom najít lineární předpis pro funkci $\mathbf{\Phi}$

$$\mathbf{x} = \underbrace{\mathbf{Hx} + \mathbf{g}}_{\mathbf{\Phi(x)}}$$

Všechny iterační metody pro řešení soustavy lineárních algebraických rovnic budou používat iterační formuli ve tvaru

$$\mathbf{x^{(k+1)}} = \mathbf{H} \cdot \mathbf{x^{(k)}} + \mathbf{g},$$

samozřejmě s různou iterační maticí $\mathbf{H}$ a vektorem $\mathbf{g}$ a je zřejmé, že o kvalitě metody budou rozhodovat právě vlastnosti matice $\mathbf{H}$.

Nyní si stručně zformulujeme principy některých iteračních metod.

Jacobiova metoda

Princip: Z i-té rovnice vyjádříme i-tou složku vektoru $\mathbf{x}$

i-tá rovnice: $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i$

pro $a_{ii} \neq 0$: $x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1 \ j\neq i}^{n} a_{ij}x_j \right)$

Iterační formule:

$$\boxed{x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1 \ j\neq i}^{n} a_{ij}x_j^{(k)} \right)}$$

Gaussova-Seidelova metoda

Princip: Stejný jako u Jacobiovy metody s tím rozdílem, že jestliže při výpočtu $(k+1)$-iterace již známe $(k+1)$-iteraci některých složek, tak ji použijeme.

Iterační formule:

$$\boxed{x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right)}$$

Relaxační metoda SOR

Princip: Vyjdeme z Gaussovy-Seidelovy metody jejíž iterační formulu lze psát takto:

$$x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)} \quad \text{kde} \quad r_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)} \right)$$

Abychom urychlili výpočet, nebudeme přičítat $r_i^{(k)}$, ale $\omega r_i^{(k)}$, tj. $x_i^{(k+1)} = x_i^{(k)} + \omega r_i^{(k)}$
Iterační formule:

$$\boxed{x_i^{(k+1)} = x_i^{(k)} + \omega \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)} \right)}$$

Poznámka: $(k+1)$-iterace metody SOR je lineární kombinací $(k+1)$-iterace získané Gauss-Seidlovou metodou a předchozí k-té iterace.

$$x_i^{(k+1)} = \omega \underbrace{\frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right)}_{x_i^{(k+1)} \text{ z Gaussovy-Seidelovy metody}} + (1-\omega)x_i^{(k)}$$

Uvedené iterační formule lze přepsat do maticové podoby. Nejprve rozložíme matici $\mathbf{A}$:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U},$$

kde $\mathbf{L}$ je dolní trojúhelníková část matice $\mathbf{A}$ s nulami na diagonále, $\mathbf{D}$ je diagonální matice a $\mathbf{U}$ je horní trojúhelníková část matice $\mathbf{A}$ s nulami na diagonále.

Jacobiova metoda:

$$\begin{gathered}
\mathbf{A}\mathbf{x} = \mathbf{b} \\
(\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{x} = \mathbf{b} \\
\mathbf{D}\mathbf{x} + (\mathbf{L} + \mathbf{U})\mathbf{x} = \mathbf{b} \\
\mathbf{D}\mathbf{x} = \mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x} \\
\hline
\mathbf{x} = \underbrace{-\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})}_{\mathbf{H}_J}\mathbf{x} + \underbrace{\mathbf{D}^{-1}\mathbf{b}}_{\mathbf{g}_J}
\end{gathered}$$

Gauss-Seidlova metoda:

$$\begin{gathered}
\mathbf{A}\mathbf{x} = \mathbf{b} \\
(\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{x} = \mathbf{b} \\
(\mathbf{L} + \mathbf{D})\mathbf{x} + \mathbf{U}\mathbf{x} = \mathbf{b} \\
(\mathbf{L} + \mathbf{D})\mathbf{x} = \mathbf{b} - \mathbf{U}\mathbf{x} \\
\hline
\mathbf{x} = \underbrace{-(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}}_{\mathbf{H}_{GS}}\mathbf{x} + \underbrace{(\mathbf{L} + \mathbf{D})^{-1}\mathbf{b}}_{\mathbf{g}_{GS}}
\end{gathered}$$

Relaxační metoda SOR:

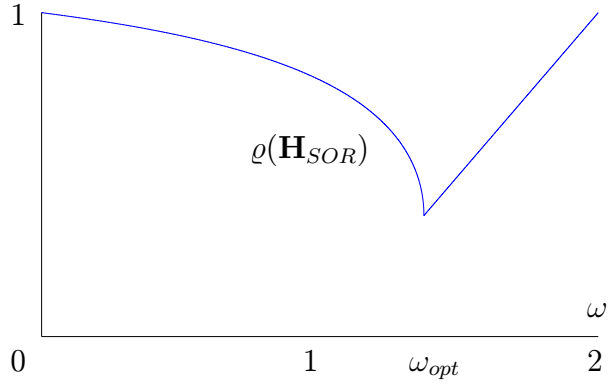
$$\begin{gathered}
\mathbf{A}\mathbf{x} = \mathbf{b} \\
\omega\mathbf{A}\mathbf{x} = \omega\mathbf{b} \qquad / + \mathbf{D}\mathbf{x} \\
(\omega\mathbf{A} + \mathbf{D})\mathbf{x} = \omega\mathbf{b} + \mathbf{D}\mathbf{x} \\
\left[\omega(\mathbf{L} + \mathbf{D} + \mathbf{U}) + \mathbf{D}\right]\mathbf{x} = \omega\mathbf{b} + \mathbf{D}\mathbf{x} \\
(\omega\mathbf{L} + \mathbf{D})\mathbf{x} = \omega\mathbf{b} + \mathbf{D}\mathbf{x} - \omega\mathbf{D}\mathbf{x} - \omega\mathbf{U}\mathbf{x} \\
(\omega\mathbf{L} + \mathbf{D})\mathbf{x} = \left[(1-\omega)\mathbf{D} - \omega\mathbf{U}\right]\mathbf{x} + \omega\mathbf{b} \\
\hline
\mathbf{x} = \underbrace{(\omega\mathbf{L} + \mathbf{D})^{-1}\left[(1-\omega)\mathbf{D} - \omega\mathbf{U}\right]}_{\mathbf{H}_{SOR}}\mathbf{x} + \underbrace{(\omega\mathbf{L} + \mathbf{D})^{-1}\omega\mathbf{b}}_{\mathbf{g}_{SOR}}
\end{gathered}$$

Poznámka:
Parametr ω v relaxační metodě SOR volíme z intervalu $(0, 2)$. Pro $\omega = 1$ přejde relaxační metoda na Gauss-Seidlovu metodu. Volba parametru ω samozřejmě ovlivní rychlost konvergence iteračního procesu metody SOR. Lze ukázat, že existuje optimální hodnota parametru omega

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \varrho^2}}, \quad \text{kde } \varrho \text{ je spektrální poloměr Jacobiovy iterační matice } \mathbf{H}_J.$$

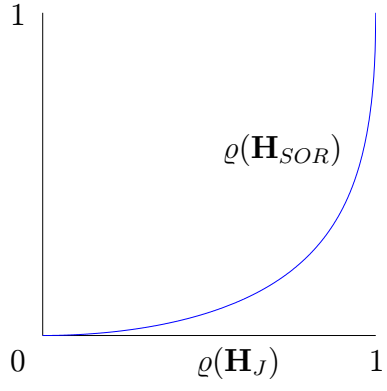
(spektrální poloměr = maximální vlastní číslo v absolutní hodnotě)

Pro spektrální poloměr iterační matice $\mathbf{H}_{SOR}$ relaxační metody lze odvodit následující závislosti:



Obr. 1

Závislost spektrálního poloměru iterační matice metody SOR na relaxačním parametru ω



Obr. 2

Závislost spektrálního poloměru matice metody SOR na spektrálním poloměru iterační matice Jacobiovy metody

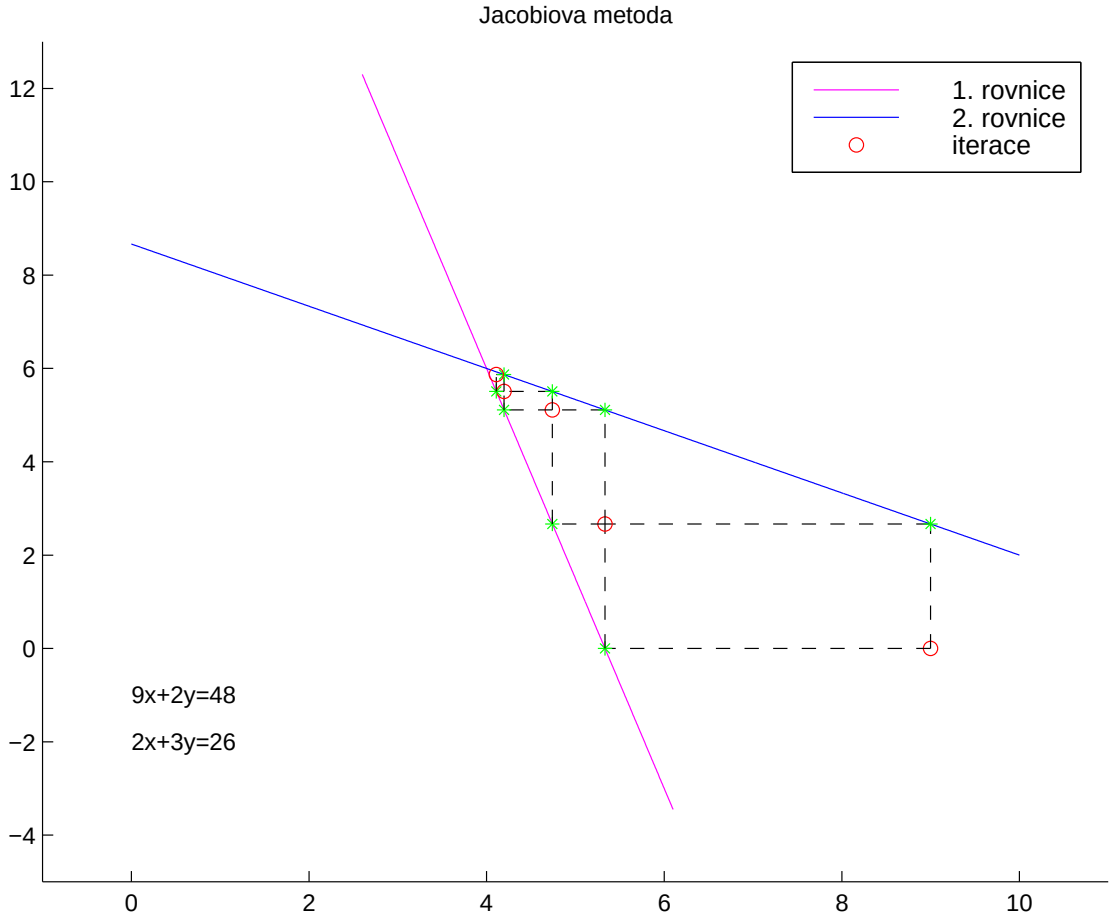
Věta: Konzistentní iterační metoda daná předpisem $\mathbf{x}^{(k+1)} = \mathbf{H}\mathbf{x}^{(k)} + \mathbf{g}$ konverguje pro libovolnou počáteční iteraci právě tehdy, když $\varrho(\mathbf{H}) < 1$.

Poznámka: Konzistentní metoda splňuje podmínku $\mathbf{A}^{-1}\mathbf{b} = \mathbf{H}\mathbf{A}^{-1}\mathbf{b} + \mathbf{g}$.

Poznámka: Čím je menší spektrální poloměr iterační matice $\mathbf{H}$, tím rychleji metoda konverguje.

Pro soustavu 2 rovnic pro 2 neznámé si lze princip uvedených iteračních metod znázornit geometricky.

GEOMETRICKÝ VÝZNAM JACOBIOVY METODY

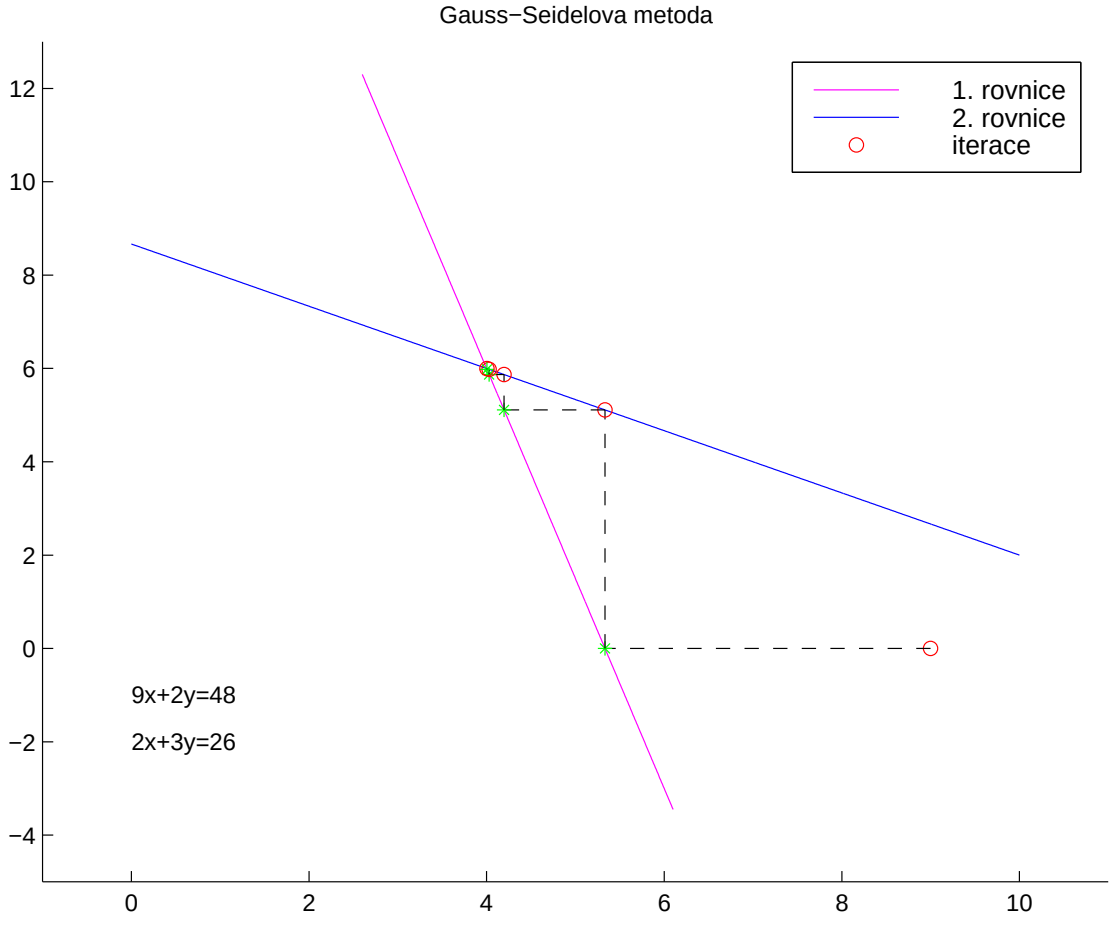


$$x^{(k+1)} = \frac{1}{9}(48 - 2y^{(k)})$$

$$y^{(k+1)} = \frac{1}{3}(26 - 2x^{(k)})$$

k	$x^{(k)}$	$y^{(k)}$
0	9.0000	0
1	5.3333	2.6667
2	4.7407	5.1111
3	4.1975	5.5062
4	4.1097	5.8683
5	4.0293	5.9268

GEOMETRICKÝ VÝZNAM GAUSSOVY-SEIDELOVY METODY

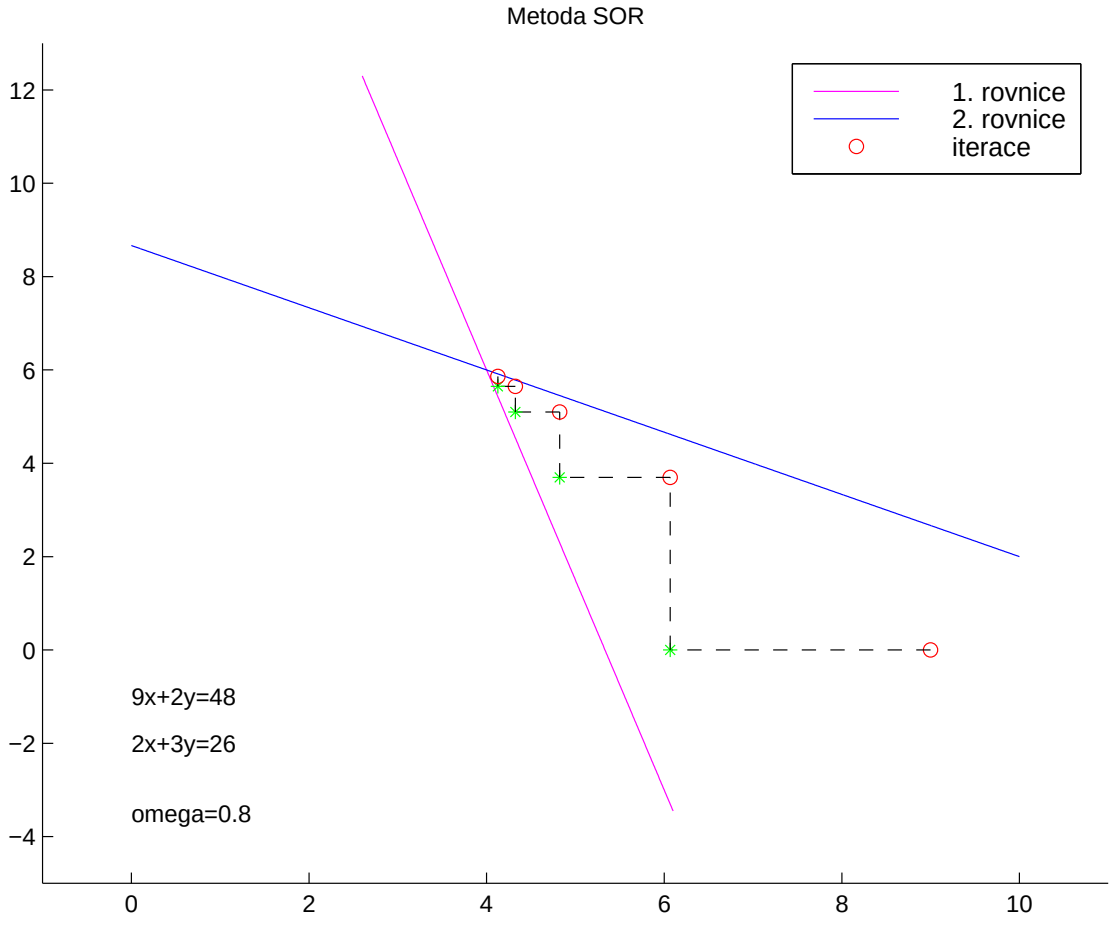


$$x^{(k+1)} = \frac{1}{9}(48 - 2y^{(k)})$$

$$y^{(k+1)} = \frac{1}{3}(26 - 2x^{(k+1)})$$

k	$x^{(k)}$	$y^{(k)}$
0	9.0000	0
1	5.3333	5.1111
2	4.1975	5.8683
3	4.0293	5.9805
4	4.0043	5.9971
5	4.0006	5.9996

GEOMETRICKÝ VÝZNAM METODY SOR ($\omega = 0.8$)

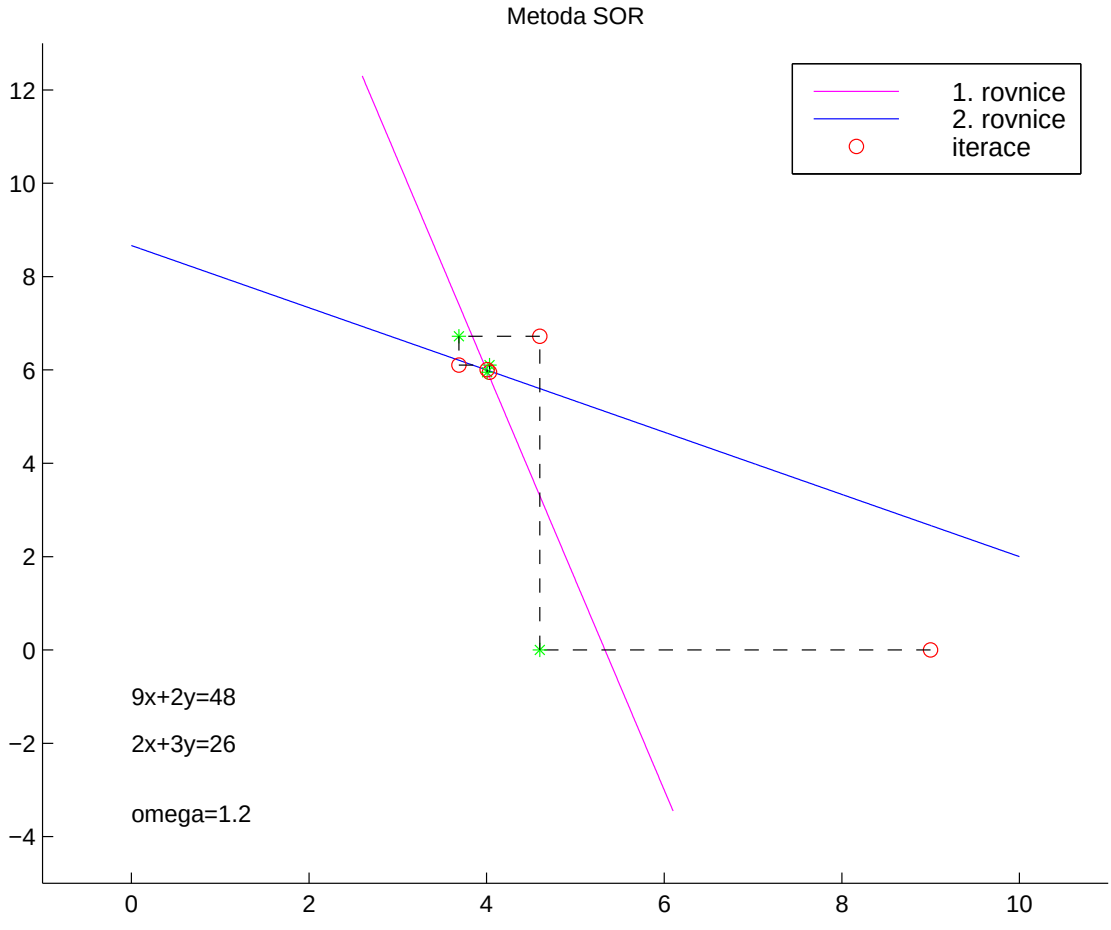


$$x^{(k+1)} = \omega\frac{1}{9}(48 - 2y^{(k)}) + (1-\omega)x^{(k)}$$

$$y^{(k+1)} = \omega\frac{1}{3}(26 - 2x^{(k+1)}) + (1-\omega)y^{(k)}$$

k	$x^{(k)}$	$y^{(k)}$
0	9.0000	0
1	6.0667	3.6978
2	4.8226	5.1008
3	4.3244	5.6472
4	4.1276	5.8614
5	4.0502	5.9455

GEOMETRICKÝ VÝZNAM METODY SOR ($\omega = 1.2$)



$$x^{(k+1)} = \omega\frac{1}{9}(48 - 2y^{(k)}) + (1-\omega)x^{(k)}$$

$$y^{(k+1)} = \omega\frac{1}{3}(26 - 2x^{(k+1)}) + (1-\omega)y^{(k)}$$

k	$x^{(k)}$	$y^{(k)}$
0	9.0000	0
1	4.6000	6.7200
2	3.6880	6.1056
3	4.0342	5.9515
4	4.0061	6.0048
5	3.9975	6.0010

Uveďme nyní speciální typ iteračních metod pro řešení soustav lineárních algebraických rovnic, tzv. **gradientní metody**. Jako motivace nám může posloužit příklad pro funkci jedné reálné proměnné. Uvažujme kvadratickou funkci

$$f(x) = \frac{1}{2}ax^2 - bx + c, \quad a > 0.$$

Nutná a postačující podmínka minima má tvar

$$ax = b.$$

To znamená, že místo řešení rovnice můžeme řešit úlohu najít minimum konvexní kvadratické funkce $f(x)$ (obě úlohy mají stejné řešení). Uvědomme si, že v případě funkce více proměnných je třeba splnit další podmínky kladené na matici soustavy $\mathbf{A}$, abychom zaručili konvexnost příslušné kvadratické funkce.

Uvažujeme soustavu

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

kde matice $\mathbf{A}$ je symetrická, pozitivně definitní.

Dále uvažujeme kvadratickou formu, tzv. **energetický funkcionál**

$$\mathbf{F}(\mathbf{x}) = \frac{1}{2}\,\mathbf{x}^T\mathbf{A}\,\mathbf{x} - \mathbf{b}^T\mathbf{x}.$$

Platí

$$\text{grad}\ \mathbf{F}(\mathbf{x}) = \mathbf{A}\,\mathbf{x} - \mathbf{b}.$$

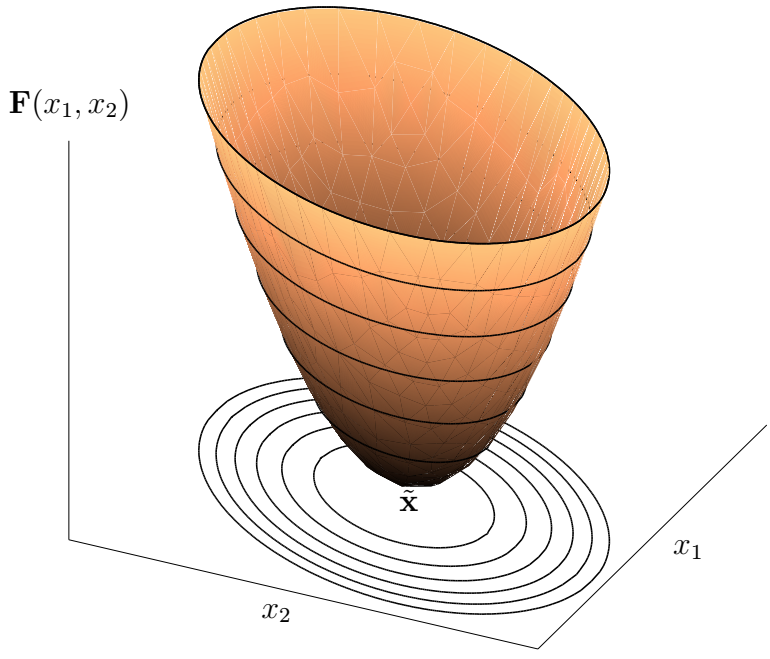
Funkce $\mathbf{F}(\mathbf{x})$ je konvexní a kvadratická $\Rightarrow$ $\mathbf{F}(\mathbf{x})$ má globální minimum. Pro bod minima $\tilde{\mathbf{x}}$ platí

$$\text{grad}\ \mathbf{F}(\tilde{\mathbf{x}}) = \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b} = \mathbf{0}.$$

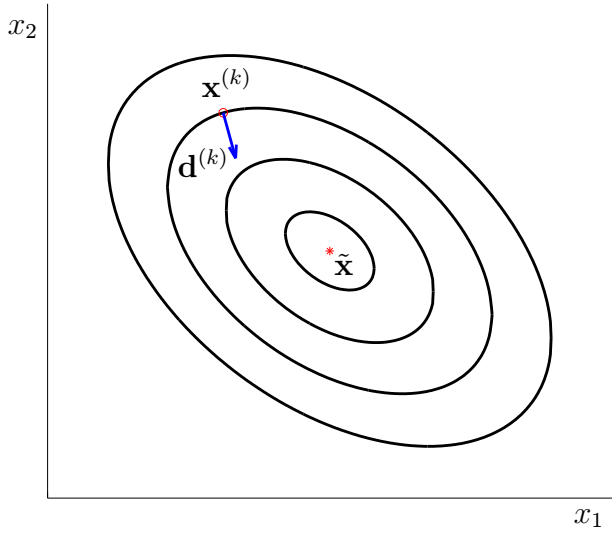
Bod minima $\tilde{\mathbf{x}}$ je tedy řešením soustavy $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Poznámka:
Úlohy najít bod minima funkce $\mathbf{F}$ a řešit soustavu $\mathbf{A}\mathbf{x} = \mathbf{b}$ jsou ekvivalentní.

Poznámka: V případě soustavy 2 rovnic si lze udělat geometrickou představu, neboť pro $\mathbf{x} \in \mathbb{R}^2$ je grafem funkce $\mathbf{F}(\mathbf{x})$ eliptický paraboloid, jehož vrstevnice jsou elipsy. Minima $\mathbf{F}(\mathbf{x})$ se nabývá ve vrcholu paraboloidu.



Stejně jako u každé iterační metody nejprve zvolíme počáteční aproximaci řešení $\mathbf{x}^{(0)}$. Princip gradientních metod spočívá v tom, že zvolíme směr a v tomto směru se budeme chtít co nejvíce přiblížit k přesnému řešení. Gradientní metoda je tedy dána volbou směrů, ve kterých minimalizujeme funkci $\mathbf{F}$.



V případě soustavy dvou rovnic získáme promítnutím grafu funkce $\mathbf{F}(\mathbf{x})$ do roviny proměnných x_1, x_2 systém soustředných elips - hladin (vrstevnic).

První možností je za směrový vektor volit směr největšího spádu, tj. vektor

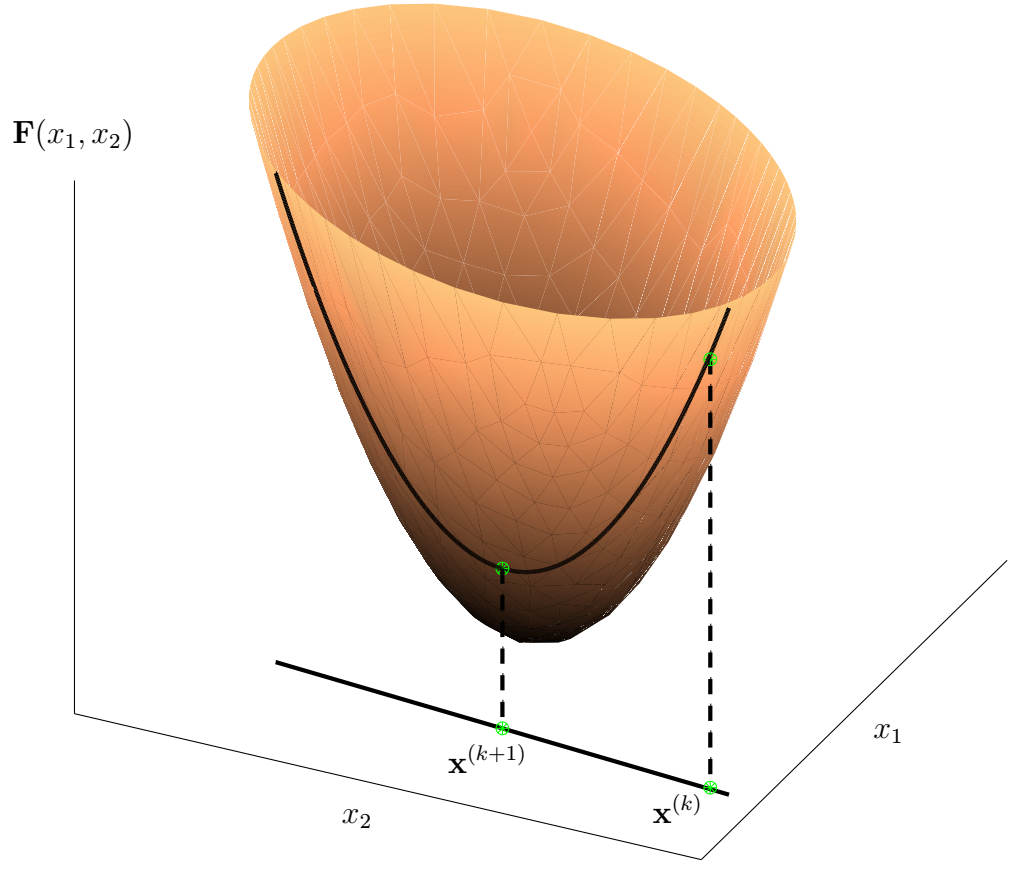
$$\mathbf{d}^{(k)} = -\operatorname{grad} \mathbf{F}(\mathbf{x}^{(k)}) = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}.$$

Získáme tzv. **metodu největšího spádu**. Iterační formuli volíme ve tvaru

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}.\ \mathbf{d}^{(k)},$$

v každém kroku metody určíme směr největšího spádu $\mathbf{d}^{(k)}$ a provedeme jednorozměrnou minimalizaci v tomto směru, tj.

$$\min_{t>0} \mathbf{F}(\mathbf{x}^{(k)} + t\,\mathbf{d}^{(k)}).$$



Minimalizovanou funkci proměnné t označíme $\Psi(t)$. Platí:

$$\Psi(t) = \mathbf{F}(\mathbf{x}^{(k)} + t\,\mathbf{d}^{(k)}) = \frac{1}{2}\,(\mathbf{x}^{(k)} + t\,\mathbf{d}^{(k)})^T\mathbf{A}(\mathbf{x}^{(k)} + t\,\mathbf{d}^{(k)}) - \mathbf{b}^T(\mathbf{x}^{(k)} + t\,\mathbf{d}^{(k)}).$$

$$\frac{d\Psi(t)}{dt} = t\,{\mathbf{d}^{(k)}}^T\mathbf{A}\mathbf{d}^{(k)} + \underbrace{{\mathbf{x}^{(k)}}^T\mathbf{A}\mathbf{d}^{(k)} - \mathbf{b}^T\mathbf{d}^{(k)}}_{\underbrace{({\mathbf{x}^{(k)}}^T\mathbf{A} - \mathbf{b}^T)}_{-{\mathbf{d}^{(k)}}^T}\,\mathbf{d}^{(k)}}$$

$$\frac{d\Psi(t)}{dt} = t\,{\mathbf{d}^{(k)}}^T\mathbf{A}\mathbf{d}^{(k)} - {\mathbf{d}^{(k)}}^T\,\mathbf{d}^{(k)} = 0$$

$$t^{(k)} = \frac{{\mathbf{d}^{(k)}}^T\,\mathbf{d}^{(k)}}{{\mathbf{d}^{(k)}}^T\mathbf{A}\,\mathbf{d}^{(k)}}$$

Poznámka: Ve výrazu pro derivaci $\Psi(t)$ jsme využili symetrii matice $\mathbf{A}$.

Algoritmus metody největšího spádu potom můžeme zapsat takto:

1) volba $\mathbf{x}^{(0)}$, ε

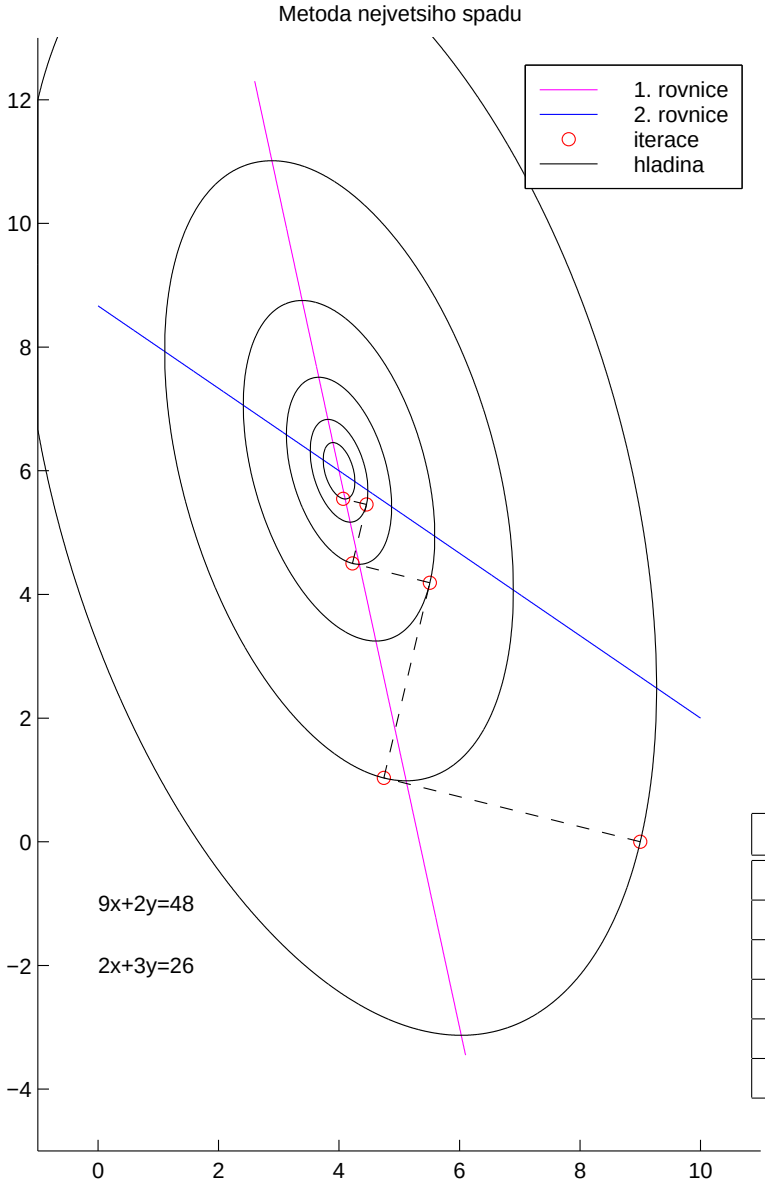
2) výpočet směru spádu $\mathbf{d}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$

3) výpočet koeficientu $t^{(k)} = \dfrac{{\mathbf{d}^{(k)}}^T\,\mathbf{d}^{(k)}}{{\mathbf{d}^{(k)}}^T\mathbf{A}\,\mathbf{d}^{(k)}}$

4) výpočet nové iterace $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\mathbf{d}^{(k)}$

5) $k = k + 1$ a zpět na 2) pokud $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| > \varepsilon$

GEOMETRICKÝ VÝZNAM METODY NEJVĚTŠÍHO SPÁDU



k	$x^{(k)}$	$y^{(k)}$
0	9.0000	0
1	4.7425	1.0321
2	5.5081	4.1902
3	4.2240	4.5015
4	4.4549	5.4541
5	4.0676	5.5480

Pro soustavu dvou rovnic si opět lze udělat geometrickou představu. Všimněme si faktu, že vždy po sobě jdoucí iterace směru spádu, tj. $\mathbf{d}^{(k)}$ a $\mathbf{d}^{(k+1)}$ jsou na sebe kolmé.

Cvičení: Dokažte, že platí $\mathbf{d}^{(k)^T}\mathbf{d}^{(k+1)} = 0$

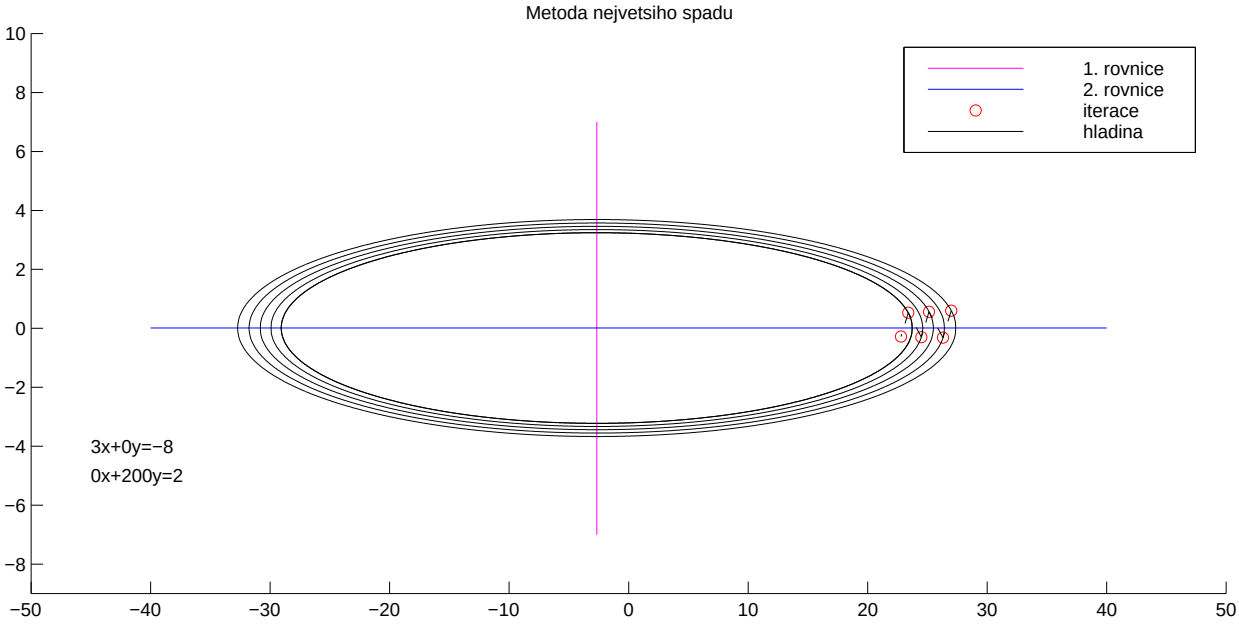
$$\begin{aligned}
\mathbf{d}^{(k)^T}\mathbf{d}^{(k+1)} &= \mathbf{d}^{(k)^T}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)}) = \\
&= \mathbf{d}^{(k)^T}(\mathbf{b} - \mathbf{A}(\mathbf{x}^{(k)} + t^{(k)}\mathbf{d}^{(k)})) = \\
&= \mathbf{d}^{(k)^T}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)} - t^{(k)}\mathbf{A}\,\mathbf{d}^{(k)}) = \\
&= \mathbf{d}^{(k)^T}(\mathbf{d}^{(k)} - t^{(k)}\mathbf{A}\,\mathbf{d}^{(k)}) = \\
&= \mathbf{d}^{(k)^T}\mathbf{d}^{(k)} - t^{(k)}\mathbf{d}^{(k)^T}\mathbf{A}\,\mathbf{d}^{(k)} = \\
&= \mathbf{d}^{(k)^T}\mathbf{d}^{(k)} - \frac{\mathbf{d}^{(k)^T}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)^T}\mathbf{A}\,\mathbf{d}^{(k)}}\,\mathbf{d}^{(k)^T}\mathbf{A}\,\mathbf{d}^{(k)} = \\
&= \mathbf{d}^{(k)^T}\mathbf{d}^{(k)} - \mathbf{d}^{(k)^T}\mathbf{d}^{(k)} = 0
\end{aligned}$$

Poznámka: V případě, že budou hladiny (elipsy) „velmi protáhlé“, bude metoda největšího spádu konvergovat velmi pomalu, nastane tzv. *cik-cak efekt*. Na druhou stranu, pokud budou hladiny (elipsy) „skoro kružnice“, bude metoda největšího spádu konvergovat velmi rychle. Nevýhodu cik-cak efektu odstraňuje nová metoda, tzv. **metoda sdružených gradientů**, která využívá důmyslnější volby směrů minimalizace, a sice tak, aby se neopakovali, jak k tomu docházelo u metody největšího spádu.

Příklad 1. Pomocí metody největšího spádu řešte soustavu

$$3x + 0y = -8$$
$$0x + 200y = 2$$

Jako počáteční aproximaci volte $x^{(0)} = 27$, $y^{(0)} = 0.6$.



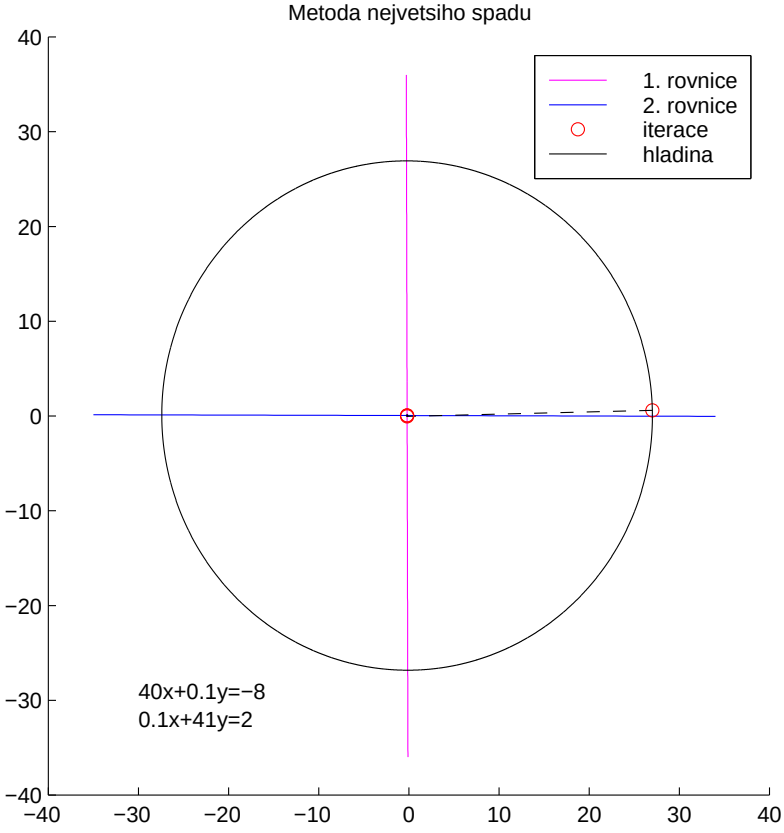
k	$x^{(k)}$	$y^{(k)}$
0	27.000000	0.600000
1	26.307758	-0.317804
2	25.139940	0.563008
3	24.491101	-0.297251
4	23.396504	0.528335
5	22.788346	-0.277987
⋮	⋮	⋮
250	-2.657606	0.010180

(Přesné řešení soustavy je $[\tilde{x}, \tilde{y}] = \left[-\frac{8}{3}, \frac{1}{100}\right]$.)

Příklad 2. Pomocí metody největšího spádu řešte soustavu

$$40x + 0.1y = -8$$
$$0.1x + 41y = 2$$

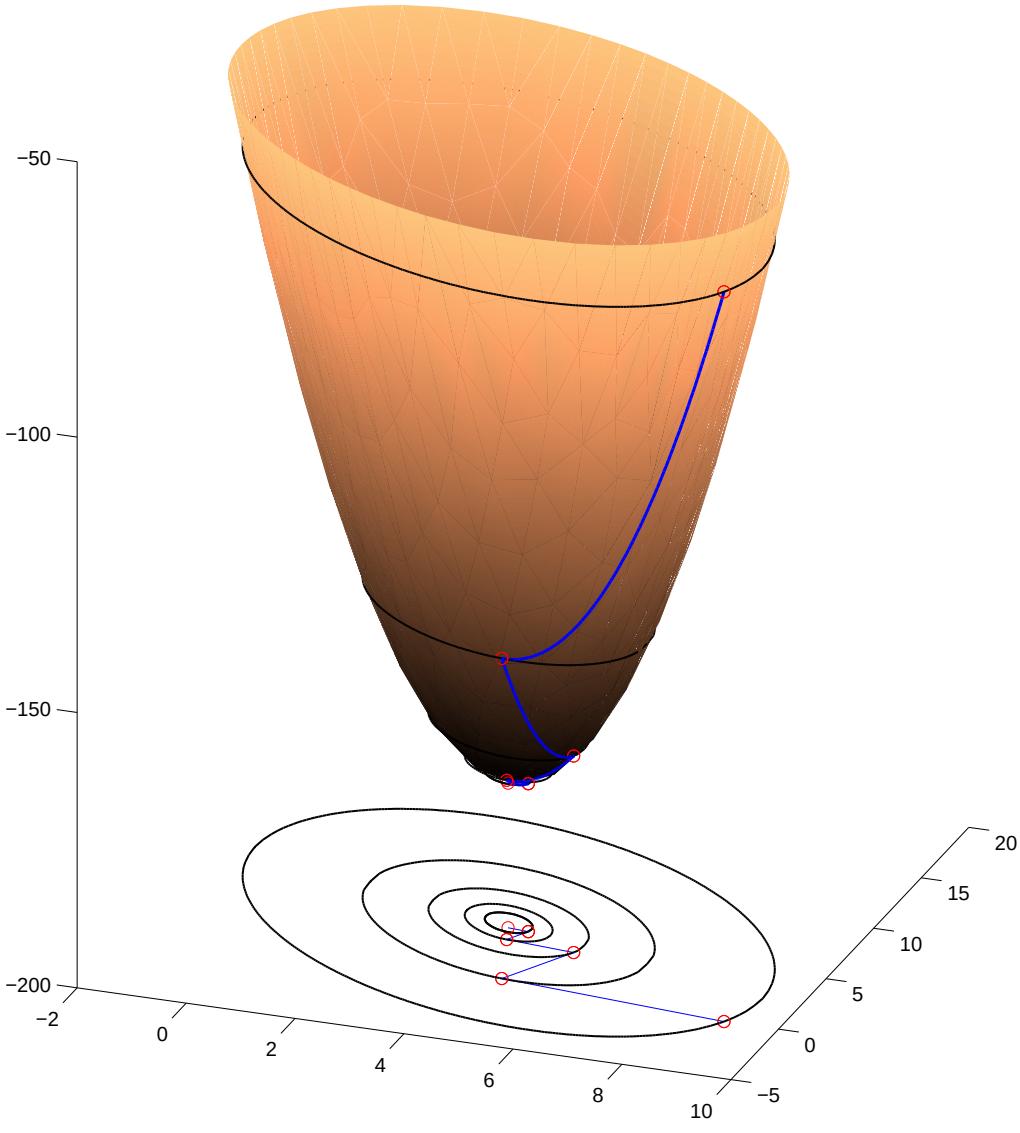
Jako počáteční aproximaci volte $x^{(0)} = 27$, $y^{(0)} = 0.6$.



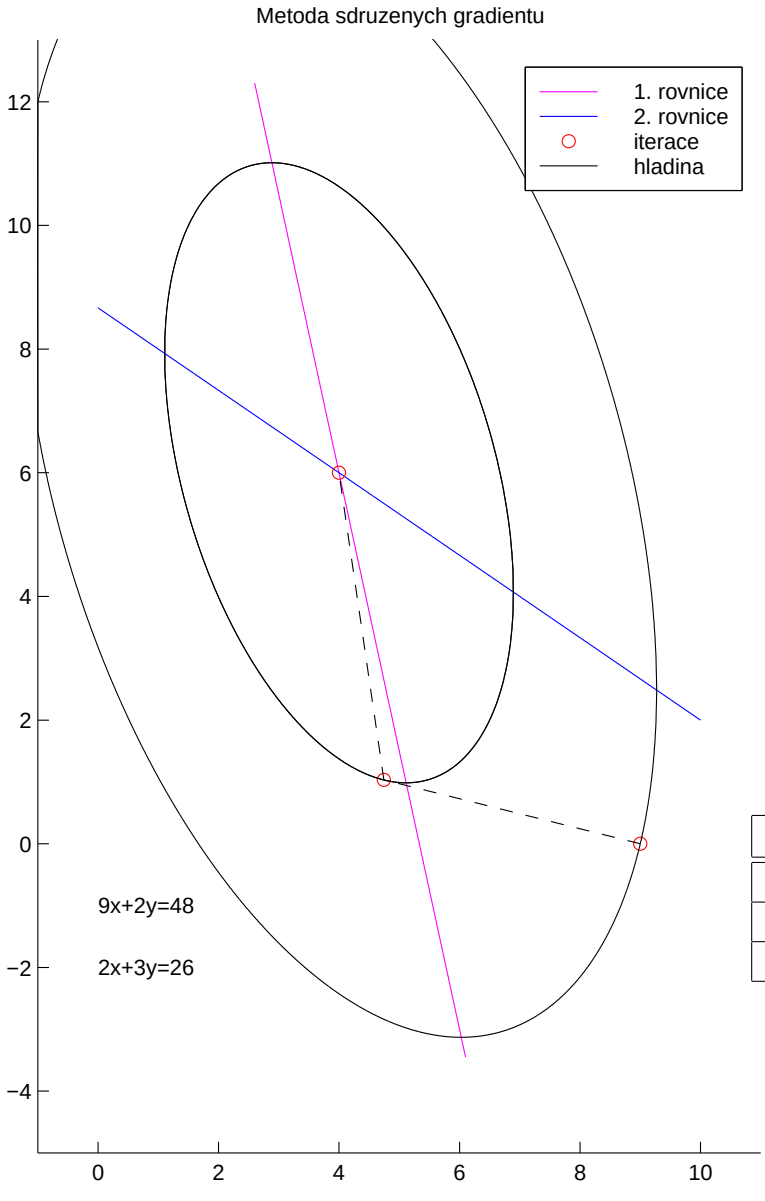
k	$x^{(k)}$	$y^{(k)}$
0	27.000000	0.600000
1	-0.197972	-0.032418
2	-0.199872	0.049274
3	-0.200123	0.049268

(Přesné řešení soustavy se na 6 desetiných míst shoduje s $[x^{(3)}, y^{(3)}]$.)

GEOMETRICKÝ VÝZNAM METODY NEJVĚTŠÍHO SPÁDU VE 3D

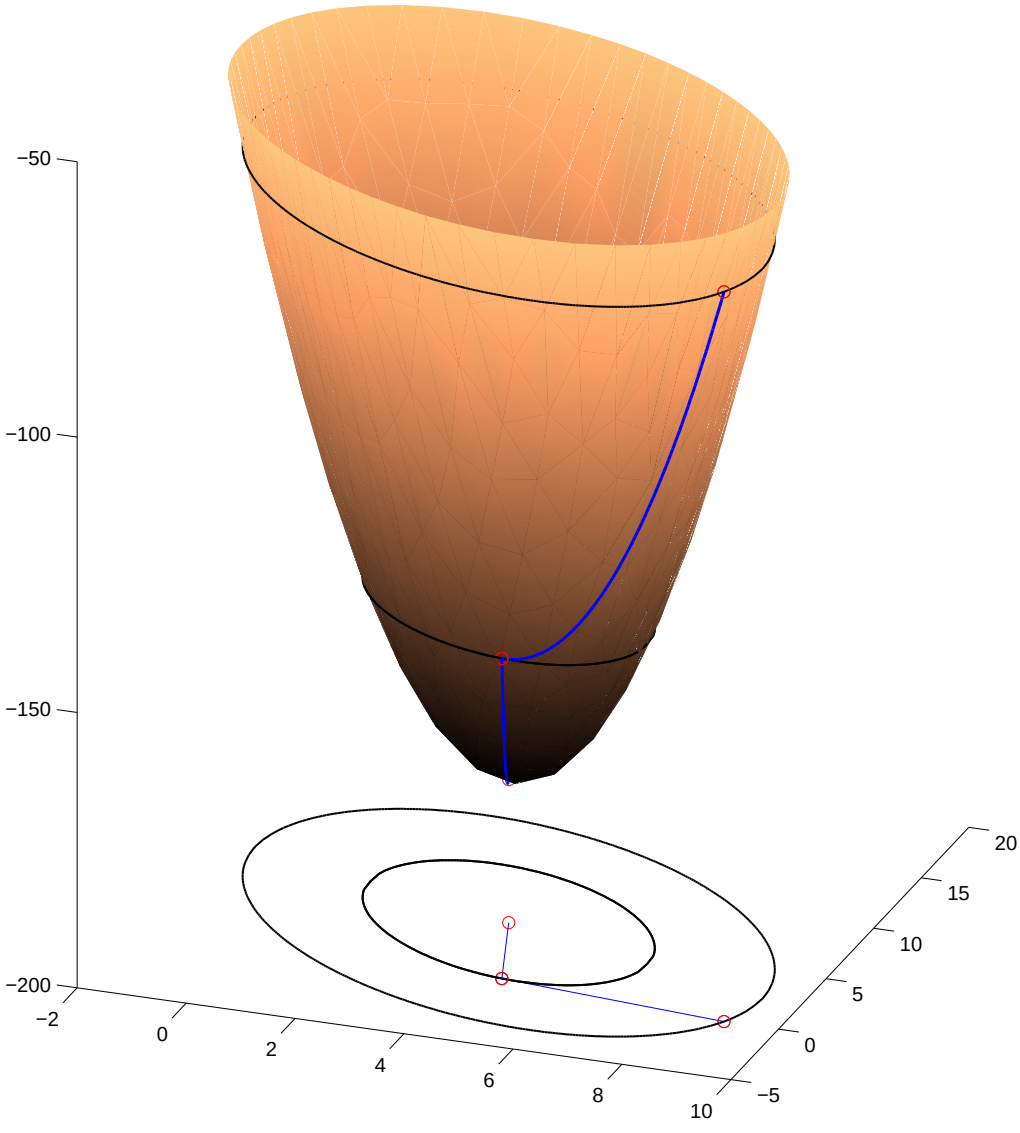


GEOMETRICKÝ VÝZNAM METODY SDRUŽENÝCH GRADIENTŮ



k	$x^{(k)}$	$y^{(k)}$
0	9.0000	0
1	4.7425	1.0321
2	4.0000	6.0000

GEOMETRICKÝ VÝZNAM METODY SDRUŽENÝCH GRADIENTŮ VE 3D



Výpočet vlastních čísel a vlastních vektorů

S pojmem *vlastního čísla* jsme se již setkali například u iteračních metod pro řešení soustavy lineárních algebraických rovnic. Velikosti vlastních čísel iterační matice rozhodovaly o konvergenci příslušné iterační metody. S úlohou na vlastní čísla se setkáme i v aplikacích při řešení řady technických a fyzikálních problémů.

Úlohu na vlastní čísla si připomeneme na příkladu.

Příklad: Stanovte taková čísla λ, pro která má homogenní soustava $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ nenulové řešení, a určete toto řešení, kde matice

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 0 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}.$$

Řešíme tedy soustavu

$$(\mathbf{A} - \lambda\mathbf{I})\,\mathbf{v} = \begin{bmatrix} 2-\lambda & 0 & 0 \\ 2 & 2-\lambda & 1 \\ 1 & 1 & 2-\lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Aby homogenní soustava měla nenulové řešení, musí být determinant soustavy nulový. Hledáme proto taková λ, aby

$$\det(\mathbf{A} - \lambda I) = -\lambda^3 + 6\lambda^2 - 11\lambda + 6 = 0.$$

Dostali jsme algebraickou rovnici stupně 3 a pouze pro její kořeny

$$\lambda_1 = 3, \qquad \lambda_2 = 2, \qquad \lambda_3 = 1$$

nazývané *vlastní čísla* matice $\mathbf{A}$, bude mít uvažovaná soustava nenulové řešení.

Ke každému vlastnímu číslu λ_i můžeme najít nenulové řešení homogennísoustavy

$$(\mathbf{A} - \lambda_i I)\,\mathbf{v} = \mathbf{0}.$$

Např. pro $\lambda_1 = 3$ řešíme soustavu

$$\begin{bmatrix} -1 & 0 & 0 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Matice soustavy je samozřejmě singulární a proto bude existovat celý systém řešení v závislosti na parametru $r \in \mathbb{R}$. Každý vektor $[0, r, r]^T$ řeší danou soustavu. Ze systému vybereme jednoho zástupce, např. $\mathbf{v}^{(1)} = [0, 1, 1]^T$, a říkáme, že $\mathbf{v}^{(1)}$ je *vlastní vektor* odpovídající vlastnímu číslu λ_1. Podobně bychom nalezli vlastní vektory odpovídající vlastním číslům λ_2 a λ_3. □

Definice: Je dána čtvercová matice $\mathbf{A}$ řádu n. *Vlastní čísla* $\lambda_1, \lambda_2, \ldots, \lambda_n$ jsou kořeny *charakteristické rovnice*

$$p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}) = 0.$$

Ke každému vlastnímu číslu λ_i existuje alespoň jedno nenulové řešení soustavy rovnic $\mathbf{Av} = \lambda_\mathbf{i}\mathbf{v}$. Toto řešení $\mathbf{v}$ nazveme *vlastním vektorem.*

Poznámka: Charakteristický polynom je stupně $n \Rightarrow \exists\, n$ vlastních čísel.

Definice: Matici $\mathbf{\Lambda} = \mathrm{diag}(\lambda_\mathbf{1}, \lambda_\mathbf{2}, \ldots, \lambda_\mathbf{n})$ nazýváme *spektrální maticí* matice $\mathbf{A}$.

Poznámka: Vlastní čísla horní trojúhelníkové matice jsou rovna jejím diagonálním prvkům, neboť charakteristický polynom má tvar:

$$p_{\mathbf{A}}(\lambda) = (a_{11} - \lambda)(a_{22} - \lambda)\ldots(a_{nn} - \lambda).$$

Úlohy na nalezení vlastních čísel rozdělíme do dvou skupin:

Úplný problém vl. čísel – úloha najít všechna vlastní čísla

Částečný problém vl. čísel – úloha najít pouze některá vl. čísla (obvykle s nejmenší nebo největší abs. hodnotou).

Příklad: Určete vlastní čísla a vlastní vektory těchto matic:

$$\mathbf{A} = \left[\begin{array}{c|c|c} 2 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 2 \end{array}\right], \quad \mathbf{B} = \left[\begin{array}{cc|c} 2 & 1 & 0 \\ 0 & 2 & 0 \\ \hline 0 & 0 & 2 \end{array}\right],$$

$$\mathbf{C} = \left[\begin{array}{c|cc} 2 & 0 & 0 \\ \hline 0 & 2 & 1 \\ 0 & 0 & 2 \end{array}\right], \quad \mathbf{D} = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix}.$$

Řešení: Všechny zadané matice mají stejný charakteristický polynom

$$p_{\mathbf{A}}(\lambda) = p_{\mathbf{B}}(\lambda) = p_{\mathbf{C}}(\lambda) = p_{\mathbf{D}}(\lambda) = (2 - \lambda)^3,$$

Vidíme, že $\lambda = 2$ je trojnásobné vl. číslo všech čtyř matic.

Vlastní vektory

A: $v^{(1)} = (1,0,0)^T$
$v^{(2)} = (0,1,0)^T$
$v^{(3)} = (0,0,1)^T$

Pozn.: matice $\mathbf{A} - \lambda\mathbf{I}$ je nulová, tj. systém všech řešení rovnice $\mathbf{A} - \lambda\mathbf{I} = \mathbf{0}$ je lin. kombinací $v^{(1)}$, $v^{(2)}$, $v^{(3)}$.

B: $v^{(1)} = (1,0,0)^T$
$v^{(3)} = (0,0,1)^T$

Pozn.: $\mathbf{B} - \lambda\mathbf{I} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

C: $v^{(1)} = (1,0,0)^T$
$v^{(2)} = (0,1,0)^T$

D: $v^{(1)} = (1,0,0)^T$

Pozn.: $\mathbf{D} - \lambda\mathbf{I} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

□

Poznámka:
Počet lineárně nezávislých vlastních vektorů může být menší než je řád matice.

Věnujme se nejprve metodám na řešení **úplného problému**. Metody rozdělíme takto:

1) metody založené na výpočtech vlastních čísel pomocí charakteristického polynomu
   Nevýhodné pro velká n (řád matice $\mathbf{A}$), protože je obtížné vypočítat $p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I})$ z definice determinantu.

2) metody využívající podobnosti matic
   Tato kategorie metod využívá faktu, že podobné matice mají stejná vlastní čísla.
   Princip: konstruujeme posloupnost navzájem podobných matic, která konverguje k matici, jejíž vlastní čísla se dají jednoduchým způsobem určit.

3) smíšené metody založené na převodu obecné matice na matici třídiagonální (např. Givensova, Householderova a Lanczosova metoda) a následný efektivní výpočet kořenů charakteristického polynomu této upravené matice.

Metoda LU-rozkladu

(LR-transformace, LR-algoritmus)

$\mathbf{A} = \mathbf{LU}$ ... rozklad matice $\mathbf{A}$ na dolní trojúhelníkovou matici $\mathbf{L}$ a horní trojúhelníkovou matici $\mathbf{U}$, kde na diagonále matice $\mathbf{L}$ jsou pro jednoznačnost rokladu jednotky. Sestrojíme matici $\mathbf{B}$, která bude podobná matici $\mathbf{A}$.

$$\mathbf{B} = \mathbf{UL} \qquad (\mathbf{U} = \mathbf{L}^{-1}\mathbf{A} \Rightarrow \mathbf{B} = \mathbf{UL} = \mathbf{L}^{-1}\mathbf{AL}).$$

Postup:

Sestrojíme posloupnost matic $\mathbf{A}_k$:

(i) $\mathbf{A}_0 = \mathbf{A}, \ \ k = 0$

(ii) provedeme LU rozklad matice $\mathbf{A}_k = \mathbf{L}_k\mathbf{U}_k$

(iii) sestrojíme matici $\mathbf{A}_{k+1} = \mathbf{U}_k\mathbf{L}_k$

(iv) je-li matice $\mathbf{A}_{k+1}$ horní trojúhelníková $\Rightarrow$ konec, jinak $k = k + 1$ a jdi na (ii)

Poznámka: Dá se ukázat, že když matice $\mathbf{B}_k = \mathbf{L}_0\mathbf{L}_1 \dots \mathbf{L}_k$ konvergují k regulární matici, potom matice $\mathbf{A}_k$ také konvergují, a to k horní trojúhelníkové matici s vlastními čísly na diagonále. Platí

$$\mathbf{A}_{k+1} = \underbrace{\mathbf{L}_k^{-1}\mathbf{A}_k}_{\mathbf{U}_k}\mathbf{L}_k$$

a tedy

$$\mathbf{A}_{k+1} = \underbrace{\mathbf{L}_k^{-1}\mathbf{L}_{k-1}^{-1}\dots\mathbf{L}_0^{-1}}_{\mathbf{B}_{k+1}^{-1}}\mathbf{A}_0\underbrace{\mathbf{L}_0\mathbf{L}_1\dots\mathbf{L}_k}_{\mathbf{B}_{k+1}}$$

Poznámka: Je-li matice $\mathbf{A}$ symetrická a pozitivně definitní, provádíme LU-rozklad ve smyslu Choleského rozkladu, tj. $\mathbf{A} = \mathbf{LL}^T$. Potom lze ukázat, že $\mathbf{A}_k$ konverguje k diagonální matici.

Nevýhody:

- pomalá konvergence posloupnosti $\mathbf{A_k}$
- velký počet operací pro matice větších řádů
- nelze realizovat pro obecné matice $\mathbf{A}$

Poznámka: Jestliže pro dostatečně velké k je $\mathbf{A}_k$ horní trojúhelníková matice, potom vlastní vektory jsou (přibližně) sloupce matice $\mathbf{B}_k = \mathbf{L}_0\mathbf{L}_1 \dots \mathbf{L}_{k-1}$.

Metody ortogonálních transformací

Použijeme podobný princip jako v předchozím případě, tj. sestrojíme posloupnost podobných matic $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2 \ldots$ tak, že

$$\mathbf{A}_{k+1} = \mathbf{Q}_k^T \mathbf{A}_k \mathbf{Q}_k, \quad k = 0, 1, 2 \ldots.$$

Požadujeme, aby posloupnost $\mathbf{A}_k$ konvergovala k matici, jejíž vlastní čísla lehce určíme. Ortogonální matici $\mathbf{Q}_k$ vybíráme speciálním postupem. Výhodou tohoto algoritmu je *numerická stabilita*.

Poznámka: Pro obecnou matici používáme metodu *QU-rozkladu* (*QR-transformace*).

$\mathbf{A} = \mathbf{QU}$ $\quad\mathbf{Q} \ldots$ ortogonální matice ($\mathbf{QQ}^{\mathrm{T}} = \mathbf{I}$, tj. $\mathbf{Q}^{\mathrm{T}} = \mathbf{Q}^{-1}$)
$\quad\mathbf{U} \ldots$ horní trojúhelníková matice

$\mathbf{B} = \mathbf{UQ}$ $\quad(\mathbf{U} = \mathbf{Q}^{-1}\mathbf{A} \Rightarrow \mathbf{B} = \mathbf{Q}^{-1}\mathbf{AQ} \Rightarrow \mathbf{B} = \mathbf{Q}^{\mathrm{T}}\mathbf{AQ})$.

Motivační příklad: Příkladem ortogonální matice je matice rovinné rotace o úhel α:

$$\mathbf{Q}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \overset{\text{ozn}}{=} \begin{bmatrix} c & -s \\ s & c \end{bmatrix}.$$

Pro matici

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

stanovte matici $\mathbf{B} = \mathbf{Q}^{\mathrm{T}}(\alpha)\mathbf{AQ}(\alpha)$ tak, aby $b_{12} = 0$.

Řešení: Rozepíšeme si prvky matice $\mathbf{B}$:

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \cdot \begin{bmatrix} c & -s \\ s & c \end{bmatrix} =$$

$$= \begin{bmatrix} 2c + s & c + 3s \\ -2s + c & -s + 3c \end{bmatrix} \cdot \begin{bmatrix} c & -s \\ s & c \end{bmatrix} =$$

$$= \begin{bmatrix} 2c^2 + cs + cs + 3s^2 & -2cs - s^2 + c^2 + 3cs \\ -2cs + c^2 - s^2 + 3cs & 2s^2 - cs - cs + 3c^2 \end{bmatrix}.$$

Pro splnění podmínky $b_{12} = 0$ musí platit

$$-2cs - s^2 + c^2 + 3cs = cs - s^2 + c^2 = 0,$$

tj.

$$\underbrace{\cos\alpha\sin\alpha}_{\frac{1}{2}\sin 2\alpha}\underbrace{-\sin^2\alpha+\cos^2\alpha}_{+\cos 2\alpha}=0.$$

$$\cos 2\alpha=-\frac{1}{2}\sin 2\alpha$$

$$-2=\tan 2\alpha$$

$$\underline{\alpha\doteq -0,5535}$$

Po dosazení dostaneme, že

$$\mathbf{B}=\begin{bmatrix} 3,6180 & 0 \\ 0 & 1,3819 \end{bmatrix}.$$

B je diagonální matice s vlastními čísly na diagonále a stejná vlastní čísla má i matice **A**. □

Poznámka: Podobně jako v předchozí metodě, pro dostatečně velké k je $\mathbf{A}_k$ horní trojúhelníková matice a vlastní vektory jsou (přibližně) sloupce matice $\mathbf{Q}_0\mathbf{Q}_1\ldots\mathbf{Q}_{k-1}$.

Poznámka: Pro symetrickou matici **A** vede uvedený postup na tzv. *metodu Jacobiovy diagonalizace*.

Speciální případ QR-transformace
Jacobiova diagonalizace

Věta: Je-li $\mathbf{A}$ reálná symetrická matice, potom existuje ortogonální matice $\mathbf{Q}$ tak, že $\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \mathbf{\Lambda}$ (diagonální matice s vlastními čísly na diagonále – *spektrální matice*).

Princip: Matici $\mathbf{Q}$ získáme součinem matic $\mathbf{Q}_{p,q}(\alpha)$, kde

$$\mathbf{Q}_{p,q}(\alpha) = \begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & \cos\alpha & \dots & \dots & \dots & -\sin\alpha & & & \\ & & & \vdots & 1 & & & \vdots & & & \\ & & & \vdots & & \ddots & & \vdots & & & \\ & & & \vdots & & & 1 & \vdots & & & \\ & & & \sin\alpha & \dots & \dots & \dots & \cos\alpha & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \leftarrow\ p\text{-tý řádek} \\ \\ \\ \leftarrow\ q\text{-tý řádek} \\ \\ \\ \\ \end{matrix}$$

$$\begin{matrix} \uparrow & & \uparrow \\ p\text{-tý sloupec} & & q\text{-tý sloupec} \end{matrix}$$

a α volíme tak, abychom vynulovali prvek v pozici p, q a tedy i v pozici q, p.

$\mathbf{Q} = \prod_{p,q} \mathbf{Q}_{p,q}(\alpha)$ — postupně vynulujeme všechny nediagonální prvky.

Poznámka: Při výpočtech nemusíme určovat úhel α, ale lze odvodit přímé vzorce.

Poznámka: Zbývá zvolit strategii na volbu indexů p a q. Nejjednodušší je postupně nulovat všechny mimodiagonální prvky (podobně jako v Gaussově eliminační metodě pro řešení soustavy lineárních rovnic). Uvědomme si ale, že se získané nuly z předchozího kroku obecně nezachovají. Další možností je nulovat vždy mimodiagonální prvek, který je největší v absolutní hodnotě (zde je třeba v každé iteraci vyhledat tento prvek, což zpomalí výpočet). Iterační proces zastavíme, je-li norma trojúhelníkové matice pod diagonálou menší než zadaná tolerance.

Pro řešení Částečného problému si uvedeme dvě metody.

Mocninná metoda

Chceme určit vl. číslo matice $\mathbf{A}$ s největší absolutní hodnotou (*dominantní vlastní číslo*).

Předpoklady:

1. $\mathbf{A}$ má n-lineárně nezávislých vlastních vektorů
2. existuje jediné dominantní vlastní číslo
3. vlastní čísla lze seřadit: $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$.

Odvození:

1. Zvolíme $\mathbf{y}^{(0)}$ jako lin. kombinaci vl. vektorů

$$\mathbf{y}^{(0)} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n.$$

2. Sestrojíme posloupnost

$$\mathbf{y}^{(k)} = \mathbf{A}\mathbf{y}^{(k-1)}, \quad \text{tj. } \mathbf{y}^{(k)} = \mathbf{A}^k\mathbf{y}^{(0)}.$$

$$\mathbf{y}^{(k)} = \alpha_1\mathbf{A}^k\mathbf{v}_1 + \alpha_2\mathbf{A}^k\mathbf{v}_2 + \ldots + \alpha_n\mathbf{A}^k\mathbf{v}_n.$$

3. Platí: $\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i$, potom

$$\mathbf{y}^{(k)} = \alpha_1 \underbrace{\lambda_1^k}_{*}\mathbf{v}_1 + \alpha_2\lambda_2^k\mathbf{v}_2 + \ldots + \alpha_n\lambda_n^k\mathbf{v}_n. \qquad {}^*\text{dominantní vl. číslo (vytkneme)}$$

4. dostaneme:

$$\mathbf{y}^{(k)} = \lambda_1^k\Big[\alpha_1\mathbf{v}_1 + \underbrace{\sum_{i=2}^{n}\alpha_i\overbrace{\left(\frac{\lambda_i}{\lambda_1}\right)^k}^{\to 0}\mathbf{v}_i}_{\varepsilon_k \to \mathbf{0}}\Big].$$

5. analogicky pro $\mathbf{y}^{(k+1)}$

6. vybereme j-tou složku $\mathbf{y}^{(k)}$ a $\mathbf{y}^{(k+1)}$, vydělíme je a provedeme limitní přechod

$$\lim_{k\to\infty}\frac{y_j^{(k+1)}}{y_j^{(k)}} = \lim_{k\to\infty}\frac{\lambda_1^{k+1}(\alpha_1 v_{1,j} + \overbrace{\varepsilon_{k+1,j}}^{\to 0})}{\lambda_1^k(\alpha_1 v_{1,j} + \underbrace{\varepsilon_{k,j}}_{\to 0})} = \lambda_1$$

Příklad: Mocninnou metodou stanovte dominantní vlastní číslo matice $\mathbf{A}$, kde

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \text{a} \qquad \mathbf{y}^{(0)} = [1,1,1]^T.$$

Řešení: Použijeme iterační formuli $\mathbf{y}^{(k+1)} = \mathbf{A}\mathbf{y}^{(k)}$, pro $k = 0, 1, \ldots$

$$\mathbf{y}^{(1)} = [2;3;2]^T \qquad \lambda_1^{(1)} = \frac{y_2^{(1)}}{y_2^{(0)}} = 3,$$

$$\mathbf{y}^{(2)} = [5;7;5]^T \qquad \lambda_1^{(2)} = \tfrac{7}{3} \approx 2,3333,$$

$$\mathbf{y}^{(3)} = [12;17;12]^T \quad \lambda_1^{(3)} = \tfrac{17}{7} \approx 2,4285,$$

$$\mathbf{y}^{(4)} = [29;41;29]^T \quad \lambda_1^{(4)} = \tfrac{41}{17} \approx 2,4117,$$

$$\mathbf{y}^{(5)} = [70;99;70]^T \quad \lambda_1^{(5)} = \tfrac{99}{41} \approx \underline{\underline{2,4146}}.$$

□

Poznámka: Zastavovací podmínka použijeme ve tvaru $|\lambda_1^{(k+1)} - \lambda_1^{(k)}| < \delta$.

Poznámka: Nejlepší aproximaci dostaneme, dělíme-li složky, které mají největší absolutní hodnotu.

Poznámka: Abychom zamezili přetečení, resp. podtečení při zobrazení čísel v počítači je vhodné v každém kroku normovat vektor $\mathbf{y}^{(k)}$ (norma $\mathbf{y}^{(k)}$ roste, resp. klesá pro vlastní číslo v absolutní hodnotě větší, resp. menší než 1).

Poznámka: Nevýhody mocninné metody:

- odhad chyby
- konvergence (obvykle v praxi nevíme, zda jsou splněny předpoklady mocninné metody)
- volba $\mathbf{y}^{(0)}$ (bude-li vektor $\mathbf{y}^{(0)}$ takovou lineární kombinací vlastních vektorů, že koeficient u vlastního vektoru odpovídajícího dominantnímu vlastnímu číslu bude roven 0, potom mocninná metoda nevypočte dominantní vlastní číslo)

Metoda Rayleighova podílu

Pro použití metody Rayleighova podílu budeme navíc předpokládat, že matice $\mathbf{A}$ je symetrická (reálná). Potom musí být vlastní vektory ortonormální, tj.

$$\left(\mathbf{v}_i^T\mathbf{v}_j = 0 \text{ pro } i \neq j \quad \text{a } \mathbf{v}_i^T\mathbf{v}_i = 1\right).$$

Odvození: 6. krok z odvození mocninné metody nahradíme vyjádřením součinu $\mathbf{y}^{(k)^T}\mathbf{y}^{(k)}$

$$\mathbf{y}^{(k)^T}\mathbf{y}^{(k)} = \lambda_1^k\Big[\alpha_1\mathbf{v}_1^T + \overbrace{\sum_{i=2}^{n}\alpha_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\mathbf{v}_i^T}^{\varepsilon_k^T}\Big] \,.\, \lambda_1^k\Big[\alpha_1\mathbf{v}_1 + \overbrace{\sum_{i=2}^{n}\alpha_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\mathbf{v}_i}^{\varepsilon_k}\Big] =$$

$$= \lambda_1^{2k}\Big[\alpha_1^2 + \underbrace{\sum_{i=2}^{n}\alpha_i^2\left(\frac{\lambda_i}{\lambda_1}\right)^{2k}}_{\varepsilon_k^T\varepsilon_k}\Big]$$

a součinu $\mathbf{y}^{(k)^T}\mathbf{y}^{(k+1)^T}$

$$\mathbf{y}^{(k)^T}\mathbf{y}^{(k+1)} = \lambda_1^k\Big[\alpha_1\mathbf{v}_1^T + \overbrace{\sum_{i=2}^{n}\alpha_i\left(\frac{\lambda_i}{\lambda_1}\right)^k\mathbf{v}_i^T}^{\varepsilon_k^T}\Big] \,.\, \lambda_1^{k+1}\Big[\alpha_1\mathbf{v}_1 + \overbrace{\sum_{i=2}^{n}\alpha_i\left(\frac{\lambda_i}{\lambda_1}\right)^{k+1}\mathbf{v}_i}^{\varepsilon_{k+1}}\Big] =$$

$$= \lambda_1^{2k+1}\Big[\alpha_1^2 + \underbrace{\sum_{i=2}^{n}\alpha_i^2\left(\frac{\lambda_i}{\lambda_1}\right)^{2k+1}}_{\varepsilon_k^T\varepsilon_{k+1}}\Big]$$

Dostáváme:

$$\lim_{k\to\infty}\frac{\mathbf{y}^{(k)\mathrm{T}}\mathbf{A}\mathbf{y}^{(k)}}{\mathbf{y}^{(k)\mathrm{T}}\mathbf{y}^{(k)}} = \lim_{k\to\infty}\frac{\mathbf{y}^{(k)\mathrm{T}}\mathbf{y}^{(k+1)}}{\mathbf{y}^{(k)^T}\mathbf{y}^{(k)}} = \frac{\lambda_1^{2k+1}(\alpha_1^2 + \overbrace{\varepsilon_k^T\varepsilon_{k+1}}^{\to 0})}{\lambda_1^{2k}(\alpha_1^2 + \underbrace{\varepsilon_k^T\varepsilon_k}_{\to 0})} = \lambda_1.$$

Poznámka: Součin $\varepsilon_k^T\varepsilon_k$ konverguje k nule (pro $k \to \infty$) zhruba dvakrát rychleji než ε_k k nulovému vektoru $\Rightarrow$ metoda Rayleighova podílu bude rychlejší než mocninná metoda.

Příklad: Metodou Rayleighova podílu určete dominantní vlastní číslo matice $\mathbf{A}$, kde

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \text{a} \qquad \mathbf{y}^{(0)} = [1; 1; 1]^T.$$

Řešení:

$$\mathbf{y}^{(1)} = [2; 3; 2]^T \qquad \lambda_1^{(1)} = \frac{\mathbf{y}^{(0)^T}\mathbf{y}^{(1)}}{\mathbf{y}^{(0)^T}\mathbf{y}^{(0)}} = \tfrac{7}{3} \approx 2,3333,$$

$$\mathbf{y}^{(2)} = [5; 7; 5]^T \qquad \lambda_1^{(2)} = \tfrac{41}{17} \approx 2,4117,$$

$$\mathbf{y}^{(3)} = [12; 17; 12]^T \quad \lambda_1^{(3)} = \tfrac{60+119+60}{25+49+25} = \tfrac{239}{99} \approx \underline{\underline{2,41417}}.$$

APROXIMACE FUNKCÍ

Jedním ze základních úkolů numerických metod matematické analýzy je studium aproximací funkcí. Při numerickém řešení úloh matematické analýzy totiž často nahrazujeme danou funkci f, vystupující v řešené matematické úloze, jinou funkcí φ, která v nějakém vhodném smyslu napodobuje funkci f a snadno se přitom matematicky zpracovává či modeluje na počítači. Tuto funkci φ nazýváme **aproximací funkce** f.

Poznamenejme zde, že jsme již aproximaci funkce používali u řešení nelineární rovnice. Například u Newtonovy metody jsme danou funkci f z řešené rovnice $f(x) = 0$ aproximovali lineární funkcí (tečnou ke grafu funkce f); podobně tak tomu bylo u metody sečen.

Poznámka: Již pouhý výpočet funkčních hodnot některých základních funkcí ($\sin x$, e^x, $\ln x$, ...) v počítači či na kalkulačce se provádí užitím aproximace těchto funkcí. Tyto aproximace jsou ovšem zabudovány do výpočetního systému a uživatel si často ani neuvědomuje, že píše-li v programu např. `y=sin(x)`, nahrazuje výpočet hodnoty funkce $\sin x$ výpočtem hodnoty jistého polynomu.

Typickým příkladem užití aproximace funkcí jsou numerické metody pro výpočet určitého integrálu. Další oblastí numerické matematiky založenou na užití aproximací je zpracování výsledků měření. Hledáme zde zpravidla jednoduchý analytický výraz vyjadřující přibližnou funkční závislost pro tabulkou zadané hodnoty.

Při výběru vhodné aproximace postupujeme tak, že předem zvolíme tvar aproximující funkce, ve které vystupují nějaké proměnné parametry, a hodnoty těchto parametrů se pak snažíme určit tak, aby získaná aproximace vyhovovala našim požadavkům.

Základní otázkou zůstává, v jakém tvaru budeme hledat aproximaci φ funkce f. Jednou z možností je aproximaci volit např. ve tvaru polynomu $\varphi(x) = a_3x^3 + a_2x^2 + a_1x + a_0$.

Obecně tedy nejprve určíme systém jednoduchých **základních (bázových) funkcí** (ne nutně polynomů) φ_0, φ_1, φ_2, ..., φ_n a funkci f aproximujeme lineární kombinací základních funkcí, tj.

$$\varphi(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + \ldots + c_n\varphi_n(x). \tag{1}$$

Otázka výběru aproximace se tedy převede na určení hodnot parametrů c_0, c_1, ..., c_n podle nějakého kritéria vhodného pro konkrétní úlohu.

Poznámka: Velmi často budeme za základní funkce volit funkce 1, x, x^2, ..., x^n, tj. aproximaci φ budeme hledat ve třídě polynomů nejvýše n-tého stupně.

Nyní se zamyslíme nad kritérii, podle kterých budeme určovat parametry z lineární kombinace (1). Ty samozřejmě vyplývají z charakteru konkrétní úlohy a zhruba je můžeme rozdělit do tří skupin

Aproximace na okolí bodu - Použijeme, chceme-li aproximovat chování funkce v malém okolí bodu. Příkladem může být např. vyčíslení hodnoty $\sin \frac{\pi}{4}$ na kalkulačce.

Interpolace - Použijeme, chceme-li tabulkou danými body proložit polynom, tj. požadujeme-li, aby aproximace přesně procházela zadanými body.

L_2-aproximace - Použijeme, hledáme-li funkční závislost mezi tabulkou danými body (získaných například měřením), kde nutně nevyžadujeme, aby aproximace danými body procházela. Důvodem můžou být např. chyby, se kterými jsme hodnoty naměřili.

Aproximace na okolí bodu

Nyní se budeme věnovat jednotlivým úlohám. Začneme aproximací na okolí bodu, tj. mluvíme o **aproximaci Taylorovým polynomem**.

Předpokládáme, že daná funkce f má v daném bodě x_0 aspoň n derivací, a že hodnoty těchto derivací v bodě x_0 známe. Podmínky pro funkci, která co nejlépe napodobuje chování funkce f matematicky zapíšeme takto:

$$\varphi^{(j)}(x_0) = f^{(j)}(x_0), \quad j = 0, 1, \ldots, n,$$

tj. hodnoty derivací funkcí f a φ v bodě x_0 jsou stejné až do řádu n. Tuto podmínku samozřejmě splňuje Taylorův polynom

$$T_n(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n.$$

Pro chybu aproximace Taylorovým polynomem platí

$$e(x) = f(x) - T_n(x) = f^{(n+1)}(\xi)\frac{(x - x_0)^{n+1}}{(n+1)!}, \quad \xi \in \mathcal{U}(x_0)$$

a umíme-li odhadnout $n + 1$ derivaci funkce f na daném okolí bodu x_0, můžeme provést následující odhad chyby aproximace.

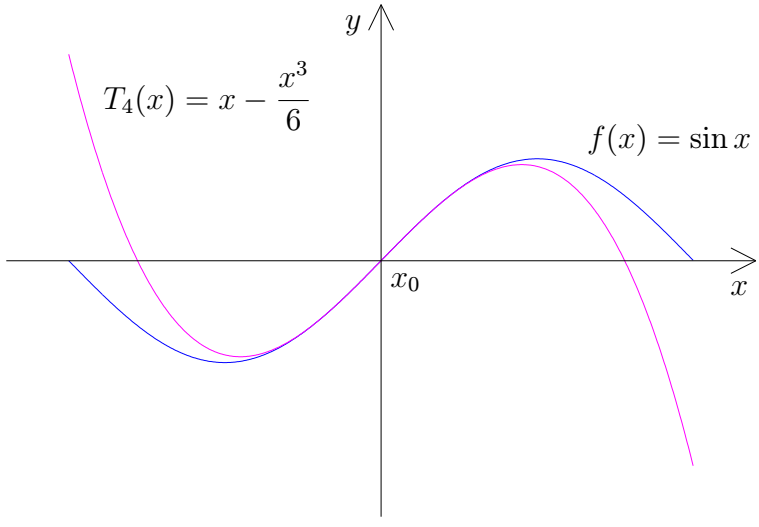
$$\text{Platí-li } \; |f^{(n+1)}(x)| \leq M \quad \forall x \in \mathcal{U}(x_0), \quad \text{potom } \; |e(x)| \leq \frac{M}{(n+1)!}|x - x_0|^{n+1}.$$

Příklad: Stanovte Taylorův polynom 4.stupně, který aproximuje funkci $f(x) = \sin x$ v bodě $x_0 = 0$.

Řešení:

$$T_4(x) = \sin 0 + x.\cos 0 - \frac{x^2}{2}\sin 0 - \frac{x^3}{6}\cos 0 + \frac{x^4}{24}\sin 0 \;=\; x - \frac{x^3}{6}.$$

Vidíme, že v Taylorově polynomu 4.stupně vypadly členy s x^2 a x^4, protože obsahovaly $\sin 0$. Takže v našem případě platí, že $T_4(x) = T_3(x)$.



Odpovězme na otázku, pro jaké x lze psát $\sin x \approx x - \frac{x^3}{6}$, aby chyba nebyla větší než 10^{-6}?

Výraz pro chybu má tvar

$$e(x) = \frac{x^5}{5!}(-\cos\xi)$$

Dále víme, že platí

$$|\cos\xi| \le 1$$

a proto

$$\underbrace{|e(x)|}_{\le 10^{-6}} \le \frac{|x^5|}{5!} = \frac{|x^5|}{120}$$

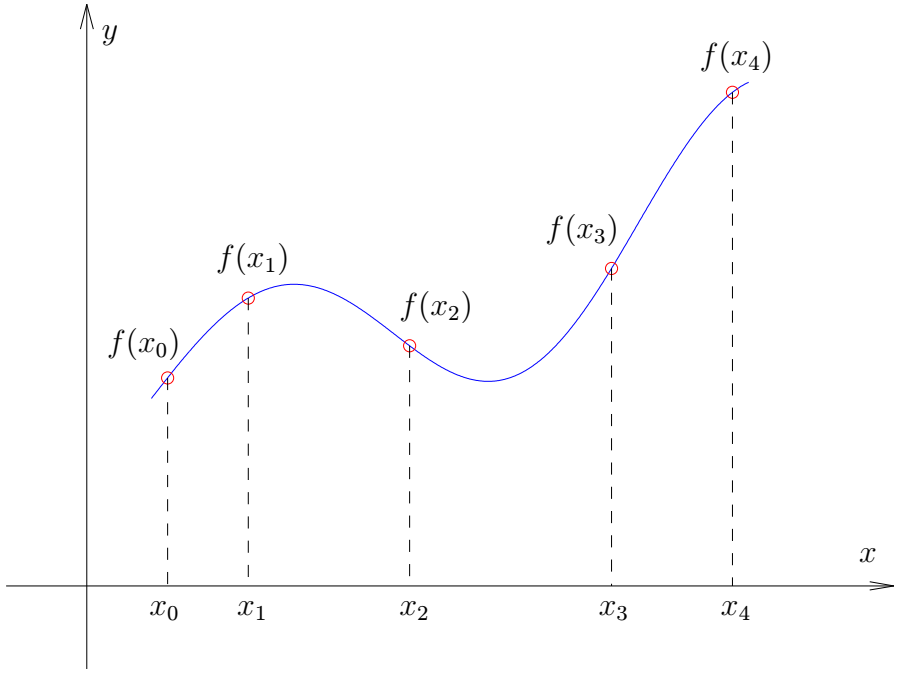
tj. chceme, aby

$$\frac{|x^5|}{120} \le 10^{-6} \quad \Rightarrow \quad |x^5| \le 120.10^{-6} \quad \Rightarrow \quad |x| \le \underbrace{0.164375}_{\approx 9°25'} \text{ (rad)}$$

□

Aproximace interpolačním polynomem

Chceme-li aproximovat funkci, která je dána svými hodnotami v $n+1$ bodech x_i, $i = 0, 1, \ldots, n$ (body x_i nazýváme uzly interpolace), a požadujeme-li, aby aproximace procházela zadanými body, použijeme aproximaci interpolačním polynomem. Aproximace nám potom poslouží k získání přibližné hodnoty zadané funkce v libovolném bodě intervalu $\langle x_0, x_n \rangle$.



Máme-li zadány hodnoty funkce f v $n+1$ různých bodech, tzn. máme zadáno $n+1$ tzv. **interpolačních podmínek** pro polynom φ, je zřejmé, že stupeň hledaného polynomu bude n (polynom n-tého stupně má $n+1$ koeficientů). Lze ukázat, že mezi všemi polynomy nejvýše n-tého stupně existuje právě jeden, který je interpolačním polynomem pro zadanou funkci. Pro určení interpolačního polynomu existuje několik postupů, ale je třeba si uvědomit, že pro zadanou funkci všechny postupy určí stejný polynom. Ukážeme si dvě možnosti určení interpolačního polynomu, tzv. **Lagrangeův** a **Newtonův interpolační polynom**.

Lagrangeův interpolační polynom

Označme si hledaný polynom n-tého stupně $L_n(x)$. Víme, že musí být splněny interpolační podmínky

$$L_n(x_i) = f(x_i), \quad i = 0, 1, \ldots, n.$$

Lagrangeův interpolační polynom hledáme ve tvaru

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \cdot l_i(x),$$

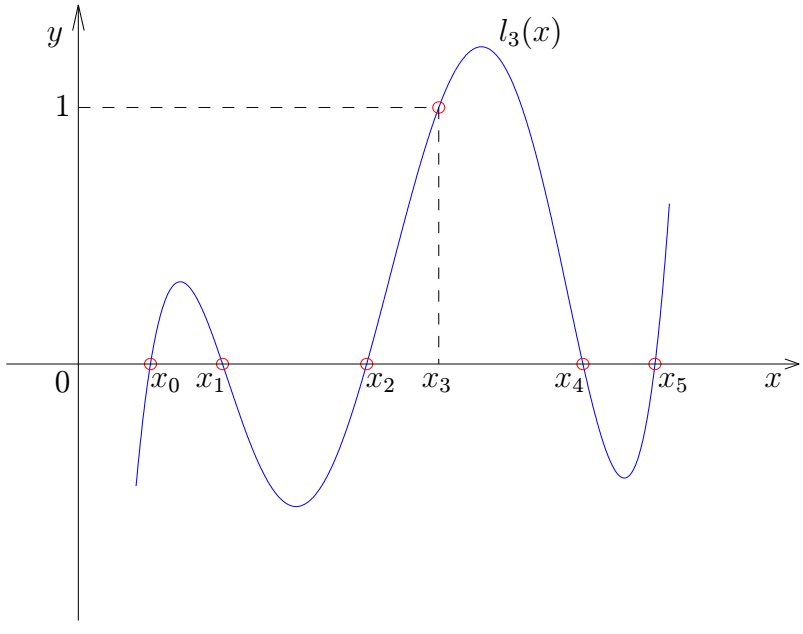
kde $l_i(x)$ jsou polynomy n-tého stupně takové, že platí

$$l_i(x_j) = \delta_{ij} = \begin{array}{ll} 1, & i = j \\ 0, & i \neq j \end{array}$$

(snadno se přesvědčíte, že dosadíte-li do předpisu pro $L_n(x)$ uzly interpolace, získáte zadané interpolační podmínky). Konkretizujme nyní dílčí polynomy $l_i(x)$. Víme, že $l_i(x)$ má kořeny $x_0, x_1, x_{i-1}, x_{i+1}, \ldots x_n$ a nabývá hodnoty 1 v bodě x_i. Můžeme jej tedy zapsat ve tvaru

$$l_i(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)}$$

Na obrázku je ukázán příklad dílčího polynomu $l_3(x)$:



Příklad: Stanovte Lagrangeův interpolační polynom pro funkci f, která je dána tabulkou a určete přibližnou hodnotu $f(2)$.

i	0	1	2
x_i	0	1	3
$f(x_i)$	1	2	0

Řešení:

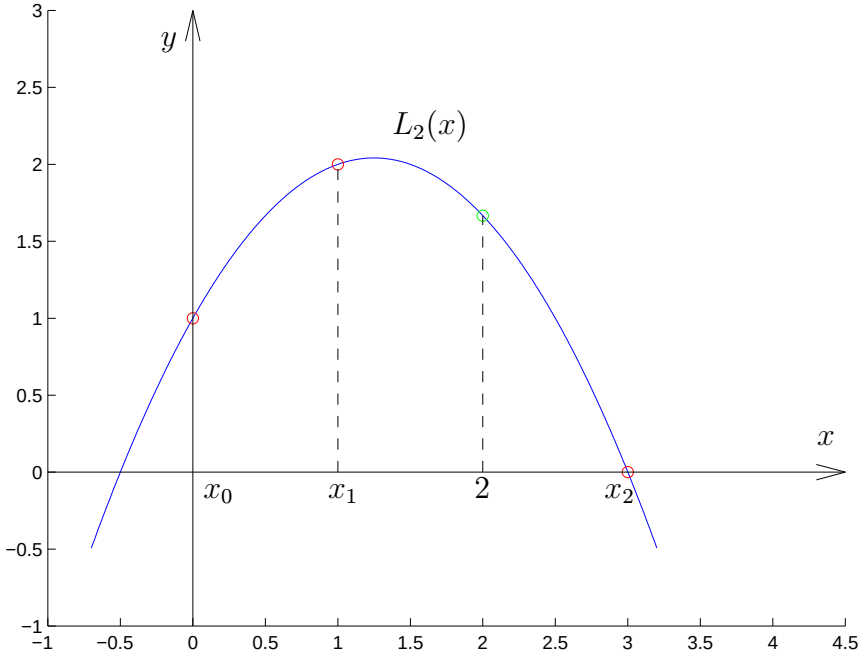
$$L_2(x) = f(x_0) \cdot l_0(x) + f(x_1) \cdot l_1(x) + f(x_2) \cdot l_2(x),$$

kde

$$l_0(x) = \frac{(x-1)(x-3)}{(0-1)(0-3)} = \frac{1}{3}(x-1)(x-3)$$
$$l_1(x) = \frac{(x-0)(x-3)}{(1-0)(1-3)} = -\frac{1}{2}x(x-3)$$
$$l_2(x) = \frac{(x-0)(x-1)}{(3-0)(3-1)} = \frac{1}{6}x(x-1)$$

Dosadíme

$$L_2(x) = 1 \cdot \left(\frac{1}{3}(x-1)(x-3)\right) + 2 \cdot \left(-\frac{1}{2}x(x-3)\right) + 0 \cdot \left(\frac{1}{6}x(x-1)\right) =$$
$$= \frac{1}{3}(x^2 - 4x + 3) - (x^2 - 3x) = \underline{\underline{-\frac{2}{3}x^2 + \frac{5}{3}x + 1}}$$



$L_2(2) = ?$

$$L_2(2) = -\frac{2}{3} \cdot 2^2 + \frac{5}{3} \cdot 2 + 1 = \frac{-8 + 10 + 3}{3} = \frac{5}{3}$$

□

Newtonův interpolační polynom

Označme si hledaný polynom n-tého stupně $N_n(x)$. Pro jeho odvození použijeme jinou kontrukci. Polynom volíme ve tvaru

$$N_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}).$$

Opět požadujeme splnění interpolačních podmínek

$$N_n(x_i) = f(x_i), \quad i = 0, 1, \ldots, n.$$

Výhodou volby tohoto zdánlivě složitého předpisu je fakt, že přidáme-li další bod interpolace $[x_{n+1}, f(x_{n+1})]$, nemusíme celý výpočet opakovat, ale stačí dopočítat příslušný koeficient a_{n+1} (ostatní koeficienty a_i zůstávají beze změny). U Lagrangeova polynomu bychom museli celý výpočet provést znovu.

Ukažme si co dostaneme dosazováním interpolačních podmínek do předpisu polynomu:

$$N_n(x_0) = a_0 = f(x_0)$$

$$N_n(x_1) = a_0 + a_1(x_1 - x_0) = f(x_1) \quad \Rightarrow \quad a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$N_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = f(x_2) \quad \Rightarrow$$

$$\Rightarrow \quad a_2 = \frac{f(x_2) - f(x_0) - a_1 \overbrace{(x_2 - x_0)}^{x_2 - x_1 + x_1 - x_0}}{(x_2 - x_0)(x_2 - x_1)} =$$

$$= \frac{f(x_2) - f(x_0) - \frac{f(x_1)-f(x_0)}{x_1-x_0}(x_2 - x_1) - \frac{f(x_1)-f(x_0)}{x_1-x_0}(x_1 - x_0)}{(x_2 - x_0)(x_2 - x_1)} =$$

$$= \frac{f(x_2) - f(x_1) - \frac{f(x_1)-f(x_0)}{x_1-x_0}(x_2 - x_1)}{(x_2 - x_0)(x_2 - x_1)} =$$

$$a_2 = \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_0}$$

Poznámka: Počítat koeficienty a_i přímo ze soustavy není praktické. Koeficienty budeme počítat pomocí tzv. **poměrných diferencí**. Kostrukci Newtonova polynomu si ukážeme v následujícím příkladě.

Příklad: Stanovte Newtonův interpolační polynom pro funkci f, která je dána tabulkou:

i	0	1	2	3
x_i	0	1	-1	3
$f(x_i)$	1	2	2	0

Řešení: Stupeň polynomu je $n = 3$ (jsou zadány 4 funkční hodnoty). Celý postup můžeme přehledně zapsat do tabulky:

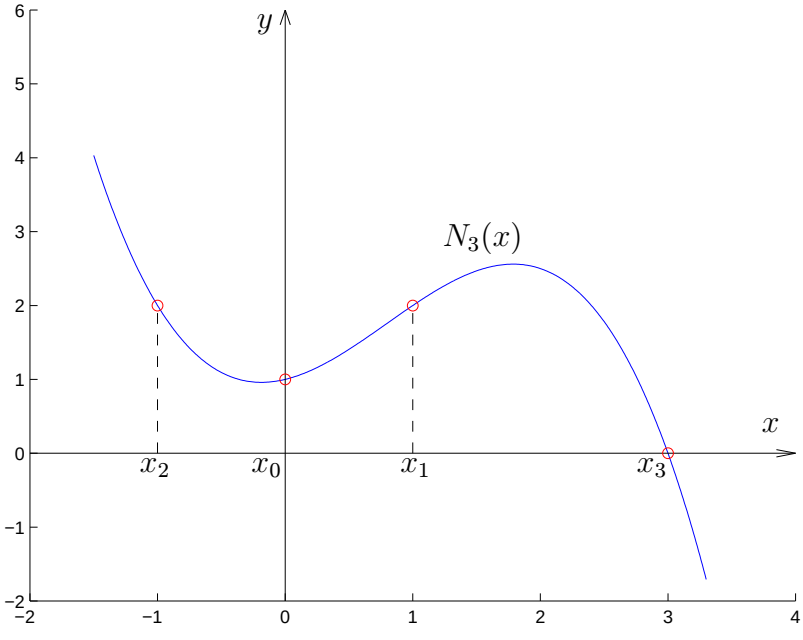
i	x_i	$f(x_i)$	$\dfrac{f(x_i)-f(x_{i-1})}{x_i-x_{i-1}} \overset{ozn.}{=} f^I(x_i)$	$\dfrac{f^I(x_i)-f^I(x_{i-1})}{x_i-x_{i-2}} \overset{ozn.}{=} f^{II}(x_i)$	$\dfrac{f^{II}(x_i)-f^{II}(x_{i-1})}{x_i-x_{i-3}} \overset{ozn.}{=} f^{III}(x_i)$
0	0	**1**			
1	1	2	$\dfrac{2-1}{1-0} = \mathbf{1}$		
2	-1	2	$\dfrac{2-2}{-1-1} = 0$	$\dfrac{0-1}{-1-0} = \mathbf{1}$	
3	3	0	$\dfrac{0-2}{3-(-1)} = -\dfrac{1}{2}$	$\dfrac{-\frac{1}{2}-0}{3-1} = -\dfrac{1}{4}$	$\dfrac{-\frac{1}{4}-1}{3-0} = -\dfrac{\mathbf{5}}{\mathbf{12}}$

V tabulce jsou na diagonále postupně uvedeny koeficienty a_0, a_1, a_2 a a_3 Newtonova interpolačního polynomu

$$N_3(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + a_3(x-x_0)(x-x_1)(x-x_2).$$

Po dosazení

$$N_3(x) = 1 + 1(x-0) + 1(x-0)(x-1) - \frac{5}{12}(x-0)(x-1)(x+1) = \ldots =$$
$$= \underline{\underline{-\frac{5}{12}x^3 + x^2 + \frac{5}{12}x + 1}}$$



□

Poznámka: Všimněme si, že v konstrukci interpolačního polynomu nezáleží na pořadí zadaných tabulkových bodů.

Poznámka: V řadě případů potřebujeme kromě a_i vypočítat hodnotu polynomu v daném bodě α, tj.

$$N_n(\alpha) = a_0 + a_1(\alpha - x_0) + a_2(\alpha - x_0)(\alpha - x_1) + \ldots + a_n(\alpha - x_0)(\alpha - x_1)\ldots(\alpha - x_{n-1}).$$

Při vhodném uzávorkování můžeme výpočet zefektivnit (zmenšíme počet operací sčítání a násobení):

$$N_n(\alpha) = a_0 + (\alpha - x_0)\Big[a_1 + (\alpha - x_1)\Big[a_2 + (\alpha - x_2)[a_3 + \ldots]\Big]\Big].$$

Tento postup můžeme samozřejmě použít jen tehdy, když už známe koeficienty a_i.

Chceme-li vypočítat pouze hodnotu polynomu $N_n(\alpha)$ v bodě α za co nejmenšího počtu operací a nepotřebujeme-li koeficienty a_i, použijeme tzv. **Nevilleův algoritmus**. Princip je podobný jako v algoritmu pro určení koeficientů Newtonova polynomu.

Nevilleův algoritmus:

1. $P_{i,0} = f(x_i); \qquad i = 0, 1, \ldots, n$
2. $P_{i,k} = P_{i,k-1} + (\alpha - x_i)\dfrac{P_{i,k-1} - P_{i-1,k-1}}{x_i - x_{i-k}};$
3. $N_n(\alpha) = P_{nn}$

Princip Nevilleova algoritmu je ukázán v následujícím příkladu.

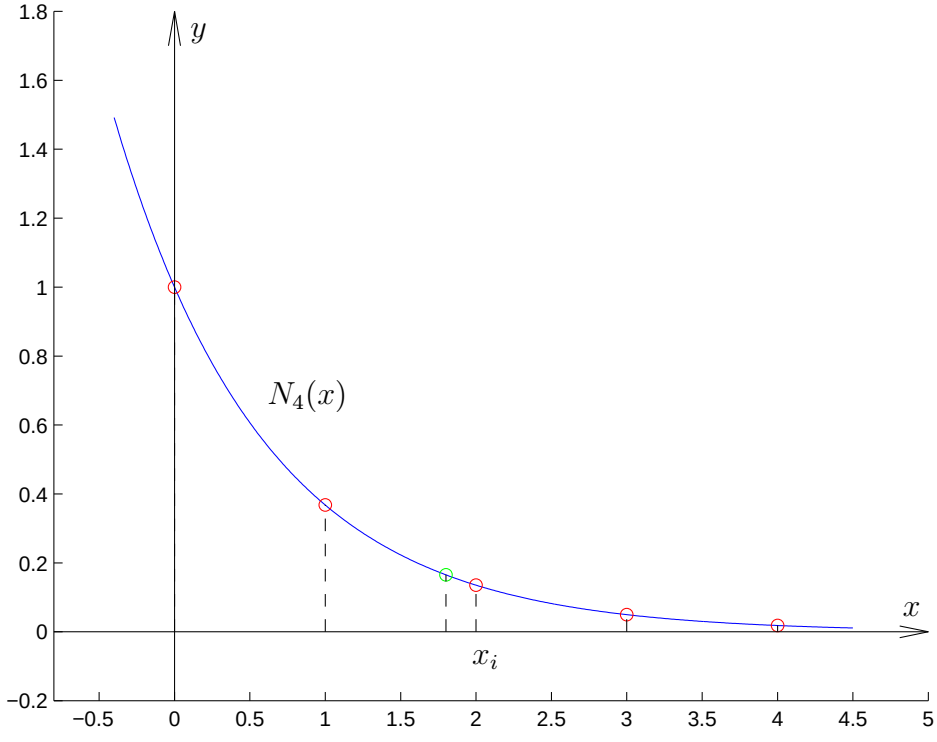
Příklad: Vypočtěte $f(1{,}8)$, kde funkce $f(x)$ je dána tabulkou:

x_i	0	1	2	3	4
$f(x_i)$	1,0000	0,36788	0,13534	0,04979	0,01832

Řešení: Uzly x_i je výhodné uspořádat podle rostoucí vzdálenosti od bodu α, v němž chceme stanovit přibližnou hodnotu funce $f(x)$. Podle rozdílu hodnot $P_{i,i}$ a $P_{i-1,i-1}$ $(i = 1, \ldots, n)$ lze rozhodnout o předčasném ukončení Nevillova algoritmu, popř. o vhodnosti interpolace pomocí $N_n(x)$.

Nevillovo schéma:

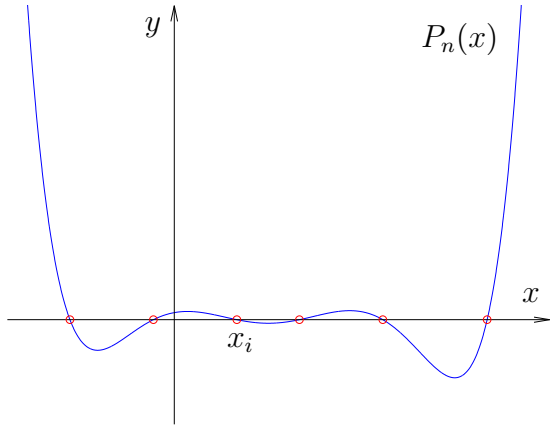
$\lvert\alpha - x_i\rvert$	x_i	$f(x_i)$				
0,2	2	**0,13534**				
0,8	1	0,36788	**0,18185**			
1,2	3	0,04979	0,24064	**0,17009**		
1,8	0	1,00000	0,42987	0,08926	**0,16201**	
2,2	4	0,01832	0,55824	0,27583	0,13901	**0,16431**



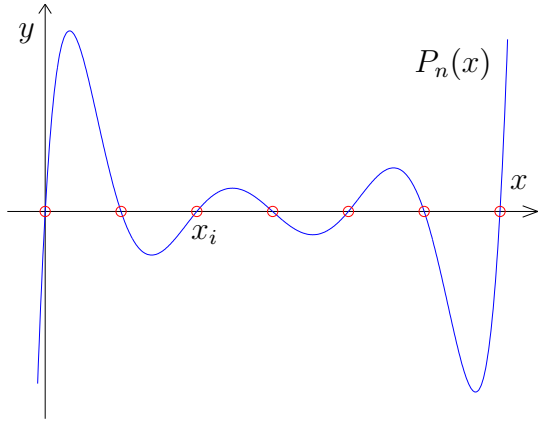
□

Poznámky:

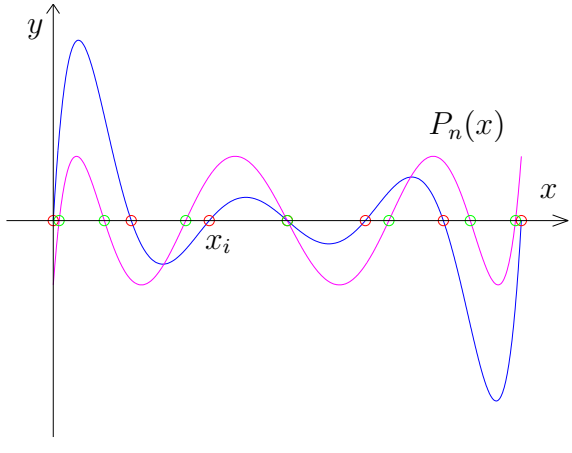
1) Interpolační polynomy vyšších řádů není vhodné užívat pro aproximaci hodnot funkce mimo interval obsahující uzly interpolace (tzv. extrapolaci), protože absolutní hodnota polynomu nabývá velkých hodnot.



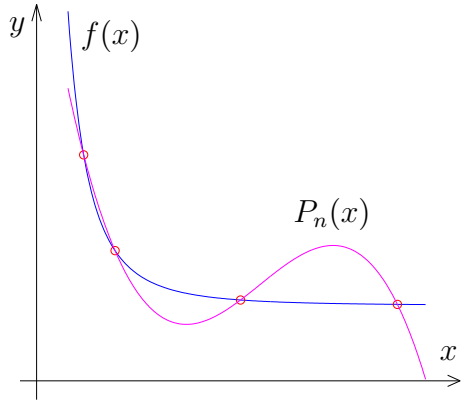
2) Dále není obecně vhodné interpolovat polynomem funkci, která je dána velkým počtem svých hodnot. Stupeň interpolačního polynomu by potom byl velký a vlastnosti polynomu vysokého stupně nejsou dobré nejen mimo interval obsahující uzly interpolace, ale i uvnitř. Interpolační polynom směrem ke krajním bodům intervalu může vlivem nepřesností ve vstupních datech výrazně oscilovat, zejména při použití ekvidistantních uzlů.



3) Použijeme-li vhodně zvolené neekvidistantní uzly, můžeme tyto oscilace minimalizovat. (Vhodnou volbou jsou uzly zvolené jako kořeny tzv. Čebyševových polynomů.)

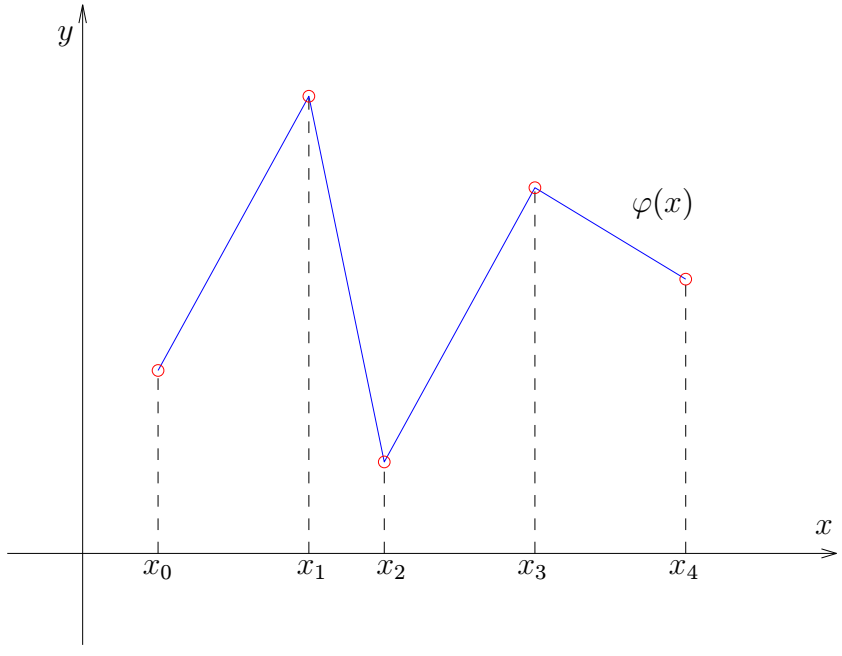


4) Interpolace polynomem není obecně vhodná např. pro funkce, které mají asymptotu. Toto je jedním z důvodů, proč zavádíme tzv. **interpolaci spline funkcemi**.



Interpolace spline funkcemi

Nejjednodušší spline funkcí je tzv. **lineární spline funkce**; jde vlastně o lomenou čáru spojující zadané interpolované body.



Nejvíce používanou je tzv. **kubická spline interpolace** vycházející z myšlenky, že danou funkci f můžeme na $\langle a, b\rangle$ aproximovat funkcemi, které jsou po částech polynomy 3.stupně. Poznamenejme zde, že ostatní volby stupně polynomů nepřináší lepší výsledky a výpočty jsou v případě větších stupňů složitější.

Interval $\langle a, b\rangle$ rozdělíme na n dílčích intervalů

$$a = x_0 < x_1 < x_2 < \ldots < x_n = b.$$

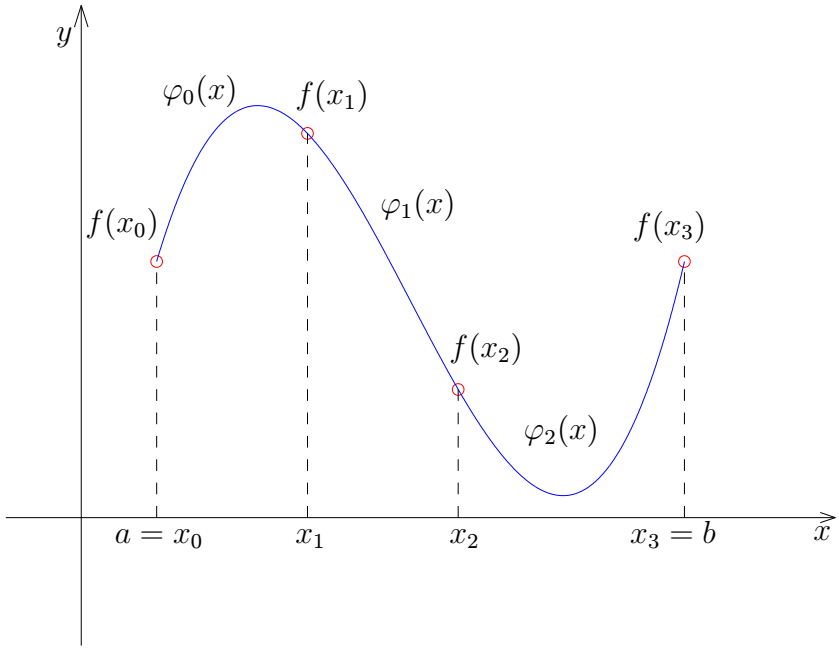
Funkci f aproximujeme funkcí φ, pro kterou platí:

a) $\varphi \in \mathbb{C}^2(\langle a, b\rangle)$ (funkce φ je spojitá a má spojité první i druhé derivace)

b) na každém intervalu $\langle x_i, x_{i+1}\rangle$ je φ polynom 3. stupně

V následujícím příkladu jsou rozepsány podmínky pro konstrukci kubické spline funkce.

Příklad :



Jednotlivé funkce $\varphi_i(x)$ (na každém intervalu $\langle x_i, x_{i+1}\rangle$ jde o jinou funkci) mají tvar:

$$\varphi_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3$$

Musí platit: ($\varphi \in \mathbb{C}^2(\langle a, b\rangle)$ a podmínky interpolace)

1) $\left.\begin{array}{l}\varphi_0(x_1) = \varphi_1(x_1)\\ \varphi_1(x_2) = \varphi_2(x_2)\end{array}\right\}$ spojitost funkce φ

2) $\left.\begin{array}{l}\varphi_0'(x_1) = \varphi_1'(x_1)\\ \varphi_1'(x_2) = \varphi_2'(x_2)\end{array}\right\}$ spojitost 1. derivace funkce φ

3) $\left.\begin{array}{l}\varphi_0''(x_1) = \varphi_1''(x_1)\\ \varphi_1''(x_2) = \varphi_2''(x_2)\end{array}\right\}$ spojitost 2. derivace funkce φ

4) $\left.\begin{array}{l}\varphi_0(x_0) = f(x_0)\\ \varphi_1(x_1) = f(x_1)\\ \varphi_2(x_2) = f(x_2)\\ \varphi_2(x_3) = f(x_3)\end{array}\right\}$ interpolační podmínky

Poznámka

Dostali jsme 10 podmínek na funkci φ, která je však určena 3×4 parametry a_i, b_i, c_i, d_i (tj. 12 podmínek) $\Rightarrow$ chybí ještě 2 podmínky.

Je vhodné doplnit některou z následujících dvojic podmínek:

5) a) $\varphi'(a) = f'(a),\ \varphi'(b) = f'(b)$ ... podmínky tečen

b) $\varphi'(a) = \varphi'(b),\ \varphi''(a) = \varphi''(b)$ ... podmínky periodicity

c) $\varphi''(a) = 0, \quad \varphi''(b) = 0$ ... tzv. přirozené podmínky

Poznámka

Uvědomme si, že

$$\varphi(x_i) = a_i$$
$$\varphi'(x_i) = b_i$$
$$\varphi''(x_i) = c_i$$
$$\varphi'''(x_i-) = d_{i-1}, \quad \varphi'''(x_i+) = d_i$$

Závěr

Podmínky 1) - 5) představují soustavu lineárních algebraických rovnic s řídkou maticí. Jejím vyřešením určíme koeficienty a_i, b_i, c_i, d_i a tím i funkce φ_i, tj. φ.

Diskrétní L_2-aproximace

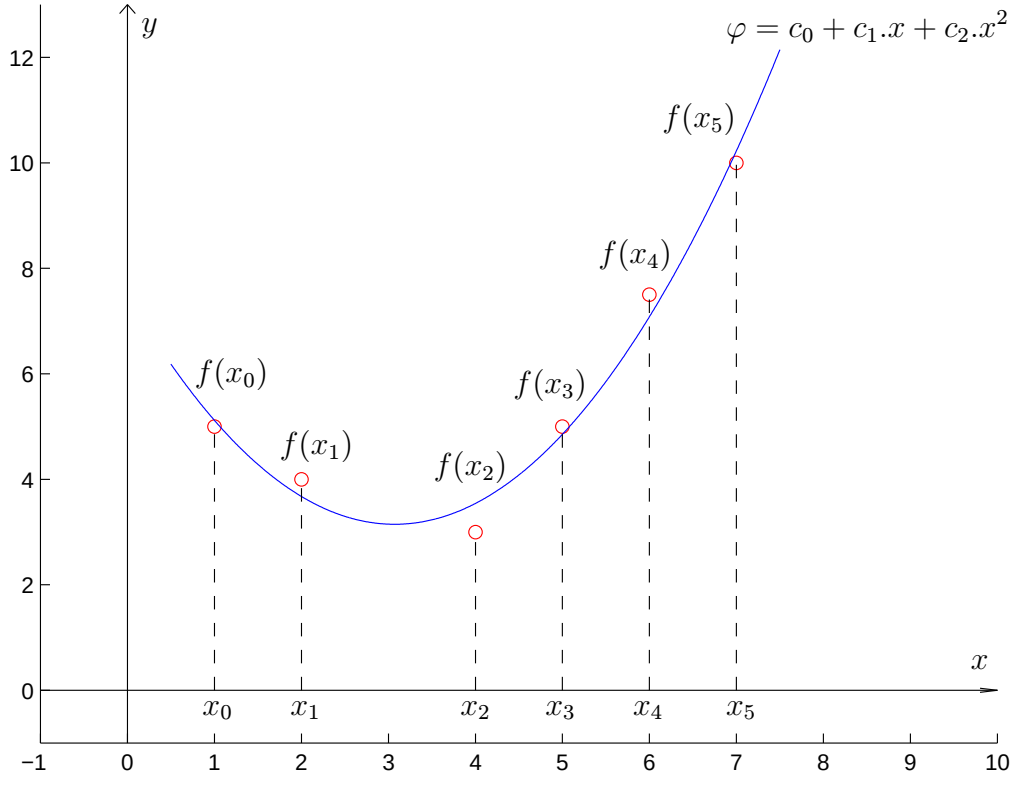
Myšlenka:

Chceme aproximovat funkci, která je dána tabulkou $\{(x_i, f(x_i))\}$. V případě, kdy jsou $f(x_i)$ zatíženy chybou (např. výsledky měření), není vhodné provádět interpolaci. Aproximaci φ hledáme ve tvaru

$$\varphi(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + \ldots + c_n\varphi_n(x),$$

kde φ_i jsou zadané funkce a c_i hledané parametry (počet bázových funkcí φ_i je menší než počet zadaných bodů, v případě rovnosti se již jedná o interpolaci). Naším cílem je minimalizovat „souhrnou odchylku“ funkce φ od zadaných dat.

Ilustrační obrázek:



Metodu diskrétní L_2-aproximace popíšeme na následujícím příkladu.

Příklad
Je dána funkce f tabulkou

x_i	$0,5$	$0,8$	$0,9$	$1,1$	$1,2$
$f(x_i)$	$2,25$	$0,72$	$0,33$	$-0,27$	$-0,48$

Tuto funkci chceme aproximovat lineární funkcí metodou nejmenších čtverců.

Lineární funkce $\varphi = c_0 . \underbrace{1}_{\varphi_0(x)} + c_1 . \underbrace{x}_{\varphi_1(x)}$

Minimalizujeme funkci

$$r(c_0, c_1) = \sum_{i=1}^{5} |f(x_i) - \varphi(x_i)|^2 = \sum_{i=1}^{5} |f(x_i) - c_0 - c_1 x_i|^2$$

Podmínky minima

$$\left.\begin{array}{ll} (1) & \dfrac{\partial r}{\partial c_1} = -2\sum_{i=1}^{5}(f(x_i) - c_0 - c_1 x_i)x_i = 0 \\ (2) & \dfrac{\partial r}{\partial c_0} = -2\sum_{i=1}^{5}(f(x_i) - c_0 - c_1 x_i) = 0 \end{array}\right\} \text{ 2 rovnice pro neznámé } c_0, c_1.$$

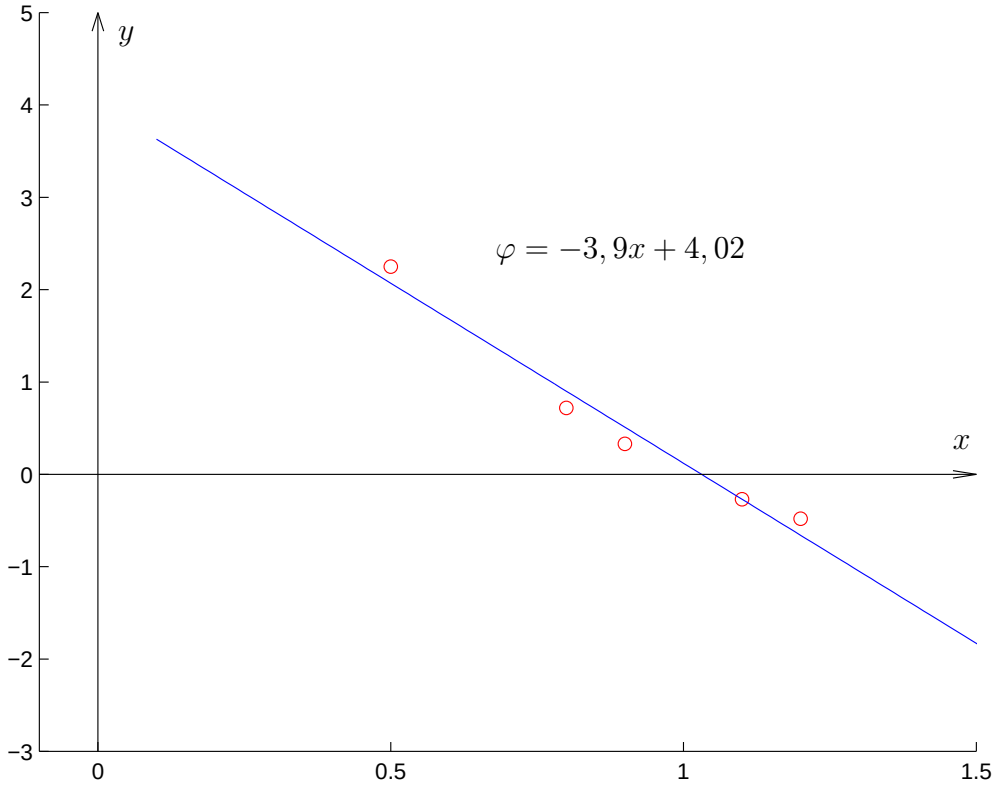
Konkrétně

$$\begin{array}{ll} (1) & (2,25 - c_0 - 0,5c_1)0,5 + (0,72 - c_0 - 0,8c_1)0,8 + \\ & +(0,33 - c_0 - 0,9c_1)0,9 + (-0,27 - c_0 - 1,1c_1)1,1 + \\ & +(-0,48 - c_0 - 1,2c_1)1,1 = 0 \\ (2) & (2,25 - c_0 - 0,5c_1) + (0,72 - c_0 - 0,8c_1) + \\ & +(0,33 - c_0 - 0,9c_1) + (-0,27 - c_0 - 1,1c_1) + \\ & +(-0,48 - c_0 - 1,2c_1) = 0 \end{array}$$

Po sečtení

$$\left.\begin{array}{llllll} (1) & -4,35c_1 & - & 4,5c_0 & = & -1,125 \\ (2) & -4,5c_1 & - & 5c_0 & = & -2,55 \end{array}\right\} \quad \Rightarrow \quad \begin{array}{l} c_1 = -3,9 \\ c_0 = 4,02 \end{array}$$

$$\underline{\varphi = -3,9x + 4,02}$$



Jiný přístup:

Zapišme interpolační podmínky

$$c_1 x_i + c_0 = f(x_i).$$

Jejich přesné splnění samozřejmě obecně nelze zaručit (pouze ve speciálních případech), protože získáme tzv. přeurčenou soustavu (více rovnic než neznámých):

$$\begin{bmatrix} 0,5 & 1 \\ 0,8 & 1 \\ 0,9 & 1 \\ 1,1 & 1 \\ 1,2 & 1 \end{bmatrix} . \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 2,25 \\ 0,72 \\ 0,33 \\ -0,27 \\ -0,48 \end{bmatrix} \qquad \Leftrightarrow \qquad Q.c = f$$

Získanou přeurčenou soustavu řešíme ve smyslu nejmenších čtverců, tj.

$$Q^T Q c = Q^T f$$

(jedná se o stejnou soustavu jako v prvním přístupu).

Poznámka

V případě, že některé hodnoty chceme eliminovat, například důsledkem špatného měření, nebo když jsou hodnoty pro větší x_i zatíženy větší chybou, je vhodné použít váhy, tj. minimalizujeme

$$r(c_i) = \sum_{i=1}^{n} |f(x_i) - \varphi(x_i)|^2 \, w_i$$

Poznámka

Otázkou ještě zůstává volba tvaru $\varphi(x)$.
První možností je volit

$$\varphi_0(x) = 1, \quad \varphi_1(x) = x, \quad \varphi_2(x) = x^2, \ \ldots \ , \quad \varphi_n(x) = x^n.$$

Opět je třeba určit ještě stupeň polynomu (a to např. ze znalosti chování funkce f nebo pomocí statistických metod).

Ukazuje se, že takto volené $\varphi_i(x)$ nejsou nejlepší pro výpočty.

→ Soustava normálních rovnic je pro větší n špatně podmíněná.

Za bázové funkce φ_i je vhodné volit ortogonální polynomy, pro které platí

$$(\varphi_i, \varphi_j) = \delta_{ij} = \begin{array}{ll} 1, & i = j \\ 0, & i \neq j \end{array}$$

Symbol (f, g) představuje skalární součin funkcí, tj.

$$\sum_{i=1}^{n} f(x_i)\, g(x_i) \text{ resp. } \int_a^b f g \, dx \text{ ve spojitém případě.}$$

Příklady ortogonálních polynomů:

Čebyševovy, Legendrovy, Laguerrovy, Gramovy ... viz literatura

→ Soustava normálních rovnic má pro ortogonální polynomy diagonální matici.
(rozmyslet!)

Poznámka

Analogické úvahy jako v případě diskrétní L_2-aproximace pro funkci zadanou tabulkou můžeme provést i pro funkci zadanou na celém intervalu $\langle a, b \rangle$. Získáme tzv. **spojitou L_2-aproximaci** funkce. (Součty přejdou formálně na integrály přes daný interval.) Princip výpočtu demostruje následující příklad.

Spojitá L_2-aproximace

Příklad

Stanovte spojitou L_2-aproximaci funkce $f(x) = \sqrt{x}$, $x \in \langle 0, 1\rangle$ lineární funkcí $\varphi(x) = c_1 x + c_0$.

Minimalizujeme funkci

$$r(c_0, c_1) = \int_0^1 |f(x) - \varphi(x)|^2 \, dx = \int_0^1 (\sqrt{x} - c_0 - c_1 x)^2 \, dx.$$

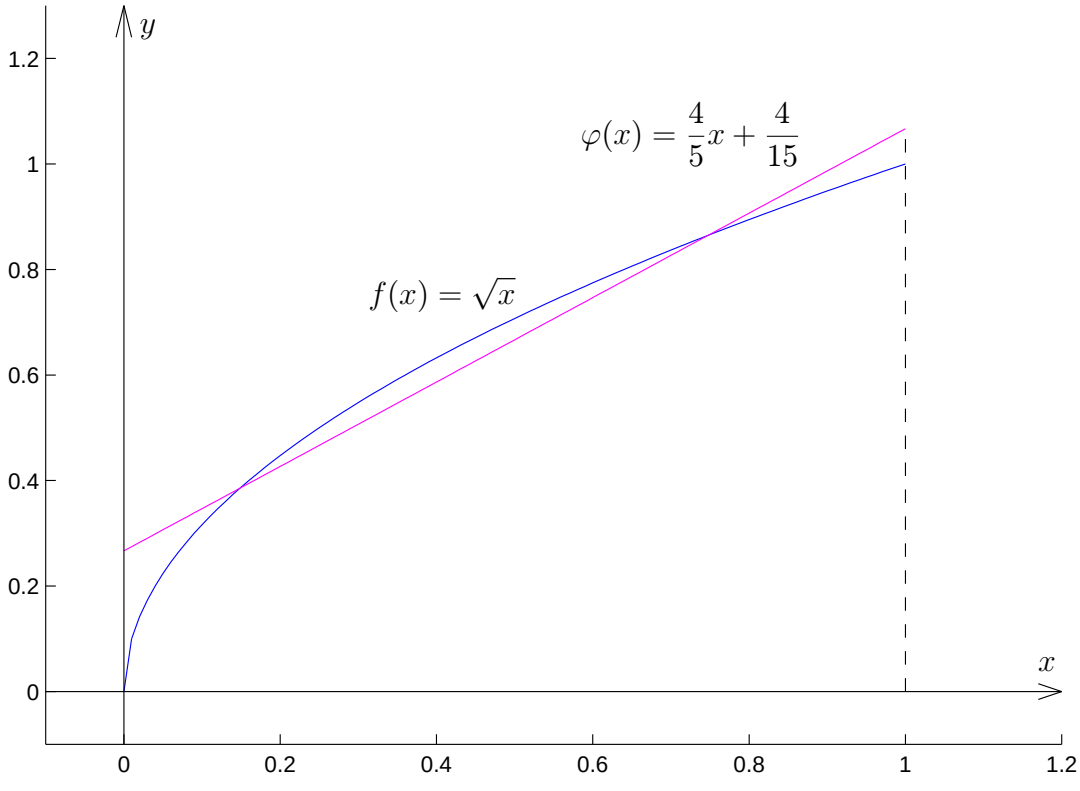
Podmínky minima

$$\left.\begin{array}{ll} (1) & \dfrac{\partial r}{\partial c_1} = -2 \displaystyle\int_0^1 (\sqrt{x} - c_0 - c_1 x) x \, dx = 0 \\ (2) & \dfrac{\partial r}{\partial c_0} = -2 \displaystyle\int_0^1 (\sqrt{x} - c_0 - c_1 x) \, dx = 0 \end{array}\right\} \text{ 2 rovnice pro neznámé } c_0, c_1.$$

$$\begin{array}{ll} (1) & -2\left[\dfrac{2}{5} x^{\frac{5}{2}} - c_0 \dfrac{x^2}{2} - c_1 \dfrac{x^3}{3}\right]_0^1 = 0 \\ (2) & -2\left[\dfrac{2}{3} x^{\frac{3}{2}} - c_0 x - c_1 \dfrac{x^2}{2}\right]_0^1 = 0 \end{array}$$

$$\left.\begin{array}{ll} (1) & \dfrac{2}{5} - \dfrac{1}{2} c_0 - \dfrac{1}{3} c_1 = 0 \\ (2) & \dfrac{2}{3} - c_0 - \dfrac{1}{2} c_1 = 0 \end{array}\right\} \quad \Rightarrow \quad \left.\begin{array}{rcrcl} \dfrac{1}{2} c_0 & + & \dfrac{1}{3} c_1 & = & \dfrac{2}{5} \\ c_0 & + & \dfrac{1}{2} c_1 & = & \dfrac{2}{3} \end{array}\right\} \quad \Rightarrow \quad \begin{array}{rcl} c_1 & = & \dfrac{4}{5} \\ c_0 & = & \dfrac{4}{15} \end{array}$$

$$\varphi(x) = \frac{4}{5} x + \frac{4}{15}$$



□

Poznámka

Obecně lze opět zavést váhovou funkci $w = w(x)$ a minimalizovat

$$r(c_i) = \int_a^b |f(x) - \varphi(x)|^2 w(x)\, dx$$

Fourierova analýza

Do této chvíle jsme se zabývali aproximacemi funkce pouze pomocí polynomů. V úvodu jsme uvedli, že za bázové funkce můžeme volit libovolné funkce. Například pro aproximaci periodických funkcí není vhodné použít polynomy (a to jak ve smyslu interpolace tak ve smyslu L_2-aproximace). Pro aproximaci periodických funkcí je vhodné použít nějaký systém periodických bázových funkcí, např. systém tzv. **trigonometrických polynomů**:

$$\begin{aligned} \varphi_0(x) &= 1 \\ \varphi_{2k-1}(x) &= \cos\frac{2\pi kx}{T} \qquad k = 1, 2, \ldots \\ \varphi_{2k}(x) &= \sin\frac{2\pi kx}{T} \qquad k = 1, 2, \ldots, \end{aligned}$$

kde T představuje periodu zadané funkce (vzdálenost prvního a posledního uzlu v diskrétním případě, resp. délku zadaného intervalu ve spojitém případě). Pro jednoduchost uvažujeme ekvidistantní uzly (v diskrétním případě). Počet uvažovaných bázových funkcí volíme buď menší než je počet zadaných bodů (ve smyslu L_2-aproximace), nebo roven počtu zadaných bodů (ve smyslu interpolace).

Jednoduchým cvičením je ukázat, že systém trigonometrických polynomů je ortogonální (jak v diskrétním tak ve spojitém případě). Ověřte!

Úlohu najít koeficienty c_i u bázových funkcí φ_i z vyjádření

$$\varphi(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + \ldots + c_n\varphi_n(x).$$

nazýváme v tomto případě **Fourierovou analýzou**.

Formálně pouze přeznačíme koeficienty c_i, tj. u bázové funkce $\varphi_0(x) = 1$ použijeme koeficient A_0, u bázových funkcí $\varphi_{2k-1}(x) = \cos(2\pi kx)/T$ použijeme koeficienty A_k a u bázových funkcí $\varphi_{2k}(x) = \sin(2\pi kx)/T$ použijeme koeficienty B_k.

Následující jednoduchý příklad naznačí princip Fourierovy analýzy.

Příklad

Aproximujte 2π-periodickou funkci zadanou tabulkou za použití maximálního počtu bázových funkcí (tj. ve smyslu interpolace).

x_i	0	$\pi/2$	π	$3\pi/2$
$f(x_i)$	12	-4	0	4

Řešení

Ze zadání je zřejmé, že perioda zadané funkce je 2π. Aproximující trigonometrický polynom budeme tedy volit ve tvaru

$$\varphi(x) = A_0 + A_1\cos x + B_1\sin x + A_2\cos 2x.$$

Zapíšeme interpolační podmínky

$$\varphi(x_j) = f(x_j), \quad j = 0, 1, 2, 3,$$

tj.

$$\begin{bmatrix} 1 & \cos x_0 & \sin x_0 & \cos 2x_0 \\ 1 & \cos x_1 & \sin x_1 & \cos 2x_1 \\ 1 & \cos x_2 & \sin x_2 & \cos 2x_2 \\ 1 & \cos x_3 & \sin x_3 & \cos 2x_3 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \end{bmatrix},$$

tj.

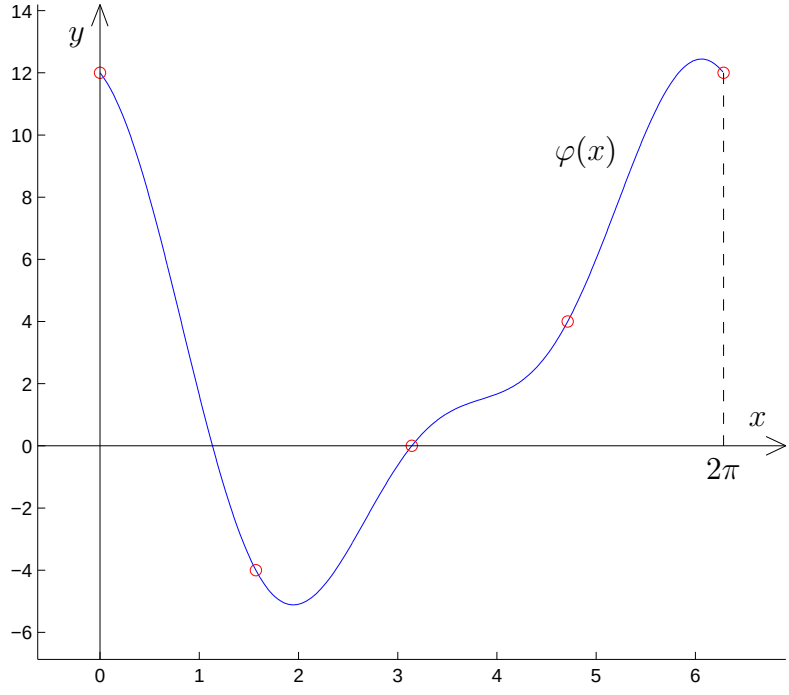
$$\begin{bmatrix} 1 & \cos 0 & \sin 0 & \cos 0 \\ 1 & \cos \pi/2 & \sin \pi/2 & \cos \pi \\ 1 & \cos \pi & \sin \pi & \cos 2\pi \\ 1 & \cos 3\pi/2 & \sin 3\pi/2 & \cos 3\pi \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} 12 \\ -4 \\ 0 \\ 4 \end{bmatrix},$$

tj.

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ B_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} 12 \\ -4 \\ 0 \\ 4 \end{bmatrix}.$$

Vyřešením soustavy získáme hledané koeficienty $A_0 = 3$, $A_1 = 6$, $B_1 = -4$, $A_2 = 3$ a tím i aproximující trigonometrický polynom

$$\underline{\varphi(x) = 3 + 6\cos x - 4\sin x + 3\cos 2x.}$$



□

Úlohu a řešení Fourierovy analýzy lze formulovat elegantně použitím komplexní proměnné. Uvažujme pro jednoduchost lichý počet bázových funkcí ($N = 2L + 1$) a periodu dané funkce 2π. Potom má aproximující funkce tvar

$$\varphi(x) = A_0 + \sum_{k=1}^{L} (A_k \cos kx + B_k \sin kx). \qquad (*)$$

Pro funkce $\sin x$ a $\cos x$ platí vztahy

$$\cos x = \frac{e^{ix} + e^{-ix}}{2} \ , \quad \sin x = \frac{e^{ix} - e^{-ix}}{2i} = -\frac{1}{2}\, i\, (e^{ix} - e^{-ix})$$

a tedy

$$\varphi(x) = A_0 + \sum_{k=1}^{L} \left(\frac{1}{2}\, A_k\, (e^{ikx} + e^{-ikx}) - \frac{1}{2}\, i\, B_k\, (e^{ikx} - e^{-ikx}) \right) =$$

$$= A_0 + \sum_{k=1}^{L} \left(\frac{1}{2}\, (A_k - i\, B_k)\, e^{ikx} + \frac{1}{2}\, (A_k + i\, B_k)\, e^{-ikx} \right).$$

Označíme-li

$$C_0 = A_0 \ , \quad C_k = \frac{1}{2}\, (A_k - i\, B_k) \ , \quad C_{-k} = \frac{1}{2}\, (A_k + i\, B_k)$$

dostaneme

$$\underline{\varphi(x) = \sum_{k=-L}^{L} C_k\, e^{ikx}.}$$

Pro koeficienty dostaneme vynásobením $(*)$ jednotlivými bázovými funkcemi, využitím jejich ortogonality a interpolačních podmínek předpisy:

$$A_0 = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j)$$

$$A_k = \frac{2}{N} \sum_{j=0}^{N-1} f(x_j)\, \cos kx_j$$

$$B_k = \frac{2}{N} \sum_{j=0}^{N-1} f(x_j)\, \sin kx_j$$

$$C_k = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j)\, e^{-ikx_j}$$

Poznámka

Vezmeme-li aproximující polynom o menším počtu bázových funkcí než je počet zadaných bodů, jedná se o aproximaci ve smyslu metody nejmenších čtverců, tj. diskrétní L_2-aproximaci. Potom obecně nemohou být splněny interpolační podmínky přesně (pouze ve speciálních případech).

Poznámka

Výpočet koeficientů C_k představuje sčítání konečné řady. Uvažujeme-li počet aproximujících bázových funkcí N jako mocninu čísla 2 (tj. $N = 2^M$), lze odvodit velmi rychlý a efektivní algoritmus pro výpočet koeficientů C_k. Tento algoritmus se potom nazývá **rychlá Fourierova analýza**. Podrobněji viz literatura.

DERIVACE FUNKCE

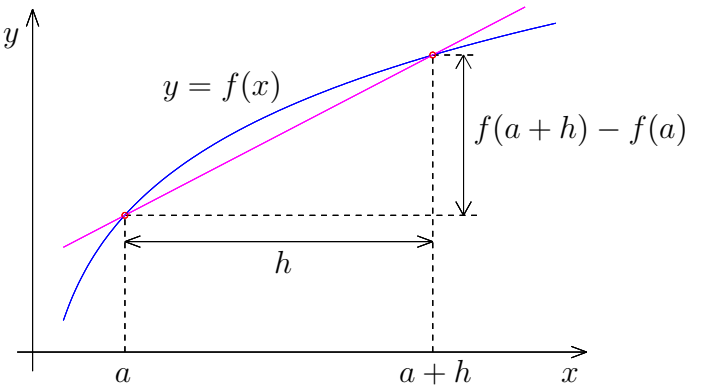
Na úvod si připomeňme definici derivace reálné funkce jedné reálné proměnné.

Definice: Existuje-li pro danou funkci $f : \mathbb{R} \to \mathbb{R}$ vlastní (tj. konečná) limita

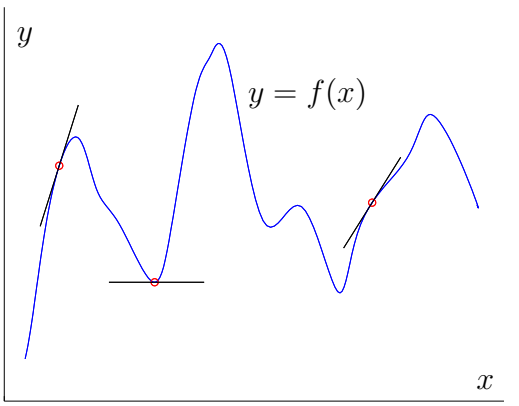
$$\lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

říkáme, že funkce $f(x)$ má v bodě a derivaci. Příslušnou limitu značíme $f'(a)$.

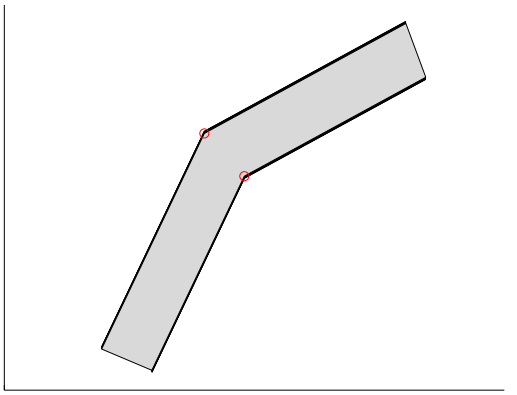
Poznámka: Geometrický význam derivace $f'(a)$ (viz obrázek) je směrnice tečny křivky dané rovnicí $y = f(x)$ v bodě a (neboť tečna v bodě a je limitní polohou sečny pro $h \to 0$). Fyzikálně značí derivace funkce $y = f(x)$, kde x je čas a y dráha pohybu, limitu z průměrné rychlosti, tedy okamžitou rychlost v čase a.

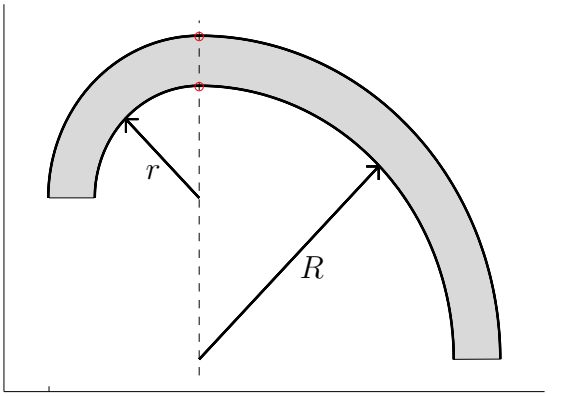


Poznámka: Pro danou funkci $f(x)$ vyjadřuje derivace $f'(x_0)$ míru „stoupání“, resp. „klesání“ v bodě x_0 .



Poznámka: Geometrický význam druhé derivace $f''(x_0)$ souvisí s mírou „zakřivení“ grafu funce f v bodě x_0. Pro ilustraci si uveďme dva příklady špatné výstavby pomyslné silnice. V prvním případě ve vyznačených bodech vůbec neexistuje první derivace (derivace se v těchto bodech mění *skokově*).





Ve druhém případě je tvořena komunikace částmi dvou kružnic o různých poloměrech r a R. První derivace je v tomto případě spojitá i ve vyznačených bodech (říkáme, že funkce je „hladká“), nevhodnost tohoto případu je způsobena skokem v hodnotách tzv. **křivosti**, která je v případě kružnice konstantní a je rovna převrácené hodnotě poloměru (tj. $1/r$ a $1/R$). Pro úplnost dodejme, že vztah pro křivost je dán vzorcem

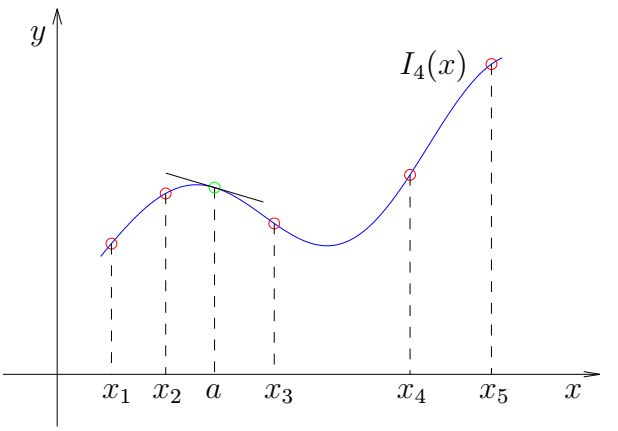
$$k = \frac{f''^2}{(1+f'^2)^3},$$

kde f'' je druhá derivace funkce f.

Způsoby odvození vzorců pro výpočet derivace

1. Odvození pomocí **interpolačního polynomu**

   Pro funkci f, která je zadána tabulkou, sestrojíme interpolační polynom a derivaci funkce f v bodě a ztotožníme s derivací tohoto interpolačního polynomu v bodě a.



$$f'(a) \approx I_n'(a)$$

$$f^{(k)}(a) \approx I_n^{(k)}(a)$$

Poznámky:

- Stupeň polynomu nemůže být nižší než řád počítané derivace.
- Pro jednoduchost hledáme hodnotu derivace v uzlovém bodě a navíc uvažujeme ekvidistantní uzly s krokem h.

2. Odvození pomocí **Taylorova rozvoje**

Pro dostatečně hladkou funkci f platí (pro $h > 0$):

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(\xi_1), \quad \xi_1 \in (x_0, x_0 + h)$$

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(\xi_2), \quad \xi_2 \in (x_0 - h, x_0)$$

Z první rovnice potom plyne vztah

$$f'(x_0) = \underbrace{\frac{f(x_0 + h) - f(x_0)}{h}}_{=D_P f(x_0,h)} - \frac{1}{2}hf''(\xi_1)$$

Podobně ze druhé rovnice

$$f'(x_0) = \underbrace{\frac{f(x_0) - f(x_0 - h)}{h}}_{=D_L f(x_0,h)} + \frac{1}{2}hf''(\xi_2)$$

Obdrželi jsme dva základní **dvoubodové** vzorce $D_P f(x_0, h)$ a $D_L f(x_0, h)$, tzv. pravou a levou poměrnou diferenci.

Podobně odvodíme další vzorce pomocí Taylorova rozvoje vyšších řádů. Platí:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{6}f'''(\xi_1), \quad \xi_1 \in (x_0, x_0 + h)$$

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) - \frac{h^3}{6}f'''(\xi_2), \quad \xi_2 \in (x_0 - h, x_0)$$

Po odečtení obdržíme:

$$f(x_0 + h) - f(x_0 - h) = 2hf'(x_0) + \frac{h^3}{6}(f'''(\xi_1) + f'''(\xi_2))$$

Odtud vyjádříme první derivaci a získáme **tříbodový** vzorec $D_C f(x_0, h)$, tzv. centrální poměrnou diferenci

$$f'(x_0) = \underbrace{\frac{f(x_0 + h) - f(x_0 - h)}{2h}}_{D_C f(x_0,h)} - \underbrace{\frac{h^2}{12}(f'''(\xi_1) + f'''(\xi_2))}_{O(h^2)}$$

Uvedené vzorce jsou pro výpočet první derivace $f'(x_0)$, pro výpočet druhé derivace $f''(x_0)$ můžeme použít například vzorec, který dostaneme po sečtení vztahů:

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{6}f'''(x_0) + \frac{h^4}{24}f^{(4)}(\xi_1), \quad \xi_1 \in (x_0, x_0+h)$$

$$f(x_0-h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) - \frac{h^3}{6}f'''(x_0) + \frac{h^4}{24}f^{(4)}(\xi_2), \quad \xi_2 \in (x_0-h, x_0)$$

$$f(x_0+h) + f(x_0-h) = 2f(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^4}{24}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2))$$

Odtud vyjádříme druhou derivaci a získáme **tříbodový** vzorec pro druhou derivaci

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - \underbrace{\frac{h^2}{24}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2))}_{O(h^2)}$$

Poznámka: Samozřejmě lze odvodit řadu dalších vzorců, přičemž platí, že čím více bodů použijeme, tím bude řád chyby vyšší.

Příklad: Pomocí uvedených tří vzorců vypočtěte přibližnou hodnotu první derivace funkce $f(x) = e^x(1-x)$ v bodě $x_0 = 1$. Použijte krok $h = 0,1$.

Řešení: (Nejprve si pro kontrolu analyticky zjistíme přesnou hodnotu první derivace funkce f bodě x_0:

$$f'(x) = e^x(1-x) + e^x(-1) = -xe^x, \quad \text{tj.} \quad f'(1) = -1e^1 = -e \approx -2,7182.)$$

Nyní použijeme pravou, levou a centrální poměrnou diferenci:

1. $D_P f(x_0,h) = \dfrac{f(x_0+h) - f(x_0)}{h} = \dfrac{e^{1,1}(1-1,1) - e^1(1-1)}{0,1} =$

$$= \frac{-0,1e^{1,1}}{0,1} = -e^{1,1} \approx \mathbf{-3,0041} \quad \text{tj. chyba je přibližně } 0,2858$$

2. $D_L f(x_0,h) = \dfrac{f(x_0) - f(x_0-h)}{h} = \dfrac{e^1(1-1) - e^{0,9}(1-0,9)}{0,1} =$

$$= \frac{-0,1e^{0,9}}{0,1} = -e^{0,9} \approx \mathbf{-2,4596} \quad \text{tj. chyba je přibližně } 0,2586$$

3. $D_C f(x_0,h) = \dfrac{f(x_0+h) - f(x_0-h)}{2h} = \dfrac{e^{1,1}(1-1,1) - e^{0,9}(1-0,9)}{0,2} =$

$$= \frac{-0,1e^{1,1} - 0,1e^{0,9}}{0,2} = -\frac{e^{1,1} + e^{0,9}}{2} \approx \mathbf{-2,7318}$$

tj. chyba je přibližně $0,0136$

Všimněme si velikosti chyb v jednotlivých případech. Potvrzuje se fakt, že chyba prvních dvou (dvoubodových) vzorců je řádu h, tj. v řádu desetin a chyba posledního (tříbodového) vzorce je řádu h^2, tj. v řádu setin. □

Podmíněnost úlohy numerického derivování

Uvažujme nyní např. vzorec s pravou diferencí $D_P f(x_0, h)$, tj. platí

$$f'(x_0) = \underbrace{\frac{f(x_0+h)-f(x_0)}{h}}_{D_P f(x_0,h)} - \underbrace{\frac{1}{2}hf''(\xi)}_{\text{chyba metody}}$$

Chybu metody označme r_1. Platí-li $|f''(x)| < M$ pro $x \in (x_0, x_0+h)$, potom $|r_1| \leq \frac{M}{2}h$.
Dále je třeba uvážit chyby měření, resp. zaokrouhlovací chyby, které označíme r_2.
Označíme-li

$f(x_0), f(x_0+h)$ ..... přesné hodnoty

$f^\star(x_0), f^\star(x_0+h)$ ..... vstupní hodnoty

Potom pro r_2 platí

$$r_2 = \underbrace{\frac{f(x_0+h)-f(x_0)}{h}}_{\text{přesná hodnota vzorce}} - \underbrace{\frac{f^\star(x_0+h)-f^\star(x_0)}{h}}_{\text{vypočtená hodnota vzorce}}$$

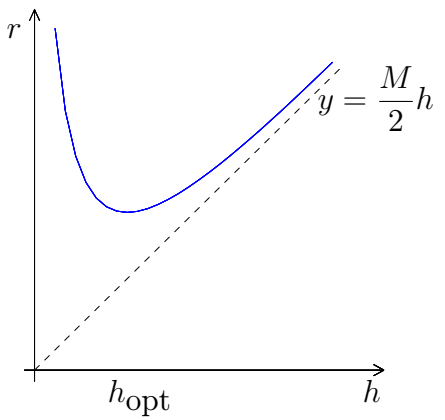
A dále

$$\begin{aligned} |r_2| &= \left|\frac{f(x_0+h)-f^\star(x_0+h)}{h} + \frac{f^\star(x_0)-f(x_0)}{h}\right| \leq \\ &\leq \frac{|f(x_0+h)-f^\star(x_0+h)|}{h} + \frac{|f^\star(x_0)-f(x_0)|}{h} \leq \\ &\leq \frac{\varepsilon}{h} + \frac{\varepsilon}{h} = \frac{2\varepsilon}{h} \end{aligned}$$

Využili jsme zde odhady $|f^\star(x_0+h)-f(x_0+h)| \leq \varepsilon$ a $|f^\star(x_0)-f(x_0)| \leq \varepsilon$, kde číslo ε může představovat např. strojovou přesnost.

Pro celkovou chybu r potom platí

$$|r| \leq |r_1| + |r_2| \leq \frac{M}{2}h + \frac{2\varepsilon}{h}$$



- Úloha numerického derivování je **špatně podmíněná** ! (pro zmenšující se h roste chyba)
- Lze najít optimální krok h_{opt}

Poznámka: Na základě špatné podmíněnosti se zdá, že nebude možné při výpočtu derivace dosáhnout libovolné přesnosti. Zvýšení přesnosti ale můžeme dosáhnout

1) použitím vzorce s chybou vyššího řádu
2) použitím tzv. Richardsonovy extrapolace

Věnujme nyní pozornost **Richardsonově extrapolaci**. Je třeba zdůraznit, že se jedná o **obecný princip**, který se nepoužívá jen u numerického výpočtu derivace. Myšlenka vychází z toho, že na základě znalosti výrazu pro rozvoj chyby využijeme dvou přibližných výsledků k získání třetího, který bude přesnější. Tento proces eliminace chyb budeme demonstrovat např. na poměrné centrální diferenci

$$f'(x_0,h) = \underbrace{\frac{f(x_0+h)-f(x_0-h)}{2h}}_{D_C f(x_0,h)} - \frac{h^2}{6}f'''(\xi_1), \qquad \xi_1 \in (x_0-h, x_0+h). \tag{1}$$

Podobný vztah musí platit i v případě, že použijeme místo kroku h krok $2h$, tj.

$$f'(x_0,2h) = \underbrace{\frac{f(x_0+2h)-f(x_0-2h)}{2\cdot 2h}}_{D_C f(x_0,2h)} - \frac{(2h)^2}{6}f'''(\xi_2), \qquad \xi_2 \in (x_0-2h, x_0+2h). \tag{2}$$

Pro jednoduchost předpokládáme, že hodnoty $f'''(\xi_1)$ a $f'''(\xi_2)$ jsou si rovny. Vhodnou kombinací (1) a (2) dosáhneme eliminace chyby řádu h^2, tj. od čtyřnásobku rovnice (1) odečteme rovnici (2) a výsledek dělíme třemi (4-1). Dostaneme přesnější aproximaci derivace funkce f v bodě x_0:

$$f'(x_0) \approx \frac{4f'(x_0,h)-f'(x_0,2h)}{3} = \frac{4}{3}f'(x_0,h) - \frac{1}{3}f'(x_0,2h) \tag{3}$$

Poznámka: V názvu metody se objevuje slovo extrapolace. Je to proto, že nová hodnota derivace je lineární kombinací dvou hodnot, ovšem neleží mezi těmito hodnotami (kdyby tomu tak bylo, mluvili bychom o interpolaci).

Poznámka: Algoritmus Richardsonovy extrapolace lze samozřejmě použít opakovaně pro eliminaci chyb vyšších řádů. Tato metoda je potom velmi efektivní.

Příklad: Použijte opakovanou Richardsonovu extrapolaci pro výpočet derivace funkce $f(x) = \ln x$ v bodě $x_0 = 3$ pomocí centrální poměrné diference s kroky $h = 0,8;\ 0,4;\ 0,2$ a $0,1$.

Řešení: Dá se ukázat (viz. odvození), že pro dostetečně hladkou funkci f platí tento vztah

$$f'(x_0) = \underbrace{\frac{f(x_0+h)-f(x_0-h)}{2h}}_{D_C f(x_0,h)} + \underbrace{c_1h^2 + c_2h^4 + c_3h^6 + \ldots}_{\text{rozvoj chyby}},$$

kde čísla c_1, c_2, c_3 představují kontanty obsahující příslušné derivace.

Pro přehlednost budeme výsledky zapisovat do tabulky:

h	$f'(x_0, h)$	po 1. korekci - vztah (3)	po 2. korekci - vztah (4)
$0,8$	$0,341589$		
$0,4$	$0,335329$	$\frac{4}{3}0,335329 - \frac{1}{3}0,341589 = 0,333242$	
$0,2$	$0,333828$	$\frac{4}{3}0,333828 - \frac{1}{3}0,335329 = 0,333327$	$\frac{16}{15}0,333327 - \frac{1}{15}0,333242 = \mathbf{0,333332}$
$0,1$	$0,333456$	$\frac{4}{3}0,333456 - \frac{1}{3}0,333828 = 0,333332$	$\frac{16}{15}0,333332 - \frac{1}{15}0,333327 = \mathbf{0,333332}$

Ve výpočtu jsme použili jednak 1. korekci pro eliminaci chyby řádu h^2, ale dále také 2. korekci, která eliminovala chybu řádu h^4. Vztah (4) pro 2. korekci jsme dostali podobně jako vztah (3), tj.

$$\begin{aligned} f'(x_0,h) &= D_C f(x_0,h) \;+\; c_2 h^4 \quad /\cdot 2^4 \\ f'(x_0,2h) &= D_C f(x_0,2h) \;+\; c_2 (2h)^4 \quad /\cdot (-1) \end{aligned}$$

$$f'(x_0) \approx \frac{2^4 f'(x_0,h) - f'(x_0,2h)}{2^4 - 1} \;=\; \frac{16}{15} f'(x_0,h) - \frac{1}{15} f'(x_0,2h) \tag{4}$$

V tabulce chybí sloupec pro 3. korekci. Důvod je ten, že se hodnoty, ze kterých by se extrapolovala nová hodnota, rovnají (dostali bychom to samé číslo). Výraz pro 3. korekci bychom opět odvodili podobně jako vztah (4), pouze místo 4 mocniny by se v něm objevila 6 mocnina.

Hodnota hledané derivace funkce $f(x) = \ln x$ v bodě $x_0 = 3$ je **0,333332**. Pro úplnost dodejme, že přesná hodnota derivace je $f'(x) = \frac{1}{x}$, tj. $f'(3) = \frac{1}{3}$.

□

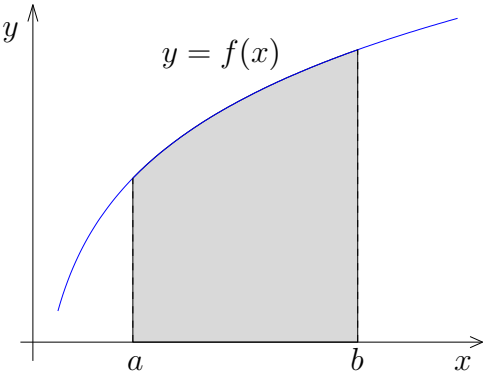
URČITÝ INTEGRÁL FUNKCE

Formulace: Naším cílem je určit přibližnou hodnotu určitého integrálu

$$I(f) = \int_a^b f(x)\,dx,$$

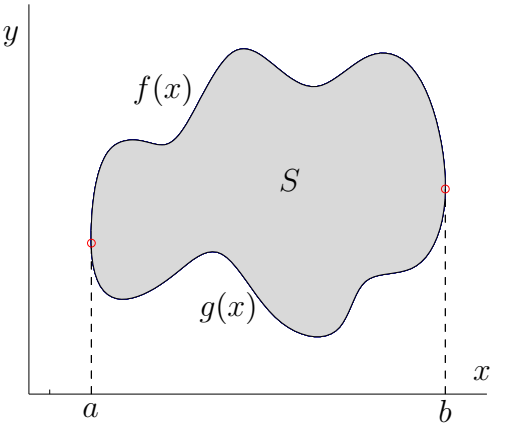
kde předpokládáme, že funkce f je na intervalu $\langle a, b\rangle$ integrovatelná.

Poznámka: Geometrický význam integrálu $I(f)$ (viz obrázek) je obsah plochy mezi grafem funkce f a osou x na intervalu $\langle a, b\rangle$.



Numerické metody výpočtu integrálu užíváme zejména tehdy, když $I(f)$ není možno spočítat analyticky (velmi častý případ) nebo je sice analytické řešení možné, ale je velmi pracné. V případě že máme zadánu funkci f tabulkou, není ani jiný přístup možný.

Příklad: Chceme-li určit obsah plohy mezi grafy funkcí f a g, užijeme určitý integrál.



Pro obsah potom platí $$S = \int_a^b f(x) - g(x)\,dx$$

Přirozený princip numerických metod pro výpočet integrálu vychází z aproximace funkce. Danou funkci f nahradíme její vhodnou aproximací φ a jako aproximaci integrálu $I(f)$ prohlásíme hodnotu integrálu $I(\varphi)$, tj.

$$I(f) \approx I(\varphi) = \int_a^b \varphi(x)\,dx.$$

Poznámka: Narozdíl od výpočtu derivace je výpočet integrálu stabilní, protože je-li φ dobrou aproximací funkce f na intervalu $\langle a, b\rangle$, je integrál $I(\varphi)$ dobrou aproximací $I(f)$.

$$\left|\int_a^b f(x)\,dx - \int_a^b \varphi(x)\,dx\right| \le \int_a^b |f(x)-\varphi(x)|\,dx \le (b-a)\underbrace{\sup_{x\in\langle a,b\rangle} |f(x)-\varphi(x)|}_{\varepsilon}.$$

Princip většiny metod na výpočet určitého integrálu $\int_a^b f(x)\,dx$ je založen na tom, že interval $\langle a, b\rangle$ rozdělíme na N podintervalů $\langle x_k, x_{k+1}\rangle$ tak, že

$$a = x_0 < x_1 < x_2 < \ldots\ldots < x_{N-1} < x_N = b.$$

Na těchto podintervalech nahradíme funkci f polynomem a integrujeme tento polynom. Vzorce pro výpočet integrálu (tzv. **kvadraturní vzorce**) na intervalech $\langle x_k, x_{k+1}\rangle$ budeme nazývat **základní** a vzorec pro výpočet hodnoty integrálu přes celý interval $\langle a, b\rangle$ budeme nazývat **složený** (složený kvadraturní vzorec je dán součtem základních kvadraturních vzorců).

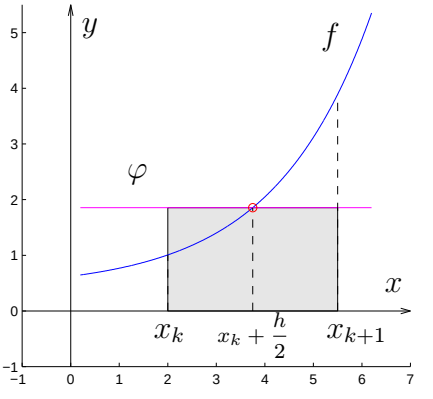
Pro jednoduchost budeme předpokládat, že jsou všechny podintervaly $\langle x_k, x_{k+1}\rangle$ stejně velké, tj. máme tzv. ekvidistantní uzly, které můžeme vyjádřit jako $x_k = x_0 + kh$, kde $k = 0, 1, \ldots, N-1$ a $h = \dfrac{b-a}{N}$.

Uveďme si nyní tři nejjednodušší základní kvadraturní vzorce, které patří mezi tzv. **Newtonovy-Cotesovy kvadraturní vzorce**.

1) Obdélníkové pravidlo (funkci f nahrazujeme konstantní funkcí φ)

$$\int_{x_k}^{x_{k+1}} f(x)\,\mathrm{d}x \approx$$

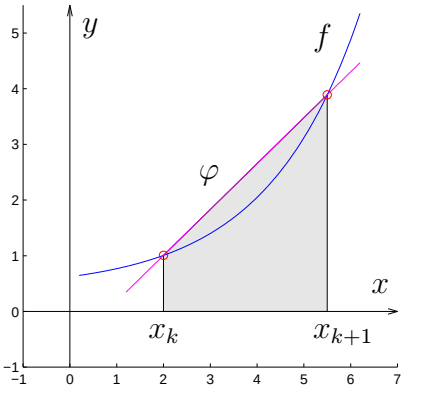
$$\approx h\cdot f(x_k + \frac{h}{2}) \equiv R_Z(f,h)$$



2) Lichoběžníkové pravidlo (funkci f nahrazujeme lineární funkcí φ)

$$\int_{x_k}^{x_{k+1}} f(x)\,\mathrm{d}x \approx$$

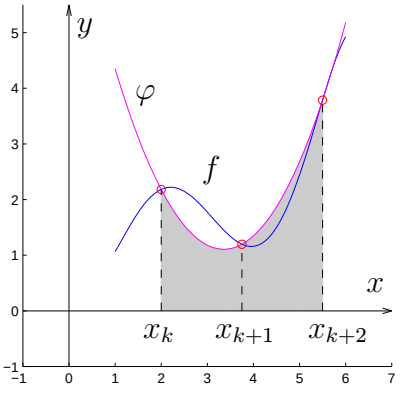
$$\approx \frac{h}{2}\left[f(x_k) + f(x_{k+1})\right] \equiv T_Z(f,h)$$



3) Simpsonovo pravidlo (funkci f nahrazujeme kvadratickou funkcí φ)

$$\int_{x_k}^{x_{k+2}} f(x)\,\mathrm{d}x \approx$$

$$\approx \frac{h}{3}\left[f(x_k) + 4f(x_{k+1}) + f(x_{k+2})\right] \equiv S_Z(f,h)$$



Příklad k procvičení: Odvoďte základní vzorec pro Simpsonovo pravidlo.

Poznámka: Základní vzorce v předchozím textu jsme odvodili na základě geometrické interpretace. V případě, že bychom chtěli vyjádřit současně i vztahy pro chyby těchto vzorců, museli bychom použít k odvození Taylorův rozvoj. Získali bychom tyto vztahy:

$$\int_{x_k}^{x_{k+1}} f(x)\,dx = R_Z(f,h) + \frac{h^3}{24} f''(\xi)$$

$$\int_{x_k}^{x_{k+1}} f(x)\,dx = T_Z(f,h) - \frac{h^3}{12} f''(\xi)$$

$$\int_{x_k}^{x_{k+2}} f(x)\,dx = S_Z(f,h) - \frac{h^5}{90} f^{(IV)}(\xi)$$

Příklad: Pomocí výše uvedených Newtonových-Cotesových vzorců vypočtěte integrál

$$\int_1^{1,2} e^x \, dx.$$

Řešení: (Přesné řešení je $[e^x]_1^{1,2} = e^{1,2} - e^1 \doteq 0{,}601835$.)

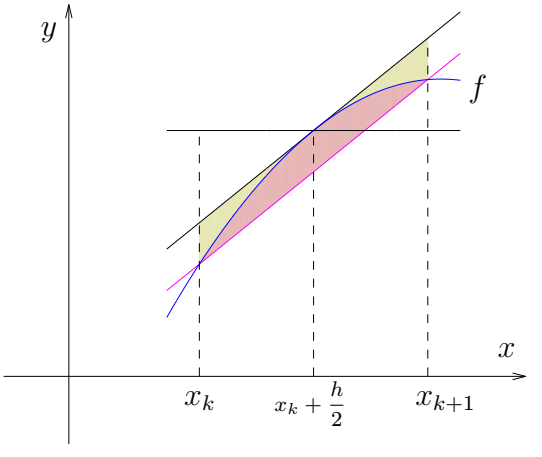
$$R_Z(e^x;\, 0, 2) = 0,2e^{1,1} \doteq 0{,}600833 \qquad \text{chyba: } 0{,}001002$$

$$T_Z(e^x;\, 0, 2) = \frac{0,2}{2}(e^{1,0} + e^{1,2}) \doteq 0{,}603839 \qquad \text{chyba: } 0{,}002003$$

$$S_Z(e^x;\, 0, 1) = \frac{0,1}{3}(e + 4e^{1,1} + e^{1,2}) \doteq 0{,}601835 \qquad \text{chyba: } 0{,}000000$$

□

Poznámka: Všimněme si chyb. U obdélníkového pravidla vyšla chyba menší než u lichoběžníkového, přestože u lichoběžníkového pravidla jsme funkci f aproximovali „lepší“ funkcí φ (lineární). Chyba u Simpsonova pravidla vyšla menší než u ostatních. Tyto výsledky potvrzují vztahy pro chyby jednotlivých vzorců na minulé straně. Fakt, že obdélníkové pravidlo je přesnější než lichoběžníkové můžeme demonstrovat na obrázku:



Chceme-li získat složené kvadraturní vzorce, je třeba sečíst základní kvadraturní vzorce. Pro uvedené základní Newtonovy-Cotesovy dostaneme tyto složené kvadraturní vzorce:

$$\begin{aligned}
R(f,h) &\equiv h\cdot \sum_{k=0}^{N-1} f(x_k + \frac{h}{2}) \\
T(f,h) &\equiv \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{N-1}) + f(x_N)\right] = \\
&\quad = h\cdot \left[\frac{1}{2}f(x_0) + \sum_{k=1}^{N-1} f(x_k) + \frac{1}{2}f(x_N)\right] \\
S(f,h) &\equiv \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \right. \\
&\quad \left. + \ldots + 2f(x_{N-2}) + 4f(x_{N-1}) + f(x_N)\right]
\end{aligned}$$

Pro chyby složených vzorců potom platí:

$$\begin{aligned} I &= R(f,h) + (b-a)\frac{h^2}{24}f''(\xi) \\ I &= T(f,h) - (b-a)\frac{h^2}{12}f''(\xi) \\ I &= S(f,h) - (b-a)\frac{h^4}{180}f^{(IV)}(\xi) \end{aligned}$$

Pro zpřesňování výsledků nám, stejně jako u numerického výpočtu derivace, poslouží **Richardsonova extrapolace**. Někdy se tato metoda také nazývá Rungeova metoda nebo metoda poloviční kroku. Ukažme si nyní ještě jednou, jak se vztah pro zpřesnění odvodí:

Předpokládejme, že výraz pro chybu má tvar $e(f) = h^k M, \quad h = \dfrac{b-a}{N}$.
Přesná hodnota integrálu je potom

$$I = K(h) + h^k M \tag{5}$$

Integrál vypočteme stejným vzorcem, ale s krokem $\dfrac{h}{2}$.
Dostaneme

$$I = K\left(\frac{h}{2}\right) + \underbrace{\left(\frac{h}{2}\right)^k M_1}_{\text{ozn. } \varepsilon} \qquad \Rightarrow \qquad h^k = \frac{\varepsilon\, 2^k}{M_1} \tag{6}$$

Dosadíme-li h^k do (5), získáme

$$I = K(h) + \frac{\varepsilon\, 2^k\, M}{M_1} \tag{5'}$$

Předpokládáme-li, že se hodnota derivace ve výrazu $e(f)$ pro chybu příliš nemění (tj. $M \approx M_1$), potom $\dfrac{M}{M_1} \approx 1$ a pro (5′) a (6) musí platit

$$K\left(\frac{h}{2}\right) + \varepsilon \approx K(h) + 2^k\varepsilon$$

Odtud plyne odhad chyby ε

$$\varepsilon \approx \frac{1}{2^k - 1}\left[K\left(\frac{h}{2}\right) - K(h)\right]$$

a přesnější hodnota integrálu je potom

$$I = K\left(\frac{h}{2}\right) + \frac{1}{2^k - 1}\left[K\left(\frac{h}{2}\right) - K(h)\right].$$

Příklad: Pomocí lichoběžníkového pravidla vypočtěte $\int_1^5 \ln x \, dx$. Ke zpřesnění použijte Richardsonovu extrapolaci.

Řešení: Pro rozvoj chyby lichoběžníkového pravidla platí

$$I = T(f,h) + \underbrace{a_1 h^2}_{\text{tab. k=2}} + \underbrace{a_2 h^4}_{\text{tab. k=4}} + a_3 h^6 + \dots$$

Výsledky opět zapíšeme do tabulky

h	$T(f,h)$	1. zpřesnění ($k=2$)	2. zpřesnění ($k=4$)
4	$\frac{4}{2}(\ln 1 + \ln 5) = 3,2188$		
2	$\frac{2}{2}(\ln 1 + 2\ln 3 + \ln 5 = 3,8066$	$\frac{3,8066 - 3,2188}{3} + 3,8066 = \underline{4,0025}$	
1	$\frac{1}{2}(\ln 1 + 2\ln 2 + 2\ln 3 + 2\ln 4 + \ln 5) = 3,9827$	$\frac{3,9827 - 3,8066}{3} + 3,9827 = \underline{4,0414}$	$\frac{4,0414 - 4,0025}{15} + 4,0414 = \underline{\underline{4,04399}}$

Pro kontrolu uveďme přesnou hodnotu integrálu:

$$\int_1^5 \ln x \, \mathrm{d}x = \left| \begin{array}{ll} u = \ln x & v' = 1 \\ u' = \dfrac{1}{x} & v = x \end{array} \right| = [x \, \ln x]_1^5 - \int_1^5 \mathrm{d}x = 5\ln 5 - 4 \doteq \underline{\underline{4,04719}}$$

□

Poznámka: Newtonovy-Cotesovy vzorce používají $(m+1)$ ekvidistantních uzlů a integrují přesně polynomy až do m-tého stupně (máme na mysli základní vzorce na intervalu (x_k, x_{k+m})). Pro zvýšení přesnosti by se mohlo zdát výhodné použít více uzlů a funkci f aproximovat polynomem vyššího řádu. Ze zkušeností z aproximace funkce polynomem ovšem víme, že limitní případ polynomu stupně $m \to \infty$ nemusí odpovídat původní funkci (říkáme, že Newton-Cotesovy vzorce **nejsou konvergentní**).

Další skupinou metod pro výpočet hodnoty určitého integrálu jsou tzv. **Gaussovy kvadraturní vzorce**. Jejich princip spočívá v tom, že se snažíme, aby kvadraturní vzorec integroval přesně polynomy co možná nejvyššího řádu. Obecně kvadraturní vzorec budeme uvažovat ve tvaru

$$K(f) = \sum_{i=0}^{m} w_i f(x_i),$$

kde w_i jsou tzv.*váhy* a x_i jsou *uzly*.

Uvažujeme-li, že na základním intervalu máme $m+1$ bodů, dá se ukázat, že nejvyšší možný stupeň polynomu, který se pomocí kvadraturního vzorce integruje přesně, je $2m+1$ (tomuto číslu říkáme **algebraický řád přesnosti**). U Newtonových-Cotesových vzorců byl stupeň m. Cenou za vyšší přesnost budou ovšem **neekvidistantní uzly**. V následujícím příkladě je ukázán postup pro nalezení nejjednoduššího Gaussova kvadraturního vzorce.

Příklad: Určete Gaussův kvadraturní vzorec pro $m = 0$ (tj. v intervalu uvažujeme pouze jeden uzel) a pro interval $\langle -1, 1\rangle$.

Řešení: Pro $m = 0$ má hledaný kvatraturní vzorec tvar $K(f) = w_0 f(x_0)$, kde vystupují 2 neznámé w_0 a x_0. Víme, že vzorec musí přesně integrovat:

1) konstantu

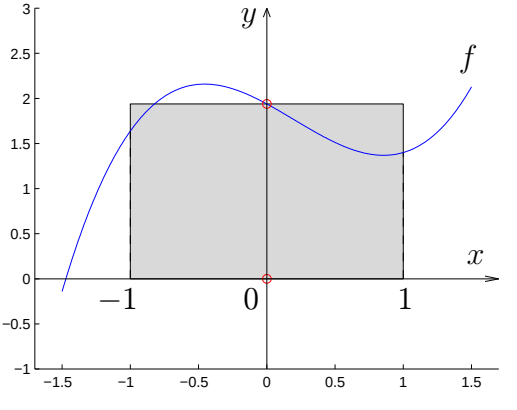
$$\int_{-1}^{1} b\,dx = 2b \overset{\text{pož.}}{=} w_0 \cdot \overbrace{f(x_0)}^{b} \quad \Rightarrow \quad w_0 = 2.$$

2) lineární funkci

$$\int_{-1}^{1} (ax+b)\,dx = \left[a\frac{x^2}{2} + bx\right]_{-1}^{1} = \underbrace{\frac{a}{2} - \frac{a}{2}}_{=0} + 2b \overset{\text{pož.}}{=} w_0 \cdot \overbrace{f(x_0)}^{ax_0+b} \quad \Rightarrow \quad 2b = 2(ax_0+b) \Rightarrow x_0 = 0.$$

Nejjednodušší Gaussův kvadraturní vzorec je:

$$K(f) = \int_{-1}^{1} f(x)\,\mathrm{d}x = 2f(0) + \underbrace{\frac{1}{3}f''(\xi)}_{\text{chyba}}$$

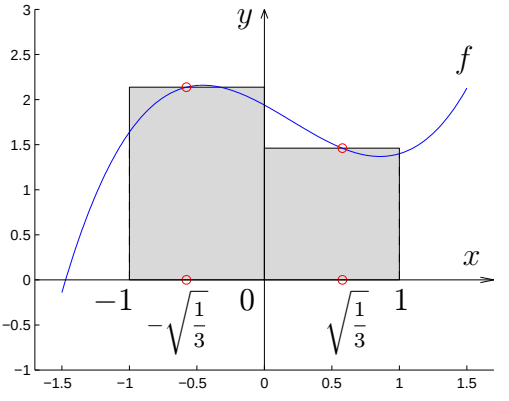


□

Poznámka: Další Gaussův kvadraturní vzorec (pro $m = 1$) vypadá takto:

$$K(f) = \int_{-1}^{1} f(x)\,\mathrm{d}x = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) + \underbrace{\frac{1}{135} f^{(IV)}(\xi)}_{\text{chyba}}.$$

Geometricky si jej lze představit takto:



Poznámka: Koeficienty a uzly vzorců vyšších řádů jsou uvedeny v tabulkách. Opět lze používat složené vzorce.

Poznámka: To, že jsme vyjádřili $\int_{-1}^{1} f(x)\,dx$ neubírá nic na obecnosti, můžeme totiž libovolný interval $\langle a, b\rangle$ transformovat na $\langle -1, 1\rangle$ a použít odvozené vztahy.

Příklady k zamyšlení z numerické matematiky

1. Uveďte příklad nelineární rovnice a intervalu $\langle a, b\rangle$, na kterém rovnici řešíte, tak, aby

   (a) *metoda bisekce* nalezla řešení této rovnice na $\langle a, b\rangle$ pro zastavovací podmínku s $\varepsilon = 0.01$ po více než 10-ti krocích, zatímco *metoda regula falsi* nejvýše do 5-ti kroků.

   (b) *metoda regula falsi* nalezla řešení této rovnice na $\langle a, b\rangle$ pro zastavovací podmínku s $\varepsilon = 0.01$ po více než 10-ti krocích, zatímco *metoda bisekce* nejvýše do 5-ti kroků.

2. Uveďte příklad nelineární rovnice, intervalu $\langle a, b\rangle$, na kterém rovnici řešíte, počáteční iterace x_0 a dvou různých předpisů $x = \varphi(x)$ tak, aby metoda prosté iterace pro první předpis nekonvergovala k přesnému řešení, zatímco pro druhý předpis ano.

3. Uveďte příklad nelineární rovnice, intervalu $\langle a, b\rangle$, na kterém rovnici řešíte, počáteční iterace x_0 a předpisu $x = \varphi(x)$ tak, aby *metoda prosté iterace* sice konvergovala k přesnému řešení, ale velmi pomalu, tj. aby např. pro zastavovací podmínku s $\varepsilon = 0.01$ nalezla přibližné řešení, které se od přesného řešení liší více než o $0.1\,(= 10\varepsilon)$.

4. Uveďte příklad nelineární rovnice $f(x) = 0$ a počáteční aproximace x_0 tak, aby přibližné řešení této rovnice nalezené *Newtonovou metodou* při použití zastavovací podmínky ve tvaru $|f(x_k)| < \varepsilon$ bylo zatíženo chybou nejméně 10ε a
   aby přibližné řešení této rovnice nalezené *Newtonovou metodou* při použití zastavovací podmínky ve tvaru $|x_k - x_{k-1}| < \varepsilon$ bylo zatíženo chybou nejvýše $\varepsilon/10$.

5. Uveďte příklad nelineární rovnice $f(x) = 0$ a počáteční aproximace x_0 tak, aby přibližné řešení této rovnice nalezené *Newtonovou metodou* při použití zastavovací podmínky ve tvaru $|f(x_k)| < \varepsilon$ bylo zatíženo chybou nejvýše $\varepsilon/10$ a
   aby přibližné řešení této rovnice nalezené *Newtonovou metodou* při použití zastavovací podmínky ve tvaru $|x_k - x_{k-1}| < \varepsilon$ bylo zatíženo chybou nejméně 10ε.

6. Pomocí *metody prosté iterace* najděte všechna řešení soustavy nelineárních rovnic

$$\begin{aligned} x^2 + 4y^2 - 4 &= 0 \\ x^2 - y^2 - 4x + 2y - 3 &= 0 \end{aligned}$$

7. Uveďte příklad soustavy 4 lineárních algebraických rovnic pro 4 neznámé $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ tak, aby tato soustava měla jediné řešení a aby algoritmus Gaussovy eliminační metody nebyl pro tuto soustavu realizovatelný. Řešení určete pomocí Gaussovy eliminační metody se sloupcovou pivotací.

8. Najděte matici $\mathbf{A}$ řádu 4 a vektor $\mathbf{b}$ typu 4|1 tak, aby při řešení soustavy rovnic

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

Jacobiova metoda konvergovala, zatímco *Gauss-Seidelova metoda* divergovala.

9. Najděte matici $\mathbf{A}$ řádu 4 a vektor $\mathbf{b}$ typu 4|1 tak, aby při řešení soustavy rovnic

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

Jacobiova metoda divergovala, zatímco *Gauss-Seidelova metoda* konvergovala.

10. Najděte matici $\mathbf{A}$ řádu 2, vektor $\mathbf{b}$ typu 2|1 a dvě různé počáteční volby $\mathbf{x}_I^{(0)}$ a $\mathbf{x}_{II}^{(0)}$ tak, aby při řešení soustavy rovnic

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

metoda největšího spádu při počáteční volbě $\mathbf{x}_I^{(0)}$ dosáhla přesného řešení během 1. iterace zatímco při počáteční volbě $\mathbf{x}_{II}^{(0)}$ byla norma chyby 10-té iterace větší než 0.1, tj. $\|\mathbf{x}_{II}^{(10)} - \tilde{\mathbf{x}}\| > 0.1$, kde $\tilde{\mathbf{x}}$ je přesné řešení.

11. Uveďte příklad čtvercové regulární matice, pro kterou nelze určit vlastní čísla pomocí *LR-transformace*.

12. Uveďte příklad čtvercové regulární matice, pro kterou nelze určit vlastní čísla pomocí *QR-transformace*.

13. Uveďte příklad čtvercové matice $\mathbf{A}$, pro kterou *mocninná metoda* s počáteční aproximací $\mathbf{y}^{(0)} = [1, 1, \ldots, 1]^T$ nevypočte dominantní vlastní číslo matice $\mathbf{A}$. Dominantní vlastní číslo matice vypočtěte pomocí jiné volby počáteční aproximace $\mathbf{y}^{(0)}$.

14. Uveďte příklad symetrické čtvercové matice $\mathbf{A}$, pro kterou *metoda Rayleighova podílu* s počáteční aproximací $\mathbf{y}^{(0)} = [1, 0, 0 \ldots, 0]^T$ nevypočte dominantní vlastní číslo matice $\mathbf{A}$. Dominantní vlastní číslo matice vypočtěte pomocí jiné volby počáteční aproximace $\mathbf{y}^{(0)}$.

15. Uveďte příklad funkce $f(x)$, bodu x_0 a okolí tohoto bodu $\mathcal{U}(x_0)$ tak, aby *Taylorův polynom* 5.stupně $T_5(x)$ byl přesně roven funkci f na okolí $\mathcal{U}(x_0)$, zatímco *Taylorův polynom* 4.stupně $T_4(x)$ byl různý od $f(x)$ pro $\forall x \in \mathcal{U}(x_0) - x_0$.

16. Uveďte příklad funkce f zadané tabulkou o 5-ti bodech tak, aby *interpolační polynom* byl třetího stupně a po vynechání libovolného jednoho bodu z tabulky se interpolační polynom nezměnil.

17. Uveďte příklad funkce f zadané tabulkou o 5-ti bodech a bodu α, ve kterém chceme určit přibližnou hodnotu funkce f pomocí *Nevilleova algoritmu* tak, aby se výpočet mohl ukončit dříve než po 5-ti krocích se zadanou tolerancí $\varepsilon = 0,001$ pro rozdíl dvou po sobě jdoucích aproximací $f(\alpha)$.

18. Uveďte příklad funkce f zadané tabulkou o 4-ti bodech, pro kterou *kubický spline* s přirozenými podmínkami totožný s interpolačním polynomem.

19. Uveďte příklad funkce f zadané tabulkou o 5-ti bodech, pro kterou se *diskrétní L_2-aproximace* lineární funkcí nezmění při vypuštění jednoho konkrétního bodu z tabulky, zatímco při vypuštění libovolného jiného bodu se změní.

20. Uveďte příklad funkce f zadané tabulkou o 4-ti bodech, pro kterou je *diskrétní L_2-aproximace* kvadratickou funkcí pouze lineární funkce.

21. Uveďte příklad funkce f zadané tabulkou o 5-ti bodech, pro kterou dá *Fourierova analýza* s užitím prvních čtyř bázových trigonometrických polynomů (ve smyslu diskrétní L_2-aproximace) výsledek, u kterého jsou současně splněny interpolační podmínky.

22. Odvoďte vzorec *centrální poměrné diference* $D_C f(x_0, h)$ pro přibližný výpočet derivace funkce f v bodě x_0 pomocí interpolačního polynomu. (Interpolační polynom je dán hodnotami v uzlech $x_0 - h$, x_0 a $x_0 + h$.)

23. Uveďte příklad hladké funkce f, bodu x_0 a kroku h tak, aby přibližná hodnota derivace $f'(x_0)$ vypočtená pomocí *centrální poměrné diference* byla horší než přibližná hodnota derivace vypočtená pomocí *pravé poměrné diference*. Jaký výsledek dostaneme použitím *levé poměrné diference* (a proč) ?

24. Odvoďte základní vzorec *Sipsonova pravidla* pro přibližný výpočet integrálu funkce f přes interval $\langle x_k, x_{k+2} \rangle$.

25. Uveďte příklad funkce f a intervalu $\langle a, b \rangle$ tak, aby přibližná hodnota integrálu

$$\int_a^b f(x)\,dx$$

vypočtená pomocí *obdélníkového pravidla* byla lepší než přibližná hodnota integrálu vypočtená pomocí *lichoběžníkového* a dokonce i *Simpsonova pravidla*. Pro jednoduchost použijte základní vzorce.

26. Uveďte příklad funkce f a intervalu $\langle a, b \rangle$ tak, aby přibližná hodnota integrálu

$$\int_a^b f(x)\,dx$$

vypočtená pomocí *Gaussova kvadraturního vzorce se dvěma uzly* byla lepší než přibližná hodnota integrálu vypočtená pomocí *Gaussova kvadraturního vzorce se třemi uzly*. Pro jednoduchost použijte základní vzorce.

27. Uveďte příklad počáteční úlohy pro obyčejnou diferenciální rovnici 1. řádu (s nenulovým řešením), pro kterou bude *Eulerova metoda* totožná s *metodou Taylorova typu 2. řádu*.

28. Uveďte příklad počáteční úlohy pro obyčejnou diferenciální rovnici 1. řádu (s nenulovým řešením), pro kterou bude *metoda Taylorova typu 2. řádu* totožná s *metodou Taylorova typu 3. řádu*, ale různá od *Eulerovy metody*.

29. Uveďte příklad počáteční úlohy pro obyčejnou diferenciální rovnici 1. řádu (s nenulovým řešením), pro kterou bude *modifikovaná Eulerova metoda* totožná s *Heunovou metodou*, ale různá od *Eulerovy metody*.

Jméno a příjmení:..... **Osobní číslo:** .....

Příklad 1. Na intervalu $\langle 0; 3{,}5\rangle$ určete všechna řešení nelineární rovnice

$$e^{\sin(x)} + \sin(e^x) = 0.$$

Uveďte jakou numerickou metodu a zastavovací podmínku jste použili. Pro jedno z řešení vypište posloupnost přibližných řešení, které jste získali. Vykreslete obrázek s grafem funkce na daném intervalu.

[5 b.]

Příklad 2. Použijte mocninnou metodu pro určení dominantního vlastního čísla matice

$$\mathbf{A} = \begin{bmatrix} 40 & 10 & 20 & 5 \\ 40 & 30 & -20 & 10 \\ 20 & 10 & -30 & 20 \\ -50 & 20 & 40 & 20 \end{bmatrix}.$$

Jaký jste použili počáteční vektor? Uveďte, kolik iterací musíte provést pro Vaši volbu počátečního vektoru, pokud chcete výpočet ukončit podmínkou $|\lambda_1^{(k+1)} - \lambda_1^{(k)}| < 0{,}01$. Vypište poslední 4 iterace a ověřte, zda výsledek odpovídá dominantnímu vlastnímu číslu.

[5 b.]

Příklad 3. Je dána funkce $f(x) = (x^2 + 2x + 3) \cdot \ln x$.

a) Určete interpolační polynom procházející body $[x_i, f(x_i)]$, kde $x_i = 1, 2, \ldots, 8$.

b) Určete polynom stupně nejvýše 3, který je nejlepší L_2-aproximací funkce f zadané v bodech $x_i = 1, 2, \ldots, 8$.

c) Určete polynom stupně nejvýše 4, který je nejlepší spojitou L_2-aproximací funkce f zadané na intervalu $\langle 1; 8\rangle$.

[5 b.]

Příklad 4. Pomocí opakované Richardsonovy extrapolace aplikované na vzorec centrální poměrné diference určete přibližnou hodnotu derivace funkce

$$f(x) = \ln(-x^3 + 3x - 1) \quad \text{v bodu } x_0 = 1 \quad \text{s kroky } h = \frac{1}{2}, \frac{1}{4} \text{ a } \frac{1}{8}.$$

Jaký je výsledek po 2. korekci? Jaká je chyba přibližného výsledku?

[5 b.]

Příklad 5. Pomocí složených a) Simpsonova a b) 3-bodového Gaussova kvadraturního vzorce určete přibližnou hodnotu určitého integrálu

$$\int_{0.2}^{2} \sin\frac{\pi}{x}\,\mathrm{d}x\,.$$

Interval, přes který se integruje, rozdělte na 8 podintervalů a vypište chyby obou získaných výsledků. Nakreslete graf integrované funkce na intervalu, přes který se integruje.

[5 b.]

Informace ke zpracování testu:

1. Veškerý postup a výsledky uložte do souboru pojmenovaném podle Vašeho příjmení.
2. Soubor převeďte do formátu PDF a odešlete v daném časovém limitu na adresu **danek@kma.zcu.cz**.

Začínáme (`help general`)

`help`	nápověda, pro konkrétní fci `help sin`
`doc`	dokumentace, pro konkrétní fci `doc sin`
`lookfor`	vyhledání funkce (`lookfor cosine`)
`clc`	vymazání Command Window

Proměnné

MATLAB rozlišuje velikost znaků. Název proměnné musí začínat písmenem. Základní proměnnou je dvourozměrné pole reálných čísel (64 bitů).

Skalár je pole 1×1.
Řádkový vektor délky n je pole $1 \times n$.
Sloupcový vektor délky m je pole $m \times 1$.
Matice s m řádky a n sloupci je pole $m \times n$.
Řetězec obsahuje text. (`help strfun`)

`clear a`	smazání proměnné `a`
`clear`	smazání všech proměnných

Předdefinované konstanty

`ans`	výsledek předchozího výpočtu
`eps`	tzv. strojová přesnost
`Inf`	nekonečno, např. `1/0 = Inf`
`NaN`	nedefinovaný výraz, např. `0/0 = NaN`
`pi`	hodnota π (3.1415...)
`1i`, `1j`	komplexní jednotka

Vytvoření matice/vektoru

Prvky na řádku jsou odděleny mezerou nebo čárkou, středník ukončuje řádek.

`v = [1 2 3 4 5]`	řádkový vektor
`w = [5; 2; 3; 4]`	sloupcový vektor
`A = [1,2,3; 4,5,6; 7,8,9]`	matice

Nebo též takto

```
A = [1,2,3
   4,5,6
   7,8,9]
```

`length`	délka vektoru nebo největší rozměr pole
`size`	rozměry pole, `[m,n] = size(A)`

Vytvoření speciálních matic/vektorů
(`help elmat`)

`linspace(a,b,N)`	pravidelné dělení intervalu $[a, b]$ obsahující N bodů (včetně a, b)
`i:j:k`	aritmetická posloupnost, první prvek i, diference j, horní mez k (resp. dolní mez pro záporné j)
`i:k`	aritmetická posloupnost s diferencí 1
`zeros(m,n)`	pole nul $m \times n$
`zeros(n)`	pole nul $n \times n$
`ones(m,n)`	pole jedniček $m \times n$
`ones(n)`	pole jedniček $n \times n$
`eye(m,n)`	pole nul $m \times n$ s jedničkami na diagonále
`eye(n)`	jednotková matice $n \times n$
`rand(m,n)`	pole $m \times n$ náhodných čísel
`rand(n)`	pole $n \times n$ náhodných čísel

Indexování proměnných

Číslování začíná od jedničky!

`v(1)`	první prvek vektoru v
`v(end)`	poslední prvek vektoru v
`v(2:3:9)`	druhý, pátý a osmý prvek vektoru v
`A(2,3)`	prvek matice A ve druhém řádku, třetím sloupci
`A(:,3)`	třetí sloupec matice A
`A(1,:)`	první řádek matice A
`A(1:2:end,:)`	matice obsahující liché řádky matice A
`A(1:2,2:4)`	podmatice A obsahující 1.– 2. řádek, 2.– 4. sloupec
`A(1,end)`	poslední prvek prvního řádku matice A

Operace s maticemi/vektory (`help arith`, `help ops`)

`+`	sčítání		
`-`	odčítání		
`*`	násobení	`.*`	násobení po složkách
`/`	dělení	`./`	dělení po složkách
`^`	mocnění	`.^`	mocnění jednotlivých složek
`'`	komplexní sdružení	`.'`	transpozice

Matematické funkce (`help elfun, help matfun`)

Následující funkce mají očekávaný význam:
`abs`, `exp`, `log`, `log10`, `log2`, `sqrt`, `sin`, `asin`, `cos`, `acos`, `tan`, `atan`, `floor`, `ceil`, `round`, `max`, `min`, `norm`, `rank`, `det`, `inv`, `sort`, `sum`.

Vstupem těchto funkcí mohou být i vektory a matice.

Skripty ("m-files")

Posloupnost příkazů, tzv. skript je uložen v textovém souboru s příponou `.m`. Skript lze spustit zadáním jména souboru (bez přípony) v příkazovém řádku, nebo tlačítkem "Run" v okně editoru.

`;`	potlačení výstupu, umísťuje se na konec příkazu
`%`	začátek komentáře (celý zbytek řádku)
`...`	pokračování příkazu/výrazu na dalším řádku

Funkce

Funkce jsou definovány v jednotlivých souborech pojmenovaných stejně jako funkce `NazevFunkce.m` následovně:

```
function [out1,...,outN] = NazevFunkce(in1,...inM)
% NazevFunkce: Stručný popis (volitelně)
% ...
příkazy;
```

Funkci poté spustíme takto

```
[výstup1,...,výstupN] = NazevFunkce(vstup1,...vstupM)
```

Logické operátory (`help relop`)

`<`	menší než
`<=`	menší nebo rovný
`>`	větší než
`>=`	větší nebo rovný
`==`	rovný
`~=`	různý
`&`	logické AND ("a zároveň")
`|`	logické OR ("nebo")
`~`	negace

Cykly (`help lang`)

```
for k = Vektor
    příkazy;
end
```

Obvykle: `for k = 1:n`

```
while LogickýVýraz
    příkazy;
end
```

Pro náročnější výpočty je žádoucí se cyklům vyhýbat.

Větvení kódu (help lang)

```
if LogickýVýraz
    příkazy;
elseif LogickýVýraz    % Volitelně
    příkazy;
else                   % Volitelně
    příkazy;
end
```

Vykreslování výsledků (help graph2d, help graph3d)

`figure`	vytvoření nového okna pro graf
`close`	zavření aktuálního okna
`close all`	zavření všech oken
`plot`	základní 2D graf
`semilogy`	2D graf s logaritmicky škálovanou osou y
`imagesc`	zobrazení obrázku, vhodné pro zobrazení prvků matice
`plot3`	zobrazení křivky/bodů ve 3D
`surf`	zobrazení plochy ve 3D
`legend`	legenda ke grafu
`title`	název grafu
`xlabel`	popisek osy x
`ylabel`	popisek osy y