



# NUMERICKÉ METODY I

## Vít Vondrák, Lukáš Pospíšil

Text byl vytvořen v rámci realizace projektu *Matematika pro inženýry 21. století* (reg. č. CZ.1.07/2.2.00/07.0332), na kterém se společně podílela Vysoká škola báňská – Technická univerzita Ostrava a Západočeská univerzita v Plzni


INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vít Vondrák, Lukáš Pospíšil
Numerické metody 1


ISBN

# Předmluva

Vážení čtenáři,

dostává se Vám do rukou učební text *Numerické metody 1*. Základ tohoto textu vznikl již před několika lety pro podporu výuky předmětu Numerické metody na Fakultě elektrotechniky a informatiky Vysoké školy báňské-Technické Univerzity Ostrava. Předmět byl zejména určen pro obor Výpočetní matematika. Tento předmět si stejně jako i učební text klade za cíl seznámit studenty se základními numerickými metodami, které se běžně používají v inženýrské praxi. A nejen tam. Se základními numerickými metodami se totiž můžete setkat i ve fyzice, chemii, elektrotechnice, informatice, environmentálních vědách a vůbec všude, kde je zapotřebí řešit základní matematické úlohy, jejichž analytické řešení buď není známo, nebo je jen velmi těžko řešitelné.

Celkový obsah učebního textu je možné rozdělit do tří základních oblastí. Úvod učebního textu je věnován problematice chyb v inženýrských výpočtech, jejich zdrojům a analýze. Druhou oblastí je oblast numerických metod pro řešení základních úloh lineární algebry. Zde se seznámíte s metodami řešení soustav lineárních rovnic, nebo s problémem hledání vlastních čísel. Do této oblasti byla taktéž zařazena problematika řešení nelineárních rovnic a jejich soustav, byť by mohla být zařazena i do oblasti matematické analýzy. Třetí oblastí je oblast numerických metod používaných pro řešení úloh vyskytujících se v matematické analýze. Mezi tyto úlohy se řadí interpolace a aproximace funkcí, numerická derivace a integrování či řešení počátečních úloh pro obyčejné diferenciální rovnice.

Pro pochopení učebního textu je proto potřeba základních znalostí z výše uvedené problematiky. Na Fakultě elektrotechniky a informatiky jsou tyto prerekvizity zajištěny následujícími předměty: Lineární algebra, Matematická analýza 1 a 2. Důležitá je taktéž znalost základů algoritmizace a programování, neboť popis jednotlivých numerických metod je orientován na jejich praktické použití na počítači.

Odvození a popis metod je však doplněn i konvergenční teorií a odhady chyb. Tyto poznatky jsou totiž nezbytné pro správnou volbu numerické metody při řešení konkrétních úloh.

V Ostravě 1. 9. 2011 Vít Vondrák a Lukáš Pospíšil

# Obsah

# Kapitola 1

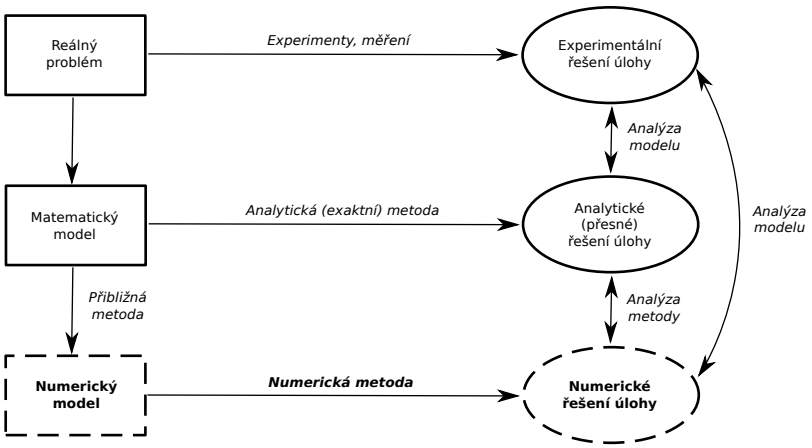
# Úvod do numerických metod

## Průvodce studiem


*V této kapitole si představíme základní principy při řešení inženýrských úloh. Důraz bude kladen na jejich numerické řešení, které připouští jistou míru nepřesnosti. Následovat proto bude obecná teorie chyb. Budou zavedeny základní pojmy jako jsou stabilita, podmíněnost matematických a numerických úloh, diskutovány budou typy a zdroje chyb. Speciální pozornost bude věnována šíření chyb v numerických výpočtech. Na závěr bude představena reprezentace reálných čísel na počítačích.*

## 1.1 Matematické řešení inženýrských problémů

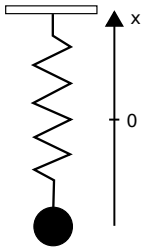
Původně experimentální řešení je s masivním nástupem počítačů nahrazováno numerickými simulacemi. Na Obrázku 1.1 je zobrazeno standardní schéma řešení inženýrských úloh.




Obr. 1.1: Postup při řešení inženýrské úlohy.

Na následujícím vzorovém příkladě budeme demonstrovat postup řešení modelové inženýrské úlohy. Za tento modelový příklad jsme vybrali úlohu kmitání tělesa na pružině.

**Příklad 1.1.** Úloha kmitání tělesa na pružině

| | |
|---|---|
| Reálný problém | Hmotné těleso je zavěšeno na pružině. Uvolnění tělesa v nerovnovážném stavu způsobí jeho kmitání ve svislém směru. Cílem je zjistit polohu tělesa v libovolně určeném časovém okamžiku. Pro zjišťování této polohy si zvolíme souřadný systém ve svislém směru jak je zobrazeno na obrázku. Počátek souřadného systému $x = 0$ reprezentuje rovnovážný stav.   |
| Experimentální řešení problému | Sestrojení fyzikálního modelu a provedení experimentu. Měření je možné provádět například pomocí kamerového systému a pravítka. |
| Matematický model | Pomocí matematického aparátu je možné při zanedbání veličin, které nemají podstatný vliv na kmítání tělesa (odpor vzduchu, hmotnost pružiny atd) popsat úlohu pomocí lineární diferenciální rovnice druhého řádu s neznámou $x$ popisující polohu tělesa v čase $t$. V této rovnici veličina $m$ popisuje hmotnost tělesa na pružině, $b$ reprezentuje buzení a $k$ představuje tuhost pružiny. Pro jednoznačnost řešení úlohy musíme zvolit počáteční polohu $x_0$ a počáteční rychlost $x_1$. $$mx''(t) = -bx'(t) - kx(t)$$ $$x(t_0) = x_0,\ x'(t_0) = x_1$$ $m = 1\text{kg}$, $b = 1\text{N}$, $k = 2\text{N.m}^{-1}$, $t_0 = 0\text{s}$, $x_0 = -5\text{m}$, $x_1 = 0\text{m.s}^{-1}$. |
| Analytické (přesné) řešení úlohy | Čtenáři, kteří jsou již seznámeni s řešením lineárních diferenciálních úloh, snadno odvodí, že funkce |

$$x(t) = -5\mathrm{e}^{-t} - 5t\mathrm{e}^{-t}$$

je řešením výše uvedeného matematického modelu. Ti, kteří se s problematikou řešení diferenciálních rovnic ještě neseznámili, mohou tak učinit ve skriptech Obyčejné diferenciální rovnice [3], případně stačí když do diferenciální rovnice funkci $x(t)$ dosadí a ověří tak její správnost.

**Přibližná metoda a numerický model**

Pro sestavení numerického modelu, který budeme chtít řešit na kalkulačce, počítači či logaritmickém pravítku budeme muset nejdříve vyřešit problém spojité závislosti polohy na čase. Ani jedno ze zmíněných zařízení nám totiž neumožní počítat s nekonečně mnoha údaji, které popisují polohy tělesa v libovolném časovém intervalu. Z tohoto důvodu se proto omezíme jen na několik vybraných časových okamžiků v nichž budeme hledat řešení dané úlohy. Této technice se říká *metoda časové diskretizace*. Pro naši modelovou úlohu si tedy zvolme 6 časových okamžiků
$t_0 = 0\mathrm{s}$, $t_1 = 0.2\mathrm{s}$, $t_2 = 0.4\mathrm{s}$, $t_3 = 0.6\mathrm{s}$, $t_4 = 0.8\mathrm{s}$, $t_5 = 1\mathrm{s}$.

**Numerická metoda**

Snad nejznámnější a nejjednodušší numerická metoda pro řešení diferenciálních rovnic se nazývá Eulerova metoda. S odvozením této metody se blíže seznámíte v kapitole 8.1.1. Pokud si přibližné řešení $x(t_n)$ v čase $t_n$ označíme $x_n$, pak řešení v následujícím časovém okamžiku $t_{n+1}$ obdržíme ze vzorce

$$x_{n+1} = x_n + h.f(t_n, x_n) \ ,$$

kde $f : \mathbb{R}^2 \to \mathbb{R}$ je funkce popisující diferenciální rovnici a $h$ je diskretizační krok.

**Analýza metody**

V následující tabulce můžeme srovnat přesnost numerického řešení s řešením analytickým.

| Čas $t$[s] | Numerické řešení $x$[m] | Analytické řešení $x$[m] |
|---|---|---|
| 0 | -5 | -5 |
| 0.2 | -5 | -4.9124 |
| 0.4 | -4.8 | -4.6992 |
| 0.6 | -4.48 | -4.3905 |
| 0.8 | -4.036 | -4.0440 |
| 1 | -3.6864 | -3.6788 |

## 1.2 Korektnost úloh a stabilita výpočetních postupů

Při studiu numerického řešení matematických modelů se budeme zabývat pouze takovými úlohami, které jsou jednoznačně řešitelné a jejichž řešení spojitě závisí na vstupních datech. Takové úlohy budeme nazývat *korektní* a jejich přesnou definici bychom mohli formulovat následovně:

**Definice 1.2.** Buď dána množina vstupních dat $B_1$ a množina výstupních dat $B_2$. Úloha $y = U(x), x \in B_1, y \in B_2$ se nazývá *korektní* na dvojici $(B_1, B_2)$ jestliže

1. ke každému $x \in B_1$ existuje jediné řešení $y \in B_2$
2. řešení spojitě závisí na vstupních datech,
   tzn. $x_n \to x, U(x_n) = y_n \Rightarrow U(x_n) \to U(x) = y$

Při řešení praktických úloh se bohužel velmi často stává, že i korektní úloha vykazuje při zanedbatelné změně vstupních dat obrovské chyby ve výsledcích. Takové úlohy nazýváme *špatně podmíněné* a jsou velmi problematicky řešitelné pokud jsou jejich vstupní data zatíženy chybami jako jsou chyby měření, měřících zařízení, zaokrouhlovacími chybami apod. Níže uvedená definice nám vymezuje pojmy *dobře a špatně podmíněné* úlohy.

**Definice 1.3.** Budeme říkat, že korektní úloha je *dobře podmíněná* jestliže malá změna ve vstupních datech vyvolá malou změnu řešení.
Je-li $x + \triangle x$ resp. $x$ změna řešení resp. řešení a $y + \triangle y$ resp. $y$ změna vstupních dat resp. vstupní data, pak číslo

$$C_p = \frac{\frac{\|\triangle x\|}{\|x\|}}{\frac{\|\triangle y\|}{\|y\|}} = \frac{\text{relativní chyba na výstupu}}{\text{relativní chyba na vstupu}}$$

nazýváme *číslem podmíněnosti úlohy* $y = U(x)$.

**Příklad 1.4.** Následující příklad demonstuje případ špatně podmíněné úlohy.

*Řešení.* Uvažujme soustavu lineárních rovnic

$$\begin{aligned} x_1 &+ 0.99x_2 = 1.99 \\ 0.99x_1 &+ 0.98x_2 = 1.97 \end{aligned}$$

Snadno nalezneme řešení této *původní* soustavy $x = (x_1, x_2) = (1, 1)$.

Dále uvažujme stejnou soustavu s mírně pozměněnou pravou stranou

$$\begin{array}{rcrcl} \tilde{x}_1 & + & 0.99\tilde{x}_2 & = & 1.989903 \\ 0.99\tilde{x}_1 & + & 0.98\tilde{x}_2 & = & 1.970106 \end{array}$$

Řešení této *pozměněné* soustavy je $\tilde{x} = (\tilde{x}_1, \tilde{x}_2) = (3, -1.021)$.

Velikost změny řešení soustavy při změně pravé strany spočítáme snadno

$$\|\triangle x\| = \|\tilde{x} - x\| = \|(2, -2.021)\| \doteq 2.843315 \ .$$

Pod změnou vstupních dat budeme v tomto případě rozumět velikost změny pravé strany (původní pravé strany $b$ a změněné pravé strany $\tilde{b}$)

$$\|\triangle b\| = \|\tilde{b} - b\| = \|(1.989903, 1.970106) - (1.99, 1.97)\| \doteq 1.436836.10^{-4} \ .$$

Pak číslo podmíněnosti úlohy spočítame dle Definice 1.3 dosazením

$$C_p = \frac{\frac{\|\triangle\triangle x\|}{\|x\|}}{\frac{\|\triangle b\|}{\|b\|}} = \frac{\frac{2.843315}{1.414213}}{\frac{1.436836.10^{-4}}{2.800178}} \doteq 39182 \ .$$

▲

**Poznámka 1.5.** Za dobře podmíněné úlohy považujeme úlohy s číslem podmíněnosti $1 < C_p < 100$. Pokud je $C_p > 100$ považujeme úlohy za hůře podmíněné a $C_p >> 10000$ považujeme za špatně podmíněné.

Katastrofické důsledky na výslednou přesnost numerického řešení může mít i nevhodně zvolená numerická metoda. Šíření numerických chyb při takto nevhodně zvolené metodě může vést ke zcela chybnému výsledku. Následující příklad ukazuje případ takovéto metody.

**Příklad 1.6.** Pro libovolné $n \in \mathbb{N}$ určete hodnotu integrálu

$$J_n = \frac{1}{\mathrm{e}} \int_0^1 x^n \mathrm{e}^x dx \ .$$

*Řešení.* Při použití základních znalostí z matematické analýzy bychom mohli udělat následující odhady

$$|J_n| \leqq \frac{1}{n+1}, \quad J_n \geqq 0 \ .$$

Zřejmě můžeme snadno analyticky spočítat hodnotu integrálu pro volbu $n = 1$

$$J_0 = 1 - \mathrm{e}^{-1} \ .$$

S využitím pravidla „per-partes“ můžeme získat následující předpis pro výpočet integrálu

$$J_n = \frac{1}{\mathrm{e}}([x^n \mathrm{e}^x]_0^1 - n \int_0^1 x^{n-1}\mathrm{e}^x dx) = 1 - nJ_{n-1} \ .$$

Pokud použijeme tento rekurzivní předpis pro výpočet, získáme hodnoty uvedené v následující tabulce.

| V přesné aritmetice: | Na 2 platné cifry: |
|---|---|
| $J_0 = 1 - \frac{1}{\mathrm{e}} \doteq 0.6321$ | $J_0 = 0.63$ |
| $J_1 = \frac{1}{\mathrm{e}} \doteq 0.3679$ | $J_1 = 1 - 1 \cdot 0.63 \doteq 0.37$ |
| $J_2 = \frac{\mathrm{e}-2}{\mathrm{e}} \doteq 0.2642$ | $J_2 = 1 - 2 \cdot 0.37 \doteq 0.26$ |
| ... | |
| $J_6 = \frac{265\mathrm{e}-720}{\mathrm{e}} \doteq 0.1268$ | $J_6 = -1.4$ |
| ... | |
| $J_{10} = \frac{24-9\mathrm{e}}{\mathrm{e}} \doteq 0.0835$ | $J_{10} = -7800$ |
| ... | |
| $J_{20} \doteq 0$ | $J_{20} = -5 \ .10^{15}$ |

Při výpočtu na 2 platné cifry tedy zaokrouhlovací chyby způsobily, že již $J_6$ je záporné (rozpor s odhadem $J_n \geqq 0$) a $J_{20}$ je výrazně mimo rozsah $|J_n| \leqq 1/(n+1)$.

Očividně tato numerická metoda není *stabilní* (viz následující definice). ▲

**Definice 1.7.** Numerická metoda je *numericky stabilní*, jestliže malé odchylky v průběhu výpočtu (zaokrouhlení) nevedou k velkým změnám v řešení.

**Příklad 1.8.** Uvažujme stejnou úlohu jako v předchozím příkladě 1.6. Pokusme se nalézt předpis stabilnější metody.

*Řešení.* Upravme předpis pro rekurzivní výpočet integrálu

$$\begin{aligned} J_n &= 1 - nJ_{n-1} \\ nJ_{n-1} &= 1 - J_n \\ J_{n-1} &= \tfrac{1}{n}(1 - J_n) \ . \end{aligned}$$

Pro užití této formule odhadneme

$$J_{20} \doteq 0 \leqq \frac{1}{1+20} \ .$$

Pak užitím nové formule při výpočtu na dvě platné cifry získáme tyto odhady.

$$J_{20} = 0, J_{10} = 0.084, J_6 = 0.13, J_2 = 0.26, J_1 = 0.37, J_0 = 0.63 \ .$$

▲

**Poznámka 1.9.** Numerická stabilita se posuzuje podobně jako podmíněnost úlohy pomocí čisla podmíněnosti algoritmu (stejná definice jako číslo podmíněnosti úlohy - viz Definice 1.3). Určuje se pomocí tzv. experimentálních perturbací, tzn. že mírně pozměňujeme vstupní data a sledujeme výsledky na výstupu.
Zatímco špatná podmíněnost úlohy je dána úlohou samou a v podstatě ji nemůžeme ovlivnit, stabilitu výpočetního postupu ovlivňujeme sami volbou metody.

## 1.3 Typy chyb

Při řešení reálných úloh se setkáváme s celou řadou možných chyb, které ovlivňují přesnost výsledného řešení. Podle zdroje chyb bychom mohli chyby rozdělit do několika následujících skupin:

a) **Chyba matematického modelu**
Zjednodušováním původní úlohy jakož i zanedbáváním veličin, které nemají velký vliv na matematický model vznikají nepřesnosti v řešení matematického modelu. Tuto chybu lze kvantifikovat rozdílem mezi řešením reálného problému a řešením matematického modelu. Tato chyba vypovídá o přesnosti matematického modelu.
Př. Zanedbání odporu vzduch v Příkladě 1.1 vede k nepřesnosti v řešení matematického modelu.

b) **Chyba numerické metody**
Použití nepřesné metody pro řešení matematického modelu vede k chybě numerického řešení.
Př. Limitu posloupnosti aproximujeme členem s dostatečně velkým indexem, $\lim\limits_{n\to\infty} a_n \approx a_{5000}$.

c) **Chyba aproximace**
Aplikací přibližné metody na matematický model vzniká chyba aproximace. Pak rozdíl mezi řešením matematické úlohy a řešením numerické úlohy je chyba aproximace.
Př. Nahrazení spojitých vstupních dat diskrétními, viz. diskretizace času v Příkladě 1.1.

d) **Chyby ve vstupních datech**
Vznikají chybami měření a reprezentací čísel v počítači.

e) **Chyby zaokrouhlovací**
Nepřesnosti způsobené realizací algoritmu v počítači a nepřesné provádění aritmetických operací.

### 1.3.1 Aproximace čísla

Při realizaci výpočtu nahrazujeme reálnou (přesnou) hodnotu $x$ jeho aproximací (přiblížením) $\tilde{x}$. Značíme $\tilde{x} \approx x$.

**Definice 1.10.** Rozdíl $\triangle x = x - \tilde{x}$ nazýváme *absolutní chybou* aproximace $\tilde{x}$. Hodnotu $\varepsilon \geqq 0$ takovou, že $|x - \tilde{x}| = |\triangle x| \leqq \varepsilon(\tilde{x})$ nazýváme *odhad absolutní chyby.*

**Definice 1.11.** Číslo $\frac{\triangle x}{x} = \frac{x-\tilde{x}}{x}, x \neq 0$ nazýváme *relativní chybou* aproximace $\tilde{x}$. Číslo $\delta(x) \geqq 0$ takové, že $|\frac{x-\tilde{x}}{x}| \leqq \frac{\epsilon(\tilde{x})}{|x|} = \delta(\tilde{x})$ nazýváme *odhad relativní chyby.*

**Poznámka 1.12.** Jelikož hodnota $x$ není obvykle známa, pro odhad relativní chyby používáme vzorec $\frac{\triangle x}{\tilde{x}}$.

**Poznámka 1.13.** Z definice plyne, že $x = \tilde{x}.(1 \pm \delta)$, kde $\delta = \frac{|\triangle x|}{|\tilde{x}|}$.

**Příklad 1.14.** Určete odhady absolutní a relativní chyby čísla $\tilde{x} = 3.141$, které nahrazuje číslo $\pi$.

*Řešení.* Platí (odhadněme)

$$3.141 < \pi < 3.142 \quad \Rightarrow \quad 0 < \pi - 3.141 < 0.001 \ .$$

Pak pro absolutní chybu lze psát

$$|\triangle x| = |\pi - \tilde{x}| = |\pi - 3.141| < 0.001 \ .$$

Tedy odhadem absolutní chyby je $\epsilon(\tilde{x}) = 0.001$.
Pro odhad relativní chyby dosaďme do vzorce z Definice 1.11

$$\delta(\tilde{x}) = \frac{\epsilon(\tilde{x})}{|\tilde{x}|} = \frac{0.001}{3.141} = 0.0003 \ ,$$

tedy lze psát $\pi = 3.141 \cdot (1 \pm 0.0003)$. ▲

### 1.3.2 Šíření chyb při aritmetických operacích

Aplikací výpočetního postupu na řešení numerických úloh provádíme celou řadu různých početních operací. Pokud do těchto operací vstupují veličiny, které jsou zatíženy nějakou aproximační chybou, zřejmě dochází k další propagaci těchto chyb do dílčích výsledků. Abychom lépe pochopily jak se tyto chyby v rámci složitějších výpočtů šíří či eliminují, je nutné si ukázat jaký vliv na jejich šíření mají základní aritmetické operace.

V dalším textu uvažujme přesnou hodnotu $x_i$ a její aprioximaci $\tilde{x}_i$. Dále $\epsilon_i$ je odhad její absolutní chyby a $\delta_i$ odhad relativní chyby. Označme pomocí $u$ přesný výsledek aritmetické operace zatímco $\tilde{u}$ výsledek operace s nepřesnými operandy. Dále označme $\epsilon$ odhad absolutní chyby $u$ a $\delta$ odhad relativní chyby $u$.

Pak dle předchozího výkladu platí

$$\begin{array}{rclcrcl} x_i & = & \tilde{x}_i + \triangle x_i \ , & & u & = & \tilde{u} + \triangle u \ , \\ |\triangle x_i| & \leqq & \epsilon_i \ , & & |\triangle u| & \leqq & \epsilon \ , \\ |\frac{\triangle x_i}{x_i}| & \leqq & \delta_i \ , & & |\frac{\triangle u}{u}| & \leqq & \delta \ . \end{array} \tag{1.1}$$

Tedy následující výklad si klade za cíl nalézt $\epsilon$ a $\delta$ pro jednotlivé elementární algebraické operace.

a) **Chyba násobku veličiny**
Uvažujme operaci násobení skalárem $k \in \mathbb{R}$, pak pro výsledek násobení přesné vstupní hodnoty, resp. její aproximace, platí

$$u = k.x_1 \ , \quad \text{resp. } \tilde{u} = k.\tilde{x}_1 \ .$$

Dle (1.1) lze psát

$$\begin{array}{rcl} u & = & k \ .x_1 = k(\tilde{x}_1 + \triangle x_1) = k.\tilde{x}_1 + k.\triangle x_1 \\ \triangle u & = & u \ - \tilde{u} = (k.\tilde{x}_1 + k.\triangle x_1) - (k.\tilde{x}_1) = k.\triangle x_1 \end{array}$$

Pak pro odhad absolutní a relativní chyby lze psát

$$\begin{array}{rcl} |\triangle u| \leqq |k|.|\triangle x_1| \leqq |k|.\epsilon_1 & = & \epsilon \ , \\ |\frac{\triangle u}{u}| \leqq \frac{|u|.|\triangle x_1|}{|u|.|x_1|} = |\frac{\triangle x_1}{x_1}| \leqq \delta_1 & = & \delta \ . \end{array}$$

b) **Chyba algebraického součtu**
Uvažujme operaci součtu dvou čísel $x_1$ a $x_2$, které jsou zatížené chybou. Pak pro výsledek součtu přesných vstupních hodnot, resp. součtu aproximací, platí

$$u = x_1 + x_2 \ , \quad \text{resp. } \tilde{u} = \tilde{x}_1 + \tilde{x}_2 \ .$$

Dle (1.1) lze psát a upravit

$$\begin{array}{rcl} u & = & x_1 + x_2 = (\tilde{x}_1 + \triangle x_1) + (\tilde{x}_2 + \triangle x_2) = \tilde{u} + \triangle x_1 + \triangle x_2 \\ \triangle u & = & \triangle x_1 + \triangle x_2 \\ \frac{\triangle u}{u} & \leqq & \frac{\triangle x_1 + \triangle x_2}{x_1 + x_2} \approx \frac{\tilde{x}_1}{\tilde{x}_1 + \tilde{x}_2}.\frac{\triangle x_1}{x_1} + \frac{\tilde{x}_2}{\tilde{x}_1 + \tilde{x}_2}.\frac{\triangle x_2}{x_2} \end{array}$$

A pro odhad absolutní a relativní chyby lze psát

$$\begin{aligned} |\triangle u| \leqq |\triangle x_1| + |\triangle x_2| \leqq \epsilon_1 + \epsilon_2 &= \epsilon \\ |\tfrac{\triangle u}{u}| \leqq \tfrac{1}{|\tilde{x}_1+\tilde{x}_2|}(|\tilde{x}_1|\delta_1 + |\tilde{x}_2|\delta_2) &= \delta \end{aligned}$$

c) **Chyba rozdílu**
Uvažujme operaci rozdílu dvou čísel $x_1$ a $x_2$, které jsou zatížené chybou. Pak pro výsledek rozdílu přesných vstupních hodnot, resp. rozdílu aproximací, platí

$$u = x_1 - x_2 \ , \quad \text{resp. } \tilde{u} = \tilde{x}_1 - \tilde{x}_2 \ .$$

Dle (1.1) lze psát a upravit

$$\begin{aligned} u &= x_1 - x_2 = (\tilde{x}_1 + \triangle x_1) - (\tilde{x}_2 + \triangle x_2) = \tilde{u} + \triangle x_1 - \triangle x_2 \\ \triangle u &= \triangle x_1 - \triangle x_2 \\ \tfrac{\triangle u}{u} &\approx \tfrac{\tilde{x}_1}{\tilde{x}_1-\tilde{x}_2}.\tfrac{\triangle x_1}{x_1} - \tfrac{\tilde{x}_2}{\tilde{x}_1-\tilde{x}_2}.\tfrac{\triangle x_2}{x_2} \end{aligned}$$

A pro odhad absolutní a relativní chyby lze psát

$$\begin{aligned} |\triangle u| \leqq |\triangle x_1| + |\triangle x_2| \leqq \epsilon_1 + \epsilon_2 &= \epsilon \\ |\tfrac{\triangle u}{u}| \leqq \tfrac{1}{|\tilde{x}_1-\tilde{x}_2|}(|\tilde{x}_1|\delta_1 + |\tilde{x}_2|\delta_2) &= \delta \end{aligned}$$

d) **Chyba součinu**
Uvažujme operaci součinu dvou čísel $x_1$ a $x_2$, které jsou zatížené chybou. Pak pro výsledek součinu přesných vstupních hodnot, resp. součinu aproximací, platí

$$u = x_1.x_2 \ , \quad \text{resp. } \tilde{u} = \tilde{x}_1.\tilde{x}_2 \ .$$

Dle (1.1) lze psát a upravit

$$\begin{aligned} u &= x_1.x_2 = (\tilde{x}_1 + \triangle x_1)(\tilde{x}_2 + \triangle x_2) \\ u &= \tilde{x}_1\tilde{x}_2 + \tilde{x}_1\triangle x_2 + \tilde{x}_2\triangle x_1 + \triangle x_1\triangle x_2 = \tilde{u} + \triangle u \\ \triangle u &\approx \tilde{x}_1\triangle x_2 + \tilde{x}_2\triangle x_1 \quad (\triangle x_1\triangle x_2 \text{ zanedbáváme neboť je blízké } 0) \\ \tfrac{\triangle u}{u} &\approx \tfrac{\tilde{x}_1\triangle x_2+\tilde{x}_2\triangle x_1}{x_1x_2} \approx \tfrac{\tilde{x}_1\triangle x_2+\tilde{x}_2\triangle x_1}{\tilde{x}_1\tilde{x}_2} = \tfrac{\triangle x_1}{\tilde{x}_1} + \tfrac{\triangle x_2}{\tilde{x}_2} \end{aligned}$$

A pro odhad absolutní a relativní chyby lze psát

$$\begin{aligned} |\triangle u| \leqq |\tilde{x}_1|\epsilon_2 + |\tilde{x}_2|\epsilon_1 &= \epsilon \\ |\tfrac{\triangle u}{u}| \leqq |\tfrac{\triangle x_1}{x_1}| + |\tfrac{\triangle x_2}{x_2}| \leqq \delta_1 + \delta_2 &= \delta \end{aligned}$$

e) **Chyba podílu**
Uvažujme operaci podílu dvou čísel $x_1$ a $x_2$, které jsou zatížené chybou. Pak pro výsledek podílu přesných vstupních hodnot, resp. podílu aproximací, platí

$$u = \frac{x_1}{x_2} \ , \quad \text{resp. } \tilde{u} = \frac{\tilde{x}_1}{\tilde{x}_2} \ .$$

Dle (1.1) lze psát a upravit

$$\begin{array}{rcl} u & = & \frac{x_1}{x_2} = \frac{\tilde{x}_1+\triangle x_1}{\tilde{x}_2+\triangle x_2} \approx \frac{(\tilde{x}_1+\triangle x_1)(\tilde{x}_2-\triangle x_2)}{(\tilde{x}_2+\triangle x_2)(\tilde{x}_2-\triangle x_2)} \\ u & \approx & \frac{\tilde{x}_1\tilde{x}_2+\tilde{x}_2\triangle x_1-\tilde{x}_1\triangle x_2-\triangle x_1\triangle x_2}{\tilde{x}_2^2-(\triangle x_2)^2} \approx \frac{\tilde{x}_1}{\tilde{x}_2}+\frac{\tilde{x}_2\triangle x_1-\tilde{x}_1\triangle x_2}{\tilde{x}_2^2} \\ \triangle u & \approx & \frac{\tilde{x}_2\triangle x_1-\tilde{x}_1\triangle x_2}{\tilde{x}_2^2} \\ \frac{\triangle u}{u} & = & \frac{\tilde{x}_2\triangle x_1-\tilde{x}_1\triangle x_2}{\tilde{x}_2^2}.\frac{x_2}{x_1} \approx \frac{x_2\triangle x_1-x_1\triangle x_2}{x_2^2}.\frac{x_2}{x_1} = \frac{\triangle x_1}{x_1}-\frac{\triangle x_2}{x_2} \end{array}$$

A pro odhad absolutní a relativní chyby lze psát

$$\begin{array}{rcl} |\triangle u| \leqq \frac{|\tilde{x}_2|\epsilon_1-|\tilde{x}_1|\epsilon_2}{|\tilde{x}_2^2|} & = & \epsilon \\ |\frac{\triangle u}{u}| \leqq |\frac{\triangle x_1}{x_1}|+|\frac{\triangle x_2}{x_2}| \leqq \delta_1+\delta_2 & = & \delta \end{array}$$

f) **Chyba mocniny**
Uvažujme operaci $n$-tá mocnina čísla $x_1$, které je zatížené chybou. Pak pro výsledek mocniny přesné vstupní hodnoty, resp. mocniny aproximace, platí

$$u = x_1^n \ , \quad \text{resp. } \tilde{u} = \tilde{x}_1^n \ .$$

Dle (1.1) lze psát a upravit

$$\begin{array}{rcll} u & = & (\tilde{x}_1+\triangle x_1)^n = \tilde{x}_1^n + \sum_{k=1}^{n} \left( \begin{array}{c} n \\ k \end{array} \right) \tilde{x}_1^{n-k}(\triangle x_1)^k & \text{(binomická věta)} \\ \triangle u & = & n.\tilde{x}_1^{n-1}\triangle x_1 & \\ \frac{\triangle u}{u} & = & \frac{n\tilde{x}_1^{n-1}\triangle x_1}{x_1^n} \approx \frac{nx_1^{n-1}\triangle x_1}{x_1^n} = n\frac{\triangle x_1}{x_1} & \end{array}$$

A pro odhad absolutní a relativní chyby lze psát

$$\begin{array}{rcl} |\triangle u| \leqq n|\tilde{x}_1|^{n-1}\epsilon_1 & = & \epsilon \\ |\frac{\triangle u}{u}| \leqq n.\delta_1 & = & \delta \end{array}$$

g) **Chyba funkční hodnoty**
Uvažujme funkci $f : \mathbb{R} \to \mathbb{R}$ do které vstupuje jako argument číslo $x_1$, které je zatížené chybou. Pak pro funkční hodnotu v přesné hodnotě, resp. v aproximaci, platí

$$u = f(x_1) \ , \quad \text{resp. } \tilde{u} = f(\tilde{x}_1) \ .$$

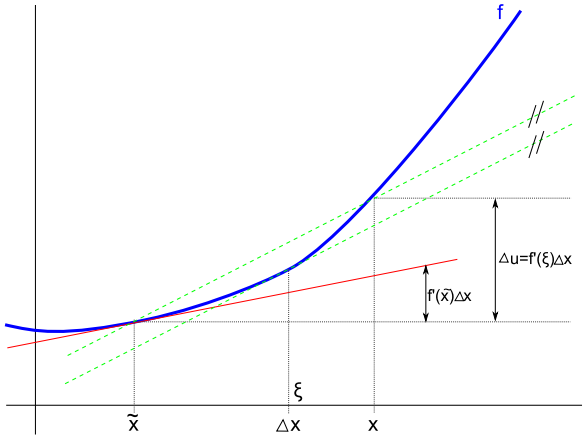
Dle (1.1) lze psát a upravit (viz Obrázek 1.3.2)

$$\begin{array}{rcl} \triangle u & = & u \ -\tilde{u} = f(x_1)-f(\tilde{x}_1) = f(\tilde{x}_1+\triangle x_1)-f(\tilde{x}_1) \\ \exists & & \\ xi \in \mathbb{R}: \quad f(x_1) & = & f(\tilde{x}_1)+f'(\xi)\triangle x_1 \\ \triangle u & = & f'(\xi)\triangle x_1 \end{array}$$

A pro odhad absolutní a relativní chyby lze psát

$$\begin{array}{rcl} |\triangle u| \leqq |f'(\xi)|.|\triangle x| \leqq M\xi & = & \epsilon \\ |\frac{\triangle u}{u} \leqq \frac{M.\epsilon_1}{|u|} \leqq \frac{|\tilde{x}_1|M\delta_1}{|f'(\tilde{x}_1)|} & = & \delta \end{array}$$

kde $M = \sup_{z\in\langle\tilde{x}_1,x_1\rangle} |f'(z)|$.




Obr. 1.2: Chyba funkční hodnoty.

**Poznámka 1.15.** Z výše uvedených odhadů je patrné, že nekritičtější operací z hlediska šíření chyb je rozdíl dvou téměř shodných veličin, kdy odhad relativní chyby díky dělení tilde u výrazně narůstá a dělení číslem blízkým nule, které má za následek enormní nárust odhadu absolutní chyby. Ve složitých výpočetních postupech je proto nutné se těchto operací vyvarovat !!!

### 1.3.3 Obecný vzorec chyb

Zobecněním chyby funkční hodnoty (viz případ *g)* v předešlé kapitole) na funkce více proměnných můžeme získat *obecný odhad chyby*. Jednotlivé aritmetické operace totiž můžeme reprezentovat jako funkce dvou proměnných (např. pro součet $f(x_1, x_2) = = x_1 + x_2$) Navíc do funkce $f$ můžeme zahrnout celou řadu operací s více veličinami $x_1, \dots x_n$.
Pro odhad absolutní chyby pak můžeme využít obdobně jako v případě funkce jedné proměnné diferenciálu funkce $f(x_1, x_2, \dots x_n)$

$$du = df(x_1, x_2, \dots x_n) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} dx_i \leqq \sum_{i=1}^{n} |\frac{\partial f}{\partial x_i} dx_i| \ .$$

Nahradíme-li diferenciál přírůstkem funkce, dostaneme pro odhad absolutní chyby

$$|\triangle u| \approx |du| \leqq \sum_{i=1}^{n} |\frac{\partial f}{\partial x_i}|.|dx_i| \leqq \sum_{i=1}^{n} |\frac{\partial f}{\partial x_i}|\epsilon_i \quad = \epsilon \ .$$

Pro získání odhadu relativní chyby lze psát

$$\begin{aligned} |\tfrac{\triangle u}{u}| \quad &\leqq \quad \sum_{i=1}^{n} \tfrac{|\frac{\partial f}{\partial x_i}|}{|u|}.|\triangle x_i| = \sum_{i=1}^{n} |\tfrac{\frac{\partial f}{\partial x_i}}{u}|.|\triangle x_i| = \sum_{i=1}^{n} |\tfrac{\partial}{\partial x_i} \inf(x_1, \dots, x_n)|.|\triangle x_i| \\ &\leqq \quad \sum_{i=1}^{n} |\tfrac{\partial ln\ u}{\partial x_i}||\triangle x_i| \approx \sum_{i=1}^{n} |\tilde{x}_i||\tfrac{\partial ln\ u}{\partial x_i}|\delta_i \quad = \delta \ . \end{aligned}$$

**Příklad 1.16.** Určete odhad absolutní a relativní chyby objemu koule $V = \frac{1}{6}\pi d^3$ je-li

$$d = 3.7\text{cm} \pm 0.05\text{cm}, \quad \pi = 3.14 \ .$$

*Řešení.* Pro určení odhadu absolutní chyby použijeme výše odvozený obecný vzorec

$$\epsilon = \sum_{i=1}^{n} |\frac{\partial f}{\partial x_i}|\epsilon_i \ .$$

Očividně funkce objemu $V$ je závislá na přesnosti proměnných $\pi$ a $d$. Nejdříve napočítejme potřebné parciální derivace

$$\begin{array}{ll} \frac{\partial V(\pi,d)}{\partial \pi} = \frac{1}{6}d^3 \ , & \frac{\partial V(3.14,3.7)}{\partial \pi} = \frac{1}{6}\cdot 3.7^3 \doteq 8.44 \ , \\ \frac{\partial V(\pi,d)}{\partial d} = \frac{1}{2}\pi d^2 \ , & \frac{\partial V(3.14,3.7)}{\partial d} = \frac{1}{2}\cdot 3.14 \cdot 3.7^2 \doteq 21.5 \ . \end{array}$$

a dosazením

$$\epsilon = |\frac{\partial V}{\partial \pi}||\triangle \pi| + |\frac{\partial V}{\partial d}||\triangle d| = 8.44 \cdot 0.0016 + 21.5 \cdot 0.05 \doteq 1.1$$

Tedy $V = \frac{1}{6}\pi d^3 \approx 27.4\text{cm}^3 \pm 1.1\text{cm}^3$.

Pro určení odhadu relativní chyby použijeme snažší vyjádření

$$\delta = |\frac{\triangle V}{V}| \approx \frac{1.088}{27.4} \doteq 3.97 \ .$$

▲

### 1.3.4 Obrácená úloha

Mnohdy je nutné odhadnout chybu nezávislé proměnné tak, aby přitom byla zaručena přesnost výsledné veličiny.
Platí

$$|\triangle u| \leqq \sum_{i=1}^{n} |\frac{\partial f}{\partial x_i}|\epsilon_i \ .$$

Jsou-li chyby $n$ nezávislých proměnných $x_i$ řádově stejné, pak můžeme jejich příspěvek do celkové chyby rozdělit stejným dílem tj.

$$|\frac{\partial f}{\partial x_i}|\epsilon_i \doteq \frac{\triangle u}{n} \text{ pro } \forall i = 1, 2, \ldots, n \ ,$$

pak

$$\epsilon_i \doteq \frac{|\triangle u|}{n|\frac{\partial f}{\partial x_i}|} \ . \tag{1.2}$$

**Příklad 1.17.** Nechť je dán válec o poloměru $r \approx 2\text{m}$ a výšce $h \approx 3\text{m}$. Jaké musí být absolutní chyby $\triangle r$ a $\triangle h$ aby objem válce byl určen s přesností na $0.1\text{m}^3$?

*Řešení.* Objem válce lze spočítat pomocí vzorce

$$V = \pi r^2 h \ ,$$

přičemž ze zadání víme, že

$$\begin{aligned} r &= 2\text{m} \ , & h &= 3\text{m} \ , \\ \pi &= 3.14 \ , & \triangle V &= 0.1\text{m}^3 \ . \end{aligned}$$

Potřebné odhady budeme počítat užitím vzorce (1.2), je proto nutno nejdříve napočítat potřebné parciální derivace

$$\begin{aligned} \tfrac{\partial V(r,h,\pi)}{\partial r} &= 2\pi r h \ , & \tfrac{\partial V(2,3,3.14)}{\partial r} &= 2 \cdot 3.14 \cdot 2 \cdot 3 \doteq 37.7 \ , \\ \tfrac{\partial V(r,h,\pi)}{\partial h} &= \pi r^2 \ , & \tfrac{\partial V(2,3,3.14)}{\partial h} &= 3.14 \cdot 2^2 \doteq 12.6 \ , \\ \tfrac{\partial V(r,h,\pi)}{\partial \pi} &= r^2 h \ , & \tfrac{\partial V(2,3,3.14)}{\partial \pi} &= 2^2 \cdot 3 = 12 \ . \end{aligned}$$

Při dosazování do vzorce je nutno si uvědomit, že přesnost výsledného objemu je závislá na třech vstupních parametrech, tedy do vzorců budeme dosazovat $n = 3$.

$$\begin{aligned} |\triangle r| &\leqq \epsilon_r = \tfrac{|\triangle V|}{n|\frac{\partial V}{\partial r}|} \doteq \tfrac{0.1}{3 \cdot 37.7} &< 0.001 \\ |\triangle h| &\leqq \epsilon_h = \tfrac{|\triangle V|}{n|\frac{\partial V}{\partial h}|} \doteq \tfrac{0.1}{3 \cdot 12.6} &< 0.003 \\ |\triangle \pi| &\leqq \epsilon_\pi = \tfrac{|\triangle V|}{n|\frac{\partial V}{\partial \pi}|} \doteq \tfrac{0.1}{3 \cdot 12} &< 0.003 \end{aligned}$$

Proto pro dosažení požadované přesnosti výpočtu objemu válce je nutno, aby $\triangle r < 0.001\text{m}$ a $\triangle h < 0.003\text{m}$. ▲

## 1.4 Aproximace čísel na počítači

Zásadím problémem použití počítačů pro výpočty s požadavkem na vysokou přesnost je nemožnost reprezentace realných, přesněji iracionálních čísel v počítači. Je zřejmé, že irracionální čísla, která mají nekonečný počet desetinných míst nelze nikterak uložit v paměti počítače. Abychom tato čísla byli schopni v počítači reprezentovat, musíme je aproximovat čísly s konečným počtem desetiných míst.

Pro tyto účely se nejvíce hodí tzv. *semilogaritmický tvar s normalizovanou mantisou.*

**Definice 1.18.** Pod pojmem *semilogaritmický tvar s normalizovanou mantisou* rozumíme zápis libovolného racionálního čísla $a$ ve tvaru

$$a = sgn\ a.(a_1 q^{-1} + a_2 q^{-2} + \cdots + a_t q^{-t}) q^b\ ,$$

kde $q \in \mathbb{N} \setminus \{1\}$ je *základ číselné soustavy*, $a_i \in \{0, 1, 2 \ldots, q-1\}$ jsou *číslice mantisy* a $b \in \mathbb{Z}$ je *exponent*, který je omezen maximální a minimální hodnotou $m_1, m_2 \in \mathbb{Z}$, tj.

$$m_1 \leqq b \leqq m_2\ .$$

**Příklad 1.19.** Dříve než budeme pokračovat ve výkladu, zopakujme si na konkrétním příkladě alespoň převody mezi dekadickou ($q = 10$) a binární ($q = 2$) soustavou.

Převeďte čísla $[10011]_2, [10.011]_2$ do dekadické soustavy[1].
Převeďte čísla $[1234]_{10}, [12.34]_{10}$ do binární soustavy.

*Řešení.* Dle standartní teorie číselných soustav známé ze střední školy lze psát

$$\begin{aligned} [10011]_2 &= [1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0]_{10} \\ &= [16 + 0 + 0 + 2 + 1]_{10} = [19]_{10} \end{aligned}$$

$$\begin{aligned} [10.011]_2 &= [1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}]_{10} \\ &= [2 + 0 + 0 + 0.25 + 0.125]_{10} = [2.375]_{10} \end{aligned}$$

Převod z dekadické do binární soustavy lze provést obdobně - hledáním vhodných koeficientů v součtu jednotlivých mocnin základu $q = 2$ (tj. cifer v binárním zápisu). Nicméně opět ze střední školy je znám rychlejší postup. Uvedeme jej bez zbytečného komentáře.

$$\left.\begin{array}{rll} 1234 & :2 = & 617, \quad \text{zbytek } 0 \\ 617 & :2 = & 38, \quad \text{zbytek } 1 \\ 38 & :2 = & 19, \quad \text{zbytek } 0 \\ 19 & :2 = & 8, \quad \text{zbytek } 1 \\ 8 & :2 = & 4, \quad \text{zbytek } 0 \\ 4 & :2 = & 2, \quad \text{zbytek } 0 \\ 2 & :2 = & 1, \quad \text{zbytek } 0 \\ 1 & :2 = & 0, \quad \text{zbytek } 1 \end{array}\right\} \quad \Rightarrow \quad [1234]_{10} = [10001010]_2$$

[1] Zápis $[x]_q$ znamená, že číslo $x$ je uvedeno v číselné soustavě se základem $q$.

Neméně zajímavá je i modifikace postupu pro převod desetinných čísel.

$$\begin{array}{rcll} 12 & :2 = & 6, & \text{zbytek } 0 \\ 6 & :2 = & 3, & \text{zbytek } 0 \\ 3 & :2 = & 1, & \text{zbytek } 1 \\ 1 & :2 = & 0, & \text{zbytek } 1 \end{array} \qquad \begin{array}{rcll} 0.34 & \cdot 2 = & 0.68, & \text{celá část } 0 \\ 0.68 & \cdot 2 = & 1.36, & \text{celá část } 1 \\ 0.36 & \cdot 2 = & 0.72, & \text{celá část } 0 \\ 0.72 & \cdot 2 = & 1.44, & \text{celá část } 1 \\ 0.44 & \cdot 2 = & 0.88, & \text{celá část } 0 \\ 0.88 & \cdot 2 = & 1.76, & \text{celá část } 1 \\ 0.76 & \cdot 2 = & 1.52, & \text{celá část } 1 \\ 0.52 & \cdot 2 = & 1.04, & \text{celá část } 1 \\ & & & \dots \end{array}$$

$$\Rightarrow \quad [12.34]_{10} = [1100.01010111\dots]_2$$

▲

## Pro zájemce:

Pozornému čtenáři jistě neuniklo několik zajímavých vlastností vyskytujících se v předchozím příkladě:

$$\begin{array}{rcl} 10011 \cdot 10^{-3} & = & 10.011 \\ 19 \cdot 2^{-3} & = & 2.375 \end{array} \qquad \begin{array}{rcl} 1234 \cdot 10^{-2} & = & 12.34 \\ 10001010 \cdot 2^{-2} & \neq & 1100.01010111\dots \end{array}$$

Tato pozorování ponecháváme čtenáři k zamyšlení v kontextu s Definicí 1.18.

**Definice 1.20.** Množinu všech čísel zapsaných pomocí semilogaritmického tvaru s normalizovanou mantisou (viz Definice 1.18) nazýváme *systémem s pohyblivou řádovou čárkou*[a] a značíme

$$F(q, t, m_1, m_2) \ .$$

[a] angl. *floating point system*

**Poznámka 1.21.** V případě tzv. dvojnásobné aritmetiky nahrazujeme $t$ hodnotou $2t$ a zvětšujeme $m_1$ a $m_2$.

V případě reálných čísel je tedy možné každé číslo $x \in \mathbb{R}$ reprezentovat následujícím způsobem

$$x = a.q^b = sgn\ x(\sum_{n=1}^{\infty} x_k q^{-k})q^b$$

Jeho aproximace v semilogaritmickém tvaru s normalizovanou mantisou pak může mít tvar

$$\tilde{x} \in F(q, t, m_1, m_2) \Rightarrow \tilde{x} = \tilde{a}q^b = sgn\ \tilde{x}(\sum_{n=1}^{t} \tilde{x}_k q^{-k})q^b$$

Reprezentaci reálných čísel na počítači tedy definujme pomocí zobrazení

$$fl : \mathbb{R} \to F(q, t, m_1, m_2)$$

Zobrazení provádíme

a) řezáním - $\tilde{x}_k = x_k$, $k = 1, 2, \ldots t$

b) zaokrouhlováním -
$\tilde{x}_k = x_k$, pro $k = 1, 2, \ldots t-1$
$x_t = \begin{cases} x_t + 1 & \text{pro} \quad x_{t+1} \geqq \frac{q}{2} \\ x_t & \text{pro} \quad x_{t+1} < \frac{q}{2} \end{cases}$

**Poznámka 1.22.** $|\frac{x - fl(x)}{x}| = \kappa q^{1-t}$, kde $\kappa = 1$ pro řezání a $\kappa = \frac{1}{2}$ pro zaokrouhlování.
$|x - fl(x)| = \kappa q^{b-t}$

Nejmenší možné kladné číslo, které jsme schopni uložit je určeno délkou mantisy. Čísla lišící se o toto číslo se jeví jako totožná.

**Definice 1.23.** Nejmenší číslo $x \in F(q, t, m_1, m_2)$ takové, že $1 + x > 1$, $x > 0$ nazýváme počítačovým $\epsilon$ (počítačová jednotka).

# Kapitola 2

# Numerické řešení nelineárních rovnic

### Průvodce studiem

*Cílem této kapitoly je představit základní numerické metody pro řešení nelineárních rovnic a jejich soustav. Hledání kořenů obecně nelineárních rovnic je snad nejběžnější matematickou úlohou, se kterou se můžeme setkat nejen v inženýrské praxi, ale i v každodenním životě, například při řešení jednoduchých geometrických úloh. Analytické řešení těchto rovnic je však velmi často netriviální a nezbývá než nasadit na jejich řešení numerické metody. V této kapitole se nejprve čtenář seznámí s metodami separace kořenů těchto rovnic. Následně budou odvozeny a analyzovány základní numerické metody, které je na řešení nelineárnich rovnic a jejich soustav možné použít.*

## 2.1 Separace kořenů

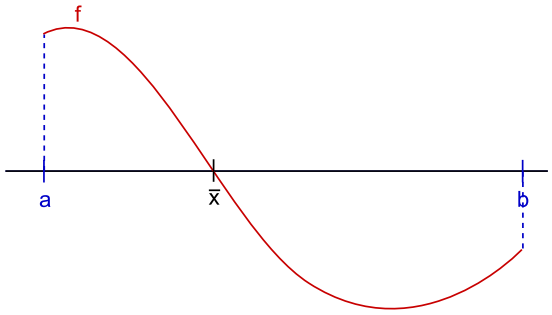
Prvním krokem, který předchází numerickému řešení nelineárních rovnic je lokalizace jejich kořenů. Pro obecné nelineární rovnice může být tento úkol velmi problematicky řešitelný a nelze jej řešit jinak než zobrazením grafu funkce. V případě soustav mnoha nelineárních rovnic je však i tento postup prakticky neřešitelný. Naopak pro některé specifické typy rovnic jako jsou algebraické rovnice, existuje již poměrně rozsáhlý aparát. Procesu určování počtu kořenů a jejich lokalizace říkáme separace kořenů a základní principy si ukážeme v následujícím textu.
Nejprve si definujme co budeme rozumět pod pojmem rovnice a jí kořen.

**Definice 2.1.** Nechť funkce $f : \mathbb{R} \to \mathbb{R}$ je definována na intervalu $\langle a, b\rangle \subset \mathbb{R}$. Číslo $\bar{x} \in \langle a, b\rangle$, pro které $f(\bar{x}) = 0$, nazýváme *kořenem rovnice* $f(x) = 0$.

Pro určení intervalu, ve kterém zcela jistě nalezneme kořen rovnice, můžeme použít následujícího triviálního tvrzení.

**Lemma 2.2.** *Nechť $f$ je spojitá na $\langle a, b\rangle \subset \mathbb{R}$. Pak je-li $f(a).f(b) < 0$ má rovnice $f(x) = 0$ alespoň jeden kořen v intervalu $\langle a, b\rangle$.*

*Důkaz.* Viz Obrázek 2.1. □




Obr. 2.1: Příklad funkce spojité na $\langle a, b\rangle$ s vlastností $f(a).f(b) < 0$.

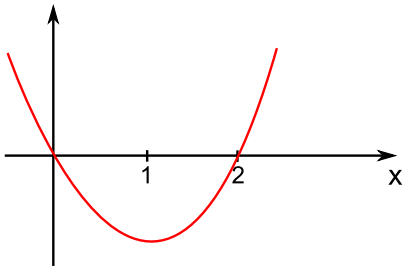
Uvedené Lemma nám sice zaručuje existenci kořenu v daném intervalu, avšak nezaručuje, že kořen v daném intervalu bude právě jeden.
Pro nasazení numerických metod je však mnohdy velmi důležité aby kořen byl v daném intervalu pouze jeden.

Navíc se jedná pouze o podmínku postačující neboť v případě, že v daném intervalu budou kořeny dva nebo obecněji sudý počet, nebude podmínka splněna.

Pro obecné funkce existuje několik základních způsobů separace kořenů:

1. Grafická separace

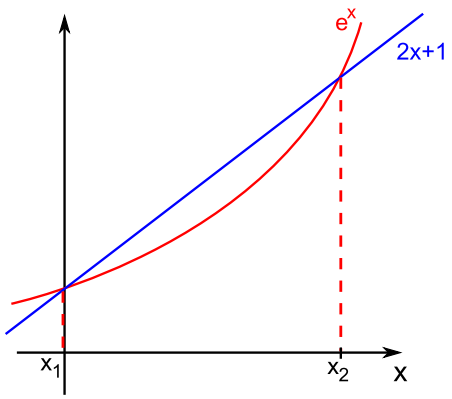
   a) polohu a počet kořenů určíme z grafu funkce $f(x)$.
   Například z grafu funkce $f(x) = (x-1)^2 - 1$ (viz Obrázek 2.2) lze určit




Obr. 2.2: Průběh funkce $f(x)$.

   přibližnou polohu koeficientů $\bar{x}_1 \in (-\frac{1}{2}, \frac{1}{2}), \bar{x}_2 \in (1, 5)$.

b) polohu a počet kořenů určíme vhodným rozložením funkce $f(x)$ na více funkcí a z grafů těchto funkcí.
$(f(x) = f_1(x) - f_2(x) = 0, f_1(x) = f_2(x))$
Například funkci $f(x) = \mathrm{e}^x - 2x - 1$ lze rozložit na rozdíl $f(x) = \mathrm{e}^x - (2x + 1) \overset{\text{ozn}}{=} f_1(x) - f_2(x)$ a pak graficky hledat řešení rovnice $f_1(x) = f_2(x)$ (viz Obrázek 2.3).
Z obrázku pak lze odhadnout polohu kořenů $\bar{x}_1 \in (-\frac{1}{2}, \frac{1}{2}), \bar{x}_2 \in (1, 2)$.




Obr. 2.3: Průběh funkcí $f_1(x)$ a $f_2(x)$.

2. Tabelací funkce
Například pro funkci $f(x) \overset{\text{def}}{=} \mathrm{e}^x - 2x - 1$ lze spočítat funkční hodnotu v několika bodech a ze změny znaménka následně odhadnout polohu kořenů.

| $x_i$ | $f(x_i)$ | sgn $f(x_i)$ |
|---|---|---|
| -1 | 1.3679 | + |
| -0.5 | 0.6065 | + |
| 0 | 0 | 0 |
| 0.5 | -0.3513 | - |
| 1 | -0.4817 | - |
| 1.5 | 0.4817 | + |
| 2 | 2.3691 | + |

Ve zvoleném příkladu jsme měli štěstí a podařilo se nám přímo nalézt jeden kořen $\bar{x}_1 = 0$. Očividně pro druhý kořen platí $\bar{x}_2 \in (1, 1.5)$.

**Poznámka 2.3.** Pro polynomické rovnice $(p(x) = 0, p \in P_n)$ existují speciální metody pro separaci kořenů (viz např. [7]).

Podobně jako pro funkce jedné reálné proměnné definujeme nelineární rovnici a její kořen, tak pro vektorové funkce více proměnných můžeme definovat soustavu rovnic a vektor kořenů soustavy.

**Definice 2.4.** Nechť funkce $F : \mathbb{R}^n \to \mathbb{R}^n$ je definována na souvislé oblasti (ohraničená, otevřená podmnožina $\mathbb{R}^n$) $\Omega \subseteq \mathbb{R}^n$. Vektor $\underline{\bar{x}} \in \Omega$, pro který $F(\underline{\bar{x}}) = \underline{o}$ nazýváme *vektorem kořenů soustavy* nelineárních rovnic $F(\underline{\bar{x}}) = \underline{o}$.

**Poznámka 2.5.** Soustavu $F(\underline{x}) = \underline{o}$ obvykle zapisujeme ve tvaru:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

Pro srovnání kvality jednotlivých numerických metod co do rychlosti konvergence, zavádíme pojem *řád iterační metody.*

**Definice 2.6.** Nechť $\{x_n\}$ je posloupnost aproximací čísla $x$, $x_n \to \bar{x}$ pro $n \to \infty$. Říkáme, že iterační metoda je *řádu* $p$, jestliže

$$\exists C = \texttt{konst} > 0, \quad \text{takové, že} \quad |\bar{x} - x_{n+1}| \leqq C|\bar{x} - x_n|^p, \quad \text{pro } n = 0, 1, 2, \ldots \ .$$

Prakticky tedy znamená, že metoda řádu $p$ má chybu nové aproximace až na konstantu menší než $p$-tou mocninu chyby předchozí aproximace. Čím větší řád metody, tím rychleji tedy metoda konverguje.

## 2.2 Metoda půlení intervalu

Snad nejznámnější a nejjednodušší numerickou metodou je *metoda půlení intervalu.* Tato metoda je založena na Větě 2.2.

Nechť $a_0 = a, b_0 = b$ jsou takové, že $f(a_0).f(b_0) < 0$ a nechť $f$ je spojitá na $\langle a, b\rangle$. Z Věty 2.2 vyplývá, že v intervalu $\langle a, b\rangle$ leží alespoň jeden kořen. Předpokládejme dále, že zde leží právě jeden kořen.
Metoda půlení intervalu je založena na konstrukci posloupnosti do sebe vnořených podintervalů $\{a_n\}, \{b_n\}$

$$\langle a_0, b_0\rangle \supset \langle a_1, b_1\rangle \supset \ldots \supset \langle a_n, b_n\rangle \supset \langle a_{n+1}, b_{n+1}\rangle \supset \ldots$$

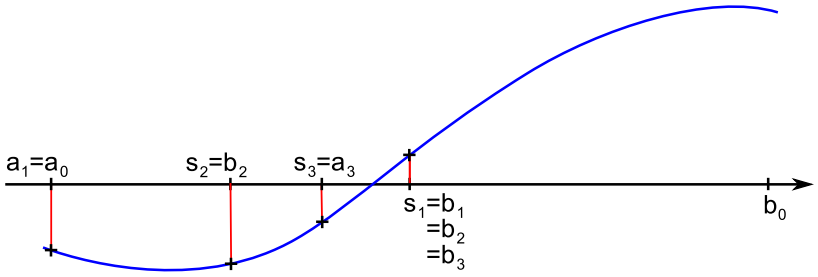
takových, že

$$f(a_k).f(b_k) \leqq 0, \qquad \forall k = 0, 1, 2, \ldots \ .$$

Je-li $\langle a_k, b_k\rangle$ interval pro něhož je $f(a_k).f(b_k) \leqq 0$, pak $\langle a_{k+1}, b_{k+1}\rangle$ dostaneme následujícím způsobem:

1. vypočteme $s_{k+1} = \frac{1}{2}(a_k + b_k)$ (střed intervalu),
2. je-li $f(a_k).f(s_{k+1}) < 0$, pak $a_{k+1} = a_k, b_{k+1} = s_{k+1}$,
3. je-li $f(s_{k+1}).f(b_k) < 0$, pak $a_{k+1} = s_{k+1}, b_{k+1} = b_k$.

Princip metody můžeme snadno demonstrovat na Obrázku 2.4. Z obrázku je taktéž patrné odkud pochází název metody, neboť v každém kroce dochází k rozpůlení intervalu a zmenšení intervalu, ve kterém se nachází kořen, na polovinu.




Obr. 2.4: Metoda půlení intervalu.

**Algoritmus 1** METODA PŮLENÍ INTERVALU

**Input:** $a_0$, $b_0$, přesnost $\epsilon > 0$

```
k := 0
while ||b_k − a_k|| ≥ ε do
    s_{k+1} := (a_k + b_k)/2
    if f(a_k).f(s_{k+1}) < 0 then
        a_{k+1} := a_k
        b_{k+1} := s_{k+1}
    else
        if f(s_{k+1}).f(b_k) < 0 then
            a_{k+1} := s_{k+1}
            b_{k+1} := b_k
        else
            a_{k+1} := s_{k+1}
            b_{k+1} := s_{k+1}
        end if
    end if
    k := k + 1
end while
```

**Output:** $s_k, a_k, b_k$

### 2.2.1 Konvergence

Nyní ukažme, že výše uvedená konstrukce podintervalů konverguje k hledanému kořenu.

**Lemma 2.7.** *Jsou-li splněny předpoklady věty 2.2, pak je metoda půlení intervalu konvergentní. Řád konvergence je 1.*

*Důkaz.* Označme

$$\begin{array}{rcll} s_{k+1} & := & \frac{a_k+b_k}{2} & \text{střed intervalu } \langle a_k, b_k\rangle \ , \\ d_k & := & \frac{b_k-a_k}{2} & \text{polovina délky intervalu } \langle a_k, b_k\rangle \ . \end{array}$$

Pak očividně pro hledané řešení $\bar{x}$ platí

$$\bar{x} \in \langle a_k, b_k\rangle \quad \Rightarrow \quad \bar{x} \in \langle s_{k+1} - d_k, s_{k+1} + d_k\rangle \quad \Rightarrow \quad |\bar{x} - s_{k+1}| \leqq d_k \ .$$

Při půlení intervalu platí rekurentní vzorec

$$b_k - a_k = \frac{b_{k-1} - a_{k-1}}{2}$$

a opakovaným dosazováním do tohto vzorce získáme

$$b_k - a_k = \frac{b_{k-1} - a_{k-1}}{2} = \frac{b_{k-2} - a_{k-2}}{2^2} = \cdots = \frac{b_0 - a_0}{2^k} \ .$$

Dosazením do předpisu pro $d_k$ získáme

$$d_k = \frac{b_k - a_k}{2} = \frac{b_0 - a_0}{2^{k+1}}$$

Spojením předchozích pozorování získáme

$$0 \leqq |\bar{x} - s_{k+1}| \leqq d_k = \frac{b_k - a_k}{2} = \frac{b_0 - a_0}{2^{k+1}}$$

Jelikož pravá strana konverguje k nule pro $k \to \infty$, musí jednoznačně platit

$$|\bar{x} - s_{k+1}| \quad \to \quad 0 \ ,$$

z čehož plyne

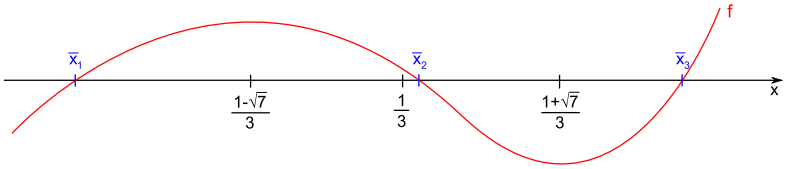
$$s_{k+1} \quad \to \quad \bar{x} \ .$$

Metoda půlení intervalu je tedy konvergentní.
Navíc $d_{k+1} = \frac{1}{2} d_k$, řád metody je tedy 1, s konstantou $C = \frac{1}{2} > 0$. □

**Příklad 2.8.** Nalezněte kořeny polynomu

$$f(x) \stackrel{\text{def}}{=} x^3 - x^2 - 2x + 2$$

s přesností $\epsilon = 10^{-4}$.




Obr. 2.5: K Příkladu 2.8 - kvalitativní vlastnosti funkce.

*Řešení.* Nejdříve provedeme jednoduchou analýzu dané funkce pro nalezení intervalových odhadů jednotlivých kořenů.
Z první a druhé derivace zadané funkce

$$f'(x) = 3x^2 - 2x - 2, \quad f''(x) = 6x - 2$$

lze sestavit tabulky monotonie a konvexnosti/konkávnosti dané funkce.

| $x$ | $f'(x)$ | $f$ |
|---|---|---|
| $\left(-\infty, \frac{1-\sqrt{7}}{3}\right)$ | $> 0$ | rostoucí |
| $\left(\frac{1-\sqrt{7}}{3}, \frac{1+\sqrt{7}}{3}\right)$ | $< 0$ | klesající |
| $\left(\frac{1+\sqrt{7}}{3}, \infty\right)$ | $> 0$ | rostoucí |

| $x$ | $f'(x)$ | $f$ |
|---|---|---|
| $\left(-\infty, \frac{1}{3}\right)$ | $< 0$ | konkávní |
| $\left(\frac{1}{3}, \infty\right)$ | $> 0$ | konvexní |

Následně lze dopočítat potřebné funkční hodnoty a načrtnout průběh funkce $f$ (viz Obrázek 2.5).

Budeme pátrat po hodnotách $\bar{x}_1, \bar{x}_2, \bar{x}_3$. Očividně

$$\bar{x}_2 \in \left(\frac{1}{3}, \frac{1+\sqrt{7}}{3}\right) . \tag{2.1}$$

Zbývá určit intervalové odhady kořenů $\bar{x}_1, \bar{x}_2$ a to například pomocí tabelace funkce. Volme tabelační parametr $h = 0,2$.

| $x$ | $f$ |
|---|---|
| $\frac{1-\sqrt{7}}{3}$ | $2,6311$ |
| $\frac{1-\sqrt{7}}{3} - h$ | $2,5173$ |
| $\frac{1-\sqrt{7}}{3} - 2h$ | $2,1438$ |
| $\frac{1-\sqrt{7}}{3} - 3h$ | $1,4627$ |
| $\frac{1-\sqrt{7}}{3} - 4h$ | $0,4258$ |
| $\frac{1-\sqrt{7}}{3} - 5h$ | $-1,0146$ |

| $x$ | $f$ |
|---|---|
| $\frac{1+\sqrt{7}}{3}$ | $-0,1126$ |
| $\frac{1+\sqrt{7}}{3} + h$ | $0,0012$ |

Z předchozí tabelace jsou zřejmé odhady kořenů

$$\begin{aligned} \bar{x}_1 &\in \left(\tfrac{1-\sqrt{7}}{3} - 1 \quad , \quad \tfrac{1-\sqrt{7}}{3} - 0,8\right) \\ \bar{x}_3 &\in \left(\tfrac{1+\sqrt{7}}{3} \quad , \quad \tfrac{1+\sqrt{7}}{3} + 0,2\right) . \end{aligned} \tag{2.2}$$

Pak metodou půlení intervalu získáme v jednotlivých iteracích následující odhady. Pro hledání kořene $\bar{x}_1$ získáme následující tabulku.

| $i$ | $a$ | $s$ | $b$ |
|---|---|---|---|
| 0. | −1.5486 | −1.4486 | −1.3486 |
| 1. | −1.4486 | −1.3986 | −1.3486 |
| 2. | −1.4486 | −1.4236 | −1.3986 |
| 3. | −1.4236 | −1.4111 | −1.3986 |
| 4. | −1.4236 | −1.4173 | −1.4111 |
| 5. | −1.4173 | −1.4142 | −1.4111 |
| 6. | −1.4173 | −1.4158 | −1.4142 |
| 7. | −1.4158 | −1.4150 | −1.4142 |
| 8. | −1.4150 | −1.4146 | −1.4142 |
| 9. | −1.4146 | −1.4144 | −1.4142 |
| 10. | −1.4144 | −1.4143 | −1.4142 |
| 11. | −1.4143 | −1.4143 | −1.4142 |

Pro hledání kořene $\bar{x}_2$ získáme následující tabulku.

| $i$ | $a$ | $s$ | $b$ |
|---|---|---|---|
| 0. | 0.3333 | 0.3333 | 1.2153 |
| 1. | 0.7743 | 0.7743 | 1.2153 |
| 2. | 0.9948 | 0.9948 | 1.2153 |
| 3. | 0.9948 | 1.1050 | 1.2153 |
| 4. | 0.9948 | 1.0499 | 1.1050 |
| 5. | 0.9948 | 1.0223 | 1.0499 |
| 6. | 0.9948 | 1.0086 | 1.0223 |
| 7. | 0.9948 | 1.0017 | 1.0086 |
| 8. | 0.9982 | 0.9982 | 1.0017 |
| 9. | 0.9999 | 0.9999 | 1.0017 |
| 10. | 0.9999 | 1.0008 | 1.0017 |
| 11. | 0.9999 | 1.0004 | 1.0008 |
| 12. | 0.9999 | 1.0002 | 1.0004 |
| 13. | 0.9999 | 1.0000 | 1.0002 |
| 14. | 1.0000 | 1.0000 | 1.0000 |

Pro hledání kořene $\bar{x}_3$ získáme následující tabulku.

| $i$ | $a$ | $s$ | $b$ |
|---|---|---|---|
| 0. | 1.2153 | 1.3153 | 1.4153 |
| 1. | 1.3153 | 1.3653 | 1.4153 |
| 2. | 1.3653 | 1.3903 | 1.4153 |
| 3. | 1.3903 | 1.4028 | 1.4153 |
| 4. | 1.4028 | 1.4090 | 1.4153 |
| 5. | 1.4090 | 1.4121 | 1.4153 |
| 6. | 1.4121 | 1.4137 | 1.4153 |
| 7. | 1.4137 | 1.4145 | 1.4153 |
| 8. | 1.4137 | 1.4141 | 1.4145 |
| 9. | 1.4141 | 1.4143 | 1.4145 |
| 10. | 1.4141 | 1.4142 | 1.4143 |
| 11. | 1.4142 | 1.4142 | 1.4143 |

Hledané kořeny mají hodnoty $\bar{x}_1 = -1.4143, \bar{x}_2 = 1, \bar{x}_3 = 1.4142$ s přesností $\epsilon = 10^{-4}$. ▲

## 2.3 Metoda prostých iterací

Metoda prostých iterací je založena na konstrukci posloupnosti aproximací

$$\{x_k\}, \quad k = 0, 1, 2, \ldots$$

rekurzivním předpisem

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

Pokud má zobrazení $G : \mathbb{R} \to \mathbb{R}$ určité níže uvedené vlastnosti, tak je metoda konvergentní a konverguje k hledanému řešení.

**Definice 2.9.** Zobrazení $G : \mathbb{R}^n \to \mathbb{R}^n$ se nazývá *kontraktivní*, jestliže existuje $0 < \rho < 1$ takové, že

$$\forall x, y \in \mathbb{R}^n : \|G(x) - G(y)\| \leqq \rho \|x - y\|$$

**Definice 2.10.** Nechť je dáno zobrazení $G : \mathbb{R}^n \to \mathbb{R}^n$. Bod $\tilde{x} \in \mathbb{R}^n$ se nazývá *pevný bod* zobrazení $G$ jestliže $G(\tilde{x}) = \tilde{x}$.

Otázku existence pevného bodu zobrazení $G$ řeší následující věta:

**Věta 2.11.** *(o pevném bodě)*
*Nechť*

- *je dána množina* $\Omega \subset \mathbb{R}^n$, $\Omega \neq \emptyset$, *souvislá, ohraničená a uzavřená,*
- *je dáno zobrazení* $G : \mathbb{R} \to \mathbb{R}^n$ *kontraktivní v* $\Omega$,
- *postupné aproximace* $x_k = G(x_{k-1}), k = 1, 2, \ldots$ *leží v* $\Omega$.

*Pak*

1. $\{x_k\}$ *konverguje k* $\bar{x}$ *nezávisle na volbě počáteční aproximace* $x_0$ *t.j.* $x_k \to \bar{x}, k \to \infty$,
2. $\bar{x} = G(\bar{x})$ *v* $\Omega$ *($\bar{x}$ je pevným bodem),*
3. $\|\bar{x} - x_k\| \leqq \frac{\rho^k}{1-\rho}\|x_1 - x_0\|$.

*Důkaz.* 1. Nejdříve uvažujme libovolný index aproximace $k = 0, 1, 2, \ldots$ a odhadněme pomocí vlastnosti normy

$$\begin{aligned} \|x_{k+n} - x_k\| &= (x_{k+1} - x_k) + (x_{k+2} - x_{k+1}) + \cdots + (x_{k+n} - x_{k+n-1}) \\ &\leqq \|x_{k+1} - x_k\| + \|x_{k+2} - x_{k+1}\| + \cdots + \|x_{k+n} - x_{k+n-1}\| \ . \end{aligned} \tag{2.3}$$

Jelikož $G$ je dle předpokladů kontraktivní, lze pro libovolný index $s$ odhadnout

$$\begin{aligned} \|x_{s+1} - x_s\| &= \|G(x_s) - G(x_{s-1})\| \leqq \rho\|x_s - x_{s-1}\| \\ &\leqq \rho^2\|x_{s-1} - x_{s-2}\| \leqq \ \ldots \ \leqq \rho^s\|x_1 - x_0\| \ . \end{aligned} \tag{2.4}$$

Spojením odhadů (2.3) a (2.4) lze získat

$$\begin{aligned} \|x_{k+n} - x_k\| &\leqq \rho^k\|x_1 - x_0\| + \rho^{k+1}\|x_1 - x_0\| + \cdots + \rho^{k+n-1}\|x_1 - x_0\| \\ &\leqq \tfrac{\rho^k}{1-\rho}\|x_1 - x_0\| \ . \end{aligned} \tag{2.5}$$

Z poslední nerovnosti plyne, že pokud $k \to \infty$, pak[1] $\rho^k \to 0$ a

$$\|x_{k+n} - x_k\| \to 0 \ .$$

Jinými slovy - posloupnost $\{x_k\}$ je cauchyovská a dle známého tvrzení, že reálná posloupnost je konvergentní právě tehdy, je-li cauchyovská lze usoudit, že posloupnost $\{x_k\}$ je konvergentní (samozřejmě nezávisle na volbě $x_0$).

2. $G(x)$ je spojité (plyne z kontraktivity) v $\Omega$, pak

$$\bar{x} = \lim_{k\to\infty} x_k = \lim_{k\to\infty} G(x_{k-1}) = G(\lim_{k\to\infty} x_{k-1}) = \bar{x}$$

[1] připomeňme, že $0 < \rho < 1$

Nechť navíc sporem existuje „druhý pevný bod“ $\hat{x} = G(\hat{x})$ a nechť $\hat{x} \neq \bar{x}$. Pak

$$\begin{aligned} \tilde{x} - \hat{x} &= G(\tilde{x}) - G(\hat{x}) \\ \|\tilde{x} - \hat{x}\| &= \|G(\tilde{x}) - G(\hat{x})\| \leqq \rho\|\tilde{x} - \hat{x}\| \\ (1-\rho)\|\tilde{x} - \hat{x}\| &\leqq 0 \end{aligned}$$

Jelikož $0 < \rho < 1$, pak $(1-\rho) > 0$. Z předchozí nerovnosti pak plyne, že musí platit

$$\|\tilde{x} - \hat{x}\| \leqq 0 \ ,$$

což z vlastnosti nezápornosti normy je splněno pouze, pokud $\|\tilde{x} - \hat{x}\| = 0$, tedy $\tilde{x} = \hat{x}$. Což je spor s předpokladem.
$G$ má tedy v $\Omega$ pouze jeden pevný bod.

3. viz (2.5).

□

## Pro zájemce:

Výše uvedená Věta 2.11 je aplikací Banachovy věty o pevném bodě na reálný vektorový prostor. Banachovu větu o pevném bodě pro obecné metrické prostory lze nalézt ve skriptu [4].

Pro důkaz existence pevného bodu zobrazení $G$ je tedy zásadní rozhodnout, zda-li je zobrazení G kontraktivní. To však je velmi netriviální úloha. Z tohoto důvodu vystačíme s postačující podmínkou, která nám taktéž zajistí existenci pevného bodu a všechna tvrzení Věty 2.11.

**Lemma 2.12.** *(postačující podmínka)*
*Nechť $G : \mathbb{R}^n \to \mathbb{R}^n, \Omega \subset \mathbb{R}^n$ z věty 2.11. Nechť $G$ je v $\Omega$ spojitá a nechť existuje $G'$ v $\Omega$ a je spojitá. Nechť navíc*

$$\|G'(x)\| \leqq M < 1 \quad na \quad \Omega$$

*kde $M \in \mathbb{R}$ je konstanta.*
*Jestliže postupně aproximace $x_k = G(x_{k-1})$ leží v $\Omega$, pak platí tvrzení Věty 2.11.*

*Důkaz.* Nechť $x_1, x_2 \in \Omega, y_1 = G(x_1), y_2 = G(x_2)$.
Pak lze odhadnout

$$\|y_1 - y_2\| = \|G(x_1) - G(x_2)\| \leqq \|x_1 - x_2\|.\|G'(\xi)\|, \quad \xi \in \Omega \ .$$

Pak užitím předpokladu věty lze dále odhadnout

$$\|y_1 - y_2\| \leqq \|x_1 - x_2\|.\|G'(\xi)\| \leqq M\|x_1 - x_2\| \ .$$

Navíc $0 < M < 1$, proto zobrazení $G$ je kontraktivní a můžeme použít Větu 2.11. □

Hledání pevného bodu zobrazení $x = G(x)$ je totéž jako hledání kořene rovnice $F(x) = x - G(x) = 0$. Vždy můžeme rovnici $F(x) = 0$ přepsat na tvar $x = G(x)$, kde zobrazení $G$ splňuje dané předpoklady a pro výpočet kořene rovnice použít postupné aproximace hledání pevného bodu. Otázka existence pevného bodu $G$, tedy i kořene rovnice $F(x) = 0$, je pak dána Větou 2.11 a 2.12.
Je zřejmé, že neexistuje pouze jedinný předpis a najít ten, který bude splňovat podmínky konvergence nemusí být zcela triviální.

**Algoritmus 2** METODA PROSTÝCH ITERACÍ

**Input:** $x_0$, přesnost $\epsilon > 0$

1: $x_1 := G(x_0)$
2: $k := 1$
3: **while** $\|x_k - x_{k-1}\| \geq \epsilon$ **do**
4: $\quad x_{k+1} := G(x_k)$
5: $\quad k := k + 1$
6: **end while**

**Output:** $x_k$

**Poznámka 2.13.** $f(x) = 0 \Leftrightarrow x = g(x), \quad f : \mathbb{R} \to \mathbb{R}, g : \mathbb{R} \to \mathbb{R}$
Postačující podmínka (viz věta 2.12) pak obsahuje odhad $|g'(x)| < M < 1$ na $\langle a, b\rangle \subset \mathbb{R}$

- $1 > g'(x) > 0$
  Monotonní konvergence (viz Obrázek 2.6 a))
- $-1 < g'(x) < 0$
  Oscilační konvergence (viz Obrázek 2.6 c))
- $|g'(x)| > 1$
  Nekonverguje (viz Obrázek 2.6 b))

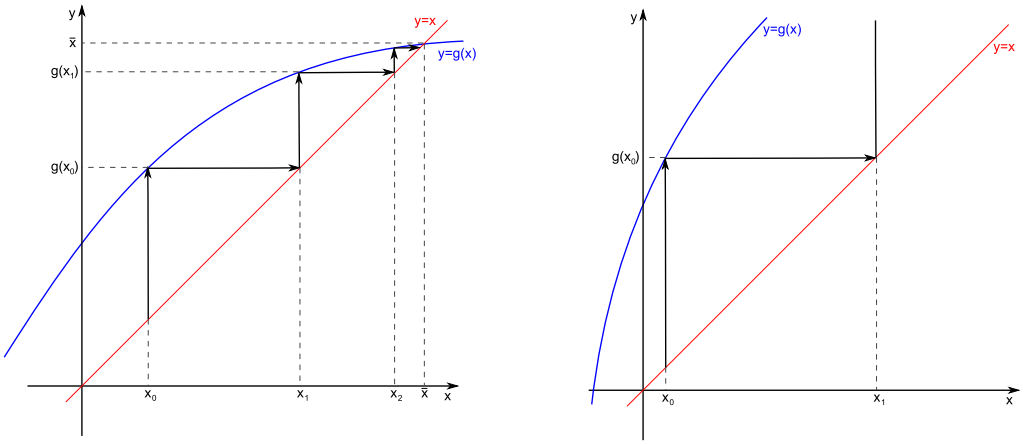
**Příklad 2.14.** Vyčíslete hodnotu $\sqrt{a}, a \in \mathbb{R}^+$ s libovolnou přesností.

*Řešení.* Sestavme nejdříve funkci $f$ popisující rovnici, jejíž řešení hledáme.

$$\begin{aligned} x &= \sqrt{a} \\ x^2 &= a \\ x^2 - a &= 0, \quad f(x) = x^2 - a \\ f(x) &= 0 \end{aligned}$$

Hledejme vhodný předpis pro pevný bod ekvivalentní s původní rovnicí.

$$\begin{aligned} x^2 &= a \\ x &= \tfrac{a}{x} \\ 2x &= x + \tfrac{a}{x} \\ x &= \tfrac{x+\frac{a}{x}}{2}, \quad g(x) = \tfrac{x+\frac{a}{x}}{2} \\ x &= g(x) \end{aligned}$$




a) monotonní

b) nekonverguje

c) oscilační

Obr. 2.6: Příklad konvergence metody prostých iteraci.

Ověřme splnění podmínek konvergence - předpokladů Věty 2.12

$$\begin{aligned} g'(x) &= \frac{1+\frac{-a}{x^2}}{2} = \frac{1}{2}\frac{x^2-a}{x^2}, \quad x > 0 \\ g'(x) &> 0 \quad \text{pro } x > \sqrt{a} \\ g'(x) &< 0 \quad \text{pro } x < \sqrt{a} \end{aligned}$$

Extrémy $g'(x)$:

$$\begin{aligned} \varphi(x) &:= g'(x), \quad \varphi'(x) = \frac{1}{2}\frac{2a}{x^3} = \frac{a}{x^3} \\ \varphi'(x) &> 0 \quad \text{je} - \text{li } a > 0 \\ \varphi'(x) &< 0 \quad \text{je} - \text{li } a < 0 \end{aligned}$$

Funkce $g'$ je tedy pro $a > 0$ rostoucí.

$$\begin{aligned} &\min_{x\in\langle\alpha,\beta\rangle}\{g'(x)\} = g'(\alpha) \\ &\max_{x\in\langle\alpha,\beta\rangle}\{g'(x)\} = g'(\beta) \\ &|g'(x)| < \max\{|g'(\alpha)|, |g'(\beta)|\}, x \in \langle\alpha,\beta\rangle\} \end{aligned}$$

Například pro konkrétní hodnotu úlohy $a = 2$ dostáváme

$$x \in \langle 1,2\rangle,\ g'(2) = \frac{1}{4},\ g'(1) = -\frac{1}{2},\ |g'(x)| = |\frac{1}{2}\frac{x^2-2}{x^2}| < \frac{1}{2}\ .$$

Předpis

$$x_{k+1} = g(x_k) = \frac{1}{2}\left(x_k + \frac{2}{x_k}\right)$$

bude tedy konvergovat pro libovolné $x_0 \in \langle 1,2\rangle$.
Volme proto počáteční aproximaci například $x_0 = 2$.
Při požadované přesnosti $\epsilon = 10^{-5}$ je řešení nalezeno po pěti iteracích, viz následující tabulka.

| $k$ | $x$ | $\|x_k - x_{k-1}\|$ |
|---|---|---|
| 0 | 2 | |
| 1 | 1.5 | 0.5 |
| 3 | 1.4167 | 0.0833 |
| 4 | 1.4142 | 0.0025 |
| 5 | 1.4142 | $2.1239 \cdot 10^{-6}$ |

▲

**Poznámka 2.15.** Označme chybu

$$\mathrm{e}_k = x_k - \bar{x}, \quad \bar{x} = g(\bar{x})\ .$$

Pak lze upravit

$$\begin{aligned} \mathrm{e}_{k+1} &= x_{k+1} - \bar{x} = g(x_k) - g(\bar{x}) \\ &= (x_k - \bar{x})g'(\bar{x}) + (x_k - \bar{x})^2\frac{g''(\bar{x})}{2!} + (x_k - \bar{x})^3\frac{g'''(\bar{x})}{3!} + \dots \\ &= \mathrm{e}_k g'(\bar{x}) + \mathrm{e}_k^2\frac{g''(\bar{x})}{2!} + \mathrm{e}_k^3\frac{g'''(\bar{x})}{3!} + \dots \end{aligned}$$

Řád konvergence tedy závisí na derivaci $g$ v bodě $\bar{x}$.
Je-li

$$\begin{aligned} g^{(k)}(\bar{x}) &= 0, \quad \forall k = 1, 2, \ldots, p-1 \ , \\ g^{(p)}(\bar{x}) &\neq 0 \ , \end{aligned}$$

tak je metoda prostých iterací řádu $p$.

**Poznámka 2.16.** V případě, že $x = G(x)$, $G : \mathbb{R}^n \to \mathbb{R}^n$,
$G$ je diferencovatelná v oblasti $\Omega \subset \mathbb{R}^n$ je

$$G'(x) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1}(x) & \ldots & \frac{\partial g_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial g_n}{\partial x_1}(x) & \ldots & \frac{\partial g_n}{\partial x_n}(x) \end{pmatrix}, G(x) = \begin{pmatrix} g_1(x) \\ \vdots \\ g_n(x) \end{pmatrix}$$

Odhad $\|G'(x)\| < M$ je pak odhad normy matice.
Vypočíst normu takovéto matice nemusí být vůbec snadná úloha!

## 2.4 Newtonova metoda

Původní řešenou rovnici

$$f(x) = 0, \quad f : \mathbb{R} \to \mathbb{R} \tag{2.6}$$

lze upravit na tvar

$$x = x + \lambda(x).f(x) \ , \tag{2.7}$$

kde $\lambda : \mathbb{R} \to \mathbb{R}$ je libovolná nenulová funkce.
Z rovnice (2.7) lze odvodit předpis kontraktivního zobrazení pro metodu prostých iterací s pevným bodem řešení původní rovnice (2.6) (tj. $G(x) = x$)

$$G(x) = x + \lambda(x).f(x) \ . \tag{2.8}$$

Budeme-li požadovat, aby byla metoda řádu 2, pak

$$\begin{aligned} G'(\bar{x}) &= 0 \ , \\ G'(x) &= 1 + \lambda'(x)f(x) + \lambda(x)f'(x) \ , \\ G'(\bar{x}) &= 1 + \lambda'(\bar{x})f(\bar{x}) + \lambda(\bar{x})f'(\bar{x}) = 1 + \lambda(\bar{x})f'(\bar{x}) \ . \end{aligned}$$

Odtud $\lambda = -\frac{1}{f'(\bar{x})}$ pro $f'(\bar{x}) \neq 0$.
Předpokládejme tedy, že $f'(x)$ je nenulová, pak můžeme označit

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

Metoda, která hledá pevný bod takovéto funkce se pak nazývá *Newtonova metoda* pro řešení rovnice (2.6) a má předpis

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots \tag{2.9}$$

**Algoritmus 3** NEWTONOVA METODA

**Input:** $x_0$, přesnost $\epsilon > 0$

1: $x_1 := x_0 - f(x_0)/f'(x_0)$
2: $k := 1$
3: **while** $\|x_k - x_{k-1}\| \geq \epsilon$ **do**
4: $\quad x_{k+1} := x_k - f(x_k)/f'(x_k)$
5: $\quad k := k + 1$
6: **end while**

**Output:** $x_k$

**Poznámka 2.17.** Předpis Newtonovy metody lze snadno získat i z aproximace řešení pomocí Taylorova polynomu. Jelikož

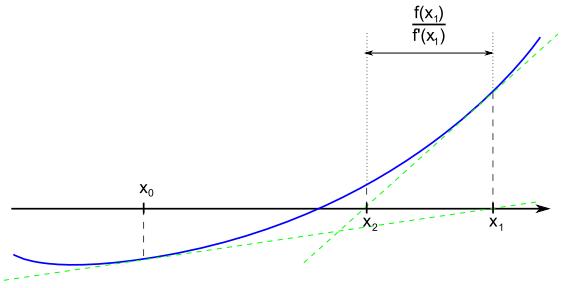
$$f(\bar{x}) \approx f(x_k) + (\bar{x} - x_k)f'(x_k) \qquad \text{(plyne z Taylorovy věty)}$$

a jelikož požadujeme splnění řešení rovnice $f(\bar{x}) = 0$, pak

$$f(x_k) + (\bar{x} - x_k)f'(x_k) = 0 \Rightarrow \bar{x} = x_k - \frac{f(x)}{f'(x)} \ .$$

Iteračně pak získáme předpis (2.9).

Newtonově metodě se taktéž říká metoda tečen. Volba tohoto názvu je patrná z geometrické interpretace , kterou naleznete na Obrázku 2.7.




Obr. 2.7: Newtonova metoda.

Podmínky konvergence Newtonovy metody zaručuje následující věta.

**Lemma 2.18.** *Nechť funkce $f : \mathbb{R} \to \mathbb{R}$ vyhovuje následujícím podmínkám na $\langle a, b\rangle$:*

1. $f(a).f(b) < 0$
2. $f'(x) \neq 0, \forall x \in \langle a, b\rangle$
3. $f''$ *nemá znaménkové změny na* $\langle a, b\rangle$

4. $|f(a)/f'(a)| < b - a$
   $|f(b)/f'(b)| < b - a$

*Pak pro libovolné* $x_0 \in \langle a, b\rangle$ *konverguje Newtonova metoda pro řešení rovnice* $f(x) = 0$ *na* $\langle a, b\rangle$.

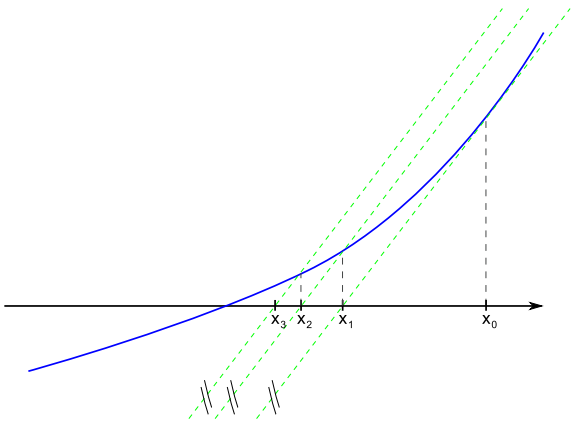
*Důkaz.* Viz literatura, například [8]. □

### 2.4.1 První modifikace Newtonovy metody

V každém kroce Newtonovy metody musíme kromě funkční hodnoty $f(x_k)$ vyhodnocovat i funkční honotu její první derivace $f'(x)$. To však může být výpočetně velmi náročné. Z tohoto důvodu se nám může vyplatit vyhodnotit tuto derivaci jen jednou za čas a počítat několik kroků se stejnou, byť nepřesnou, hodnotou.
Předpis má potom tvar

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

kde $f'(x_l) = f'(x_k)$, je-li $k$ dělitelné $m$.
Geometrickou interpretaci této modifikaci lze naléz na Obrázku 2.4.1.




Obr. 2.8: První modifikace Newtonovy metody.

### 2.4.2 Druhá modifikace Newtonovy metody

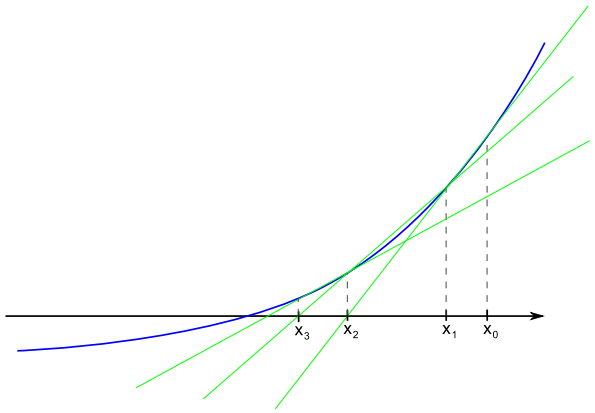
Vyhodnocení první derivace lze rovněž obejít její aproximací předpisem

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \ ,$$

pak předpis metody má tvar

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

Tato metoda se nazývá *metoda sečen.*




Obr. 2.9: Druhá modifikace Newtonovy metody.

### 2.4.3 Newtonova metoda pro soustavy

Uvažujme nyní soustavu

$$F(x) = o, x \in \Omega \ ,$$

tj.

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \ , \\ f_2(x_1, \ldots, x_n) &= 0 \ , \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \ . \end{aligned}$$

Pak v předpisu (2.8) bude $\lambda(x) = -[F'(x)]^{-1}$, kde

$$F'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \ldots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \ldots & \frac{\partial f_n}{\partial x_n}(x) \end{bmatrix} \ .$$

Pak předpis pro Newtonovu formuli bude vypadat následovně

$$x_{k+1} = x_k - [F'(x_k)]^{-1}.F(x_k), \quad x_{k+1} = (x_{k+1}^1, x_{k+1}^2, \ldots, x_{k+1}^n)$$

Uvedený předpis však znamená v každé iteraci sestavit matici $F'(x_k)$.
Pokud označíme přírustek řešení v $k$-tém kroce $\Delta x_k = x_{k+1} - x_k$ a vynásobíme-li předpis maticí $F'(x_k)$, dostaneme nový předpis, ve kterém řešíme soustavu

$$F'(x_k)\Delta x_k = -F(x_k), x_{k+1} = x_k + \Delta x_k \ .$$

V uvedeném předpise je jedinou neznámou přírustek $\Delta x_k$. Jeho výpočet je řešením soustavy lineárních rovnic s maticí $F'(x_k)$ a pravou stranou $-F(x_k)$. Této metodě se taktéž říká *Newtonova-Raphsonova metoda*. V této metodě se jako velmi výhodná ukazuje její první modifikace. Při jejím použití se totiž, řeší několik soustav lineárních rovnic se stejnou maticí. K efektivnímu řešení je pak možné použít LU rozklad, který provedeme pouze jednou za čas a soustavu pro různé pravé strany řešíme pouze dopřednou a zpětnou substitucí.

**Algoritmus 4** NEWTONOVA METODA PRO SOUSTAVY

**Input:** $x_0$, přesnost $\epsilon > 0$

1: nalezni $\Delta x_0$, které řeší soustavu $F'(x_0)\Delta x_0 = -F(x_0)$
2: $x_1 := x_0 + \Delta x_0$
3: $k := 1$
4: **while** $\|x_k - x_{k-1}\| \geq \epsilon$ **do**
5: nalezni $\Delta x_k$, které řeší soustavu $F'(x_k)\Delta x_k = -F(x_k)$
6: $x_{k+1} := x_k + \Delta x_k$
7: $k := k + 1$
8: **end while**

**Output:** $x_k$

**Příklad 2.19.** Pomocí Newtonovy metody nalezněte průsečík křivky

$$\sigma : x^2 + y = 0$$

a elipsy

$$\rho : x^2 + 4y^2 = 1 \ .$$

*Řešení.* Úkolem je tedy řešit soustavu

$$F(x) = o \ ,$$

kde

$$F(x) \stackrel{\text{def}}{=} \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix} = \begin{bmatrix} x^2 + y \\ x^2 + 4y^2 - 1 \end{bmatrix} \ .$$

Pro předpis Newtonovy metody bude nutno spočítat kvadratickou formu

$$F'(x) = \begin{bmatrix} \frac{\partial f_1(x,y)}{\partial x} & \frac{\partial f_1(x,y)}{\partial y} \\ \frac{\partial f_2(x,y)}{\partial x} & \frac{\partial f_2(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x & 1 \\ 2x & 8y \end{bmatrix} \ .$$

Následně inverzi

$$\left[\begin{array}{cc|cc} 2x & 1 & 1 & 0 \\ 2x & 8y & 0 & 1 \end{array}\right] \sim \left[\begin{array}{cc|cc} 2x & 1 & 1 & 0 \\ 0 & 1-8y & 1 & -1 \end{array}\right] \sim \left[\begin{array}{cc|cc} 2x & 1 & 1 & 0 \\ 0 & 1 & \frac{1}{1-8y} & -\frac{1}{1-8y} \end{array}\right] \sim$$

$$\sim \left[\begin{array}{cc|cc} 2x & 0 & 1-\frac{1}{1-8y} & \frac{1}{1-8y} \\ 0 & 1 & \frac{1}{1-8y} & -\frac{1}{1-8y} \end{array}\right] \sim \left[\begin{array}{cc|cc} 1 & 0 & \frac{8y}{2x(1-8y)} & \frac{1}{2x(1-8y)} \\ 0 & 1 & \frac{1}{1-8y} & -\frac{1}{1-8y} \end{array}\right],$$

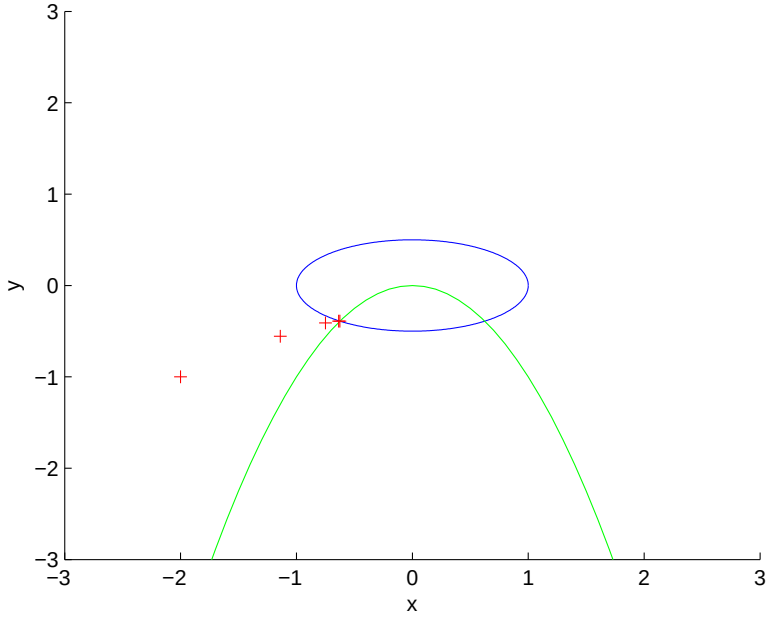
tedy

$$-[F'(x)]^{-1} = \frac{1}{2x(8y-1)} \left[\begin{array}{cc} 8y & 1 \\ 2x & -2x \end{array}\right].$$

Pak předpis pro Newtonovu metodu má tvar

$$\left(\begin{array}{c} x_{k+1} \\ y_{k+1} \end{array}\right) = \left(\begin{array}{c} x_k \\ y_k \end{array}\right) + \frac{1}{2x_k(8y_k-1)} \left[\begin{array}{cc} 8y_k & 1 \\ 2x_k & -2x_k \end{array}\right] \cdot \left[\begin{array}{c} x_k^2 + y_k \\ x_k^2 + 4y_k^2 - 1 \end{array}\right].$$

Při přesnosti $\epsilon = 10^-5$ a volbě počáteční aproximace $[x_0, y_0]^T = [-2, -1]^T$ je algoritmus je ukončen po 4 iteracích. ▲




Obr. 2.10: Konvergence Newtonovy metody v Příkladě 2.19.

# Kapitola 3

# Iterační řešení soustav lineárních rovnic

### Průvodce studiem

*V této kapitole si představíme několik základních metod numerického řešení soustav lineárních rovnic.*
*Řada praktických úloh popisující lineární závislost výstupních dat na datech vstupních vede na řešení soustav lineárních rovnic. Prakticky všechny fyzikální jevy popisované parciálními diferenciálními rovnicemi vedou na tuto úlohu. Na konci předchozí kapitoly jsme si dokonce ukázali, že i numerické metody pro řešení soustav nelineárních rovnic se rovněž převádí na posloupnost řešení soustav lineárních rovnic. Zvládnutí efektivního řešení tohoto problému je tedy klíčové.*
*S analytickými metodami řešení soustav lineárních rovnic jako je Gaussova eliminační metoda či LU rozklad se čtenář může seznámit ve skriptech [2]. Tyto metody však v současnosti ustupují do pozadí, neboť pro řešení velmi rozsáhlých úloh s velkým počtem neznámých jsou prakticky nepoužitelné díky své velké výpočetní a paměťové náročnosti. Z tohto důvodu jsou nahrazovány metodami iteračními, které mají výrazně menší paměťové nároky a v určitých případech jsou i výrazně rychlejší. Hlavní výhodou iteračních metod je však jejich snadná paralelizace pro nasazení na velmi výkonné výpočetní prostředky.*
*V této kapitole se proto seznámíme se základními iteračními metodami a postupy, kterak zefektivnit rychlost jejich konvergence.*

## 3.1 Normy vektorů a matic

Úvodem této kapitoly si připomeňme několik základních pojmů Lineární algebry (viz [2]), které při popisu a analýze iteračních metod řešení soustav lineárních rovnic budeme často používat.

**Definice 3.1.** Nechť $\mathcal{V}$ je vektorový prostor. Zobrazení $\|.\| : \mathcal{V} \to \mathbb{R}_0^+$ nazýváme *normou* jestliže platí

1. $\|x\| \geqq 0, \|x\| = 0 \Leftrightarrow x = 0$,
2. $\|\alpha x\| = |\alpha|.\|x\|$,
3. $\|x + y\| \leqq \|x\| + \|y\|$.

Nechť $\mathcal{V} = \mathbb{R}^n$. Pak prvky vektorového prostoru jsou $n$-dimenzionální aritmetické vektory a hovoříme o *vektorových normách*. Uveďme si několik příkladů vektorových norem:

- $\|x\|_\infty = \max\{|x_i|, i = 1, 2, \ldots n\}$
- $\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}}$
- $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

## Pro zájemce:

Dokažte, že zobrazení definována výše jsou skutečně normy (tj. splňují 3 vlastnosti normy z Definice 3.1).

**Příklad 3.2.** Načrtněte v $\mathbb{R}^2$ množiny

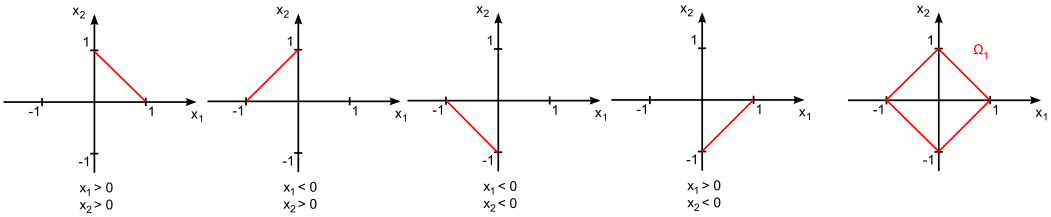
$$\begin{aligned} \Omega_1 &:= \{x \in \mathbb{R}^2 : \|x\|_1 = 1\}\ , \\ \Omega_2 &:= \{x \in \mathbb{R}^2 : \|x\|_2 = 1\}\ , \\ \Omega_\infty &:= \{x \in \mathbb{R}^2 : \|x\|_\infty = 1\}\ . \end{aligned}$$

*Řešení.* 1. $\Omega_1$ - dle definice $\|x\|_1 = 1 \quad \Rightarrow \quad |x_1| + |x_2| = 1$.
Rozdělme naše úvahy na jednotlivé kvadranty:

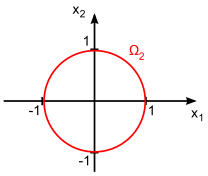
- pro $x_1 > 0 \wedge x_2 > 0$ - přímka s restrikcí na první kvadrant $x_1 + x_2 = 1$,
- pro $x_1 < 0 \wedge x_2 > 0$ - přímka s restrikcí na druhý kvadrant $-x_1 + x_2 = 1$,
- pro $x_1 < 0 \wedge x_2 < 0$ - přímka s restrikcí na třetí kvadrant $-x_1 - x_2 = 1$,
- pro $x_1 > 0 \wedge x_2 < 0$ - přímka s restrikcí na čtvrtý kvadrant $x_1 - x_2 = 1$.

Spojením řešení v jednotlivých kvadrantech získáme celkové řešení (viz Obrázek 3.1).

2. $\Omega_2$ - dle definice $\|x\|_2 = 1 \quad \Rightarrow \quad x_1^2 + x_2^2 = 1$, což je kružnice s poloměrem 1 (viz Obrázek 3.2).




Obr. 3.1: Konstrukce množiny $\Omega_1$ v Příkladě 3.2.




Obr. 3.2: Konstrukce množiny $\Omega_2$ v Příkladě 3.2.

3. $\Omega_\infty$ - Dle definice $\|x\|_\infty = 1 \quad \Rightarrow \quad \max\{|x_1|, |x_2|\} = 1$.
   Uvažujme dva případy (ve kterých je maximum jeden z prvků $x_1, x_2$):

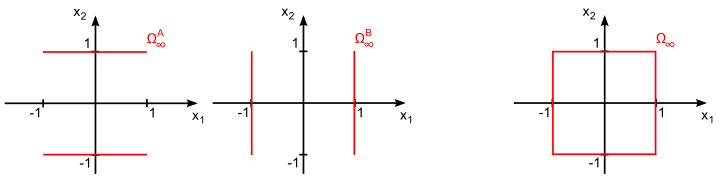
   - pokud $|x_1| \leqq |x_2| = 1$, pak se jedná o množinu bodů

     $$\{[x_1, x_2] \in \mathbb{R}^2 : x_1 \in \langle -1, 1\rangle \wedge x_2 \in \{-1, 1\}\} \overset{\text{ozn}}{=} \Omega_\infty^A$$

   - pokud $|x_2| \leqq |x_1| = 1$, pak se jedná o množinu bodů

     $$\{[x_1, x_2] \in \mathbb{R}^2 : x_1 \in \{-1, 1\} \wedge x_2 \in \langle -1, 1\rangle\} \overset{\text{ozn}}{=} \Omega_\infty^B$$

   Spojením těchto dvou případů získáme celkové řešení (viz Obrázek 3.3).




Obr. 3.3: Konstrukce množiny $\Omega_\infty$ v Příkladě 3.2.

▲

**Poznámka 3.3.** Pro $\mathcal{V} = \mathbb{R}^{m,n}$ nazýváme normu *normou maticovou*.

**Příklad 3.4.** Zobrazení $\|.\| : \mathbb{R}^{m,n} \to \mathbb{R}$ definované předpisem

$$\|A\|_F := \left( \sum_{i,j} |a_{ij}|^2 \right)^{\frac{1}{2}}$$

nazýváme *Frobeniova norma*.

**Definice 3.5.** Nechť je dána vektorová norma $\|.\|_V$ na prostoru $\mathbb{R}^n$. Maticovou normu odpovídající vektorové normě $\|.\|_M$ definujeme následujícím předpisem

$$\|A\|_M := \max_{x \neq o} \left\{ \frac{\|Ax\|_V}{\|x\|_V} \right\} .$$

Uveďme maticové normy odpovídající dříve uvedeným vektorovým normám

- $\|A\|_\infty := \max\{\sum_{k=1}^{n} |a_{jk}|, j = 1, \ldots, n\}$ odpovídá $\|.\|_\infty$ a nazývá se *řádková norma*,
- $\|A\|_1 := \max\{\sum_{j=1}^{n} |a_{jk}|, k = 1, \ldots, n\}$ odpovídá $\|.\|_1$ a nazývá se *sloupcová norma*,
- $\|A\|_2 := \sqrt{\rho(A^T A)} = \sqrt{\rho(A A^T)} = \rho(A)$, kde $A$ je normální matice[1][2], odpovídá $\|.\|_2$ a nazývá se *spektrální norma*.

**Lemma 3.6.** *Nechť $\|.\|_M$ značí maticovou a $\|.\|_V$ odpovídající vektorovou normu. Pak platí*

$$\forall A \in \mathbb{R}^{n \times n}\ \forall x \in \mathbb{R}^n : \quad \|Ax\|_V \leqq \|A\|_M . \|x\|_V ,$$
$$\forall A, B \in \mathbb{R}^{n \times n} : \quad \|AB\|_M \leqq \|A\|_M . \|B\|_M .$$

*Důkaz.* • Dokažme $\|Ax\|_V \leqq \|A\|_M . \|x\|_V$.
Tvrzení je splněno pro $x = o$ triviálně. Předpokládejme tedy, že $x \neq o$, tj. $\|x\|_V \neq 0$ (dle první vlastnosti normy z Definice 3.1). Pak dokazované tvrzení je ekvivalentní s nerovností

$$\frac{\|Ax\|_V}{\|x\|_V} \leqq \|A\|_M .$$

Místo pravé strany lze dosadit z definice odpovídající maticové normy 3.5

$$\frac{\|Ax\|_V}{\|x\|_V} \leqq \max_{x \neq o} \left\{ \frac{\|Ax\|_V}{\|x\|_V} \right\} ,$$

což dle definice maxima platí.

[1] Matice $A$ se nazývá *normální*, pokud $AA^T = A^T A$
[2] $\rho(A)$ je spektrální poloměr matice $A$

- Dokažme $\|AB\|_M \leqq \|A\|_M.\|B\|_M$.
  Nejdříve uvažujme nerovnost

$$\forall x \in \mathbb{R}^n \setminus \{o\} : \|ABx\|_V \leqq \|A\|_M.\|Bx\|_V \ , \tag{3.1}$$

  které je platné díky již dokázanému prvnímu tvrzení Věty 3.6.
  Dále pak úpravou rovnice (3.1) díky stejnému tvrzení získáme

$$\|ABx\|_V \leqq \|A\|_M.\|Bx\|_V \leqq \|A\|_M.\|B\|_M.\|x\|_V \ .$$

  Úpravou (předpokládáme $x \neq o$) dostáváme

$$\frac{\|ABx\|}{\|x\|} \leqq \|A\|_M.\|B\|_M \ .$$

  Uvažujeme-li největší možnou hodnotu výrazu na levé straně (nerovnost platí $\forall x \neq \neq o$), lze nahradit

$$\max_{x \neq o} \left\{ \frac{\|ABx\|}{\|x\|} \right\} \leqq \|A\|_M.\|B\|_M \ .$$

  Dle Definice 3.5 pak přímo dostáváme tvrzení

$$\|AB\|_M \leqq \|A\|_M.\|B\|_M \ .$$

□

Uveďme si ještě jednu důležitou charakteristiku matic, kterou je tzv. *číslo podmíněnosti matice*. V dalším textu ukážeme, že rychlost konvergence některých metod řešení soustav lineárních rovnic souvisí s tímto číslem.

**Definice 3.7.** Nechť $A \in \mathbb{R}^{n,n}$ je regulární. Pak *číslem podmíněnosti matice* rozumíme číslo

$$\mathrm{cond}(A) \stackrel{\mathrm{def}}{=} \|A\|.\|A^{-1}\| \ .$$

*Spektrálním číslem podmíněnosti matice* rozumíme číslo

$$\kappa(A) \stackrel{\mathrm{def}}{=} \rho(A)\rho(A^{-1}) \ .$$

**Poznámka 3.8.** Je-li navíc matice $A$ symetrická a positivně definitní, pak

$$\kappa(A) = \mathrm{cond}(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \ ,$$

kde $\lambda_{\max}, \lambda_{\min}$ jsou největší a nejmenší vlastní čísla matice $A$. Očividně

$$0 < \lambda_{\min} \leqq \lambda_{\max} \quad \Rightarrow \quad \kappa(A) \geqq 1 \ .$$

## 3.2 Lineární metody

Uvažujme soustavu lineárních rovnic

$$Ax = b \ , \tag{3.2}$$

kde

- $A \in \mathbb{R}^{n,n}$ je matice soustavy,
- $b \in \mathbb{R}^n$ je vektor pravých stran,
- $x \in \mathbb{R}^n$ je vektor neznámých.

Je-li matice soustavy regulární, pak existuje jediné řešení $\bar{x}$ takové, že $A\bar{x} = b$. Jinými slovy, je-li matice $A$ regulární, existuje *právě jedno řešení* $\bar{x} = A^{-1}b$.

Zvolme počáteční aproximaci řešení soustavy $x_0 \neq \bar{x}$. Našim úkolem je sestavit posloupnost aproximací

$$x_0, x_1, \ldots, x_k, x_{k+1}, \ldots$$

konvergující k řešení $\bar{x}$, tedy

$$\lim_{k\to\infty} x_k = \bar{x} \ .$$

V následujícím výkladu budeme pátrat po předpisech zobrazení, které každé předešlé aproximaci přiřadí novou aproximaci řešení soustavy $\bar{x}$. Toto zobrazení budeme nazývat *iterační metoda*.

**Definice 3.9.** *Iterační metoda* je zobrazení $\Phi : \mathbb{R}^n \to \mathbb{R}^n$

$$x_{k+1} = \Phi(x_k), \quad k \geqq 0.$$

kde $x_k$ je $k$-tá iterace.

Pokud navrhneme nějakou iterační metodu, bude nutné zajistit, aby metoda konvergovala i k řešení soustavy lineárních rovnic pro kterou byla navržena. O takových metodách pak říkáme, že jsou *konzistentní* s danou soustavou lineárních rovnic.

**Definice 3.10.** Iterační metoda $\Phi$ se nazývá *konzistentní* se soustavou lineárních rovnic $Ax = b$, jestliže pro každou pravou stranu $b \in \mathbb{R}^n$ všechna řešení $\bar{x}$ jsou *pevným bodem* $\Phi$, tj.

$$\bar{x} = \Phi(\bar{x}) \ .$$

Také je nutno, aby se metoda postupnými iteracemi vždy blížila k řešení, a to nezávisle na volbě počáteční aproximace $x_0$.

**Definice 3.11.** Iterační metoda $\Phi$ se nazývá *konvergentní*, jestliže pro $\forall b \in \mathbb{R}^n$ je limita

$$\lim_{k\to\infty} x_k = \bar{x}$$

nezávislá na počáteční aproximaci $x_0$.

V následujícím textu se omezíme na nejjednodušší třídu iteračních metod, které nazýváme *lineární iterační metody*. Na těchto metodách si ukážeme podmínky konzistence a konvergence těchto metod a uvedeme si příklady nejběžněji používaných metod.

**Definice 3.12.** Iterační metoda se nazývá *lineární*, jestliže

$$\Phi(x) = Mx + Nb$$

kde $M, N \in \mathbb{R}^{n\times n}$ jsou regulární matice.
Lineární iterační metoda má pak předpis

$$x_{k+1} = Mx_k + Nb, \quad k \geqq 0.$$

**Věta 3.13.** *Lineární iterační metoda je konzistentní se soustavou lineárních rovnic* $Ax = b$, *právě když*

$$M = I - NA\ .$$

*Důkaz.* Tvrzení je ekvivalence, proto je nutno dokázat jednotlivé implikace.

$\Rightarrow$ Dokažme nejprve, že je-li metoda konzistentní se soustavou lineárních rovnic $Ax = b$, pak $M = I - NA$.
Uvažujme řešení $\bar{x}$

$$A\bar{x} = b \quad \Rightarrow \quad \bar{x} = A^{-1}b\ . \tag{3.3}$$

Jelikož $\bar{x}$ je pevným bodem zobrazení definujícím metodu (je konzistentní s touto soustavou), tak

$$\bar{x} = \Phi(\bar{x}) \quad \Rightarrow \quad \bar{x} = M\bar{x} + Nb\ .$$

Dosaďme nyní předpis (3.3)

$$A^{-1}b = MA^{-1}b + Nb$$

a upravme

$$\begin{aligned} I\ A^{-1}b - MA^{-1}b &= Nb \\ (I\ - M)A^{-1}b &= Nb\ . \end{aligned}$$

Jelikož tato rovnost je z důvodu konzistence splněna pro libovolnou pravou stranu $b$, pak musí platit následující rovnost

$$\begin{aligned} (I\ - M)A^{-1} &= N \\ I\ - M &= NA \\ M &= I\ - NA\ . \end{aligned}$$

⇐ Dokažme dále, že je-li $M = I - NA$, pak je metoda konzistentní se soustavou lineárních rovnic $Ax = b$.
Dosaďme předpoklad tvrzení do předpisu zobrazení iterační metody

$$\Phi(x) = Mx + Nb = (I - NA)x + Nb = x - NAx + Nb = x - N(Ax - b) \ . \quad (3.4)$$

Očividně pevným bodem je řešení $\bar{x}$, jelikož

$$\Phi(\bar{x}) = \bar{x} - N(A\bar{x} - b) = \bar{x} - N(b - b) = \bar{x} \ .$$

Předpokládejme sporem, že $\hat{x} \in \mathbb{R}^n$ je pevným bodem iterační metody $\Phi$ (tj. $\Phi(\hat{x}) = \hat{x}$), ale není řešením soustavy (tj. $A\hat{x} \neq b$). Využitím rovnice (3.4) lze psát

$$\hat{x} = \hat{x} - N(A\hat{x} - b) \ ,$$

úpravou pak

$$N(A\hat{x} - b) = o \ .$$

Uvažujeme-li $N$ regulární (viz Definice 3.12), pak musí platit $A\hat{x} - b = o$, tedy $A\hat{x} = b$. Což je spor.

□

**Poznámka 3.14.** Konzistentní lineární iterační metody mají tvary
1) $x_{k+1} = x_k - N(Ax_k - b)$,
2) $W(x_k - x_{k+1}) = Ax_k - b, N = W^{-1}$.

**Lemma 3.15.** Chyba lineární iterační metody *v k-té iteraci, tj.*

$$\mathrm{e}_k := \bar{x} - x_k$$

*je rovna*

$$\mathrm{e}_k = M^k \mathrm{e}_0 \ .$$

*Důkaz.* Uvažujme přesné řešení soustavy $\bar{x}$, které je pevným bodem zobrazení definujícího lineární metodu, tj.

$$\bar{x} = M\bar{x} + Nb$$

a předpis $k$-té iterace

$$x_k = Mx_{k-1} + Nb.$$

Pak pro jejich rozdíl platí

$$\begin{aligned} \bar{x} - x_k &= M(\bar{x} - x_{k-1}) \\ \mathrm{e}_k &= M\mathrm{e}_{k-1} \ . \end{aligned}$$

Opakovaným užitím předchozí rovnosti získáme

$$\mathrm{e}_k = M\mathrm{e}_{k-1} = M(M\mathrm{e}_{k-2}) = \cdots = M^k \mathrm{e}_0 \ .$$

□

**Věta 3.16.** *Lineární metoda je konvergentní, pokud*

$$\rho(M) < 1 \ ,$$

*kde $\rho(M)$ je spektrální poloměr matice $M$, tj.*

$$\rho(M) := \max\{|\lambda_1|, \ldots, |\lambda_n|\}$$

*a $\lambda_i$ jsou vlastní čísla matice $M$.*

*Důkaz.* Pro důkaz konvergence iterační metody je nutno dokázat, že

$$\lim_{k\to\infty} \mathrm{e}_k = o \ .$$

Dle předpisu chyby z Lemma 3.15 lze dosadit

$$\lim_{k\to\infty} M^k \mathrm{e}_0 = o \ .$$

Jelikož $\mathrm{e}_0$ je v této posloupnosti konstantní, je nutno, aby posloupnost $M^k$ konvergovala k nulové matici.
Nechť

$$J = TMT^{-1}$$

je *Jordanův kanonický tvar* matice $M$ a $T$ je regulární matice (viz [2]).
Úpravou

$$\begin{aligned} M &= T^{-1}JT \\ M^k &= T^{-1}J^kT. \end{aligned}$$

Nyní ukážeme, že $\lim_{k\to\infty} J^k = O$. Dle předchozí rovnice pak bude zřejmé, že i $\lim_{k\to\infty} M^k = O$.
Jelikož $J$ je blokově diagonální, pak její mocnina má tvar

$$J^k = \begin{pmatrix} J_1^k & & & \\ & J_2^k & & \\ & & \ddots & \\ & & & J_m^k \end{pmatrix},$$

kde $J_i, i = 1, \ldots, m$ jsou *Jordanovy bloky* příslušné k vlastním číslům $\lambda_1, \ldots, \lambda_m$.
Nyní se omezme pouze na jeden blok $J_i \in \mathbb{R}^{r,r}$, kde $r$ je násobnost vlastního čísla, kterému daný blok přísluší.
Dle [2] lze tento blok napsat ve tvaru součtu

$$J_i = \lambda_i I + S \ ,$$

kde $I \in \mathbb{R}^{r,r}$ je jednotková matice odpovídající dimenze a

$$S := \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} .$$

Dle binomické věty pro $k$-tou mocninu Jordanova bloku lze psát

$$J_i^k = (\lambda_i I + S)^k = \sum_{j=0}^{k} \begin{pmatrix} k \\ j \end{pmatrix} \lambda_i^{k-j} S^j \ . \tag{3.5}$$

Všimněme si, že mocněním matice $S$ se nenulová vedlejší matice posouvá, proto

$$S^j = O \quad \text{pro } j \geqq r \ ,$$

Pokud v součtu (3.5) vynecháme nulové sčítance, získáme

$$J_i^k = \sum_{j=0}^{m} \begin{pmatrix} k \\ j \end{pmatrix} \lambda_i^{k-j} S^j, \quad m = \min\{k, r-1\} \ .$$

$$J_i^k = c_0 \begin{pmatrix} \lambda_i^k & & & \\ & \lambda_i^k & & \\ & & \ddots & \\ & & & \lambda_i^k \end{pmatrix} + c_1 \begin{pmatrix} 0 & \lambda_i^{k-1} & & \\ & 0 & \ddots & \\ & & \ddots & \lambda_i^{k-1} \\ & & & 0 \end{pmatrix} + \dots$$

$$\dots + c_m \begin{pmatrix} 0 & & & \lambda_i^{k-m} \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{pmatrix} = \begin{pmatrix} c_0\lambda_i^k & c_1\lambda_i^{k-1} & & c_m\lambda_i^{k-m} \\ & c_0\lambda_i^k & \ddots & \\ & & \ddots & c_1\lambda_i^{k-1} \\ & & & c_0\lambda_i^k \end{pmatrix}, \quad c_i := \begin{pmatrix} k \\ i \end{pmatrix} .$$

Pokud platí předpoklad věty $\rho(A) < 1$, pak libovolné vlastní číslo $|\lambda_i| < 1$. Pak jednotlivé prvky matice $J_i^k$ se zvyšujícím se $k$ tvoří geometrickou posloupnost konvergující k nule.
Proto $\lim_{k\to\infty} J_i^k = O$. □

**Poznámka 3.17.** Číslo $-\log(\rho(M))$ nazváme *řád konvergence*.

Určení spektrálního poloměru je výpočetně náročný proces, který je složitější než samo řešení soustavy lineárních rovnic. Abychom tedy vůbec rozhodli o konvergenci lineární iterační metody, museli bychom řešit ještě náročnější problém, což by nebylo příliš efektivní. Z tohoto důvodu můžeme k rozhodnutí o konvergenci lineární iterační metody využít následující postačující podmínku.

**Lemma 3.18.** *Nechť* $\|.\|_M$ *je maticová norma a*

$$\|M\|_M < 1 \ .$$

*Pak konzistentní lineární iterační metoda je konvergentní.*

*Důkaz.* Jelikož (viz Důkaz Lemma 3.15)

$$\mathrm{e}_{k+1} = M\mathrm{e}_k \ ,$$

musí rovnost platit i pro libovolnou vektorovou normu, tj.

$$\|\mathrm{e}_{k+1}\|_V = \|M\mathrm{e}_k\|_V \ .$$

Odpovídá-li maticová norma $\|.\|_M$ vektorové normě $\|.\|_V$, pak užitím nerovnosti 3.6 lze psát

$$\|\mathrm{e}_{k+1}\|_V \leqq \|M\|_M \|\mathrm{e}_k\|_V$$

Při rekurentním zápisu nerovnosti a postupným snižováním $k$ získáme

$$\|\mathrm{e}_{k+1}\|_V \leqq \|M\|_M \|\mathrm{e}_k\|_V \leqq \|M\|_M^2 \|\mathrm{e}_{k-1}\|_V \leqq \ \ldots \ \leqq \|M\|_M^k \|\mathrm{e}_0\|_V \ ,$$

tedy

$$0 \leqq \|\mathrm{e}_{k+1}\|_V \leqq \|M\|_M^k \|\mathrm{e}_0\|_V \ .$$

Jelikož $\mathrm{e}_0$ je nezávislé na $k$, pak očividně metoda je konvergentní, pokud

$$\lim_{k\to\infty} \|M\|_M^k = 0$$

a to platí pouze pokud

$$\|M\|_M < 1 \ .$$

□

Uveďme si několik základních lineárních iteračních metod. Nejprve matici $A$ rozložme

$$A = L + D + U \ , \tag{3.6}$$

kde

- $D$ je diagonální matice,
- $L$ je ostře dolní trojúhelníková matice,
- $U$ je ostře horní trojúhelníková matice.

### 3.2.1 Jacobiova metoda

$$M = -D^{-1}(L+U), \quad N = D^{-1} \ \text{ pokud } D_i \neq 0 \ .$$

Předpis iterační metody má tvar

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

### 3.2.2 Gaussova-Seidelova metoda

$$M = -(D+L)^{-1}U, \quad N = (D+L)^{-1} \ .$$

Předpis iterační metody má tvar

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

### 3.2.3 Successive over-relaxation method (SOR)

$$M = -(D+\omega L)^{-1}(\omega U + (1-\omega)D), \quad N = \omega(D+\omega L)^{-1} \ .$$

Předpis iterační metody má tvar

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

kde $0 < \omega < 2$ je reálná konstanta.

**Poznámka 3.19.** (Odvození lineárních iteračních metod)

Uvažujme soustavu

$$\begin{array}{rcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & = & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & b_3 \end{array}$$

kde $a_{ij}, b_i$ jsou koeficienty soustavy a $x_i$ jsou neznámé.
Soustavu lze upravit na tvar

$$\begin{array}{rcrcrcl} a_{11}x_1 & = & b_1 & - & a_{12}x_2 & - & a_{13}x_3 \\ a_{22}x_2 & = & b_2 & - & a_{21}x_1 & - & a_{23}x_3 \\ a_{33}x_3 & = & b_3 & - & a_{31}x_1 & - & a_{32}x_2 \end{array}$$

a předpokládáme-li nenulové prvky na diagonále, lze například pro $x_1$ psát (vydělením prvního řádku upravené soustavy koeficientem $a_{11}$)

$$x_1 = \frac{1}{a_{11}} \left( b_1 - a_{12}x_2 - a_{13}x_3 \right) \ .$$

Uvažujeme-li navíc iterační metodu, pak pro první složku následující iterace spočtené ze složek předchozí iterace bude platit

$$[x_{k+1}]_1 = \frac{1}{a_{11}} \left( b_1 - a_{12}[x_k]_2 - a_{13}[x_k]_3 \right) \ . \tag{3.7}$$

Pokud budeme uvažovat postupně všechny složky $x$, získáme předpis Jacobiho metody. Z něj již snadno určíme matice $M$ a $N$.
Jelikož v předpisu (3.7) lze při výpočtu $[x_{k+1}]_i$ místo $x_k$ použít již vypočtené složky $x_{k+1}$. Získáme tak předpis pro Gaussovu-Seidelovu metodu.

Princip SOR metody pak spočívá v tom, že novou aproximaci $x_{k+1}$ určíme jako kombinaci předchozí aproximace $x_k$ a aproximace určenou Gaussovou-Seidelovou metodou. Váhu této kombinace pak určuje parametr $\omega$. Pro $\omega = 1$ se jedná o Gaussovu-Seidelovu metodu.

**Algoritmus 5** LINEÁRNÍ ITERAČNÍ METODA

**Input:** $x_0$, $M$, $N$, $b$, přesnost $\epsilon > 0$

1: $k := 0$
2: **while** $\|g_k\| \geq \epsilon\|b\|$ **do**
3: $\quad x_{k+1} := Mx_k - Nb$
4: $\quad k := k + 1$
5: **end while**

**Output:** $x_k$

**Poznámka 3.20.** Jako ukončovací kritérium byla v metodě použita podmínka na relativní chybu rezidua soustavy $g_k := Ax_k - b$, tj. $\|g_k\|/\|b\| < \epsilon$.
Abychom se v praxi vyhli podílu dvou malých čísel, které je co do šíření chyby velmi kritické, používáme raději podmínku

$$\|g_k\| < \epsilon\|b\| \ .$$

**Příklad 3.21.** Řešte soustavu

$$\begin{array}{llll} 5x_1 & +2x_2 & & = 3 \\ -x_1 & +4x_2 & +x_3 & = 0 \\ 2x_1 & -x_2 & +6x_3 & = 1 \end{array}$$

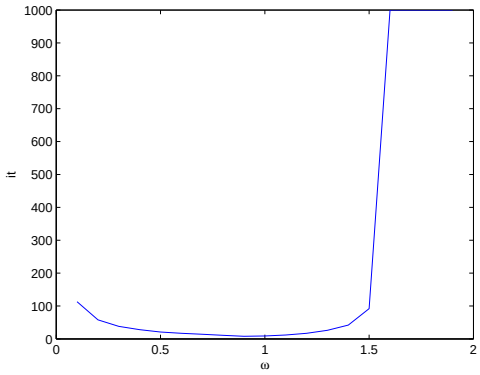
pomocí Jacobiho metody, Gaussovy-Seidelovy metody a SOR s vhodnou volbou parametru $\omega \in (0, 2)$.

*Řešení.* Nejdříve matici soustavy rozložíme dle jednoduchého rozkladu (3.6)

$$A = \begin{bmatrix} 5 & 2 & 0 \\ -1 & 4 & 1 \\ 2 & -1 & 6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 2 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 5 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix} + \begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} =: L+D+U \ .$$

Dle předpisů jednotlivých metod získáme matice $M, N$ a pomocí algoritmu obecné lineární metody spočteme řešení.
Při požadované přesnosti $\epsilon = 10^{-6}$ je Jacobiho metoda ukončena po 17-ti iteracích,

Gaussova-Seidelova metoda po 9-ti iteracích. Na počet iterací algoritmu SOR vplývá volba parametru $\omega$, pro jednotlivé volby je počet iterací znázorněn na Obrázku 3.4 (přičemž pro lepší názornost byl maximální počet iterací omezen na 1000).
Při volbě optimálního $\omega \approx 0.9$ je počet iterací algoritmu SOR 8. ▲



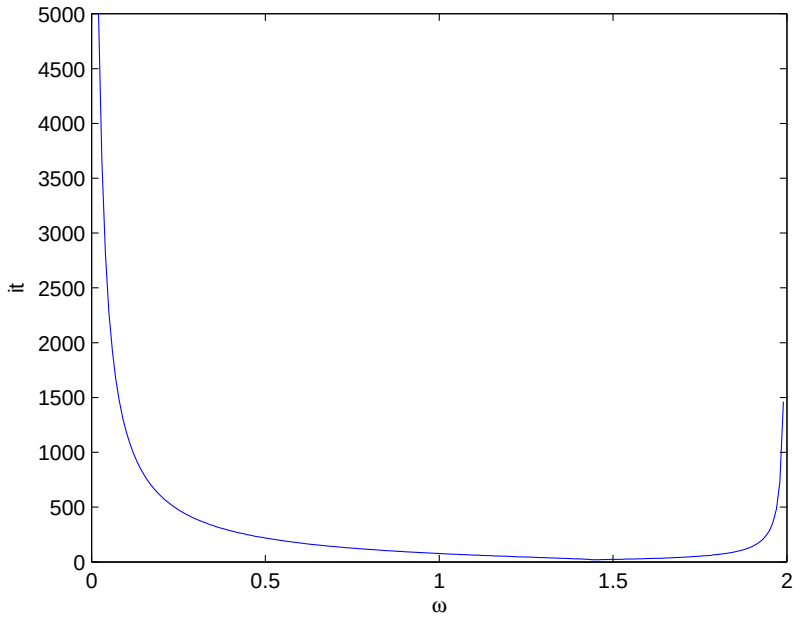

Obr. 3.4: Vliv volby $\omega$ na počet iterací SOR - Příklad 3.21.

**Příklad 3.22.** Otestujte rychlost výše uvedených algoritmů na testovací úloze

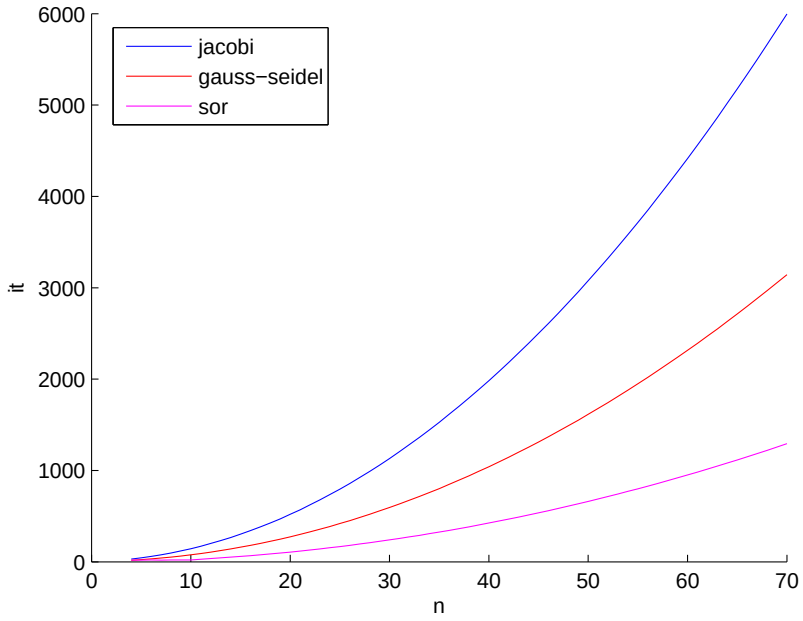
$$\begin{aligned} A &:= fivediag(-1,-1,4,-1,-1) \in \mathbb{R}^{n,n} \\ b &:= [1,\ldots,1]^T \in \mathbb{R}^n \\ x_0 &:= [0,\ldots,0]^T \in \mathbb{R}^n \\ \epsilon &:= 10^{-6} \end{aligned}$$

přičemž volte různou dimenzi úlohy $n = 4, 5, \ldots, 70$.

*Řešení.* Nejdříve nalezněme vhodný parametr $\omega$ pro algoritmus SOR. Volme menší dimenzi úlohy, například $n = 10$. Vliv volby parametru $\omega$ na počet iterací SOR je vykreslen na Obrázku 3.5 (přičemž pro lepší názornost byl maximální počet iterací omezen na 5000).
Ukazuje se, že pro algoritmus SOR je vhodné použít optimální hodnotu $\omega \approx 1.44$.
Počet iterací pro jednotlivé metody je pak vykreslen na Obrázku 3.6. ▲




Obr. 3.5: Vliv volby $\omega$ na počet iterací SOR - Příklad 3.22.




Obr. 3.6: Počet iterací algoritmů lineárních iteračních metod v závislosti na dimenzi úlohy - Příklad 3.22.

## 3.3 Gradientní metody

Uvažujme soustavu $Ax = b$, kde matice $A$ je symetrická pozitivně definitní,tj.

$$A \in \mathbb{R}^{n,n} : \quad A = A^T \quad \wedge \quad \forall v \in \mathbb{R}^n, v \neq 0 : v^T A v > 0 \ .$$

Jiným přístupem k iteračnímu řešení soustav lineárních rovnic je řešení minimalizační úlohy, jejíž výsledek je totožný s řešením soustavy lineárních rovnic. Za tímto účelem tedy uvažujme úlohu

$$\min_{x \in \mathbb{R}^n} f(x) \ , \tag{3.8}$$

kde

$$f(x) := \frac{1}{2}(Ax, x) - (b, x), \quad f : \mathbb{R}^n \to \mathbb{R} \ . \tag{3.9}$$

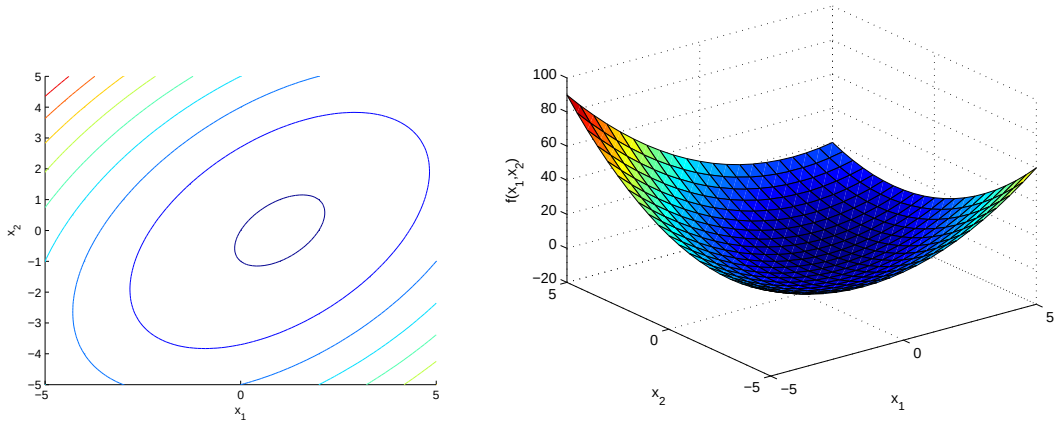
V předpisu tzv. *kvadratické funkce* (3.9) symbol $(.,.)$ představuje Eukleidovský skalární součin definovaný předpisem

$$\forall u, v \in \mathbb{R}^n : \quad (u, v) := u^T v = \sum_{i=1}^{n} u_i v_i \ . \tag{3.10}$$

**Příklad 3.23.** Nechť je dána kvadratická funkce předpisem (3.9) s hodnotami

$$n := 2, \quad A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \ .$$

Grafickou reprezentací této kvadratické funkce je paraboloid znázorněný na Obrázku 3.7.




Obr. 3.7: Grafická reprezentace dvoudimenzionální kvadratické funkce.

**Lemma 3.24.** *Nechť matice $A$ je symetrická pozitivně definitní. Řešení soustavy $Ax = b$ je ekvivalentní s minimalizační úlohou (3.8).*

*Důkaz.* Uvažujme přírůstek $x + \alpha v$ bodu $x$, kde $x, v \in \mathbb{R}^n, \alpha \in \mathbb{R}$.
Pak rozdíl funkčních hodnot v bodě s přírůstkem a v bodě bez přírůstku je roven

$$\begin{aligned} f(x+\alpha v) - f(x) &= \tfrac{1}{2}(A(x+\alpha v), x+\alpha v) - (b, x+\alpha v) - \tfrac{1}{2}(Ax, x) - (b, x) \\ &= \alpha(Ax, v) - \alpha(b, v) + \tfrac{1}{2}\alpha^2(Av, v) \\ &= \tfrac{1}{2}\alpha^2(Av, v) + \alpha(Ax - b, v) \end{aligned}$$

V úpravách jsme využili základních vlastností skalárního součinu viz skripta Lineární algebra [2].

Dále dokažme ekvivalenci tak, že postupně dokážeme jednotlivé implikace:

( $\Rightarrow$ ) Nejprve předpokládejme, že $\bar{x}$ je řešením soustavy lineárních rovnic, tedy

$$A\bar{x} = b \quad \Rightarrow \quad A\bar{x} - b = o \ .$$

Pak z pozitivní definitnosti matice $A$ plyne, že

$$f(\bar{x} + \alpha v) - f(\bar{x}) = \frac{1}{2}\alpha^2(Av, v) \geqq 0, \quad \forall \alpha \in \mathbb{R}, \forall v \in \mathbb{R}^n \ .$$

Úpravou pak

$$\Rightarrow f(\bar{x} + \alpha v) \geqq f(\bar{x}), \quad \forall \alpha \in \mathbb{R}, \forall v \in \mathbb{R}^n \ .$$

Z poslední nerovnosti tedy vyplývá, že ve všech směrech z bodu $\bar{x}$ nabývá funkce $f$ pouze větších nebo stejných hodnot. $\bar{x}$ je tedy minimem funkce $f$.

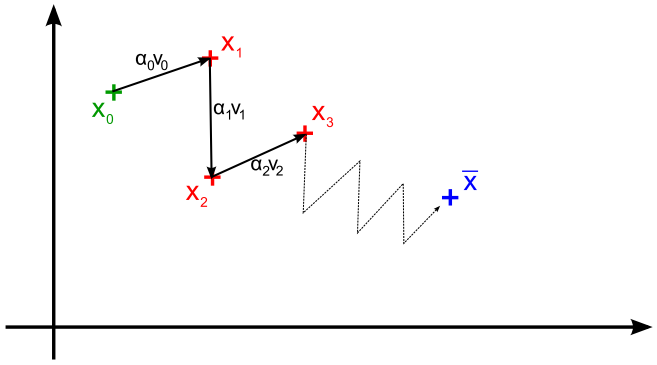
( $\Leftarrow$ ) V teorii minimalizace funkcí více proměnných (viz skripta [13]) je známo, že nutnou podmínkou existence extrému jsou nulové parciální derivace. V našem případě je to tedy požadavek na nulovost gradientu funkce $f$ tj. $\nabla f(\bar{x}) = o$. Jelikož $\nabla f(\bar{x}) = A\bar{x} - b$, dostáváme z této nutné podmínky, že $A\bar{x} - b = o$ a tedy, že $\bar{x}$ řeší soustavu lineárních rovnic $A\bar{x} = b$.

□

### 3.3.1 Metoda největšího spádu

Metoda největšího spádu patří do skupiny tzv. *line-search* minimalizačních metod. Následující iteraci získáme tak, že funkci v každém iteračním kroku minimalizujeme ve směru vektoru $v$ s délkou iteračního kroku $\alpha$.
Pro nalezení nové iterace používáme předpis

$$\begin{gathered} x_{k+1} = x_k + \alpha_k v_k \ . \\ x_k \in \mathbb{R}^n, v_k \in \mathbb{R}^n, \alpha_k \in \mathbb{R} \ . \end{gathered} \tag{3.11}$$




Obr. 3.8: Princip line-search metody - předpis (3.14).

- Při návrhu vhodného směru minimalizace funkce se jako logické jeví vydat se ve směru největšího spádu, tj. ve opačném směru než je gradient.
  $v_k$ volíme tak, aby $\frac{\mathrm{d}f(x_k)}{\mathrm{d}v_k}$ (derivace funkce $f$ ve směru $v_k$, viz [13]) byla co nejmenší

$$\frac{\mathrm{d}f(x_k)}{\mathrm{d}v_k} = (\nabla f(x_k), v_k) = \cos\varphi.\|g_k\|.\|v_k\| \ .$$

  Tato hodnota bude nejmenší při volbě $\varphi = \pi \Leftrightarrow \cos\varphi = -1$, tedy úhel mezi vektory $g_k$ a $v_k$ je roven $\pi$, tedy $v_k = -g_k$ (opačný gradient je skutečně vektorem největšího spádu, tj. směru ve kterém funkce nejvíc klesá).

- Předpokládejme tedy, že směr minimalizace $v_k$ je vhodně zvolen. Otázka zůstává, jak daleko je možné se v tomto směru pohybovat, než-li hodnota funkce začne opět stoupat. Za tím účelem musíme zvolit optimální délku kroku $\alpha_k$.
  Za tímto účelem si tedy zvolme funkci $\varphi(\alpha) := f(x_k + \alpha v_k)$ jedné reálné proměnné, kterou je délka kroku $\alpha$. Vhodnými úpravami dostáváme

$$\begin{aligned}
\varphi(\alpha) &= \tfrac{1}{2}(A(x_k+\alpha v_k), x_k+\alpha v_k) - (b, x_k+\alpha v_k) \\
&= \tfrac{1}{2}(Ax_k+\alpha Av_k, x_k+\alpha v_k) - (b,x_k) - (b,\alpha v_k) \\
&= \tfrac{1}{2}(Ax_k,x_k) + \tfrac{1}{2}(Ax_k,\alpha v_k) + \tfrac{1}{2}(\alpha Av_k, x_k) \\
&\quad +\tfrac{1}{2}(\alpha Av_k,\alpha v_k) - (b,x_k) - (b,\alpha v_k) \\
&= \tfrac{1}{2}(Ax_k,x_k) - b, x_k) + \tfrac{1}{2}(A\alpha v_k,\alpha v_k) + (\alpha Ax_k, v_k) - (b,\alpha v_k) \\
&= f(x_k) + \tfrac{1}{2}(A\alpha v_k,\alpha v_k) + \alpha((Ax_k,v_k) - (b,v_k)) \\
&= f(x_k) + \tfrac{1}{2}(A\alpha v_k,\alpha v_k) + \alpha(Ax_k - b, v_k) \\
&= f(x_k) + \tfrac{1}{2}\alpha^2(Av_k,v_k) + \alpha(g_k,v_k) \ .
\end{aligned} \tag{3.12}$$

  Hledáme co nejvhodnější $\alpha$, tj. takové, při kterém $\varphi(\alpha)$ je minimální. Za tímto účelem musíme minimalizovat funkci $\varphi(\alpha)$ jako funkci o jediné proměnné $\alpha$

$$\varphi'(\alpha) = \alpha(Av_k, v_k) + (g_k, v_k) = 0 \ .$$

V našem případě je stacionárním bodem

$$\alpha = -\frac{(g_k, v_k)}{(Av_k, v_k)} \ . \tag{3.13}$$

V průběhu funkce $\varphi(\alpha)$ nedochází k žádným inflexím, funkce je konvexní na celém $\mathbb{R}$, protože

$$\varphi''(\alpha) = (Av_k, v_k) > 0 \ ,$$

což plyne z pozitivní definitnosti matice $A$, $\alpha_k = -\frac{(g_k, v_k)}{(Av_k, v_k)}$ je tedy minimum funkce $\varphi$.

Po dosazení do předpisu (3.14) dostáváme

$$x_{k+1} = x_k + \alpha_k v_k = x_k - \frac{(g_k, v_k)}{(Av_k, v_k)} g_k = x_k - \frac{(g_k, g_k)}{(Ag_k, g_k)} g_k \ . \tag{3.14}$$

**Algoritmus 6** METODA NEJVĚTŠÍHO SPÁDU

**Input:** $x_0$, $A$, $b$, přesnost $\epsilon > 0$

$g_0 := Ax_0 - b$
$k := 0$
**while** $\|g_k\| \geq \epsilon \|b\|$ **do**
    $\alpha_k := -\|g_k\|^2 / (Ag_k, g_k)$
    $x_{k+1} := x_k + \alpha_k g_k$
    $g_{k+1} := Ax_{k+1} - b$
    $k := k + 1$
**end while**

**Output:** $x_k$

**Poznámka 3.25.**

$$g_{k+1} = Ax_{k+1} - b = A(x_k + \alpha_k g_k) - b = Ax_k + A\alpha_k g_k - b = g_k + \alpha_k A g_k$$

Tento způsob rezidua je výpočetně výhodnější, neboť jeho využitím nemusíme v algoritmu násobit $Ax_k$ ale k výpočtu gradientu můžeme využít $Ag_k$ (které jsme již spočetli při výpočtu $\alpha_k$).

**Lemma 3.26.** *Metoda největšího spádu konverguje pro libovolnou volbu $x_0 \in \mathbb{R}^n$ k řešení rovnice $Ax = b$. Navíc pro chyby dvou po sobě jdoucích iterací metody největšího spádu platí*

$$(Ae_{k+1}, e_{k+1}) \leqq \left(1 - \frac{1}{\kappa(A)}\right) (Ae_k, e_k) \ . \tag{3.15}$$

*Důkaz.* Označme chybu v $k$-té iteraci

$$\mathrm{e}_k = \bar{x} - x_k \ . \tag{3.16}$$

Cílem důkazu je ukázat, že chyba s postupem iterací klesá, tedy

$$\lim_{k\to\infty} \|\mathrm{e}_k\| = 0 \ .$$

Úpravou rovnice (3.16) získáme

$$x_k = \bar{x} - \mathrm{e}_k$$

a dosadíme do rozdílu funkčních hodnot

$$f(x_k) - f(\bar{x}) = f(\bar{x} - \mathrm{e}_k) - f(\bar{x}) = \frac{1}{2}(A\mathrm{e}_k, \mathrm{e}_k) + (A\bar{x} - b, \mathrm{e}_k) = \frac{1}{2}(A\mathrm{e}_k, \mathrm{e}_k) \ . \tag{3.17}$$

Využitím rovnice (3.17) lze upravit rozdíl funkčních hodnot dvou po sobě jdoucích iterací

$$\begin{array}{rcl} f(x_{k+1}) - f(x_k) & = & (f(x_{k+1}) - f(\bar{x})) - (f(x_k) - f(\bar{x})) \\ & = & \frac{1}{2}(A\mathrm{e}_{k+1}, \mathrm{e}_{k+1}) - \frac{1}{2}(A\mathrm{e}_k, \mathrm{e}_k) \ . \end{array} \tag{3.18}$$

Předpis pro rozdíl funkčních hodnot dvou po sobě jdoucích iterací lze získat také využitím předpisu metody (3.14)

$$\begin{array}{rcl} f(x_{k+1}) - f(x_k) & = & f(x_k - \alpha_k g_k) - f(x_k) \\ & = & \frac{1}{2}(A(x_k - \alpha_k g_k), x_k - \alpha_k g_k) - (b, x_k - \alpha_k g_k) - f(x_k) \\ & = & \frac{1}{2}(A\alpha_k g_k, \alpha_k g_k) - (Ax_k, \alpha_k g_k) + (b, \alpha_k g_k) \\ & = & \frac{1}{2}\alpha_k^2 (Ag_k, g_k) - \alpha_k (Ax_k - b, g_k) \\ & = & \frac{1}{2}\frac{(g_k, g_k)^2}{(Ag_k, g_k)^2}(Ag_k, g_k) - \frac{(g_k, g_k)}{(Ag_k, g_k)}(g_k, g_k) \\ & = & -\frac{1}{2}\frac{(g_k, g_k)^2}{(Ag_k, g_k)} \ . \end{array} \tag{3.19}$$

Porovnáním rovnic (3.18) a (3.19) získáme

$$\frac{1}{2}(A\mathrm{e}_{k+1}, \mathrm{e}_{k+1}) - \frac{1}{2}(A\mathrm{e}_k, \mathrm{e}_k) = -\frac{1}{2}\frac{(g_k, g_k)^2}{(Ag_k, g_k)} \ .$$

Jednoduchými úpravami dostáváme

$$\frac{(A\mathrm{e}_{k+1}, \mathrm{e}_{k+1})}{(A\mathrm{e}_k, \mathrm{e}_k)} = 1 - \frac{(g_k, g_k)^2}{(Ag_k, g_k).(A\mathrm{e}_k, \mathrm{e}_k)} \ . \tag{3.20}$$

Jelikož

$$-g_k = b - Ax_k = A\bar{x} - Ax_k = A(\bar{x} - x_k) = A\mathrm{e}_k \tag{3.21}$$

a užitím Lemma 3.6 o nerovnosti mezi maticovou a vektorovou normou lze tvrdit

$$\mathrm{e}_k = -A^{-1}g_k \quad \Rightarrow \quad \|\mathrm{e}_k\| \leqq \|A^{-1}\|.\|-g_k\| = \|A^{-1}\|.\|g_k\| \ . \tag{3.22}$$

Kombinací rovnosti (3.20) a nerovnosti (3.22) získáme Dosazením (3.21) a (3.22) do (3.20) dostáváme nerovnost

$$\frac{(Ae_{k+1}, e_{k+1})}{(Ae_k, e_k)} \leqq 1 - \frac{(g_k, g_k)^2}{\|A\|.\|g_k\|^2.\|A^{-1}\|.\|g_k\|^2} =\leqq 1 - \frac{1}{\|A\|.\|A^{-1}\|} =: q \ . \qquad (3.23)$$

Jelikož jsme v předchozích odvozeních používali Eukleidovskou vektorovou normu a jí odpovídající spektrální maticovou normu, dostáváme dále odhad

$$q = 1 - \frac{1}{\|A\|.\|A^{-1}\|} = 1 - \frac{1}{\kappa(A)} < 1 \quad \text{neboť } \kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \geqq 1 \ . \qquad (3.24)$$

Kombinací nerovností (3.23) a (3.24) dostáváme

$$\frac{(Ae_{k+1}, e_{k+1})}{(Ae_k, e_k)} \leqq q < 1 \ ,$$

dále úpravou

$$(Ae_{k+1}, e_{k+1}) \leqq q(Ae_k, e_k), \quad q \in \langle 0, 1) \ .$$

Odtud díky ekvivalenci norem platí, že

$$\lim_{k\to\infty} (Ae_k, e_k) = 0 \quad \Rightarrow \quad \lim_{k\to\infty} \|e_k\| = 0$$

Z předchozího důkazu je zřejmé, že rychlost poklesu chyby (rychlost konvergence) metody největšího spádu závisí na čísle pomíněnosti matice soustavy, konkrétně

$$(Ae_{k+1}, e_{k+1}) \leqq \left(1 - \frac{1}{\kappa(A)}\right) (Ae_k, e_k) \ .$$

□

**Poznámka 3.27.** Jelikož matice $A$ je symetrická a positivně definitní, lze skalární součin v odhadu chyby metody největšího spádu (3.15) značit jednoduše

$$(Ae_k, e_k) =: (e_k, e_k)_A = \|e_k\|_A^2 \ .$$

Příslušnou normu *indukovanou* tímto skalárním součinem označujeme pojmem *energetická*.

## Pro zájemce:

V algoritmu metody největšího spádu lze volit

$$\alpha_k = \frac{1}{\|A\|} \ .$$

Tato volba má obecně pomalejší konvergenci, nicméně konstantní volba koeficientů má svůj smysl - není nutno každý zvlášť počítat. Metoda se nazývá *Richardsonova metoda* a souvisí s odhadem posloupnosti koeficientů pomocí vlastních čísel, jak je vidět v podkapitole 4.2.1. Tyto koeficienty jsou totiž převrácené hodnoty Rayleighových podílů.

**Příklad 3.28.** Nalezněte průsečík přímek

$$\begin{aligned} p_1: \quad y &= 2x - 1 \\ p_2: \quad y &= \tfrac{1}{3}x - \tfrac{1}{3} \end{aligned}$$

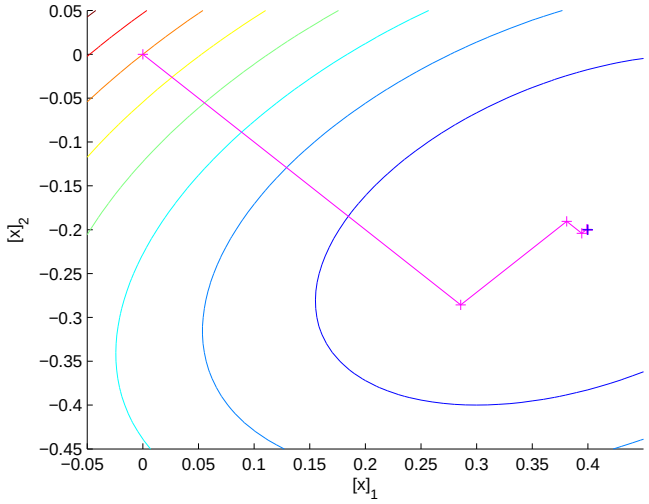
pomocí metody největšího spádu.

*Řešení.* Jistě není problém tuto úlohu vyřešit analyticky, nicméně na tomto příkladě bychom rádi ukázali vlastnosti největšího spádu. Jelikož se jedná pouze o dvě dimenze, lze postup iterací vykreslit.

Soustavu lze maticově zapsat ve tvaru $Ax = b$, kde

$$A = \begin{pmatrix} 1 & -1 \\ -1 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

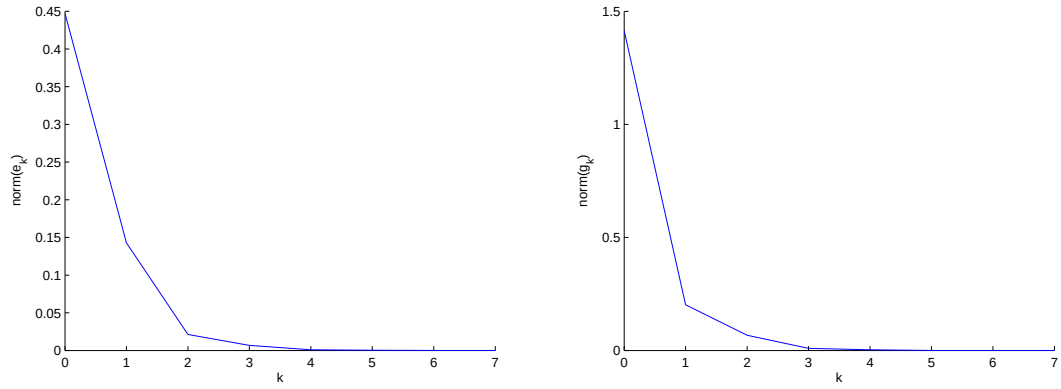
Pokud volíme například počáteční aproximaci $x_0 = (0, 0)^T$ a požadovanou přesnost $\epsilon = 10^{-4}$, algoritmus je ukončen po sedmi iteracích. Na Obrázku 3.9 je vykreslen postup iterací algoritmu vzhledem k vrstevnicím odpovídající kvadratické funkce,




Obr. 3.9: Postup iterací metody největšího spádu - Příklad 3.28.

na Obrázku 3.10 pak pokles normy chyby v jednotlivých iteracích a pokles normy residua.

▲




Obr. 3.10: Pokles normy chyby a normy residua při použití algoritmu největšího spádu - Příklad 3.28.

### 3.3.2 Metoda sdružených gradientů

Ačkoliv se může zdát, že nejrychlejší minimalizaci lze dosáhnout použitím vektorů největšího spádu, není tomu tak. Existuje i mnohem efektivnější volba využívající tzv. *sdružených směrů*. Právě těchto směrů využívá metoda *sdružených gradientů*. Nejprve si tedy zavedeme pojem *sdružené směry* a poté si ukážeme jak je můžeme využít pro vhodnou volbu minimalizačního směru.

#### Volba sdružených směrů

**Definice 3.29.** O neprázné množině nenulových vektorů $\{p_0, p_1, \ldots, p_m\}$ řekneme, že je množinou *sdružených (A-ortogonálních) vektorů*, pokud

$$\forall i, j = 0, 1, \ldots, m : \; i \neq j \;\Rightarrow\; p_i^T A p_j = 0 \; ,$$

kde matice $A$ je symetrická pozitivně definitní.

**Poznámka 3.30.** Množinu sdružených vektorů lze ekvivalentně definovat pomocí skalárního součinu

$$\forall i, j = 0, 1, \ldots, m : \; i \neq j \;\Rightarrow\; (Ap_i, p_j) = 0 \; .$$

**Lemma 3.31.** *Prvky množiny sdružených vektorů jsou lineárně nezávislé.*

*Důkaz.* Uvažujme lineární kombinaci vektorů dané množiny

$$\alpha_0 p_0 + \alpha_1 p_1 + \cdots + \alpha_m p_m = 0 \; . \tag{3.25}$$

Je nutno dokázat, že tato rovnice má právě jedno řešení, kterým je nulový vektor, tj. $\alpha_0 = \alpha_1 = \cdots = \alpha_m = 0$.
Pokud rovnici (3.25) skalárně přenásobíme zleva vektorem $Ap_k$ (kde $k$ je libovolný index $k \in \{0, 1, \ldots, m\}$), po využití vlastností skalárního součinu získáme $k$ rovnic

$$\alpha_0 (Ap_k, p_0) + \alpha_1 (Ap_k, p_1) + \cdots + \alpha_m (Ap_k, p_m) = 0 \; .$$

Jelikož vektory dané množiny jsou sdružené (tj. $\forall i \neq k : (Ap_k, p_i) = 0$) pak součet na levé straně lze zjednodušit a pro $k$-tou rovnici dostaneme

$$\alpha_k (Ap_k, p_k) = 0 \; . \tag{3.26}$$

Předpokládáme, že matice $A$ je pozitivně definitní (tj. $\forall k : p_k^T A p_k = (Ap_k, p_k) > 0$). Pak rovnost (3.26) je splněna, pouze pokud $\alpha_k = 0$.
Navíc $k$ jsme volili libovolně, tedy původní rovnice (3.25) má nulové řešení ($\forall k : \alpha_k = = 0$).
Proto vektory jsou skutečně lineárně nezávislé. □

Předpokládejme, že v $k$-tém kroce máme k dispozici množinu sdružených vektorů $\mathcal{K} := \{p_0, p_1, \ldots, p_k\}$ a následující iteraci získáme předpisem

$$x_{k+1} = x_k + \alpha_k p_k \ . \tag{3.27}$$

Koeficient $\alpha_k$ volíme tak, abychom minimalizovali funkci $f$ v daném směru co nejvíce. Tedy podobně jako v případě metody nejvetšího spádu zkonstruujeme funkci popisující funkci $f$ v závislosti na volbě parametru $\alpha$ a následně ji minimalizujeme (viz (3.12)). Obdobně jako v případě metody největšího spádu získáme koeficient

$$\alpha_k = -\frac{(p_k, g_k)}{(Ap_k, p_k)} \ . \tag{3.28}$$

Otázkou však zůstává, proč volit směr $p_k$ a nikoliv směr největšího poklesu $-g_k$ jako u metody největšího spádu (viz podkapitola 3.3.1).

**Lemma 3.32.** *Pro libovolnou počáteční aproximaci $x_0 \in \mathbb{R}^n$ posloupnost $\{x_k\}$ generována předpisem (3.27) dosáhne řešení $\bar{x} \in \mathbb{R}^n$ soustavy lineárních rovnic (3.2) po maximálně $n$ krocích.*

*Důkaz.* K dosáhnutí aproximace $x_{n-1}$ využijeme směry z množiny sdružených vektorů $\{p_0, p_1, \ldots, p_{n-1}\}$. Jelikož tato množina je lineárně nezávislá a má právě tolik prvků jako je dimenze uvažovaného prostoru $\mathbb{R}^n$, tak tvoří bázi vektorového prostoru $\mathbb{R}^n$. Tedy libovolný prvek lze vyjádřit jako lineární kombinaci vektorů množiny $\{p_0, p_1, \ldots, p_{n-1}\}$.
(Jinak řečeno - množina $n$ sdružených vektorů dimenze $n$ tvoří bázi vektorového prostoru $\mathbb{R}^n$).
Uvažujme rozdíl přesného řešení $\bar{x}$ a počáteční aproximace $x_0$. Tento rozdíl je také prvkem prostoru $\mathbb{R}^n$, tedy jej lze vyjádřit jako lineární kombinaci vektorů báze. Existují jednoznačné souřadnice v bázi $\sigma_0, \sigma_1, \ldots, \sigma_{n-1} \in \mathbb{R}$ takové, že

$$\bar{x} - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \cdots + \sigma_{n-1} p_{n-1} \ . \tag{3.29}$$

Postupným skalárním přenásobením rovnice (3.29) vektorem $Ap_k$ postupně pro všechna $k \in \{0, 1, \ldots, n-1\}$ dostáváme

$$(Ap_k, \bar{x} - x_0) = \sigma_0 (Ap_k, p_0) + \sigma_1 (Ap_k, p_1) + \cdots + \sigma_{n-1} (Ap_k, p_{n-1}) \ .$$

Využijme-li vlastnost sdružených vektorů (podobně jako v důkazu Věty 3.31), získáme

$$\sigma_k = \frac{(Ap_k, \bar{x} - x_0)}{(Ap_k, p_k)} \ . \tag{3.30}$$

Nyní ukážeme, že koeficienty $\sigma_k$ se shodují s koeficienty $\alpha_k$ generovanými předpisem (3.28).

Uvažujme iterační předpis (3.27). Postupným rekurentním dosazováním získáme předpis pro $k$-tou aproximaci

$$x_k = x_{k-1} + \alpha_{k-1} p_{k-1} = \cdots = x_0 + \sum_{i=0}^{k-1} \alpha_i p_i \ . \tag{3.31}$$

Pokud skalárně přenásobíme levou a pravou stranu vektorem $Ap_k$

$$(Ap_k, x_k) = (Ap_k, x_0) + \sum_{i=0}^{k-1} \alpha_i (Ap_k, p_i)$$

a využijeme vlastnosti sdružených vektorů (tentokráte však $k$ bylo pevně zvoleno již v předpisu (3.31)) dostáváme

$$(Ap_k, x_k) = (Ap_k, x_0) \ ,$$

a po úpravě

$$(Ap_k, x_k - x_0) = 0.$$

S využitím této znalosti lze upravit předpis (3.30)

$$\sigma_k = \frac{(Ap_k, \bar{x} - x_0) - 0}{(Ap_k, p_k)} = \frac{(Ap_k, \bar{x} - x_0) - (Ap_k, x_k - x_0)}{(Ap_k, p_k)} = \frac{(Ap_k, \bar{x} - x_k)}{(Ap_k, p_k)} \ .$$

Navíc $\bar{x}$ je řešením dané soustavy, tj. $A\bar{x} = b$, odkud

$$\sigma_k = \frac{(p_k, A\bar{x} - Ax_k)}{(Ap_k, p_k)} = \frac{(p_k, b - Ax_k)}{(Ap_k, p_k)} = -\frac{(p_k, g_k)}{(Ap_k, p_k)} \ . \tag{3.32}$$

Porovnáním předpisů (3.28) a (3.32) lze usoudit, že algoritmus přímo generuje souřadnice vektoru $\bar{x} - x_0$ v bázi $\{p_0, \ldots, p_{n-1}\}$ (viz (3.29)). Jelikož těchto koeficientů je $n$, algoritmus bude ukončen maximálně po $n$ krocích (další koeficienty nelze nalézt neboť neexistují).
Navíc pro libovolně zvolenou počáteční aproximaci $x_0$ je rozdíl $\bar{x} - x_0$ určen jednoznačně a proto algoritmus konverguje k hledanému řešení. □

**Poznámka 3.33.** Všechna tvrzení, která jsme dosud použili pro důkaz konvergence metody sdružených algoritmů jsou založeny na přesné reprezentaci reálných čísel. Při počítačové realizaci je však nutno počítat s nepřesnou aritmetikou. Pak navržená metoda algoritmus nebude ukončena po $n$ krocích a výpočetní postup bude muset být zastaven vhodnou ukončovací podmínkou.

### Gramův-Schmidtův $A$-ortogonalizační proces

Smysl využití sdružených vektorů pro generování nových aproximací je nám znám. Otázkou zůstává, jak tyto vektory generovat.

Překvapivě se dá k této úloze využít postup dobře známý z Lineární algebry, tzv. *Gramův-Schmidtův ortogonalizační proces* (viz [6]).
Jako první $A$-ortogonální vektor $p_0$ budeme volit gradient v bodě $x_0$. O tomto gradientu víme, že je nenulový (pokud by byl nulový, pak $g_0 = Ax_0 - b = 0$ a to znamená, že $x_0$ je řešení a algoritmus je ukončen).
Uvažujme $k$-tou iteraci. Nechť máme k dispozici množinu $A$-ortogonálních vektorů $\{p_0, p_1, \ldots, p_k\}$ a úkolem je nalézt $p_{k+1}$ splňující podmínku sdruženosti ($A$-ortogonality) ke stávajícím sdruženým vektorům, tj.

$$\forall i = 0, 1, \ldots, k : \; p_{k+1}^T A p_i = 0 \quad (\text{ekvivalentně } (Ap_{k+1}, p_i) = 0 \;) \; .$$

Pak lze vektor $p_{k+1}$ vyjádřit jako lineární kombinaci

$$p_{k+1} = g_{k+1} - \beta_{k+1,0} p_0 - \beta_{k+1,1} p_1 - \ldots - \beta_{k+1,k} p_k = g_{k+1} - \left( \sum_{j=0}^{k} \beta_{k+1,j} p_j \right) \; . \quad (3.33)$$

Dosazením do podmínky $A$-ortogonality

$$0 = (Ap_{k+1}, p_i) = (A \left( g_{k+1} - \sum_{j=0}^{k} \beta_{k+1,j} p_j \right), p_i) = (Ag_{k+1}, p_i) - \sum_{j=0}^{k} \beta_{k+1,j} (Ap_j, p_i) \; .$$

Využitím $A$-ortogonality stávajících vektorů $p_i, i = 0, \ldots, k$, se rovnice zjednoduší na tvar

$$0 = (Ag_{k+1}, p_i) - \beta_{k+1,i} (Ap_i, p_i) \; .$$

Odkud lze snadno získat předpis pro $i$-tý koeficient lineární kombinace

$$\beta_{k+1,i} = \frac{(Ag_{k+1}, p_i)}{(Ap_i, p_i)} = \frac{(Ap_i, g_{k+1})}{(Ap_i, p_i)} \; . \quad (3.34)$$

Poslední úpravu jsme mohli provést díky symetrii skalárního součinu
(tj. $\forall u, v \in \mathbb{R}^n : (u, v) = (v, u)$). Toto vyjádření je snažší z důvodu pozdější implementace, protože budeme násobit maticí $A$ pouze jednou.
Zpětným dosazením do lineární kombinace (3.33) získáme předpis pro výpočet $p_{k+1}$

$$p_{k+1} = g_{k+1} - \sum_{j=0}^{k} \frac{(Ap_j, g_{k+1})}{(Ap_j, p_j)} p_j \; . \quad (3.35)$$

Tento předpis je však výpočetně i paměťově náročný, neboť je nutno uchovávat v paměti všechny předchozí směry a pro výpočet směru nového je nutno v sumaci procházet tyto směry. Díky vlastnostem sdruženosti, je však možné situaci značně zjednodušší. Nejprve si však uveďme následující pomocná tvrzení.

**Lemma 3.34.** *(o ortogonalitě gradientů metody sdružených gradientů)*
*Nechť $\{g_0, \ldots, g_{n-1}\}$ je množina gradientů v příslušných bodech množiny aproximací $\{x_0, \ldots, x_{n-1}\}$, které jsou zkonstruovány předpisem (3.27).*
*Tato množina je ortogonální.*

*Důkaz.* Cílem důkazu je ukázat, že daná množina je ortogonální, tj.

$$\forall i,j \in \{0,1,\ldots,n-1\}: \quad (g_i,g_j) = \begin{cases} = 0 & \text{pro } i \neq j \ , \\ \neq 0 & \text{pro } i = j \ . \end{cases}$$

Z rovnice (3.33)

$$p_i = g_i - \sum_{l=0}^{i-1} \beta_{i,l} p_l$$

plyne

$$g_i = p_i + \sum_{l=0}^{i-1} \beta_{i,l} p_l \ . \tag{3.36}$$

(gradient je lineární kombinací aktuálního a předchozích sdružených směrů.)

Dále lze také jednoduše upravit (viz předpis CG metody (3.27))

$$\begin{aligned} x_{i+1} &= x_i + \alpha_i p_i \\ Ax_{i+1} &= A(x_i + \alpha_i p_i) \\ Ax_{i+1} &= Ax_i + \alpha_i Ap_i \\ Ax_{i+1} - b &= Ax_i - b + \alpha_i Ap_i \end{aligned}$$

a jelikož $g_i = Ax_i - b$, lze psát

$$g_{i+1} = g_i + \alpha_i Ap_i \ . \tag{3.37}$$

Pro $i = j$ dostáváme $(g_i, g_i) = \|g_i\|^2 > 0$ (předpokládáme nenulová residua, v opačném případě by byl algoritmus předčasně ukončen).
Bez újmy na obecnosti předpokládejme, že $i > j$ (případně lze $i, j$ zaměnit, protože $(g_i, g_j) = (g_j, g_i)$).
Dále indukcí dokažme, že

$$\forall i,j = 0,1,\ldots,n-1, \ i > j: \ (g_i,g_j) = 0 \ . \tag{3.38}$$

- Nechť $i = 1$, pak je-li $i > j \geqq 0$, tak $j = 0$.
  Upravme dosazením předpisu (3.37)

$$\begin{aligned} (g_i,g_j) &= (g_1,g_0) = (g_0 + \alpha_0 Ap_0, g_0) = (g_0,g_0) + \alpha_0 (Ap_0,g_0) \\ &= (g_0,g_0) - \tfrac{(g_0,p_0)}{(Ap_0,p_0)} (Ap_0,g_0) = 0 \ , \end{aligned}$$

  což platí díky volbě $p_0 = g_0$.

- *Indukční krok* - dokažme implikaci $\forall i,j = 0,1,\ldots,n-1, \ i > j$ :

$$(g_i,g_j) = 0 \quad \Rightarrow \quad (g_{i+1},g_j) = 0 \ . \tag{3.39}$$

Dosadíme-li předpis (3.37) do (3.39), pak dostáváme

$$(g_{i+1}, g_j) = (g_i + \alpha_i A p_i, g_j) = (g_i, g_j) + \alpha_i (A p_i, g_j) = \alpha_i (A p_i, g_j) \ ,$$

kde jsme využili indukčního předpokladu $(g_i, g_j) = 0$. Jelikož pravděpodobně $\alpha_i \neq$ $\neq 0$, je nutno ukázat, že pro $i > j$ platí $(A p_i, g_j) = 0$.
Využijeme-li předpis (3.36) dostaneme díky A-ortogonalitě vektorů $p$

$$(A p_i, g_j) = \left( A p_i, p_j + \sum_{l=0}^{j-1} \beta_{j,l} p_l \right) = (A p_i, p_j) + \sum_{l=0}^{j-1} \beta_{j,l} \, (A p_i, p_l) = 0 \ .$$

□

**Lemma 3.35.** *Nechť je dána množina gradientů* $\{g_0, g_1, \ldots, g_{k+1}\}$ *a množina sdružených vektorů* $\{p_0, p_1, \ldots, p_k\}$ *zkonstruovaná z množiny gradientů A-ortogonalizačním procesem* (3.35).
*Pak platí*

$$(A p_j, g_{k+1}) = \begin{cases} = 0 & \text{pro } j < k \ , \\ \neq 0 & \text{pro } j = k \ . \end{cases}$$

*Důkaz.* Nejdříve si všimněme, že předpis (3.37) lze jednoduše upravit na tvar

$$A p_k = \frac{1}{\alpha_k} (g_{k+1} - g_k) \ . \tag{3.40}$$

Pak dosazením předpisu (3.40) do levé strany dokazovaného tvrzení získáme

$$(A p_j, g_{k+1}) = (g_{k+1}, A p_j) = \left( g_{k+1}, \frac{1}{\alpha_j} (g_{j+1} - g_j) \right) = \frac{1}{\alpha_j} (g_{k+1}, g_{j+1}) - \frac{1}{\alpha_j} (g_{k+1}, g_j) \ .$$

Při volbě $j < k$ je díky ortogonalitě gradientů (Věta 3.34) tento výraz nulový a naopak při volbě $j = k$ je tento výraz roven

$$(A p_j, g_{k+1}) = \frac{1}{\alpha_k} (g_{k+1}, g_{k+1}) \ .$$

Jelikož předpokládáme nenulové gradienty, tak je poslední výraz nenulový. □

**Poznámka 3.36.** Přímým důsledkem předchozí věty 3.35 je

$$\beta_{k+1,i} = \frac{(A p_i, g_{k+1})}{(A p_i, p_i)} = \begin{cases} = 0 & \text{pro } i < k \ , \\ \neq 0 & \text{pro } i = k \ . \end{cases}$$

Z toho vyplývá, že ze všech koeficientů $\beta_{k+1,i}$ je nenulový jediný koeficient

$$\beta_{k+1} := \beta_{k+1,k} = \frac{(A p_k, g_{k+1})}{(A p_k, p_k)} \ . \tag{3.41}$$

Využitím předchozí věty lze předpis (3.35) zjednodušit na tvar

$$p_{k+1} = g_{k+1} - \beta_{k+1} p_k = g_{k+1} - \frac{(A p_k, g_{k+1})}{(A p_k, p_k)} p_k \ , \tag{3.42}$$

což znamená, že při výpočtu nového -sdruženého směru vystačíme pouze s jediným sdruženým směrem z předchozí iterace.

### Algoritmus CG a jeho vlastnosti

Zhrneme-li všechna předchozí pozorování, můžeme sestavit algoritmus metody sdružených gradientů.

**Algoritmus 7** METODA SDRUŽENÝCH GRADIENTŮ

**Input:** $x_0$, $A$, $b$, přesnost $\epsilon > 0$

$g_0 := Ax_0 - b$
$p_0 := g_0$
$k := 0$
**while** $\|g_k\| \geq \epsilon\|b\|$ **do**
  $\alpha_k := -(p_k, g_k)/(Ap_k, p_k)$
  $x_{k+1} := x_k + \alpha_k p_k$
  $g_{k+1} := Ax_{k+1} - b$
  $\beta_{k+1} := (Ap_k, g_{k+1})/(Ap_k, p_k)$
  $p_{k+1} := g_{k+1} - \beta_{k+1} p_k$
  $k := k + 1$
**end while**

**Output:** $x_k$

**Lemma 3.37.** *(pomocné) Nechť $\{x_0, x_1, x_2, \dots\}$ jsou postupně aproximace generované metodou sdružených gradientů, nechť $\{g_0, g_1, g_2, \dots\}$ jsou gradienty v těchto aproximacích a $\{p_0, p_1, p_2, \dots\}$ jsou příslušné sdružené vektory.*
*Pak platí následující tvzení*

$$\begin{array}{lllcl} \text{a)} & \forall k \geqq 0: & (Ap_k, p_k) & = & (Ap_k, g_k)\ , \\ \text{b)} & \forall k \geqq 0: & (Ap_k, g_k) & \leqq & (Ag_k, g_k)\ , \\ \text{c)} & \forall i > j \geqq 0: & (g_i, p_j) & = & 0\ , \\ \text{d)} & \forall k \geqq 0: & (g_k, g_k) & = & (g_k, p_k)\ , \\ \text{e)} & \forall a, b \in \mathbb{R}: & ab & \leqq & \frac{1}{4}(a+b)^2\ . \end{array} \tag{3.43}$$

*Důkaz.* a) Připomeňme předpis (3.42)

$$p_k = g_k - \beta_k p_{k-1}\ . \tag{3.44}$$

Pak dosazením a úpravou získáme

$$\begin{array}{rcl} (Ap_k, p_k) & = & (Ap_k, g_k - \beta_k p_{k-1}) \\ & = & (Ap_k, g_k) - \beta_k (Ap_k, p_{k-1}) \\ & = & (Ap_k, g_k)\ . \end{array}$$

b) Dosadíme-li rovnici (3.44) do levé strany dokazované nerovnosti, získáme

$$(Ap_k, g_k) = (Ag_k, g_k) - \beta_k (Ap_{k-1}, g_k)\ .$$

Dále dostadíme za $\beta$ předpis (3.34). Získáme

$$\begin{aligned}(Ap_k, g_k) &= (Ag_k, g_k) - \tfrac{(Ag_k, p_{k-1})}{(Ap_{k-1}, p_{k-1})}(Ap_{k-1}, g_k) \\ &= (Ag_k, g_k) - \tfrac{(Ag_k, p_{k-1})^2}{(Ap_{k-1}, p_{k-1})} \ .\end{aligned}$$

Zlomek, který odečítáme je však vždy nezáporný, protože

| | | |
|---|---|---|
| čitatel: | $(Ag_k, p_{k-1})^2 \geqq 0$ | (druhá mocnina) |
| jmenovatel: | $(Ap_{k-1}, p_{k-1}) = p_{k-1}^T A p_{k-1} \geqq 0$ | ($A$ je pozitivně definitní) , |

a tedy

$$(Ap_k, g_k) = (Ag_k, g_k) - \frac{(Ag_k, p_{k-1})^2}{(Ap_{k-1}, p_{k-1})} \leqq (Ag_k, g_k)$$

c) Rekurentním dosazováním do předpisu (3.37) dostaneme

$$g_i = g_{i-1} - \alpha_{i-1} p_{i-1} = \cdots = g_0 + \sum_{k=0}^{i-1} \alpha_k A p_k \ . \tag{3.45}$$

Dosadíme-li tento předpis do levé strany dokazované rovnice, získáme

$$(g_i, p_j) = (g_0 + \sum_{k=0}^{i-1} \alpha_k A p_k, p_j) = (g_0, p_j) + \sum_{k=0}^{i-1} \alpha_k (Ap_k, p_j) \ .$$

Dle předpokladu je $i > j$, tedy sčítance v sumě jsou všechny nulové krom $k = j$ (dle Věty 3.35), takže

$$(g_i, p_j) = (g_0, p_j) + \alpha_j (Ap_j, p_j) \ .$$

Následným dosazením předpisu pro $\alpha_k$ (viz (3.28)) dostaneme

$$(g_i, p_j) = (g_0, p_j) - \frac{(g_j, p_j)}{(Ap_j, p_j)}(Ap_j, p_j) = (g_0, p_j) - (p_j, g_j) \ .$$

Dosazením předpisu (3.45) do posledního výrazu dostáváme

$$(g_i, p_j) = (g_0, p_j) - \left( (p_j, g_0) - \sum_{k=0}^{j-1} \alpha_k (Ap_k, p_j) \right)$$

a jelikož $p_k$ jsou $A$-ortogonální, získáme

$$(g_k, p_j) = (g_0, p_j) - (g_0, p_j) = 0$$

d) Do levé stany dokazovaného tvrzení dosaďme předpis (3.44), získáme

$$(g_k, p_k) = (g_k, g_k - \beta_k p_{k-1}) = (g_k, g_k) - \beta_k (g_k, p_{k-1}) \ .$$

Užitím pomocného tvrzení (3.43) c) získáme

$$(g_k, p_k) = (g_k, g_k) \ .$$

e) Stačí upravit danou nerovnici

$$\begin{array}{rcl} ab & \leqq & \frac{1}{4}(a+b)^2 \\ 4ab & \leqq & (a+b)^2 \\ 4ab & \leqq & a^2+2ab+b^2 \\ 0 & \leqq & a^2-2ab+b^2 \\ 0 & \leqq & (a\ -b)^2\ , \end{array}$$

což platí pro libovolnou volbu $a, b \in \mathbb{R}$.

□

**Lemma 3.38.** *V algoritmu metody sdružených gradientů můžeme nahradit*

$$\alpha_k = \frac{\|g_k\|^2}{p_k^T A p_k}\ , \qquad \beta_{k+1} = -\frac{\|g_{k+1}\|^2}{\|g_k\|^2}\ .$$

*Důkaz.* Upravme přímo dané koeficienty

- Původní předpis pro $\alpha_k$ (3.28) lze upravit využím pomocného Lemma 3.37 d)

$$\alpha_k = -\frac{(p_k, g_k)}{(Ap_k, p_k)} = -\frac{(g_k, g_k)}{(Ap_k, p_k)} = -\frac{\|g_k\|^2}{(Ap_k, p_k)}\ .$$

- Budeme vycházet z předpisu (3.41)

$$\beta_{k+1} = \frac{(Ap_k, g_{k+1})}{(Ap_k, p_k)}$$

a postupně jej budeme upravovat. Nejdříve využijeme pomocné tvrzení (3.43) a) a následně dosadíme rovnici (3.40) a upravíme

$$\frac{(Ap_k, g_{k+1})}{(Ap_k, g_k)} = \frac{(\frac{1}{\alpha_k}(g_{k+1}-g_k), g_{k+1})}{(\frac{1}{\alpha_k}(g_{k+1}-g_k), g_k)} = \frac{(g_{k+1}, g_{k+1}) - (g_k, g_{k+1})}{(g_{k+1}, g_k) - (g_k, g_k)}\ ,$$

dále pak pomocí tvrzení (3.38) lze zjednodušit na tvar

$$\frac{(g_{k+1}, g_{k+1}) - (g_k, g_{k+1})}{(g_{k+1}, g_k) - (g_k, g_k)} = -\frac{(g_{k+1}, g_{k+1})}{(g_k, g_k)} = -\frac{\|g_{k+1}\|^2}{\|g_k\|^2}\ .$$

□

**Poznámka 3.39.** 1. Pro řešení $\bar{x}$ soustavy $Ax = b$ platí

$$\begin{array}{rcl} \bar{x} \in \mathcal{K}_n & = & \langle p_0, \ldots, p_{n-1} \rangle\ , \\ & = & \langle g_0, \ldots, g_{n-1} \rangle\ , \\ & = & \langle g_0, Ag_0, \ldots, A^{n-1} g_0 \rangle\ , \end{array}$$

kde $\langle v_0, \ldots, v_{n-1} \rangle$ značí lineární obal množiny vektorů $\{v_0, \ldots v_{n-1}\}$, viz [2]. Prostory $\mathcal{K}_n$ nazýváme *Krylovovy prostory.*

2. Metoda sdružených gradientů musí v přesné aritmetice nalézt řešení po $n$ krocích. Může však nastat situace, že řešení

$$\bar{x} \in \langle g_0, Ag_0, \ldots, A^{m-1}g_0 \rangle \ .$$

   Pak bude $g_{m-1} = 0$ a řešení je nalezeno po $m$ krocích.

3. Na metodu sdružených gradientů můžeme taktéž pohlížet jako na iterační metodu. Hledání řešení ukončíme po splnění určité přesnosti (viz algoritmus).

**Lemma 3.40.** *Pro chyby dvou po sobě jdoucích iterací metody sdružených gradientů platí*

$$(Ae_{k+1}, e_{k+1}) \leqq \left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^2 (Ae_k, e_k) \ . \tag{3.46}$$

*Důkaz.* Podobně jako u metody největšího spádu platí

$$\frac{(Ae_{k+1}, e_{k+1})}{(Ae_k, e_k)} = 1 - \frac{(g_k, p_k)^2}{(Ap_k, p_k).(Ae_k, e_k)} \ . \tag{3.47}$$

K tomuto předpisu lze dojít stejným způsobem jako v důkazu věty 3.26.
Uvažujme zlomek na pravé straně rovnice (3.47). Dle pomocného tvrzení 3.37 *d*) a rovnosti (3.21) (která platí i pro sdružené gradienty) lze psát

$$\frac{(g_k, p_k)^2}{(Ap_k, p_k).(Ae_k, e_k)} = \frac{(g_k, g_k)^2}{(Ap_k, p_k).(Ae_k, e_k)} = \frac{(g_k, g_k)^2}{(Ap_k, p_k).(A^{-1}g_k, g_k)}$$

a dle nerovnosti 3.37 *b*) platí

$$\frac{(g_k, g_k)^2}{(Ap_k, p_k).(A^{-1}g_k, g_k)} \geqq \frac{(g_k, g_k)^2}{(Ag_k, g_k).(A^{-1}g_k, g_k)} \ . \tag{3.48}$$

Jelikož matice soustavy $A$ je symetrická a pozitivně definitní, existují její lineárně nezávislé a ortonormální vlastní vektory $u_1, \ldots u_n$. Protože jich je právě $n$, tak tvoří bázi vektorového prostoru $\mathbb{R}^n$ a každý prvek tohto prostoru lze vyjádřit jako lineární kombinaci těchto vektorů. Tedy i pro vektor $g_k$ existují jednoznačné souřadnice $\xi_1, \ldots, \xi_n \in \mathbb{R}$ v bázi $U = \{u_1, \ldots, u_n\}$ takové, že

$$g_k = \xi_1 u_1 + \cdots + \xi_n u_n = \sum_{i=1}^{n} \xi_i u_i \ .$$

Dosazením do pravé strany nerovnice (3.48)

$$\frac{(g_k, g_k)^2}{(Ag_k, g_k).(A^{-1}g_k, g_k)} = \frac{\left(\sum_{i=1}^{n} \xi_i^2\right)^2}{\left(\sum_{i=1}^{n} \lambda_i \xi_i^2\right) \cdot \left(\sum_{i=1}^{n} \frac{1}{\lambda_i} \xi_i^2\right)} \tag{3.49}$$

kde $\lambda_i > 0$ jsou vlastní čísla pozitivně definitní matice $A$ (připomeňme $Au_i = \lambda_i u_i$, navíc $u_i$ jsou ortonormální).
Našim úkolem bude tento zlomek dále odhadnout.
Uvažujme nyní jmenovatel na pravé straně nerovnice (3.49). Pro konkrétní $\lambda_i$ ze spektra matice $A$, tj.

$$\lambda_i \in \sigma(A) := \{\lambda_1, \ldots, \lambda_n\} \subset \langle \lambda_{\min}, \lambda_{\max} \rangle \ ,$$

se jedná o polynom tvaru

$$A_i \lambda_i + B_i \frac{1}{\lambda_i} + C_i, \quad \text{kde } A_i, B_i, C_i > 0 \tag{3.50}$$

a konstanty $A_i, B_i, C_i$ jsou tvořeny kombinací koeficientů $\xi_1^2, \ldots, \xi_n^2$ a zbývajících vlastních čísel (prvky množiny $\sigma(A) \backslash \{\lambda_i\}$). Tato funkce však nemá lokální maximum pro kladná $\lambda_i$, neboť druhá derivace funkce $\Psi_i$, definované jako

$$\Psi_i(\lambda) := A_i \lambda + B_i \frac{1}{\lambda} + C_i \ ,$$

je kladná

$$\begin{array}{rcl} \Psi_i'(\lambda) & = & A_i - B_i \frac{1}{\lambda^2} \\ \Psi_i''(\lambda) & = & 2B_i \frac{1}{\lambda^3} \\ \lambda > 0 & \Rightarrow & \Psi_i''(\lambda) > 0 \ . \end{array}$$

Maximum tedy nastává v krajních bodech spektra, tedy v $\lambda_{\min}$ nebo v $\lambda_{\max}$.
Lze tedy tvrdit, že pro libovolné $\lambda_i \in \Lambda(A)$ a polynom tvaru (3.50) platí

$$\exists \hat{\lambda}_i \in \{\lambda_{\min}, \lambda_{\max}\} : A_i \lambda_i + B_i \frac{1}{\lambda_i} + C_i \leqq A_i \hat{\lambda}_i + B_i \frac{1}{\hat{\lambda}_i} + C_i \ .$$

Nahradíme-li *vhodně*[1] všechna vlastní čísla nejmenším nebo největším vlastním číslem, získáme horní odhady všech polynomů tvaru (3.50).
Pak pro jmenovatel na pravé straně nerovnice (3.49) platí

$$\left( \sum_{i=1}^{n} \lambda_i \xi_i^2 \right) \cdot \left( \sum_{i=1}^{n} \frac{1}{\lambda_i} \xi_i^2 \right) \leqq \left( \sum_{i=1}^{n} \hat{\lambda}_i \xi_i^2 \right) \cdot \left( \sum_{i=1}^{n} \frac{1}{\hat{\lambda}_i} \xi_i^2 \right) \ . \tag{3.51}$$

Rozdělme

$$\sum_{i=1}^{n} \xi_i^2 = S_1 + S_2 \ ,$$

kde

- $S_1$ je součet druhých mocnin koeficientů $\xi_i$ příslušných k $\hat{\lambda}_i = \lambda_{\min}$
- $S_2$ je součet druhých mocnin koeficientů $\xi_i$ příslušných k $\hat{\lambda}_i = \lambda_{\max}$.

[1] Toto nahrazení nebudeme prakticky realizovat - pro nás je důležité pouze to, že existuje.

Po tomto označení lze upravit pravou stranu nerovnice (3.51) a pokud využijeme šikovnou úpravu na čtverec, lze jednoduše získat

$$\begin{aligned}\left(\sum_{i=1}^{n}\hat{\lambda}_i\xi_i^2\right).\left(\sum_{i=1}^{n}\tfrac{1}{\hat{\lambda}_i}\xi_i^2\right) &= (S_1\lambda_{\min}+S_2\lambda_{\max})\left(\tfrac{S_1}{\lambda_{\min}}+\tfrac{S_2}{\lambda_{\max}}\right)\\ &= S_1^2+S_2^2+S_1S_2\left(\tfrac{\lambda_{\max}}{\lambda_{\min}}+\tfrac{\lambda_{\min}}{\lambda_{\max}}\right)\\ &= (S_1+S_2)^2+S_1S_2\left(\sqrt{\tfrac{\lambda_{\max}}{\lambda_{\min}}}-\sqrt{\tfrac{\lambda_{\min}}{\lambda_{\max}}}\right)^2 .\end{aligned}$$

Využijeme-li navíc pomocné tvrzení 3.37 e), lze dále odhadnout

$$(S_1+S_2)^2+S_1S_2\left(\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}-\sqrt{\frac{\lambda_{\min}}{\lambda_{\max}}}\right)^2 \leqq (S_1+S_2)^2\frac{1}{4}\left(\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}+\sqrt{\frac{\lambda_{\min}}{\lambda_{\max}}}\right)^2 .$$

Dosazením odvozených odhadů do původní rovnice (3.47) a úpravou získáme

$$\frac{(Ae_{k+1},e_{k+1})}{(Ae_k,e_k)} \leqq 1-\frac{1}{\frac{1}{4}\left(\sqrt{\kappa(A)}+\sqrt{\kappa(A)^{-1}}\right)^2} = \left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^2 .$$

Tedy skutečně pro chyby dvou po sobě jdoucích iterací metody sdružených gradientů platí

$$(Ae_{k+1},e_{k+1}) \leqq \left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^2 (Ae_k,e_k) .$$

□

**Poznámka 3.41.** Často se nerovnost (3.46) udává s energetickou normou, viz Poznámka 3.27.

### 3.3.3 Předpodmínění

Z předchozích kapitol jasně vyplývá, že konvergence gradientních metod závisí na čísle podmíněnosti matice soustavy. Čím menší je číslo podmíněnosti, tím rychleji metoda konverguje k předem zadané chybě. Pokud chceme zajistit rychlou konvergenci gradientních metod, musíme se pokusit nahradit původní soustavu lineárních rovnic $Ax=b$ jinou ekvivalentní soustavou, jejíž číslo podmíněnosti bude nižší.
Proces nahrazení soustavy $Ax=b$ ekvivalentní soustavou $\tilde{A}x=\tilde{b}$, kde číslo podmíněnosti nové matice soustavy $\tilde{A}$ je menší než číslo podmíněnosti matice $A$, nazýváme *předpodmínění*.
Zvolíme-li regulární matici $C$, která je „blízká“ matici $A$ v tom smyslu, že $C^{-1}A$ má číslo podmíněnosti ideálně blízké 1, pak tzv. *předpodmíněnou soustavu* získáme přenásobením obou stran původní soustavy maticí $C$.

$$\begin{aligned} A\,x &= b\\ C^{-1}Ax &= C^{-1}b\\ \tilde{A}x &= \tilde{b}, \quad \tilde{A}\overset{\text{def}}{=}C^{-1}A, \tilde{b}\overset{\text{def}}{=}C^{-1}b .\end{aligned}$$

Výsledná matice předpodmíněné soustavy $\tilde{A} := C^{-1}A$ však nemusí být symetrická a ani positivně definitní. V takovém případě by nešlo na předpodmíňenou soustavu použít gradientních metod.
Abychom se této situaci vyhli, volíme za matici předpodmínění matici ve tvaru $C := C^{1/2}C^{T/2}$, která je symetrická a positivně definitní.
Pak původní soustavu $Ax = b$ přenásobíme zleva $C^{-1/2}$ a provedeme transformaci vektoru řešení $\tilde{x} = C^{T/2}x$.
Tímto dostáváme novou předpodmíněnou soustavu

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

kde

$$\tilde{A} := C^{-1/2}AC^{-T/2} \ , \quad \tilde{x} := C^{T/2}x \ , \quad \tilde{b} := C^{-1/2}b \ .$$

Pro metodu sdružených gradientů pak platí

$$\begin{aligned}
\tilde{g}_k &= C^{-1/2}g_k \\
s_k &= C^{-1/2}\tilde{g}_k \\
v_k &= C^{-1/2}\bar{v}_k \\
\tilde{\beta}_{k-1} &= -\frac{(As_k, v_{k-1})}{(Av_{k-1}, v_{k-1})} \\
v_k &= s_k + \tilde{\beta}_{k-1} \\
\tilde{\alpha}_k &= -\frac{(\tilde{g}_k, \tilde{v}_k)}{(\tilde{A}\tilde{v}_k, \tilde{v}_k)} = -\frac{(C^{-1/2}g_k, C^{\frac{1}{2}}v_k)}{(C^{-\frac{1}{2}}AC^{-\frac{1}{2}}C^{\frac{1}{2}}v_k, C^{\frac{1}{2}}v_k)} = -\frac{(g_k, v_k)}{(Av_k, v_k)} \\
x_{k+1} &= x_k + \tilde{\alpha}_k v_k
\end{aligned}$$

Je tedy vidět, že se algoritmus liší pouze ve výpočtu $\beta_{k-1}$ a konstrukce $v_k$.
Je nutné vypočítat vektor $s_k$ ze soustavy $Cs_k = r_k$. Proto je taktéž nutné, aby tato soustava byla snadno řešitelná.

**Algoritmus 8** METODA SDRUŽENÝCH GRADIENTŮ S PŘEDPODMÍNĚNÍM

**Input:** $x_0$, $A$, $b$, $C$, přesnost $\epsilon > 0$

$g_0 := Ax_0 - b$
$s_0 := C^{-1}g_0$
$p_0 := s_0$
**while** $\|g_k\| \geq \epsilon\|b\|$ **do**
    $\alpha_k := -(s_k, g_k)/(Ap_k, p_k)$
    $x_{k+1} := x_k + \alpha_k p_k$
    $g_{k+1} := g + \alpha_k Ap_k$
    $s_{k+1} := C^{-1}g_{k+1}$
    $\beta_{k+1} := (s_{k+1}, g_{k+1})/(s_k, g_k)$
    $p_{k+1} := s_{k+1} + \beta_{k+1}p_k$
    $k := k + 1$
**end while**

**Output:** $x_k$

### Jacobiho diagonální předpodmínění

Jednou z nejjednodušších metod předpodmínění je použití diagonály matice $A$. Matice předpodmínění pak má tvar

$$\begin{aligned} C &= \text{diag}\,(A) \\ C^{1/2} &= C^{T/2} = \text{diag}\,(A)^{1/2} = \text{diag}\,(\sqrt{a_{ii}})\ . \end{aligned}$$

Tento typ předpodmínění není příliš robustní a největšího efektu dosáhne u soustav, kde v matici soustavy $A$ výrazně dominují diagonální prvky, tj. jsou výrazně větší než prvky mimodiagonální.

### Symmetric successive over-relaxation method (SSOR)

Významnějšího efektu předpodmínění u obecnějších typů matic je možné dosáhnout využitím tzv. SSOR metody.
Matice předpodmínění se konstruuje následujícím způsobem

$$\begin{aligned} A &= D + L + L^T \\ C_\omega &= \tfrac{1}{2-\omega}(\tfrac{1}{\omega}D + L)(\tfrac{1}{\omega}D)^{-1}(\tfrac{1}{\omega}D + L)^T, \quad 0 < \omega < 2 \\ C_\omega^{\frac{1}{2}} &= \sqrt{\tfrac{1}{2-\omega}}(\tfrac{1}{\omega}D + L)(\tfrac{1}{\omega}D)^{-1/2}\ . \end{aligned}$$

Velmi efektivní implementace této metody je využití tzv. *Eisenstatova triku*. Výpočetní náročnost metody předpodmíněných gradientů se pak prakticky nezvyšuje oproti použití standartní metody sdružených gradientů.

### Neúplná Choleského faktorizace

Nejrobustnějším typem předpodmínění, se kterým se seznámíme je tzv. *neúplná Choleského faktorizace*. Jak je patrné z jejího názvu, vychází z *Choleského rozkladu*, se kterým jste se seznámili v předmětu Lineární algebra [2].
Nejprve si tedy zopakujme co se rozumí pod pojmem Choleského rozklad.

**Věta 3.42.** *(Choleského rozklad)*
*Nechť $A \in \mathbb{R}^{n,n}$ je symetrická positivně definitní matice. Pak existuje jednoznačně daná dolní trojuhelníková matice $L \in \mathbb{R}^{n,n}$ s kladnými diagonálními prvky taková, že*

$$A = LL^T\ . \tag{3.52}$$

*Důkaz.* Viz [19]. □

Myšlenka neúplné Choleského faktorizace spočívá v nalezení matice $\tilde{L}$, která je *blízká* matici $L$. Pak matice předpodmínění pomocí neúplné Choleského faktorizace má tvar

$$C = \tilde{L}\tilde{L}^T\ , \quad C^{1/2} = \tilde{L}\ .$$

Očividně čím více se bude matice $\tilde{L}$ blížit matici $L$ z plného Choleského rozkladu, tím blíže bude matice $\tilde{A}$ k jednotkové matici, která má číslo podmíněnosti 1.
Speciálně pokud bychom vzali $\tilde{L} = L$, dostáváme ideální předpodmínění. Je však jasné, že použitím metody předpodmíněných sdružených gradientů bychom pak nemohli překonat co do výpočetní náročnosti Cholekého rozklad. Takový typ předpodmínění by proto nedával smysl.
Vraťme se tedy ke konstrukci matice $\tilde{L}$. V principu existují 2 základní přístupy k sestavení této matice. V obou případech je konstrukce založena na sestavování matice $L$ jako v případě standardního Choleského rozkladu (viz [2] a [1]), s tím, že některé prvky matice $L$ nebudeme do výsledného faktoru zapisovat. Podívejme se tedy na tyto dva přístupy.

1. *Neúplný rozklad podle vzorku*
   Pokud je některý prvek $[A]_{i,j}$ nulový, položíme odpovídající $[\tilde{L}]_{i,j}$ automaticky roven nule. Počet operací nutných k výpočtu je pak mnohem menší. Tento postup je obvzláště efektivní, pokud původní matice soustavy $A$ je řídká, pak zřejmě i matice předpodmínění $C = \tilde{L}\tilde{L}^T$ bude řídká. Při tomto zjednodušení však není zaručeno, že matice $C$ bude positivně definitní. Toto lze však napravit přenásobením diagonálních prvků matice $A$ vhodnou konstantou $\omega \gg 1$.

2. *Neúplný rozklad podle hodnoty*
   Vypočtený nový prvek $[\tilde{L}]_{i,j}$ zanedbáváme, tj. položíme roven nule, pokud je relativně menší než diagonální prvek matice $A$, tj. $|\tilde{L}_{i,j}| \leqq \epsilon |A_{i,i}|$. Tento způsob zaručuje positivní definitnost výsledné matice předpodmínění, nicméně tato matice může mít více nenulových prvků než původní matice. Je zřejmé, že pro volbu $\epsilon = \infty$ dostáváme úplnou Choleského faktorizaci. Volbou většího $\epsilon$ můžeme dosáhnout lepšího efektu předpodmínění a tím i menšího počtu iterací metodou sdružených gradientů. Toto je však vyváženou větší výpočetní náročností sestavení matice předpodmínění a následného výpočtu předpodmíněného gradientu $s_k$ v každém kroce metody PCG. Výsledná efektivita tak nemusí být zřejmá a liší se případ od případu.

**Algoritmus 9** NEÚPLNÁ CHOLESKÉHO FAKTORIZACE

**Input:** $A$, přesnost $\epsilon > 0$

$n :=$ velikost $A$
**for** $j = 1, \ldots, n$ **do**
    $v = s_j^A - \sum_{k=1}^{j-1} [L]_{j,k} s_k^L$
    $s = \frac{1}{\sqrt{[v]_j}} v$
    **for** $i = j, \ldots, n$ **do**
        **if** $|s_i| < \epsilon |A_{j,j}|$ **then**
            $\tilde{L}_{j,i} = 0$
        **else**
            $\tilde{L}_{j,i} = s_i$
        **end if**
    **end for**
**end for**

**Output:** $\tilde{L}$

Více o předpodmiňovacích technikách lze nalézt například v [18], případně [22].

**Příklad 3.43.** Porovnejte rychlosti numerických řešičů soustav lineárních rovnic uvedených v této kapitole na testovací soustavě

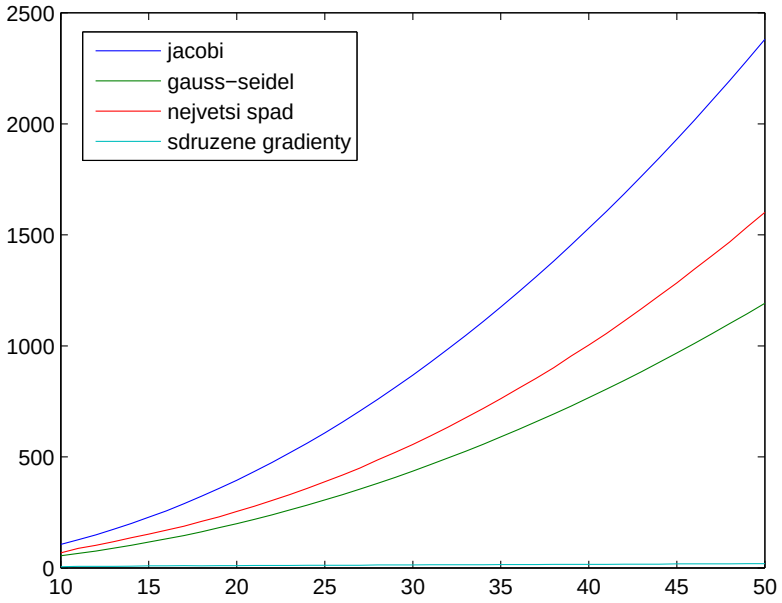
$$Ax = b \,,$$

kde

$$A = \text{fivediag}\,(-1,-1,4,-1,-1) \in \mathbb{R}^{n,n}, \quad b = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n$$

a volte různou dimenzi úlohy $n$.

*Řešení.* Pro požadovanou přesnost $\epsilon = 10^{-4}$ získáme následující počty iterací pro jednotlivé metody ▲




Obr. 3.11: Počet iterací algoritmů pro řešení soustavy lineárních rovnic pro různou dimenzi úlohy.

# Kapitola 4

# Výpočet vlastních čísel a vektorů matic

### Průvodce studiem

*Další velmi významnou skupinou úloh lineární algebry je problém hledání vlastních čísel a vlastních vektorů. Jejich fyzikální interpretace spočívá v popisu vlnění či vibrací, nacházejí však uplatnění i více teoretické při dělení grafů apod. Největší a nejmenší vlastní číslo jsme potřebovali i v předchozí kapitole pro výpočet čísla podmíněnosti.*
*V této kapitole se proto seznámíme se základními metodami řešení těchto úloh. Nejprve se budeme zabývat tzv. částečným problémem vlastních čísel, kdy budeme hledat jen několik vlastních čísel a na závěr si ukážeme i řešení úplného problému, kdy výstupem numerických metod bude celé spektrum matice.*

## 4.1 Výpočet charakteristického polynomu

Nejprve si zopakujme pojmy vlastních čísel a vlastních vektorů.

**Definice 4.1.** Nechť $A \in \mathbb{R}^{n,n}$ je matice.
Jestliže existuje nenulový vektor $\mathrm{e} \in \mathbb{C}^n$ a skalár $\lambda \in \mathbb{C}$ tak, že

$$A\mathrm{e} = \lambda \mathrm{e},$$

pak se $\lambda$ nazývá *vlastní číslo* matice $A$, e se nazývá *vlastní vektor* příslušný k $\lambda$ a $(\lambda, \mathrm{e})$ se nazývá *vlastní dvojice* matice $A$.

**Definice 4.2.** Pod pojmem *charakteristický polynom matice* $A$ rozumíme polynom

$$p(\lambda) := \det(A - \lambda I) \tag{4.1}$$

a pojmem *charakteristická rovnice matice* $A$ rozumíme rovnici

$$p(\lambda) = 0 \ . \tag{4.2}$$

Analytický postup pro nalezení vlastních čísel matice spočívá v sestavení charakteristického polynomu (4.2) a následně nalezení jeho kořenů, tj. řešení charakteristické rovnice (4.2) (viz např. [6]).

**Příklad 4.3.** Nalezněte vlastní čísla matice

$$A := \begin{pmatrix} 4 & -2 \\ 1 & 2 \end{pmatrix} \ .$$

*Řešení.* Nejdříve sestavíme charakteristický polynom dosazením do předpisu (4.2)

$$p(\lambda) = \det(A - \lambda I) = \begin{vmatrix} 4-\lambda & -2 \\ 1 & 2-\lambda \end{vmatrix} = (4-\lambda)(2-\lambda) - (-2).1 = \lambda^2 - 6\lambda + 10$$

a vlastní čísla jsou kořeny tohto charakteristického polynomu, tj. jsou řešením rovnice

$$\lambda^2 - 6\lambda + 10 = 0 \ .$$

Využitím např. diskriminantu získáme hodnoty $\lambda_1 = 3 + i$, $\lambda_2 = 3 - i$. ▲

Čtenář si jistě všiml základního nedostatku výše uvedeného postupu. Problém nečiní hledání kořenů charakteristického polynomu (lze použít například některý z algoritmů z kapitoly 2), ale sestavení samotného charakteristického polynomu pomocí symbolického výpočtu deteminantu. Pro matici řádu $n$ je nutno provést $(n-1)n!$ součinů, což pro matice vyšších řádů je prakticky nemožné.

### 4.1.1 Geršgorinova věta

Nejjednodušší postup, jak lokalizovat spektrum matice se skrývá v následující větě.

**Lemma 4.4.** *(Geršgorinova)*
*Všechny vlastní čísla matice* $A \in \mathbb{R}^n$ *leží ve sjednocení kruhů*

$$S := S_1 \cup S_2 \cup \ldots \cup S_n \ ,$$

*kde*

$$S_i := \{z \in \mathbb{C} : |z - a_{ii}| \leqq \sum_{j=1, j\neq i}^{n} |a_{ij}|\}, \quad i = 1 \ldots n \ .$$

**Příklad 4.5.** Nechť je dána matice

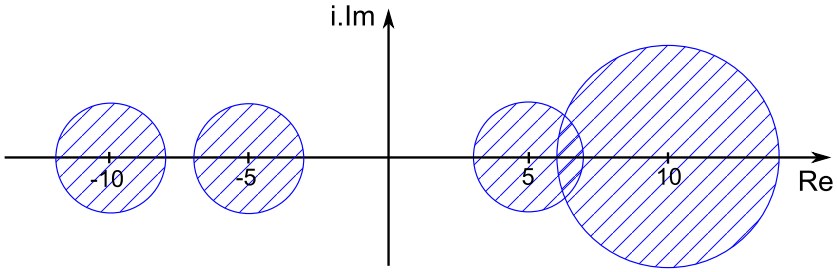
$$A = \begin{pmatrix} 10 & 1 & 1 & 2 \\ 1 & 5 & 1 & 0 \\ 1 & 1 & -5 & 0 \\ 2 & 0 & 0 & -10 \end{pmatrix}.$$

Lokalizujte spektrum této matice pomocí Geršgorinovy věty.

*Řešení.* Prostým dosazením do předpisu ve větě (4.4) získáme předpisy pro jednotlivé kruhy

$$\begin{aligned} S_1 &:= \{z \in \mathbb{C} : |z - 10| \leqq 1 + 1 + 2 = 4\} \\ S_2 &:= \{z \in \mathbb{C} : |z - 5| \leqq 1 + 1 + 0 = 2\} \\ S_3 &:= \{z \in \mathbb{C} : |z + 5| \leqq 1 + 1 + 0 = 2\} \\ S_4 &:= \{z \in \mathbb{C} : |z + 10| \leqq 2 + 0 + 0 = 2\} \end{aligned}$$

a výsledek lze zakreslit do Gaussovy roviny, viz Obrázek 4.1.
Spektrum zadané matice leží ve sjednocení těhto kruhů. ▲




Obr. 4.1: Příklad 4.5, lokalizace spektra pomocí Geršgorinovy věty.

**Poznámka 4.6.** O vlastních číslech reálné symetrické matice víme, že jsou reálná (viz [2]). V takovém případě se Geršgorinovy kruhy redukují na intervaly reálných čísel a spektrum matice pak leží ve sjednocení těchto intervalů.

### 4.1.2 Krylovova metoda

V úvodu kapitoly jsme se zmínili, že jedním z možných postupů výpočtu vlastních čísel matice je řešení charakteristické rovnice. Výpočet charakteristického polynomu pomocí determinantu je však netriviální úloha. Ukažme si nyní, jak charakteristický polynom nalézt mnohem efektivněji.

Za tímto účelem uvažujme *charakteristický polynom matice* $A$ ve tvaru

$$p(\lambda) := \lambda^n + \sum_{i=0}^{n-1} b_i \lambda^i, \tag{4.3}$$

kde $\lambda$ je libovolné vlastní číslo matice $A$.
Obecně samozřejmě nemusí být zaručeno, že charakteristický polynom má koeficient u nejvyšší mocniny roven 1, nicméně na řešení charakteristické rovnice toto zjednodušení nemá vliv, neboť celou rovnici můžeme vydělit nenulovým koeficientem u nejvyšší mocniny.
Metoda výpočtu charakteristického polynomu je založena na následující větě.

**Věta 4.7.** *(Cayley-Hamilton)*
*Nechť $p(\lambda)$ je charakteristický polynom matice $A$.*
*Pak*

$$p(A) = O \ .$$

*Důkaz.* Viz [2]. □

Dosaďme do předpisu charakteristického polynomu (4.3) místo vlastního čísla matice matici $A$, tj.

$$p(A) = A^n + \sum_{i=0}^{n-1} b_i A^i \ .$$

Pak dle Věty (4.7) platí $p(A) = O$.
Přenásobíme-li tuto soustavu libovolným $y \in \mathbb{R}^n$ zprava, získáme

$$A^n y + \sum_{i=0}^{n-1} b_i A^i y = o$$

a po drobné úpravě

$$\sum_{i=0}^{n-1} b_i A^i y = -A^n y \ .$$

Předchozí rovnici lze zapsat v maticové formě

$$\tilde{A} b = \tilde{p} \ , \tag{4.4}$$

kde

$$\tilde{A} := \left(A^0 y, A^1 y, A^2 y, \dots A^{n-1} y\right) \ , \ \tilde{p} := -A^n y \ ,$$
$$b := (b_0, b_1, \dots, b_{n-1})^T \ .$$

Podařilo se nám nalézt postup, kterým lze získat koeficienty charakteristického polynomu bez nutnosti symbolického výpočtu determinantu jako řešení soustavy lineárních rovnic (4.4).

**Algoritmus 10** KRYLOVOVA METODA

**Input:** $A$

1: $n :=$ velikost $A$
2: $y :=$ libovolný jednotkový vektor dimenze $n$
3: **for** $j = 1, \ldots, n$ **do**
4: $\quad s_j^{\tilde{A}} = A^{j-1}y$
5: **end for**
6: $\tilde{b} = -A^n y$
7: $b$ je řešením soustavy $\tilde{A}b = \tilde{b}$

**Output:** $b_0, \ldots, b_{n-1}$

**Poznámka 4.8.** Pro velké matice je však sestavení matice soustavy poměrně náročné, neboť obsahuje $n$ násobení matice a vektoru.

**Příklad 4.9.** Sestavte charakteristický polynom matice

$$A = \begin{pmatrix} 10 & 1 & 1 & 2 \\ 1 & 5 & 1 & 0 \\ 1 & 1 & -5 & 0 \\ 2 & 0 & 0 & -10 \end{pmatrix}.$$

pomocí Krylovovy metody.

*Řešení.* Nejdříve zvolíme libovolný vektor $y \in \mathbb{R}^n$. Jistě každého napadne, že nulový vektor není právě vhodnou volbou, jelikož soustava (4.4) by měla nulovou matici soustavy a nulový vektor pravých stran. Takové rovnici pak vyhovuje libovolné $b \in \mathbb{R}^n$. Volme tedy

$$y = [1, 0, 0, 0]^T .$$

Sestavme soustavu (4.4). Pro jednotlivé sloupce matice $\tilde{A}$ a vektor $\tilde{p}$ platí

$$\begin{aligned} s_1^{\tilde{A}} &= A^0 y = y = (1, 0, 0, 0)^T \\ s_2^{\tilde{A}} &= A^1 y = A s_1^{\tilde{A}} = (10, 1, 1, 2)^T \\ s_3^{\tilde{A}} &= A^2 y = A s_2^{\tilde{A}} = (106, 16, 6, 0)^T \\ s_4^{\tilde{A}} &= A^3 y = A s_3^{\tilde{A}} = (1082, 192, 92, 212)^T \\ \tilde{p} &= -A^4 y = -A s_3^{\tilde{A}} = (-11528, -2134, -814, -44)^T . \end{aligned}$$

Hledaná soustava má tvar

$$\begin{pmatrix} 1 & 10 & 106 & 1028 \\ 0 & 1 & 16 & 192 \\ 0 & 1 & 6 & 92 \\ 0 & 2 & 0 & 212 \end{pmatrix} b = \begin{pmatrix} -11528 \\ -2134 \\ -814 \\ -44 \end{pmatrix}$$

a jejím řešením (analyticky nebo lépe využitím libovolné metody z Kapitoly 3) je

$$b = (2684, -22, -132, 0)^T \ .$$

Dosazením hodnot do předpisu (4.3) získáme charakteristický polynom v tvaru

$$p(\lambda) = \lambda^4 - 132\lambda^2 - 22\lambda + 2684 \ .$$

▲

### 4.1.3 Výpočet charakteristického polynomu třídiagonální matice

V kapitolách (4.3.2) a (4.3.3) si představíme numerické metody pro převod reálných symetrických matic na třídiagonální matice. Je tedy důležité si ukázat, jak je možné efektivně sestavit charakteristický polynom třídiagonálních matic.

**Definice 4.10.** Pojmem *třídiagonální matice* rozumíme matici v tvaru

$$T_n := \begin{pmatrix} a_1 & b_1 & 0 & 0 & 0 & \dots & 0 \\ c_1 & a_2 & b_2 & 0 & 0 & \dots & 0 \\ 0 & c_2 & a_3 & b_3 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & c_{n-3} & a_{n-2} & b_{n-2} & 0 \\ 0 & \dots & 0 & 0 & c_{n-2} & a_{n-1} & b_{n-1} \\ 0 & \dots & 0 & 0 & 0 & c_{n-1} & a_n \end{pmatrix} ,$$

kde $a_i, b_j, c_j \in \mathbb{R},\ i = 1, \dots, m,\ j = 1, \dots, n-1.$

**Lemma 4.11.** *Nechť je dána třídiagonální matice $T_n$.*
*Pak pro charakteristický polynom matice $T_n$ platí*

$$p_n(\lambda) = (a_n - \lambda)p_{n-1}(\lambda) - b_{n-1}c_{n-1}p_{n-2}(\lambda) \ ,$$

*kde $p_{n-1}$ a $p_{n-2}$ jsou charakteristické polynomy matic $T_{n-1}$ a $T_{n-2}$ a*

$$p_0(\lambda) = 1, \quad p_1(\lambda) = a_1 - \lambda \ .$$

*Důkaz.* Plyne z rozvoje determinantu podle sloupce a z definice charakteristického polynomu. □

**Příklad 4.12.** Určete charakteristický polynom matice

$$A = \begin{pmatrix} 1 & -3 & 0 & 0 \\ 4 & 2 & -2 & 0 \\ 0 & 5 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix} .$$

*Řešení.* Postupně určíme dle Věty 4.11 potřebné polynomy

$$\begin{aligned}
p_0(\lambda) &= 1 \\
p_1(\lambda) &= a_1 - 1 = 1 - \lambda \\
p_2(\lambda) &= (a_2 - \lambda)p_1(\lambda) - b_1c_1p_0(\lambda) = (2-\lambda)(1-\lambda) + 12 = \lambda^2 - 3\lambda + 14 \\
p_3(\lambda) &= (a_3 - \lambda)p_2(\lambda) - b_1c_1p_1(\lambda) = (3-\lambda)(\lambda^2 - 3\lambda + 14) + 10.(1-\lambda) \\
&= -\lambda^3 + 6\lambda^2 - 33\lambda + 52 \\
p_4(\lambda) &= (a_4 - \lambda)p_3(\lambda) - b_1c_1p_2(\lambda) \\
&= (3-\lambda)(-\lambda^3 + 6\lambda^2 - 33\lambda + 52) + 0.(\lambda^2 - 3\lambda + 14) \\
&= \lambda^4 - 8\lambda^3 + 45\lambda^2 - 118\lambda + 104 \ .
\end{aligned}$$

Tedy charakteristický polynom dané matice má tvar

$$p(\lambda) = \lambda^4 - 8\lambda^3 + 45\lambda^2 - 118\lambda + 104 \ .$$

▲

**Poznámka 4.13.** Numerický výpočet koeficientů $p_n(\lambda)$ však bohužel neumožňuje symbolickou manipulaci s proměnnou $\lambda$. Nicméně polynomy lze nahradit aritmetickými vektory, jejichž složky reprezentují koeficienty u mocnin $\lambda$.
Označme koeficienty v polynomu $p_n(\lambda)$ jako vektor $p_n \in \mathbb{R}^{n+1}$ (daný polynom je $n$-tého stupně, proto bude mít $n+1$ koeficientů). Tedy polynom ve tvaru

$$p_n(\lambda) = \gamma_0\lambda^0 + \gamma_1\lambda^1 + \cdots + \gamma_n\lambda^n$$

bude mít příslušný vektor koeficientů

$$p_n = (\gamma_0, \gamma_1, \ldots, \gamma_n) \ .$$

Zbývá odvodit rekurentní předpis ve Větě 4.11 pro vektory koeficientů.
Nechť

$$\begin{aligned}
p_{n-2} &= (\alpha_0, \alpha_1, \ldots, \alpha_{n-2}) \\
p_{n-1} &= (\beta_0, \beta_1, \ldots, \beta_{n-1}) \ ,
\end{aligned}$$

pak

$$p_n = (\gamma_0, \gamma_1, \ldots, \gamma_n) \ ,$$

kde

$$\begin{aligned}
\gamma_0 &= \alpha_n\beta_0 - b_{n-1}c_{n-1}\alpha_0 \\
\gamma_i &= \alpha_n\beta_i - b_{n-1}c_{n-1}\alpha_i, \quad i = 1, \ldots, n-2 \\
\gamma_{n-1} &= \alpha_n\beta_{n-1} + \beta_{n-2} \\
\gamma_n &= -\beta_{n-1} \ .
\end{aligned}$$

## 4.2 Výpočet dominantního vlastního čísla

V následujícím textu se zaměříme na metody řešící částečný problém vlastních čísel. Zejména si představíme metody, které nám umožní vypočítat tzv. *dominantní*

vlastní číslo. Případné další vlastní čísla budeme schopni dopočíst pomocí maticové redukce či posunem spektra. Nejprve si však zavedeme pojem dominantního vlastního čísla.

**Definice 4.14.** Nechť je dána matice $A$ a vlastní čísla splňující nerovnost

$$|\lambda_1| > |\lambda_2| \geqq \ldots \geqq |\lambda_n|$$

a nechť $x_1, x_2, x_3, \ldots x_n \in \mathbb{R}^n$ jsou odpovídající lineárně nezávislé vlastní vektory. Pak *dominantním vlastním číslem* rozumíme vlastní číslo $\lambda_1$.

V následujícím výkladu se zaměříme na mocninnou metodu, která dokáže nalézt dominantní vlastní číslo.

### 4.2.1 Mocninná metoda

Předpokládejme, že

$$|\lambda_1| > |\lambda_2| \geqq |\lambda_3| \geqq \ldots \geqq |\lambda_n|$$

jsou vlastní čísla matice $A$ řádu $n$, a nechť $U = \{x_1, \ldots, x_n\}$ je množina jim odpovídajících lineárně nezávislých vektorů. Pak množina $U$ tvoří bázi vektorového prostoru $\mathbb{R}^n$ a libovolný vektor $v_0 \in \mathbb{R}^n$ lze vyjádřit jako lineární kombinaci prvků množiny $U$, tj. existují reálné konstanty $\alpha_1, \ldots \alpha_n$ takové, že

$$v_0 = \alpha_1 x_1 + \cdots + \alpha_n x_n \ . \tag{4.5}$$

Definujme posloupnost $\{v_k\}, v_k \in \mathbb{R}^n$ takovou, že

- $v_0$ je libovolné takové, že $(v_0, x_1) \neq 0$
- $v_k$ vypočteme předpisem

$$v_k = A v_{k-1} \ . \tag{4.6}$$

Dosazením lineární kombinace (4.5) do předpisu (4.6) získáme

$$v_k = A v_{k-1} = A^k v_0 = A_k \sum_{i=1}^{n} \alpha_i x_i = \sum_{i=1}^{n} \alpha_i A^k x_i = \sum_{i=1}^{n} \alpha_i \lambda_i^k x_i.$$

Tento předpis se nám hodí v důkazu Věty 4.16.

**Definice 4.15.** (Schwarzova konstanta)
Nechť $y \in \mathbb{R}^n$ takové, že $(y, x_1) \neq 0$, kde $x_1$ je vlastní vektor odpovídající dominantnímu vlastnímu číslu.
Číslo

$$\sigma_k := (y, v_k) = (y, A^k v_0)$$

nazýváme *Schwarzovou konstantou.*

**Lemma 4.16.** *Podíl Schwarzových konstant*

$$\frac{\sigma_{k+1}}{\sigma_k} = \frac{(y, A^{k+1}v_0)}{(y, A^k v_0)}$$

*konverguje k dominantnímu vlastnímu číslu* $\lambda_1$*, tj.*

$$\lim_{k\to\infty} \frac{\sigma_{k+1}}{\sigma_k} = \lambda_1 \ .$$

*Důkaz.* Upravme

$$\frac{\sigma_{k+1}}{\sigma_k} = \frac{(y, A^{k+1}v_0)}{(y, A^k v_0)} = \frac{\sum_{i=1}^{n} \alpha_i \lambda_i^{k+1}(y, x_i)}{\sum_{i=1}^{n} \alpha_i \lambda_i^k (y, x_i)} \cdot \frac{\frac{1}{\lambda_1^k}}{\frac{1}{\lambda_1^k}} = \frac{\alpha_1\lambda_1 \frac{\lambda_1^k}{\lambda_1^k}(y, x_1) + \sum_{i=2}^{n} \alpha_i \lambda_i \frac{\lambda_i^k}{\lambda_1^k}(y, x_i)}{\alpha_1 \frac{\lambda_1^k}{\lambda_1^k}(y, x_1) + \sum_{i=2}^{n} \alpha_i \frac{\lambda_i^k}{\lambda_1^k}(y, x_i)}$$

a jelikož

$$\lim_{k\to\infty} \left(\frac{\lambda_i}{\lambda_1}\right)^k = \lim_{k\to\infty} \frac{\lambda_i^k}{\lambda_1^k} = 0 \ ,$$

dostáváme

$$\lim_{k\to\infty} \frac{(y, A^{k+1}v_0)}{(y, A^k v_0)} = \lim_{k\to\infty} \frac{a_1\lambda_1(y, x_1)}{a_1(y, x_1)} = \lambda_1 \ .$$

□

**Algoritmus 11** MOCNINNÁ METODA

**Input:** $v_0$, $y$, přesnost $\epsilon > 0$

1: $v_1 := Av_0$
2: $\lambda_{1,1} := (y, v_1)/(y, v_0)$
3: $k := 1$
4: **while** $\|Av_k - \lambda_{1,k}v_k\| \geq \epsilon$ **do**
5: $\quad v_{k+1} := Av_k$
6: $\quad \lambda_{1,k+1} := (y, v_{k+1})/(y, v_k)$
7: $\quad k := k + 1$
8: **end while**

**Output:** $\lambda_{1,k}, v_k$

**Poznámka 4.17.** V průběhu algoritmu může nastat, že složky vlastního vektoru budou velmi rychle růst. Toto je důsledek toho, že $v_{k+1} = A^k v_0$ a s vysokými mocninami se budou složky velmi zvětšovat. To má ovšem za následek velké chyby v počítačové aritmetice. Proto je nutné použít normalizaci vlastních vektorů.

**Algoritmus 12** NORMOVANÁ MOCNINNÁ METODA

**Input:** $v_0$, přesnost $\epsilon > 0$

$y_0 := v_0/\|v_0\|$
$v_1 := Ay_0$
$y_1 := v_1/\|v_1\|$
$\lambda_{1,1} := \|v_1\|$
$k := 1$
**while** $\|Ay_k - \lambda_{1,k}y_k\| \geq \epsilon$ **do**
    $v_{k+1} := Ay_k$
    $y_{k+1} := v_{k+1}/\|v_{k+1}\|$
    $\lambda_{1,k+1} := \|v_{k+1}\|$
    $k := k + 1$
**end while**

**Output:** $\lambda_{1,k}, y_k$

**Příklad 4.18.** Otestujte algoritmus normované mocinné metody na matici

$$A \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 2 & 1 & -1 \\ -2 & 0 & 3 & 5 \\ 4 & -1 & 0 & 2 \end{pmatrix} .$$

Nalezněte touto metodou dominantní vlastní číslo s přesností $\epsilon = 10^{-4}$.

*Řešení.* Jako počáteční vektor volme $v_0 = (1, 0, 0, 0)^T$. Pak velikost dominantního vlastního čísla $\lambda_1$ v průběhu iterací je znázorněn na Obrázku 4.2. Algoritmus je ukončen po 12-ti iteracích, dominantním vlastním číslem s požadovanou přesností je

$$\lambda_1 = 6.5445 .$$

▲

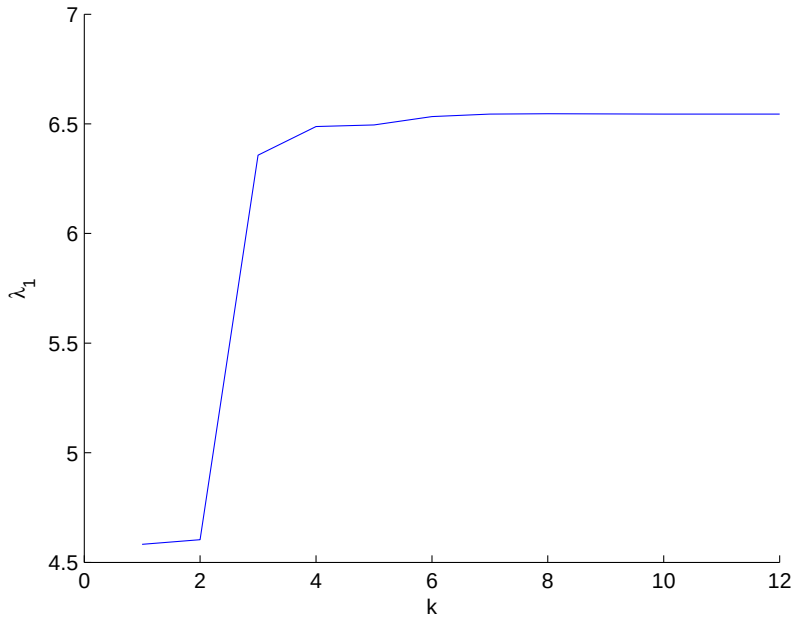
Pro symetrické matice počítáme místo podílů Schwarzových konstant tzv. *Rayleighovy podíly*:

$$r(v_k) \stackrel{\text{def}}{=} \frac{(v_k, v_{k+1})}{(v_k, v_k)}, \quad \lim_{k\to\infty} \frac{(v_k, v_{k+1})}{(v_k, v_k)} = \lambda_1 .$$

Pokud však uvažujeme pouze symetrické positivně definitní matice $A$, pak jsou všechna vlastní čísla kladná reálná, tedy je lze uspořádat

$$\lambda_1 > \lambda_2 \geqq \lambda_3 \geqq \ldots \geqq \lambda_n . \tag{4.7}$$

Kromě konvergence k největšímu vlastnímu číslu mají Rayleighovy podíly mnohem silnější vlastnost.




Obr. 4.2: Příklad 4.18 - numerický test algoritmu normované mocninné metody.

**Lemma 4.19.** *Nechť $A \in \mathbb{R}^{n,n}$ je symetrická positivně definitní reálná matice s vlastními čísly (4.7). Pak pro libovolné $v \in \mathbb{R}^n$ platí*

$$\lambda_n v^T v \leqq v^T A v \leqq \lambda_1 v^T v \ . \tag{4.8}$$

*Důkaz.* Označme

$$A = U \Lambda U^T$$

jako spektrální rozklad matice $A$ (kde $U \in \mathbb{R}^n$ je ortogonální matice vlastních vektorů a $\Lambda \in \mathbb{R}^n$ je diagonální matice s vlastními čísly na diagonále).
Pak úpravou získáme

$$v^T A v = v^T U \Lambda U^T v = (U^T v)^T \Lambda (U^T v)$$

označme $y \overset{\text{ozn}}{=} U^T v$, pak

$$(U^T v)^T \Lambda (U^T v) = y^T \Lambda y = \sum_{i=1}^{n} \lambda_i [y]_i^2 \ .$$

Očividně však

$$\begin{aligned} \sum_{i=1}^{n} \lambda_i [y]_i^2 &\leqq \lambda_1 y^T y = \lambda_1 v^T v \ , \\ \sum_{i=1}^{n} \lambda_i [y]_i^2 &\geqq \lambda_n y^T y = \lambda_n v^T v \ . \end{aligned}$$

□

**Poznámka 4.20.** Uvažujme-li $v \neq 0$, pak rovnici (4.8) lze upravit

$$\lambda_n \leqq \frac{v^T A v}{v^T v} \leqq \lambda_1$$

a získat tak odhad pro Rayleighovy podíly.

### 4.2.2 Posun a redukce spektra

Mocninná metoda slouží pouze k výpočtu dominantního vlastního čísla. Pokud bychom chtěli vypočíst i jiná vlastní čísla, museli bychom využít některých maticových operací, které upravují spektrum požadovaným způsobem.
Podívejme se na některé z nich.

**Lemma 4.21.** *(Posun spektra)*
*Nechť $\lambda$ je vlastní číslo matice $A$ a $v$ odpovídající vlastní vektor. Pak matice*

$$A - qI, \quad q \in \mathbb{R}$$

*má vlastní číslo $\lambda - q$ a stejný vlastní vektor $v$.*

*Důkaz.*

$$(A - qI)v = Av - qv = \lambda v - qv = (\lambda - q)v$$

□

**Poznámka 4.22.** Výše uvedené věty se dá využít například na zjištění nejmenšího vlastního čísla mocninnou metodou. Je-li

$$|\lambda_1| > |\lambda_2| \geqq \ldots \geqq |\lambda_{n-1}| > |\lambda_n|$$

Pak $A - \lambda_1 I$ má vlastní čísla

$$0, \lambda_2 - \lambda_1, \ldots, \lambda_{n-1} - \lambda_1, \lambda_n - \lambda_1$$

přičemž bude platit

$$|\lambda_n - \lambda_1| > |\lambda_{n-1} - \lambda_1| > \cdots > |\lambda_2 - \lambda_1| > 0$$

Použijeme-li tedy mocninnou metodu na takto upravenou matici, dostaneme dominantní vlastní číslo $\tilde{\lambda} := \lambda_n - \lambda_1$ a odtud

$$\lambda_n = \tilde{\lambda} + \lambda_1 \ .$$

**Lemma 4.23.** *(Maticová redukce)*
*Nechť $\lambda, v$ jsou vlastní číslo a vlastní vektor matice $A$. Nechť $w$ je* levý *vlastní vektor $\lambda$, tj. $A^T w = \lambda w$, a nechť $(v, w) = 1$.*
*Pak matice*

$$W \stackrel{\text{def}}{=} A - \lambda v w^T$$

*má místo vlastního čísla $\lambda$ nulu a všechny ostatní vlastní čísla stejná.*

*Důkaz.*

$$Wv = Av - \lambda v w^T v = Av - \lambda v = 0$$

Označme $\mu, x$ jako ostatní vlastní čísla a vlastní vektory. Pak

$$Wx = Ax - \lambda v w^T x = Ax = \mu x$$

□

**Poznámka 4.24.** Maticové redukce se dá využít k výpočtu druhého dominantního čísla.
Je-li

$$|\lambda_1| > |\lambda_2| \geqq |\lambda_3| \geqq \ldots \geqq |\lambda_n| \ ,$$

můžeme vypočíst pomocí mocninné metody $\lambda_1$ a jemu odpovídající vlastní vektor $v_1$ i levý vlastní vektor $\tilde{w}_1$ aplikací mocninné metody na matici $A^T$. Normováním

$$w_1 = \frac{\tilde{w}_1}{(v_1, \tilde{w}_1)} \ ,$$

pak splníme podmínku $(v_1, w_1) = 1$. Použitím maticové redukce $W = A - \lambda_1 v_1 w_1^T$ můžeme mocninou metodou spočítat vlastní číslo $\lambda_1$ a jemu odpovídající vlastní vektor, neboť původní dominantní vlastní číslo $\lambda_1$ se maticovou redukcí vynulovalo.

Pochopitelně lze celý proces opakovat a získat tak $\lambda_3$, atd.
Výpočet všech vlastních čísel je však tímto postupem výpočetně velmi náročný a nepoužívá se. Vhodnější je využití některé z metod pro úplný problém vlastních čísel.

## 4.3 Metody založené na podobnosti matic

Na rozdíl od předchozí Kapitoly 4.2, kdy jsme se zabývali hledáním dominantního vlastního čísla, v dalším textu představíme metody pro hledání všech vlastních čísel a vlastních vektorů.
Úlohu, která řeší všechny vlastní čísla a vektory matice nazýváme *úplný problém vlastních čísel*. Všechny metody, které si v následujícím textu představíme budou založeny na základní vlastnosti, které říkáme *podobnost matic*.

**Definice 4.25.** O maticích $A, B \in \mathrm{R}^{n,n}$ řekneme, že jsou *podobné*, existuje-li regulární matice $T$ taková, že

$$B = TAT^{-1} \ . \tag{4.9}$$

**Poznámka 4.26.** Naopak pokud čtvercovou matici $A$ přenásobíme zleva regulární maticí $T$ a zprava inverzí $T^{-1}$ (tj. (4.9)), získáme matici $B$, která je *podobná* matici $A$.

**Lemma 4.27.** *Podobné matice mají stejná vlastní čísla.*

*Důkaz.* Nechť $A$ je libovolná čtvercová matice s vlastním číslem $\lambda$ a příslušným vlastním vektorem $u$, tj.

$$Au = \lambda u \ .$$

Přenásobíme-li předchozí rovnici libovolnou regulární maticí $T$ zleva, získáme

$$TAu = \lambda Tu \ ,$$

a jelikož $T^{-1}T = I$, lze upravit

$$\begin{aligned} TAT^{-1}Tu &= \lambda Tu \\ TAT^{-1}(Tu) &= \lambda(Tu) \ . \end{aligned}$$

Označíme-li $B = TAT^{-1}$ a $v = Tu$, pak předchozí rovnici lze zapsat ve tvaru

$$Bv = \lambda v \ .$$

Tedy matice $B$ má vlastní číslo $\lambda$ (stejné jako matice $A$) jemuž odpovídá příslušný vlastní vektor $v = Tu$. □

V následujícím výkladu se zaměříme na metody založené na Větě 4.27. Nebudeme hledat vlastní čísla zadané matice $A$, nýbrž se budeme snažit nalézt vhodnou regulární matici $T$, která původní matici $A$ převede na „jednodušší matici“ $B$, která bude buď diagonální nebo alespoň třídiagonální.
Z důvodu lepší stability výpočtů se jeví jako vhodnější použití místo regulární matice $T$ tzv. matici *ortogonální*. Pokud navíc však zvolíme ortogonální matici $T$, výpočet inverze nebude nutný.

**Definice 4.28.** (Ortogonální matice)
*Ortogonální matice* je symetrická regulární matice pro kterou platí

$$Q^{-1} = Q^T \ .$$

Pak pro každou ortogonální matici platí $Q^{-1}Q = Q^TQ = I$.

**Lemma 4.29.** *Matice $A \in \mathbb{R}^{n,n}$ je ortogonální, právě když její řádky (případně sloupce) tvoří ortonormální množinu vektorů.*

*Důkaz.* Rozepišme

$$QQ^T = \begin{bmatrix} r_1^Q \\ \vdots \\ r_n^Q \end{bmatrix} [r_1^Q \dots r_n^Q] = \begin{bmatrix} (r_1^Q, r_1^Q) & \dots & (r_1^Q, r_n^Q) \\ \vdots & & \vdots \\ (r_1^Q, r_n^Q) & \dots & (r_n^Q, r_n^Q) \end{bmatrix}$$

Pokud $Q$ je ortogonální, pak $QQ^T = I$ (viz definice 4.28), tedy dle předchozí úpravy musí platit

$$\forall i, j \in \{1, \ldots, n\} : \ (r_i^Q, r_j^Q) = \begin{cases} 1 & \text{pro } i = j \\ 0 & \text{pro } i \neq j \end{cases} ,$$

což je přesně definice množiny ortonormálních vektorů.

□

Pokud $Q$ je ortogonální matice, lze předpis pro podobné matice (4.9) zjednodušit na tvar

$$B = QAQ^T \ . \tag{4.10}$$

### 4.3.1 Algoritmus QR

**Věta 4.30.** *Nechť $A \in \mathbb{R}^{n,n}$ je regulární matice. Pak existují horní trojuhelníková matice $R \in \mathbb{R}^{n,n}$ a ortogonální matice $Q \in \mathbb{R}^{n,n}$ takové, že*

$$A = QR \ . \tag{4.11}$$

*Důkaz.* Důkaz této věty nebudeme zde uvádět, neboť je zložen na konstrukcí matice $Q$ z původní matice $A$ pomocí Gramova-Schmidtova ortonormalizačního procesu. Tento algoritmus je popsán na konci této podkapitoly. □

Jelikož $Q$ v rozkladu (4.11) je ortogonální, lze jí využít jako matici podobnostní transformace. Nechť $A_k = Q_k R_k$, pak předpis pro novou aproximaci bude mít tvar

$$A_{k+1} = Q_k^T A_k Q_k = Q_k^T (Q_k R_k) Q_k = R_k Q_k, \quad A_0 = A \ . \tag{4.12}$$

**Věta 4.31.** *Nechť matice $A$ je reálná a symetrická a nechť $A_0, A_1, A_2, \ldots, A_k, \ldots$ je posloupnost generovaná rekurzivním předpisem*

$$A_0 = A, \qquad A_{k+1} = R_k Q_k, \ k > 0 \ ,$$

*kde matice $Q_k, R_k$ jsou $QR$ rozkladem matice $A_k$. Pak*

$$\lim_{k \to \infty} A_k = D \ ,$$

*kde $D$ je diagonální matice s vlastními čísly matice $A$ na diagonále.*

*Důkaz.* Důkaz není zcela triviální, viz [15] nebo [16]. □

**Poznámka 4.32.** Věta 4.31 garantuje konvergenci matice $A_k$ k diagonální matici, avšak obecně po nekonečně mnoha podobnostních transformacích

$$Q_k^T A_k Q_k \ .$$

Při praktickém výpočtu se spokojíme s maticí $\tilde{D}$, která má mimodiagonální prvky pouze blízké nule.
Tato situace nastane po $m$ krocích. Tedy výsledkem bude $\tilde{D} = \tilde{Q}^T A \tilde{Q}$, kde

$$\tilde{Q} := Q_1 Q_2 \dots Q_m$$

je ortogonální matice jejíž sloupce aproximují ortonormální vlastní vektory odpovídající vlastním číslům v matici $D$.
Diagonální prvky matice $\tilde{D}$ jsou aproximacemi vlastních čísel v matici $D$.

QR algoritmus pro výpočet vlastních čísel a vlastních vektorů matice $A$, můžeme shrnout do následujících kroků

**Algoritmus 13** ALGORITMUS QR

**Input:** $A$, přesnost $\epsilon > 0$

1: $A_0 := A$
2: $n :=$ velikost $A$
3: $k := 0$
4: **while** $\exists [A_k]_{i,j} > \epsilon, \ \ i, j = 1, \dots, n : i \neq j$ **do**
5: najdi $Q_k, R_k$ jako QR rozklad matice $A_k$
6: $A_{k+1} := R_k Q_k$
7: $k := k + 1$
8: **end while**

**Output:** $A_k$

Zbývá otázka, jak určit QR rozklad dané regulární matice.
Matici $A$ lze převést na ortogonální matici pomocí Gramova-Schmidtova ortonormalizačního procesu, který z množiny lineárně nezávislých vektorů $\{s_1^A, \dots, s_n^A\}$ (ze sloupců regulární matice $A$) vytvoří množinu ortonormálních vektorů $\{s_1^Q, \dots, s_n^Q\}$ (ortonormální sloupce ortogonální matice $Q$). Grammův-Schmitův ortonormalizační algoritmus lze nalézt například v [2].
V následujícím výkladu budeme pomocí $s_1^A, s_2^A, \dots, s_n^A$ značit sloupce matice $A$, pomocí $\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_n$ ortogonální sloupcové vektory vytvořené Gramovým-Schmitovým ortogonalizačním procesem a výsledné ortonormální sloupce matice $Q$ pomocí $s_1^Q, s_2^Q, \dots, s_n^Q$.
Pak Gramův-Schmitův ortonormalizační proces lze vyjádřit pomocí následujících

rovnic

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\tilde{s}_1 &=& s_1^A & s_1^Q &=& \frac{1}{\|\tilde{s}_1\|}\tilde{s}_1 \\
\tilde{s}_2 &=& s_2^A + \alpha_{1,2}s_1^Q & s_2^Q &=& \frac{1}{\|\tilde{s}_2\|}\tilde{s}_2 \\
\tilde{s}_3 &=& s_3^A + \alpha_{1,3}s_1^Q + \alpha_{2,3}s_2^Q & s_3^Q &=& \frac{1}{\|\tilde{s}_3\|}\tilde{s}_3 \\
&\vdots& & &\vdots& \\
\tilde{s}_n &=& s_n^A + \alpha_{1,n}s_1^Q + \alpha_{2,n}s_2^Q + \cdots + \alpha_{n-1,n}s_{n-1}^Q & s_n^Q &=& \frac{1}{\|\tilde{s}_n\|}\tilde{s}_n
\end{array}$$

kde $\alpha$ jsou příslušné ortogonalizační konstanty.
Jednoduchou úpravou předchozích rovnic lze získat

$$\begin{array}{rcl}
s_1^A &=& \|\tilde{s}_1\|s_1^Q \\
s_2^A &=& -\alpha_{1,2}s_1^Q + \|\tilde{s}_2\|s_2^Q \\
s_3^A &=& -\alpha_{1,3}s_1^Q - \alpha_{2,3}s_2^Q + \|\tilde{s}_3\|s_3^Q \\
&\vdots& \\
s_n^A &=& -\alpha_{1,n}s_1^Q - \alpha_{2,n}s_2^Q - \ldots - \alpha_{n-1,n}s_{n-1}^Q + \|\tilde{s}_n\|s_n^Q
\end{array}$$

nebo maticově

$$A = Q \begin{pmatrix} \|\tilde{s}_1\| & -\alpha_{1,2} & -\alpha_{1,3} & \ldots & -\alpha_{1,n} \\ 0 & \|\tilde{s}_2\| & -\alpha_{2,3} & \ldots & -\alpha_{2,n} \\ 0 & 0 & \|\tilde{s}_3\| & \ldots & -\alpha_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \|\tilde{s}_n\| \end{pmatrix} = QR\ .$$

Předchozí pozorování lze šikovně zakomponovat do výsledného algoritmu.

**Algoritmus 14** KONSTRUKCE QR ROZKLADU

**Input:** $A$

1: $n :=$ velikost $A$
2: $R :=$ nulová matice typu $n \times n$
3: **for** j := 1,...,n **do**
4: $\quad v := s_j^A$
5: $\quad$ **for** i := 1,...,j-1 **do**
6: $\quad\quad r_{i,j} := (s_i^Q)^T s_j^A$
7: $\quad\quad v := v - r_{i,j}s_i^Q$
8: $\quad$ **end for**
9: $\quad r_{j,j} := \|v\|$
10: $\quad s_j^Q := v/\|v\|$
11: **end for**

**Output:** $Q, R$

**Příklad 4.33.** Nalezněte QR rozklad matice

$$A := \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} .$$

*Řešení.* Nejprve pomocí Grammova-Schmidtova ortonormalizačního procesu zkonstruujeme sloupce matice $Q$.

$$\begin{aligned} s_1^Q &= \frac{s_1^A}{\|s_1^A\|} = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right] \\ s_2^Q &= \frac{s_2^A - (s_2^A, s_1^Q)s_1^Q}{\|s_2^A - (s_2^A, s_1^Q)s_1^Q\|} = \left[\frac{1}{\sqrt{6}}, -\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}\right] \\ s_3^Q &= \frac{s_3^A - (s_3^A, s_2^Q)s_2^Q - (s_3^A, s_1^Q)s_1^Q}{\|s_3^A - (s_3^A, s_2^Q)s_2^Q - (s_3^A, s_1^Q)s_1^Q\|} = \left[-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right] , \end{aligned}$$

pak

$$Q = \left[s_1^Q, s_2^Q, s_3^Q\right] = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{pmatrix} .$$

Pro konstrukci matice $R$ využijeme předpis (4.11), získáme

$$R = Q^T A = \begin{pmatrix} (s_1^Q, s_1^A) & (s_1^Q, s_2^A) & (s_1^Q, s_3^A) \\ (s_2^Q, s_1^A) & (s_2^Q, s_2^A) & (s_2^Q, s_3^A) \\ (s_3^Q, s_1^A) & (s_3^Q, s_2^A) & (s_3^Q, s_3^A) \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{3}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & 0 & \frac{2}{\sqrt{3}} \end{pmatrix} .$$

▲

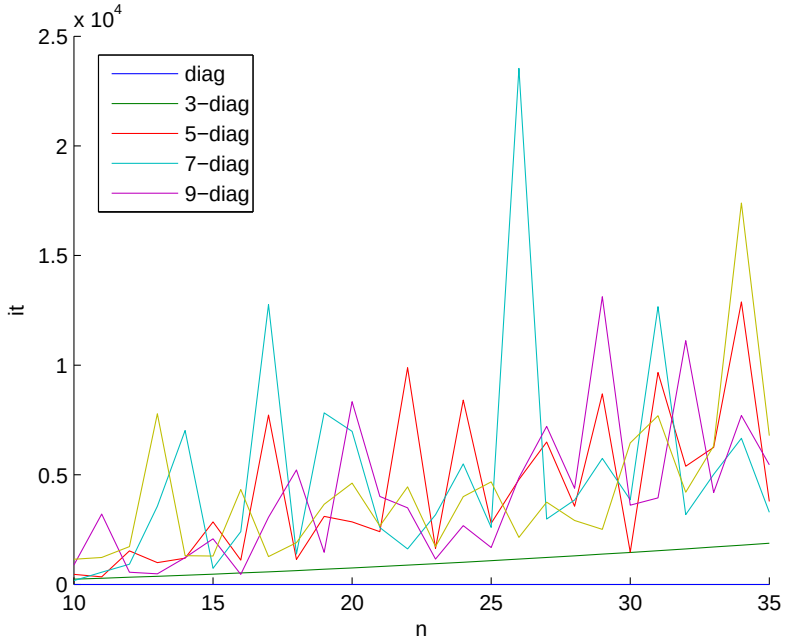
**Příklad 4.34.** Otestujte rychlost konvergence algoritmu QR na různě-diagonálních testovacích maticích

$$\begin{aligned} A_1 &= \text{diag } (5) \\ A_3 &= \text{3-diag } (-1, 5, -1) \\ A_5 &= \text{5-diag } (-1, -1, 5, -1, -1) \\ A_7 &= \text{7-diag } (-1, -1, -1, 5, -1, -1, -1) \\ A_9 &= \text{9-diag } (-1, -1, -1, -1, 5, -1, -1, -1, -1) . \end{aligned}$$

Volte různé dimenze testovacích matic.

*Řešení.* Pro různou volbu dimenze matic $n$ jsou počty iterací zobrazeny na obrázku 4.3. ▲

**Poznámka 4.35.** Výpočet QR rozkladu pomocí Gramova-Schmidtova procesu je pro velké dimenze matic numericky nestabilní. Proto se v praxi k výpočtu QR rozkladu používají ortogonální matice, jako je Givensova matice rovinné rotace nebo Householderova matice rovinného zrcadlení, viz [1].
Tyto matice využijeme i v daším textu při řešení problému vlastních čísel.




Obr. 4.3: Rychlost konvergence QR a různě diagonální matice.

### 4.3.2 Givensova metoda

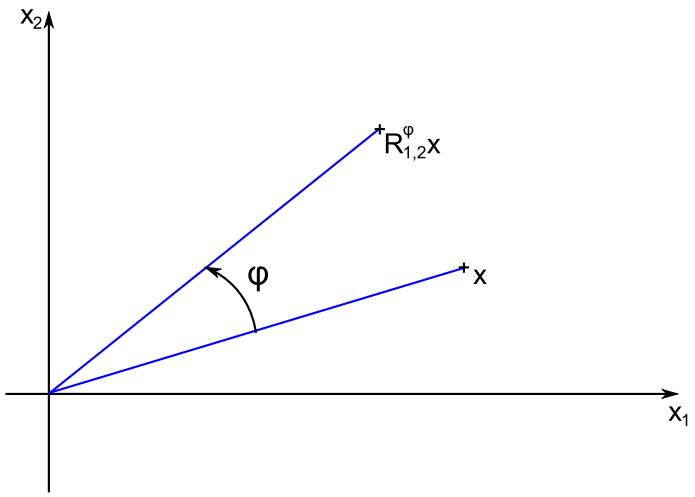
V předchozí kapitole jsme si popsali QR algoritmus, který postupně nuluje mimodiagonální prvky. Tento postup je však poměrně zdlouhavý.
Proto se nabízí postup, který by v konečném počtu kroků převedl matici $A$ pomocí podobnostních tranformací na matici třídiagonální. Na tuto matici pak lze aplikovat zmiňovaný QR algoritmus, nebo lze přímo vyčíslit charakteristický polynom. K podobnostním transformacím využijeme ortogonálních matic rovinné rotace.

**Definice 4.36.** *Maticí rovinné rotace* rozumíme matici s předpisem

$$R^{\varphi}_{i,j} := \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & \cos\varphi & \dots & \sin\varphi & & & \\ & & & \vdots & \ddots & \vdots & & & \\ & & & -\sin\varphi & \dots & \cos\varphi & & & \\ & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix}, \tag{4.13}$$

ve které se goniometrické funkce nachází v $i$-tém a $j$-tém řádku a sloupci.

Tato matice otáčí bod v rovině $(i, j)$ o úhel $\varphi$ kolem počátku.




Obr. 4.4: Givensovy rotace.

**Poznámka 4.37.** Matice rovinné rotace je ortogonální.

Cílem Givensovy metody je nalézt ortogonální matici $R$ takovou, že

$$T := R^T A R \tag{4.14}$$

je třídiagonální matice. Přitom matice $A$ a $T$ jsou podobné a tudíž mají stejná vlastní čísla.

Hledejme matici rovinné rotace $R_{p,q,r}$ v rovině $(p, q)$, která pomocí transformace

$$\tilde{A} = R^T_{p,q,r} A \tag{4.15}$$

nuluje prvek $[A]_{q,r}$ a mění jen prvky v řádcích $p, q$.
Prvek $[A]_{q,r}$ získáme jako skalární skalární součin $q$-tého řádku matice $R^T_{p,q,r}$ a $r$-tého sloupce matice $A$, tj.

$$\tilde{A}_{q,r} = (r_q^{R^T}, s_r^A) \quad = \cos\varphi A_{p,r} - \sin\varphi A_{q,r} = 0 \ .$$

Pokud nyní přenásobíme rovnici koeficientem

$$\alpha := \frac{1}{\sqrt{A^2_{p,r} + A_{q,r}}} \ ,$$

pak získáme rovnici

$$\cos\varphi(\alpha A_{p,r}) - \sin\varphi(\alpha A_{q,r}) = 0 \ ,$$

s triviálním řešením

$$\begin{aligned} \cos\varphi &= \alpha A_{q,r} \\ \sin\varphi &= \alpha A_{p,r} \ , \end{aligned} \tag{4.16}$$

které splňuje podmínky oboru hodnot goniometrických funkcí $\cos\varphi, \sin\varphi \in \langle -1, 1\rangle$. Toto řešení lze přímo využít pro sestavení dané matice rotace, není potřeba vyčíslovat goniometrické nebo cyklometrické funkce.

Pokud provedeme na matici $A$ podobnostní transformaci

$$\tilde{A} = R^T_{p,q,r} A R_{p,q,r} \ , \tag{4.17}$$

mění se prvky v řádcích $p, q$ a sloupcích $p, q$. Pokud transformace (4.17) nuluje prvek $[A]_{r,q}$, je také prvek na pozici $[A]_{q,r}$ vynulován. Navíc matice $A$ a $\tilde{A}$ jsou podobné.

**Příklad 4.38.** Nechť je dána matice

$$A = \begin{bmatrix} 4 & -1 & -1 & 2 \\ -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 \\ 2 & -1 & -1 & 4 \end{bmatrix} .$$

Nalezněte matici $B$, která má na pozici $[3,1]$ nulový prvek, tj. $[B]_{3,1} = 0$, a je s maticí $A$ podobná. Využijte vhodnou matici Givensových rotací.

*Řešení.* Nejdříve dosaďme do předpisů (4.16) dané hodnoty $r = 3, q = 1$ (dle pozice nulovaného prvku $[A]_{r,q}$) a libovolné $p \in \{2, 4\}$.
Při volbě $p = 2$ získáme

$$\begin{aligned} \sin\varphi &= -\frac{[A]_{3,1}}{\sqrt{[A]^2_{3,2}+[A]^2_{3,1}}} = \frac{1}{\sqrt{2}} \\ \cos\varphi &= -\frac{[A]_{3,2}}{\sqrt{[A]^2_{3,2}+[A]^2_{3,1}}} = -\frac{1}{\sqrt{2}} \end{aligned}$$

a dosadíme do předpisu matice Givensových rotací (4.3.2) na pozice $p, q$

$$R^{\varphi}_{2,1} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_{2,1,3} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .$$

Následně provedeme transformaci podobnosti (4.14)

$$B = R^T_{2,1,3} A R_{2,1,3} = \begin{bmatrix} 5 & 0 & 0 & -\frac{3}{\sqrt{2}} \\ 0 & 3 & \frac{2}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{2}{\sqrt{2}} & 4 & -1 \\ -\frac{3}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 & 4 \end{bmatrix} .$$

Při volbě $p = 4$ dopadne situace obdobně

$$R_{4,1,3} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}, \quad B = R_{4,1,3}^T A R_{4,1,3} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 4 & -1 & \frac{2}{\sqrt{2}} \\ 0 & -1 & 4 & \frac{2}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 6 \end{bmatrix}.$$

▲

Algoritmus Givensových rotací je pak založen na opakování podobnostních transformací

$$A_k = R_k^T A_{k-1} R_k, \quad k = 1, \dots m, \ A_0 = A \ ,$$

kde matice $R_k$ postupně nulují prvky mimo třídiagonály.
Hledaná matice podobnostní transformace má tedy tvar

$$R = R_1 R_2 R_3 \dots R_m \ .$$

Indexy $p, q, r$ v předpisu (4.17) je však nutno volit tak, aby algoritmus nezaplňoval prvky, které byly předchozích krocích již vynulované. K nulování je proto nutno přistupovat systematicky.
Na následujícím schématu je pořadovými čísly naznačen postup nulování na matici $\mathbb{R}^{6,6}$.

$$\begin{bmatrix} x & x & 1 & 2 & 3 & 4 \\ x & x & x & 5 & 6 & 7 \\ 1 & x & x & x & 8 & 9 \\ 2 & 5 & x & x & x & 10 \\ 3 & 6 & 8 & x & x & x \\ 4 & 7 & 9 & 10 & x & x \end{bmatrix}$$

Pak algoritmus převodu symetrické matice na třídiagonální pomocí Givensových rotací má tvar

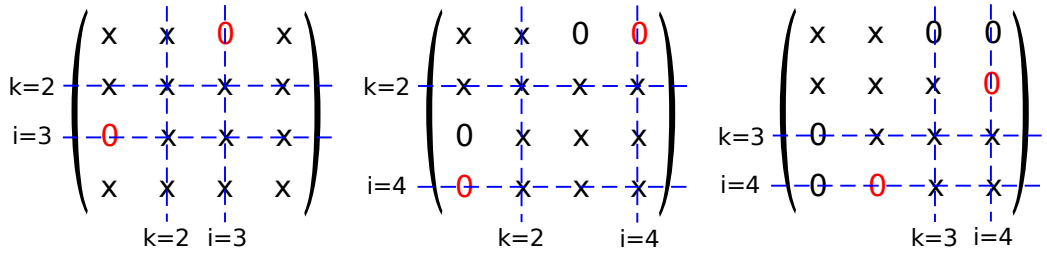
**Algoritmus 15** GIVENSOVA METODA

**Input:** $A$

1: $T := A$
2: $n :=$ velikost $A$
3: **for** $k := 2, \dots, n$ **do**
4: **for** $i := k + 1, \dots, n$ **do**
5: sestav $R_{k,i,k-1}$
6: $T := R_{k,i,k-1}^T T R_{k,i,k-1}$
7: **end for**
8: **end for**

**Output:** $T$

**Poznámka 4.39.** Transformace s maticí $R_{k,i,k-1}$ znamená, že je nulován prvek $[A]_{k-1,i}$ a přitom se mění pouze řádky a sloupce s indexy $k$ a $i$. Navíc ty řádky, které jsou již vynulovány, zůstávají nulové.




$A_1 = R^T_{2,3,1} A_0 R_{2,3,1}$ $A_2 = R^T_{2,4,1} A_1 R_{2,4,1}$ $A_3 = R^T_{3,4,2} A_2 R_{3,4,2s}$

V případě symetrické matice je výsledkem třídiagonální matice podobná původní matici. Hledat vlastní čísla pak můžeme iteračně algoritmem QR (viz Kapitola 4.3.1) nebo přes charakteristický polynom třídiagonální matice (viz Kapitola 4.1.3).

**Poznámka 4.40.** Algoritmus můžeme použít i na nesymetrické matice. Pak obdržíme matici v tzv. dolním *Hessenbergově tvaru*, t.j

$$\begin{bmatrix} x & x & 0 & 0 & \dots & 0 \\ x & x & x & 0 & \dots & 0 \\ x & x & x & x & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ x & x & x & x & \dots & x \\ x & x & x & x & \dots & x \end{bmatrix}$$

V tomto případě je také možné, podobně jako u třídiagonální matice, nalézt rekurzivní vztah pro nalezení charakteristického polynomu.

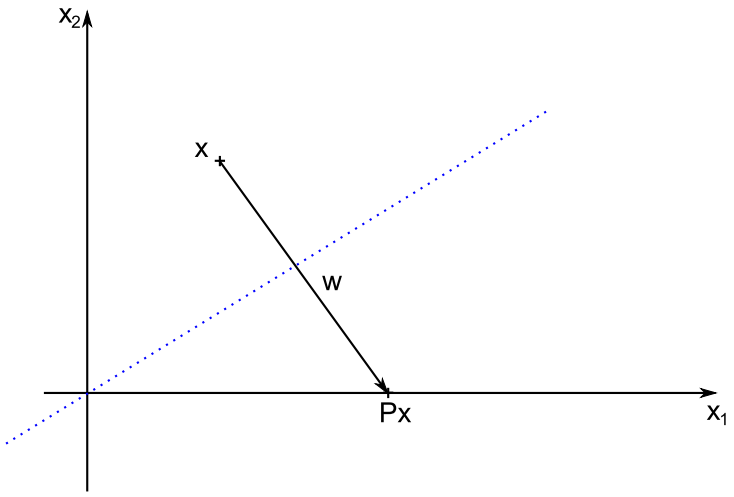
### 4.3.3 Householderova metoda

Obdobně jako matice rovinné rotace v případě Givensovy transfromace, můžeme k výpočtu QR rozkladu použít tzv. *matice rovinného zrcadlení*. Aplikaci matic rovinného zrcadlení taktéž říkáme *Householderovy transformace*.

**Definice 4.41.** *Maticí rovinného zrcadlení* rozumíme matici s předpisem

$$P := I - 2ww^T = \begin{pmatrix} 1-2w_1^2 & -2w_1w_2 & \dots & -2w_1w_n \\ -2w_2w_1 & 1-2w_2^2 & \dots & -2w_2w_n \\ \vdots & \vdots & \ddots & \vdots \\ -2w_nw_1 & -2w_nw_2 & \dots & 1-2w_n^2 \end{pmatrix} \tag{4.18}$$

kde $w^T w = 1$.




Obr. 4.5: Householderovo zrcadlení.

**Poznámka 4.42.** Matice rovinného zrcadlení je ortogonální a symetrická.

Ještě efektivněji než-li v případě Givensovy metody můžeme pro transformaci matice na třídiagonální tvar využít Housholderových transformací pomocí matic rovinného zrcadlení.

Vektory $w_j$ pro konstrukci matice $P_j = I - 2w_j w_j^T$ můžeme brát ve tvaru

$$w_j := [0, \ldots, 0, w_{j+1}^{(j)}, \ldots, w_n^{(j)}]^T \ .$$

Vhodnou volbou koeficientů $w^{(j)}$ lze docílit toho, že matice $A_j = P_j A_{j-1} P_j$ má všechny prvky

$$[A]_{i,k} = 0, \qquad \text{pro } i+1 > k, k = 1, \ldots, n$$

tzn. že v každém kroku se nulují prvky

$$[A]_{i,j}, i = j+2, \ldots, n \ .$$

Touto volbou je vektor se složkami

$$w_{j+1}^{(j)} = \frac{s_j + |[A_{j-1}]_{j+1,j}|}{2s_j}$$

$$w_k^{(j)} = \text{sgn } ([A_{j-1}]_{j+1,j}) \frac{[A_{j-1}]_{k,j}}{2w_{j+1}^{(j)} s_j}, \quad k = j+2, \ldots, n$$

kde

$$s_j \stackrel{\text{def}}{=} \sqrt{\sum_{k=j+1}^{n} ([A_{j-1}]_{k,j})^2}$$

a $[A_{j-1}]_{k,j}$ jsou prvky matice $A_{j-1}$.

**Algoritmus 16** HOUSEHOLDEROVA METODA

**Input:** $A$

$T := A$
$n :=$ velikost $A$
**for** $j := 1, \dots, n-2$ **do**
    $s := \sqrt{\sum_{k=j+1}^{n} [T]_{k,j}^2}$
    $[w]_{j+1} := (s + |[T]_{j+1,j}|)/(2s)$
    **for** $k := j+2, \dots, n$ **do**
        $[w]_k := \operatorname{sgn}([T]_{j+1,j}) \,.[T]_{k,j}/(2[w]_{j+1}s)$
    **end for**
    $P := I - 2ww^T$
    $T := P^T T P$
**end for**

**Output:** $T$

**Poznámka 4.43.** V případě nesymetrické matice dostáváme matici v Hessenbergově tvaru.
Postup dalšího řešení je pak stejný jako u Givensovy metody.

### 4.3.4 Lanczosova metoda

Stejně jako u předchozí Givensovy a Householderovy metody budeme hledat takovou matici ortogonální matici $Q$, která v podobnostní transformaci $T = Q^T A Q$ převede symetrickou matici $A$ na třídiagonální matici.

Předpokládejme nejdříve, že matici $Q$ máme k dispozici a označme

$$T := \begin{pmatrix} a_1 & b_1 & 0 & 0 & \dots & 0 \\ b_1 & a_2 & b_2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & b_{n-2} & a_{n-1} & b_{n-1} \\ 0 & \dots & 0 & 0 & b_{n-1} & a_n \end{pmatrix} .$$

Jelikož

$$T = Q^T A Q \quad \Rightarrow \quad QT = AQ \ , \tag{4.19}$$

tak dosazením

$$[s_1^Q,\dots,s_n^Q]\begin{pmatrix} a_1 & b_1 & 0 & 0 & \dots & 0\\ b_1 & a_2 & b_2 & 0 & \dots & 0\\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & b_{n-2} & a_{n-1} & b_{n-1}\\ 0 & \dots & 0 & 0 & b_{n-1} & a_n \end{pmatrix} = A[s_1^Q,\dots,s_n^Q]$$

a porovnáním $k$-tých řádků získáme rovnice

$$b_{k-1}s_{k-1}^Q + a_k s_k^Q + b_k s_{k+1}^Q = As_k^Q,\quad k=1,\dots,n-1\ , \tag{4.20}$$

přičemž $b_0 = 0, s_0^Q = o$.
Koeficienty $a_k, b_k$ můžeme určit následujícím způsobem.

- Přenásobíme rovnici (4.20) skalárně vektorem $s_k^Q$, získáme

  $$b_{k-1}(s_{k-1}^Q, s_k^Q) + a_k(s_k^Q, s_k^Q) + b_k(s_{k+1}^Q, s_k^Q) = (As_k^Q, s_k^Q)$$

  a díky ortogonalitě matice $Q$ (ortonormalitě vektorů $s_1^Q,\dots,s_n^Q$) se předpis zjednoduší na tvar

  $$a_k = (As_k^Q, s_k^Q). \tag{4.21}$$

- Přenásobíme rovnici (4.20) skalárně vektorem $s_{k+1}^Q$, získáme

  $$b_{k-1}(s_{k-1}^Q, s_{k+1}^Q) + a_k(s_k^Q, s_{k+1}^Q) + b_k(s_{k+1}^Q, s_{k+1}^Q) = (As_k^Q, s_{k+1}^Q)$$

  a díky ortonormalitě vektorů $s_1^Q,\dots,s_n^Q$ se předpis zjednoduší na tvar

  $$b_k = (As_k^Q, s_{k+1}^Q). \tag{4.22}$$

Otázkou zůstává, jak co nejjednodušeji vytvořit ortogonální matici $Q$.
Uvažujme rovnici (4.20) a upravme ji na tvar

$$s_{k+1}^Q = \frac{1}{b_k}((A - a_k I)s_k^Q - b_{k-1}s_{k-1}^Q). \tag{4.23}$$

Tento předpis lze považovat za jakýsi ortonormalizační proces, zvláště pokud označíme

$$r_k := (A - a_k I)s_k^Q - b_{k-1}s_{k-1}^Q \tag{4.24}$$

a volíme

$$b_k = \|r_k\|\ . \tag{4.25}$$

Pak dosazením do rovnice (4.23) získáme

$$s_{k+1}^Q = \frac{r_k}{\|r_k\|}. \tag{4.26}$$

Navíc lze dokázat, že volba (4.25) vyhovuje rovnici (4.22) a zkonstruovaná matice $Q$ je ortogonální.

Celý postup můžeme shrnout do následujícího algoritmu.

**Algoritmus 17** LANCZOSOVA METODA

**Input:** $A$

1: $n :=$ velikost $A$
2: $T :=$ nulová matice typu $n \times n$
3: $s_0^Q :=$ nulový vektor dimenze $n$
4: $r_1 :=$ libovolný jednotkový vektor dimenze $n$
5: $b_1 := 1$
6: **for** j := 1,. . . ,n **do**
7: $\quad s_j^Q := r_j / b_j$
8: $\quad a_j := (s_j^Q)^T A s_j^Q$
9: $\quad r_{j+1} := (A - a_j I) s_j^Q - b_j s_{j-1}^Q$
10: $\quad b_{j+1} := \|r_{j+1}\|$
11: $\quad [T]_{j,j} := a_j$
12: $\quad$ **if** $j < n$ **then**
13: $\quad\quad [T]_{j+1,j} := b_j$
14: $\quad\quad [T]_{j,j+1} := b_j$
15: $\quad$ **end if**
16: **end for**

**Output:** $T$

# Kapitola 5

# Interpolace

## Průvodce studiem


*Zřejmě nejznámější a nejčastěji používanou úlohou matematické analýzy je interpolace. V podstatě každý z nás se setkal s úlohou, kdy jsme museli odhadnout hodnotu funkce v bodě a přitom jsme znali pouze její hodnoty v krajních bodech. Tuto úlohu běžně řešíme při pohledu na ručičku ukazatele množství paliva v nádrži a snažíme se odhadnout kolik kilometrů ještě ujedeme. V této kapitole se tedy seznámíme se základními interpolačními metodami a ukážeme si, jak odhadnout chybu, kterou se při interpolaci dopouštíme.*

**Definice 5.1.** Nechť je dána funkce $f : \mathbb{R} \to \mathbb{R}$. Funkce $\varphi : \mathbb{R} \to \mathbb{R}$, která v bodech $x_0, \ldots, x_n \in D_f$ splňuje

$$\forall i = 0, \ldots, n : \varphi(x_i) = f(x_i) \tag{5.1}$$

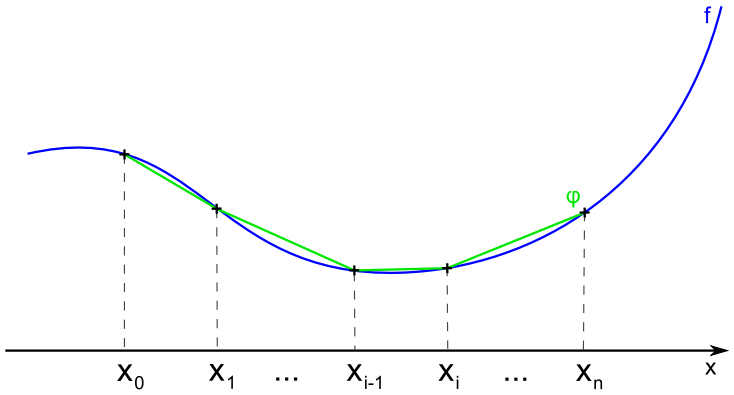
se nazývá interpolační funkce v uzlech $x_0, \ldots, x_n$ a podmínky nazýváme *interpolační podmínky.*

Důvod, proč nahrazujeme funkci $f$ funkcí $\varphi$ může být různý. Většinou požadujeme od hledané funkce $\varphi$ *lepší* vlastnosti, než má původní funkce $f$ - spojitost, diferencovatelnost, snažší vyčíslitelnost apod.
Jednou z nejžádanějších vlastností je, aby interpolační funkce byla prvkem vektorového prostoru konečné dimenze a tudíž, aby bylo možné ji vyjadřovat pomocí souřadnic. Takovou třídou funkcí je prostor všech polynomů stupně nejvýše $n$.

## 5.1 Interpolace polynomem

Polynomem $n$-tého stupně $p_n$ budeme rozumět funkci $p_n : \mathbb{R} \to \mathbb{R}$ s předpisem

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \sum_{i=0}^{n} a_i x^i$$




Obr. 5.1: Úloha interpolace.

kde $a_0, a_1, \ldots, a_n \in \mathbb{R}$ jsou konstantní koeficienty.
Za interpolační funkci budeme v dalším uvažovat právě polynom stupně $n$. Uvažujme hledanou interpolační funkci jako polynom stupně $n$.

Budeme dále uvažovat síť $n+1$ bodů

$$x_0 < x_1 < \cdots < x_n \ ,$$

ve kterých je zadána funkční hodnota

$$f(x_0), f(x_1), \ldots, f(x_n) \ .$$

Po interpolačním polynomu $p_n$ požadujeme splnění interpolačních podmínek

$$\forall i = 0, \ldots, n: \ p_n(x_i) = f(x_i) \ . \tag{5.2}$$

Jelikož předchozí interpolační podmínky reprezentují $n+1$ algebraických rovnic a v polynomu je právě $n+1$ neznámých koeficientů $a_i$, lze k nalezení těchto koeficientů přistupovat jako k řešení soustavy $n+1$ lineárních rovnic o $n+1$ neznámých. Konkrétně

$$\begin{array}{rcll}
p_n(x_1) & = & f(x_1) & \Rightarrow \quad a_0 + a_1 x_1 + a_2 x_1^2 + \ldots a_n x_1^n = f(x_1) \\
p_n(x_2) & = & f(x_2) & \Rightarrow \quad a_0 + a_1 x_2 + a_2 x_2^2 + \ldots a_n x_2^n = f(x_2) \\
 & \vdots & & \qquad\qquad\qquad\qquad\qquad\qquad\quad \vdots \\
p_n(x_n) & = & f(x_n) & \Rightarrow \quad a_0 + a_1 x_n + a_2 x_n^2 + \ldots a_n x_n^n = f(x_n)
\end{array}$$

maticově pak

$$Aa = b, \qquad \begin{array}{l} \text{kde } A_{i,j} = x_i^j, b_i = f(x_i), \\ i = 0, \ldots, n, \quad j = 0, \ldots, n \ . \end{array}$$

**Lemma 5.2.** *Nechť jsou zadány interpolační uzly*

$$x_0, x_1, \ldots, x_n \in \langle a, b \rangle$$
$$x_0 < x_1 < \cdots < x_n$$

*a příslušné funkční hodnoty* $f(x_0), f(x_1), \ldots, f(x_n) \in \mathbb{R}$.
*Pak existuje právě jeden interpolační polynom* $P_n$ *stupně* $n$ *takový, že*

$$\forall i = 0, \ldots, n : P_n(x_i) = f(x_i) \ .$$

*Důkaz.* Předpokládejme sporem, že existují dva různé interpolační polynomy $P_n$ a $Q_n$. Označme $R_n(x) = P_n(x) - Q_n(x)$. Jelikož $P_n$ a $Q_n$ splňují interpolační podmínky, tj.

$$\forall i = 0, \ldots, n : P(x_i) = Q(x_i) = f(x_i) \ ,$$

pak

$$\forall i = 0, \ldots, n : R(x_i) = P(x_i) - Q(x_i) = 0, \ ,$$

tedy polynom $n$-tého stupně $R_n$ má právě $n+1$ kořenů. Což je ve sporu se základní větou algebry (každý polynom $n$-tého stupně má maximálně $n$ různých reálných kořenů). □

**Algoritmus 18** INTERPOLACE POLYNOMEM

**Input:** $x_0, \ldots, x_n, f(x_0), \ldots, f(x_n)$

1: **for** $i := 0, \ldots, n$ **do**
2: **for** $j := 0, \ldots, n$ **do**
3: $[A]_{i,j} = x_i^j$
4: **end for**
5: $[b]_i := f(x_i)$
6: **end for**
7: $[a_0, \ldots, a_n]^T := a$ je řešení soustavy $Aa = b$

**Output:** $a_0, \ldots, a_n$

**Příklad 5.3.** Sestavte interpolační polynom 4-tého stupně, pokud je zadáno

$$\begin{aligned} x_0 &= 0, \quad f(x_0) = 1, \\ x_1 &= 2, \quad f(x_1) = 2, \\ x_2 &= 3, \quad f(x_2) = -1, \\ x_3 &= 4, \quad f(x_3) = 3, \\ x_4 &= 5, \quad f(x_4) = 2. \end{aligned}$$

*Řešení.* Budeme hledat polynom ve tvaru

$$P_4(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$$

a uvažujme 5 požadovaných interpolačních vlastností

$$\begin{array}{rcll} P_4(x_0) & = & f(x_0) & \Leftrightarrow \quad a_0 + a_1.0 + a_2.0^2 + a_3.0^3 + a_4.0^4 = 1 \\ P_4(x_1) & = & f(x_1) & \Leftrightarrow \quad a_0 + a_1.2 + a_2.2^2 + a_3.2^3 + a_4.2^4 = 2 \\ P_4(x_2) & = & f(x_2) & \Leftrightarrow \quad a_0 + a_1.3 + a_2.3^2 + a_3.3^3 + a_4.3^4 = -1 \\ P_4(x_3) & = & f(x_3) & \Leftrightarrow \quad a_0 + a_1.4 + a_2.4^2 + a_3.4^3 + a_4.4^4 = 3 \\ P_4(x_4) & = & f(x_4) & \Leftrightarrow \quad a_0 + a_1.5 + a_2.5^2 + a_3.5^3 + a_4.5^4 = 2 \end{array}$$

tedy pro nalezení koeficientů budeme řešit soustavu lineárních rovnic

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 8 & 16 \\ 1 & 3 & 9 & 27 & 81 \\ 1 & 4 & 16 & 64 & 256 \\ 1 & 5 & 25 & 125 & 625 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 3 \\ 2 \end{bmatrix} .$$

Řešením této soustavy je

$$[a_0, a_1, a_2, a_3, a_4]^T = [-\frac{19}{30}, \frac{103}{15}, -\frac{352}{15}, \frac{751}{30}, 1]^T ,$$

tedy algebraický interpolační polynom má předpis

$$P_4(x) = -\frac{19}{30}x^4 + \frac{103}{15}x^3 - \frac{352}{15}x^2 + \frac{751}{30}x + 1 .$$

▲

**Poznámka 5.4.** (O nešikovné volbě báze)
Matice v předchozím příkladě se nazývá *Vandermondova matice*, která má obecný tvar

$$V = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \dots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_m & \alpha_m^2 & \dots & \alpha_m^{n-1} \end{bmatrix}, \quad \text{nebo po složkách } [V]_{i,j} = \alpha_i^{j-1} ,$$

kde $\alpha_1, \dots, \alpha_m$ je posloupnost reálných konstant. O této matici je známo, že je velmi špatně podmíněná (viz [23]).
Volme například posloupnost $\alpha_i = i, i = 1, \dots, n$ (tj. interpolace na uzlech $x_i = \alpha_i$) a podívejme se na číslo podmíněnosti čtvercové Vandermondova matice ($m = n$) pro jednotlivá $n$.

| **n** | **cond(V)** |
|---|---|
| 1 | 1 |
| 2 | $6,85$ |
| 3 | $7,09 \,.\, 10^1$ |
| 4 | $1,17 \,.\, 10^3$ |
| 5 | $2,62 \,.\, 10^4$ |
| 6 | $7,31 \,.\, 10^5$ |
| 7 | $2,45 \,.\, 10^7$ |
| 8 | $9,52 \,.\, 10^8$ |
| 9 | $4,23 \,.\, 10^{10}$ |
| 10 | $2,11 \,.\, 10^{12}$ |

Jednodušeji řečeno - pokud bychom počítali interpolaci na uzlech

$$x_1 = 1,\ x_2 = 2,\ \ldots\ x_{10} = 10\ ,$$

pak největší prvek v matici soustavy pro výpočet požadovaných koeficientů by byl $[A]_{10,10} = 10^9$. Při výpočtu s tak velkým číslem nelze zaručit požadovanou přesnost. Navíc v praxi se vyskytují úlohy s interpolací na mnohem vyšším počtu uzlů.

Existuje mnohem rafinovanější pohled na na daný problém: úloha hledání interpolačního polynomu je úlohou hledání prvku vektorového prostoru všech polynomů stupně nejvýše $n$, který je jednoznačně určen požadovanými interpolačními vlastnostmi (viz Věta 5.2 o jednoznačnosti). Jelikož množina polynomů $\{1, x, x^2, \ldots x^n\}$ tvoří bázi vektorového prostoru, ze kterého pochází hledaný prvek, pak způsob hledání koeficientů $a_0, a_1, \ldots, a_n$ je de-facto stejný jako hledání souřadnic požadovaného polynomu vzhledem k této bázi.
Volba báze, kterou jsme použili v předchozím příkladě, však není právě ideální, neboť vede na soustavu s Vandermondovou maticí.
Očividně však existují mnohem šikovnější volby báze. Pak soustava pro hledání koeficientů je snadněji řešitelná, jak se ukáže v dalším výkladu u Lagrangeova a Newtonova interpolačního polynomu.

## 5.2 Lagrangeův interpolační polynom

Opět uvažujme hledanou interpolační funkci jako polynom stupně $n$ a značme ji $L_n(x)$. Dále uvažujme síť $n+1$ interpolačních uzlů $x_0, \ldots, x_n$ a množinu funkčních hodnot v těchto uzlech $f(x_0), \ldots, f(x_n)$, přičemž

$$x_0 < x_1 < \cdots < x_n\ .$$

Funkci $L_n$ můžeme vyjádřit ve tvaru

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

kde funkce $l_i$ jsou polynomy stupně $n$, které tvoří bázi prostoru všech polynomů stupně $n$.
Tuto bázi budeme šikovně volit tak, aby každý bázový polynom byl roven 1 pouze v jednom uzlu interpolace a v ostatních by byl roven 0. Takovou volbu báze lze snadno nalézt a popisuje ji následující definice.

**Definice 5.5.** Polynom $L_n(x) = \sum_{i=0}^{n} f(x_i) l_i(x)$ nazýváme *Lagrangeův interpolační polynom* jestliže $l_i$ jsou polynomy $n$-tého stupně, kde

$$\begin{aligned} l_i(x_k) &= 0 \quad \text{pro } k \neq i \\ l_i(x_i) &= 1 \end{aligned}$$

a jsou zadány předpisem

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

**Příklad 5.6.** Určete hodnotu Lagrangeova interpolačního polynomu v bodě $L_n(1)$ pokud jsou zadány hodnoty

$$\begin{aligned} L_n(0) &= 1, & L_n(4) &= 3, \\ L_n(2) &= 2, & L_n(5) &= 2. \\ L_n(3) &= -1, \end{aligned}$$

*Řešení.* Nejdříve označme

$$\begin{aligned} x_0 &= 0, & x_3 &= 4, \\ x_1 &= 2, & x_4 &= 5. \\ x_2 &= 3, \end{aligned}$$

a jelikož hledaná funkce $L_n$ je zadána v pěti bodech, budeme uvažovat polynom čtvrtého stupně ($n = 4$) s předpisem

$$L_4(x) = \sum_{i=0}^{4} f(x_i) l_i(x) = 1.l_0(x) + 2.l_1(x) + (-1).l_2(x) + 3.l_3(x) + 2.l_4(x)$$

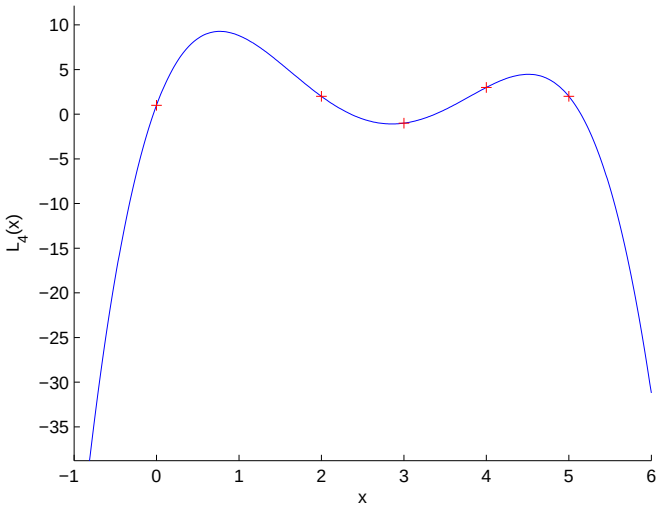
kde jednotlivé funkce $l_i(x)$ mají předpis

$$\begin{aligned} l_0(x) &= \frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)} = \frac{(x-2)(x-3)(x-4)(x-5)}{(0-2)(0-3)(0-4)(0-5)} \\ &= \frac{1}{120}x^4 - \frac{7}{60}x^3 + \frac{71}{120}x^2 - \frac{77}{60}x + 1 \\ l_1(x) &= \frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)} = \frac{(x-0)(x-3)(x-4)(x-5)}{(2-0)(2-3)(2-4)(2-5)} \\ &= -\frac{1}{12}x^4 + x^3 - \frac{47}{12}x^2 + 5x \end{aligned}$$

$$
\begin{aligned}
l_2(x) &= \frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)} = \frac{(x-0)(x-2)(x-4)(x-5)}{(3-0)(3-2)(3-4)(3-5)} \\
&= \frac{1}{6}x^4 - \frac{11}{6}x^3 + \frac{19}{3}x^2 - \frac{20}{3}x \\
l_3(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)} = \frac{(x-0)(x-2)(x-3)(x-5)}{(4-0)(4-2)(4-3)(4-5)} \\
&= -\frac{1}{8}x^4 + \frac{5}{4}x^3 - \frac{31}{8}x^2 + \frac{15}{4}x \\
l_4(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_3)} = \frac{(x-0)(x-2)(x-3)(x-4)}{(5-0)(5-2)(5-3)(5-4)} \\
&= \frac{1}{30}x^4 - \frac{3}{10}x^3 + \frac{13}{15}x^2 - \frac{4}{5}x \ .
\end{aligned}
$$

Dosazením a úpravou získáme polynom čtvrtého stupně

$$L_4(x) = -\frac{19}{30}x^4 + \frac{103}{15}x^3 - \frac{352}{15}x^2 + \frac{751}{30}x + 1 \ .$$




Obr. 5.2: Příklad interpolace Lagrangeovým polynomem.

Zbývá určit hodnotu interpolačního polynomu v bodě 1

$$L_4(1) = -\frac{19}{30} + \frac{103}{15} - \frac{352}{15} + \frac{751}{30} + 1 = \frac{44}{5} \ .$$

▲

**Poznámka 5.7.** Množina vektorů $\{l_0(x), l_1(x), \dots l_n(x)\}$ tvoří bázi vektorového prostoru všech polynomů nejvýše $n$-tého stupně a vektor $[f(x_0), \dots, f(x_n)]$ je vektor souřadnic hledaného polynomu vzhledem k této bázi. Narozdíl od algebraického interpolačního polynomu tedy není nutno řešit soustavu lineárních rovnic, stačí *pouze* zkonstruovat potřebnou bázi (tj. $l_i(x), i = 0, \dots, n$) a dosadit do předpisu dle Definice 5.5.

**Lemma 5.8.** *Nechť $\langle a,b\rangle$ je libovolný interval , který obsahuje body $x_0,\dots,x_n$. Nechť $f, f', \dots, f^{(n)} \in C(\langle a,b\rangle)$ a $\forall x \in (a,b)$ existuje $f^{(n+1)}(x)$. Pak pro chybu Lagrangeova interpolačního polynomu $L_n$ platí*

$$E(x) = f(x) - L_n(x) = (x-x_0)\dots(x-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}, \quad \xi \in (a,b), x \in (a,b) \ .$$

*Důkaz.* Zaveďme pomocnou funkci

$$u(x) := f(x) - L_n(x) - k\prod_{n+1}(x) \ , \tag{5.3}$$

kde symbolem $\prod_{n+1}(x)$ značíme součin

$$\prod_{n+1}(x) := (x-x_0)(x-x_1)\dots(x-x_n)$$

a $k \in \mathbb{R}$ je konstanta.
Funkce $u(x)$ má $n+1$ nenulových kořenů

$$\forall x_0, x_1, \dots x_n \in \langle a,b\rangle : u(x) = 0 \ .$$

Koeficient $k$ zvolíme tak, aby $(n+2)$-hým kořenem $u(x)$ byl libovolný, ale pevný bod $\bar{x} \in \langle a,b\rangle$, různý od kořenů $x_0, x_1, \dots x_n$. Jelikož $\bar{x}$ má být kořenem $u(x)$, platí

$$u(\bar{x}) = 0 \quad \Leftrightarrow \quad f(\bar{x}) - L_n(\bar{x}) - k\prod_{n+1}(\bar{x}) = 0$$

a protože $\prod_{n+1}(\bar{x}) \neq 0$, je

$$k = \frac{f(\bar{x}) - L_n(\bar{x})}{\prod_{n+1}(\bar{x})} \ . \tag{5.4}$$

Pak funkce

$$u(x) = f(x) - L_n(x) - \frac{f(\bar{x}) - L_n(\bar{x})}{\prod_{n+1}(\bar{x})}\prod_{n+1}(x)$$

má v intervalu $\langle a,b\rangle$ $n+2$ kořenů, a tedy bude rovna nule v krajních bodech intervalů

$$\langle x_0, x_1\rangle, \langle x_1, x_2\rangle, \dots, \langle x_i, \bar{x}\rangle, \langle \bar{x}, x_{i+1}\rangle, \dots, \langle x_{n-1}, x_n\rangle \ .$$

Pak dle Rolleovy věty (viz [14]) má derivace $u'(x)$ v intervalu $\langle a,b\rangle$ alespoň $n+1$ kořenů a druhá derivace $u''(x)$ má v tomto intervalu alespoň $n$ nulových bodů atd. Zobecníme-li toto pozorování, zjistíme, že v uvažovaném intervalu $\langle a,b\rangle$ má $(n+1)$-ní derivace $u^{(n+1)}(x)$ alespoň jeden kořen. Označme jej $\xi$ a platí tedy

$$u^{(n+1)}(\xi) = 0 \ . \tag{5.5}$$

Jelikož

$$L_n^{(n+1)}(x) = 0 \qquad \text{a} \qquad \prod_{n+1}^{(n+1)}(x) = (n+1)! \ ,$$

plyne z předpisu (5.3)

$$u^{(n+1)}(x) = f^{(n+1)}(x) - k(n+1)! \ .$$

Pro $x = \xi$ a užitím rovnice (5.5) získáme

$$0 = f^{(n+1)}(\xi) - k(n+1)! \ ,$$

odkud snadno vyjádříme

$$k = \frac{f^{(n+1)}(\xi)}{(n+1)!} \ . \tag{5.6}$$

Porovnáním rovnic (5.4) a (5.6) získáme rovnici

$$\frac{f(\bar{x}) - L_n(\bar{x})}{\prod_{n+1}(\bar{x})} = \frac{f^{(n+1)}(\xi)}{(n+1)!} \ ,$$

tj.

$$f(\bar{x}) - L_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{n+1}(\bar{x}) \ . \tag{5.7}$$

Jelikož $\bar{x}$ je libovolné (až na $\bar{x} \neq x_i$), můžeme rovnici (5.7) zapsat ve tvaru

$$E(x) = f(x) - L_n(x) = \prod_{n+1}(x) \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

kde $\xi$ závisí na $x$ a leží uvnitř intervalu $\langle a, b \rangle$. □

**Algoritmus 19** LAGRANGEŮV INTERPOLAČNÍ POLYNOM

**Input:** $x_0, \ldots, x_n, f(x_0), \ldots, f(x_n), x$

1: $L_n(x) := 0$
2: **for** $i := 1, \ldots, n$ **do**
3: $\quad l_i := \prod_{j=0, j\neq i}^{n} (x - x_j) / \prod_{j=0, j\neq i}^{n} (x_i - x_j)$
4: $\quad L_n(x) := L_n(x) + l_i f(x_i)$
5: **end for**

**Output:** $L_n(x)$

Nyní předpokládejme, že máme možnost zvolit si uzly interpolace. Zvolme je tak, aby chyba interpolace byla co nejmenší.

Existuje-li konstanta $M \in \mathbb{R}^+$ taková, že $\forall x \in (a,b) : |f^{(n+1)}(x)| \leqq M$, pak pro chybu interpolačního polynomu platí

$$|E_n(x)| \leqq \frac{M}{(n+1)!}|\mathrm{e}_n(x)| \leqq \frac{M}{(n+1)!} \max_{x\in\langle a,b\rangle} |\mathrm{e}_n(x)| \ ,$$

kde jsme použili označení

$$\mathrm{e}_n(x) := (x-x_0)\dots(x-x_n) \ .$$

To znamená, že vhodnou volbou sítě $x_0,\dots,x_n$ lze zlepšit interpolační funkci. Nejvhodnější volbou jsou kořeny tzv. *Čebyševových polynomů.*

**Lemma 5.9.** *Nechť jsou dány uzly interpolace $x_0, x_1, \dots, x_n \in \langle -1, 1\rangle$ jako kořeny Čebyševova polynomu, tj.*

$$T_n(x_i) := \cos((n+1)\arccos x_i) = 0, \quad \text{pro } i = 0,\dots,n \ .$$

*Pak pro odhad chyby interpolace polynomem n-tého stupně $P_n$ platí*

$$\forall \bar{x} \in \langle -1,1\rangle : |f(\bar{x}) - P_n(\bar{x})| \leqq \frac{M_{n+1}}{(n+1)!2^n} \ ,$$

*kde $M \in \mathbb{R}^+$ je konstanta taková, že*

$$\forall x \in (-1,1) : \quad |f^{(n+1)}(x)| \leqq M \ .$$

*Důkaz.* Viz [24]. □

**Poznámka 5.10.** Při interpolaci pomocí Čebyševových polynomů na obecném intervalu $\langle a,b\rangle$ užijeme lineární transformaci

$$x_k = \frac{1}{2}((b-a)z_k + b + a), \quad k = 0,1,\dots,n \ ,$$

kde $z_k \in \langle -1,1\rangle$ jsou kořeny Čebyševova polynomu $(n+1)$-ního stupně.
Pro odhad chyby interpolace pak bude platit

$$\forall \bar{x} \in \langle a,b\rangle : |f(\bar{x}) - P_n(\bar{x})| \leqq \frac{M_{n+1}}{(n+1)!}\frac{(b-a)^{n+1}}{2^{2n+1}} \ .$$

## 5.3 Newtonův interpolační polynom

Volba Lagrangeova tvaru interpolačního polynumu se jeví jako velmi výhodná z hlediska výpočetní stability. V některých případech však není ideální. Typicky se jedná o případy, kdy uzly interpolace jsou časové okamžiky, které v čase přibývají. V takovém případě je nutné pokaždé, když přibyde nový časový údaj znovu přepočítat

všechny bázové funkce. Tento nedostatek lze odstranit využitím tzv. *Newtonova tvaru* interpolačního polynomu.

Uvažujme nyní interpolační polynom tvaru

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

Opět budeme vyžadovat splnění interpolačních podmínek

$$i = 0, 1, \dots, n : \ N_n(x_i) = f(x_i) \ ,$$

kde $x_0, x_1, \dots, x_n \in \langle a, b\rangle$ jsou uzly interpolace a $f(x_0), f(x_1), \dots, f(x_n)$ jsou zadané funkční hodnoty v těchto uzlech.
Neznámé koeficienty $a_0, \dots, a_n$ získáme řešením následující soustavy

$$\begin{array}{lll} N_n(x_0) = & a_0 & = f(x_0) \\ N_n(x_1) = & a_0 + a_1(x_1 - x_0) & = f(x_1) \\ & \vdots & \\ N_n(x_n) = & a_0 + a_1(x_n - x_0) + \cdots + a_n(x_n - x_0)\dots(x_n - x_{n-1}) & = f(x_n) \end{array}$$

Množina vektorů $\{(x - x_0)(x - x_1)\dots(x - x_i),\ i = 0, \dots, n-1\}$ tvoří bázi vektorového prostoru všech polynomů nejvýše $n$-tého stupně. Z předešlých úvah je vidět, že pro nalezení souřadnic hledaného polynomu v této bázi bude nutno vyřešit dolní trojúhelníkovou matici. Tuto soustavu však není nutné řešit pokaždé, když rozšíříme uzly interpolace o nové, které jsou větší než ty dosavadní.
V takovém případě stačí stávající soustavu rozšířit o nové rovnice a dopřednou substitucí dopočítat nové koeficienty. Původní rovnice nadále zůstávají v platnosti.
Historicky se však problém řešil i bez sestavování soustavy pomocí tzv. *poměrných diferencí*.

**Definice 5.11.** Poměrnými diferencemi 0. řádu nazýváme (v bodě $x_i$) hodnoty

$$f[x_i] = f(x_i), \quad i = 0, \dots, n$$

Poměrnými diferencemi 1. řádu nazýváme

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}, \quad i = 0, \dots, n-1$$

Poměrnými diferencemi k. řádu nazýváme

$$f[x_i, x_{i+1}, \dots, x_{i+k+1}] = \frac{f[x_{i+1}, \dots, x_{i+k+1}] - f[x_i, \dots, x_{i+k}]}{x_{i+k+1} - x_i}, i = 0, \dots, n-k-1$$

**Poznámka 5.12.** Poměrné diference lze zapsat do tabulky

$$\begin{array}{c|ccccccc}
x_0 & f(x_0)=f[x_0] & \searrow & & & & & \\
x_1 & f(x_1)=f[x_1] & \to & f[x_0,x_1] & & & & \\
\vdots & & & \ddots & & & & \\
x_{n-1} & f(x_{n-1})=f[x_{n-1}] & \searrow & & \searrow & & & \\
x_n & f(x_n)=f[x_n] & \to & f[x_{n-1},x_n] & \to & f[x_{n-2},x_{n-1},x_n] & \cdots \to & f[x_0,\ldots,x_n]
\end{array}$$

Tato tabulka dává návod, jak jednoduše rekurentně konstruovat poměrné diference bez nutnosti odvozování komplikovaných vzorců. Vypočtených poměrných diferencí lze pak výhodně využít pro sestavení *Newtonova interpolačního polynomu*.

**Definice 5.13.** Polynom

$$N_n(x) := f[x_0] + f[x_0,x_1](x-x_0) + \cdots + f[x_0,\ldots,x_n](x-x_0)\ldots(x-x_{n-1}) \ , \quad (5.8)$$

kde

$$x_0 < x_1 < \cdots < x_n \quad (5.9)$$

jsou uzly interpolace, nazýváme *Newtonův interpolační polynom s poměrnými diferencemi*.

**Poznámka 5.14.** Lze ukázat, že Newtonův interpolační polynom s poměrnými diferencemi splňuje podmínky interpolace (5.2).

**Algoritmus 20** NEWTONŮV INTERPOLAČNÍ POLYNOM

**Input:** $x_0,\ldots,x_n, f(x_0),\ldots,f(x_n), x$

**for** $i := 0,\ldots,n$ **do**
    $sum_1 := 0$
    $sum_2 := 1$
    **for** $j := 1,\ldots,i-1$ **do**
        $sum_1 := sum_1 + f[x_0,\ldots,x_j].sum_2$
        $sum_2 := sum_2.(x_i - x_j)$
    **end for**
    $f[x_0,\ldots,x_i] := (f(x_i) - sum_1)/sum_2$
**end for**

$sum_1 := 0$
$sum_2 := 1$
**for** $j := 1,\ldots,n$ **do**
    $sum_1 := sum_1 + f[x_0,\ldots,x_j].sum_2$
    $sum_2 := sum_2.(x - x_j)$
**end for**
$N_n(x) := sum_1$

**Output:** $N_n(x)$

**Příklad 5.15.** Nechť jsou dány uzly interpolace

$$\begin{array}{rclcl} x_0 & = & 0, & f(x_0) & = & 1, \\ x_1 & = & 2, & f(x_1) & = & 2, \\ x_2 & = & 3, & f(x_2) & = & -1, \\ x_3 & = & 4, & f(x_3) & = & 3, \\ x_4 & = & 5, & f(x_4) & = & 2. \end{array}$$

Sestavte Newtonův interpolační polynom s poměrnými diferencemi a určete hodnotu tohoto polynomu v bodě $\bar{x} = 1$.

*Řešení.* Nejdříve sestavíme tabulku poměrných diferencí pro polynom 5-tého stupně

$$\begin{array}{c|c|c|c|c|c} x_0 & f[x_0] & & & & \\ x_1 & f[x_1] & f[x_0,x_1] & & & \\ x_2 & f[x_2] & f[x_1,x_2] & f[x_0,x_1,x_2] & & \\ x_3 & f[x_3] & f[x_2,x_3] & f[x_1,x_2,x_3] & f[x_0,x_1,x_2,x_3] & \\ x_4 & f[x_4] & f[x_3,x_4] & f[x_3,x_4,x_5] & f[x_1,x_2,x_3,x_4] & f[x_0,x_1,x_2,x_3,x_4] \end{array}$$

kde jednotlivé poměrné diference vypočítáme užitím tabulky pro výpočet poměrných diferencí.

$$\begin{array}{c|c|c|c|c|c} x_0 = 0 & 1 & & & & \\ x_1 = 2 & 2 & 1/2 & & & \\ x_2 = 3 & -1 & -3 & -7/6 & & \\ x_3 = 4 & 3 & 4 & 7/2 & 7/6 & \\ x_4 = 5 & 2 & -1 & -5/2 & -2 & -19/30 \end{array}$$

Pak dosazením do předpisu Newtonova interpolačního polynomu s poměrnými diferencemi a po úpravě získáme polynom

$$N_4(x) = -\frac{19}{30}x^4 + \frac{103}{15}x^3 - \frac{352}{15}x^2 + \frac{751}{30}x + 1 \ .$$

Zbývá určit hodnotu interpolačního polynomu v bodě 1

$$N_4(1) = -\frac{19}{30} + \frac{103}{15} - \frac{352}{15} + \frac{751}{30} + 1 = \frac{44}{5} \ .$$

▲

### 5.3.1 Ekvidistantní uzly interpolace

Uvažujme nyní ekvidistantní síť interpolačních uzlů, tj. existuje konstanta $h \in \mathbb{R}^+$ taková, že

$$\forall i = 0, \ldots, n-1 : x_{i+1} - x_i = h, \quad x_i = x_0 + hi.$$

Tato volba sítě nám umožňuje výpočet poměrných diferencí významně zjednodušit. Místo poměrných diferencí totiž můžeme k sestavení Newtonova interpolačního polynomu využít tzv. *dopředných diferencí*.

**Definice 5.16.** Obyčejnou dopřednou diferencí k. řádu rozumíme

$$\triangle^k f = \sum_{j=0}^{n} (-1)^{n-j} \binom{n}{j} f(x_j)$$

Dopřednou diferencí 0. řádu rozumíme

$$\triangle^0 f_i = f(x_i)$$

Dopřednou diferencí 1. řádu rozumíme

$$\triangle^1 f_i = \triangle^0 f_{i+1} - \triangle^0 f_i = f(x_{i+1}) - f(x_i)$$

Dopřednou diferencí k. řádu rozumíme

$$\triangle^k f_i = \triangle^{k-1} f_{i+1} - \triangle^{k-1} f_i$$

**Poznámka 5.17.** Očividně

$$\triangle^k f_i = \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} f(x_{i+j}) \ .$$

**Poznámka 5.18.** Tabulka dopředných diferencí

$$\begin{array}{l|llllll}
x_0 & f_0 = \triangle^0 f_0 & \to & \triangle^1 f_0 & \to & \triangle^2 f_0 \ \cdots \to & \triangle^n f_0 \\
x_1 & f_1 = \triangle^0 f_1 & \nearrow & & \nearrow & & \\
\vdots & & & \vdots & & & \\
x_{n-1} & f_{n-1} = \triangle^0 f_{n-1} & \to & \triangle^1 f_{n-1} & \nearrow & & \\
x_n & f_n = \triangle^0 f_n & \nearrow & & & &
\end{array}$$

**Poznámka 5.19.** Lze ukázat, že na ekvidistantní síti uzlů $x_0, x_1, \ldots, x_n$ platí

$$f[x_0, \ldots, x_n] = \frac{\triangle^n f_0}{n! h^n} \ .$$

**Věta 5.20.** *Nechť* $x_i = x_0 + hi, i = 0, \dots, n$
$x = x_0 + ht, t \in \mathbb{R}_0^+$ *a nechť*

$$\binom{t}{j} := \frac{t!}{j!(t-j)} = \frac{t(t-1)\dots(t-j+1)}{j!}, \binom{t}{0} = 1 \quad \text{pro } t, j \in \mathbb{R}_0^+, t \geqq j \ .$$

*Pak polynom*

$$N_n^+(t) = f_0 + \binom{t}{1} \triangle f_0 + \dots + \binom{t}{n} \triangle^n f_0 \tag{5.10}$$

*je* Newtonův interpolační polynom s dopřednými diferencemi.

**Příklad 5.21.** Interpolujte funkci $f$ pomocí Newtonova interpolačního polynomu s dopřednými diferencemi na zadané ekvidistantní síti uzlů a příslušných funkčních hodnotách

$$\begin{array}{llll} x_0 = 0\,, & f(x_0) = 1 \\ x_1 = 1\,, & f(x_1) = 3 \\ x_2 = 2\,, & f(x_2) = 2 \\ x_3 = 3\,, & f(x_3) = 3 \\ x_4 = 4\,, & f(x_4) = 4\,. \end{array}$$

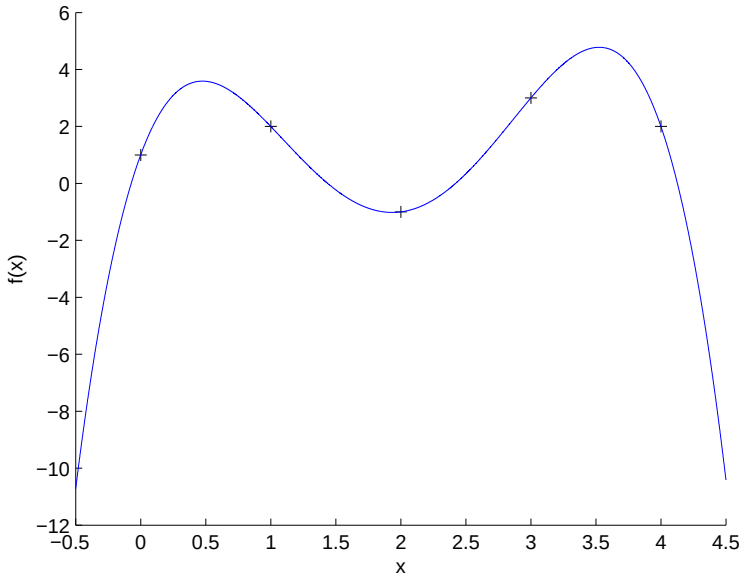
*Řešení.* Nejdříve vyplníme tabulku dopředných diferencí dle předpisů v definici 5.16.

| $x_i = i$ | $\triangle^0 f_i$ | $\triangle^1 f_i$ | $\triangle^2 f_i$ | $\triangle^3 f_i$ | $\triangle^4 f_i$ |
|---|---|---|---|---|---|
| 0 | $f_0 = 1$ | $2 - 1 = 1$ | $-3 - 1 = -4$ | $7 - (-4) = 11$ | $-12 - 11 = -23$ |
| 1 | $f_1 = 2$ | $-1 - 2 = -3$ | $4 - (-3) = 7$ | $-5 - 7 = -12$ | |
| 2 | $f_2 = -1$ | $3 - (-1) = 4$ | $-1 - 4 = -5$ | | |
| 3 | $f_3 = 3$ | $2 - 3 = -1$ | | | |
| 4 | $f_4 = 2$ | | | | |

Následně stačí dosadit do předpisu Newtonova interpolačního polynomu s dopřednými diferencemi (5.10).

$$\begin{aligned} N_4^+(x) &= f_0 + \binom{x}{1} \triangle^1 f_0 + \binom{x}{2} \triangle^2 f_0 + \binom{x}{3} \triangle^3 f_0 + \binom{x}{4} \triangle^4 f_0 \\ &= 1 + \binom{x}{1}.1 + \binom{x}{2}.(-4) + \binom{x}{3}.11 + \binom{x}{4}.(-24) \\ &= 1 + \tfrac{x}{1!} - \tfrac{4x(x-1)}{2!} + \tfrac{11x(x-1)(x-2)}{3!} - \tfrac{24x(x-1)(x-2)(x-3)}{4!} \\ &\dots \\ &= -\tfrac{23}{24}t^4 + \tfrac{91}{12}t^3 - \tfrac{433}{24}t^2 + \tfrac{149}{12}t + 1 \end{aligned}$$

Výsledný interpolační polynom je vykreslen na obrázku 5.3. ▲




Obr. 5.3: Příklad 5.21 - Newtonův interpolační polynom s dopřednými diferencemi.

### Pro zájemce:

Obdobně lze také definovat *zpětné* diference.

$$\nabla^0 f_i = f_i$$

$$\nabla^1 f_i = \nabla^0 f_i - \nabla^0 f_{i-1} = f_i - f_{i-1}$$

$$\nabla^k f_i = \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1}$$

V takovém případě má Newtonův interpolační polynom tvar

$$N_n^-(t) = f_n - \begin{pmatrix} -t \\ 1 \end{pmatrix} \triangle f_n + \cdots + (-1)^n \begin{pmatrix} -t \\ n \end{pmatrix} \triangle^n f_n \ .$$

## 5.4 Trigonometrická interpolace

Uvažujme interpolační funkci $T_n$ ve tvaru

$$T_n(x) \stackrel{\text{def}}{=} \frac{1}{2}A_0 + \sum_{i=1}^{n}(A_i \cos ix + B_i \sin ix), \quad i = 0, \ldots, 2n \ .$$

Opět budeme předpokládat splnění interpolačních podmínek

$$T_n(x_i) = f(x_i)$$

na ekvidistatntní síti interpolačních uzlů

$$x_i = \frac{2\pi i}{2n+1}, \quad i = 0, \ldots, 2n \tag{5.11}$$

Koeficienty interpolační funkce $T_n$ lze získat na základě následující věty.

**Věta 5.22.** *Polynom*

$$T_n(x) \overset{\text{def}}{=} \frac{1}{2}A_0 + \sum_{i=1}^{n}(A_i \cos ix + B_i \sin ix) \tag{5.12}$$

*kde*

$$A_i := \frac{2}{2n+1}\sum_{j=0}^{2n} f(x_j)\cos ix_j, i = 0, \ldots, n \tag{5.13}$$

$$B_i := \frac{2}{2n+1}\sum_{j=0}^{2n} f(x_j)\sin ix_j, i = 1, \ldots, n \tag{5.14}$$

*je interpolační polynom na síti interpolačních uzlů*

$$x_0, x_1, \ldots, x_n$$
$$x_{j+1} = x_j + h, j = 0, \ldots, n-1, \ h \in \mathbb{R}^+ \text{ const } .$$

*Důkaz.* Viz literatura [8]. □

**Algoritmus 21** TRIGONOMETRICKÁ INTERPOLACE

**Input:** $f, n, x$

1: **for** $i := 0, \ldots, n$ **do**
2: $x_i = (2\pi i)/(2n+1)$
3: **if** $i > 0$ **then**
4: $A_i := \frac{2}{2n+1}\sum_{j=0}^{2n} f(x_j)\cos ix_j$
5: $B_i := \frac{2}{2n+1}\sum_{j=0}^{2n} f(x_j)\sin ix_j$
6: **else**
7: $A_0 := \frac{2}{2n+1}\sum_{j=0}^{2n} f(x_j)$
8: **end if**
9: **end for**
10: $T_n(x) := \frac{1}{2}A_0 + \sum_{i=1}^{n}(A_i \cos ix + B_i \sin ix)$

**Output:** $T_n(x)$

**Příklad 5.23.** Uvažujme funkci

$$f(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{pro } x \in \langle 0, \pi/2) \\ 2 & \text{pro } x \in \langle \pi/2, \pi) \\ 3 & \text{pro } x \in \langle \pi, 3/2\pi) \\ 4 & \text{pro } x \in \langle 3/2\pi, 2\pi) \end{cases}$$

a její periodické rozšíření $f^P(x)$.
Interpolujte tuto nespojitou a $2\pi$-periodickou funkci na síti bodů (5.11). Volte různý počet uzlů a porovnejte interpolační funkci s původní funkcí.

*Řešení.* Volme například počet uzlů $2n+1 \stackrel{\text{def}}{=} 5$, pak $n = 2$. Budeme interpolovat na síti bodů (viz (5.11))

$$\begin{array}{rclcrcl} x_0 &=& 0, & & f(x_0) &=& 1 \\ x_1 &=& \frac{2}{5}\pi, & & f(x_1) &=& 1 \\ x_2 &=& \frac{4}{5}\pi, & & f(x_2) &=& 2 \\ x_3 &=& \frac{6}{5}\pi, & & f(x_3) &=& 3 \\ x_4 &=& \frac{8}{5}\pi, & & f(x_4) &=& 4\ . \end{array}$$

Nejdříve spočteme potřebné koeficienty dle (5.13) a (5.14).

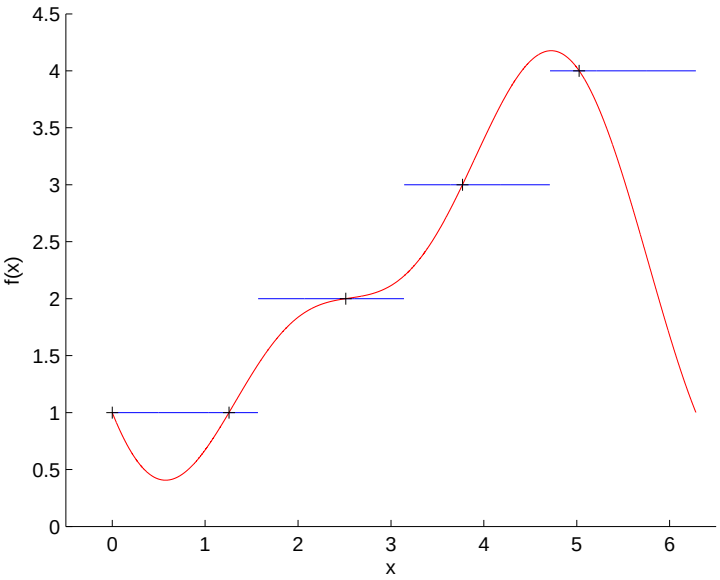
$$\begin{aligned} A_0 &= \tfrac{2}{5}\sum_{j=0}^{4} f(x_j) = \tfrac{22}{5} \\ A_1 &= \tfrac{2}{5}\sum_{j=0}^{4} f(x_j)\cos x_j = \tfrac{2}{5}\left(1 + 1.\cos\tfrac{2\pi}{5} + 2.\cos\tfrac{4\pi}{5} + 3.\cos\tfrac{6\pi}{5} + 4.\cos\tfrac{8\pi}{5}\right) \\ A_2 &= \tfrac{2}{5}\sum_{j=0}^{4} f(x_j)\cos 2x_j = \tfrac{2}{5}\left(1 + 1.\cos\tfrac{4\pi}{5} + 2.\cos\tfrac{8\pi}{5} + 3.\cos\tfrac{12\pi}{5} + 4.\cos\tfrac{16\pi}{5}\right) \\ B_1 &= \tfrac{2}{5}\sum_{j=0}^{4} f(x_j)\sin x_j = \tfrac{2}{5}\left(1.\sin\tfrac{2\pi}{5} + 2.\sin\tfrac{4\pi}{5} + 3.\sin\tfrac{6\pi}{5} + 4.\sin\tfrac{8\pi}{5}\right) \\ B_2 &= \tfrac{2}{5}\sum_{j=0}^{4} f(x_j)\sin 2x_j = \tfrac{2}{5}\left(1.\sin\tfrac{4\pi}{5} + 2.\sin\tfrac{8\pi}{5} + 3.\sin\tfrac{12\pi}{5} + 4.\sin\tfrac{16\pi}{5}\right)\ , \end{aligned}$$

které následně dosadíme do předpisu interpolačního polynomu (5.12)

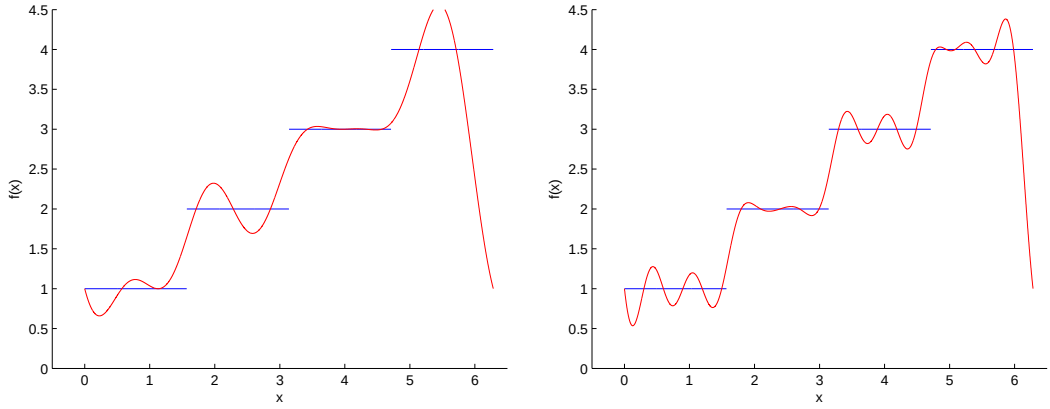
$$T_2(x) = \frac{11}{5} + A_1\cos x + B_1\sin x + A_2\cos 2x + B_2\sin 2x\ .$$

Jedna perioda výsledné funkce je vykreslena na Obrázku 5.4. Při použití většího počtu interpolačních uzlů získáme přesnější řešení, viz Obrázek 5.5 a 5.6.

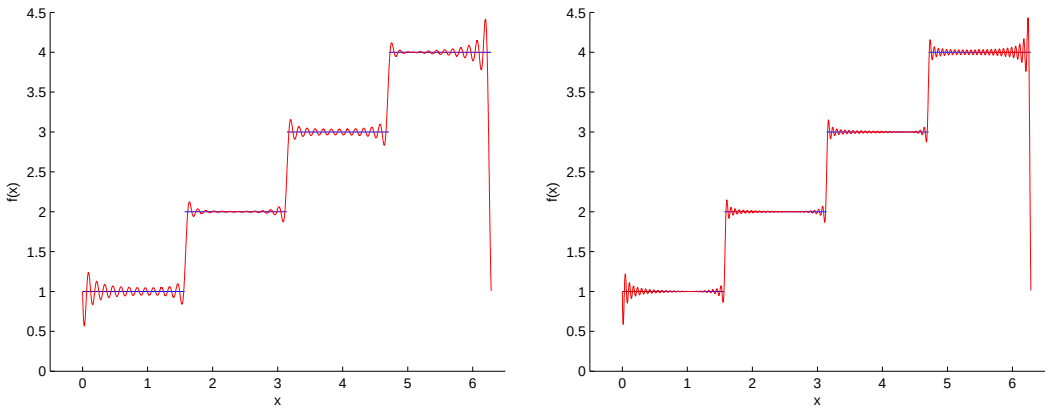
▲




Obr. 5.4: Příklad 5.23 - trigonometrická interpolace s volbou $n = 2$.



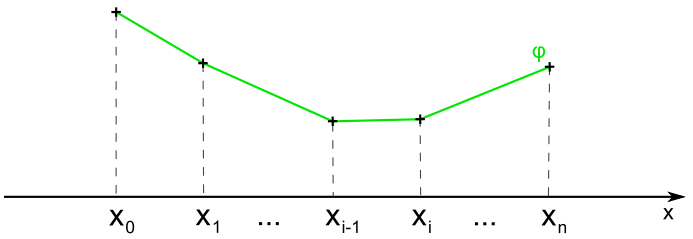

Obr. 5.5: Příklad 5.23 - trigonometrická interpolace s volbou $n = 5, 10$.




Obr. 5.6: Příklad 5.23 - trigonometrická interpolace s volbou $n = 50, 100$.

## 5.5 Interpolace lineárními spline funkcemi

Použití algebraických či trigonometrických interpolačních polynomů je velice omezené. V případě velkého počtu interpolačních uzlů totiž začné velmi narůstat řád interpolačního polynomu a polynom samotný se velmi „rozkmitá“.
V takovém případě se jako vhodnější ukazuje použití tzv. *spline funkcí*, které jsou postupně definovány jako polynomy nižších stupňů na intervalech ohraničených uzly interpolace. Nejjednodušším případem spline funkce je tzv. *lineární spline*, kterou můžeme charakterizovat jako spojitou po částech lineární funkci.




Obr. 5.7: Interpolace lineární spline funkcí.

**Definice 5.24.** Funkce $\varphi \in C$ se nazývá *lineární spline funkce*, jestliže na síti interpolačních uzlů $x_0, x_1, \ldots, x_n$ splňuje interpolační vlastnost a na jednotlivých disjunktních intervalech

$$\langle x_j, x_{j+1}\rangle, \quad j = 0, \ldots, n-1$$

je lineární.

Uvažujme tedy posloupnost interpolačních uzlů

$$a = x_0 < x_1 < \cdots < x_n = b \ ,$$

která generuje rozklad intervalu $\langle a, b\rangle$ na $n-1$ podintervalů

$$\langle a, b\rangle = \langle x_0, x_1\rangle \cup \langle x_0, x_1\rangle \cup \ldots \cup \langle x_{i-1}, x_i\rangle \cup \langle x_i, x_{i+1}\rangle \cup \ldots \cup \langle x_{n-1}, x_n\rangle \ .$$

Označme podinterval $I_i = \langle x_i, x_{i+1}\rangle, i = 0, \ldots, n-1$ a definujme na něm lineární funkci

$$\varphi_i(x) = a_i + b_i x, x \in \langle x_i, x_{i+1}\rangle \ .$$

Neznámé koeficienty $a_i, b_i \in \mathbb{R}$ lze snadno určit z požadované interpolační vlastnosti v krajních intervalu, tj. koeficienty jsou řešením soustavy

$$\begin{aligned} f(x_i) &= a_i + b_i x_i \\ f(x_{i+1}) &= a_i + b_i x_{i+1} \ . \end{aligned}$$

Po substituci $h_i = x_{i+1} - x_i$ (velikost uvažovaného intervalu) získáme

$$a_i = \frac{1}{h_i}\begin{vmatrix} f(x_i) & x_i \\ f(x_{i+1}) & x_{i+1} \end{vmatrix} \quad b_i = \frac{1}{h_i}\begin{vmatrix} 1 & f(x_i) \\ 1 & f(x_{i+1}) \end{vmatrix} .$$

Pak část interpolační funkce $\varphi_i$ na intervalu $I_i$ má tvar

$$\varphi_i(x) = \begin{cases} f(x_i) + \frac{f(x_{i+1})-f(x_i)}{h_i}(x - x_i) & \text{na } \langle x_i, x_{i+1}\rangle \\ 0 & \text{mimo } \langle x_i, x_{i+1}\rangle \ . \end{cases} \tag{5.15}$$

Pak interpolační funkci $\varphi$ lze vyjádřit jako součet

$$\varphi(x) = \sum_{i=0}^{n-1} \varphi_i(x), x \in \langle a, b\rangle \ .$$

**Algoritmus 22** INTERPOLACE LINEÁRNÍMI SPLINE FUNKCEMI

**Input:** $x_0, \ldots, x_n, f(x_0), \ldots, f(x_n)$

1: **for** $i := 0, \ldots, n-1$ **do**
2: $\quad a_i := f(x_i)$
3: $\quad h_i := x_{i+1} - x_i$
4: $\quad b_i := (f(x_{i+1} - f(x_i))/h_i$
5: **end for**

**Output:** $a_0, \ldots, a_n, \ b_0, \ldots, b_n$

**Příklad 5.25.** Nechť je dána funkce $f$ body a funkčními hodnotami

$$\begin{array}{llll} x_0 = 1 & & f(x_0) = 1 \\ x_1 = 3 & & f(x_1) = 4 \\ x_2 = 4 & & f(x_2) = 2 \\ x_3 = 5 & & f(x_3) = 0 \\ x_4 = 8 & & f(x_4) = 3 \\ x_5 = 9 & & f(x_5) = 3 \ . \end{array}$$

Na této síti interpolujte funkci $f$ pomocí lineární spline funkce $\varphi$.

*Řešení.* Nejdříve spočteme jednotlivé diference

$$\begin{array}{lll} h_0 = x_1 - x_0 = 2 \\ h_1 = x_2 - x_1 = 1 \\ h_2 = x_3 - x_2 = 1 \\ h_3 = x_4 - x_3 = 3 \\ h_4 = x_5 - x_4 = 1 \end{array}$$

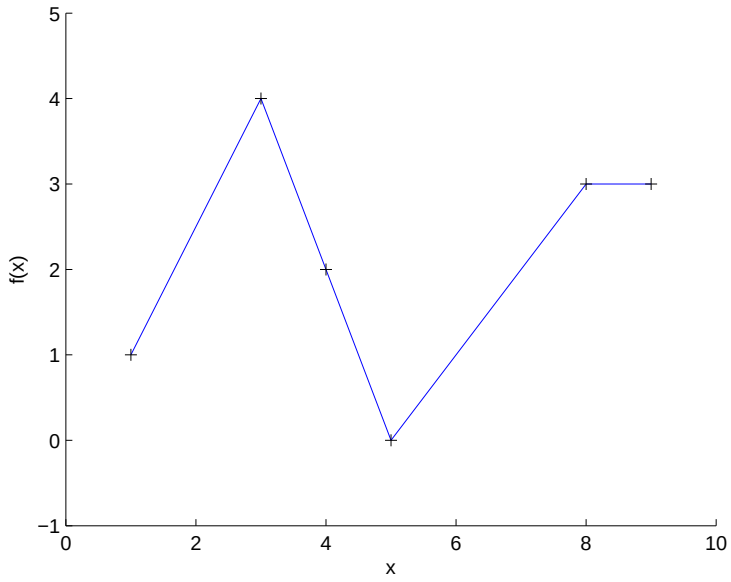
a jednotlivé části hledané interpolační funkce $\varphi$ zkonstruujeme pomocí předpisu (5.15)

$$\begin{aligned}
\varphi_0 &= f(x_0) + \tfrac{f(x_1)-f(x_0)}{h_0}(x - x_0) = \tfrac{3}{2}x - \tfrac{1}{2} \\
\varphi_1 &= f(x_1) + \tfrac{f(x_2)-f(x_1)}{h_1}(x - x_1) = -2x + 10 \\
\varphi_2 &= f(x_2) + \tfrac{f(x_3)-f(x_2)}{h_2}(x - x_2) = -2x + 10 \\
\varphi_3 &= f(x_3) + \tfrac{f(x_4)-f(x_3)}{h_3}(x - x_3) = x - 5 \\
\varphi_4 &= f(x_4) + \tfrac{f(x_5)-f(x_4)}{h_4}(x - x_4) = 3 \ .
\end{aligned}$$

Pak výsledná interpolační funkce má tvar

$$\varphi(x) = \begin{cases} \frac{3}{2}x - \frac{1}{2} & \text{pro } x \in \langle 1, 3) \\ -2x + 10 & \text{pro } x \in \langle 3, 5) \\ x - 5 & \text{pro } x \in \langle 5, 8) \\ 3 & \text{pro } x \in \langle 8, 9\rangle \ . \end{cases}$$

Výsledná funkce je vykreslena na obrázku 5.8.




Obr. 5.8: Příklad 5.25 - interpolace lineární spline funkcí.

▲

## 5.6 Interpolace kubickými spline funkcemi

Lineární spline představený v předchozí kapitole sice řeší problém zvyšujícího se stupně polynomu se zvyšujícím se počtem uzlů interpolace, avšak výsledná interpolační funkce díky nediferencovatelnosti v uzlech interpolace není hladká, ačkoliv

bychom to v řadě případů očekávali. V takovém případě je možné využít spline funkcí vyšších řádů, tj. takových, které jsou počástech kvadratické, kubické, kvintické atp. V takovém případě se dá zajistit i určitá hladkost výsledné interpolační funkce.
Jedním s nejčastěji používaných splinů je *kubický spline*. V následujícím textu si tento spline odvodíme.

**Definice 5.26.** Nechť jsou dány dvojice $(x_i, f_i), i = 0, \ldots, n$ a nechť $f(x_i) = f_i$. Funkci $\varphi(x)$ nazýváme *kubickou spline funkcí* jestliže

1. $\varphi \in C^2(\langle a, b\rangle)$
2. $\varphi$ je kubický polynom na $\langle x_i, x_{i+1}\rangle, \forall i = 0, \ldots, n-1$
3. $\varphi(x_i) = f_i, i = 0, \ldots, n$

**Poznámka 5.27.** Z definice je patrné, že kubická spline funkce $\varphi$ není určena jednoznačně.
Abychom jednoznačně určili interpolační funkci a její části na jednotlivých intervalech

$$\varphi(x) := \sum_{i=0}^{n-1} \varphi_i(x)$$
$$\varphi_i(x) := a_i + b_i x + c_i x^2 + d_i x^3 \quad \text{na } \langle x_i, x_{i+1}\rangle$$

je nezbytné pro jednotlivé intervaly určit $4n$ neznámých $a_i, b_i, c_i, d_i,\ i = 0, 1, \ldots, n-1$.
Z interpolačních podmínek

$$\varphi(x_i) = f_i, i = 0, \ldots, n$$

dostáváme $n+1$ rovnic. Ze spojitosti $\varphi, \varphi', \varphi''$ v bodech interpolace $x_i, i = 1, \ldots, n-1$ dostáváme dalších $3(n-1) = 3n-3$ rovnic.
Celkem tedy máme $4n$ neznámých a $4n-2$ rovnic.
Dvě podmínky tedy musíme dodat.
Například

1. $\varphi'(a) = f'_0, \varphi'(b) = f'_n$
2. $\varphi''(a) = f''_0, \varphi''(b) = f''_n$
3. $\varphi''(a) = 0, \varphi''(b) = 0$ (tzv. přirozený kubický spline)

Tyto podmínky nazýváme *okrajové podmínky*.

### 5.6.1 Konstrukce spline funkce

Na každém intervalu interpolace budeme hledat kubický spline ve tvaru

$$\varphi_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, i = 0, \ldots, n-1 \qquad (5.16)$$

$\varphi_i''$ je tedy polynom prvního stupně a tudíž $\varphi''$ je funkce po částech lineární a spojitá neboť $\varphi \in C^2(\langle a, b\rangle)$.
Označme $\varphi''(x_i) := M_i, i = 0, \ldots, n$ jako *i-tý moment spline funkce.*

Můžeme sestavit interpolaci po částech lineárními funkcemi (viz 5.5)

$$\varphi_i''(x) = \varphi_i''(x_i)\frac{x_{i+1} - x}{h_i} + \varphi_i''(x_{i+1})\frac{x - x_i}{h_i}, x \in \langle x_i, x_{i+1}\rangle, i = 0, \ldots, n-1 \ .$$

$$\varphi_i''(x) = M_i\frac{x_{i+1} - x}{h_i} + M_{i+1}\frac{x - x_i}{h_i}$$

Integrováním $\phi_i''$ dostáváme

$$\varphi_i'(x) = \int \varphi_i''(x)dx = -M_i\frac{(x_{i+1} - x)^2}{2h_i} + M_{i+1}\frac{(x - x_i)^2}{2h_i} + A_i$$

a následně

$$\varphi_i(x) = \int \varphi_i'(x)dx = M_i\frac{(x_{i+1} - x)^3}{6h_i} + M_{i+1}\frac{(x - x_i)^3}{6h_i} + A_i(x - x_i) + B_i \ .$$

Z podmínek interpolace plyne

$$\varphi_i(x_i) = f(x_i), \varphi_i(x_{i+1}) = f(x_{i+1}), i = 0, \ldots, n-1 \ ,$$

tedy

$$\varphi_i(x_i) = M_i\frac{h_i^2}{6} + B_i = f(x_i)$$

a

$$\varphi_i(x_{i+1}) = M_{i+1}\frac{h_i^2}{6} + A_ih_i + B_i = f(x_{i+1}) \ .$$

Řešením této soustavy 2 rovnic o 2 neznámých $A_i$, $B_i$ dostáváme

$$B_i = f(x_i) - M_i\frac{h_i^2}{6}$$

$$A_i = \frac{f(x_{i+1} - f(x)}{h_i} - \frac{h_i}{6}(M_{i+1} - M_i) \ .$$

**Poznámka 5.28.** Pomocí $M_i$ lze jednoznačně spočítat všechny funkce $\varphi_i$, tzn. můžeme určit $a_i, b_i, c_i, d_i$ následujícími předpisy

$$\begin{array}{ll} a_i = & \varphi_i(x_i) = f(x_i) \\ b_i = & \varphi_i'(x_i) = -M_i\frac{h_i}{2} + A_i = \frac{f(x_{i+1})-f(x_i)}{h_i} - \frac{2M_i+M_{i+1}}{6}h_i \\ c_i = & \frac{1}{2}\varphi_i''(x_i) = \frac{1}{2}M_i \\ d_i = & \frac{1}{6}\varphi_i'''(x_i^+) = \frac{M_{i+1}-M_i}{6h_i} \end{array} \tag{5.17}$$

kde $\varphi_i'''(x_i^+)$ je derivace $\varphi_i''$ zprava v $x_i$.

Nyní stačí pouze určit momenty $\varphi''(x_i), i = 0, \ldots, n$. K tomu využijeme spojitost $\varphi'$ v $x_i, i = 1, \ldots, n-1$, tzn.

$$\varphi_{i-1}'(x_i^-) = \varphi_i'(x_i^+), i = 1, \ldots, n-1 \ .$$

Vyjádříme si příslušné jednostrané derivace

$$\varphi_{i-1}'(x_i^-) = \frac{f(x_i) - f(x_{i-1})}{h_{i-1}} + \frac{h_{i-1}}{3}M_i + \frac{h_{i-1}}{6}M_{i-1} \ ,$$

$$\varphi_i'(x_i^+) = \frac{f(x_{i+1}) - f(x_i)}{h_i} - \frac{h_i}{3}M_i - \frac{h_i}{6}M_{i+1} \ .$$

Dosazením do podmínek spojitosti $\varphi'$ dostaneme rovnice

$$\begin{gathered} \frac{h_{i-1}}{6}M_{i-1} + \frac{h_{i-1}+h_i}{3}M_i + \frac{h_i}{6}M_{i+1} = \frac{f(x_{i+1})-f(x_i)}{h_i} - \frac{f(x_i)-f(x_{i-1})}{h_{i-1}}, \\ i \ = 1, \ldots, n-1 \end{gathered}$$

Čtenář si určitě všimnul, že se jedná o $n-1$ rovnic pro $n+1$ neznámých. Pro jednoznačnost řešení je proto doplníme libovolnou okrajovou podmínkou.
Označíme-li dále

$$\lambda_i = \frac{h_{i-1}}{h_{i-1}+h_i}, \quad \frac{h_i}{h_{i-1}+h_i} = 1 - \lambda_i = \mu_i \ , \tag{5.18}$$

$$g_i = \frac{6}{h_{i-1}+h_i}\left(\frac{f(x_{i+1}) - f(x_i)}{h_i} - \frac{f(x_i) - f(x_{i-1})}{h_{i-1}}\right) \ . \tag{5.19}$$

Dostáváme třídiagonální soustavu lineárních rovnic, jejíž řešením jsou hledané momenty kubického splinu.

$$\begin{array}{cccccc} 2M_1 & +\mu_1 M_2 & & & & = g_1 - \lambda_1 f_0'' \\ \lambda_2 M_1 & +2M_2 & +\mu_2 M_3 & & & = g_2 \\ & & & & & \vdots \\ & & \lambda_{n-2}M_{n-3} & +2M_{n-2} & +\mu_{n-1}M_{n-1} & = g_{n-2} \\ & & & \lambda_{n-1}M_{n-2} & +2M_{n-1} & = g_{n-1} - \mu_{n-1}f_n'' \end{array} \tag{5.20}$$

**Lemma 5.29.** *Nechť $f \in C^4(\langle a,b\rangle)$, $K$ je konstanta taková, že při dělení $\langle a,b\rangle$ na $h_i, i = 0,1,\ldots,n-1$ platí*

$$\frac{\max h_i}{\min h_i} \leqq K \ .$$

*Nechť $\varphi$ je kubická spline funkce pro funkci $f$ a nechť $\varphi$ splňuje okrajové podmínky. Pak pro $\forall x \in \langle a,b\rangle$ a $j = 0,1,2,3$ platí*

$$|f^{(j)}(x) - \varphi^{(j)}(x)| \leqq CKh^{4-j} \ ,$$

*kde $h = \max h_i$, $C$ je konstanta, která nezávisí na $x$ ani na dělení $\langle a,b\rangle$.*

**Algoritmus 23** INTERPOLACE KUBICKÝMI SPLINE FUNKCEMI

**Input:** $x_0,\ldots,x_n, f(x_0),\ldots,f(x_n), f''(x_0), f''(x_n)$

1: **for** $i := 0,\ldots,n-1$ **do**
2: $\quad h_i := x_{i+1} - x_i$
3: **end for**
4: **for** $i := 0,\ldots,n-1$ **do**
5: $\quad \lambda_i := h_i/(h_i + h_{i+1})$
6: $\quad \mu_i := 1 - \lambda_i$
7: $\quad g_i := 6/(h_{i-1} + h_i)\,((f(x_{i+1}) - f(x_i))/h_i - (f(x_i) - f(x_{i-1}))/h_{i-1})$
8: **end for**

9: $\tilde{M}$ je nulová matice typu $(n-1)\times(n-1)$
10: $\tilde{m}$ je nulový vektor dimenze $n-1$
11: $[\tilde{M}]_{1,1} := 2, \quad [\tilde{M}]_{1,2} := \mu_1$
12: $[\tilde{m}]_1 := g_1 - \lambda_1 f''(x_0)$
13: **for** $i := 2,\ldots,n-2$ **do**
14: $\quad [\tilde{M}]_{i,i-1} := \lambda_i$
15: $\quad [\tilde{M}]_{i,i} := 2$
16: $\quad [\tilde{M}]_{i,i+1} := \mu_i$
17: **end for**
18: $[\tilde{M}]_{n-1,n-1} := 2, \quad [\tilde{M}]_{n-1,n-2} := \lambda_{n-1}$
19: $[\tilde{m}]_{n-1} := g_{n-1} - \mu_{n-1} f''(x_n)$
20: $[M_1,\ldots,M_{n-1}]^T := m$ je řešení soustavy $\tilde{M}m = \tilde{m}$

21: **for** $i := 0,\ldots,n-1$ **do**
22: $\quad a_i := f(x_i)$
23: $\quad b_i := (f(x_{i+1} - f(x_i))/h_i - h_i(2M_i + M_{i+1})/6$
24: $\quad c_i := (1/2)M_i$
25: $\quad d_i := (M_{i+1} - M_i)/(6h_i)$
26: **end for**

**Output:** $a_0,\ldots,a_n, \ b_0,\ldots,b_n, \ c_0,\ldots,c_n, \ d_0,\ldots,d_n$

**Příklad 5.30.** Interpolujte funkci $f$ pomocí přirozené interpolace kubickými spline funkcemi na zadaných bodech a příslušných funkčních hodnotách

$$\begin{aligned} x_0 &= 1\,, & f(x_0) &= 1 \\ x_1 &= 2\,, & f(x_1) &= 3 \\ x_2 &= 3\,, & f(x_2) &= 2 \\ x_3 &= 4\,, & f(x_3) &= 3 \\ x_4 &= 5\,, & f(x_4) &= 4\,. \end{aligned}$$

*Řešení.* Nejprve je potřeba si všimnout, že zadané uzly jsou ekvidistatní, tj.

$$h_0 = h_1 = h_2 = h_3 = 1\,.$$

To nám vcelku zjednoduší výpočet. Nejdříve spočteme $\lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2, \mu_3$ dle předpisu (5.18)

$$\begin{aligned} \lambda_1 &= \tfrac{h_0}{h_0+h_1} = \tfrac{1}{2}\,, & \mu_1 &= 1-\lambda_1 = \tfrac{1}{2} \\ \lambda_2 &= \tfrac{h_1}{h_1+h_2} = \tfrac{1}{2}\,, & \mu_2 &= 1-\lambda_1 = \tfrac{1}{2} \\ \lambda_3 &= \tfrac{h_2}{h_2+h_3} = \tfrac{1}{2}\,, & \mu_3 &= 1-\lambda_1 = \tfrac{1}{2} \end{aligned}$$

a dosadíme do předpisu pomocných proměnných (5.19)

$$\begin{aligned} g_1 &= \tfrac{6}{h_0+h_1}\left(\tfrac{f(x_2)-f(x_1)}{h_1} - \tfrac{f(x_1)-f(x_0)}{h_0}\right) = -9 \\ g_2 &= \tfrac{6}{h_1+h_2}\left(\tfrac{f(x_3)-f(x_2)}{h_2} - \tfrac{f(x_2)-f(x_1)}{h_1}\right) = 6 \\ g_3 &= \tfrac{6}{h_2+h_3}\left(\tfrac{f(x_4)-f(x_3)}{h_3} - \tfrac{f(x_3)-f(x_2)}{h_2}\right) = 0\,. \end{aligned}$$

Pak lze sestavit soustavu (5.20) (jelikož se jedná o přirozený spline, tak $f_0'' = f_4'' = M_0 = M_4 = 0$)

$$\begin{array}{llll} 2M_1 & +\mu_1 M_2 & & = g_1 - \lambda_1 f_0'' \\ \lambda_2 M_1 & +2M_2 & +\mu_2 M_3 & = g_2 \\ & \lambda_3 M_2 & +2M_3 & = g_3 - \mu_3 f_4'' \end{array} \quad \Rightarrow \quad \begin{array}{llll} 2M_1 & +\frac{1}{2} M_2 & & = -9 \\ \frac{1}{2} M_1 & +2M_2 & +\frac{1}{2} M_3 & = 6 \\ & \frac{1}{2} M_2 & +2M_3 & = 0 \end{array}\,.$$

Řešením soustavy jsou hledané momenty

$$M_0 = 0, \quad M_1 = -\frac{159}{28}, \quad M_2 = \frac{33}{7}, \quad M_3 = -\frac{33}{28}, \quad M_4 = 0\,,$$

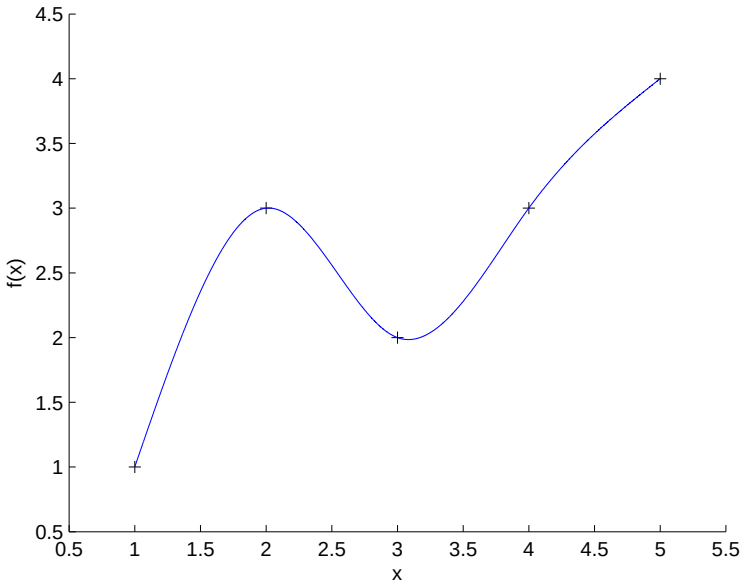
které lze následně dosadit do předpisů koeficientů pro kubické polynomy (5.17) na jednotlivých intervalech

$$\begin{aligned} a_0 &= f(x_0) = 1 \\ b_0 &= \tfrac{f(x_1)-f(x_0)}{h_0} - \tfrac{2M_0+M_1}{6} h_0 = \tfrac{165}{56} \\ c_0 &= \tfrac{1}{2} M_0 = 0 \\ d_0 &= \tfrac{M_1-M_0}{6h_0} = -\tfrac{53}{56} \end{aligned} \qquad \begin{aligned} a_2 &= f(x_2) = 2 \\ b_2 &= \tfrac{f(x_3)-f(x_2)}{h_2} - \tfrac{2M_2+M_3}{6} h_2 = -\tfrac{3}{8} \\ c_2 &= \tfrac{1}{2} M_2 = \tfrac{33}{14} \\ d_2 &= \tfrac{M_3-M_2}{6h_2} = -\tfrac{55}{56} \end{aligned}$$

$$\begin{aligned} a_1 &= f(x_1) = 3 \\ b_1 &= \tfrac{f(x_2)-f(x_1)}{h_1} - \tfrac{2M_1+M_2}{6} h_1 = \tfrac{3}{28} \\ c_1 &= \tfrac{1}{2} M_1 = -\tfrac{159}{56} \\ d_1 &= \tfrac{M_2-M_1}{6h_1} = \tfrac{97}{56} \end{aligned} \qquad \begin{aligned} a_3 &= f(x_3) = 3 \\ b_3 &= \tfrac{f(x_4)-f(x_3)}{h_3} - \tfrac{2M_3+M_4}{6} h_3 = \tfrac{39}{28} \\ c_3 &= \tfrac{1}{2} M_3 = -\tfrac{33}{56} \\ d_3 &= \tfrac{M_4-M_3}{6h_3} = \tfrac{11}{56}\,. \end{aligned}$$

Tedy výsledná interpolační funkce (5.16) má předpis

$$\varphi(x)\begin{cases} = 1 + \frac{165}{56}(x-1) - \frac{53}{56}(x-1)^3 & \text{pro } x \in \langle 1,2\rangle \\ = 3 + \frac{3}{28}(x-2) - \frac{159}{56}(x-2)^2 + \frac{97}{56}(x-2)^3 & \text{pro } x \in (2,3\rangle \\ = 2 - \frac{3}{8}(x-3) - \frac{33}{14}(x-3)^2 - \frac{55}{56}(x-3)^3 & \text{pro } x \in (3,4\rangle \\ = 3 + \frac{39}{28}(x-4) - \frac{33}{56}(x-4)^2 + \frac{11}{56}(x-4)^3 & \text{pro } x \in (4,5\rangle\ . \end{cases}$$




Obr. 5.9: Příklad 5.30 - interpolace kubickými spline funkcemi.

▲

# Kapitola 6

# Aproximace

## Průvodce studiem

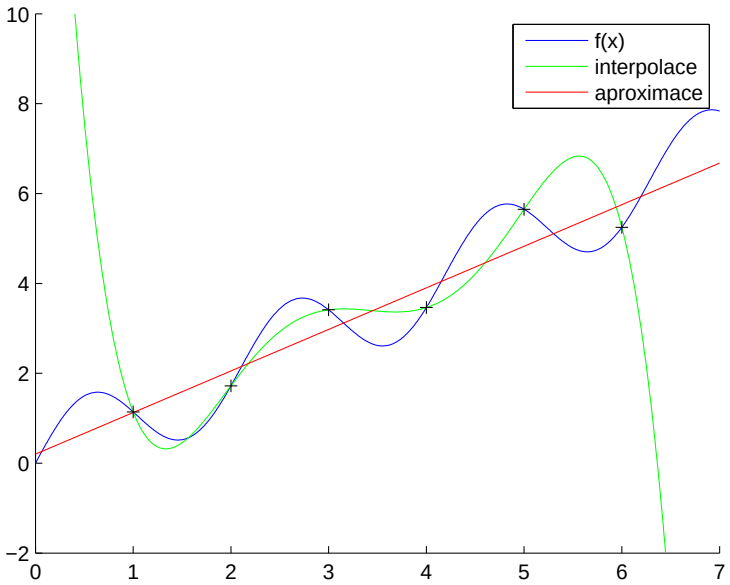
*Obdobně jako v případě interpolace se problematika aproximace funkcí zabývá metodami nalezení jednodušších funkcí, které jsou dosatečným přiblížením (aproximací) funkce jiné. V případě aproximačních metod však nebudeme vyžadovat ani splnění interpolačních podmínek, tzn. že výsledná aproximační funkce nemusí dokonce procházet známými funkčními hodnotami funkce aproximované.*
*V této kapitole se seznámíme s principy teorie aproximací funkcí a se ze základními aproximačními metodami.*

Aproximace si klade za cíl nahradit danou funkci $f$ z prostoru funkcí $\mathcal{F}$ nějakou jednodušší aproximační funkcí $\varphi$ z prostoru aproximačních funkcí $\Phi$. Na rozdíl od interpolační funkce však nevyžadujeme, aby se aproximační funkce $\varphi$ shodovala v předem daných bodech $x_i, i = 0, \ldots, m$ s původní funkcí $f$ (viz Obrázek 6.1).

Za podprostor $\Phi$ si obvykle vybíráme konečnědimenzionální podprostory funkcí, neboť se výsledné aproximační funkce dají dobře vyjádřit jako lineární kombinace vektorů báze. Typicky jsou to pak prostory polynomů, konečněrozměrné prostory trigonometrických funkcí apod. Je zřejmé, že chyba aproximace bude závislá především na volbě tohoto prostoru.
Důležitou otázkou v aproximační úloze je tedy otázka „míry vzdálenosti“ aproximační funkce $\varphi$ od původní funkce $f$. Čím je tato „vzdálenost“ menší, tím zřejmě lepší aproximační vlastnosti získáváme.
Za tímto účelem nám dobře poslouží prostory funkcí se skalárním součinem $(f, g)$ a normou $\|f\| = \sqrt{(f, f)}$ indukovanou tímto skalárním součinem. „Míru vzdálenosti“ $f$ od $\varphi$ pak můžeme měřit metrikou

$$\rho(f, \varphi) = \|f - \varphi\| = \sqrt{(f - \varphi, f - \varphi)}$$

kde $f \in \mathcal{F}$ a $\varphi \in \Phi$.




Obr. 6.1: Porovnání interpolačního polynomu s lineární aproximační funkcí.

Pro hledanou aproximační funkci $\varphi^* \in \Phi$ pak požadujeme, aby splňovala vlastnost

$$\rho(f, \varphi^*) \leqq \rho(f, \varphi) \quad \text{pro } \forall \varphi \in \Phi \ ,$$

tj.

$$\|f - \varphi^*\| \leqq \|f - \varphi\| \quad \text{pro } \forall \varphi \in \Phi \ .$$

Velmi často se v aproximačních úlohách volí za prostor funkcí $\mathcal{F}$ prostor $L_2(a, b)$ (tj. $\int_a^b f^2(x)dx < +\infty$) a za prostor aproximačních funkcí $\Phi$ můžeme například brát prostor všech polynomů stupně nejvýše $n$.
Na prostoru $L_2(a, b)$ zavádíme skalární součin

$$(f, g)_2 = \int_a^b f(x)g(x)dx$$

a normu

$$\|f\|_2 = \left( \int_a^b f^2(x)dx \right)^{\frac{1}{2}} \ .$$

Definujeme-li metriku pro vzdálenost funkcí

$$\rho(f, \varphi) \equiv \rho_2(f, \varphi) = \|f - \varphi\|_2 \ ,$$

pak aproximační úlohu můžeme předepsat jako minilalizační úlohu

$$\begin{cases} \text{Najdi } \varphi^* \in \Phi \text{ takové, že} \\ \|f-\varphi^*\|_2^2 \leqq \|f-\varphi\|_2^2 \quad \text{pro } \forall \varphi \in \Phi \ . \end{cases} \ . \tag{6.1}$$

Všimněme si, že místo normy $\|f-\phi\|_2$ jsme použili v minimalizační úloze (6.1) její kvadrát $\|f-\phi\|_2^2$. Tento tvar nemá žádný vliv na výsledek minimalizační úlohy a přitom je mnohem vhodnější pro úpravy minimalizované funkce neboť zatímco

$$\|f-\phi\|_2 = \sqrt{(f-\phi, f-\phi)_2} \ ,$$

tak

$$\|f-\phi\|_2^2 = (f-\phi, f-\phi) \ .$$

Pokud je tedy funkce $f$ zadána pouze na diskrétní síti $\{x_i, i=0,\ldots,m\}$, dostáváme funkci $f$ ve tvaru $m+1$ rozměrného aritmetického vektoru funkční hodnot

$$(f_0, \ldots, f_m) \in \mathbb{R}^m \ ,$$

kde $f_i \equiv f(x_i), i=0,\ldots,m$.
Množinu funkcí definovaných na konečných množinách bodů pak můžeme ztotožnit s vektorovým prostorem aritmetických vektorů $\mathbb{R}^{m+1}$, na kterém zavedeme skalární součin a normu ve tvaru

$$(f,g)_{2,m} := \sum_{i=0}^{m} f_i g_i, \quad \|f\|_{2,m} := \left( \sum_{i=0}^{m} f_i^2 \right)^{\frac{1}{2}} \ .$$

Původní aproximační úloha má pak tvar

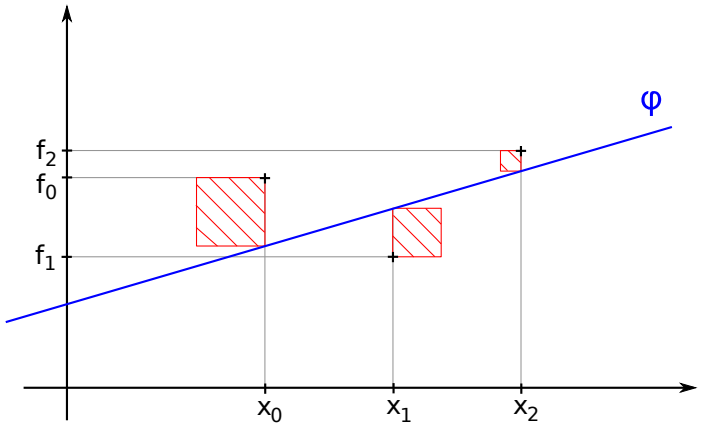
$$\begin{cases} \text{Najdi } \varphi^* \in \mathbb{R}^{m+1} \text{ tak, že} \\ \|f-\varphi^*\|_{2,m}^2 \leqq \|f-\varphi\|_{2,m}^2 \quad \text{pro } \forall \varphi \in \mathbb{R}^{m+1} \ . \end{cases} \ . \tag{6.2}$$

V obou těchto případech (6.1) a (6.2) se metoda aproximace nazývá *metoda nejmenších čtverců*. Její název vychází z geometrické interpretace této metody v diskrétním případě. Za aproximační polynom se vybírá ten polynom, jehož součet čtverců odchylek od hodnot $f_i$ je nejmenší.
Nejlépe lze princip metody pochopit z Obrázku 6.2.

Z výše uvedeného je patrné, že výsledný aproximační polynom závisí na volbě metriky pro vzdálenost funkcí. Změnou normy resp. skalárního součinu tak můžeme dostat jiné aproximační funkce, jejichž aproximační vlastnosti se budou lišit v závislosti na metrice.

Kromě metody nejmenších čtverců se pro aproximace taktéž často používá tzv. *stejnoměrná aproximace*, kdy za normu $f$ volíme

$$\|f\|_\infty := \max_{x \in \langle a,b \rangle} |f(x)|, \quad \text{kde} f \in C(\langle a,b \rangle) \ .$$




Obr. 6.2: Metoda nejmenších čtverců minimalizuje obsah čtverců.

Pokud si za třídu aproximačních funkcí vezmeme opět prostor všech polynomů stupně nejvýše $n$, pak aproximační úloha má tvar

$$\begin{cases} \text{Najdi } \varphi^* \in \Phi \text{ takové, že} \\ \min\limits_{\phi} \max\limits_{x\in\langle a,b\rangle} |f(x) - \phi(x)| \ . \end{cases}$$

Pokud funkce $f$ je opět známa pouze na konečné množině bodů, můžeme volit normu

$$\|f\|_{\infty,m} := \max_{i=0,\dots,m} |f_i|$$

a aproximační úloha má tvar

$$\begin{cases} \text{Najdi } \varphi^* \in \Phi \text{ takové, že} \\ \min\limits_{\varphi} \max\limits_{i=0,\dots,m} |f(x_i) - \varphi(x_i)| \ . \end{cases}$$

Pokud se zamyslíme nad rozdílem mezi metodou nejmenších čtverců a stejnoměrnou aproximací, tak výsledná „vzdálenost“ aproximační funkce nepřesáhne určitou mez, danou právě normou $\|f - \phi^*\|_\infty$, zatímco v případě metody nejmenších čtverců takováto mez není garantována, ale za to funkce $\phi^*$ se přibližuje k funkci v průměru po celé délce intervalu, na které aproximuje.

## 6.1 Metoda nejmenších čtverců

V předchozím textu jsme již definovali metodu nejmenších čtverců jako minimalizaci kvadrátu Eukleidovské normy chyby aproximace. Řekněme, že prostor $\Phi$ má uspořádanou bázi $(\varphi_0, \dots, \varphi_n)$. Pak $\forall \varphi \in \Phi$ platí

$$\varphi = c_0\varphi_0 + \cdots + c_n\varphi_n \ ,$$

kde $c_0, \ldots, c_n \in \mathbb{R}$ jsou souřadnice funkce $\varphi$ v dané bázi.
Budeme se tedy snažit nalézt takové $c_0^*, \ldots, c_n^* \in \mathbb{R}$, že

$$\|f - c_0^*\varphi_0 - \ldots - c_n^*\varphi_n\|_2^2 \leqq \|f - c_0\varphi_0 - \ldots - c_n\varphi_n\|_2^2$$

pro $\forall c_0, \ldots, c_n \in \mathbb{R}$.

Dostáváme tedy minimalizační úlohu

$$\min_{c_0,\ldots,c_n} \rho_2(f, c_0, \ldots, c_n)^2$$

s $\rho_2(f, c_0, \ldots, c_n)^2 = (f - c_0\varphi_0 - \ldots - c_n\varphi_n, f - c_0\varphi_0 - \ldots - c_n\varphi_n)_2$.
Pro tuto úlohu si můžeme předepsat nutnou podmínku existence minima ve tvaru

$$\frac{\partial \rho_2^2(c_0, \ldots, c_n)}{\partial c_k} = 0, k = 0, \ldots, n$$

Tedy

$$\rho_2^2(c_0, \ldots, c_n) = (f, f)_2 - 2\sum_{i=0}^{n} c_i(f, \varphi_i)_2 + \sum_{i=0}^{n}\sum_{j=0}^{n} c_i c_j(\varphi_i, \varphi_j)_2$$

a

$$\frac{\partial \rho_2^2(c_0, \ldots, c_n)}{\partial c_k} = -2(f, \varphi_k)_2 + 2\sum_{i=0}^{n} c_i(\varphi_i, \varphi_k)_2 = 0.$$

Nutná podmínka má tedy tvar

$$\sum_{i=0}^{n} c_i(\varphi_i, \varphi_k)_2 = (f, \varphi_k)_2, \quad k = 0, \ldots, n$$

což představuje soustavu lineárních rovnic s neznámými $c_0, \ldots, c_n$.

Tyto rovnice se nazývají *normální rovnice* a jejich maticový zápis vypadá následovně

$$Ac = b$$

$$A = \begin{bmatrix} (\varphi_0, \varphi_0)_2 & (\varphi_0, \varphi_1)_2 & \ldots & (\varphi_0, \varphi_n)_2 \\ (\varphi_1, \varphi_0)_2 & (\varphi_1, \varphi_1)_2 & \ldots & (\varphi_1, \varphi_n)_2 \\ \vdots & \vdots & \ddots & \vdots \\ (\varphi_n, \varphi_0)_2 & (\varphi_n, \varphi_1)_2 & \ldots & (\varphi_n, \varphi_n)_2 \end{bmatrix} = [(\varphi_i, \varphi_j)], \quad b = \begin{bmatrix} (f, \varphi_0)_2 \\ (f, \varphi_1)_2 \\ \vdots \\ (f, \varphi_n)_2 \end{bmatrix}.$$

Postačující podmínkou existence minima je pak positivní definitnost Hessiánu $D^2\rho_2^2(c_0, \ldots, c_n)$, což je právě matice $A$.
Tato matice se nazývá *Grammova* a je o ní známo, že je symetrická a positivně definitní pokud $\varphi_0, \ldots, \varphi_n$ jsou nezávislé, což v našem případě platí (neboť tvoří bázi prostoru $\Phi$).

Aproximační funkce nalezená touto metodou tedy má požadované minimalizační vlastnosti. V diskrétním případě získáváme stejné normální rovnice, ve kterých se pouze vyskytuje jiný skalární součin.
Konkrétně

$$A = [(\varphi_i, \varphi_j)_{2,m}] \, , b = [(f, \varphi_i)_{2,m}] \ ,$$

kde

$$(f, g)_{2,m} = \sum_{i=0}^{m} f_i g_i \ .$$

**Poznámka 6.1.** Je nutné poznamenat, že pro velká $n$ je matice $A$ zkonstruovaná výše uvedeným předpisem velmi špatně podmíněná pro běžné polynomiální báze jako je báze $\varphi_i(x) = x^i$.
To nás přivádí k využití ortogonálních systémů v aproximaci nejmenších čtverců (viz další podkapitola 6.2).

**Algoritmus 24** METODA NEJMENŠÍCH ČTVERCŮ

**Input:** $f$, $(\varphi_0, \ldots, \varphi_n)$

1: **for** $i := 0, \ldots, n$ **do**
2: **for** $j := 0, \ldots, n$ **do**
3: $[A]_{i,j} := (\varphi_i, \varphi_j)$
4: **end for**
5: $[b]_i := (f, \varphi_i)$
6: **end for**
7: $[c_0, \ldots, c_n]^T := c$ je řešení soustavy $Ac = b$

**Output:** $c_0, \ldots, c_n$

**Příklad 6.2.** Nechť funkce $f : \mathbb{R} \to \mathbb{R}$ je zadána v bodech

$$\begin{aligned} f(0) &= 1 \\ f(2) &= 2 \\ f(3) &= -1 \end{aligned}$$

Aproximujte tuto funkci polynomem prvního stupně $p \in \mathcal{P}_1$ metodou nejmenších čtverců.

*Řešení.* Budeme tedy hledat polynom $p$ ve tvaru

$$p(x) = c_0 + c_1 x.$$

V ideálním případě by hledaná funkce interpolovala dané body, tedy body by ležely na přímce a „míra vzdálenosti“ by byla nulová. To by ovšem znamenalo, že

hledané koeficienty řeší soustavu sestavenou pro hledání interpolačního polynomu (viz Příklad 5.3)

$$Ac = b \ ,$$

kde

$$A = \begin{bmatrix} x_0^0 & x_0^1 \\ x_1^0 & x_1^1 \\ x_2^0 & x_2^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}.$$

Samozřejmě, tato soustava nemá řešení (protože body na přímce neleží, řádky nejsou lineárně závislé).
Můžeme však nalézt „co nejlepší“ řešení, tj. takové, aby chyba řešení

$$err(c) := Ac - b$$

byla co nejmenší.
Ve smyslu nejmenších čtverců budeme uvažovat co nejmenší chybu s kvadrátem, tj.

$$c = \arg \ \min \|Ac - b\|^2.$$

Nejdříve upravme

$$\|Ac - b\|^2 = (Ac - b, Ac - b) = c^T A^T Ac - 2c^T A^T b + b^T b \ .$$

Jelikož minimum je nezávislé na konstantě $b^T b$, lze minimalizační úlohu přepsat do tvaru

$$\min \frac{1}{2} c^T A^T Ac - c^T A^T b \ .$$

Nutná podmínka minima (jedná se o kvadratickou funkci s symetrickou positivně definitní maticí, viz Věta 3.24) má tvar

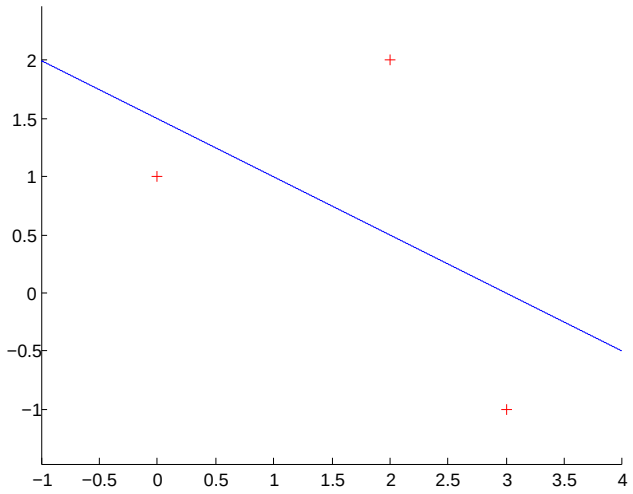
$$A^T Ac = A^T b \ .$$

Tedy místo původní interpolační soustavy budeme řešit tuto soustavu.

$$A^T A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 5 & 13 \end{bmatrix}, \quad A^T b = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Konkrétním řešením je $c_0 = 1.5, c_1 = -0.5$. Porovnání zadaných bodů a aproximační funkce je vykresleno na Obrázku 6.3. Tento spůsob řešení odpovídá výše uvedené teorii. Zvolíme-li standardní bázi vektorového prostoru $\mathcal{P}_1$, tj. $(\varphi_0, \varphi_1)$, kde

$$\begin{aligned} \varphi_0 &= x^0 \\ \varphi_1 &= x^1, \end{aligned}$$




Obr. 6.3: Příklad 6.2 - řešení, aproximace lineární funkcí.

pak odpovídající Grammova matice má tvar

$$\begin{bmatrix} (\varphi_0,\varphi_0)_2 & (\varphi_0,\varphi_1)_2 \\ (\varphi_1,\varphi_0)_2 & (\varphi_1,\varphi_1)_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{2} \varphi_0(x_i).\varphi_0(x_i) & \sum_{i=0}^{2} \varphi_0(x_i).\varphi_1(x_i) \\ \sum_{i=0}^{2} \varphi_1(x_i).\varphi_0(x_i) & \sum_{i=0}^{2} \varphi_1(x_i).\varphi_1(x_i) \end{bmatrix} =$$

$$= \begin{bmatrix} 0^0.0^0 + 2^0.2^0 + 3^0.3^0 & 0^0.0^1 + 2^0.2^1 + 3^0.3^1 \\ 0^1.0^0 + 2^1.2^0 + 3^1.3^0 & 0^1.0^1 + 2^1.2^1 + 3^1.3^1 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 5 & 13 \end{bmatrix}$$

a pravá strana normální rovnice

$$\begin{bmatrix} (f,\varphi_0)_2 \\ (f,\varphi_1)_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{2} f(x_i).\varphi_0(x_i) \\ \sum_{i=0}^{2} f(x_i).\varphi_1(x_i) \end{bmatrix} = \begin{bmatrix} 1.0^0 + 2.2^0 + (-1).3^0 \\ 1.0^1 + 2.2^1 + (-1).3^1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} .$$

▲

**Příklad 6.3.** Pro funkci $f(x) := \cos(x)$ nalezněte na intervalu $\langle 0, \pi \rangle$ aproximaci polynomem třetího stupně $\varphi(x)$ pomocí metody nejmenších čtverců.

*Řešení.* Hledaná funkce je polynom třetího stupně

$$\varphi(x) := c_0 + c_1 x + c_2 x^2 + c_3 x^3 ,$$

tedy úkolem je nalézt konstanty $c_0, c_1, c_2, c_3 \in \mathbb{R}$.
Uvažujme standardní bázi vektorového prostoru polynomů nejvýše třetího stupně

$$\begin{array}{lllllll} \varphi_0(x) & := & 1 & & \varphi_2(x) & := & x^2 \\ \varphi_1(x) & := & x & & \varphi_3(x) & := & x^3 \end{array}$$

a sestavme normální rovnice. Za tímto účelem bude nutno spočítat dané skalární součiny.

$$
\begin{aligned}
(\varphi_0,\varphi_0)_2 &= \int_0^\pi 1\cdot 1 \,\mathrm{dx} = [x]_0^\pi = \pi \\
(\varphi_1,\varphi_1)_2 &= \int_0^\pi x\cdot x \,\mathrm{dx} = \left[\tfrac{x^3}{3}\right]_0^\pi = \tfrac{\pi^3}{3} \\
(\varphi_2,\varphi_2)_2 &= \int_0^\pi x^2\cdot x^2 \,\mathrm{dx} = \left[\tfrac{x^5}{5}\right]_0^\pi = \tfrac{\pi^5}{5} \\
(\varphi_3,\varphi_3)_2 &= \int_0^\pi x^3\cdot x^3 \,\mathrm{dx} = \left[\tfrac{x^7}{7}\right]_0^\pi = \tfrac{\pi^7}{7} \\
(\varphi_0,\varphi_1)_2 &= \int_0^\pi 1\cdot x \,\mathrm{dx} = \left[\tfrac{x^2}{2}\right]_0^\pi = \tfrac{\pi^2}{2} \\
(\varphi_0,\varphi_2)_2 &= \int_0^\pi 1\cdot x^2 \,\mathrm{dx} = \left[\tfrac{x^3}{3}\right]_0^\pi = \tfrac{\pi^3}{3} \\
(\varphi_0,\varphi_3)_2 &= \int_0^\pi 1\cdot x^3 \,\mathrm{dx} = \left[\tfrac{x^4}{4}\right]_0^\pi = \tfrac{\pi^4}{4} \\
(\varphi_1,\varphi_2)_2 &= \int_0^\pi x\cdot x^2 \,\mathrm{dx} = \left[\tfrac{x^4}{4}\right]_0^\pi = \tfrac{\pi^4}{4} \\
(\varphi_1,\varphi_3)_2 &= \int_0^\pi x\cdot x^3 \,\mathrm{dx} = \left[\tfrac{x^5}{5}\right]_0^\pi = \tfrac{\pi^5}{5} \\
(\varphi_2,\varphi_3)_2 &= \int_0^\pi x^2\cdot x^3 \,\mathrm{dx} = \left[\tfrac{x^6}{6}\right]_0^\pi = \tfrac{\pi^6}{6}
\end{aligned}
$$

$$
\begin{aligned}
(\varphi_0,f)_2 &= \int_0^\pi 1\cdot\cos(x) \,\mathrm{dx} = [\sin(x)]_0^\pi = 0 \\
(\varphi_1,f)_2 &= \int_0^\pi x\cdot\cos(x) \,\mathrm{dx} = [x\sin(x)+\cos(x)]_0^\pi = \pi \\
(\varphi_2,f)_2 &= \int_0^\pi x^2\cdot\cos(x) \,\mathrm{dx} = [x^2\sin(x)+2x\cos(x)-2\sin(x)]_0^\pi = -2\pi \\
(\varphi_3,f)_2 &= \int_0^\pi x^3\cdot\cos(x) \,\mathrm{dx} = [x^3\sin(x)+3x^2\cos(x)-6x\sin(x)-6\cos(x)]_0^\pi \\
&= 12-3\pi^2
\end{aligned}
$$

Pak soustava normálních rovnic má tvar

$$
\begin{bmatrix}
\pi & \frac{\pi^2}{2} & \frac{\pi^3}{3} & \frac{\pi^4}{4} \\
\frac{\pi^2}{2} & \frac{\pi^3}{3} & \frac{\pi^4}{4} & \frac{\pi^5}{5} \\
\frac{\pi^3}{3} & \frac{\pi^4}{4} & \frac{\pi^5}{5} & \frac{\pi^6}{6} \\
\frac{\pi^4}{4} & \frac{\pi^5}{5} & \frac{\pi^6}{6} & \frac{\pi^7}{7}
\end{bmatrix}
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}
=
\begin{bmatrix} 0 \\ -2 \\ -2\pi \\ 12-3\pi^2 \end{bmatrix}
$$

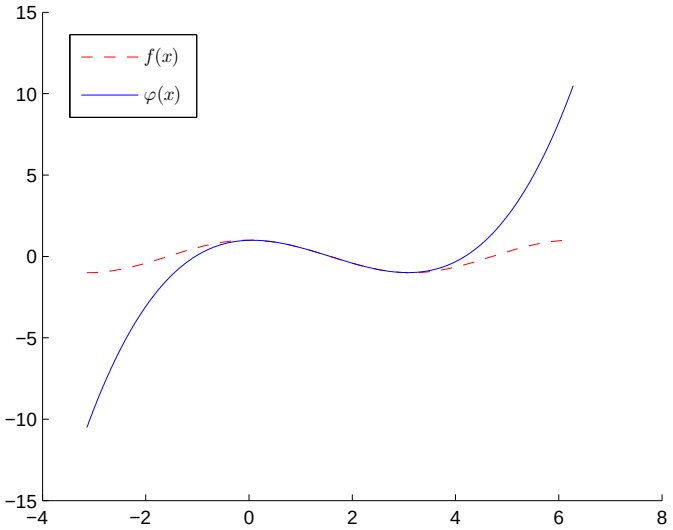
Tuto soustavu lze řešit numericky (viz kapitola 3)

$$c_0 \doteq 0.9910, \quad c_1 \doteq 0.0850, \quad c_2 \doteq -0.6836, \quad c_3 \doteq 0.1451\ .$$

Zbývá dosadit řešení do předpisu hledaného polynomu a získáme aproximační funkci

$$\varphi(x) = 0.9910 + 0.0850x - 0.6836x^2 + 0.1451x^3 \ .$$

Porovnání zadané funkce $f$ a aproximační funkce $\varphi$ je vykresleno na Obrázku 6.4. ▲




Obr. 6.4: Příklad 6.3 - řešení, aproximace metodou nejmenších čtverců.

## 6.2 Ortogonální systémy funkcí

V poznámce 6.1 jsme upozornili na skutečnost, že volba standardních polynomických bází jako je $\phi_i = x^i$ vede na velmi špatně podmíněné soustavy normálních rovnic. Této situaci se dá elegantně vyhnout pokud za bázi zvolíme ortogonální systém funkcí. Nejprve si tedy připomeňme, co rozumíme pod pojmem *ortogonální systém funkcí*.

**Definice 6.4.** Nechť je dán vektorový prostor $\Phi$ se skalárním součinem $(f, g)$, $f, g \in \Phi$. Říkáme, že množina $\{\varphi_0, \ldots, \varphi_b\} \subset \Phi$ je *ortogonální*, jestliže

$$(\varphi_i, \varphi_j) \begin{cases} = 0 & \text{pro } i \neq j \\ \neq 0 & \text{pro } i = j \end{cases}$$

Zobecněme skalární součin $(f, g)_2$ na prostoru $L_2(a, b)$ následovně

$$(f, g)_{2,\omega} = \int_a^b f(x)g(x)\omega(x)dx \ ,$$

kde $\omega(x)$ je tzv. *váhová funkce*, $\forall x \in (a,b) : \omega(x) > 0$.

Pro $f, g \in \mathbb{R}^{m+1}$ pak zobecňujeme Eukleidův skalární součin následovně

$$(f,g)_{2,\omega,m} = \sum_{i=0}^{m} f_i g_i \omega_i \ ,$$

kde $\omega_i > 0, i = 0, \ldots, m$.

Uveďme příklady některých ortogonálních systémů:

**Příklad 6.5.**

a) spojitý případ
Funkce $\varphi_j(x) = \cos jx, j = 0, 1, \ldots$ tvoří na $\langle 0, \pi \rangle$ ortogonální systém s $\omega(x) \equiv 1$

b) diskrétní případ
Funkce $\varphi_j(x) = \cos jx, j = 0, 1, \ldots, m$ tvoří na síti bodů $x_i = \frac{2i+1}{m+1}\frac{\pi}{2}, i = 0, \ldots, m$ s $\omega(x) \equiv 1$ ortogonální systém

**Příklad 6.6.** *Čebyševovy polynomy*

a) Polynomy $T_j(x) = \cos(j \arccos x), j = 0, 1, \ldots$ tvoří na intervalu $\langle -1, 1 \rangle$ ortogonální systém s $\omega(x) = (1 - x^2)^{-\frac{1}{2}}$. Tyto polynomy můžeme taktéž získat rekurentním vzorcem

$$T_0(x) \equiv 1, T_1(x) = x, T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x)$$

b) Polynomy $T_j(x) = \cos(j \arccos x), j = 0, 1, \ldots, m$ jsou ortogonální na síti $x_i = \cos\left(\frac{2i+1}{m+1}\frac{\pi}{2}\right), i = 0, \ldots, m$ s $\omega_i \equiv 1$

**Příklad 6.7.** *Legendrovy polynomy*
Polynomy

$$P_0(x) \equiv 1, P_j(x) = \frac{1}{2^j j!}\frac{d^j}{dx^j}\left[(x^2-1)^j\right], j = 1, 2, \ldots$$

tvoří na $\langle -1, 1 \rangle$ ortogonální systém s $\omega(x) \equiv 1$.
Tyto polynomy je taktéž možno získat rekurentním vzorcem

$$P_0(x) \equiv 1, P_1(x) = x, P_{j+1}(x) = \frac{2j+1}{j+1}xP_j(x) - \frac{j}{j+1}P_{j-1}(x) \ .$$

**Poznámka 6.8.** Polynomy v Příkladech 6.6 a 6.7 lze rozšířit z intervalu $\langle -1, 1 \rangle$ na interval $\langle a, b \rangle$ prostřednictvím transformace

$$y = \frac{x+1}{2}(b-a) + a, x \in \langle -1, 1 \rangle \ .$$

**Lemma 6.9.** *Nechť funkce $\varphi_0, \ldots, \varphi_n$ jsou ortogonální. Pak řešení normálních rovnic má tvar*

$$c_i = \frac{(f, \varphi_i)}{(\varphi_i, \varphi_i)}, i = 0, \ldots, n \ .$$

*Důkaz.* Vyplývá přimo z vlastností ortogonality, tj.

$$(\varphi_i, \varphi_j) \quad \text{pro } i \neq j$$

$$c_0(\varphi_i, \varphi_0) + \cdots + c_i(\varphi_i, \varphi_i) + \cdots + c_n(\varphi_i, \varphi_n) + = (f, \varphi_i)$$

$$c_i = \frac{(f, \varphi_i)}{(\varphi_i, \varphi_i)}$$

□

Z uvedeného Lemattu 6.9 tedy plyne, že při použití ortogonálního bázového systému funkcí není potřeba řešit soustavu normálních rovnic, neboť její matice je diagonální a řešení tudíž můžeme vypočíst přímo dosazením do rovnic

$$c_i = \frac{(f, \varphi_i)}{(\varphi_i, \varphi_i)} \ .$$

Tento postup pochopitelně zcela eliminuje jakýkoliv vliv numerické nestability na hledání aproximační funkce.

# Kapitola 7

# Numerická derivace a integrace

### Průvodce studiem

*Výpočet derivace a integrálu je další velmi hojně používanou matematickou úlohou. Sami víme, že se však v případě velmi složitých funkcí může jednat o velice náročnou úlohu. Proto mnohdy nezbývá, nežli využít numerických metod k jejich vyčíslení.*
*V následující kapitole se proto seznámíme se základními numerickými metodami, které umožní najít jejich aproximace.*

Numerická derivace i integrace je založena na prosté myšlence nahradit původní derivovanou funkci či integrovanou funkci interpolačním polynomem a poté derivovat resp. integrovat tento polynom. Pochopitelně čím přesnější bude interpolační polynom, tím přesnější aproximaci derivace či integrálu dostaneme.
Jak je známo z teorie interpolace, její přesnost můžeme řídit vhodnou volbou rozmístění interpolačních uzlů. Toho se pak dá využít i při přesnějším vyhodnocení numerické derivace či integrálu.

## 7.1 Numerická derivace

Jak již bylo zmíněno, numerická derivace spočívá v derivování interpolačního polynomu nahrazujícího derivovanou funkci $f$. K tomuto účelu tedy použijeme Newtonův interpolační polynom

$$f(x) \approx N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + + \cdots + \ f[x_0, \ldots, x_n](x - x_0) \ldots (x - x_n)$$

První derivace pak může být numericky vypočítána následovně

$$f'(x) \approx N_n' = f[x_0, x_1] + f[x_0, x_1, x_2]\frac{\partial}{\partial x}\left[(x - x_0)(x - x_1)\right] + \ldots$$
$$\cdots + f[x_0, \ldots, x_n]\frac{\partial}{\partial x}\left[(x - x_0) \ldots (x - x_{n-1})\right]$$

Označíme-li chybu interpolace

$$E_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\mathrm{e}(x)$$

kde

$$\begin{aligned} \mathrm{e}(x) &= (x-x_0)\dots(x-x_n) \\ \xi &\in \langle x_0, x_n\rangle \ . \end{aligned}$$

Pak chyba první derivace je

$$E'_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{d}{dx}\mathrm{e}(x) \ .$$

Předpis pro $k$-tou numerickou derivaci funkce $f$ pak má tvar

$$\begin{aligned} N_n^{(k)}(x) = f[x_0,\dots,x_k]\tfrac{d^k}{dx^k}\left[(x-x_0)\dots(x-x_{k-1})\right] + \dots \\ \dots + f[x_0,\dots,x_n]\tfrac{d^k}{dx^k}\left[(x-x_0)\dots(x-x_{n-1})\right] \end{aligned}$$

a její chyba má tvar

$$E_{n+1}^{(k)}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{d^k}{dx^k}\mathrm{e}(x) \ .$$

Pro výpočet numerické derivace pak obvykle volíme ekvidistantní síť uzlů, tj.

$$x_{i+1} - x_i = h, \quad i = 0,\dots,n-1.$$

Pak například pro volbu $x_1 = x$ získáváme interpolační body

$$\begin{aligned} x_0 &= x-h \\ x_1 &= x \\ x_2 &= x+h \\ &\dots \end{aligned}$$

Uveďme tedy několik základních formulí pro numerickou derivaci včetně jejich chyby: Vzorce pro první derivaci

1. Stupeň interpolačního polynomu $n = 1$
   Pro volbu
   $$\begin{aligned} x_0 &= x \\ x_1 &= x+h \end{aligned}$$
   dostáváme následující formuli pro numerickou derivaci
   $$f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{1}{2}hf''(\xi) \quad \xi \in \langle x, x+h\rangle, \ h > 0$$
   Volbu $x_0$ a $x_1$ můžeme však provést i následovně
   $$\begin{aligned} x_0 &= x-h \\ x_1 &= x \ . \end{aligned}$$
   Pak dostáváme následující vzorec pro první derivaci
   $$f'(x) = \frac{f(x)-f(x-h)}{h} + \frac{1}{2}hf''(\xi) \quad \xi \in \langle x-h, x\rangle, \ h > 0$$

2. Stupeň interpolačního polynomu $n = 2$
   Nejpřirozenější volbou pro uzly interpolace se v případě kvadratického interpolačního polynomu jeví

$$\begin{aligned} x_0 &= x - h \\ x_1 &= x \\ x_2 &= x + h \ . \end{aligned}$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6} h f'''(\xi) \quad \xi \in \langle x-h, x+h \rangle, \ h > 0$$

   Velmi často se stává, že je nutné počítat i jednostranné derivace. V takovém případě můžeme volit

$$\begin{aligned} x_0 &= x \\ x_1 &= x + h \\ x_2 &= x + 2h \end{aligned} \qquad \text{resp.} \qquad \begin{aligned} x_0 &= x - 2h \\ x_1 &= x - h \\ x_2 &= x \ . \end{aligned}$$

   Derivováním příslušných interpolačních polynomů pak dostáváme následující vzorce pro derivaci

$$\begin{aligned} f'(x) &= \tfrac{-3f(x)+4f(x+h)-f(x+2h)}{2h} + \tfrac{1}{3}h^2 f'''(\xi) \quad \xi \in \langle x, x+2h \rangle, \quad h > 0 \\ f'(x) &= \tfrac{3f(x)-4f(x-h)+f(x-2h)}{2h} + \tfrac{1}{3}h^2 f'''(\xi) \quad \xi \in \langle x-2h, x \rangle, \quad h > 0 \end{aligned}$$

Vzorce pro druhé derivace

1. Stupeň interpolačního polynomu $n = 2$

$$\begin{aligned} f''(x) &= \tfrac{f(x+h)-2f(x)+f(x-h)}{h^2} - \tfrac{1}{12}h^2 f^{IV}(\xi) \quad \xi \in \langle x-h, x+h \rangle \\ f''(x) &= \tfrac{f(x+2h)-2f(x+h)+f(x)}{h^2} - h f'''(\xi) \quad \xi \in \langle x, x+2h \rangle \\ f''(x) &= \tfrac{f(x)-2f(x-h)+f(x-2h)}{h^2} - h f'''(\xi) \quad \xi \in \langle x-2h, x \rangle \end{aligned}$$

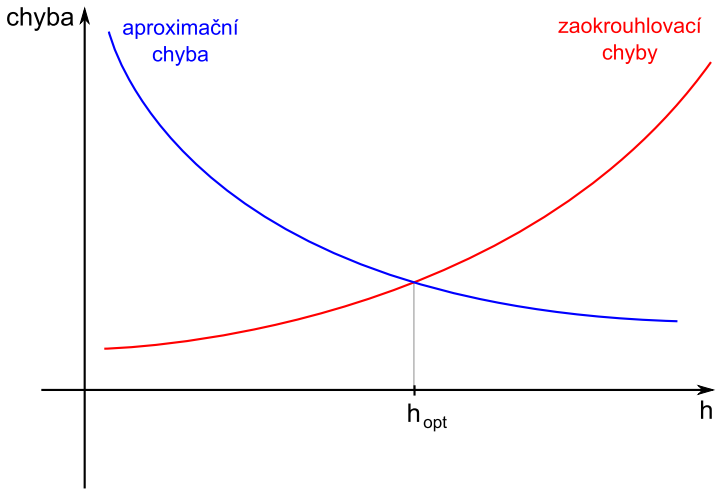
2. Stupeň interpolačního polynomu $n = 3$

$$f''(x) = \frac{12f(x) - 30f(x \pm h) + 24f(x \pm 2h) - 6f(x \pm 3h)}{6h^2} + O(h^3)$$

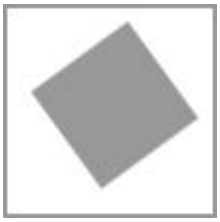
**Poznámka 7.1.** Je-li stupeň interpolačního polynomu shodný s řádem derivace (tj. $n = k$) pak

$$\begin{aligned} f^{(k)}(x) &\approx k! f[x_0, \ldots, x_n] \\ f^{(k)}(x) &\approx \tfrac{\Delta^k f_0}{h^k} \end{aligned}$$

**Poznámka 7.2.** Ze vzorců je patrné, že čím menší je $h$ tím je vzorec přesnější. Na druhé straně však vždy dělíme $h$ a pokud je $h$ příliš malé, roste zaokrouhlovací chyba. Optimální volba $h_{\text{opt}}$ je zobrazena na Obrázku 7.1.




Obr. 7.1: Volba optimální diskretizace.



Obr. 7.2: Příklad - detekce hran.

**Příklad 7.3.** Navrhněte jednoduchý gradientní algoritmus pro detekci hran na černobílém Obrázku 7.2.

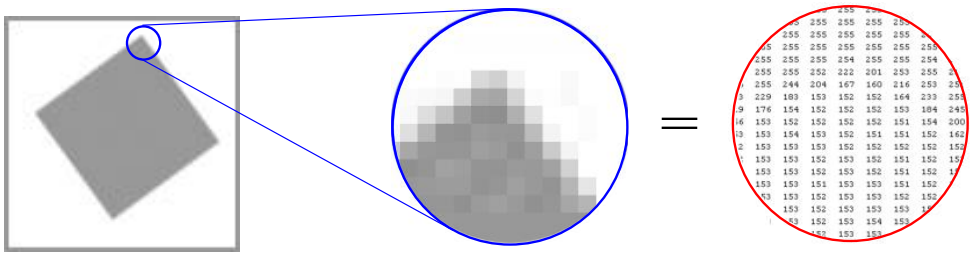
*Řešení.* Obrázek bude v počítači reprezentován jako dvourozměrné pole jasových hodnot v jednotlivých bodech (viz Obrázek 7.3).

Náš algoritmus nejprve spočítá změny jasové funkce ve směrech $x, y$ v jednotlivých obrazových bodech a pokud bude v těchto bodech změna dostatečná, pak se jedná o hranu. Jelikož změna funkce je definována gradientem, bude nutno spočítat numerický gradient. Využijeme-li předpis pro numerickou derivaci jednorozměrné reálné funkce

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

a aplikujeme-li jej v obou směrech naší obrazové funkce, získáme předpis

$$\begin{aligned} f_x(x,y) &\approx f(x+1,y) - f(x,y) \\ f_y(x,y) &\approx f(x,y+1) - f(x,y)\ . \end{aligned}$$

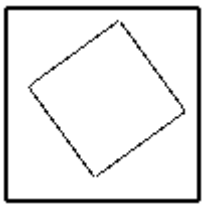


Obr. 7.3: Reprezentace obrazu v počítači.

Pak velikost změny je dána velikostí gradientu, tj.

$$\mathrm{e}(x,y) = \sqrt{(f_x(x,y))^2 + (f_y(x,y))^2}\ .$$

Následně *prahováním* určíme pouze relevantní body, t.j. budeme uvažovat pouze body s dostatečnou velikostí změny jasové funkce (jinak řečeno body $[x,y]$ pro které $\mathrm{e}(x,y) > \sigma$, kde $\sigma \in \mathbb{R}^+$ je pevně vhodně zvolená konstanta nebo-li *práh*).
Výsledek lze pozorovat na Obrázku 7.4.



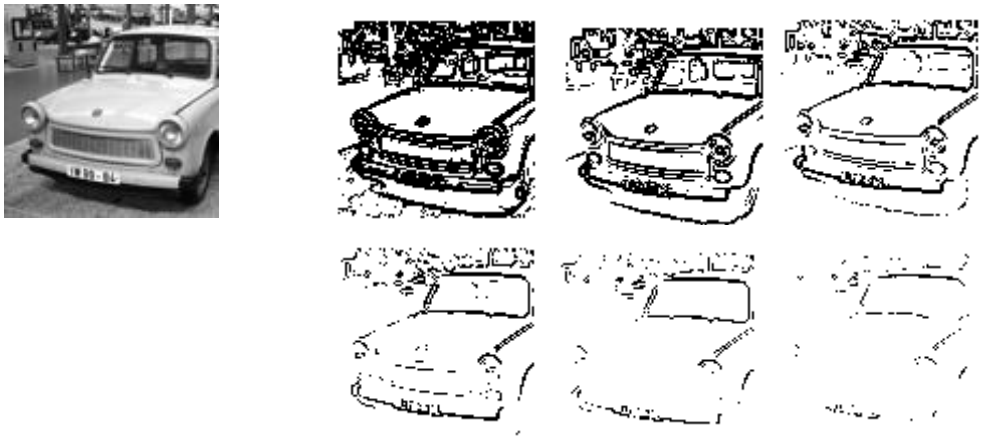
Obr. 7.4: Řešení - detekce hran.

Ačkoliv tento jednoduchý algoritmus vypadá na první pohled účinně, při komplikovanějších obrázcích je těžké určit vhodný práh (viz Obrázek 7.5). Od algoritmu využívajícího informace o jasové funkci z dvou sousedících bodů však ani nelze očekávat uspokojivější výsledky. Více informací o zpracování a analýze obrazu lze nalézt například v [17].

▲

## 7.2 Newtonovy-Cotesovy vzorce

Nejjednodušší numerická metoda pro numerickou integraci vychází z integrace Lagrangeova interpolačního polynomu na ekvidistantní síti bodů. ▮



Obr. 7.5: Příklad detekce hran na složitějším obrázku.

.

Pokud nahradíme

$$f(x) \approx \varphi_n(x) = \sum_{i=0}^{n} f(x_i) l_i(x)$$

získáme vzorec pro numerickou integraci

$$I(f) = \int_a^b \sum_{i=0}^{n} f(x_i) l_i(x) dx = \sum_{i=0}^{n} w_i f(x_i) \ , \tag{7.1}$$

kde

$$w_i := \int_a^b l_i(x) dx \ .$$

**Věta 7.4.** *Pro chybu integračního vzorce* (7.1) *platí*

$$E(f) = \begin{cases} = \frac{f^{(n+1)}(\xi)}{(n+1)!} \int_a^b w_{n+1}(x) dx & \text{pro liché } n \ , \\ = \frac{f^{(n+2)}(\xi)}{(n+2)!} \int_a^b x w_{n+1}(x) dx & \text{pro sudé } n \ . \end{cases}$$

*Důkaz.* Chybu získáme integrací chyby interpolačního polynomu, viz. [21]. □

Nyní si odvodíme základní Newtonovy-Cotesovy vzorce.

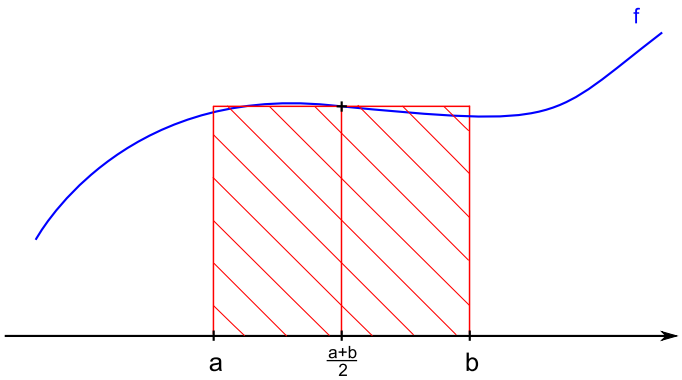
- *Obdélníkové pravidlo*

$$\begin{aligned} f &\approx \varphi_0, n = 0 \\ f(x) &\approx f\left(\tfrac{a+b}{2}\right) \\ I(f) &= \int_a^b f(x)dx \approx \int_a^b f\left(\tfrac{a+b}{2}\right) dx = (b-a) f\left(\tfrac{a+b}{2}\right) \\ I(f) &= (b-a) f\left(\tfrac{a+b}{2}\right) + E(f) \end{aligned}$$

Pro chybu obdélníkového pravidla $E(f)$ platí

$$E(f) = \frac{1}{24}(b-a)^3 f''(\xi) \ .$$

Z Obrázku 7.6 je patrná i geometrická interpretace obdélníkového pravidla i odkud pochází název této numerické metody.




Obr. 7.6: Obdélníkové pravidlo.

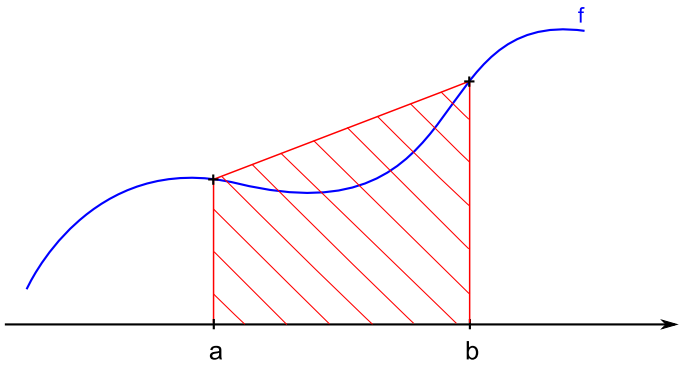
- *Lichoběžníkové pravidlo*

$$\begin{aligned} f &\approx \varphi_1, n = 1 \\ f(x) &\approx f(a) + \tfrac{f(b)-f(a)}{b-a}(x-a) \\ I(f) &= \int_a^b f(x)dx \approx \tfrac{(b-a)}{2}\left(f(a)+f(b)\right) \\ I(f) &= \tfrac{(b-a)}{2}\left(f(a)+f(b)\right) + E(f) \ , \end{aligned}$$

kde

$$E(f) = -\frac{1}{12}(b-a)^3 f''(\xi)$$

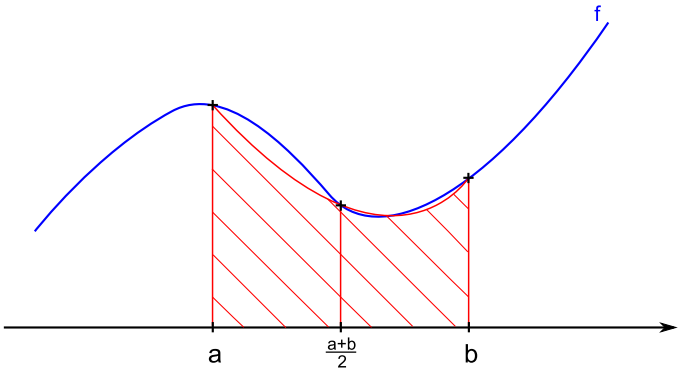
reprezentuje chybu vzorce.
Z Obrázku 7.7 je patrná i geometrická interpretace lichoběžníkového pravidla z níž je zřejmé i pojmenování integračního vzorce.




Obr. 7.7: Lichoběžníkové pravidlo.

- *Simpsonovo pravidlo*

$$\begin{aligned} f &\approx \varphi_2, n=2 \\ f(x) &\approx \frac{\left(x-\frac{a+b}{2}\right)(x-b)}{\left(a-\frac{a+b}{2}\right)(a-b)} f(a) + \frac{(x-a)(x-b)}{\left(\frac{a+b}{2}-a\right)\left(\frac{a+b}{2}-b\right)} f\left(\tfrac{a+b}{2}\right) + \frac{(x-a)\left(x-\frac{a+b}{2}\right)}{(b-a)\left(b-\frac{a+b}{2}\right)} f(b) \\ I(f) &= \frac{(b-a)}{6}\left(f(a)+4f\left(\tfrac{a+b}{2}\right)+f(b)\right)+E(f) \\ E(f) &= -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{\mathrm{IV}}(\xi) \end{aligned}$$



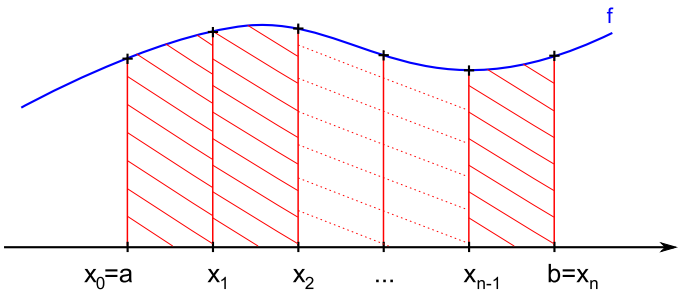

Obr. 7.8: Simpsonovo pravidlo.

Z předpisů chyb jednotlivých vorců je zřejmé, že pro velký integrační interval jsou vzorce díky velké chybě prakticky nepoužitelné. Jako vhodné řešení se nabízí rozdělení původního intrevalu $\langle a, b\rangle$ na $n$ podintervalů délky $h = \frac{b-a}{n}$.
Na jednotlivé podintervaly můžeme použít Newtonovy-Cotesovy vzorce a výsledný

integrál je pak roven součtu integrálů přes všechny podintervaly.

$$I(f) = \int_{x_0}^{x_1} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx$$

$$\int_a^b f(x)dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx, \quad x_i = x_0 + ih, i = 0, \ldots, n$$




Obr. 7.9: Složené pravidlo.

Výsledné integrační vzorce pak nazýváme *složenými* integračními vzorci a pro jednotlivé Newton-Cotesovy vzorce mají následující tvary:

- *Složené obdélníkové pravidlo*

$$\int_a^b f(x)dx = h \sum_{i=0}^{n-1} f\left(\frac{x_{i+1} + x_i}{2}\right) + \frac{1}{24}(b-a)h^2 f''(\xi)$$

- *Složené lichoběžníkové pravidlo*

$$\int_a^b f(x)dx = \frac{h}{2}\left(f(x_0) + f(x_n) + 2\sum_{i=1}^{n-1} f(x_i)\right) - \frac{1}{12}(b-a)h^2 f''(\xi)$$

- *Složené Simpsonovo pravidlo*

$$\begin{aligned}\int_a^b f(x)dx \quad = \quad & \tfrac{h}{3}\left(f(x_0) + f(x_n) + 4\sum_{i=1}^{m-1} f(x_{2i-1}) + 2\sum_{i=1}^{m-1} f(x_{2i})\right) \\ & -\tfrac{1}{180}(b-a)h^4 f^{\mathrm{VI}}(\xi), \quad n = 2m\end{aligned}$$

**Algoritmus 25** SLOŽENÉ OBDÉLNÍKOVÉ PRAVIDLO

**Input:** $f, a, b, n$

1: $h := (b-a)/n$
2: $I := 0$
3: **for** $i := 0, \ldots, n-1$ **do**
4: $\quad S_i := h.f(ih + h/2)$
5: $\quad I := I + S_i$
6: **end for**

**Output:** $I$

## 7.3 Gaussovy kvadraturní vzorce

Z teorie interpolace vyplývá, že chybu interpolačního polynomu můžeme minimalizovat vhodným rozložením uzlů interpolace. Těmito body pak jsou obvykle kořeny ortogonálních polynomů. Stejně tak, můžeme zpřesnit výpočet integrálů, pokud nebudeme integrovat interpolační polynom na ekvidistantní síti bodů, nýbrž na uzlech, které jsou kořeny ortogonálních polynomů.
Výsledné integrační vzorce nazýváme *Gaussovými kvadraturními vzorci* a o jejich přesnosti vypovídá následující věta.

**Věta 7.5.** *Nechť* $\{Q_j, j = 0, 1, \ldots\}$ *je soustava ortogonálních polynomů na intervalu* $\langle a, b\rangle$ *s volbou* $\omega(x) \equiv 1$. *Nechť* $x_i, i = 0, \ldots, n$ *jsou kořeny polynomu* $Q_{n+1}$. *Sestrojme Gaussův kvadraturní vzorec*

$$\int_a^b f(x)dx \approx \int_a^b \varphi(x)dx = \sum_{i=0}^{n} w_i f(x_i), \quad w_i = \int_a^b l_i(x)dx$$

*Pak pro každý polynom* $p_{2n+1} \in P_{2n+1}$ *stupně* $2n+1$ *platí*

$$\int_a^b p_{2n+1}(x)dx = \sum_{i=0}^{n} w_i p_{2n+1}(x_i) \ .$$

*Důkaz.* Viz [21]. □

**Poznámka 7.6.** Z výše uvedené věty vyplývá, že pokud za uzly integrace bereme kořeny ortogonálních polynomů, jsou kvadraturní formule přesné pro polynomy stupně $2n+1$ na rozdíl od Newtonových-Cotesových formulí, které jsou přesné pouze pro polynomy stupně $n$.

**Příklad 7.7.** *Gaussovy-Legendrovy kvadraturní vzorce*
Za systém ortogonálních polynomů bereme Legendrovy polynomy definované v kapitole 6.2.

1. $n = 0$

$$\int_{-1}^{1} f(x)dx = 2f(0) + \frac{1}{3}f''(\xi)$$

2. $n = 2$

$$\int_{-1}^{1} f(x)dx = f\left(-\sqrt{\frac{1}{3}}\right) + f\left(\sqrt{\frac{1}{3}}\right) + \frac{1}{135}f^{\mathrm{IV}}(\xi)$$

3. $n = 3$

$$\int_{-1}^{1} f(x)dx = \frac{5}{9}f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{\frac{3}{5}}\right) + \frac{1}{15750}f^{\mathrm{VI}}(\xi)$$

**Poznámka 7.8.** Integraci na intervalu $\langle a, b\rangle$ lze provést transformací

$$\bar{w}_i = \frac{b-a}{2}w_i, \quad \bar{x}_i = \frac{b+a}{2} + \frac{b-a}{2}x_i, \quad i = 0, \ldots, n$$

kde $w_i$ a $x_i$ jsou váhy, resp. kořeny, z intervalu $\langle -1, 1\rangle$.

**Poznámka 7.9.** Rozdělením intervalu integrace na $n$ podintervalů můžeme zpřesnit výpočet použitím složených formulí obdobně jak v případě složených Newtonových-Cotesových vzorců.

## 7.4 Výpočet integrálů pomocí extrapolace

Zatímco interpolací zjišťujeme hodnoty uvnitř intervalu interpolace, extrapolací nazýváme výpočet hodnot vně tohoto intervalu. Dá se totiž očekávat, že interpolační polynom bude dobře aproximovat i v blízkém okolí krajních bodů interpolace. Tohoto lze využít např. ke zpřesňování výpočtu integrálů.

Označme si $I(f)$ počítaný integrál a $I(f, h)$ složenou kvadraturní formuli s délkou podintervalů $h$. Provedeme-li výpočet integrálu pro dvě různé hodnoty $h$, např. $h_1, h_2$, dostáváme dvě aproximace $I(f, h_1)$ a $I(f, h_2)$. Novou aproximaci pak získáme vzorcem

$$I^{\mathrm{e}}(f, h) = \frac{1}{h_1^k - h_2^k}\left(h_1^k I(f, h_2) - h_2^k I(f, h_1)\right) \ ,$$

kde $k$ je řád kvadraturní formule.

Speciální volbou $h_1 = 2h_2 =: h$ dostáváme vztah

$$I^{\mathrm{e}}(f,h) = \frac{1}{2^k - 1}\left(2^k I(f,h) - I(f,2h)\right)$$

a odhad chyby

$$I(f) - I^{\mathrm{e}}(f,h) = \sum_{j=k+1}^{\infty} \frac{2^k - 2^j}{2^k - 1} a_j h^j$$

Z odhadu chyby je tedy patrné, že řád chyby nové aproximace je vyšší, konkrétně $k+1$.

## 7.5 Výpočet nevlastních integrálů

$$\int_{-\infty}^{\infty} f(x)dx \approx \int_{-R_1}^{R_2} f(x)dx$$

Hodnoty $R_1$ a $R_2$ přitom volíme tak, aby

$$\forall x \in \mathbb{R} \setminus \langle -R_1, R_2 \rangle : f(x) < \epsilon$$

kde $\epsilon$ je dostatečně malá konstanta.

# Kapitola 8

# Řešení počátečních úloh pro obyčejné diferenciální rovnice

## Průvodce studiem

*Celá řada fyzikálních jevů, které jsou závislé na čase nebo na jedné prostorové proměnné se popisují pomocí obyčejných diferenciálních rovnic. Analytické řešení těchto diferenciálních rovnic je však mnohdy velmi složité. Z tohoto důvodu se pro řešení těchto úloh používají metody numerické.*
*V následující kapitole seznámíme se základními metodami řešení obyčejných diferenciálních rovnic.*

**Definice 8.1.** *Obyčejnou diferenciální rovnicí prvního řádu* budeme rozumět rovnici, kterou lze napsat ve tvaru

$$y'(x) = f(x, y(x)), \tag{8.1}$$

kde

- $f : \mathbb{R}^2 \to \mathbb{R}$ je zadaná funkce definující diferenciální rovnici,
- $y : \mathbb{R} \to \mathbb{R}$ je neznámá funkce proměnné $x$,
- $x \in \mathbb{R}$ je reálná proměnná.

**Poznámka 8.2.** Samozřejmě, existují mnohem obecnější obyčejné diferenciální rovnice vyšších řádů nebo celé soustavy diferenciálních rovnic. Na jejich řešení se však obvykle dají velmi jednoduše aplikovat numerické metody, které řeší tento nejjednodušší typ diferenciální rovnice z Definice 8.1.

**Příklad 8.3.** Jednoduchým příkladem úlohy, která vede na obyčejnou diferenciální rovnici prvního řádu, je jednoduchý model množení králíků.

Označme-li $y$ jako funkci popisující počet králíků v určitém čase $t \geqq 0$, pak růst počtu králíků lze popsat rovnicí

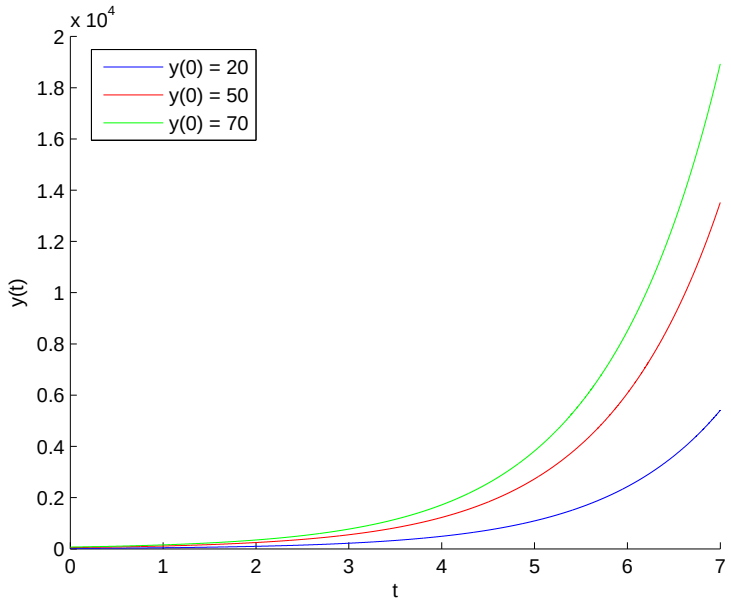
$$y'(t) = ay(t) \ ,$$

kde $a \in \mathbb{R}$ je konstanta určující závislost mezi počtem králíků a rychlostí jejich množení (kopulační parametr).
Našim úkolem je nalézt funkci určující počet králíků v daném čase, tj. nalézt předpis funkce $y(t)$ (pro $t \geqq 0$).

Analytické řešení[1] předchozí obyčejné diferenciální rovnice je

$$y(t) = c.\mathrm{e}^{at} \ ,$$

kde $c \in \mathbb{R}$ je libovolná konstanta.
Očividně předchozí úloha je nejednoznačná. Čím více králíků budeme mít na počátku (tj. v čase $t = 0$), tím strmější bude graf funkce $y$. Tato nejednoznačnost se v analytickém řešení projeví volbou parametru $c$.
Nechť $a = 0.3$, pak pro různé počáteční stavy $y(0)$ získáme různá řešení, viz Obrázek 8.1.

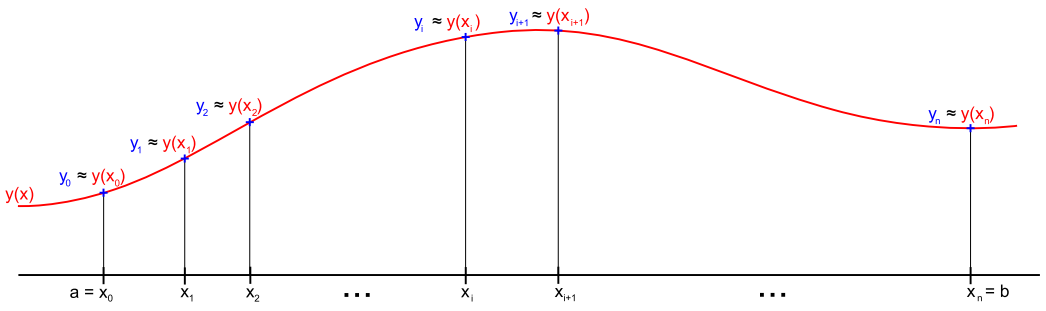



Obr. 8.1: Funkce popisující počet králíků v daném čase pro různé počáteční stavy.

Upřesňující informace o hledaném řešení v počátečním bodě uvažovaného intervalu se nazývá *počáteční podmínka*.

[1] Analytickým řešením obyčejných diferenciálních rovnic se zabývá skriptum [3].

Princip numerických metod řešení počátečních úloh pro obyčejné diferenciální rovnice prvního řádu spočívá v tom, že se nebude hledat analytický předpis řešení, ale pouze přibližné hodnoty řešení v bodech diskretizace uvažovaného intervalu, viz Obrázek 8.2.




Obr. 8.2: Algoritmus hledá řešení v pouze bodech diskretizace.

Cílem numerické metody pro řešení počátečních úloh pro diferenciální rovnice je tedy nalézt aproximace $y_0 \ldots, y_n$ řešení $y(x_0), \ldots, y(x_n)$

$$y_i \approx y(x_i), \quad i = 1, \ldots, n$$

na síti bodů $\{x_0, x_1, \ldots, x_n\}$,

$$\begin{aligned} x_{i+1} - x_i &= h_i \\ x_i &= x_0 + \sum_{j=0}^{i-1} h_j \ , \end{aligned}$$

kde $h_i$ se nazývá *krok numerické metody.*

Numerické metody pro řešení obyčejných diferenciálních úloh dělíme na dvě základní třídy:

1. *jednokrokové*
   K výpočtu nové aproximace $y_{i+1}$ v bodě $x_{i+1}$ využívají pouze předchozí hodnotu aproximace $y_i$ a mají obecný předpis
   $$y_{i+1} = \Phi(y_i, x_i, f, h_i) \ .$$
2. *vícekrokové*
   K výpočtu nové aproximace $y_{i+1}$ v bodě $x_{i+1}$ využívají k předchozích aproximací $y_i, \ldots y_{i-k+1}$ a mají obecný předpis
   $$y_{i+1} = \Psi(y_i, y_{i-1}, \ldots, y_{i-k+1}, x_i, \ldots, x_{i-k+1}, f) \ .$$

Je-li $h_i := h = \text{const.}$ nazýváme numerickou metodu *s pevným krokem*, v opačném případě *s proměnlivým krokem*.

## 8.1 Jednokrokové metody

Nechť je dána obyčejná diferenciální rovnice prvního řádu s počáteční podmínkou

$$\begin{aligned} y'(x) &= f(x, y(x)) \\ y(x_0) &= y_0 \ . \end{aligned} \tag{8.2}$$

Naším úkolem je nalézt diskrétní řešení na intervalu $\langle a, b\rangle$, resp. na síti bodů

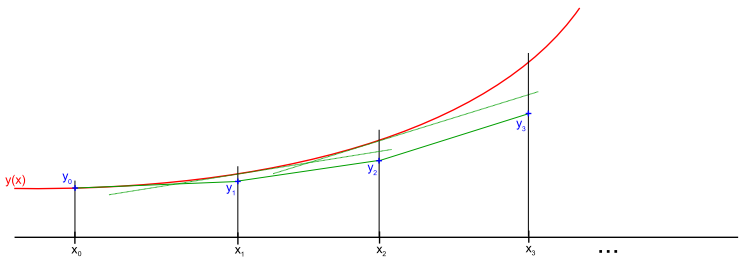
$$a \ = x_0 < x_1 < \cdots < x_n = b.$$

### 8.1.1 Eulerova metoda

Nejjednodušší metodou pro numerické řešení úlohy (8.2) je tzv. *Eulerova metoda*. Přepokládejme, že v bodě $x_0$ je předepsána počáteční podmínka $y_0$. Postupně budeme procházet intervaly diskretizace

$$\langle x_i, x_{i+1}\rangle, \quad i = 0, 1, \ldots n-1,$$

přičemž vždy vypočte aproximaci řešení $y_{i+1}$ na základě již spočteného $y_i$.




Obr. 8.3: Princip Eulerovy metody.

Konkrétně hodnoty funkce $y$ na intervalu $\langle x_i, x_{i+1}\rangle$ aproximujeme přímkou vycházející z bodu $[x_i, y_i]$

$$y(x) \ \approx \ y(x_i) + (x - x_i)y'(x_i), \quad \text{pro } x \in \langle x_i, x_{i+1}\rangle \ .$$

Jelikož pro směrnici platí $y'(x_i) = f(x_i, y(x_i))$ (dle rovnice (8.2)), pak dosazením získáme

$$y(x) \ \approx \ y(x_i) + (x - x_i)f(x_i, y(x_i)).$$

Nahradíme-li přesné hodnoty $y(x_i)$ jejich aproximacemi $y_i$, pak dostáváme předpis pro nalezení nové aproximace $y_{i+1}$ v následujícm tvaru

$$\begin{aligned} y_{i+1} &= y_i + (x_{i+1} - x_i) f(x_i, y_i) \\ y_{i+1} &= y_i + h_i f(x_i, y_i), \quad \text{pro } i = 0, 1, \ldots, n-1 \ . \end{aligned}$$

Tím jsme získali jednoduchý iterační předpis pro výpočet aproximace řešení.

**Poznámka 8.4.** Tento algoritmus lze odvodit také z vzorce pro numerickou derivace. Jelikož

$$y'(x) \approx \frac{y(x+h) - y(x)}{h} \quad \text{(pro dostatečně malé } h \text{ )},$$

pak dle rovnice (8.2)

$$y'(x) = f(x, y(x)) \quad \Rightarrow \quad \frac{y(x+h) - y(x)}{h} = f(x, y(x))$$

a úpravou získáme předešlý iterační předpis.

**Poznámka 8.5.** Tento iterační předpis lze odvodit i z Taylorova polynomu

$$\begin{aligned} y(x_{i+1}) &= y(x_i + h_i) = y(x_i) + y'(x_i) h_i + \tfrac{y''(\xi)}{2!} h_i^2, \quad \xi \in (x_i, x_{i+1}) \\ y(x_{i+1}) &\approx y(x_i) + y'(x_i) h_i \\ y_{i+1} &= y_i + h_i f(x_i, y_i) \end{aligned}$$

**Algoritmus 26** EULEROVA METODA

**Input:** $\{x_i\}_{i=0}^n$, $f(x, y)$, $y_0$

1: **for** $i = 1, 2, \ldots n-1$ **do**
2: $\quad h_i := x_{i+1} - x_i$
3: $\quad y_{i+1} := y_i + h_i f(x_i, y_i)$
4: **end for**

**Output:** $\{y_i\}_{i=0}^n$

Při studiu aproximační chyby, které se při použití metody dopouštíme, rozlišujeme dva základní typy:

- *Lokální diskretizační chyba*
  Jedná se o chybu, které se dopouštíme v jednom kroku metody (tj. na jednom intervalu $\langle x_i, x_{i+1} \rangle$) s vyloučením zaokrouhlovacích chyb. Jelikož hodnotu $y(x_{i+1})$ aproximujeme hodnotou Taylorova polynomu prvního řádu, tj. $y(x_{i+1}) \approx T_1(x_i)$, dopouštíme se chyby řádově $h_i^2$. Jinak řečeno

  $$y(x_{i+1}) = y(x_i) + h_i f(x_i, y(x_i)) + d_i = T_1(x_i) + d_i, \quad d_i = \mathcal{O}(h_i^2),$$

  kde $d_i$ je lokální diskretizační chyba, které se dopouštíme na každém intervalu $\langle x_i, x_{i+1} \rangle$.

- *Globální diskretizační chyba*
  Pod touto chybou rozumíme rozdíl přesného řešení a aproximace numerickou metodou v bodě $x_i$, tj.
  $$\mathrm{e}_i = y(x_i) - y_i \ .$$
  Očividně tato chyba akumuluje všechny předchozí zaokrouhlovací i lokální diskretizační chyby $d_0, d_1, \ldots, d_{n-1}$.

**Definice 8.6.** Jestliže
$$b := x_n = x_0 + nh, \quad h := \frac{b-a}{n}, \quad n \in \mathbb{N},$$
je-li
$$\lim_{n\to 0} y_n = y(b),$$
pak říkáme, že je metoda *konvergentní*.

**Definice 8.7.** Je-li $d_i = \mathcal{O}(h_i^{p+1})$, pak říkáme, že je numerická *metoda řádu p*, tj.
$$\exists C = \text{const.} \ : \quad |d_i| \leqq C h_i^{p+1} \text{ pro } h_i \to 0 \ .$$

**Poznámka 8.8.** Obecně lze dokázat následující tvrzení

- metody alespoň 1. řádu jsou konvergentní
- pro globální diskretizační chybu v bodě $x$ s pevným krokem $h$ platí
  $$\mathrm{e}(x,h) = \mathcal{O}(h^p)$$
  kde $p$ je řád metody
- Eulerova metoda je řádu 1

**Příklad 8.9.** Řešte počáteční úlohu pro obyčejnou diferenciální rovnici

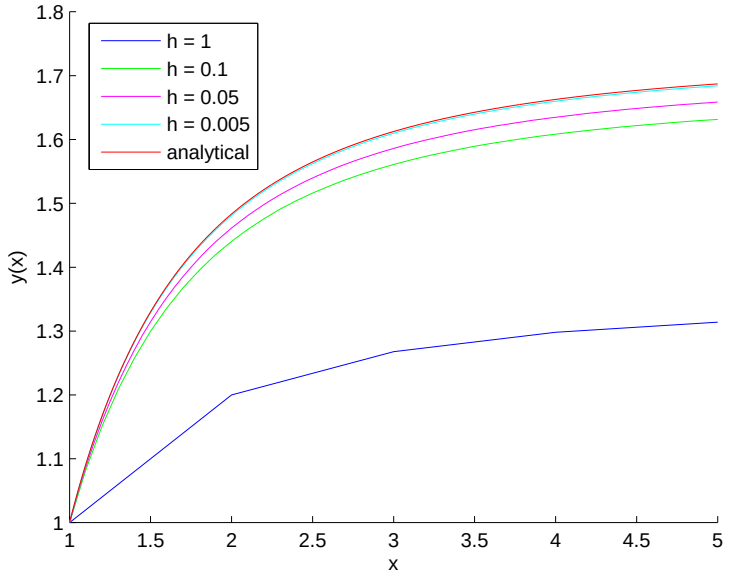
$$\begin{aligned} y' &= \tfrac{1+y^2}{xy(1+x^2)} \\ y(1) &= 1 \end{aligned}$$
na intervalu $\langle 1, 5\rangle$ pomocí Eulerovy metody.

*Řešení.* Analytické řešení úlohy (viz [3],[13]) má předpis
$$y(x) = \sqrt{\frac{3x^2-1}{1+x^2}}.$$

Do algoritmu Eulerovy metody dosaďme pravou stranu diferenciální rovnice

$$f(x,y) = \frac{1+y^2}{xy(1+x^2)},$$

pak získáme pro různou volbu diskretizačního parametru $h$ různě přesná řešení, viz Obrázek 8.4.
Z obrázku lze vidět, že pro $h \to 0$ numerické řešení konverguje k řešení přesnému. ▲




Obr. 8.4: Řešení Eulerovou metodou při použití různých diskretizačních parametrů.

### 8.1.2 Metody Runge-Kutta

Pokusme se nyní vylepšit Eulerovu metodu. Pro výpočet $y_i$ využijme odhady derivací ve více bodech. Pro celý krok potom použijeme vážený průměr těchto derivací s váhami $\alpha_i$, konkrétně

$$y_{i+1} = y_i + h_i(\alpha_1 k_1 + \cdots + \alpha_r k_r) = y_i + h_i \sum_{j=1}^{r} \alpha_j k_j \ ,$$

přičemž pro odhady derivací platí

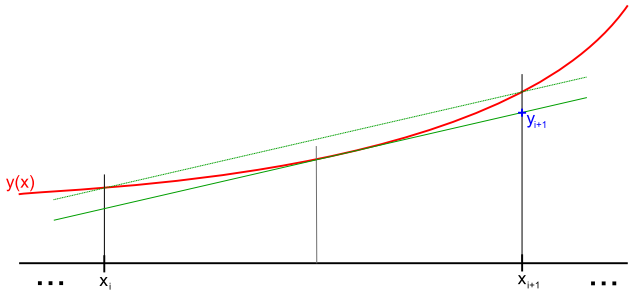
$$\begin{aligned} k_1 &= f(x_i, y_i) \\ &\vdots \\ k_j &= f(x_i + \lambda_j h_i, y_i + \mu_j h_i k_{j-1}) \quad \text{pro } j > 1 \end{aligned}$$

Pro speciální volby parametrů $\alpha, \lambda$ a $\mu$ dostáváme následující metody.

### Metody Runge-Kutta 2. řádu

1. Pro volbu $r = 2,\ \alpha_1 = 0,\ \alpha_2 = 1,\ \lambda_2 = \mu_2 = \frac{1}{2}$ získáme předpis:

$$\begin{aligned} y_{i+1} &= y_i + h_i k_2 \\ k_1 &= f(x_i, y_i) \\ k_2 &= f(x_i + \tfrac{1}{2}h_i, y_i + \tfrac{1}{2}h_i k_1) \end{aligned}$$




Obr. 8.5: Metoda Runge-Kutta 2. řádu, 1. varianta.

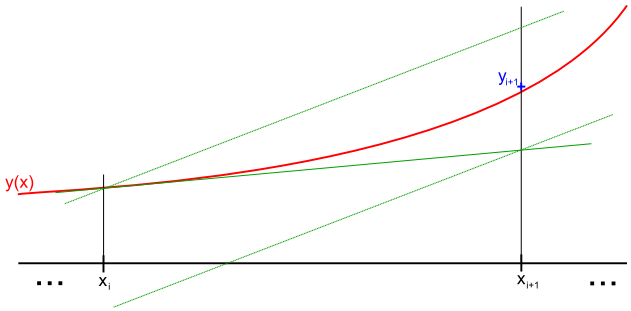
Tato volba má svou geometrickou interpretaci, viz Obrázek 8.5. Funkci $y$ na intervalu $\langle x_i, x_{i+1}\rangle$ aproximujeme přímkou vycházející z bodu $[x_i, y_i]$ se směrnicí $y'_{i+\frac{1}{2}}$, tj. se směrnicí rovnou derivaci v bodě uprostřed intervalu.
Porovnáním s Taylorovým polynomem lze dokázat, že odvozená metoda je řádu 2.

2. Volbou $r = 2,\ \alpha_1 = \alpha_2 = \frac{1}{2},\ \lambda_2 = \mu_2 = 1$ získáme předpis:

$$\begin{aligned} y_{i+1} &= y_i + h_i \tfrac{k_1+k_2}{2} \\ k_1 &= f(x_i, y_i) \\ k_2 &= f(x_{i+1}, y_i + h_i k_1) \end{aligned}$$

Tato metoda se také občas nazývá *Crank-Nicolsonova metoda.*

Funkci $y$ na intervalu $\langle x_i, x_{i+1}\rangle$ aproximujeme přímkou vycházející z bodu $[x_i, y_i]$ se směrnicí rovnu aritmetickému průměru derivací v krajních bodech intervalu (viz Obrázek 8.6).
Obdobně jako u předchozí metody lze ukázat, že Crank-Nicholsonova metoda je řádu 2.




Obr. 8.6: Metoda Runge-Kutta 2. řádu, 2. varianta.

### Metoda Runge-Kutta 4. řádu

Snad nejhojněji se používá metoda 4. řádu, která je dána předpisem.

$$\begin{aligned} y_{i+1} &= y_i + h_i \tfrac{k_1+2k_2+2k_3+k_4}{6} \\ k_1 &= f(x_i, y_i) \\ k_2 &= f(x_i + \tfrac{1}{2}h_i, y_i + \tfrac{1}{2}h_i k_1) \\ k_3 &= f(x_i + \tfrac{1}{2}h_i, y_i + \tfrac{1}{2}h_i k_2) \\ k_4 &= f(x_{i+1}, y_i + h_i k_3) \end{aligned}$$

Bylo by možné odvodit i metody vyšších řádů, ale zde bychom již odkázali na literaturu [24] a [25], kde je možné tyto metody nalézt.

**Příklad 8.10.** Řešte počáteční úlohu

$$\begin{aligned} 12x^2 + 3y + 54 &= (60 + y')x \\ y(1) &= 1 \end{aligned}$$

na intervalu $\langle 1, 10\rangle$ metodou Runge-Kutta druhého a čtvrtého řádu.

*Řešení.* Úloha má analytické řešení

$$y(x) = x^3 - 12x^2 + 30x - 18 \ .$$

Tento předpis vyhovuje jak diferenciální rovnici, tak počáteční podmínce, navíc dle Picardovy-Lindelöfovy věty (viz [13]) existuje právě jedno řešení.

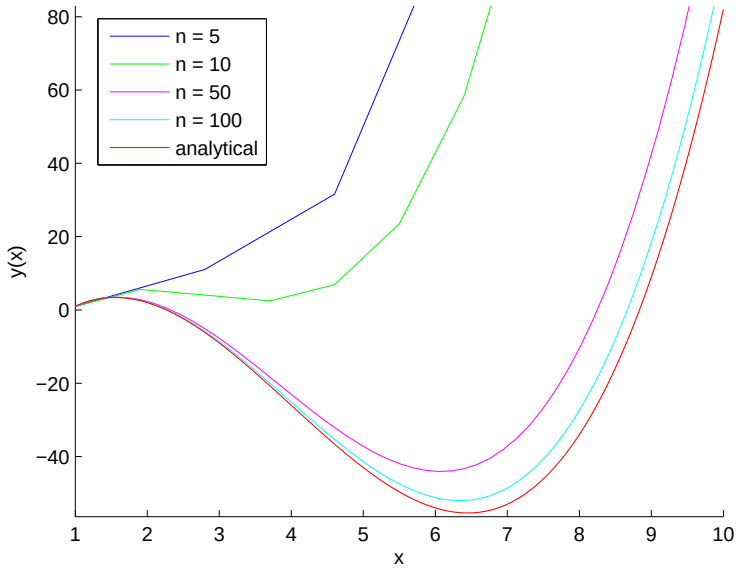
Nejdříve budeme muset upravit danou diferenciální rovnici do tvaru odpovídajícímu rovnici ((8.2)).

$$\begin{aligned} 12x^2 + 3y + 54 &= 60x + y'x \\ 3y + 12x^2 - 60x + 54 &= y'x \\ y' &= \tfrac{3y+12x^2-60x+54}{x} =: f(x, y) \ . \end{aligned}$$

Pro numerické jsme volili ekvidistantní dělení s diskretizačním parametrem

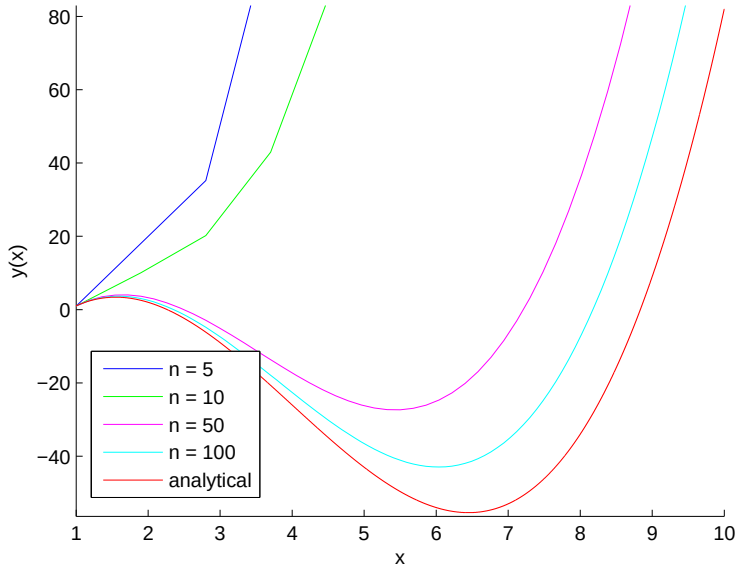
$$h = \frac{b-a}{n} = \frac{9}{n}$$

Na obrázcích 8.7 a 8.8 naleznete srovnání přesnosti řešení pro různé volby počtu dělení intervalu na n podintervalů.. ▲




Obr. 8.7: Řešení metodou Runge-Kutta 2.řádu.

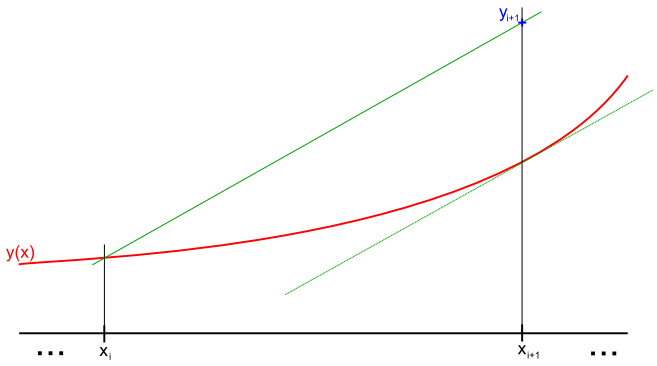
**Poznámka 8.11.** U výše uvedených Rungových-Kuttových metod se řád metody vždy rovná použitému počtu odhadů derivace. Tak to ale vždy není, například při pěti odhadech derivace se dá sestrojit pouze metoda čtvrtého řádu. I z toho důvodu patří metoda čtvrtého řádu k velice oblíbeným.




Obr. 8.8: Řešení metodou Runge-Kutta 4.řádu.

### Implicitní metody

Hledanou funkci $y$ aproximujeme na intervalu $\langle x_i, x_{i+1}\rangle$ úsečkou vychazející z bodu $[x_i, y_i]$ se směrnicí $y'_{i+1}$, viz Obrázek 8.9.




Obr. 8.9: Princip implicitní metody.

Tedy tato metoda bude mít předpis

$$y_{i+1} = y_i + h_i f(x_{i+1}, y_{i+1}) \ .$$

Tzn., že jak na levé tak na pravé straně předpisu se vyskytuje neznámá $y_{i+1}$. V každém kroku musíme tedy řešit obecně nelineární rovnici $y_{i+1} = G(y_{i+1})$.
K tomu lze použít některou numerickou metodu (např. prosté iterace).

**Příklad 8.12.** Řešte okrajovou úlohu

$$\begin{aligned} 12x^2 + 3y + 54 &= (60 + y')x \\ y(1) &= 1 \end{aligned}$$

na intervalu $\langle 1, 10\rangle$ implicitní Eulerovou metodou.

*Řešení.* Stejně jako v předchozím příkladě nejdříve nalezneme předpis pravé strany diferenciální rovnice

$$f(x, y) = \frac{3y + 12x^2 - 60x + 54}{x}$$

a pak dosazením do předpisu implicitní metody

$$y_{i+1} = y_i + h.f(x_{i+1}, y_{i+1})$$

získáme rovnici

$$y_{i+1} = y_i + h\frac{3y_{i+1} + 12x_{i+1}^2 - 60x_{i+1} + 54}{x_{i+1}} \ ,$$

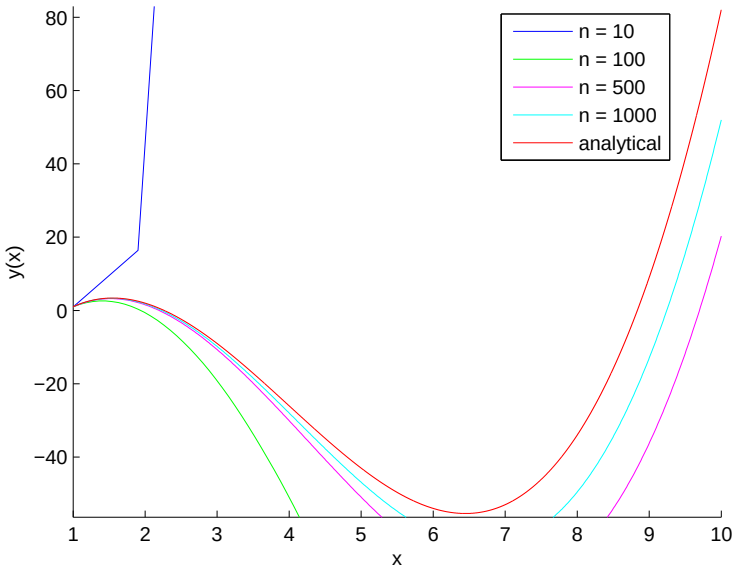
kterou budeme muset vyřešit v každé iteraci. Ještě předtím, než začneme uvažovat, kterou numerickou metodu použijeme pro řešení této dílčí úlohy, pokusíme se rovnici vyřešit analyticky. Získáme předpis

$$y_{i+1} = \frac{1}{x_{i+1} - 3h}\left[x_{i+1}y_i + h.(12x_{i+1}^2 - 60x_{i+1} + 54)\right]$$

Jelikož původní rovnice byla lineární, neznámou $y_{i+1}$ lze vyjádřit pomocí proměnných $y_i$ a $x_{i+1}$.
Řešení úlohy je vykresleno na Obrázku 8.10.

Obecně však tato rovnice může být nelineární, pak bychom museli volit například některou z metod uvedených v kapitole 2.

▲




Obr. 8.10: Řešení implicitní metodou.

**Poznámka 8.13.** Všechny výše uvedené metody lze použít i pro soustavy diferenciálních rovnic, jak si ukážeme v kapitole 8.3.

## 8.2 Vícekrokové metody

Na rozdíl od jednokrokových metod budeme v případě vícekrokových metod používat k výpočtu nové aproximace $y_{i+1}$ $k$ předchozích hodnot $y_i, y_{i-1}, \ldots, y_{i-k+1}$, $k \geqq 1$. V takovém případě budeme mluvit o tzv. $k$-krokové metodě.
Pro jednoduchost budeme uvažovat ekvidistantní síť bodů

$$a = \quad x_0 < x_1 < x_2 < \cdots < x_n \quad = b \tag{8.3}$$

s diskretizačním parametrem $h$, tj. $\forall i = 0, \ldots, n : x_i = x_0 + ih$.
Dále označme

$$f_i \overset{\text{ozn}}{=} f(x_i, y_i) \ .$$

**Definice 8.14.** *Lineární k-krokovou metodou* pro řešení počátečních úloh pro obyčejné diferenciální rovnice rozumíme metodu danou předpisem

$$\sum_{j=0}^{k} \alpha_j y_{i+j} = h \sum_{j=0}^{k} \beta_j f_{i+j}, \quad i = 0, 1, \ldots n-k \tag{8.4}$$

kde $\alpha_0, \ldots, \alpha_k, \beta_0, \ldots, \beta_k \in \mathbb{R}$ jsou dané konstanty splňující

$$|\alpha_0| + |\beta_0| > 0, \quad \alpha_k = 1 \ .$$

**Poznámka 8.15.** Pokud zvolíme $k = 1$ (jednokrokovou metodu), lze rozepsat rovnici (8.4) na tvar

$$\alpha_0 y_i + \alpha_1 y_{i+1} = h(\beta_0 f_i + \beta_1 f_{i+1}) \ .$$

Pak volbou parametrů

$$\begin{array}{lllllll} \alpha_0 & = & -1 & & \beta_0 & = & 1 \\ \alpha_1 & = & 1 & & \beta_1 & = & 0 \end{array}$$

získáme

$$-y_i + y_{i+1} = h f_i \ .$$

Tedy Eulerovu metodu.

**Poznámka 8.16.** K tomu abychom mohli předpis použít, musíme znát hodnoty $y_0, y_1, \ldots, y_{k-1}$ (při výpočtu $y_k$ použijeme těchto $k$ předchozích hodnot). Tyto hodnoty nelze získat použitím $k$-krokové metody, proto pro jejich výpočet použijeme některou z jednokrokových metod.

**Poznámka 8.17.**

- Je-li $\beta_k = 0$, dostáváme metodu *explicitní*, tj. člen $f(x_{i+k}, y_{i+k})$ se v rovnici nevyskytuje a neznámou $y_{i+k}$ lze vyjádřit přímo z předpisu

$$y_{i+k} = -\sum_{j=0}^{k-1} \alpha_j y_{i+j} + h \sum_{j=0}^{k-1} \beta_j f(x_{i+j}, y_{i+j}) \ .$$

- Je-li $\beta_k \neq 0$, dostáváme metodu *implicitní*, tj.

$$y_{i+k} + \sum_{j=0}^{k-1} \alpha_j y_{i+j} = h f(x_{i+k}, y_{i+k}) + h \sum_{j=0}^{k-1} \beta_j f(x_{i+j}, y_{i+j}) \ .$$

To znamená, že dostáváme v každém kroku úlohu

$$y_{i+k} = G(y_{i+k}) \ ,$$

kterou můžeme řešit některou numerickou metodou, např. metodou prostých iterací (viz 2.3)

$$y_{i+k}^{j+1} = G(y_{i+k}^{j}), \quad j = 0, 1, \ldots$$

Otázkou však zůstává, jak volit parametry $\alpha_0, \ldots, \alpha_k$ a $\beta_0, \ldots, \beta_k$.

### 8.2.1 Metody založené na numerické integraci

Jelikož

$$y'(x) = f(x, y(x)) \ ,$$

pak integrací přes interval $\langle x_i, x_{i+1} \rangle$ získáme

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x))dx \ ,$$

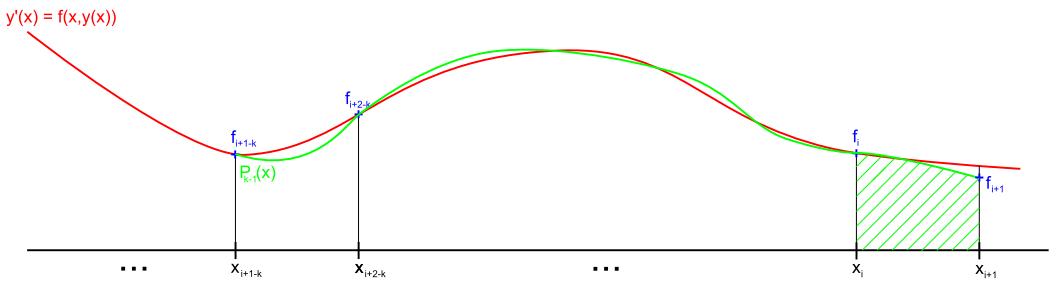
a úpravou

$$y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x, y(x))dx \ .$$

Vyjádřeno ve formě aproximací

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y(x))dx \ . \tag{8.5}$$

Integrál na pravé straně budeme počítat numericky na síti bodů $\{x_{i-k+1}, \ldots, x_{i+1}\}$.




Obr. 8.11: Princip využití numerické integrace při řešení diferenciální rovnice.

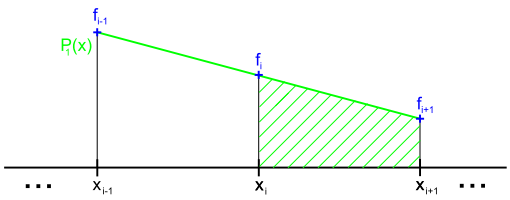
Pro tento výpočet můžeme použít Newtonovy-Cotesovy formule (viz 7.2).
K tomu však budeme potřebovat i bod $[x_{i+1}, f_{i+1}]$.
Existují dva způsoby jak tento bod získat:

- *extrapolací* (např. Adams-Bashforthovy metody) - pokud při integraci použijeme Lagrangeův interpolační polynom $P_{k-1}(x)$ stupně $k-1$ na síti $\{x_{i-k+1}, \ldots, x_i\}$, lze extrapolovat hodnotu
$$f_{i+1} \approx P_{k-1}(x_{i+1})$$
a následně ji použít pro výpočet integrálu.
- *implicitně* (např. Adams-Moultonovy metody) - potřebnou hodnotu nebudeme vyčíslovat, naopak ji ponecháme jako neznámou. Pak získáme implicitní rovnici, kterou lze řešit např. prostými iteracemi.

### Adams-Bashforthovy metody

Odvoďme vzorec pro 2-krokovou Adams-Bashforthovu metodu.
Volme ekvidistantní síť uzlů (8.3) s krokem $h$. Nechť máme k dispozici aproximace řešení $y_i, y_{i-1}$ (tj. máme k dispozici i $f_i, f_{i-1}$) a našim úkolem je nalézt předpis pro $y_{i+1}$ ve tvaru (8.5).
Pro numerický výpočet požadovaného integrálu budeme potřebovat přibližnou hodnotu $f_{i+1}$. Nejdříve sestrojíme interpolační polynom $P_1(x)$ prvního stupně na síti $x_{i-1}, x_i$, tj.

$$P_1(x) = a_0 + a_1 x \quad \text{přičemž} \quad P_1(x_{i-1}) = f_{i-1} \quad \wedge \quad P_1(x_i) = f_i \ .$$




Obr. 8.12: Princip Adams-Bashforthovy dvoukrokové metody

Z požadovaných interpolačních vlastností získáme předpis

$$P_1(x) = \frac{x_{i-1}f_i - x_i f_{i-1}}{x_{i-1} - x_i} + \frac{f_{i-1} - f_i}{x_{i-1} - x_i}x = \frac{1}{h}\left[x_i f_{i-1} - x_{i-1}f_i + (f_i - f_{i-1})x\right] \ .$$

Následně extrapolací spočteme hledanou hodnotu

$$f_{i+1} \approx P_1(x_{i+1}) = \frac{1}{h}\left[x_i f_{i-1} - x_{i-1}f_i + (f_i - f_{i-1})x_{i+1}\right] = 2f_i - f_{i-1} \ .$$

Pak využijeme Newton-Cotesova vzorce pro integraci na intervalu $\langle x_i, x_{i+1}\rangle$ (Lichoběžníkové pravidlo)

$$\int_{x_i}^{x_{i+1}} f(x, y(x))dx \approx \frac{x_{i+1} - x_i}{2}\left[f_i + (2f_i - f_{i-1})\right] = \frac{h}{2}(3f_i - f_{i-1})$$

a dosadíme do předpisu (8.5). Získáváme dvoukrokovou metodu ve tvaru

$$y_{i+1} = y_i + \frac{h}{2}(3f_i - f_{i-1}) \ .$$

**Poznámka 8.18.** Vícekrokové metody tohto typu lze odvodit podobným způsobem. Například pro $k = 3$ a $k = 4$ získáme tyto předpisy

$$\begin{array}{lll} k = 3 & y_{i+1} = & y_i + \frac{1}{12}h(23f_i - 16f_{i-1} + 5f_{i-2}) \\ k = 4 & y_{i+1} = & y_i + \frac{1}{24}h(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}) \end{array}$$

### Adams-Moultonovy metody

Odvoďme vzorec pro 2-krokovou Adams-Moultonovu metodu.
Princip je stejný jako u Adams-Bashfortových formulí - interpolační polynom sestrojíme však na síti bodů $x_i, x_{i+1}$.
V tomto případě se jedná o Lagrangeův interpolační polynom prvního stupně s předpisem

$$P_1(x) = a_0 + a_1 x \quad \text{přičemž} \quad P_1(x_i) = f_i \ \wedge \ P_1(x_{i+1}) = f_{i+1} \ .$$

Z požadovaných interpolačních vlastností získáme předpis

$$P_1(x) = \frac{x_i f_{i+1} - x_{i+1} f_i}{x_i - x_{i+1}} + \frac{f_i - f_{i+1}}{x_i - x_{i+1}} x = \frac{1}{h}\left[x_{i+1} f_i - x_i f_{i+1} + (f_{i+1} - f_i)x\right] \ .$$

Nyní spočtěme integrál potřebný pro předpis iterační metody (8.5). K jeho výpočtu použijeme Lichoběžníkové pravidlo

$$\int_{x_i}^{x_{i+1}} f(x, y(x)) \approx \int_{x_i}^{x_{i+1}} P_1(x)dx = \frac{h}{2}(f_i + f_{i+1}) \ .$$

Pak dosazením do předpisu (8.5)

$$y_{i+1} = y_i + \frac{h}{2}\left(f(x_{i+1}, y_{i+1}) + f_i\right) \ ,$$

kde $y_{i+1}$ je neznámá na levé i pravé straně. V každé iteraci proto budeme řešit obecně nelineární soustavu

$$y_{i+1} = G(y_{i+1}) \ .$$

**Poznámka 8.19.** Vícekrokové metody tohto typu lze odvodit podobným způsobem. Například pro $k = 3$ a $k = 4$ získáme tyto předpisy

$$\begin{array}{lll} k = 3 & y_{i+1} = & y_i + \frac{1}{12}h(5f(x_{i+1}, y_{i+1}) + 8f_i - f_{i-1}) \ , \\ k = 4 & y_{i+1} = & y_i + \frac{1}{24}h(9f(x_{i+1}, y_{i+1}) + 19f_i - 5f_{i-1} + f_{i-2}) \ . \end{array}$$

## 8.3 Soustavy diferenciálních rovnic a diferenciální rovnice vyšších řádů

### 8.3.1 Soustavy obyčejných diferenciálních rovnic

**Definice 8.20.** *Soustavou $m$ obyčejných diferenciálních rovnic* nazveme soubor $m$ rovnic o $m$ neznámých funkcí $y_1, \ldots, y_m : \mathbb{R} \to \mathbb{R}$

$$\begin{aligned} y_1'(x) &= f_1(x, y_1(x), y_2(x), \ldots, y_m(x)) \\ y_2'(x) &= f_2(x, y_1(x), y_2(x), \ldots, y_m(x)) \\ &\vdots \\ y_m'(x) &= f_m(x, y_1(x), y_2(x), \ldots, y_m(x)) \end{aligned}$$

**Poznámka 8.21.** Soustavu lze zapsat vektorově

$$Y'(x) = F(x, Y(x))$$

kde

$$Y(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_m(x) \end{bmatrix}, \quad F(x, Y(x)) = \begin{bmatrix} f_1(x, Y(x)) \\ f_2(x, Y(x)) \\ \vdots \\ f_m(x, Y(x)) \end{bmatrix}$$

Řešením rozumíme diferencovatelné funkce $Y(x)$ na intervalu $\langle a, b\rangle$, které splňují na $\langle a, b\rangle$ výše uvedené rovnice.
Na řešení této vektorové obyčejné diferenciální rovnice můžeme použít některou z již uvedených numerických metod.

### 8.3.2 Diferenciální rovnice vyšších řádů

Řešení libovolné diferenciální rovnice $m$-tého řádu

$$y^{(m)}(x) = f(x, y(x), y'(x), \ldots, y^{(m-1)}(x))$$

je ekvivalentní s řešením soustavy

$$\begin{aligned} y'(x) &= u_1(x) \\ u_1'(x) &= u_2(x) \\ &\vdots \\ u_m'(x) &= f(x, y(x), u_1(x), \ldots, u_{m-1}(x)) \end{aligned}$$

Tuto soustavu pak můžeme stejně jako v předešlé kapitole přepsat do vektorového tvaru

$$Y'(x) = F(x, Y(x))$$

a na její řešení můžeme opět využít numerických metod popsaných v kapitolách 8.1 a 8.2.

**Příklad 8.22.** Uvažujme matematické kyvadlo - hmotný bod zavěšený na tenkém vláknu zanedbatelné hmotnosti. Zanedbáme také odpor vzduchu při pohybu i tření v závěsu. Funkcí $\varphi(t)$ označme výchylku kyvadla z rovnovážné polohy v čase $t$.
Dá se ukázat (viz [12]), že tato úloha vede na obyčejnou diferenciální rovnici druhého řádu s okrajovými podmínkami

$$\begin{array}{rcl} \varphi''(t) & = & -\frac{g}{l}\sin\varphi(t) \\ \varphi(0) & = & \varphi_0 \\ \varphi'(0) & = & v_0 \end{array}$$

kde

- $g$ je tíhové zrychlení
- $l$ je délka vlákna
- $\varphi_0$ je počáteční výchylka v čase $t = 0$
- $v_0$ je počáteční rychlost v čase $t = 0$

Zvolme konkrétní hodnoty $g = 9,8m.s^{-2}, l = 1m, \varphi_0 = 0, v_0 = 1m.s^{-1}$.

*Řešení.* Pokud při řešení využijeme substituci

$$u(t) = \varphi'(t)\ ,$$

tak získáme soustavu s okrajovými podmínkami

$$\begin{array}{rcl} u'(t) & = & -10\sin\varphi(t) \\ \varphi'(t) & = & u(t) \end{array} \qquad \begin{array}{rcl} u(0) & = & 1 \\ \varphi(0) & = & 0 \end{array}$$

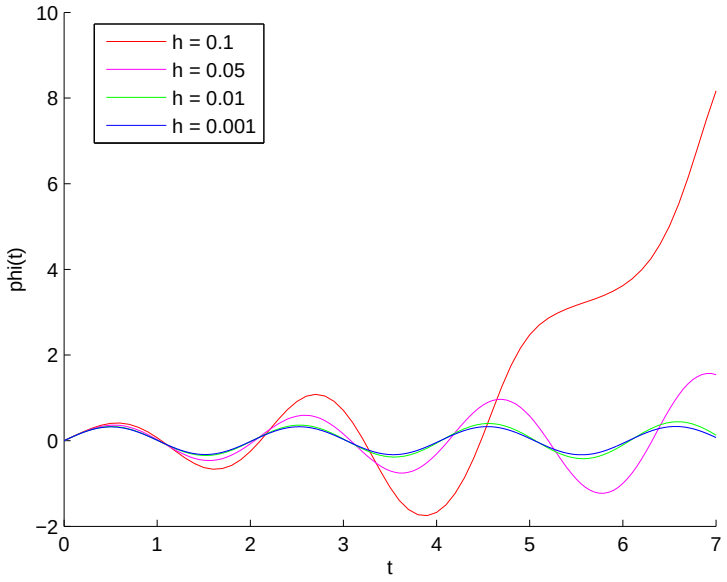
Soustavu lze zapsat vektorově

$$Y'(t) = F(t, Y(t)), \quad Y(t) = \begin{bmatrix} u\ (t) \\ \varphi(t) \end{bmatrix}, \quad F(t, Y(t)) = \begin{bmatrix} \sin\varphi(t) \\ u(t) \end{bmatrix}$$

a použít Eulerovu metodu s počáteční hodnotou a iteračním krokem (pro jednoduchost volme konstantní diskretizační krok $h$)

$$\begin{bmatrix} u_0 \\ \varphi_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} u_{i+1} \\ \varphi_{i+1} \end{bmatrix} = \begin{bmatrix} u_i \\ \varphi_i \end{bmatrix} + h \begin{bmatrix} -\sin\varphi_i \\ u_i \end{bmatrix}$$

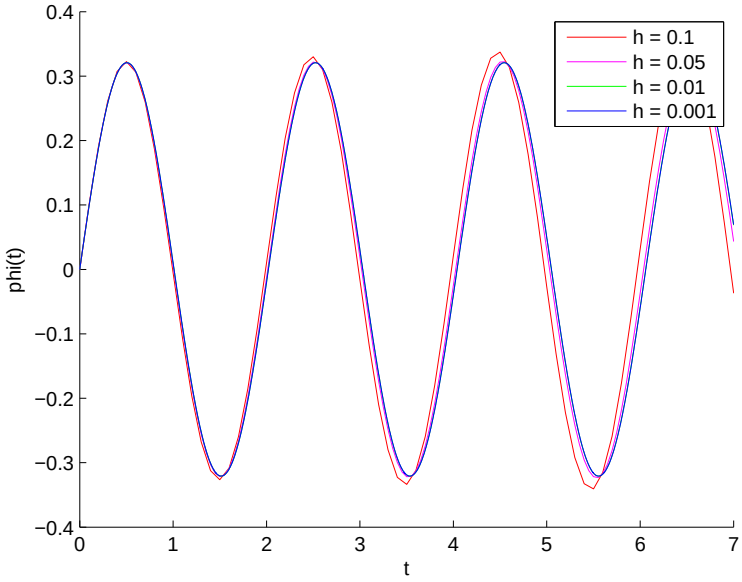
Při různé volbě diskretizačního kroku získáme různá řešení, která jsou zobrtazena na Obrázku 8.13. Nicméně dle konvergence Eulerovy metody lze s jistotou tvrdit, že čím menší tento parametr zvolíme, tím přesnější řešení získáme.




Obr. 8.13: Řešení úlohy matematického kyvadla Eulerovou metodou

Je očividné, že použití Eulerovy metody je numericky nestabilní. Předchozí výsledky naznačují, že kyvadlo postupně zvětšuje své výchylky při každém dalším kmitu, což je v rozporu s fyzikálními vlastnostmi matematického kyvadla.
Při použití metody Runge-Kutta 2.řádu, získáme pro různou volbu diskretizačního parametru $h$ získáme řešení, která jsou na obrázku 8.14. Z obrázku je patrné, že metoda Runge-Kutta je mnohem stabilnější.

Vedlejším produktem obou metod je průběh derivace funkce $\varphi(t)$, tedy průběh funkce rychlosti pohybu kyvadla. ▲




Obr. 8.14: Řešení úlohy matematického kyvadla metodou Runge-Kutta 2.řádu

# Literatura


[1] Kozubek T. – Brzobohatý T. – Jarošová M. – Hapla V. – Markopoulos A. *Lineární algebra s Matlabem*. Matematika pro inženýry 21. století, 2012.

[2] Dostál, Z. – Vondrák, V. – Lukáš, D. *Lineární algebra*. Matematika pro inženýry 21. století, 2012.

[3] Krajc, B. – Beremlijski, P. *Obyčejné diferenciální rovnice*. Matematika pro inženýry 21. století, 2012.

[4] Bouchala, J. *Úvod do funkcionální analýzy*. Matematika pro inženýry 21. století, 2012.

[5] Dostál, Z. *Optimal Quadratic Programming Algorithms*. Springer, 2009.

[6] Dostál, Z. *Lineární Algebra*. skriptum VŠB-TU Ostrava , 2000.

[7] Děmidovič, B.P. – Maron, J.A. *Základy numerické matematiky*. SNTL, 1966.

[8] Horová, I. – Zelinka, J. *Numerické metody*. Přírodovědecká fakulta Masarykovy univerzity v Brně, 2008.

[9] Nocedal, J. – Wright, S. J. *Numerical Optimization*. Springer, 1999.

[10] Greguš, M. – Švec, M. – Šeda, V. *Obyčajné diferenciálne rovnice*. SNTL, 1985.

[11] Blata, Z. *Numerické metody řešení diferenciálních rovnic, diplomová práce*. Univerzita Tomáše Bati v Zlíně , 2009.

[12] Vlček, J. *Matematické modelování, sylabus k předmětu*. VŠB-TU Ostrava , 2009.

[13] Bouchala, J. *Matematika III pro bakalářské studium*. VŠB-TU Ostrava , 2000.

[14] Bouchala, J. *Matematická analýza 1*. VŠB-TU Ostrava , 2005.

[15] Wilkinson, J.H. *Algebraic Eigenvalue Problem*. Clarendon Press, Oxford , 1965.

[16] Schatzman, M. *A simple proof of convergence of the QR algorithm for normal matrices without shifts*. IMA Preprint Series 720 , 1990.

[17] Sojka, E. *Digitální zpracování a analýza obrazů*. Skriptum VŠB-TU Ostrava , 2000.

[18] Golub, G. H. – Van Loan, Ch. F. *Matrix computations*. The Johns Hopkins University Press , 1996.

[19] Laub, A. J. *Matrix Analysis for Scientists and Engineers*. SIAM, Philadelphia, PA, 2005.

[20] Chen, K. *Matrix Preconditioning Techniques and Applications*. Cambridge University Press, 2005.

[21] Vitásek, E. *Numerické metody*. SNTL Praha, 1987.

[22] Benzi, M. *Preconditioning Techniques for Large Linear Systems: A Survey*. Journal of Computational Physics 182, 418Û477, 2002.

[23] Horn, R. A. – Johnson, Ch. J. *Topics in matrix analysis*. Cambridge University Press, 1991.

[24] Quarteroni, A. – Sacco, R. – Saleri, F. *Numerical Mathematics*. Springer, 2000.

[25] Sewell, G. *The Numerical Solution of Ordinary and Partial Differential Equations, Second Edition*. John Wiley & Sons, 2005.

# Rejstřík